红茶三杯
Vinsoney

BGP 基础实验手册

By 红茶三杯

http://t.sina.com/vinsoney

访问 http://ccietea.com 获得文档的最新版本

文档更新时间：2011-12

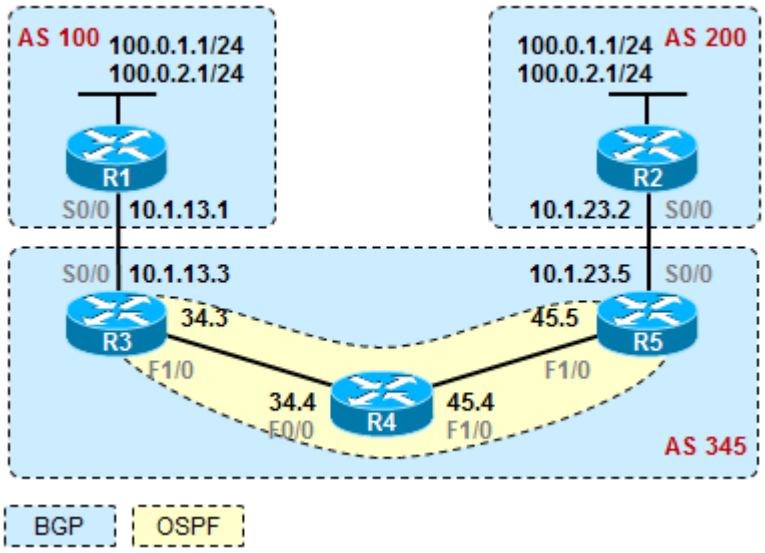
密级 ☑开放 ☐内部 ☐机密

类型 ☐讨论版 ☐测试版 ☑正式版

修订记录				
修订日期	**修订人**	**版本号**	**审核人**	**修订说明**
2011-12	**红茶三杯**			

目 录

1 实验拓扑及描述



实验描述

1. 网络拓扑及互联 IP 地址规划如图所示
2. R3、R4、R5 各自创建 LOOPBACK 接口，IP 地址为 **x.x.x.x**，x 为路由器的编号
3. R3、R4、R5 运行 OSPF，宣告三者互联接口及各自的 LOOPBACK
4. BGP 的 AS 规划如图所示
5. 完成基本的 IP、IGP 配置，建立 BGP 连接

实验需求

1. 完成基本的 BGP 配置，R1 与 R3、R2 与 R5 建立 EBGP 邻居关系；R3 与 R4、R4 与 R5 建立 IBGP 邻居关系（使用 LOOPBACK 接口作为更新源）。
2. 验证 BGP 选路规则，同时测试 BGP 相关策略工具。

2 BGP 选路规则

我们先回顾一下 BGP 的 13 条选路规则，在本实验手册中，将对选路规则中的主要条目做验证，同时也熟悉一下 BGP 的各种属性。

1. Weight 越大越优先
2. Local_Pref 越大越优先

3. 起源于本地的路由优先(如本地 network 的 ,或 aggregate 的) ,即下一跳是 0.0.0.0(在 BGP 表中,当前路由器通告的路由的下一跳为 0.0.0.0)
4. AS-Path 越短越优先
5. Origin 属性（优先顺序：IGP > EGP > Incomplete）
6. MED 越小越优先
7. 优选 EBGP 邻居发来的路由(相对于 IBGP 邻居) ,在联邦 EBGP 和 IBGP 中优选联盟 EBGP 路由
8. 优选到 BGP NEXT_HOP 最近的路由，该路由是去往下一跳路由器 IGP 度量值最小的路由
9. 如果有多条来自相同相邻 AS 的路由并通过 Maximum-paths 使多条路径可用，则将所有开销相同的路由加入 Loc-RIB
10. 如果路由都来自 EBGP 邻居，则优选最老的 EBGP 邻居传来的路由，降低滚翻的影响
11. BGP 邻居的 RID 越小越优先
12. 如果多条路径始发路由器 ID 或路由器 ID 相同，那么优选 Cluster-List 最短的路径
13. 选择邻居 ip 地址最小的路由（BGP 的 neighbor 配置中的那个邻居的地址，也就是邻居的更新源 IP）

好，那么下面我们开始实验：

3 完成基本配置

AS345 中的 R3、R4、R5 运行 OSPF。AS 内部使用 IGP 保证内部路由的互通，以满足内部的数据传输需求，同时也为 IBGP 连接提供底层路由的支持，在者作为传输 AS，”Transit AS”，运行 IGP 还能保证 BGP 路由在 AS 内的传递，而 BGP 路由的有效性(如 NEXT_HOP 属性的可达性)也需要 IGP 做一个基本的保证。所以，第一步我们先完成这个 IGP 协议的配置。

R1 的配置如下：

```
hostname R1
interface s0/0
  ip address 10.1.13.1 255.255.255.0
interface loopback1
  ip address 100.0.1.1 255.255.255.0
interface loopback2
  ip address 100.0.2.1 255.255.255.0
```

R2 的配置如下：

```
hostname R2
interface s0/0
  ip address 10.1.25.2 255.255.255.0
interface loopback1
  ip address 100.0.1.1 255.255.255.0
interface loopback2
  ip address 100.0.2.1 255.255.255.0
```

R3 的配置如下：

```
hostname R3
interface s0/0
  ip address 10.1.13.3 255.255.255.0
interface fa1/0
  ip address 10.1.34.3 255.255.255.0
interface loopback0
  ip address 3.3.3.3 255.255.255.0
router ospf 100
  network 10.1.34.0 0.0.0.255 area 0
  network 3.3.3.3 0.0.0.0 area 0
```

R4 的配置如下：

```
hostname R4
interface fa0/0
  ip address 10.1.34.4 255.255.255.0
interface fa1/0
  ip address 10.1.45.4 255.255.255.0
interface loopback0
  ip address 4.4.4.4 255.255.255.0
router ospf 100
  network 10.1.34.0 0.0.0.255 area 0
network 10.1.45.0 0.0.0.255 area 0
  network 4.4.4.4 0.0.0.0 area 0
```

R5 的配置如下：

```
hostname R5
interface s0/0
```

```
 ip address 10.1.25.5 255.255.255.0
interface fa1/0
 ip address 10.1.45.5 255.255.255.0
interface loopback0
 ip address 5.5.5.5 255.255.255.0
router ospf 100
 network 10.1.25.0 0.0.0.255 area 0
network 10.1.45.0 0.0.0.255 area 0
 network 5.5.5.5 0.0.0.0 area 0
```

如此一来，所有设备的直连网段也通了，AS345 内，路由也都通了，这是为 BGP 做准备。

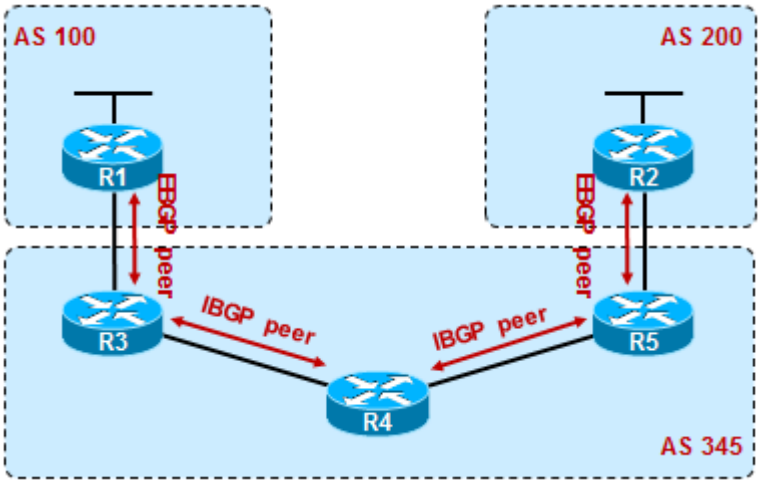
要注意我们并没有在 OSPF 中宣告 R3 的 s0/0 及 R5 的 S0/0 接口。

4 建立 BGP 邻居关系

这一步我们在上面的基础之上运行 BGP 协议，完成基本的 BGP 邻居关系的建立：

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3 及 R4、R4 及 R5 建立 IBGP 邻居关系，R3R4R5 使用 LOOPBACK 作为更新源并互指 neighbor。



R1 的配置如下：

```
router bgp 100
 no synchronization
 no auto-summary
 neighbor 10.1.13.3 remote-as 345
```

R2 的配置如下：

```
router bgp 200
  no synchronization
  no auto-summary
  neighbor 10.1.25.5 remote-as 345
```

R3 的配置如下：

```
router bgp 345
  no synchronization
  no auto-summary
  neighbor 10.1.13.1 remote-as 100          // EBGP 邻居
  neighbor 4.4.4.4 remote-as 345            // IBGP 邻居，使用 loopback0 口建邻居
  neighbor 4.4.4.4 update-source loopback 0 //  指定更新源为 loopback0
```

R4 的配置如下：

```
router bgp 345
  no synchronization
  no auto-summary
  neighbor 3.3.3.3 remote-as 345
  neighbor 3.3.3.3 update-source Loopback0
  neighbor 5.5.5.5 remote-as 345
  neighbor 5.5.5.5 update-source Loopback0
```

R5 的配置如下：

```
router bgp 345
  no synchronization
  no auto-summary
  neighbor 4.4.4.4 remote-as 345
  neighbor 4.4.4.4 update-source Loopback0
  neighbor 10.1.25.2 remote-as 200
```

注意事项：

- 一般情况下，我们会用直连接口的 IP 建立 EBGP 邻居关系。我们也通常使用 loopback 接口作为更新源、建立 IBGP 邻居关系，当我们使用 loopback 接口建立邻居关系时，务必确保本地路由器到达 IBGP 邻居的更新源接口（loopback 口）三层能通，也就是有 IBGP 邻居 LOOPBACK 接口的路由。

通过如上配置，BGP 邻居关系即可建立，我们来 show 一下：

R3#show ip bgp summary

```
BGP router identifier 3.3.3.3, local AS number 345
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
4.4.4.4         4   345      14      15        1    0    0 00:11:53        0
10.1.13.1       4   100       8       8        1    0    0 00:04:30        0
```

R3 有两个 BGP 邻居，一个是 10.1.13.1，是 EBP peer，另一个是 4.4.4.4 是 IBGP peer。

5 BGP 路由的引入及传递

R1 在 BGP 进程中宣告 LOOPBACK 网段

```
router bgp 100
  network 100.0.1.0 mask 255.255.255.0
  network 100.0.2.0 mask 255.255.255.0
```

IGP 协议，例如 OSPF，如果宣告某个直连接口（所关联的网段），则一方面该接口所对应的网络号将被引入 OSPF 路由选择进程，另一方面该接口将激活 OSPF，并开始发送 OSPF HELLO 包；而 BGP 却与此不同，BGP network 可以宣告本地接口，亦可宣告路由表中存在的路由，并且当 network 直连接口时，该接口并不会被“激活”并尝试建立邻居关系（BGP 的邻居关系需要手工 neighbor 去指的），BGP 的 network 命令，仅仅是将本地路由装载入 BGP 进程的一种方式。

在 R3 上查看 BGP 表，就能看到这两条路由

R3#show ip bgp

```
BGP table version is 3, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 100.0.1.0/24     10.1.13.1                0             0 100 i
*> 100.0.2.0/24     10.1.13.1                0             0 100 i
```

并且由于这两条学习自 R1 的 BGP 路由的 NEXT_HOP 属性为 10.1.13.1 是直连，是可达的，

因此他们都会被装进 IP 全局路由表，标记为 B。

同时从两条路由在 BGP 表中的标记：“ *> ”，说明这两条路由是 valid 可用的，且是 best 最优或者说，是优化的，因此他们会被传递给 R3 的 IBGP 邻居，也就是 R4

那么我们去 R4 上看看：

R4#sh ip bgp

```
BGP table version is 1, local router ID is 4.4.4.4
   Network          Next Hop            Metric LocPrf Weight Path
* i100.0.1.0/24    10.1.13.1                0    100      0    100 i
* i100.0.2.0/24    10.1.13.1                0    100      0    100 i
```

我们发现 R4 虽然通过 BGP 学习到了这两条前缀，也是 valid 的(打了*号)，但是却不优化(不是 best，没有>标记)，不优化，自然也就不会加载进路由表。那这是为什么呢？ 留意到两条路由的 next hop 是 10.1.13.1，而值得注意的是，10.1.13.0 的直连网段并没有被 R3 宣告进 OSPF，换而言之 R4 去往这两条路由的 next-hop 不可达，下一跳不可达，因此这两条 BGP 路由虽然在 BGP 表中，但是不 best，也无法装载进路由表。

解决办法有：1、在 R3 上将 10.1.13.0 的直连网段宣告进 OSPF，R5 同理；2、在 R3 上修改 next-hop，将 next-hop 修改为 R3 的更新源 IP 也就是 loopback 接口 IP

R3 增加配置如下：

```
router bgp 345
  neighbor 4.4.4.4 next-hop-self
```

如此一来，R3 更新给 R4 的 BGP 路由，next-hop 将会被替换成 R3 的 loopback 接口 IP，而 R3 的 loopback 接口 IP 已被宣告进 OSPF，R4 正好通过 OSPF 学习到 loopback 路由。因此，R4 的 BGP 表如下：

```
R4#sh ip b
BGP table version is 3, local router ID is 4.4.4.4
   Network          Next Hop            Metric LocPrf Weight Path
*>i100.0.1.0/24    3.3.3.3                  0    100      0 100 i
*>i100.0.2.0/24    3.3.3.3                  0    100      0 100 i
```

我们发现两条路由都 best 了，并且都被装载进了路由表中

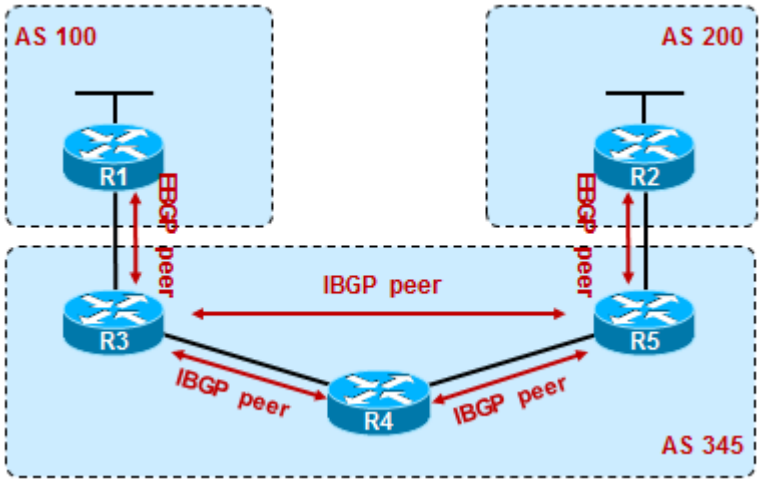
实验继续，经过上面的配置，R4 已经通过 BGP 学习到了源自 R1 的两条 BGP 路由。

但是我们却发现 R4 没有将这两条路由传递给 R5，这是因为 IBGP 的水平分割原则的效果。根据 IBGP 水平分割原则，一个 BGP 路由器，如果从它的 IBGP 邻居学习到 BGP 路由，那么它将不能把这些 BGP 路由再传递给其他 IBGP 邻居。设定这条规则的原因是，BGP 防环需要借助于

AS_PATH，而 AS_PATH 只有当路由出了本 AS 或被 BGP 路由器更新给 EBGP 邻居时才会改变，在 AS 内部是不会改变的，因此，在 AS 内部，防环需借助水平分割原则。那么如何解决 R5 学习不到路由的问题呢？

方法之一是：建立全互联 IBGP

在 R3 及 R5 之间增加一条 IBGP 连接，如此 R3 及 R5 的 BGP 路由传递就不需要借助 R4 了（虽然数据包仍需经 R4 转发，但对于 R4 而言，R3 传递给 R5 的这些数据包，只是普通的 IP 报文）。这里要留意，BGP 与 IGP 不同，IGP 如 OSPF，建立邻居关系需两台路由器直连，BGP 则无需直连，因为它是基于 TCP 建立的连接。



R3 上增加如下配置：

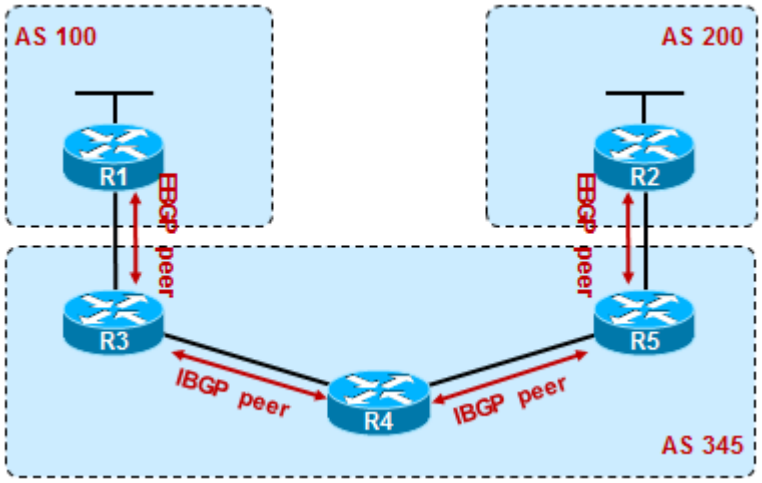
```
router bgp 345
  neighbor 5.5.5.5 remote-as 345
  neighbor 5.5.5.5 update-source Loopback0
  neighbor 5.5.5.5 next-hop-self
```

R5 上增加如下配置：

```
router bgp 345
  neighbor 3.3.3.3 remote-as 345
  neighbor 3.3.3.3 update-source Loopback0
  neighbor 3.3.3.3 next-hop-self
```

其他的解决办法，我们将在后面的实验中继续介绍

6 增加冗余路由



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系，R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

为了增加网络的可靠性及冗余性，我们让 R1 及 R2 都将 100.0.1.0 及 100.0.2.0 注入 BGP，这里是模拟冗余环境，也即我们即可通过 R1，亦可通过 R2 前往 100.0.1.0 及 2.0 网络。

R2 的配置如下：

```
router bgp 200
  no synchronization
  network 100.0.1.0 mask 255.255.255.0
  network 100.0.2.0 mask 255.255.255.0
  neighbor 10.1.25.5 remote-as 345
  no auto-summary
```

如此一来 R5 的 BGP 表就变成了如下：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 100.0.1.0/24	10.1.25.2	0		0	200 i
* i	3.3.3.3	0	100	0	100 i
*> 100.0.2.0/24	10.1.25.2	0		0	200 i
* i	3.3.3.3	0	100	0	100 i

也就是说 R5 去往 100.0.1.及 2.0 各存在两条路径，分别是通过 10.1.25.2 及 3.3.3.3，而最终 R5 选择 10.1.25.2 作为前往这两个网段的下一跳并将路由装入路由表。那么为什么 R5 会选择 R2 呢？

而不是选择 R3 呢？这是根据 BGP 的选路原则决定的（详请见红茶三杯朱 SIR 的 BGP 文档），在这里，最终影响选路的是这么一条规则“优选 EBGP 邻居发来的路由（相对于 IBGP 邻居学过来的）”，R2 是 R5 的 EBGP 邻居，所以优选了。

我们再来看看 R4

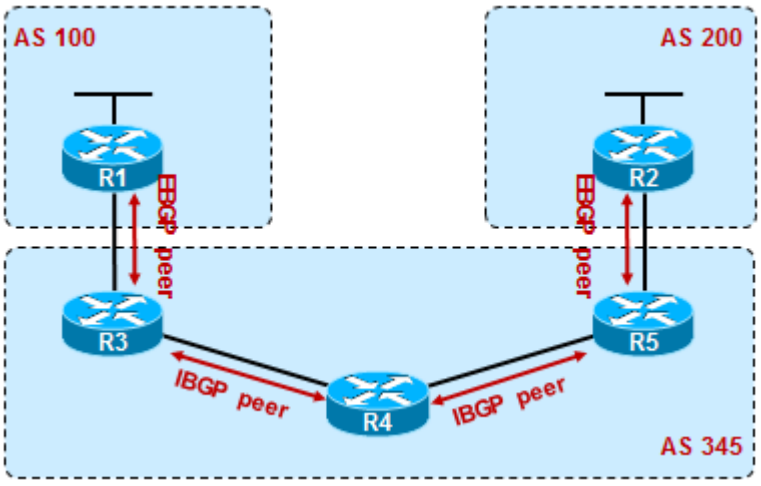
Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	5.5.5.5	0	100	0	200 i
*>i	3.3.3.3	0	100	0	100 i
* i100.0.2.0/24	5.5.5.5	0	100	0	200 i
*>i	3.3.3.3	0	100	0	100 i

发现 R4 上，前往 100.0.1.0 及 2.0 也存在冗余链路，分别通过 R3 和 R5 都能到达这两个网段，而最终 R4 选择 R3 作为前往这两个网段的下一跳，这是因为在 BGP 选路原则中，路由优选决策比较到了 BGP routerID 这一项，而 R3 的 RouterID 比 R5 要小，因此 R4 优选 R3。

到目前为止，我们已经完成了基本的配置，那么从 R4 上能 ping 同 100.0.1.1 及 2.1 么？试过之后我们一定得出否定的答案，这是因为在 R1 及 R2 上没有 AS345 内的路由，也就是回程路由出了问题，在这个拓扑中，我们可以在 R1 及 R2 上配置回程的路由或默认路由来完成实验的测试（注意，实际环境中，可能情况大不一样）。

实验完成到此处，我们设置 **MARK** 一下，接下去我们将分析各种属性对选路的影响。

7 通过修改 weight 来影响路由决策



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系，R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

WEIGHT 属性是 CISCO 私有属性。作用范围是本路由器(不传递),该值不既不会被包含在 update 消息中，也不会传递给任何 BGP 邻居。

范围 0-65535。

如果路由是从其他邻居学过来的，则默认值（在本地该路由）是 0

本地 network 产生的路由 weight 是 32768

本地重发布的直连接口路由、静态路由的 weight 为 32768

本地汇总产生的 BGP 路由 weight 值也为 32768

越大越优先

现在我们通过调整 weight 值，来影响 R4 上关于 100.0.1.0 及 2.0 路由的选择，使得 R4 优选 R5 作为去往 100.0.1.0 及 2.0 的下一跳。

在 R4 上

```
router bgp 345
  neighbor 5.5.5.5 weight 100
```

再观察一下 R4 的 BGP 表，我们发现：

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i100.0.1.0/24	5.5.5.5	0	100	100	200 i
* i	3.3.3.3	0	100	0	100 i
*>i100.0.2.0/24	5.5.5.5	0	100	100	200 i
* i	3.3.3.3	0	100	0	100 i

R4 选择了 R5 作为前往 100 网段的下一跳，这是因为这两条路由，从 R5 走，weight 权重值为 100，R3 的权重值为默认的 0，自然是 R5 要优选，因此 R4 将路径切换到 R5。

注意，这种修改方法，将对 R5 发来的所有路由生效（所有路由的权重都会变成 100），并且 weight 值的设置只能在本地做，且无法传递给任何 BGP 邻居。

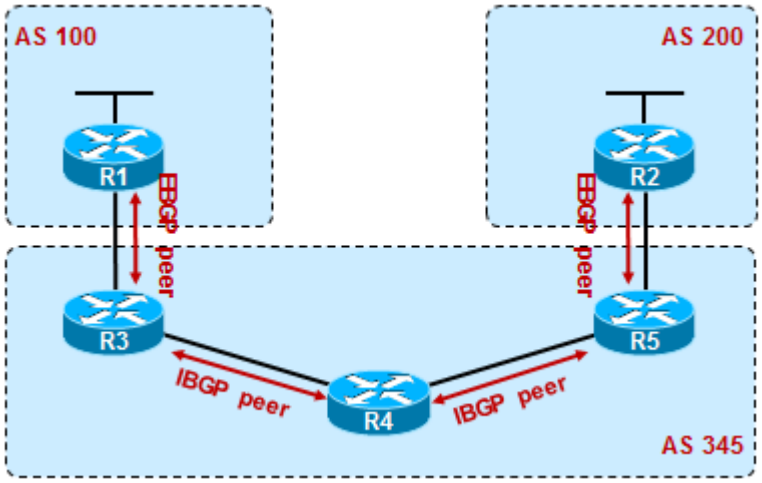
如果希望通过修改 weight，让 R4 去往 100.0.1.0 走 R3 去往 100.0.2.0 走 R5 呢？那么配置修改如下：

```
ip prefix-list 1 permit 100.0.1.0/24
```

```
ip prefix-list 2 permit 100.0.2.0/24
route-map WT2 permit 10
 match ip address prefix-list 1
 set weight 100
route-map WT2 permit 20
 match ip address prefix-list 2
 set weight 200
!
route-map WT1 permit 10
 match ip address prefix-list 1
 set weight 200
route-map WT1 permit 20
 match ip address prefix-list 2
 set weight 100

router bgp 345
neighbor 3.3.3.3 route-map WT1 in
neighbor 5.5.5.5 route-map WT2 in
```

配置完成后，使用 clear ip bgp * soft，软重置一下

8 通过修改 LOCAL_PREF 属性来影响流量



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系 ,R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

公认自决属性。LP 就是本地优先级，用于在内部对等体之间（IBGP）的 Update 消息，而不会传递给其他 EBGP 邻居，LP 值越大越优先。

1. 只能在 IBGP Peer 之间传递(除非做了策略否则 LP 值在 AS 内的 IBGP 邻居间传递不会丢失)，不能在 EBGP Peer 之间传递，如果在 EBGP Peer 之间收到的路由的路径属性中携带了 Local Preference ,则会触发 Notifacation 报文,造成会话中断 ;但是可以再 AS 边界路由器上使用 IN 方向的策略。
2. bgp default local-preference 500 //修改始发于本地的路由的默认 lp 值
3. BGP 路由器在向其 EBGP 邻居发送路由更新时，不能携带 LP 属性，对方收到该 EBGP 路由的 LP 值为空（连 LP 这个字段都木有），但是它会在本地为这条路由赋一个默认值，也就是 100，然后再传递给自己的 IBGP
4. 本地 network 及重发布的路由，LP 默认 100，并能在 AS 内向其他 IBGP 邻居传输，传输过程中除非部署策略，否则 LP 不变

我们可以通过设置 LP 值来影响路由，例如，同样实现让 R4 访问 100.0.1.0 走 R3，访问 100.0.2.0 走 R5。那么我们可以分别在 R3 及 R5 上对 R4 使用 OUT 方向的 route-map，并控制 LP 值

在 R3 使用策略

```
ip prefix-list 1 permit 100.0.1.0/24
ip prefix-list 2 permit 100.0.2.0/24
route-map LP permit 10
  match ip address pref 1
  set local-preference 200
route-map LP permit 20
  match ip address pref 2
  set local-preference 100
router bgp 345
  neighbor 4.4.4.4 route-map LP out
```

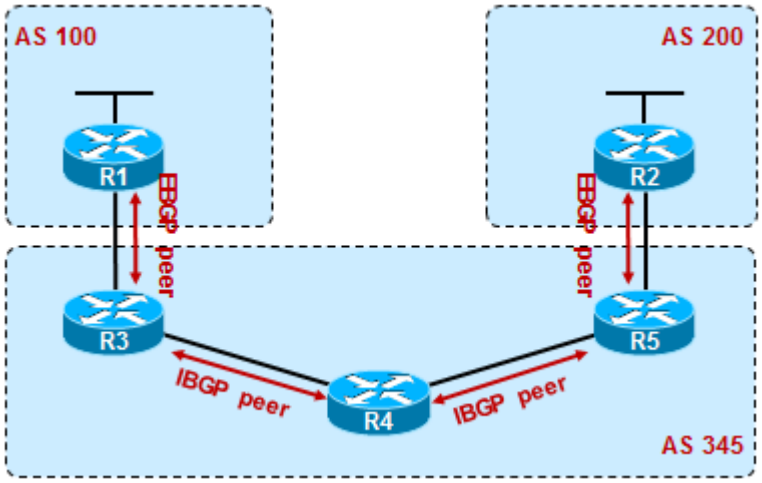
在 R5 上配置相似，只不过调整一下 LP 值

```
ip prefix-list 1 permit 100.0.1.0/24
ip prefix-list 2 permit 100.0.2.0/24
route-map LP permit 10
   match ip address pref 1
   set local-preference 100
route-map LP permit 20
   match ip address pref 2
   set local-preference 200
router bgp 345
   neighbor 4.4.4.4 route-map LP out
```

策略生效后查看 R4 的 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	5.5.5.5	0	100	0	200 i
***>i**	**3.3.3.3**	**0**	**200**	**0**	**100 i**
***>i100.0.2.0/24**	**5.5.5.5**	**0**	**200**	**0**	**200 i**
* i	3.3.3.3	0	100	0	100 i

9 通过 AS_PATH 影响路由选择



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系 ,R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

AS_PATH 是公认必遵属性 ,描述到达目标网络所要经过的 AS 号序列。最重要的作用是防环，如果 BGP 发言者发现自己的 AS 号位于接收自外部对等体的路由，则忽略该路由

仅当 update 消息被发送给其他的 AS 时，BGP 路由器才会将其 AS 号追加在 AS_PATH 中。这句话也隐含了另一个意思，那就是如果要修改 AS_PATH 属性，则必须在 AS 边界路由器上执行策略。

R4 能学习到源自 R1 及 R2 的 100.0 网段路由，并且根据此前的实验结果，我们知道，在不部署任何策略的情况下，R4 优选 RouterID 小的 R3 作为去往 100.0 网段的 BGPpeer，那么我们能否通过控制 AS_PATH 来影响路由的优选呢？下面，我们使用 route-map 在 R1 上做策略，对 R3 生效，修改 100.0.1.0 路由的 AS_PATH 路径属性。最终使得 R4 去往 1.0 网络走 R2，去往 2.0 网络仍走 R1。

R1 的配置如下：

```
ip prefix-list 1 permit 100.0.1.0/24
route-map test permit 10
 match ip address prefix-list 1
 set as-path prepend 100
route-map test permit 20

router bgp 100
 no synchronization
 network 100.0.1.0 mask 255.255.255.0
 network 100.0.2.0 mask 255.255.255.0
 neighbor 10.1.13.3 remote-as 345
 neighbor 10.1.13.3 route-map test out
 no auto-summary
```

set as-path prepend 100 这条命令，会在路由的 AS_PATH 路径属性前插入 100 这个 AS 号，注意，要谨慎使用这种控制 AS_PATH 的方法，因为有可能导致隐患，这个实验只是通过拉长

AS_PATH 的长度，以影响路径选择，在实际的环境中不是特别建议这么操作。

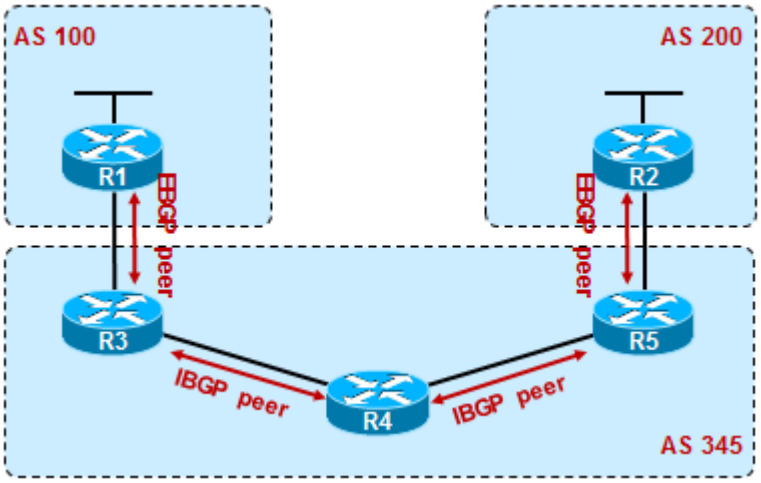
再来看一下 R4 的 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i100.0.1.0/24	5.5.5.5	0	100	0	200 i
*** i**	**3.3.3.3**	**0**	**100**	**0**	**100 100 i**
*>i100.0.2.0/24	3.3.3.3	0	100	0	100 i
* i	5.5.5.5	0	100	0	200 i

我们发现 R4 上，由 R3 传递过来的 100.0.1.0 这条路由，AS_PATH 为 100 100，而从 R5 过来的是 200，明显从 R5 来的路由 AS_PATH 要短，因此优选经 R5 到 R2 的路由。

这个策略，我们也可在 R3 上做，只不过这时策略的执行方向是 in。

10 通过 original 属性影响路由选择



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系，R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

知识回顾

对于 original 属性，它明确了路由更新的来源，有以下几种取值：

标记	缩写	描述
IGP	i	通过 BGP 手工 network，也就是起源于 IGP，因为 BGP network 必须保证该网

		络在路由表中已有。
EGP	e	是由 EGP 这种早期的协议重发布而来
Incomplete	?	从其他渠道学习到的，路由来源不完全（确认该路由来源的信息不完全）。 (重发布 IGP 或静态)

并且优选顺序是：i > e > ?

为了验证这个规则，我们先保持基本的 BGP 配置。在 R1 上使用 network 的方式引入 100.0 的两条路由，在 R2 上我们使用**重发布的方式**引入此二条路由：

R1 的配置如下：

```
router bgp 100
network 100.0.1.0 mask 255.255.255.0
 network 100.0.2.0 mask 255.255.255.0
```

R2 的配置如下

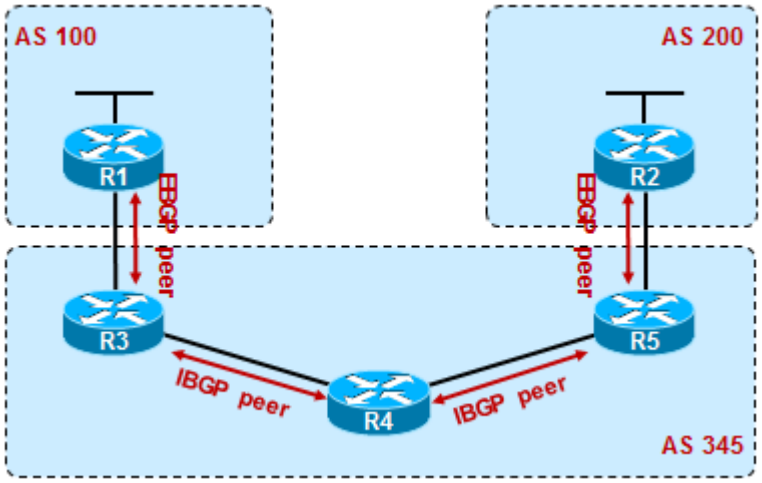
```
ip prefix-list 1 seq 5 permit 100.0.1.0/24
ip prefix-list 2 seq 5 permit 100.0.2.0/24
route-map test permit 10
 match ip address prefix-list 1 2
router bgp 200
redistribute connected route-map test
```

如此一来，在 R4 上，BGP 表如下：

```
R4#sh ip bgp
BGP table version is 4, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
                r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i100.0.1.0/24     5.5.5.5                  0    100      0 200 ?
*>i                 3.3.3.3                  0    100      0 100 i
*>i100.0.2.0/24     3.3.3.3                  0    100      0 100 i
* i                 5.5.5.5                  0    100      0 200 ?
```

我们发现，去往 100.0.1.0 及 2.0 的路由，下一跳均为 3.3.3.3，这是因为，这两条路径，original 值均为 “ i ”，要优于 “ ? ”

11 通过 MED 影响路由选择



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系，R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

CISCO 默认 MED 为 0，可选非传递

默认情况下，只比较来自同一邻居 AS 的 BGP 路由的 MED 值，就是说如果同一个目的地的两条路由来自不同的 AS，则不进行 MED 值的比较。MED 只是在直接相连的自治系统间影响业务量，而不会跨 AS 传递，MED 越小越优先。

我们仍然要实现让 R4 访问 100.0.1.0 网络走 R1，访问 100.0.2.0 网络走 R2，这一次我们通过调整 MED 实现，

R1 的配置如下：

```
ip prefix-list 1 seq 5 permit 100.0.1.0/24
ip prefix-list 2 seq 5 permit 100.0.2.0/24
route-map test permit 10
 match ip address prefix-list 1
```

```
 set metric 100
route-map test permit 20
 match ip address prefix-list 2
 set metric 200
router bgp 100
 network 100.0.1.0 mask 255.255.255.0
 network 100.0.2.0 mask 255.255.255.0
 neighbor 10.1.13.3 route-map test out
```

R2 的配置如下：

```
ip prefix-list 1 seq 5 permit 100.0.1.0/24
ip prefix-list 2 seq 5 permit 100.0.2.0/24
route-map test permit 10
 match ip address prefix-list 1
 set metric 200
route-map test permit 20
 match ip address prefix-list 2
 set metric 100
router bgp 100
 network 100.0.1.0 mask 255.255.255.0
 network 100.0.2.0 mask 255.255.255.0
 neighbor 10.1.25.5 route-map test out
```

上面的配置我们期望实现的效果是 R1 给 100.0.1.0 路由设置 MED=100 ,2.0 设置 MED=200，同时 R2 给 100.0.2.0 设置 MED=100，1.0 设置 MED 为 100，将这个属性分别对各自的 EBGP 邻居生效，于是乎给 MED 属性会随着路由在 AS345 内传递，最终抵达 R4，

那么按照我们的思路，在 R4 上应该能看到去往 100.0.1.0 选择的是 R3 也就是 3.3.3.3，去往 100.0.2.0 应该选择的是 R5，查看一下 R4 的 BGP 表：

```
R4#sh ip b
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	5.5.5.5	200	100	0	200 i
*>i	3.3.3.3	**100**	100	0	100 i
* i100.0.2.0/24	5.5.5.5	**100**	100	0	200 i

*>i	3.3.3.3	200	100	0	100 i

我们发现一个很奇怪的现象，R4 最终去往 100.0 的两个网络均选择了 3.3.3.3 作为下一跳，而不是按照 metric 这一属性来选路，为什么呢？原来 MED 属性，BGP 路由器默认只比较来源于同一宿主的路由的 MED 值，而 R1 及 R2 分属 AS100 及 AS200，那么对于 R4 来说,100.0.1.0 路由的两个路径来自不同的 AS，默认的它就绕过 MED 比较，而最终，影响选路的规则就是："BGP 邻居的 RID（越小越优先）"。

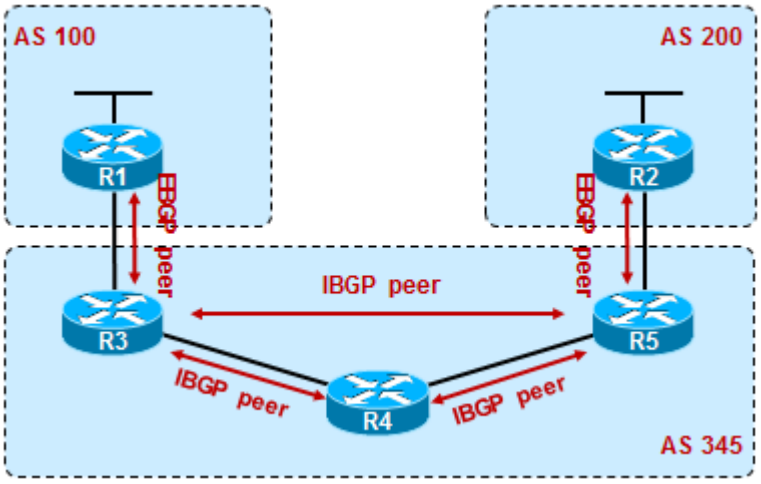
我们可以在 R4 上使用 bgp always-compare-med 命令，强制 R4 比较路由的 MED

最终：R4#sh ip b

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	5.5.5.5	200	100	0	200 i
*>i	3.3.3.3	100	100	0	100 i
*>i100.0.2.0/24	5.5.5.5	100	100	0	200 i
* i	3.3.3.3	200	100	0	100 i

选路就达到了我们的期望。

12 优选 EBGP 路由



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3、R4、R5 建立 IBGP 全互联，R3 R4 R5 使用 LOOPBACK 作为更新源并互指 neighbor。

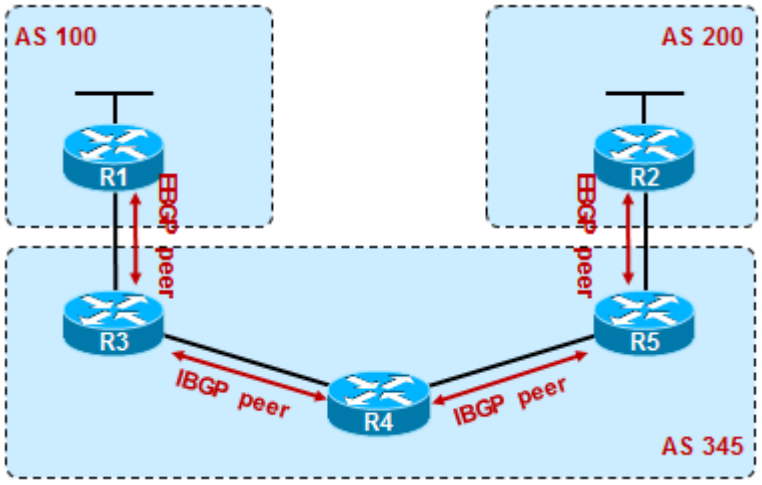
R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

优选 EBGP 邻居发来的路由（相对于 IBGP 邻居），在联邦 EBGP 和 IBGP 中优选联盟 EBGP 路由，注意，这条规则的生效，需保证 BGP 选路规则中的前面 6 条都无法做出决策的情况下才能达成。

完成上面描述的基本配置后，我们即可在 R5 上看到现象，R5 会同时从 R2 收到 100.0 网段的路由，也会从 R3 收到来自 R1 发布的 100 网段的路由，最终 R5 会优选 R2 的路由，这正匹配住了此条规则。

13 优选到 BGP NEXT_HOP 最近的路由



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3 及 R4、R4 及 R5 建立 IBGP 邻居关系，R3 R4 R5 使用 LOOPBACK 作为更新源并互指 neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

知识回顾

优选到 BGP NEXT_HOP 最近的路由，该路由是去往下一跳路由器 IGP 度量值最小的路由

在完成上述基本配置后，R4 的 BGP 表如下：

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	5.5.5.5	0	100	0	200 i
*>i	3.3.3.3	0	100	0	100 i
* i100.0.2.0/24	5.5.5.5	0	100	0	200 i

```
*>i                    3.3.3.3                  0    100      0    100 i
```

前面已经探讨过，我们在没有部署任何策略的情况下，该实验中 R4 之所以优选 3.3.3.3 也就是 R3 作为前往 100 网段的下一跳，原因是 R3 的 BGP RouterID 比 R5 的 RouterID 小，BGP 在进行决策规则比较时，一直比到了 RouterID 这一条，优选小的，因此选择了 R3。那么如何通过“NEXT_HOP”这一属性影响选路呢？

我们看看 100.0.1.0 这条路由：

R4#sh ip b 100.0.1.0

```
BGP routing table entry for 100.0.1.0/24, version 3
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Not advertised to any peer
  200
    5.5.5.5 (metric 2) from 5.5.5.5 (5.5.5.5)
      Origin IGP, metric 0, localpref 100, valid, internal
  100
    3.3.3.3 (metric 2) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

上述输出中，红色粗体的 metric 部分，就是本地到达 NEXT_HOP 属性地址的 IGP 度量值，我们发现从 R3 及 R5 走，metric 都是 2，那么如果我们将 R4 上连接 R3 的接口，也就是 F0/0 口的 OSPF cost 值调大，使得 R4 前往 3.3.3.3 路由的 OSPF metric 变大，会如何呢？

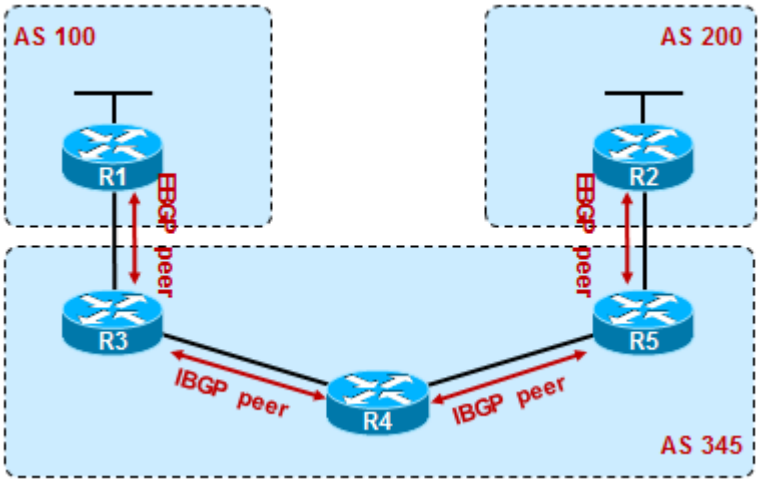
我们在 R4 的 F0/0 口，使用 ip ospf cost 100，将接口 cost 调大，于是结果如下：

R4#sh ip b 100.0.1.0

```
BGP routing table entry for 100.0.1.0/24, version 4
Paths: (2 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  200
    5.5.5.5 (metric 2) from 5.5.5.5 (5.5.5.5)
      Origin IGP, metric 0, localpref 100, valid, internal, best
  100
    3.3.3.3 (metric 101) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, internal
```

R4 上前往 3.3.3.3 的 OSPF metric 变成了 101，比前往 5.5.5.5 的 metric 要大，因此，R4 去往 100.0.1.0 的 BGP 路由，就优选了 R5。

14 使用路由反射器



我们将实验环境恢复为基本配置：BGP 的邻居关系如下

R1 及 R3、R2 及 R5 建立 EBGP 邻居关系；

R3及R4、R4及R5建立IBGP邻居关系，R3 R4 R5使用LOOPBACK作为更新源并互指neighbor。

R1 及 R2 都将 100.0.1.0 及 100.0.2.0 使用 network 的方式注入 BGP。

我们看一下 R5 的 BGP 表：

```
R5#sh ip bgp
   Network          Next Hop    Metric  LocPrf  Weight  Path
*> 100.0.1.0/24     10.1.25.2        0               0  200 i
*> 100.0.2.0/24     10.1.25.2        0               0  200 i
```

奇怪的现象：R5 的 BGP 表中，仅有 R2 通告的两条路由，而来自 R3 的 100.0 路由，却并未经由 R4 传递给 R5，换句话说，原本我们期望，AS345 去往 100.0 网络有冗余路径（通过 R1、R5 进行链路冗余），然而，此刻 R5 上却仅能通过 R2 去往目标网络。

造成这个现象的原因是？R4 并没将 R1 引入的、通过 R3 传递来的路由再传给 R5，

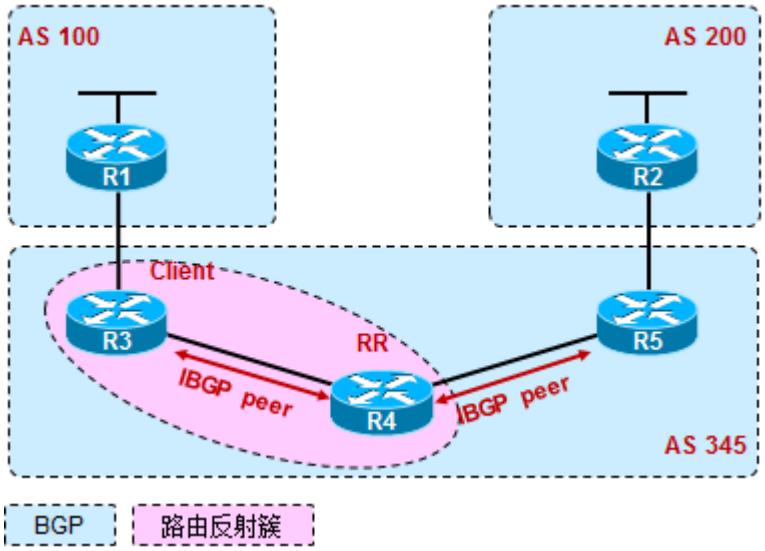
同时，R4 也收到了 R2 引入的 100.0 路由，但是并没有传递给 R3。这就是 IBGP 的水平分割原则：“BGP 路由器，不会将 IBGP 传递给他路由再传递给其他 IBGP 邻居”

那么解决的办法有哪些呢？

- 在传输 AS 内，建立 IBGP 全互联，也就是 R3 与 R4、R4 与 R5、R3 与 R5 之间都建立 IBGP 邻居关系

- 使用反射器
- BGP 联邦

建立 IBGP 全互联，配置比较简单，在上面的实验中已经有所涉及，我们来看看通过路由反射器如何实现。思路很简单，我们将 R4 配置为 RR，R3 为 R4 的 client，这样一来，R4 作为 RR，便会将学习自 Client R3 的路由，反射给 R5，也会将学习自 IBGP 邻居 R5 的路由，反射给 ClientR3。



R4 的 BGP 配置如下：

```
router bgp 345
 neighbor 3.3.3.3 remote-as 345
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 3.3.3.3 route-reflector-client
 neighbor 5.5.5.5 remote-as 345
 neighbor 5.5.5.5 update-source Loopback0
```

R4 将 R3 配置为 client，那么这就能突破水平分割的限制，将学习自 R3 的路由反射给 R5，将学习自 R5 的路由反射自 R3。再来看看 R5 的 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	3.3.3.3	0	100	0	100 i
*>	10.1.25.2	0		0	200 i
* i100.0.2.0/24	3.3.3.3	0	100	0	100 i
*>	10.1.25.2	0		0	200 i

R5 上便看到了去往 100.0 网络的冗余路径，下一跳是 3.3.3.3，最终 R5 优选 R2 作为去往 100.0 网络，这是符合拓扑期望的，但是深层原因，还是由于相比于从 R3 来的 IBGP 路由，学习自 R2

的 EBGP 路由要更优。

看过了 R5 再去看看 R3：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 100.0.1.0/24	10.1.13.1	0		0	100 i
*> 100.0.2.0/24	10.1.13.1	0		0	100 i

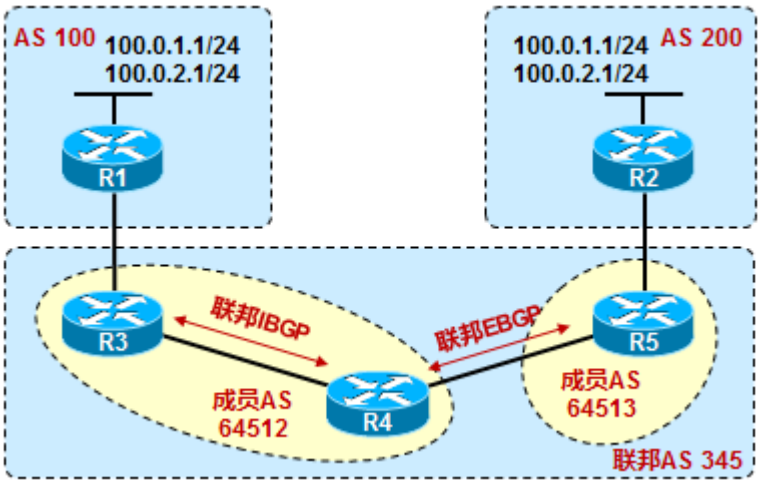
奇怪的现象出现了，似乎 R4 并没有把来自 R5 的路由反射给 R3，这是为什么？理论上说，路由反射器会将学习自非 client 的 IBGP 邻居发来的路由，反射给自己的 Client，但是显然，在这个实验中，R4 并没有将学习自 R5 的路由反射给自己的 Client。

我们还是上 R4 看看：

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i100.0.1.0/24	3.3.3.3	0	100	0	100 i
* i	5.5.5.5	0	100	0	200 i
*>i100.0.2.0/24	3.3.3.3	0	100	0	100 i
* i	5.5.5.5	0	100	0	200 i

发现什么了么？R4 的 BGP 表中，关于 100.0 网络都存在冗余路径，但是，最终经过 BGP 选路的决策，选了 R3 作为下一跳。而 BGP 仅将最优的路由（best，也就是>标记的）传递给 BGP 邻居，所以，R4 自然是不会把学习自 R3 的路由再发回给 R3 的。最后，虽然 R3 上 100.0 的两个网段都只存在一条路径，但是却不妨碍网络实现冗余性，当 R1 DOWN 掉后，R4 上便只有来自 R5 的路由，自然最后的也就是来自 R5 的路由，反射给 R3 也就没有问题了。

15 配置 BGP 联邦



联邦是解决 IBGP 全互联的另一种办法。我们可以通过在 AS 内（联邦 AS），划分一系列的小 AS，这些小 AS 被称作联邦成员 AS，在联邦成员内，BGP 路由器之间维护 IBGP 关系，遵循水平分割原则，在成员联邦 AS 之间，维护的是联邦的 EBGP 关系，不受 IBGP 水平分割的限制。而所有的联邦成员 AS，又都隶属于联邦 AS，对外统一使用联邦 AS 的标识。

在这个实验中，我们将 AS345 定义为联邦 AS，R3、R4 划分为成员 AS，使用 64512 作为 AS 号，R5 单独做为一个成员 AS，使用 64513 作为 AS 号。

R3 的配置如下：

```
router ospf 100
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.34.0 0.0.0.255 area 0
router bgp 64512
 no synchronization
 no auto-summary
 bgp confederation identifier 345
 neighbor 4.4.4.4 remote-as 64512
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 10.1.13.1 remote-as 100
```

注意，R3 创建的 BGP，使用的 AS 号是 64512，这是联邦成员 AS 号而不是联邦 AS 号 345；R3 有一个联邦 IBGP 邻居 R4。

bgp confederation identifier 345　　这条命令，用来告诉联邦外的 AS，我本地的 AS 号为 345

R4 的配置如下：

```
router bgp 64512
 no synchronization
 bgp confederation identifier 345
 bgp confederation peers 64513
 neighbor 3.3.3.3 remote-as 64512
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 5.5.5.5 remote-as 64513
 neighbor 5.5.5.5 ebgp-multihop 3
 neighbor 5.5.5.5 update-source Loopback0
 no auto-summary
```

R4 有两个 BGP 邻居，其中 R3 为其联邦 IBGP 邻居，R5 为其联邦 EBGP 邻居

为了让 R4 知道，R5 为其联邦的 EBGP 邻居，而不是普通的 EBGP 邻居，需要增加如下配置：

bgp confederation peers 64513

同时要注意，由于 R4 与 R5 为联邦的 EBGP 邻居关系，因此同样存在 TTL 为 1 的问题，如果二者使用 loopback 接口建立 BGP 关系，那么还需使用到 neighbor 5.5.5.5 ebgp-multihop 3 命令。

R5 的配置如下：

```
router bgp 64513
 no synchronization
 bgp confederation identifier 345
 bgp confederation peers 64512
 neighbor 4.4.4.4 remote-as 64512
 neighbor 4.4.4.4 ebgp-multihop 4
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 10.1.25.2 remote-as 200
 no auto-summary
```

如此一来路由传递就没有问题了。

在 R1 及 R2 上 network 100.0 的两个网络进 BGP，那么在 R4 上的 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
* i100.0.1.0/24	10.1.13.1	0	100	0	100 i
*	10.1.25.2	0	100	0	(64513) 200 i
* i100.0.2.0/24	10.1.13.1	0	100	0	100 i
*	10.1.25.2	0	100	0	(64513) 200 i

我们看到 R4 上，关于 100.0 的两个网络存在冗余路径，但是却都不是 best，自然 R4 也无法使用这些路由，为什么不是 best 呢？原因在于下一跳，nexthop 不可达。

注意在联邦外引入的路由，next-hop 属性在联邦内部传递是不会发生改变的，即使在不同的成员 AS 间传递也是如此，因此需让 R3 及 R5 对 R4 修改 next-hop 属性，设定为自己的更新源 IP，这个你已经知道怎么做了吧？

BGP 技术笔记

红茶三杯 CCIE 学习文档

文档版本： 2.0　　更新时间： 2013-01-26

文档作者： 红茶三杯　　文档地址： http://ccietea.com

1 基本概念

1.1 知识点

1. 每个 AS 都有一个标识号，范围是 1~65535，其中 64512~65535 是保留私用的。
2. BGP 的更新由 TCP 协议承载，使用的端口号是 179 因为 BGP 要求使用 TCP，所以 BGP 对等体之间必须有 IP 层连通性。
3. BGP neighbor：当两台 router 相互之间建立了一条基于 TCP 的 BGP 连接之后，就称他们为邻居或对等体。在邻居刚建立起连接时，它们交换所有的候选 BGP 路由，但在该初始路由交换之后，通常只在网络信息发生变化时才发送增量路由更新，而不会周期性更新。
4. BGP 是设计为在 AS 之间传递路由，因此，它的一跳实际上是一个 AS。
5. BGP 是无类路由选择协议、距离矢量路由协议，自动汇总默认关闭（这点要看 IOS）。
6. BGP 有三个管理距离，从 IBGP 学过来的 200，从 EBGP 学过来的 20，这是因为 BGP 的设计理念的工作于 AS 之间，而 AS 内部，BGP 希望 IGP 协议自己能搞定，因此 IBGP 路由的管理距离设置为一个大 AD 值 200，EBGP 路由设置为一个小 AD 值 20。
7. BGP 选举 routeriID 法则和 OSPF 一样。

1.2 Tables

1. **Neighbor table**

   Router# sh ip bgp summary

```
BGP router identifier 10.1.13.3, local AS number 345
BGP table version is 1, main routing table version 1
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.1.13.1       4   100       3       3        1    0    0 00:00:54        0
```

关于 BGP 邻居表字段的详细含义，请见本文档 BGP 配置章节的 show 小节

2. **BGP table**

   存放所有路由条目，以及其属性。

   关于 BGP 表相关字段的详细含义，请见本文档 BGP 配置章节的 show 小节

3. **Routing table**

   IP 路由表

1.3 TIMER

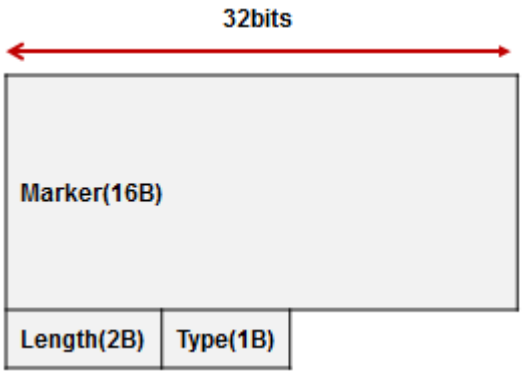
- **KEEPALIVE** 默认 60s ,keepalive 计时器不会在 Open 消息中交互 ,那么两个 BGP 邻居间该计时器如何定？
  如果当前手工配置的 keepalive timer 小于 min（holdtime）/3，则取配置值
  如果当前手工配置的 keepalive timer 大于 min（holdtime）/3，则取 int[min（holdtime）/3]
  其中 min（holdtime）为两台 BGP 邻居间 holdtime 的最小值
- **HOLDTIME** CISCO 默认 180s（3 倍 KEEPLIVE timer） ，该计时器包含在 open 报文中
  必须收到一个 KEEPLIVE 或更新消息前所允许经过的最大时间。如果两端 Holdtime 不一致 ,双方接受较小的的时间。

timer bgp 0 0　　邻居永远不 down

BGP 不会周期性更新路由，仅在需要的时候更新，由于公网的路由可能的动荡的，因此触发更新也会有一定的等待时间，IBGP peer 为 5S；EBGP peer 为 30S，而在这段时间内，BGP 仍可以进行路由信息的搜集，所以 BGP 收敛会比较慢。

1.4 消息类型

每种消息都包含 BGP 消息报头，BGP 报文的头部如下：



Marker： 用于检测 BGP 对等体之间同步丢失情况，并且在支持验证功能的情况下进行消息验证
如果消息类型为 open 或 open 消息中没有包含验证消息，标志字段被置为全 1，
否则标志字段通过某些计算得到（作为验证进程的一部分）

Length： 表示 BGP 报文的全部长度，包括头部

TYPE： 1-open； 2-update； 3-Notification； 4-keepalive

以下是 BGP 的五种报文及报文解析：

1. **OPEN 消息**

TCP 会话建立起来自以后，两个邻居都要发送一个 OPEN 消息，双方使用 OPEN 消息标识自己，并且规定自己的 BGP 运行参数。如果 open 消息被接受 ,则回送一条 keepalive 消息进行确认 ,确认后就能发送 update 消息了。OPEN 消息包含以下内容：

- Version 8bit ，V4 目前使用较多的版本
- AS 号 16bit 本地 AS 号
- HOLDTime 路由器必须收到一个 keeplive 或者更新消息之前所允许经过的最大秒数
- BGP Identifer ROUTER-ID 和 OSPF 选取 routerID 的方式一致
- 可选参数长度 用来表示后面可选参数字段的长度
- 可选参数 包含了一个可选参数列表，每个参数都由一个长为 1 个 8 位组 的类型字段、一个长 8 位组的长度字段及一个可变长的包含参数数值的字段组成。用来宣告支持验证、多协议支持和路由刷新等可选功能（常被称为“能力值”，意思就是这家伙具备什么能力）

2. **KEEPALIVE 消息**

如果路由器接受了邻居在 OPEN 消息中的参数 ,就会应答一个 keeplive 消息 ,并且在此后 1/3 的 holdtime(但不小于 1S) 为周期发送该消息，CISCO 默认 60S。如果协商后保持时间为 0，则不发送 keepalive 保活消息。

KEEPALIVE 消息实际上弥补了 TCP 无法确认对端存活情况的缺陷。

KEEPALIVE 消息仅包含 19bytes 的 BGP 头部，除此之外不包含任何其他数据。

3. **UPDATE 消息**

用来公布可用的路由、撤销的路由或者两者兼顾，

每条 update 消息只描述单条 BGP 路由，这是因为 BGP 路径属性只能描述单条路由

消息中包含：

- 网络层可达信息（NLRI） 一个或多个（长度、前缀）二元组，用来公布 IP 地址前缀和前缀长度
- 路径属性
- 被撤销路由

4. **Notification 消息**

   当检测到差错的时候发送，通常会导致 BGP 连接的终止

5. **Route-refresh**

   当路由策略发生变化时，去请求邻居重新通告路由（BGP 不会周期性发送更新）

1.5 BGP 状态机

更加详细的 BGP 状态机及邻居关系建立过程，请见笔记文档：《BGP 状态机及邻居关系建立过程》

Peer 状态名称	发什么包	在做什么
Idle	尝试建立 TCP 连接	本地寻找一条到邻居的路由，并开始准备 TCP 的连接及监视远程 peer 启动 TCP 连接。启用 BGP 时，要准备足够的资源
Connect	发 TCP 包	本地找到一条到邻居的路由，并尝试 TCP 三次握手，等待完成中，认证都是在 TCP 建立期间完成的。如果 TCP 连接不上则进入 Active 状态，反复尝试连接。 如果 TCP 建立成功，BGP 进程会向邻居发送 Open 消息并进入 Opensent 状态
Active	发 TCP 包	TCP 连接没建立成功，反复尝试 TCP 连接。
OpenSent	发 Open 包	TCP 连接建立已经成功，开始发送 Open 包，Open 包携带参数协商对等体的建立。 如果接收到 open 消息后，存在差错，则发送 Notification 消息。如果没有差错，则进入 OpenConfirm 状态
OpenConfirm	发 Keepalive 包	参数、能力特性协商成功，自己开始发送 Keepalive 包，等待对方的 Keepalive 包。 如果收到对方的 keepalive 消息则迁移到 Established 状态
Established	发 Update 包	已经收到对方的 Keepalive 包，双方能力特性一致，开始使用 Update 通告路由信息。

1.6 BGP neighbor

必须在 BGP 进程中使用 neighbor 来指定 BGP 对等体。他们交换的是路由信息，以及相关属性，而不是链路状态。

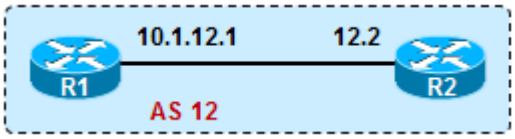
1. **EBGP 邻居**

   一般是直连，因为它会去查找直连路由。EBGP 默认 TTL 为 1

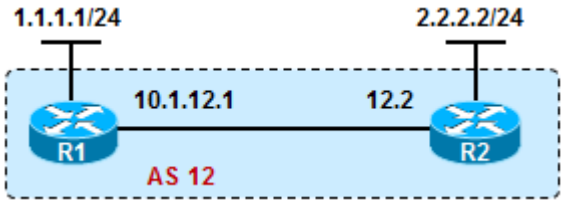
2. IBGP 邻居

无需直连。只要求有 TCP 联通性。

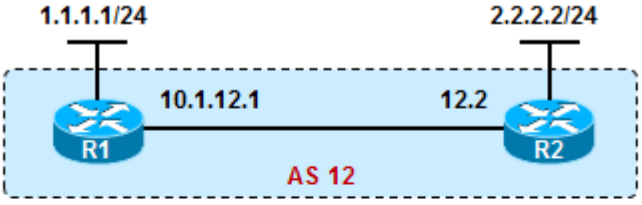
1.7 update-source



BGP 无法像 IGP 那样自动发现邻居，而需手工指定，邻居的 IP 由本地的 BGP neighbor 命令指定，而该 BGP 连接的源 IP（更新源）默认情况下为流量的出接口 IP。注意只有当本地配置的邻居 IP 与邻居用于 BGP 连接建立的源 IP 相同时，BGP 连接才能被正常建立，同时，仅需保证一方满足条件即可。



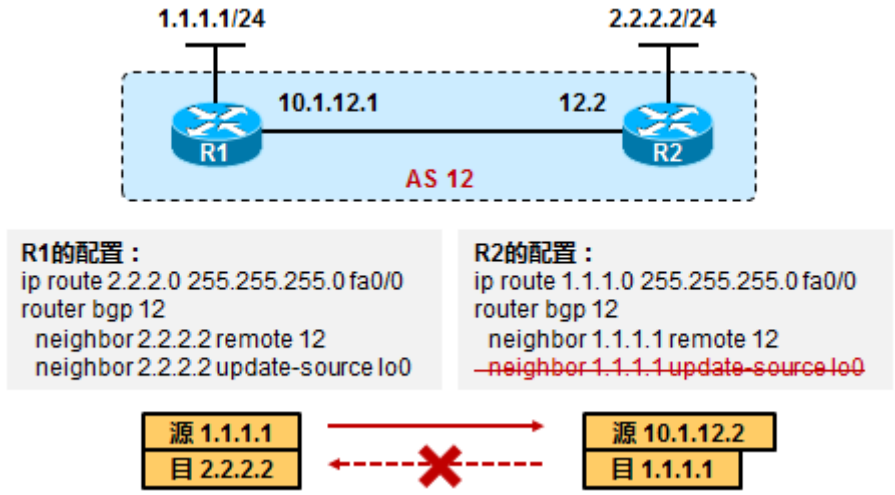
IBGP 邻居之间建立邻居，为了保证邻居关系的稳定，一般使用 loopback 接口建立，这是因为如果使用物理接口，那么物理接口故障，邻居关系就 DOWN 了，并且在 AS 内部，路径可能是冗余的，邻居之间的 LOOPBACK 路由可通过 IGP 获取并提供一定的路由冗余性（当物理线路也存在冗余的情况下）。在使用 loopback 接口建立 BGP 邻居关系时，务必注意还需要指定更新源 IP。如下图，是一个规范配置：



R1的配置：
```
ip route 2.2.2.0 255.255.255.0 fa0/0
router bgp 12
 neighbor 2.2.2.2 remote 12
 neighbor 2.2.2.2 update-source lo0
```

R2的配置：
```
ip route 1.1.1.0 255.255.255.0 fa0/0
router bgp 12
 neighbor 1.1.1.1 remote 12
 neighbor 1.1.1.1 update-source lo0
```

其实，BGP 邻居关系的建立，仅需一条连接即可，如下图的配置也是能建立起 BGP 连接的：



R1 使用 1.1.1.1 作为更新源，试图与 2.2.2.2 建立 BGP 连接，而 R2 本地配置的 neighbor IP 为 1.1.1.1，与 R1 的更新源 IP 匹配，尽管 R2 配置的更新源为默认（数据包出接口 IP 10.1.12.2），但是两者之间的 BGP 连接建立是没有问题的，因为 R1 向 R2 的连接是没问题的。同时这里还可以顺便钻研下路由的问题，例如，如果将 R2 本地到 1.1.1.0 的静态路由，改配为默认路由，我们会发现 BGP 连接依然能建立，因为此例中 R2 为 BGP 连接的被动者，R1 有明细路由即可主动发起连接。但是如果将 R1 上到 2.2.2.0 的明细静态路由，改为默认，那就不行了，R1 会认为 “no route to peer”。

- 如果 R1 R2 之间是建立 EBGP 邻居关系 ,因为 EBGP 邻居关系的建立会检查直连路由 ,并且默认 TTL=1，而这个时候实际上 R1 的 loopback 口到 R2 是需要至少 2 跳， 那么这个时候 还需要两者配置 neighbor xxxx ebgp-mul 2
- 经试验验证 ,(在 BGP 的基本配置无误的情况下) R1 和 R2 到对方的 looback 口的网段用静态路由或动态路由互相学习，BGP 邻居关系建立都没问题，但是如果两端都用默认路由互指，则 BGP 邻居关系无法建立，这是为了防环的目的， 一边用静态，一边用默认是可以的。

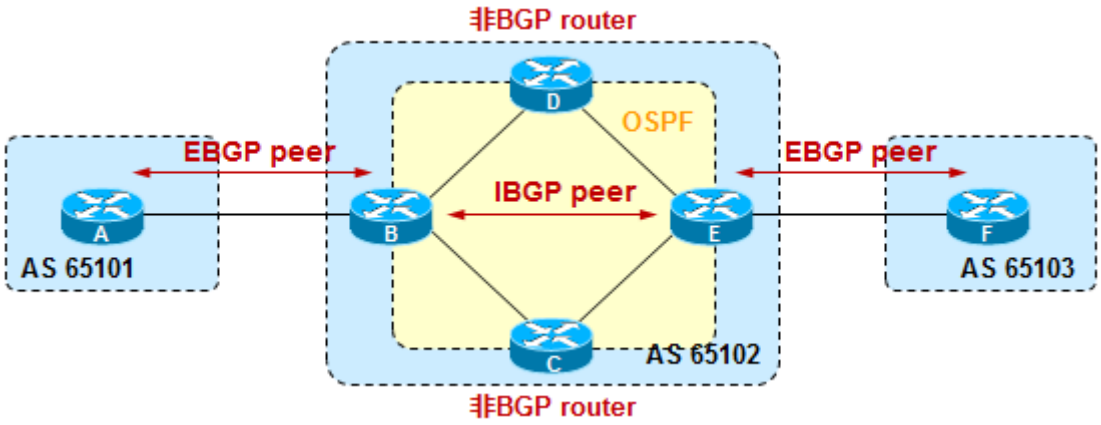
1.8 IBGP 水平分割

BGP 防环是通过 AS_PATH 实现的，而 AS_PATH 仅仅在路由离开 AS 才回被更改，因此在 AS 内，IBGP 就没有 EBGP 的防环能力，为了防止环路的出现，BGP 路由器不会将从 IBGP 邻居学习过来的路由再通告给自己其他 IBGP 邻居。

BGP 规定不将通过一个 IBGP 获悉的路由传播给其他所有 IBGP 邻居。 这个是 BGP 的水平分割规则。

由于水平分割原则存在，BGP 要求 AS 内，须保证 IBGP 全互联（这里是指 neighbor 命令指定）。（根本原因是在 AS 内部， AS-PATH 不会改变，无法使用 AS_PATH 防环，因此很容易出现环路）

1.9 IBGP 与 IGP 同步



AS65102 作为中转区域（trasit AS），在 AS 内部 BCDE 运行 OSPF，使得 AS 内路由可达；D 和 C 都没运行 BGP 协议，BE 之间建立 IBGP 邻居关系。若 A 上有个网段 1.1.1.0，A 将其注入 BGP，并且传递给 EBGP 邻居 B，B 又会传递给 IBGP 邻居 E（这个传递过程其实是将 BGP 报文放置于 IP 包内经过 C 或 D 最终传递给 E，对于 CD 而言，这些数据包都是普通 IP 包，直接转发不查看）。最终 E 成功的将 1.1.1.0 的路由传递到了 F。那么 F 即使能学习到 BGP 过来的这条路由，当有数据前往 1.1.1.0 网络时，将数据包丢给下一跳 E，而 E 上关于 1.1.1.0 的下一跳是 B，非直连，因此需递归得到其前往 B 的下一跳，到 B 的下一跳是 D 或 C，于是将数据包丢给 D 或 C，而 C 和 D 是并不知道 1.1.1.0 的（他们只运行了 OSPF，没有运行 BGP），至此成路由黑洞。注意虽然 CD 没运行 BGP，但是 BE 之间的 BGP 路由（BGP 报文）可以通过 CD 进行转发，并且对于 CD 来说，这些 BGP 的消息他们自己视为普通的 IP 包，直接转发，并不查看。

这就是路由黑洞问题，为了避免这个问题，可以考虑在 BE 上，将 BGP 路由重发布进 OSPF 来解决，但这么做后果是不可预估的，毕竟 BGP 承载的路由条目是相当巨大的。另一个解决方案是，要求 AS 内路由器都运行 IBGP 并实现 IBGP 全互联，那么该 Transit AS 内的路由器就都能知晓 BGP 路由，如此即可解决路由黑洞的问题。

【同步的概念】BGP 路由器从 IBGP 邻居学到一条路由后，是不启用的（不优化的），除非它再次从 IGP 学习到相同的路由，才会启用。这就是为了防止上面所描述的路由黑洞的问题。

而如果这个网络实现了 IBGP 全互联（彼此之间建立了 IBGP 邻居关系），那么同步就没有意义了，便可关闭同步。因此现今的 CISCO IOS 默认关闭同步规则。

综上所述，要使 IBGP 能够正常工作，就必须实施以下配置选项之一：

1. 将外部路由重发布进 IGP 中，以确保 IGP 与 BGP 同步。但该方法的缺陷在于如果从 BGP 获取大量的路由，对于 IGP 来说是个相当大的负担。
2. 建立 IBGP 全互联，且关闭同步机制。目前基本上都是使用该方法。但是这个方法有个缺陷，即如果 IBGP

邻居太多，管理这些 IBGP 的邻接关系将会是个挑战，并且对设备的负担也较大。好在我们有两种措施可以辅助解决，1 是路由反射器 2 是联邦。

1.10BGP 路由通告

当存在多条路径时，BGP Router 只选取最优的路由（BEST）来使用（没有负载均衡的情况下）

BGP 只把自己使用的路由，也就是自己认为 Best 的路由传递给 BGP peer

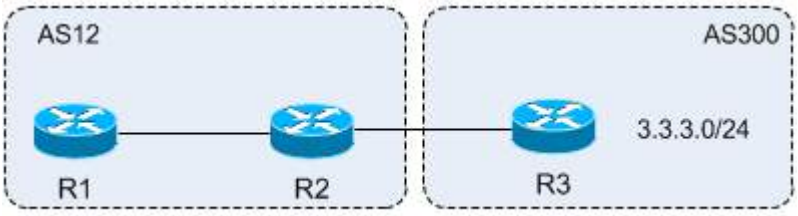
BGP Speaker 从 EBGP 获得的路由会向它所有 BGP 相邻体通告（包括 EBGP 和 IBGP）

BGP Speaker 从 IBGP 获得的路由不向它的 IBGP 相邻体通告（避免环路，水平分割；存在路由 RR 的情况除外）

BGP Speaker 从 IBGP 获得的路由是否通告给它的 EBGPpeer 要视 IGP 和 BGP 同步的情况来决定

2 Advance

2.1 关于路由的递归



R1、R2 运行 OSPF，宣告直连网络及各自的 LOOPBACK 接口，R1 LO1 为 1.1.1.1，R2 LO1 为 2.2.2.2

R1、R2 运行 IBGP，使用 LOOPBACK 为更新源，并互相指 neighbor，R2、R3 之间为 EBGP 关系，用直连接口建邻居。R3 宣告 3.3.3.0/24 进 BGP，在 R2 上对 R1 配置 next-hop-self

于是在 R1 上能学习到 3.3.3.0 的路由，R1 的路由表如下：

```
     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/65] via 10.1.12.2, 00:05:16, Serial0/0
     3.0.0.0/24 is subnetted, 1 subnets
B       3.3.3.0 [200/0] via 2.2.2.2, 00:03:01
     10.0.0.0/24 is subnetted, 1 subnets
C       10.1.12.0 is directly connected, Serial0/0
```

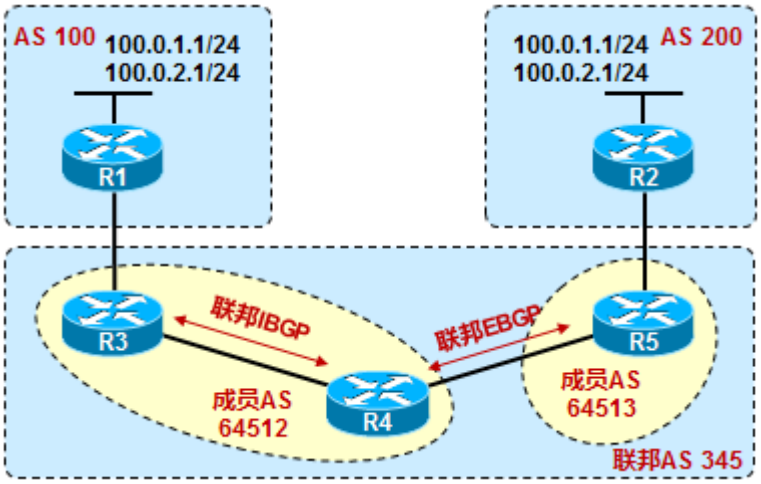
这就是 BGP 典型的递归路由，R1 上 3.3.3.0 路由的下一跳为 2.2.2.2，进一步递归 2.2.2.2，关联出接口 S0/0

2.2 联邦

联邦的相关特性：

- 在联邦内部保留联邦外部的 NEXT_HOP 属性
- 公布给联邦的路由的 MED 属性在整个联邦范围内予以保留
- 路由的 LP 属性在整个联邦范围内予以保留
- 在联邦范围内，将成员 AS 号压入 AS_PATH，但不公布到联邦外，并且使用 TYPE3、4 的 AS_PATH
- AS_PATH 中的联邦 AS 号用于在联邦内部避免环路

1. 联邦的配置及实现



为了解决路由传递的问题，除了路由反射器外，还有一个不错的解决方案，就是联邦。

通过将 AS345 定义为联邦 AS（大 AS），同时在联邦 AS 中创建成员 AS（小 AS），则可解决 IBGP 路由传递的问题。

那么 R3、R4 之间就是联邦的 IBGP 关系，R4 与 R5 之间是联邦的 EBGP 关系，

在联邦内部，R3 与 R4 都是属于 AS64512，对于 R4（AS64512），R5 属于 AS64513，但是对于联邦外部而言，R3R4R4 都是 AS345，外部压根不知道有 AS64512 和 64513 存在。

R3、R4、R5 用 OSPF 保证 AS 内路由互通，同时使用 loopback 建立 BGP 邻居

R3 上的配置如下

```
router bgp 64512                          //使用联邦成员 AS 号建立 BGP
 bgp confederation identifier 345         //这条命令用来对联邦外的 AS 通告自己的真实 AS 号
 neighbor 4.4.4.4 remote-as 64512
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 10.1.13.1 remote-as 100
```

bgp confederation iden 345 配置后，对于联邦 AS 外来说，这个 AS 不是什么 64512 了，而是 345

R4 的配置：

```
router bgp 64512
 bgp confederation identifier 345
 bgp confederation peers 64513
 neighbor 3.3.3.3 remote-as 64512
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 5.5.5.5 remote-as 64513
 neighbor 5.5.5.5 ebgp-multihop 3
 neighbor 5.5.5.5 update-source Loopback0
```

由于 R4 与 R5 是联邦的 EBGP，同样有 TTL 的问题，因此他们使用 LOOPBACK 建立邻居关系的话，要注意设置 ebgp-multihop。

另外，对于 R4 而言，R5 此刻是一个普通的 EBGP 邻居，并且是另一个 AS64513，而且跟我一点关系没有，联邦的建立就会有问题，因此，还需在 R4 上增加 **bgp confederation peers 64513 命令，**那么 R4 将对 AS64513 视为它的联邦 EBGP peer，而对除了 AS64513 外的 AS 视为普通的 AS。如果联邦内有成员 AS，那么若本地需指多个 confederation peers，则可 **bgp confederation peers xx yy zz**，写多个 AS 号。

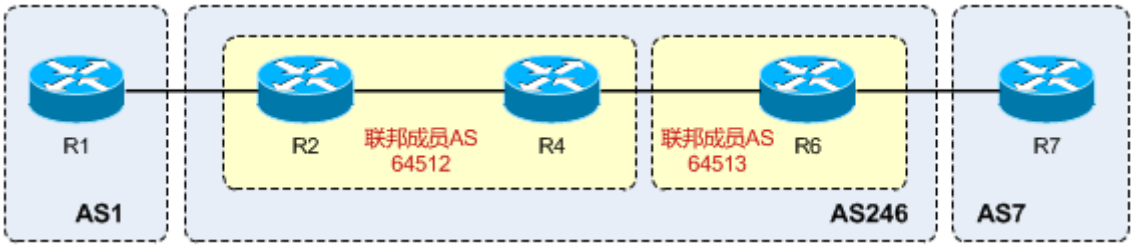
R5 的配置如下：

```
router bgp 64513
 bgp confederation identifier 345
 bgp confederation peers 64512
 neighbor 4.4.4.4 remote-as 64512
 neighbor 4.4.4.4 ebgp-multihop 4
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 10.1.25.2 remote-as 200
```

2. **AS_CONFED_SEQUENCE 及 AS_CONFED_SET**

这两个属性用于在联邦内、成员 AS 间防环

- **AS_CONFED_SEQUENCE** 一个去往特定目的地所经路径上的有序 AS 号列表，其用法与 AS_SEQUENCE 完全一样，区别在于该列表中的 AS 号属于本地联邦中的 AS
- **AS_CONFED_SET** 一个去往特定目的地所经路径上的无序 AS 号列表 ,去用方法与 AS_SET 完全一样，区别在于列表中的 AS 号属于本地联邦中的 AS



R1 上引入路由 11.11.11.0

R2 上 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 11.11.11.0/24	192.168.12.1		0	0	1 i

R4 上 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i11.11.11.0/24	2.2.2.2	0	100	0	1 i

R6 上 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 11.11.11.0/24	2.2.2.2	0	100	0	**(64512) 1 i**

R7 上 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 11.11.11.0/24	192.168.67.6			0	**246 1 i**

在 R4 上连接 R2 的接口抓包，发现更新包中 11.11.11.0 前缀的属性中携带的是 AS_PATH，内容为 AS 1

在 R6 上连接 R4 的接口抓包，发现更新包中 11.11.11.0 前缀的属性中携带的 AS_PATH 中包含 2 个内容：

```
Path attributes
  ORIGIN: IGP (4 bytes)
    Flags: 0x40 (Well-known, Transitive, Complete)
    Type code: ORIGIN (1)
    Length: 1 byte
    Origin: IGP (0)
  AS_PATH: (64512) 1 (11 bytes)
    Flags: 0x40 (Well-known, Transitive, Complete)
    Type code: AS_PATH (2)
    Length: 8 bytes
    AS path: (64512) 1      AS_PATH属性包含两个Segment
      AS path segment: (64512)
        Path segment type: AS_CONFED_SEQUENCE (3)
        Path segment length: 1 AS
        Path segment value: 64512
      AS path segment: 1
        Path segment type: AS_SEQUENCE (2)
        Path segment length: 1 AS
        Path segment value: 1
  NEXT_HOP: 2.2.2.2 (7 bytes)
```

可以看出 AS_CONFED_SEQENCE 属性用于在联邦内防环（该属性不会出联邦），而上面的 AS_SEQUENCE 属性，则将会被随着 11.11.11.0 的路由被传递给 R7，因此在 R7 上抓包，11.11.11.0 的路由中看不到 AS_CONFED_SEQENCE 的属性，因此外部 AS 将联邦视为单个 AS（外界并不知道联邦内部的情况）

2.3 路由反射器 RR

1. 基本概念

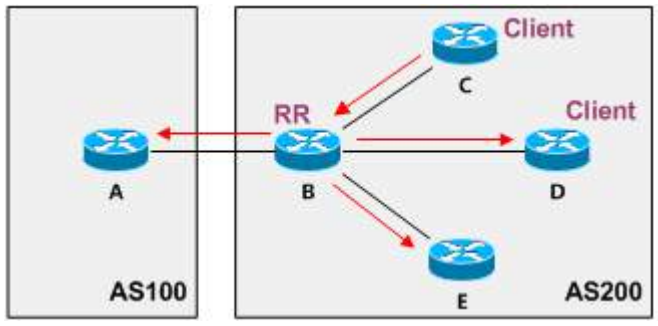
在 AS 内部，由于存在 IBGP 水平分割原则，使得 BGProuter 之间不得不两两建立 IBGP 连接，以求获得完整的 BGP 路由更新，然而这是个扩展性非常低的做法，同时也给网络设备带来了负担，解决 IBGP 扩展性问题的两种有效的办法是路由反射器及联邦。路由反射器相比于联邦，优势在于，联邦中所有路由器都需要支持并理解联邦机制，而路由反射器只需要 RR 理解反射器机制即可，另外，路由反射器的实现机制也相对简单一些。当然如果希望用各种 EBGP 机制来管理大规模 AS，那么联邦将是一个更优的解决方案。

思考路由反射器时，将簇当作一个逻辑的整体去考虑即可， RR 和 client 共同构成反射簇，但是只有 RR 知道（配置只是在 RR 上完成）。注意 RR 只通告或反射它所知道的最佳路径。

为了维护一致的 BGP 拓扑，RR 在反射路由的时候不修改某些 BGP 路径属性，这些属性包括 NH、AS_PATH、LOCAL_PREF 和 MED，并且增加了 ORIGINATOR 和 CLUSTER_LIST 用于防环。

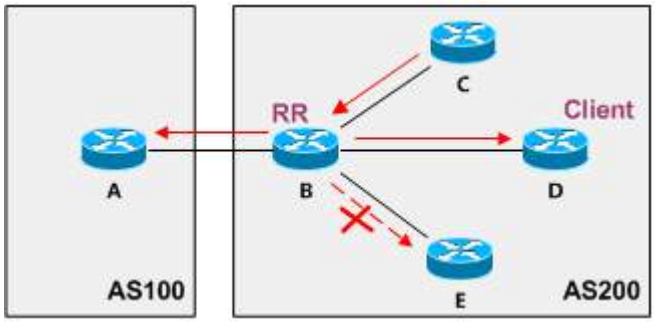
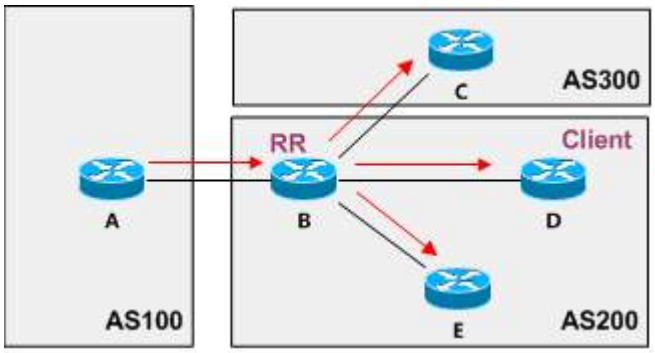
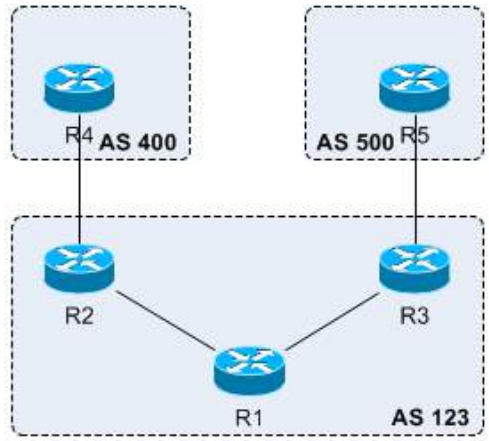
- 如果路由学习至非 client IBGP 对等体，则反射给所有 client 及 EBGP 邻居
- 如果路由学习至一 client，则反射给所有非 client IBGP 邻居和除了该 client 以外的所有 client
- 如果路由学习至 EBGP 邻居，则反射给所有 client 和非 client IBGP 邻居

2. 规则示例



B 是 RR，C 和 D 是 client，BCD 构成路由反射簇；有一条 BGP 路由从 C 通告出来。

当从簇的组成员传过来一 BGP 路由给 RR，RR 会反射给其他客户端、非客户端及 EBGP 邻居。

	将路由反射簇看做一个整体 那么C传给RR后，RR 传给它的 client（D），则不传给非 client (E)
	另一种情况，思路还是一样，将 RR 和 Client 看做一个整体，也就是将簇看做一个整体。
	考虑R2是RR的情况，R4和R5分别宣告loop口 对方能收到路由吗？ 如果R1是RR呢？ 注意 CLIENT 是不知道自己的身份的

3. 关于路由反射簇

- 路由反射簇包括反射器及其 Client
- 每一个簇都有唯一的簇 ID
- 每当一条路由被反射器反射后，该反射器（该簇）的 Cluster-ID 就会被添加至路由的 Cluster-list 属性中
- 每当反射器收到一条 Cluster-list 属性已经包含该簇的 ClusterID 的路由时，该路由基于防环的目的将不被反射

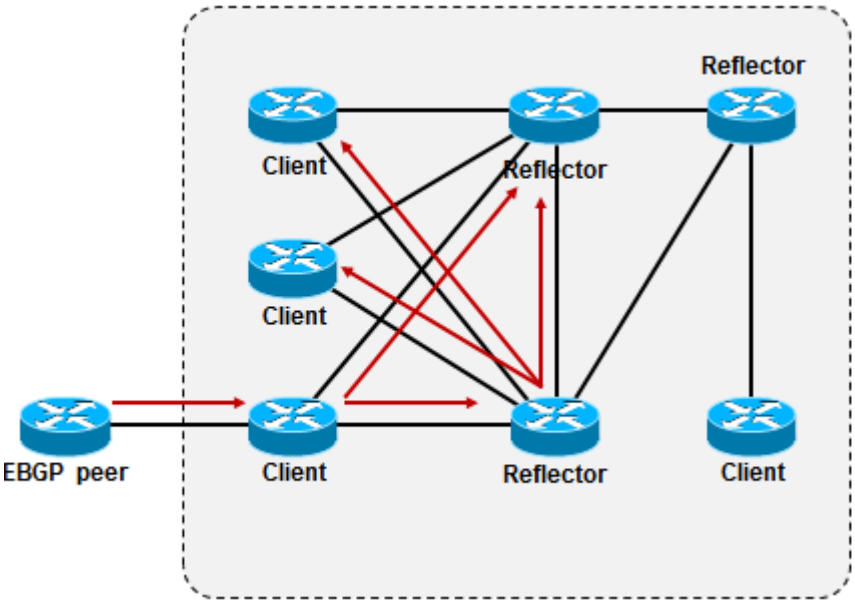
引入 Cluster 是要在 RR 的环境中提供冗余性。在传统的分簇设计中，多个 RR 用来为一个或多个 client 服务，这些 RR 都配置相同的 CLUSTER_ID，这个 ID 是 4 字节的 IP 形式的标示符，默认情况下就是 RR 自己的 BGProuterID，如果两台 RR 拥有相同的 CLUSTER_ID，那么他们就属于同一个簇。CLUSTER_ID 的另一个非常重要的作用是防环，当一台 RR 收到的 BGP 路由更新中携带了与自己 ROUTERID 相同的 CLISTER_ID，那么就该 RR 将忽略这条路由更新。

4. ORIGINATOR_ID 与 CLUSTER_LIST

本节内容请见 本文档路径属性的相关章节内容。

5. 冗余 RR 环境

单 RR 可能会存在单点故障的问题，因此从冗余性的角度，一个簇中可以拥有多台 RR，Client 与每一台 RR 都有物理连接并建立 BGP 对等体关系，在其中一台 RR 出现故障的情况下，Client 仍然有替代连接。Client 都不知道自己的 Client，因此 RR 本身也可以成为别人的 Client



- **冗余RR增加了网络的健壮性**
- **使用Originator、Cluster list属性来在冗余RR环境中避免路由环路。**
  - 例如将两个RR的ClusterID配置为一样，那么可以起到进一步的防环作用
  - 所有的RR之间建议采用全互联模型
- **Client会收到来自两个RR反射的路由，如何决策？**

由于 AS_PATH 属性在 AS 内部不会发生变化（仅当路由离开本 AS 才会被更新），因此 AS 内防环才有水平分割的机制，而路由反射器实际上是放宽了水平分割原则，这个就会给环路带来一定的隐患，因此路由反射器需使用以下两个属性防止环路：

ORIGINATOR_ID 和 CLUSTER_LIST 是路由反射器使用的可选非传递属性，用来防止环路。

详见本文档“ORIGINATOR_ID 和 CLUSTER_LIST”BGP 属性部分。

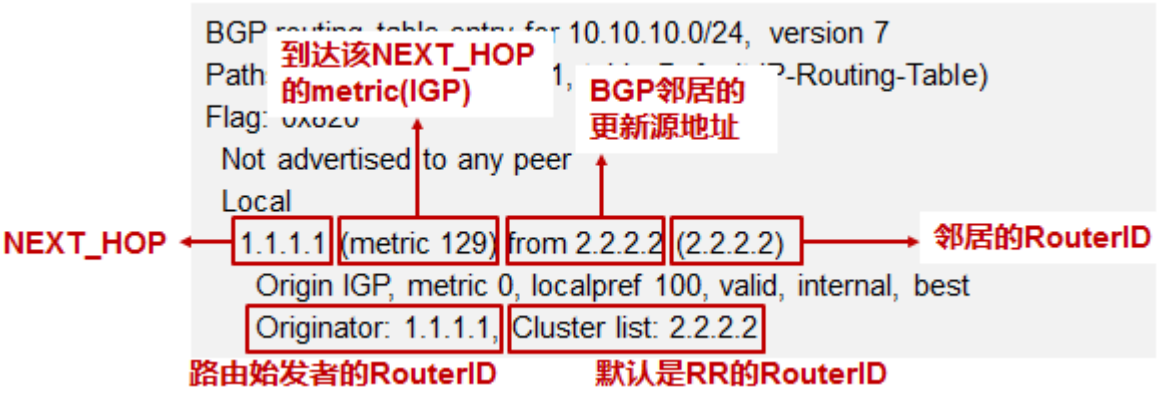
6. 配置命令

R2 的 BGP 配置如下：

```
router bgp 123
  neighbor 1.1.1.1 remote-as 123
  neighbor 1.1.1.1 update-source Loopback0
```

```
neighbor 1.1.1.1 route-reflector-client
```

R3 上 show ip bgp 10.10.10.0



RR 可修改 cluster-id

```
router bgp 123
  bgp cluster-id 222.222.222.222
```

其他配置命令：

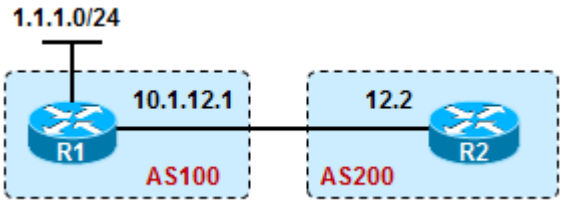
bgp client-to-client reflection

在配置反射器时，client 到 client 间的反射是默认开启的，但如果客户间是全互联的，此命令加 no，则关闭客户间的反射

7. **规划原则**
   - **路由反射器规划原则**
     - 路由反射器将传输 AS（transit AS）分割成小单元，也就是反射簇
     - 每个簇包含反射器及其 client
     - 不支持路由反射器功能的路由器可以充当单路由器簇或充当 client
   - **IBGP session 原则**
     - 路由反射簇中的所有 client 都应该与并且只与簇中所有的 RR 建立 IBGP 连接
     - AS 内的路由反射器之间要求全 IBGP 互联
     - 非反射器的路由器即可参与 IBGP 全互联也可配置为反射器的 client

2.4 自动汇总

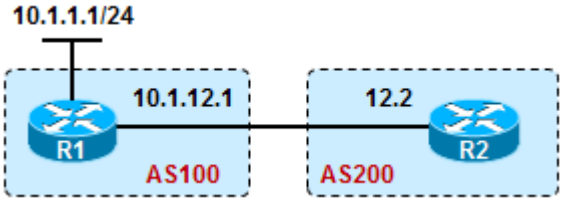
BGP 什么情况下会自动汇总？



- 若 R1 开启 auto-summary，并用重发布直连的方式引入 1.1.1.0/24，则该子网会被自动汇总给 R2
- 若 R1 开启 auto-summary，且 network 1.1.1.0 mask 255.255.255.0，则仍以明细更新给 R2

  若 R1 开启 auto-summary，且 network 1.0.0.0 mask 255.0.0.0，则通告给 R2 汇总路由 1.0.0.0/8

  上面这条 network 等同于 network 1.0.0.0（network 的有类宣告）



这个实验主要验证自动汇总如果不是发生在主类网络边界的情况：

- 若 R1 开启 auto-summary，并用重发布直连的方式引入 10.1.1.0/24，则该子网会被自动汇总给 R2
- 若 R1 开启 auto-summary，并 network 10.0.0.0，则该子网会被自动汇总给 R2

  从这个实验分析得出 BGP 的自动汇总，不要求主类网络边界，这与 IGP 要区别开。

因此 BGP 自动汇总（auto-summary）只汇总重发布引入的路由，以及使用 network 命令有类宣告方式引入的路由。目前 CISCO IOS 默认关闭自动汇总。

2.5 手工汇总

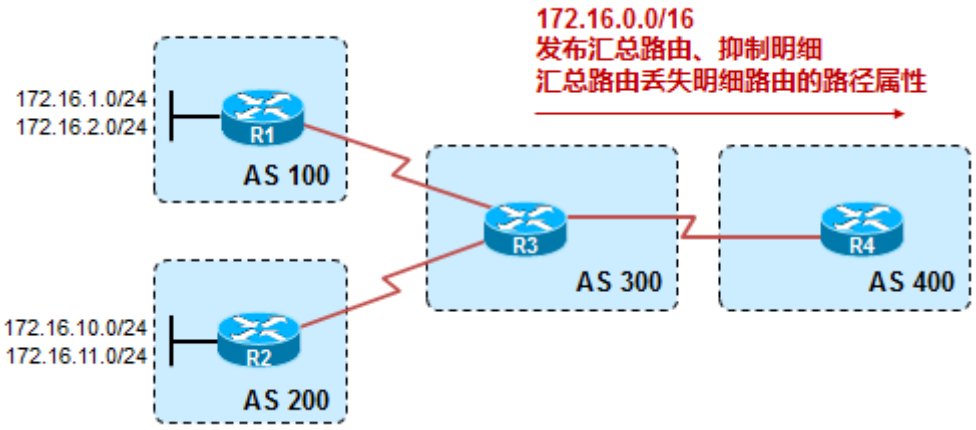
可以不 network 明细路由，而是在路由起源，配置一条汇总静态，然后 network 该汇总路由，这种方法不建议。

aggregate-address 命令是用于 BGP 手工汇总，以下是该命令所有子命令的详解

1. **aggregate-address 汇总地址 summary-only**

   如果 aggregate-address 不加任何关键字，则明细也传递，汇总路由也传递。

   加上参数 summary-only 则只传递汇总路由，明细路由被抑制。这种情况下产生的汇总路由，将会丢失底下明细路由的 AS_PATH 属性，因此可能存在一定的隐患。



R3 上 show ip bgp

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0	0.0.0.0			32768	i
s> 172.16.1.0/24	10.1.13.1		0	0	100 i
s> 172.16.2.0/24	10.1.13.1		0	0	100 i
s> 172.16.10.0/24	10.1.23.2		0	0	200 i
s> 172.16.11.0/24	10.1.23.2		0	0	200 i

本地产生的这条汇总路由 172.16.0.0/16，NH=0.0.0.0，weight=默认值 32768，origin=i。

明细路由标记为“s”，因此都被抑制了。

R4#show ip bgp 172.16.0.0

```
BGP routing table entry for 172.16.0.0/16, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  300, (aggregated by 300 3.3.3.3)
    10.1.34.3 from 10.1.34.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
```

可以看到，路由带上了 atomic-aggregate 属性，用来告知下游邻居这是汇总路由且丢失了明细的路径属性。同时 aggregator 属性标识了汇总的地点（AS 及汇总路由器的 RouterID）。

报文抓取如下（R3 发给 R4 的 BGP update 包）：

```
UPDATE Message
    Marker: 16 bytes
    Length: 63 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 37 bytes
  Path attributes
    ORIGIN: IGP (4 bytes)
    AS_PATH: 300 (7 bytes)
    NEXT_HOP: 10.1.34.3 (7 bytes)
    MULTI_EXIT_DISC: 0 (7 bytes)
    ATOMIC_AGGREGATE (3 bytes)
      Flags: 0x40 (Well-known, Transitive, Complete)
      Type code: ATOMIC_AGGREGATE (6)
      Length: 0 bytes
    AGGREGATOR: AS: 300 origin: 3.3.3.3 (9 bytes)
      Flags: 0xc0 (Optional, Transitive, Complete)
      Type code: AGGREGATOR (7)
      Length: 6 bytes
      Aggregator AS: 300
      Aggregator origin: 3.3.3.3 (3.3.3.3)
  Network layer reachability information: 3 bytes
    172.16.0.0/16
```

2. aggregate-address 汇总地址 summary-only as-set

汇总命令加上 as-set 关键字之后，产生的这条汇总路由就可以继承明细路由的某些路径属性，从而规避一些问题。as-set 继承明细属性的规则如下：

As-path: 将收到的所有明细路由的 as 号都放置在{}中，计算 AS_Path 长度时这些 AS 只被算为 1 个 AS

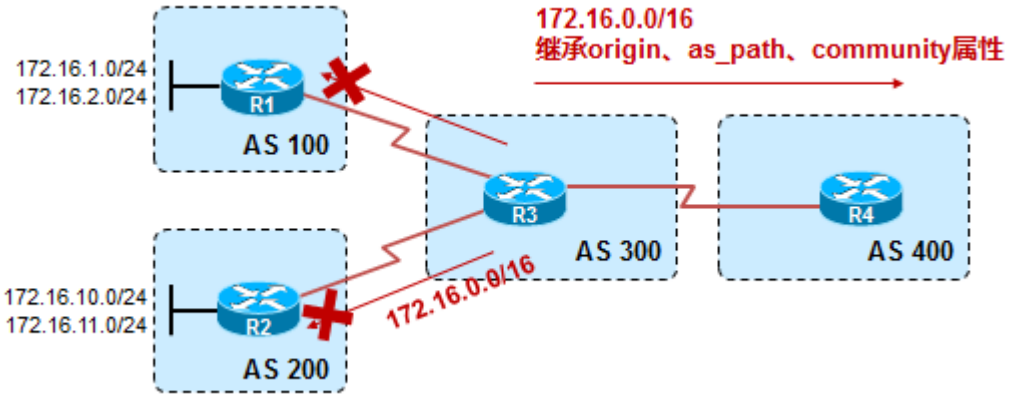
Origin: 继承最差的 origin 属性

Community: 继承所有明细路由的 community，形成一个列表

MED 不继承

LP 取明细路由中 LP 的最大值

NEXT_HOP 汇总路由为 0.0.0.0（因为汇总路由为本地产生）



在 R3 上做汇总，使用 aggregate-address summary-only as-set，则 R3 上的 BGP 表：

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0	0.0.0.0		100	32768	**{100,200}** i
s> 172.16.1.0/24	10.1.13.1	0		0	100 i
s> 172.16.2.0/24	10.1.13.1	0		0	100 i
s> 172.16.10.0/24	10.1.23.2	0		0	200 i
s> 172.16.11.0/24	10.1.23.2	0		0	200 i

看到 R3 上产生的汇总路由，AS_PATH 继承了明细的 AS_PATH，以{100，200}的形式呈现，这样就可以起到防止环路的作用，而不会由于丢失明细路由的 AS_PATH 而带来隐患。注意这里{}内的 AS_PATH 类型为 AS_SET，是无序的 AS 列表。

R4 上 show ip bgp 172.16.0.0

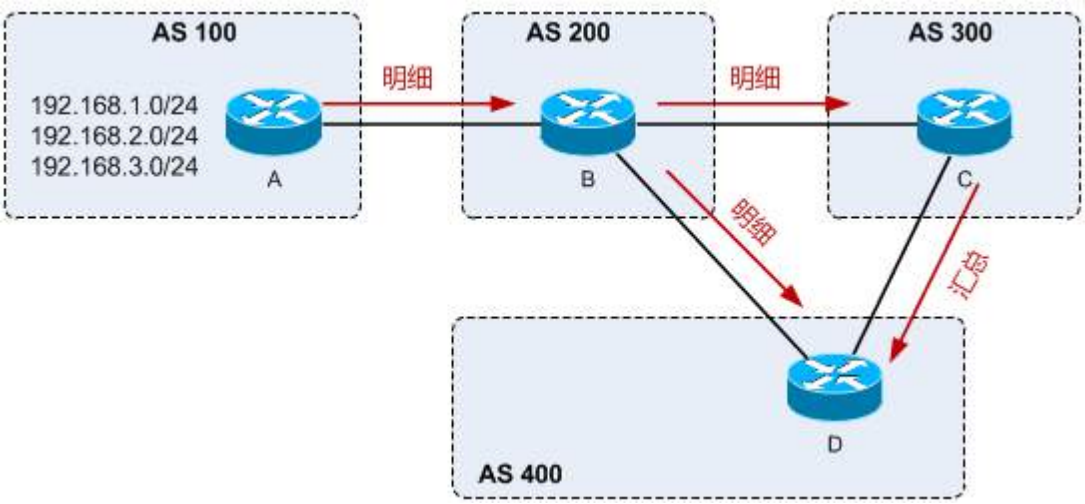
```
BGP routing table entry for 172.16.0.0/16, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  300 {100,200}, (aggregated by 300 3.3.3.3)
    10.1.34.3 from 10.1.34.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, best
```

可以看到，由于我们配置汇总的时候，使用了 as-set 关键字使得汇总路由得以继承明细的部分路径属性，因此产生的汇总路由仍保留有 **aggregator** 属性，但是没有 atomic-agg 属性，显然这个属性在 as-set 关键字使用的情况下已经没有必要了。

继续看看 R4 上抓包的结果（R3 发给 R4 的 BGP update 包）：

```
Path attributes
  ORIGIN: IGP (4 bytes)
  AS_PATH: 300 {100, 200} (13 bytes)
    Flags: 0x40 (Well-known, Transitive, Complete)
    Type code: AS_PATH (2)
    Length: 10 bytes
    AS path: 300 {100, 200}
      AS path segment: 300
        Path segment type: AS_SEQUENCE (2)
        Path segment length: 1 AS
        Path segment value: 300
      AS path segment: {100, 200}
        Path segment type: AS_SET (1)
        Path segment length: 2 ASs
        Path segment value: 100 200
  NEXT_HOP: 10.1.34.3 (7 bytes)
  MULTI_EXIT_DISC: 0 (7 bytes)
  AGGREGATOR: AS: 300 origin: 3.3.3.3 (9 bytes)
    Flags: 0xc0 (Optional, Transitive, Complete)
    Type code: AGGREGATOR (7)
    Length: 6 bytes
    Aggregator AS: 300
    Aggregator origin: 3.3.3.3 (3.3.3.3)
```

补充说明 as_set 关键字的作用：



A 将明细路由通告给 B，B 将明细路由通告给 C 及 D，C 通告汇总路由，这个时候汇总路由除了会通告给 D 外，也会被传递给 B，而这样一来可能会出问题，因此需要在 C 上 neighbor B 的 IP distribute-list x 来过滤掉汇总路由，否则会出现路由环路。

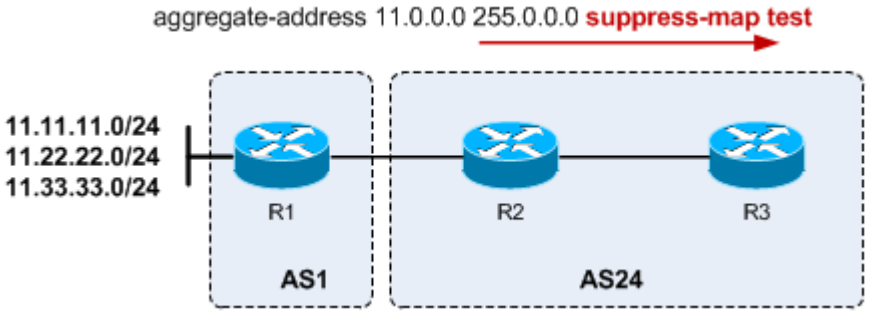
另外，汇总路由被传递给 D 后，D 会继续向 B 来传递，而由于汇总路由源于 C，因此到达 D 的 AS400 时，AS_PATH 仅有 AS300、400，于是乎 B 路由器接受了该条汇总（因为没有看到自己的 AS 号出现在 AS_PATH 中）。这样就可能出现环路。解决的办法是在 C 发布汇总路由的时候，设置 AS_PATH 关键字，以还原 AS 属性，那么这时候，这条汇总路由就会同时宣告 AS_SET，而 AS_SET 是一个无序的 AS 列表，当中就有 AS200 的信息，这样 B 路由器收到这条汇总路由，发现 AS_SET 中有自己的 AS 因此忽略该路由。

3. aggregate-address 汇总地址 suppress-map xxx as-set

用于宣告聚合及选定的明细路由（抑制特定的明细路由），后面跟上 route-map xxx，被 route-map 匹配（permit）的路由将被过滤，其他放行。

抑制列表虽然调用 route-map，但是 route-map 只能用于匹配，不能用于设置属性（不能用 set 命令）

【实验 1】 以下是几种情况




```
ip prefix-list 1 permit 11.11.11.0/24
route-map test per 10
    match ip add pre 1
// 干掉 11.0   放行除了 11.0 外的所有明细
```

```
ip prefix-list 1 permit 11.11.11.0/24
route-map test deny 10
    match ip add pre 1
// 等同于 route-map 不匹配（permit）任何条目， 因此所有明细都放行
```

```
ip prefix-list 1 deny 11.11.11.0/24
route-map test permit 10
    match ip add pre 1
// 效果同上
```

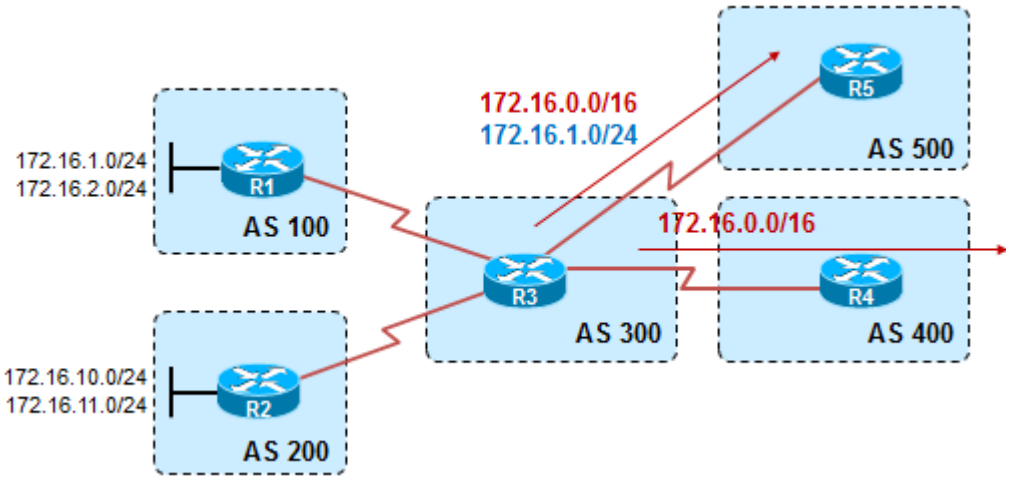
```
route-map test permit 10
// 明细全都不放行
```

```
route-map test deny 10
// 明细全放行
```

【实验 2】邻居+Route-map 实现相同功能

注意用 route-map 实现的话，最后是隐含干掉 any 的，而 **suppress-map 则不同**

【实验 3】针对特定邻居取消明细路由抑制



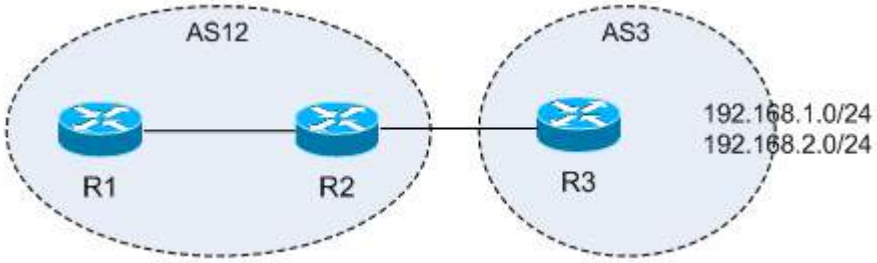
R3 的配置如下：

```
access-list 1 permit 172.16.1.0
route-map unsupp permit 10
 match ip address 11
router bgp 300
    neighbor 10.1.35.5 unsuppress-map unsupp
    aggregate-address 172.16.0.0 255.255.0.0 as-set summary-only
```

R3 上做汇总，抑制掉了所有明细，但是如果只想对特定的邻居放行部分明细，那么可以用 unsuppress-map

4. aggregate-address 汇总地址 attribute-map abc

该命令可以更改汇聚路由的属性（注，仅仅对汇总路由产生作用，对明细不起效）。



R2 上对 192.168.0.0/16 进行汇总

```
router bgp 12
aggregate-address 192.168.0.0 255.255.0.0 attribute-map test        // 关联 route-map test 对汇聚路由起效
Route-map test permit 10
  Set origin incom
  Set metric xx
```

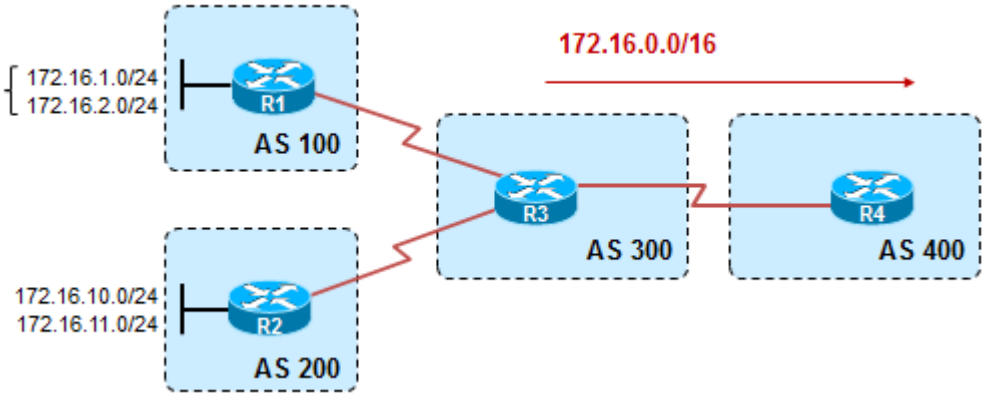
然而这个时候，R3 也会收到来自 R2 发布的汇总路由，这样就有可能形成路由环路，因此需过滤掉。

在 R2 上 BGP 进程增加配置：

```
neighbor 1.1.23.3 distribute-list 1 out
access-list 1 deny 192.168.0.0 0.0.255.255
```

5. aggregate-address 汇总地址 as-set advertise-map

advertise-map 与 summary-only 合用时，aggregate-address 的汇总地址下所有明细均被抑制，同时 advertise-map 匹配的条目中明细如果全都挂了，则汇总路由也消失，(只要 advertise-map 匹配的明细有一条在，汇总就在)；并且汇总路由仅继承 advertise-map 匹配明细路由的 BGP 路径属性



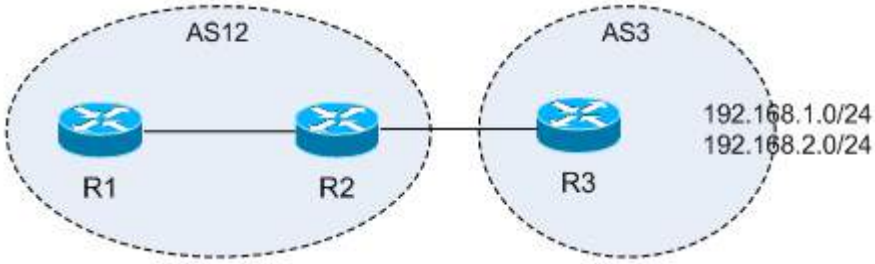
在 R3 上

```
ip prefix-list 1 permit 172.16.1.0/24
ip prefix-list 1 permit 172.16.2.0/24
route-map test permit 10
  match ip address prefix-list 1
router bgp 300
  aggregate-address 172.16.0.0 255.255.0.0 as-set summary-only advertise-map test
```

则 R1、R2 上所有 172 网段均被抑制 ;同时由于 route-map test 只匹配了 1.0 及 2.0 网段(AS100 中的)，因此这条汇总路由，AS_PATH 只继承 AS100，也就是 300 100 ；

同时，如果此时 172.16.1.0 及 172.16.2.0 全都 DOWN 掉了，则汇总路由也消失

另一个例子



R3 宣告 192.168.1.0、2.0，同时设置 1.0 的 community 属性为 no-adv

```
access-list 1 permit 192.168.1.0
route-map test permit 10
    match ip address 1
    set community no-advertise
route-map test permit 20
    set community none
router bgp 3
    neighbor 1.1.23.2 send-community
    neighbor 1.1.23.2 route-map test out
```

这时默认情况下，R2 能学习到 1.0、2.0，而 R1 只能学习到 2.0

当 R2

aggregate-address 192.168.0.0 255.255.0.0 之后，R1 能学习到汇总路由及 2.0

当 R2

aggregate-address 192.168.0.0 255.255.0.0 as-set 之后，R1 只能学习到 2.0，汇总路由被抑制掉了，原因是 1.0 携带了 no-adver 的 community，当汇总路由加了 as-set 关键字后，会继承它的 community，因此汇总路由也携带了 no-adver，这个时候

```
access-list 11 deny     192.168.1.0
access-list 11 permit any
route-map adv permit 10
match ip address 11
router bgp 12
 aggregate-address 192.168.0.0 255.255.0.0 as-set advertise-map adv          // 即可忽略 1.0 的 community
```

注意，这里是在构建汇总路由时不考虑 advertise-map 中拒绝的路由，而单纯通过 adv-map 是无法过滤明细路由的

2.6 BGP Deaggregation

BGP Deaggregation，也称为 BGP 拆分。路由汇总我们都知道是什么概念，路由汇总的优势是非常明显的，可以减少路由表的条目从而优化网络，但是同时却也丢失了邻居传递来的路由的精确性，对于汇总前的明细路由我们就一无所知了。那么拆分可以理解为汇总的逆向动作，当我收到一条汇总路由的时候，可能基于某种目的，我希望从汇总路由中抽出特定的明细路由，以此来加强路由的颗粒度，当然这条明细是依赖汇总路由而存在的。

拆分可以通过使用条件注入（conditional injection）来完成，所谓的 conditional injection 指的就是，当特定的汇总路由存在时，我可以生成其下属的特定明细，这些明细路由将被注入到本地 BGP RIB（本地路由表也会加载明细路由信息），以便在本地 AS 中提供比汇总路由更详细的路由选择信息（更长的前缀）。

Conditional inject 的配置如下（BGP 路由选择进程模式下）：

```
bgp inject-map map1 exist-map map2 [copy attributes]
```

上述命令的意思是当 map2 所匹配的汇总路由正常时，在本地 BGP RIB 中注入 map1 中定义的明细路由。当汇总路由挂掉，这条明细也就跟着消失，这就是所谓的条件注入—conditional injection。下面我们在看来一下这两个 route-map 的详细内容，这些是需要格外注意的。

- **exist-map 使用的 route-map 最少具有以下两个 match 语句：**

  match ip address prefix-list

  上面这条 match 语句用来匹配汇总路由

  match ip route-source

  上面这条 match 语句用来匹配发送该汇总路由的邻居 IP。如果指定了 copy attributes 选项，那么被 inject 的明细路由会继承汇总路由的路径属性，否则明细将被当成本地生成的路由。

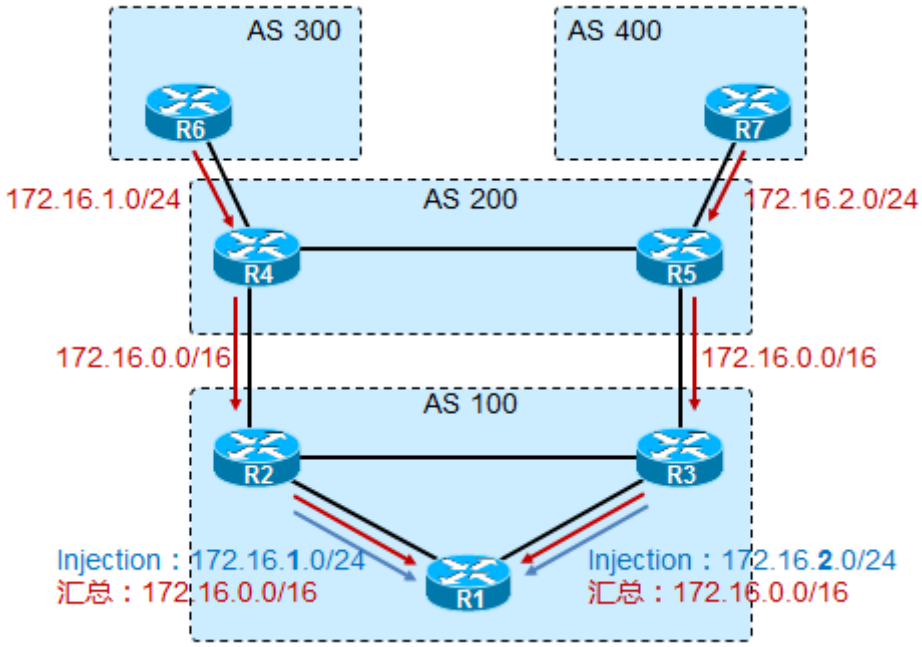
- **Inject-map 使用的 route-map 中**

  Set ip address prefix-list

  上面的这条 set 命令用来定义将被注入到本地 BGP RIB 的明细路由。被注入的前缀可以使用

  Show ip bgp injected-path 来显示

下面，我们来看一个示例：



看看上面的拓扑，AS300 中有子网路由 172.16.1.0/24，AS400 中有路由 172.16.2.0/24。

这些子网路由在传递到 AS200 后，由 R4 及 R5 做路由汇总，汇总路由被传递给 AS100。这时候 R1 如果要去往 172.16.1.0 及 2.0 子网，可能就是走一侧，要么走 R2，要么走 R3，这当然不是最优的实现方式，我们希望看到 R1 去往 1.0 子网走 R2，去往 2.0 子网走 R3，那么实现的思路就是在 R2 及 R3 上部署 BGP deaggregation，由 R2 向 AS100 注入条件明细 172.16.1.0/24，R3 注入 172.16.2.0/24，同时，为了防止这两个 conditional 子网路由回流造成不可预估的影响，我们同时为这两条路由分配两个 community 值，1 个是 no-export，另一个是 100:200，其中 100 就不说了，AS200 表示，这条 conditional 路由是针对 AS200 的。

R2 的配置如下：

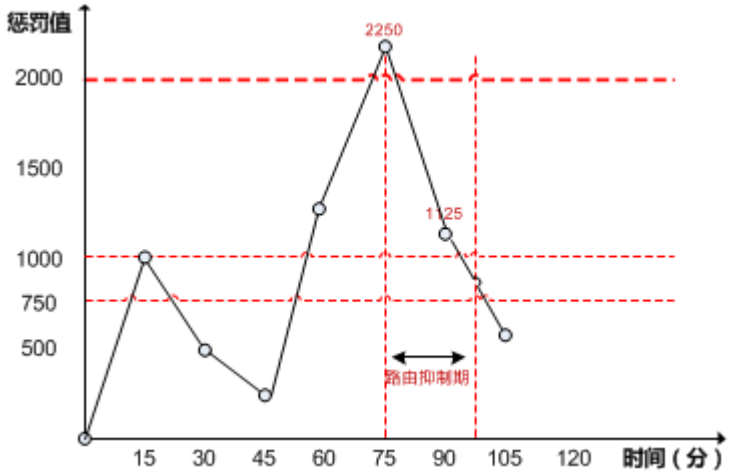
```
ip prefix-list huizong permit 172.16.0.0/16          //用来匹配汇总路由
ip prefix-list mingxi permit 172.16.1.0/24           //用来定义准备注入的条件前缀
ip prefix-list xiayitiao permit 10.1.24.4/32         //用来匹配传递给我汇总路由的 BGP 邻居，这里是 R4 的 IP
route-map RP_mingxi permit 10
 set community 100:200 no-export                     //100:200 表示这是针对 AS200 的
 set ip address prefix-list mingxi
route-map RP_huizong permit 10
 match ip address prefix-list huizong
 match ip route-source xiayitiao
router bgp 300
 bgp inject-map RP_mingxi exist-map RP_huizong copy-attributes
 neighbor 10.1.23.2 remote-as 200
```

R3 的配置大同小异。

2.7 Route Dampening

当路由出现摆动（不断失效、恢复），就给它分配一个惩罚值，摆动越多，惩罚值越大并且不停地积累。
而同时惩罚值又以一定的速率降低，每一个半衰期结束，惩罚值变成原来的一半（如果路由不再翻动的话）
如果惩罚值超出了预先设置的门限--抑制界限：惩罚值达到这个门限，路由被抑制，即不发布，直到 N 个半衰期以后，惩罚值降低到另一个门限：重新使用门限（解除抑制的门限）时，才解除对路由的抑制。

惩罚值： 每次摆动增加 1000
抑制界限： 2000
重新使用界限： 750
半衰期： 15 分钟
最大抑制时间： 60 分钟（半衰期的 4 倍）



正常所有路由的惩罚值都是 0，惩罚值是瞬间加的
例如某条路由，出现翻动，则瞬间惩罚值到 1000，这时有个半衰期（15 分钟），15 分钟后（如果路由没有再翻动），惩罚值降低为一半 500，再经过 15 分钟变成 250，这时如果路由又发生一次抖动，则惩罚值再加 1000，变成 1250，在此之前，这条路由正常使用。如果又抖动了一次，则变成 2250，超过了阀值 2000，则这条路由将被抑制，不更新也不使用。15 分钟后，（如果不再翻动了）降低为一半 1125（但仍然被抑制），如此经历数个半衰期，直到降低为 750（重新使用界限），路由再次被启用。

```
bgp dampening                    // 启动 bgp dampening，默认对全部 EBGP 路由有效，也可加 route-map
bgp dampening    ?               // 修改默认参数
bgp dampening  半衰期  重新使用界限  抑制界限  最大抑制时间
```

show ip bgp

如果路由标记 d，则表示该路由被抑制，如果是 h，则表示路由有翻动的迹象

show ip bgp flap // 查看路由翻动情况

Show ip bgp dampening // 查看哪些路由被抑制了

Dampening 只对 EBGP 路由生效，对 IBGP 路由无效

3 BGP 路径属性

路径属性的分类：

公认属性 (Well-known)	**公认必遵 (Well-known mandatory)**	BGP 必须都能识别 ,且在更新消息必须包含	Origin AS-Path Next hop
	公认自决 (Well-known discretionary)	BGP 必须都能识别 ,更新消息可包含可不包含	Local-Preference ATOMIC_Aggregate
可选属性 (Optional)	**可选传递 (Optional transitive)**	可以不支持该属性，但即使不支持，也应当接受包含该属性的路由并传递给其他邻居	Community Aggregator
	可选非传递 (Optional non-transitive)	可以不支持该属性，不识别的 BGP 进程忽略包含这个属性的更新消息，并且不传递给其他 BGP 邻居	MED Originator_ID Cluster_list Weight

3.1 Origin

公认必遵属性，明确了路由更新的来源，三种途径

- IGP i 通过 BGP network，也就是起源于 IGP，因为 BGP network 必须保证该网络在路由表中
- EGP e 是由 EGP 这种早期的协议重发布而来
- Incomplete ? 从其他渠道学习到的 (确认该路由来源的信息不完全)，重发布的路由 origin 都是这个标记

优选原则是 IGP > EGP > Incomplete

R2#sh ip bgp

BGP table version is 10, local router ID is 2.2.2.2

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete

Network Next Hop Metric LocPrf Weight Path

r >i1.1.1.0/24	1.1.1.1	0	100	0	i
* >i11.11.11.0/24	1.1.1.1	0	100	0	?

3.2 AS_Path

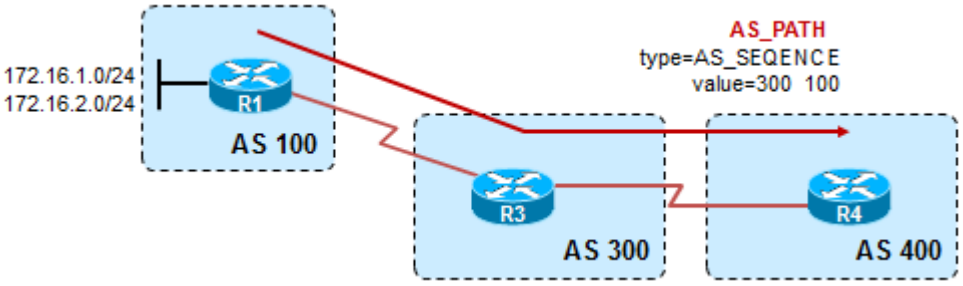
公认必遵属性，描述到达目标网络所要经过的 AS 号序列。最重要的作用是防环，如果 BGP speaker 发现自己的 AS 号出现在接收到的 BGP 路由更新的 AS_PATH 中，那么说明可能有路由环路，则忽略该路由更新。另一个重要的作用，是在 BGP 选路过则中，作为 AS 跳数的丈量，当然，AS_PATH 中包含的 AS 个数越少，表示距离目的地更近。

仅当 update 消息被发送给其他的 AS（EBGP peer）时，BGP 路由器才会将其 AS 号追加在 AS_PATH 中。这句话也隐含了另一个意思，那就是如果要修改 AS_PATH 属性，则必须在 AS 边界路由器上执行策略，对 IBGP 邻居去执行修改 AS_PATH 的策略是无效的。

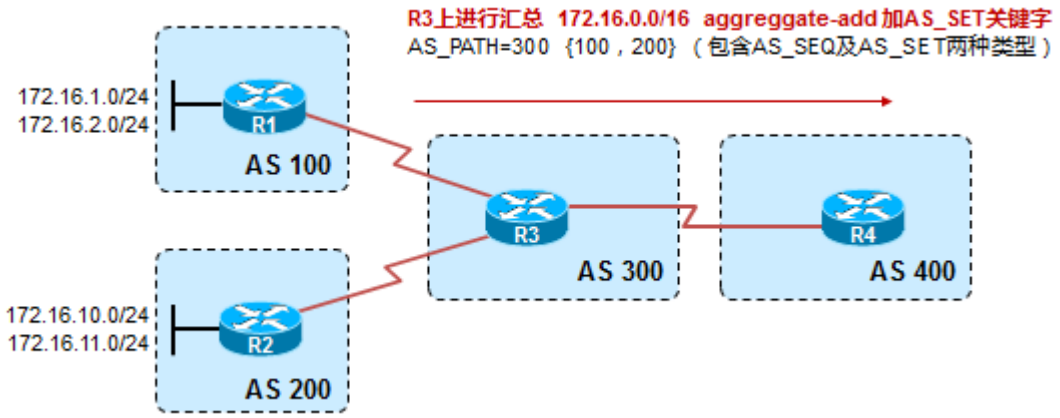
1. AS_PATH 路径属性的四种类型

- **AS_SET：** 一个去往特定目的地所经路径上的无序 AS 号列表
- **AS_SEQENCE：** 一个有序的 AS 号列表
- **AS_CONFED_SEQUENCE** 一个去往特定目的地所经路径上的有序 AS 号列表，其用法与 AS_SEQUENCE 完全一样，区别在于该列表中的 AS 号属于本地联邦中的 AS
- **AS_CONFED_SET** 一个去往特定目的地所经路径上的无序 AS 号列表 ,去用方法与 AS_SET 完全一样，区别在于列表中的 AS 号属于本地联邦中的 AS

以上四种类型是通过 AS_PATH 属性中的类型代码进行区分。关于这两个联邦特有的 AS_PATH 类型，详细内容请见本文档“联邦”一小节，对于前两种 AS_PATH 怎么理解呢？



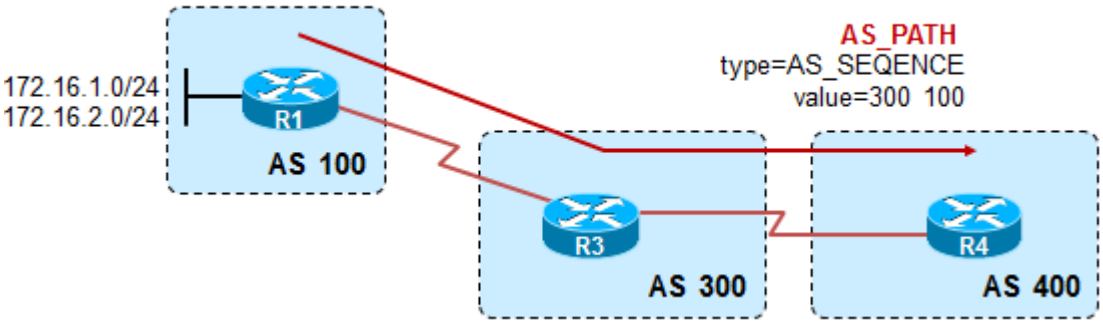
AS_Sequence 很好理解，上图在不做任何策略的情况下，传递到 R4 的 BGP 路由携带的 AS_PATH 的类型即为 AS_Sequence ,这是个有序的 AS 号列表 ,那么当 R4 收到路由更新的时候 ,AS_PATH 为 300,100，R4 就知道要去目的地，需经 AS300 再到 AS100，才可达目的。



再看上面这张图，在 R3 上我们做了手工路由汇总，这条汇总路由由 R3 产生并且传递给了 R4，那么这时候这条汇总路由的 AS_PATH 为 300，丢失了其下的明细路由的 AS_PATH 属性，那么如果 R4 与 R1 或 R2 之间如果有 EBGP 的连接，那么就出现了路由环路，R1、R2 都会收到这条汇总路由并且接受路由更新。这是因为这条汇总路由并未携带任何关于 AS100 及 AS200 的信息。

所以我们其实还是需要在汇总路由上保留其明细的 AS_PATH，而至于保留下来的明细路由的 AS_PATH 中 AS 号是什么顺序已经不重要了，因为我们出于防环的目的的话，只要 AS 号就够了。所以 R3 产生的这条汇总路由其实携带了两个 AS_PATH 类型，一是 AS_Seq，值为 300，另一个类型为 AS_set，包含{100,200}两个 AS 号，并且是无序的。于是乎 R4 上收到的 AS_PATH 综合起来就是 300 {100,200}。值得注意的是，这个括号里的 AS 号，在进行 AS_PATH 计算的时候，只当一跳来算，关于这点的验证，请看选路规则章节。

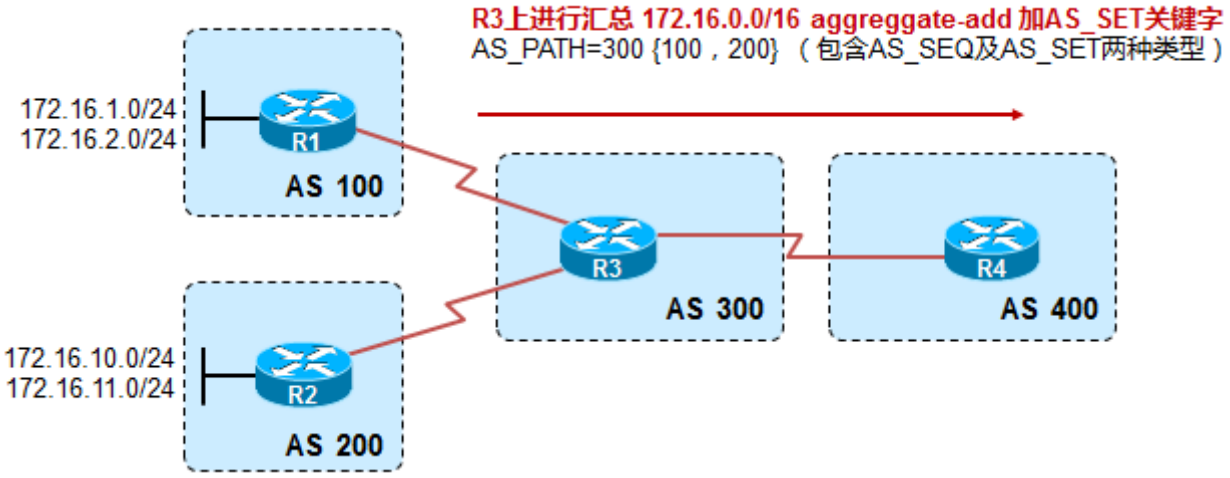
2. AS_PATH 类型 2：AS_SEQENCE 详解



上图仅做基本的 BGP 配置，在 R1 上发布 1.0 及 2.0，不做汇总，R4 收到的 1.0 及 2.0 BGP 路由，BGP 更新包如下，可以留意到 AS 的长度为 2，即 “300，100”，注意，这是个有序的 AS 列表。

```
Border Gateway Protocol
⊟ UPDATE Message BGP updata包
    Marker: 16 bytes
    Length: 51 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 20 bytes
  ⊟ Path attributes BGP路径属性
    ⊞ ORIGIN: IGP (4 bytes)
    ⊟ AS_PATH: 300 100 (9 bytes) AS_PATH属性开始
      ⊞ Flags: 0x40 (well-known, Transitive, Complete)
        Type code: AS_PATH (2)   BGP路径属性类型字段
        Length: 6 bytes
      ⊟ AS path: 300 100
        ⊟ AS path segment: 300 100
            Path segment type: AS_SEQUENCE (2) AS_PATH类型字段
            Path segment length: 2 ASs   该AS_PATH含2个AS
            Path segment value: 300 100  值为300 100
    ⊞ NEXT_HOP: 10.1.34.3 (7 bytes)
  ⊟ Network layer reachability information: 8 bytes
    ⊞ 172.16.2.0/24   路由前缀
    ⊞ 172.16.1.0/24
```

3. AS_PATH 类型 1： AS_SET 详解



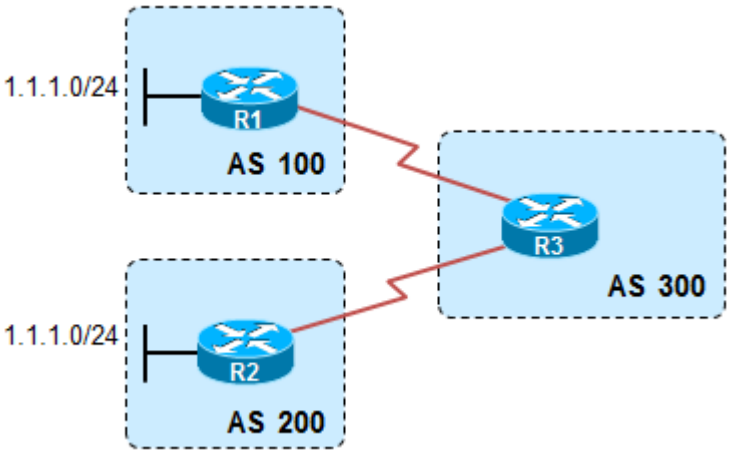
R3 上对 R1、R2 过来的明细进行汇总（汇总命令加上 AS_SET 关键字），这样一来 R3 产生的这条 BGP 汇总路由就继承了明细的 AS_PATH 属性，在 R4 上抓包会发现汇总路由的 AS_PATH=300 {100,200}：再看看抓包的结果，发现该条汇总路由的 AS_PATH 属性包含两部分（segment），值为 300 的这个 segment 为 AS_SEQ 类型，值为{100，200}的这个 segment 为 AS_SET 类型。所以 AS_SEQ 这个 AS_PATH 用来标识这条汇总路由的起源地，它是有序的 AS 列表，另外 AS_SET 这个 AS_PATH 类型用来标识汇总前的明细的 AS 列表，它是无序的，亦可用来防环。当使用了 as-set 关键字的手工汇总命令，由于继承了明细的 AS_PATH 等属性，汇总路由的路径属性已经很丰富和完备了，因此这种情况下就不需要 atomic-aggregate 这个属性了，我们看下面的抓包结果，有 aggregator，但是没有 atomic-agg。

```
Border Gateway Protocol
⊟ UPDATE Message
    Marker: 16 bytes
    Length: 66 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 40 bytes
  ⊟ Path attributes
    ⊞ ORIGIN: IGP (4 bytes)
    ⊟ AS_PATH: 300 {100, 200} (13 bytes)
      ⊞ Flags: 0x40 (Well-known, Transitive, Complete)
        Type code: AS_PATH (2)
        Length: 10 bytes
      ⊟ AS path: 300 {100, 200}
        ⊟ AS path segment: 300
            Path segment type: AS_SEQUENCE (2)
            Path segment length: 1 AS
            Path segment value: 300
        ⊟ AS path segment: {100, 200}
            Path segment type: AS_SET (1)
            Path segment length: 2 ASs
            Path segment value: 100 200
    ⊞ NEXT_HOP: 10.1.34.3 (7 bytes)
    ⊞ MULTI_EXIT_DISC: 0 (7 bytes)
    ⊞ AGGREGATOR: AS: 300 origin: 10.1.34.3 (9 bytes)
  ⊟ Network layer reachability information: 3 bytes
    ⊞ 172.16.0.0/16
```

AS_SEQ

AS_SET，是无序的

4. 使用 route-map 修改 AS_PATH



可以通过策略控制 AS_PATH 从而来影响路由选路 ,例如上图 如果希望 R3 到达 1.1.1.0 网络优先走 R1，那么可以在 R2 上做策略，如下：

```
Ip prefix-list 1 permit 1.1.1.0/24
route-map test
    match ip add pre 1
```

```
 set as-path prepend 6666
neighbor 10.1.13.3 route-map test out
```

与此一来，R3 上从 R2 学来的 1.1.1.0 BGP 路由，AS_PATH 就由原先的 “200 i”变成“200 6666 i”，当然这不太建议，因为 6666 这个 AS 并不存在，虽然加长了 AS_PATH 但是带人了一个压根不存在 AS 这不是扯淡么，我们可以 set as-path prepend 200 200 200（重复本 AS），如此来增长路由的 AS_PATH 从而实现路由策略并且规避不必要的麻烦。另外 如果同样的配置 在R3上对R2 neighbor 10.1.23.2 route-map test，in 那么 R3 上从 R2 学到的关于 1.1.1.0 的路由 AS_PATH 就会变成“6666 200 i”，这是因为 R3 收到 1.1.1.0 时，已经是“200 i”了，因此做 prepend 只能在 200 i 前面插入，请注意与前面的区别。

注意：使用策略修改 AS_PATH 只能在 BGP 路由器上对其 EBGP 邻居执行，那是因为 AS_PATH 只有在 AS 边界才会发生改变，因此如果对 IBGP 邻居做 AS_PATH 的修改策略，那就扯淡了。

5. AS_PATH 几个命令

```
Neighbor 路由来源邻居  allowas-in 几个我的 as 号
```

一般 BGP 路由器是不收包含自己 AS 号的路由信息的，但是可以 **neighbor** *路由来源邻居* **allowas-in** *几个我的 as 号*，来破例。

```
bgp bestpath as-path ignore
```

让 CISCO 路由器在其决策过程中忽略 AS-path 属性

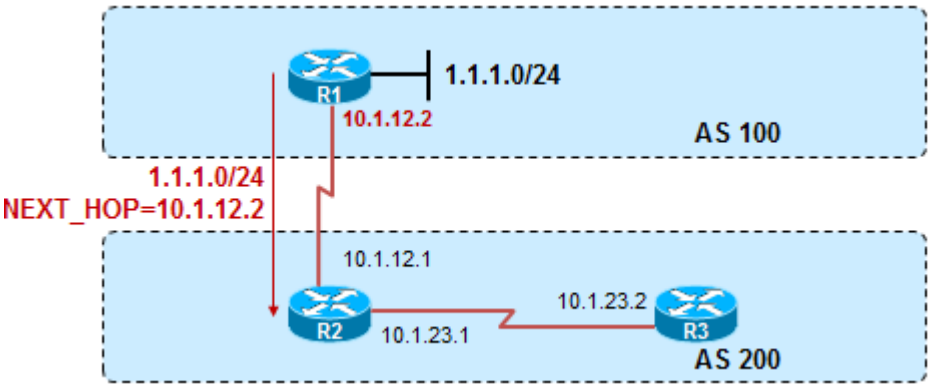
3.3 NEXT-HOP

公认必遵属性，描述了到目的地的下一跳路由器。

BGP next-hop 的规则如下：

1. 如果 BGP 路由传递自 EBGP peer

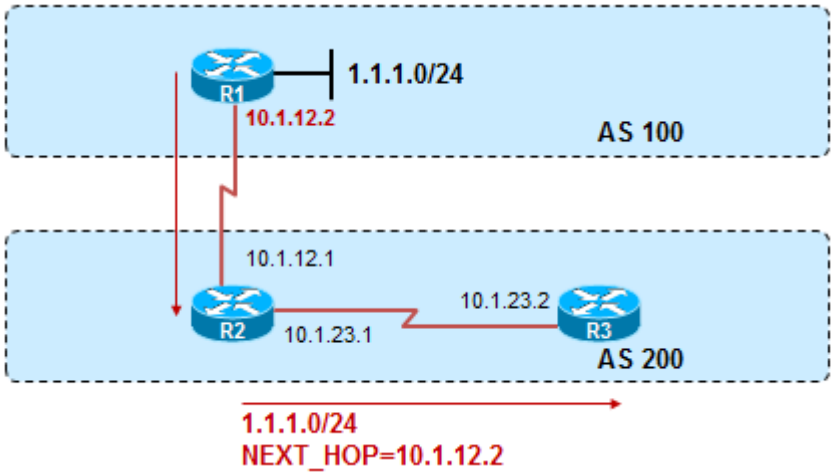
那么这条 BGP 路由的 NH 就是通告者的接口 IP。EBGP 邻居一般使用对方的直连接口 IP 互相指 neighbor，但如果是采用 LOOPBACK 接口建 EBGP 邻居，则 NH 就是 EBGP 邻居的更新源地址（当然一般 EBGP 邻居不使用 LOOPBACK 口建邻居）。



【补充】　若同 AS 内的 BGP 路由器 A 引入了某路由，A 路由器将该路由传递给了其 IBGP 邻居 B，又被 B 传递给了其 IBGP 邻居 C（做了反射器），则在 C 上，该条路由 NEXT_HOP 仍然为 A 路由器（的更新源 IP）

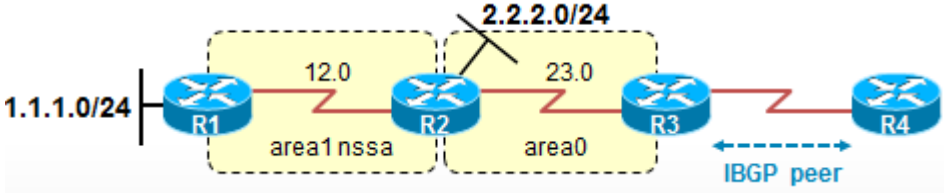
2. 如果路由传递自 IBGP 邻居，并描述的是 AS 外的目的地

从外部 AS 传递进来的路由，NH 为通告该路由的 EBGP peer，并且这个 NH 会跟随着路由在 AS 内传递而不发生改变（除非做了策略），始终指向的是下一个 AS（通告该路由的 EBGP peer 接口 IP）。



3. 如果路由传递自 IBGP 邻居，并由 AS 内 BGP 路由器引入

- 如果是通过 aggregate-address 命令被注入的，那么 next-hop 等于执行汇总路由器（的更新源 IP）
- 如果是通过 network 或重发布注入的，那么在注入前该前缀的 IGP 下一跳将成为 BGP 的 next-hop



R1、R2、R3 跑 OSPF，R3 上的路由表如下：

```
O E2     1.1.1.0 [110/20] via 10.1.23.2, 00:06:40, Serial0/0
```

```
O       2.2.2.2 [110/65] via 10.1.23.2, 00:06:40, Serial0/0
O IA    10.1.12.0 [110/128] via 10.1.23.2, 00:06:40, Serial0/0
C       10.1.23.0 is directly connected, Serial0/0
C       10.1.34.0 is directly connected, Serial0/1
```

R3 与 R4 建立 IBGP 邻居关系，并且 R3 上，将 OSPF 路由重发布进 BGP

```
redistribute ospf 1 match internal external 1 external 2
```

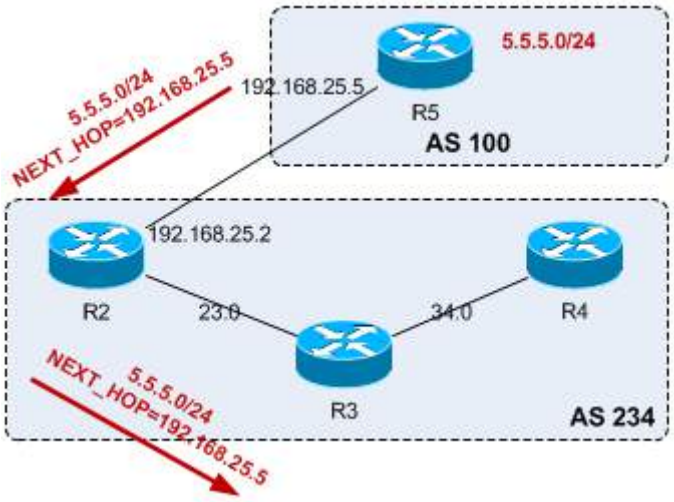
然后在 R4 上观察 BGP 表：

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i1.1.1.0/24       10.1.23.2                20    100      0 ?
*>i2.2.2.2/32       10.1.23.2                65    100      0 ?
*>i10.1.12.0/24     10.1.23.2               128    100      0 ?
*>i10.1.23.0/24     10.1.34.3                0     100      0 ?
```

我们发现，无论是 O、O IA 还是带有 FA 的 OE2 路由，被重发布进 BGP 后，这些 BGP 路由的 Next-hop 均为它们在 OSPF 中的 IGP 下一跳。同时本地激活了 OSPF 了直连接口所在网段也会被注入 BGP，注入后 Next_Hop 为本路由器(BGP 更新源 IP)。

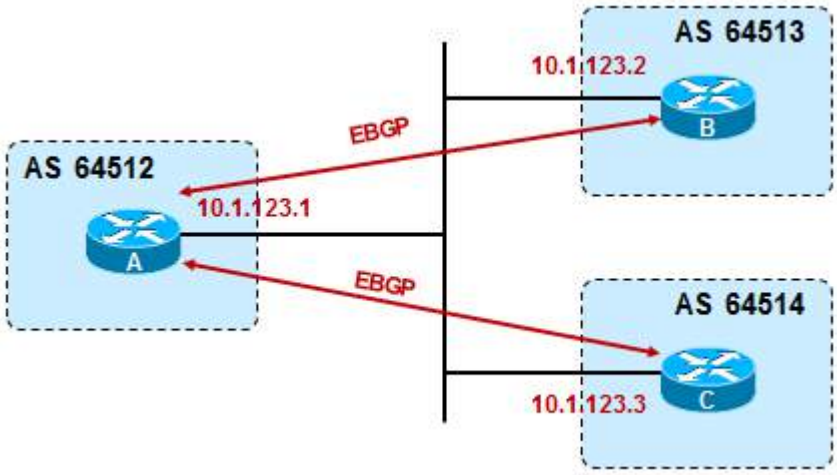
- 如果本地的 BGP 宣告者成了下一跳地址，那么在本地 BGP RIB 中的下一跳字段就是 0.0.0.0

4. 通过 next-hop-self 可以变更 next-hop 属性



上图 R5 更新给 EBGP 邻居 R2 BGP 路由 5.5.5.0 时，NEXT_HOP=192.168.25.5，该属性将一直跟随本条 BGP 路由在 AS234 中传递，这时对于 R4、R3 来说，并不知道如何前往 192.168.25.5，因此 5.5.5.0 的路由无法正常装入路由表。解决方法之一是 R2 在 IGP 路由中注入 25.0 这条外部路由，另一个方法是 R2 对其 IBGP 邻居使用 next-hop-self，修改这条前缀的 NEXT_HOP 属性。

NEXT_HOP on shared Media（在共享介质上的运作）



RouterB 将路由 100.100.100.0/24 传递给 A，NEXT_HOP 为 10.1.123.2；

RouterA 将路由 100.100.100.0/24 传递给 C，此时 NEXT_HOP 保持不变；

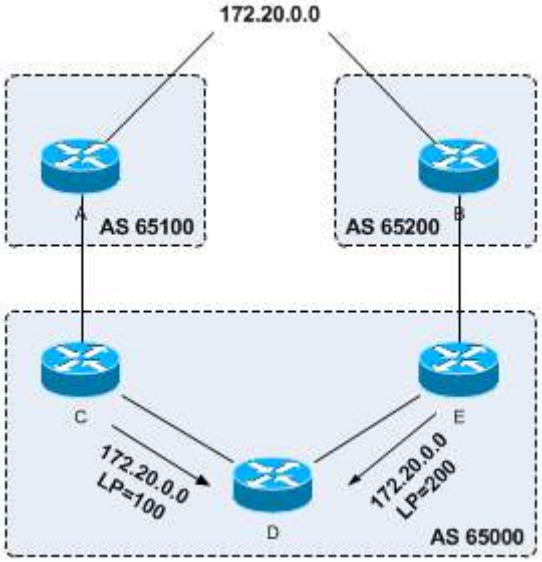
如果路由器收到某条 BGP 路由，该路由的 NEXT_HOP 地址值与该路由器的接口 IP（更新源）同属一个网段，那么该条路由的 NEXT_HOP 地址将保持不变并传递给它的（这个相同网段上的）BGP 邻居，这个其实是一种优化机制，但是这种机制在 NBMA 环境中是否有问题呢？

NEXT_HOP on NMBA network

仍然看上图，中间的网络如果不是广播多路访问网络，而是一个帧中继网络，那么就要注意，C 收到的路由，NEXT_HOP 为 10.1.123.2，那么如果 C 路由器上没有到该 IP 的 PVC，就会出问题，所以这点要考虑进去。

3.4 LOCAL-PREFERENCE

公认自决属性。LP 就是本地优先级，只能在 AS 内部，在 IBGP 对等体之间传递，而不会传递给其他 EBGP 邻居



当一个 AS 收到一个去往同一目的地的、但经过两个 AS 的路由，则根据两条路由的 local-pref 值来决定

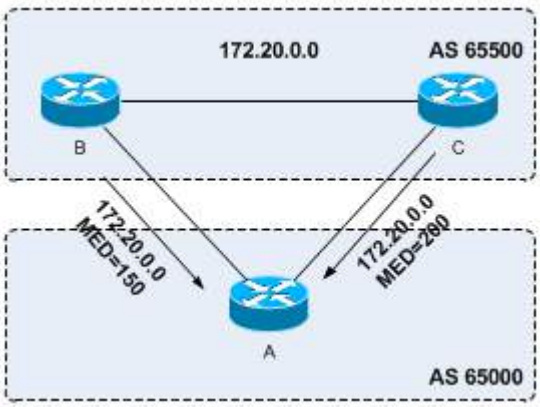
Local-pref 只影响离开 AS 的业务量（**选大的 LP 值**）

一般做在 AS 边界，告诉自己 AS 内部从自己走的 local-pref 值

1. 只能在 IBGP Peer 之间传递(除非做了策略否则 LP 值在 AS 内的 IBGP 邻居间传递不会丢失),不能在 EBGP Peer 之间传递，如果在 EBGP Peer 之间收到的路由的路径属性中携带了 Local Preference，则会触发 Notifacation 报文,造成会话中断；但是可以再 AS 边界路由器上使用 IN 方向的策略。
2. bgp default local-preference 500 //修改默认 LP 值
3. BGP 路由器在向其 EBGP 邻居发送路由更新时 ,不能携带 LP 属性 ,对方收到该 EBGP 路由的 LP 值为空(连 LP 这个字段都木有), 但是它会在本地为这条路由赋一个默认值，也就是 100，然后再传递给自己的 IBGP
4. 本地 network 及重发布的路由，LP 默认 100，并能在 AS 内向其他 IBGP 邻居传输，传输过程中除非部署策略，否则 LP 不变

3.5 MED

可选非传递，一般用于 AS 之间来影响路由。



在 B 和 C 上对 A 做策略，即在 AS 边界对外部 AS 做 MED，使得 A 选择**更低**的 MED 值。

CISCO 默认 MED 为 0

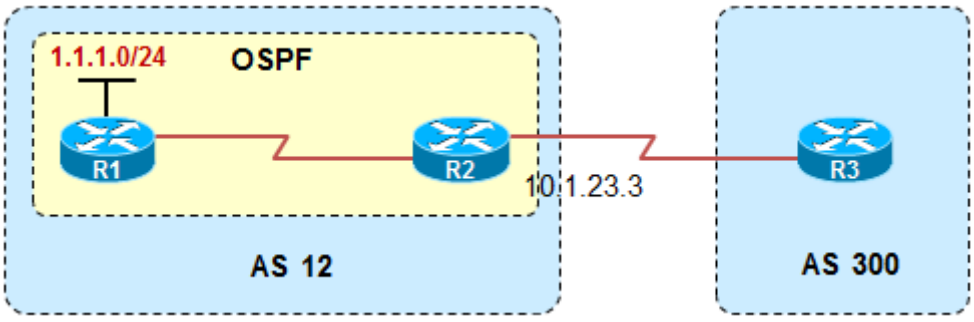
默认情况下，只比较来自同一邻居 AS 的 BGP 路由的 MED 值，就是说如果同一个目的地的两条路由来自不同的 AS，则不进行 MED 值的比较。换句话说，默认对于多条路径，只有在 AS_SEQUENCE 中的第一个 AS 相同的情况下，才比较 MED；任何打头的 AS_CONFED_SEQUENCE 都将被忽略。MED 只是在直接相连的自治系统间影响业务量，而不会跨 AS 传递

1. **MED 设置方法:**
   1) 将 IGP 路由引入 BGP 时关联 Route-map 进行设置
   2) 对 BGP Peer 应用 IN/OUT 方向的 Route-map 进行设置
   3) 非 Route-map(自动)方式:
      - 使用 network 或 redistribute 方式将 IGP 路由引入 BGP 时,MED 将继承 IGP 路由的 Metric(直联路由及静态路由的 Metric 为 0)
      - 使用 aggregate-address 方式引入路由，则 MED 为空

2. **比较原则及配置注意事项**
   1) 本地在将一条 BGP 路由通告给 EBGP Peer 时 ,是否携带 MED 值,需要根据以下条件进行判断(不对 EBGP Peer 使用 Route-map 的情况下)：
      - 如果该 BGP 路由是本地始发(network 或 redistribute)的,则携带 MED 值发送给 EBGP Peer (如果 MED 为空,则设置为 0)
      - 如果该 BGP 路由是从其他 BGP Peer 学习过来的 ,那么将该路由通告给 EBGP Peer 时不携带 MED（为空），换句话说，就是 MED 不会被传出本 AS
   2) 本地在将一条 BGP 路由通告给 IBGP Peer 时，一定会携带 MED 值
      - 如果接收或产生的路由的 MED 为空,那么在向 IBGP Peer 通告时,将 MED 设置为 0

总结 1) 2) 两点就是 MED 在 IBGP 之间传递没有问题（不会丢失），但是在 EBGP 之间传递要看路由是否起源于自己。

3. **MED 继承 IGP 的 Metric**



R1、R2 运行 OSPF，R2 学习到 1.1.1.0/24 的 OSPF 路由（metric 为 65）

在 R2 上 BGP 路由选择进程里 network 1.1.1.0 mask 255.255.255.0

则 R2 本地引入 1.1.1.0 的 BGP 路由，并继承从 OSPF 学习到的 1.1.1.0 的 metric 2，那么在 R3 上

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 1.1.1.1/32	10.1.23.2	**65**		0	12 i

所以在将 IGP 路由 network 进 BGP 时，MED 值继承 IGP 协议中的 metric 值

总结如下：

network 本地从 IGP 路由协议学习到的路由进 BGP，MED 值继承 IGP 协议中的 metric

network 本地直连接口的网段进 BGP，MED 值为 0；network 本地静态路由进 BGP，MED 值为 0

redistribute 本地从 IGP 路由协议学习到的路由进 BGP，MED 值继承 IGP 协议中的 metric

redistribute 本地直连接口网段进 BGP，MED 值为 0；redistribute 本地静态路由进 BGP，MED 值为 0

4. 其他配置命令

1) bgp always-compare-med

默认情况下只比较来自同一 AS 的路由的 med ,如果想对于所有路径都比较 MED ,则要配置此命令。同时如果使用了这个选项，那么建议在整个 AS 中都这么做，以避免路由选择环路。

2) bgp bestpath med missing-as-worst

如果某路由 MED 属性丢失一般的做法是给 MED 默认设置为 0，但是如果配置了这条命令，则如果收到没有 MED 值的路由，将该路由的 MED 设置为最大值 4294,9672,95

3) set metric-type internal

用在 route-map 中，当匹配某些条目，且使用该 set 命令后，将 route-map 应用在某个邻居（如 out 方向），则向给邻居更新这些路由时，BGP 的 MED 属性将会继承这些路由在本地 IGP 的 metric 值

4) bgp bestpath med confed

配置这条命令，在选路时，路由器只比较所有带有 AS_CONFED_SEQ 属性的路由条目，此命令用于联邦路由器，同时 weight 和 LP 比 MED 具有更高的优先级

5) bgp deterministic-med

用于确保来自相同的 AS 的不同对等体通告的路由**先**进行 MED 比较

此命令和 bgp always-compare-med 的关系如下：

如有相同前缀的路由如下（默认按路由接受顺序的逆序排序）

路由 1： AS_PATH：500	MED：150	external	RouterID：172.16.13.1
路由 2： AS_PATH：100	MED：200	external	RouterID：1.1.1.1
路由 3： AS_PATH：500	MED：100	Internal	RouterID：172.16.8.1

- 如果两条命令都关闭：

则按缺省的从上往下的顺序进行比较

则 1、2 先比较，2 优选（因为 2 具有更小的 RID）；然后 2 与 3 比较，2 为 external，所以 2 优选

- 如果打开 always-compare-med

  则 1、2 先比较，1 优选（因为 1 具有更小的 MED）；由于 1、3 在一个相同的 AS 中，所以再次比 MED，选 3

- 如果 deterministic-med 打开，路由前缀的信息按 AS 重新分组

  路由 1： AS_PATH：100　　MED：200　　external　　RouterID：1.1.1.1

  路由 2： AS_PATH：500　　MED：100　　Internal　　RouterID：172.16.8.1

  路由 3： AS_PATH：500　　MED：150　　external　　RouterID：172.16.13.1

  则 2、3 同 AS 内先比较，2 的 MED 小所以优选；2 与 1 再比较，1 为 external 路由所以被优选（MED 不比较）

- 如果都打开

  则 2、3 同 AS 内先比较，2 的 MED 小所以优选；2 与 1 再比较，由于开了 always-compare-med 所以比 MED，2 的 MED 小，所以 2 为最佳路径

6) Default-metric x

修改 MED 值的缺省值

对从其他动态路由选择域重发布进来的路由生效，对直连路由不生效（仍然为 0）

3.6 COMMUNITY

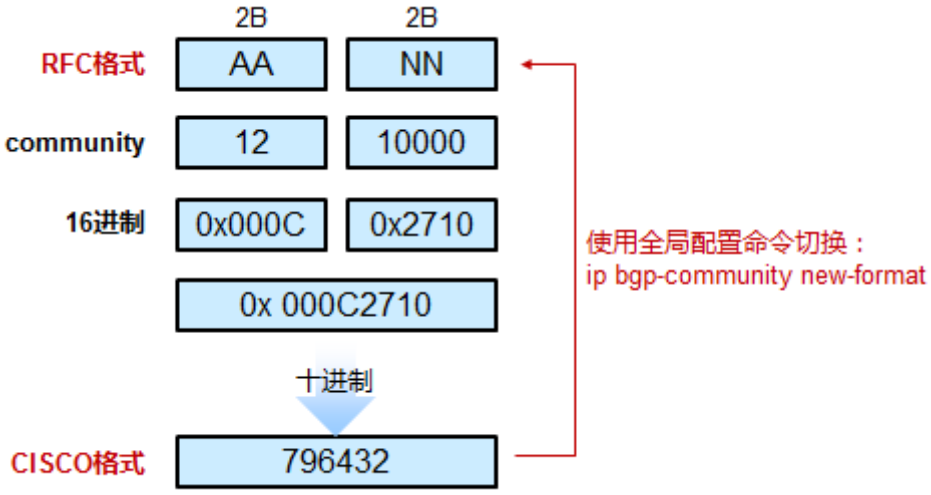
1. 基本概念

可选传递，用于简化路由策略的执行

可以将某些路由分配一个特定的 COMMUNITY 属性，之后就可以基于 COMMUNITY 值而不是每条路由进行 BGP 属性的设置了。COMMUNITY 属性对邻居起作用，在设置后，同时需要向邻居发送（SEND）

COMMUNITY 属性是一组 4 个 8 位组的数值，RFC1997 规定前 2B 表示 AS 号，后 2B 表示基于管理目的设置的标示符，格式为 AA：NN，而 CISCO 默认显示格式为 NN：AA，可使用全局配置命令 ip bgpcommunity new-format 将 CISCO 默认格式改为 RFC 格式。

例如将 AS12 的某条路由 COMM 值改为 10000，RFC 采用十六进制表示 COMMUNITY 属性，而 CISCO 采用十进制。RFC 格式为 12：10000，十六进制为 0x 000C2710，再转换为十进制 796432




```
COMMUNITIES: 12:10000 (7 bytes)
  Flags: 0xc0 (Optional, Transitive, Complete)
    Type code: COMMUNITIES (8)
    Length: 4 bytes
  Communities: 12:10000
    Community: 12:10000
        Community AS: 12
        Community value: 10000
```

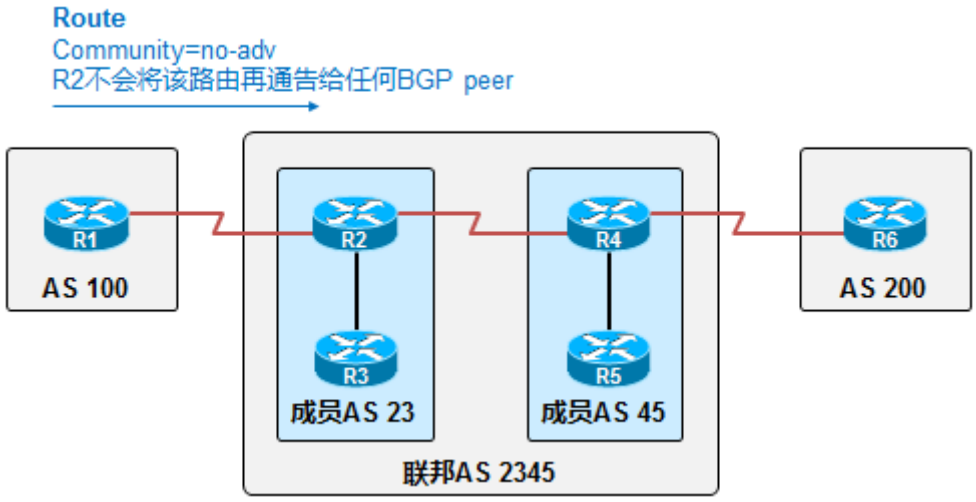
0-65535（0x00000000-0x0000FFFF）及 4294901760-429496729502710046（0xFFFF0000-0xFFFFFFFF）是保留的团体值在这些值之外还定义了几个众所周知的值如 no-export、local-as 等，具体值请见 RFC。

2. 在 route-map 中 set community

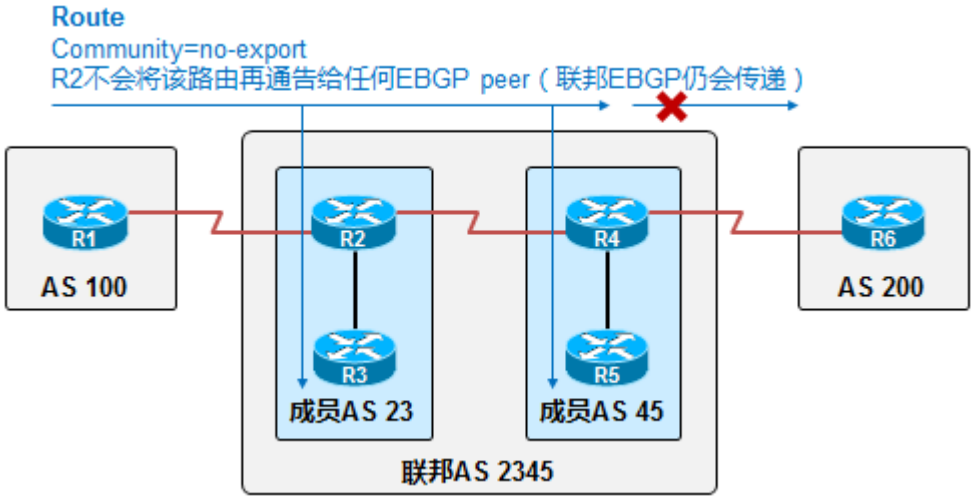
```
route-map test permit 10
  set community ?
    <1-4294967295>      community number
    aa:nn               community number in aa:nn format
    additive            Add to the existing community                设置 commu 值为附加，否则为覆盖
    internet            Internet (well-known community)              默认所有路由都属于该团体
    local-AS            Do not send outside local AS (well-known community)
    no-advertise        Do not advertise to any peer (well-known community)
    no-export           Do not export to next AS (well-known community)
    none                No community attribute
```

下边我们来看一下 community 的这几个众所周知值的作用：

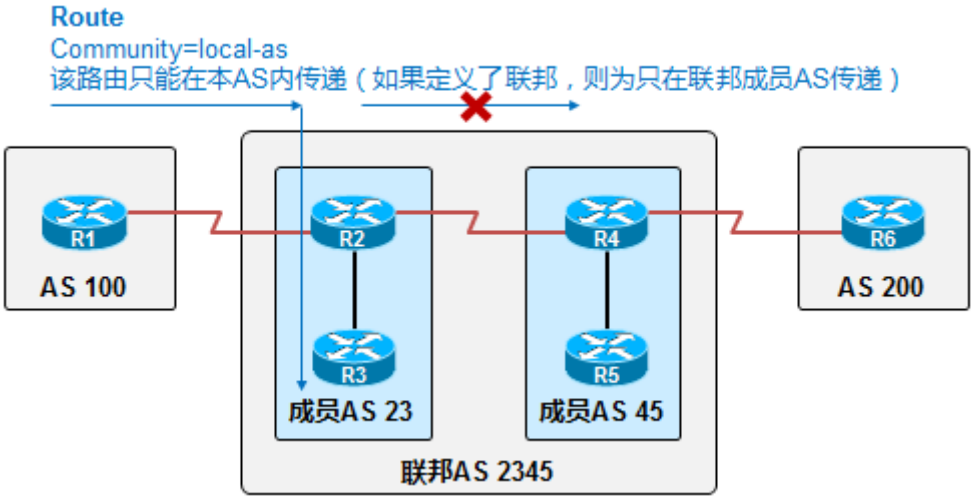
- **no-advertise**



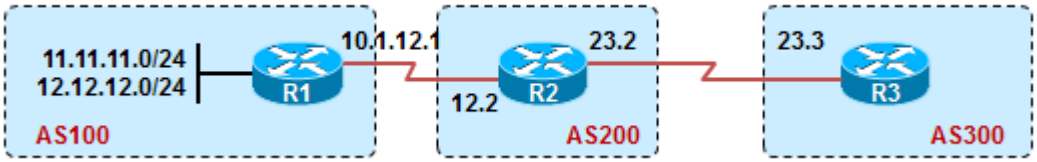
- **no-export**



- **local-as**



配置示例（为路由分配 community）：



在 R1 上为路由 11.11.11.0/24 分配 community 100:11，并且传递给 R2，那么 R1 上配置如下：

```
ip prefix-list 11 permit 11.11.11.0/24
route-map test permit 10
  match ip address prefix-list 11
  set community 100:11
router bgp 100
  network 11.11.11.0 mask 255.255.255.0
  neighbor 10.1.12.2 remote-as 200
  neighbor 10.1.12.2 send-community          // 默认 community 不发送，因此必须配置该命令
  neighbor 10.1.12.2 route-map test out
```

注意 community 默认不发送，必须 send-community。

另外一条路由前缀可以携带多个 community 形成一个列表 ,如果要针对特定路由在原有的 community 基础之上再增加一个 community，则在 route-map 中 set community 时，增加 additive 关键字。

3. 使用 ip community-list 匹配 community 值

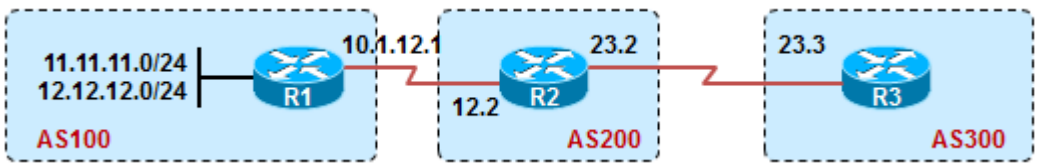
- **ip community-list ?**

```
  <1-99>        Community list number (standard)
  <100-500>     Community list number (expanded)
  expanded      Add an expanded community-list entry
  standard      Add a standard community-list entry
```

ip community-list 也像 ACL 那样，有标准和扩展之分，1-99 为标准，100-199 为扩展。扩展的 community-list 可以使用正则表达式匹配路由。

这里有一点需要注意，ip bgp-community new-format 用于转换 community 在 CISCO IOS 中的显示格式，当在扩展的 community-list 列表中使用正则表达式的过滤结果会由于格式的选择不同而不同。

- **ip community-list 示例 1：**



在上面实验的基础上，R1 传递给 R2 的 11.11.11.0/24 路由，携带了 community 值 100:11，这个值可以在 R2 上使用 ip community-list 进行匹配，从而可以进一步在 route-map 中用这个 community-list 去设置策略。我们现在在 R2 上用 community-list 去匹配 100:11，通知添加一个 no-export 的 community 到该路由。

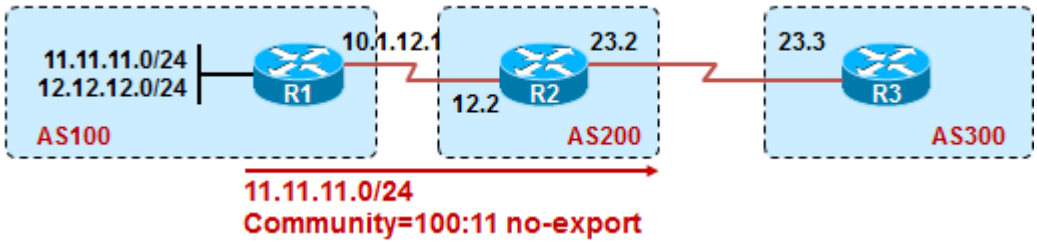
R2 的配置如下：

```
ip community-list 11 permit 100:11
route-map test permit 10
  match community 11
  set community no-export additive
router bgp 200
  neighbor 10.1.12.1 remote-as 100
  neighbor 10.1.23.3 remote-as 300
  neighbor 10.1.23.3 send-community
  neighbor 10.1.23.3 route-map test out
```

R3 上 show ip bgp 11.11.11.0

```
BGP routing table entry for 11.11.11.0/24, version 5
Paths: (1 available, best #1, table Default-IP-Routing-Table, not advertised to EBGP peer)
Flag: 0x820
  Not advertised to any peer
  200 100
    10.1.23.2 from 10.1.23.2 (10.1.23.2)
      Origin IGP, localpref 100, valid, external, best
      Community: 100:11 no-export
```

- **ip community-list 的示例 2（逻辑关系）：**



当 11.11.11.0/24 路由前缀携带的 community 属性为“100:11 no-export”，我们做如下测试：

```
Ip community-list 11 permit 100:11
```

匹配。这种写法将匹配 community 中包含 100:11 的路由。

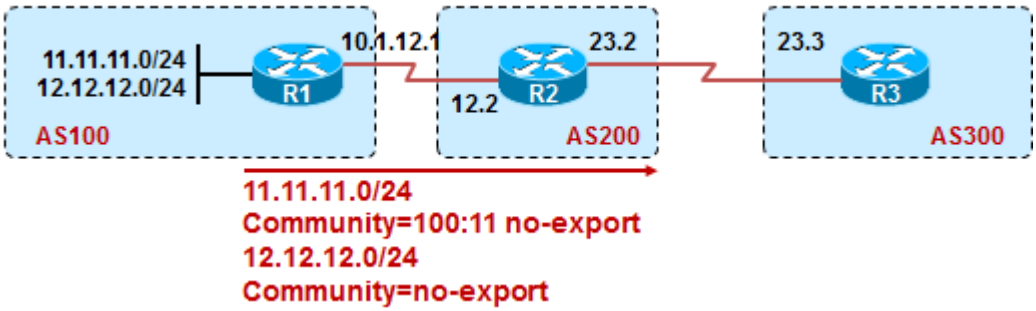
```
Ip community-list 11 permit 100:11 no-advertise
```

不匹配。这种写法要求 community 中同时包含 100:11 及 no-advertise 才匹配成立。

```
Ip community-list 11 permit 100:11
Ip community-list 11 permit no-export （或将 no-export 换成 no-adv）
```

匹配。只要 community 中包含 100:11 或 no-export

- **ip community-list 的示例 3（严格匹配 community）：**



注意我们实验环境的变化，12.12.12.0 携带的 community 值为 no-export，如果我们只希望匹配 no-export community 值（的路由），那么怎么写呢？如果是直接去匹配 no-export，会连同 11.11.11.0 也一并匹配上了，所以就要使用到 exact-match 关键字了。在 R3 上如果配置如下：

```
Ip community-list 11 permit no-export
route-map test permit 10
  match community 11 exact-match      // 严格匹配
```

如果不加 exact-match 关键字，则该 community-list 将匹配 11 及 12 路由，加了之后，则只匹配 community 为 no-export 的路由，不能多，不能少。

4. 在 community 列表中删除特定的 community 值

前面已经说了，一条路由，允许携带多个 community 值，构成一个 community 列表

那么如果想删除某个或者某几个 community 值，例如

12:11　12:1111　no-export　这个 comm 列表，要删除其中的 no-export，则可

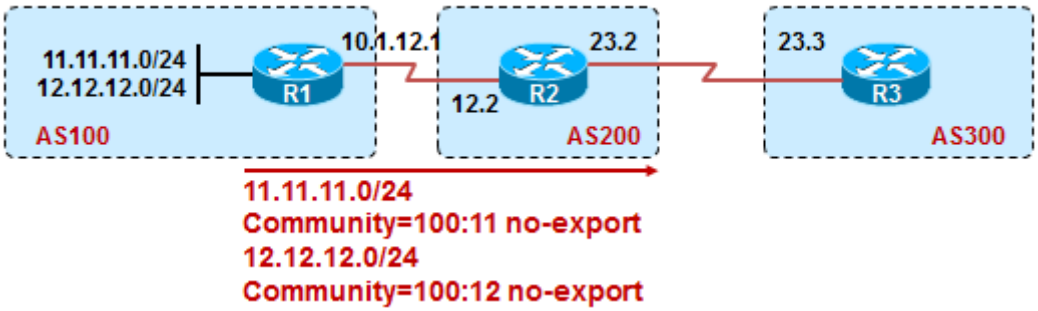
```
ip community-list 1 permit no-export          // 匹配要删除的 commu 值
route-map test permit 10
 set comm-list 1 delete                       // 用这条命令删除
```

如果要删除多个 community（而不是所有），则可在一个 community-list 中写多条，如

ip community-list 1 permit no-export

ip community-list 1 permit 12:1111

然后再用 set comm-list 1 delete 去删除，注意上面 community-list 的写法，要用多行，实验验证的结果是在同一行写多个 commu 如 ip community-list 1 permit no-export 12:1111 则不生效，删除不了这两值



R3 收到两条路由，分别携带的 community 属性如上，现在，我们只想删除 11 路由的 no-export 属性。

```
ip community-list 11 permit 100:11 no-export            //这条用来匹配 11 路由
ip community-list standard del permit no-export         //这条用来删除 no-export 属性，是个命令列表
route-map test permit 10
  match community 11
  set metric 1111
  set comm-list del delete
route-map test permit 20

router bgp 300
  neighbor 10.1.23.2 remote-as 200
  neighbor 10.1.23.2 route-map test in
```

如果想把 11 路由的 100:11 及 no-export 两个 community 值都删除，则修改 del 这条 community-list 即可：

ip community-list standard del permit no-export

ip community-list standard del permit 100:11

5. COST community ：

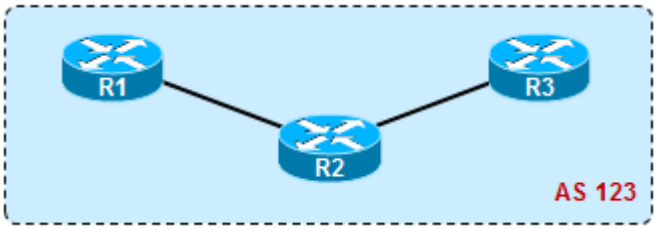
使用 set excommunity cost x 在 route-map 去设置

可以影响 BGP 选路（在等价负载均衡的前一个），有点类似于“tie breaker”在路径选择的时候，就是在纠结的时候打破纠结

可以针对路由设置 cost，cost-ID 相等的时候，优选 cost 小的。如果 cost 相等，则选择 cost-ID 小的

set excommunity cost pre-bestpath 1 20 ，则 cost 值的将极大的影响选路原则，将会成为超越 weight 值的第一条规则

注意，这个时候在 send-community 的时候要加上扩展 extend 关键字



R1、R2、R3 使用 LOOPBACK 建立邻居关系，地址分别为 1.1.1.1、2.2.2.2、3.3.3.3，运行 OSPF 使得 AS 内部 LOOPBACK 间路由能各自学习到。R1、R3 均向 R2 更新 100 网段的路由，在不做任何策略的前提下，BGP 的选路规则会一直比到 RID，最终优选 R1 的路由。

```
R2#sh ip b 100.100.100.0
BGP routing table entry for 100.100.100.0/24, version 2
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    3.3.3.3 (metric 65) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, internal
  Local
    1.1.1.1 (metric 65) from 1.1.1.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

这时候就优选了 R1，如果希望让 R2 优选 R3 呢？R2 增加配置如下：

```
ip prefix-list 100 permit 100.100.100.0/24
route-map COST1 permit 10
  match ip address prefix-list 100
  set extcommunity cost 1 10
route-map COST2 permit 10
  match ip address prefix-list 100
  set extcommunity cost 1 5
router bgp 123
  neighbor 1.1.1.1 route-map COST1 in
  neighbor 3.3.3.3 route-map COST2 in
```

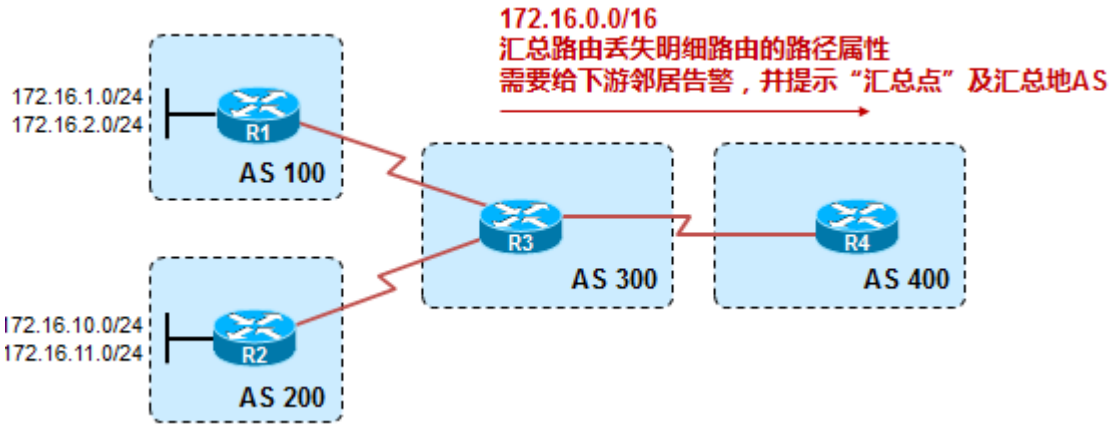
如果按以下配置，那么 COST 的比较，将会超越 weight，成为最优选比较的规则

```
route-map COST1 permit 10
  match ip address prefix-list 100
```

```
set extcommunity cost pre-bestpath 1 10
```

3.7 Atomic_Aggregate 及 aggregator

Atomic_Aggregate 是公认自决属性；Aggregator 是可选可传递。



我们看上图，R1、R2 均发布了路由，R3 上对这些明细路由进行汇总，汇总成 172.16.0.0/16，汇总路由被 R3 传递给了其他 BGP 邻居，然而，这条汇总路由丢失了底下明细路由的所有属性，其中属 AS_PATH 最关键，因为一旦丢失了明细路由的 AS_PATH 属性，这条汇总路由就极有可能产生路由环路。因此有必要让 R3 警告下游 BGP peer，告诉他们 1、这是条汇总路由，2、汇总路由发生的地方，也就是汇总路由的始发 AS 和始发路由器。

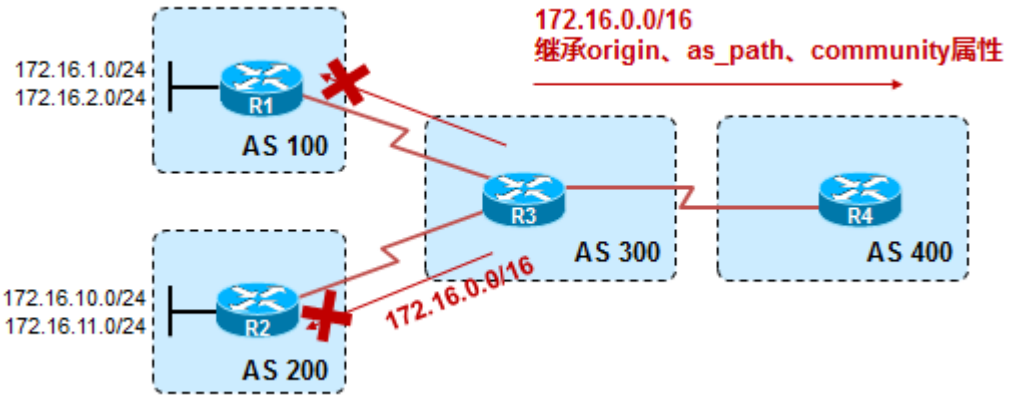
R4#show ip bgp 172.16.0.0

```
BGP routing table entry for 172.16.0.0/16, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  300, (aggregated by 300 3.3.3.3)          // 汇总路由产生自 AS300，由 BGP router 3.3.3.3 产生
    10.1.34.3 from 10.1.34.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
```

抓包如下：

```
UPDATE Message
    Marker: 16 bytes
    Length: 63 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 37 bytes
  Path attributes
    ORIGIN: IGP (4 bytes)
    AS_PATH: 300 (7 bytes)
    NEXT_HOP: 10.1.34.3 (7 bytes)
    MULTI_EXIT_DISC: 0 (7 bytes)
    ATOMIC_AGGREGATE (3 bytes)
      Flags: 0x40 (Well-known, Transitive, Complete)
      Type code: ATOMIC_AGGREGATE (6)
      Length: 0 bytes
    AGGREGATOR: AS: 300 origin: 3.3.3.3 (9 bytes)
      Flags: 0xc0 (Optional, Transitive, Complete)
      Type code: AGGREGATOR (7)
      Length: 6 bytes
      Aggregator AS: 300
      Aggregator origin: 3.3.3.3 (3.3.3.3)
  Network layer reachability information: 3 bytes
    172.16.0.0/16
```

注意这里 atomic_aggregate 为公认自由决定属性，我们来理解一下：



在 R3 上做汇总，使用 aggregate-address summary-only as-set，注意加上了 as-set 关键字

这样一来，汇总路由就继承了明细的 AS_PATH 属性，从而 atomic_aggregate 就显得多余了，因此，上面这种情况下，产生的汇总路由就不携带 atomic_aggregate 了。我们看看：

R4 上 show ip bgp 172.16.0.0

```
BGP routing table entry for 172.16.0.0/16, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  300 {100,200}, (aggregated by 300 3.3.3.3)
    10.1.34.3 from 10.1.34.3 (3.3.3.3)
```

Origin IGP, metric 0, localpref 100, valid, external, best

的确没有 atomic_aggregate。

3.8 ORIGINATOR_ID 和 CLUSTER_LIST

由于 AS_PATH 属性在 AS 内部不会发生变化（仅当路由离开本 AS 才会被更新），因此 AS 内防环才有水平分割的机制，而路由反射器实际上是放宽了水平分割原则，这个就会给环路带来一定的隐患，因此路由反射器需使用以下两个属性防止环路：

ORIGINATOR_ID 和 **CLUSTER_LIST** 是路由反射器使用的可选非传递属性，用来防止环路。

ORIGINATOR_ID 是一个路由反射器创建的 32bit 值，该数值是**本地 AS 中路由发起方的 IBGP routerID**，注意，这个发起方未必是这条路由的引入者（下面有实验验证），如果发起方发现其 RID 在所接收到的路由的 ORIGINATOR_ID 中，那么就知道已经出现了路由环路，因此忽略该路由。

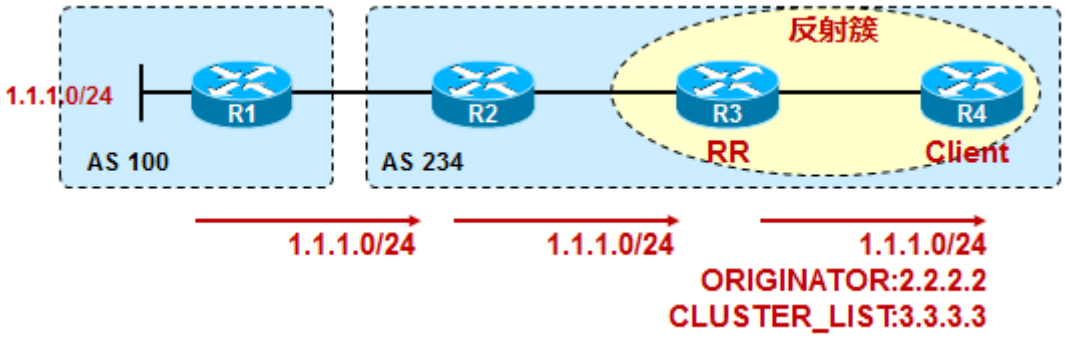
CLUSTER_LIST 是一串路由传递所经过的路由反射簇（cluster）的 ID， AS 内的每个路由反射簇都有一个 32bit 的簇 ID，如果簇中包含了多个 RR，则需手工为每个 RR 配置簇 ID。

当 RR 将来自客户的路由反射给非客户时，同时将其簇 ID 附加到 CLUSTER_LIST 中，如果 CLUSTER_LIST 为空，则创建一个。如果路由反射器发现其本地簇 ID 出现在了其收到的路由的 CLUSTER_LIST 中，那么就知道出现了环路，则忽略该路由。CLUSTER_LIST 属性只用于 RR 防环。

注意：RR 只在反射路由的时候才创建或更新 CLUSTER_LIST，而下面几种情况，RR 不会创建该属性：

- RR 自己始发的路由
- 当 RR 向 EBGP 邻居发送路由更新时，将会清楚已有的 CLUSTER_LIST 属性
- 当 RR 从 EBGP 邻居收到路由，传递给 client 或非 client，不会创建 CLUSTER_LIST。

- **验证实验 1 ORIGINATOR_ID 的取值**

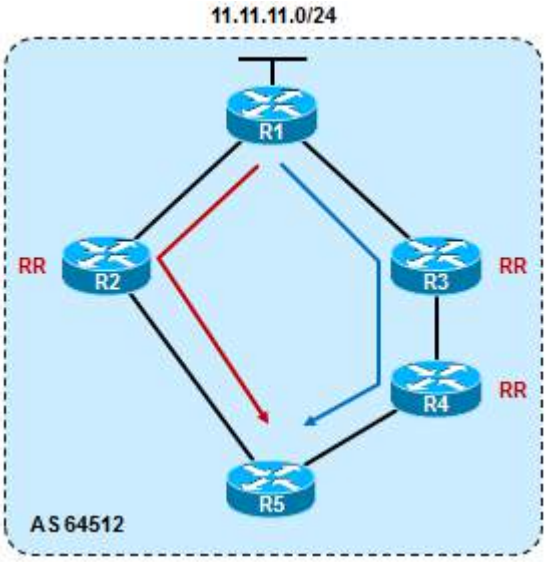


R1 处于 AS100，将 1.1.1.0/24 的路由传递给 EBGP 邻居 R2。

IBGP 邻居关系如图所示，使用各自的 loopback 口建邻居。R2 收到路由后，继续传递给 IBGP 邻居 R3，这时候 R3 由于是 RR，因此将非 client IBGP 邻居发来的路由，传递给自己的 client R4，此时，R3 在路由中添加 ORIGINATOR 及 CLUSTER_LIST 属性，其中 ORIGINATOR 属性为 R2 的 routerID 也就是 2.2.2.2（而

不是 R1 的 routerID，因为根本没必要）。

- **验证实验 2　CLUSTER_LIST 在 BGP 选路中的影响**




```
R5#show ip bgp 11.11.11.0
BGP routing table entry for 11.11.11.0/24, version 2
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    1.1.1.1 (metric 129) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator: 1.1.1.1, Cluster list: 4.4.4.4, 3.3.3.3
  Local
    1.1.1.1 (metric 129) from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 1.1.1.1, Cluster list: 2.2.2.2
```

3.9 Weight

CISCO 私有，作用范围是本路由器（不传递），该值不既不会被包含在 update 消息中，也不会传递给任何 BGP 邻居，只在路由器本地产生影响，可以理解为权重值，越大越优先。

取值是范围 0-65535。

- 如果路由是从其他邻居学过来的，则默认值（在本地该路由）是 0
- 本地 network 产生的路由 weight 是 32768

本地重发布的直连接口路由、静态路由的 weight 为 32768

本地汇总产生的 BGP 路由 weight 值也为 32768

4 BGP 配置

4.1 建立 BGP 邻居

1. **router bgp** *asnumber*

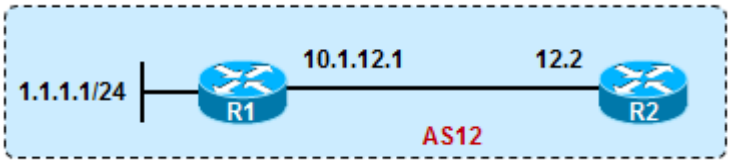
AS 号为 2B，但是随着网络发展，2B 不够用了，所以现在扩充到了 4 个 B router bgp ?可以查看（注：仅在特定 IOS 版本下支持）

2. **network** xxx **mask** yyy **route-map** zzz

在 IGP 中 network 命令用于确定要发送和接收路由更新的接口，以及通告哪些直连的网络。

而在 BGP 中，network 命令与 BGP 在哪些接口上运行无关。仅仅是将本地某个网路注入到 BGP 里。该路由我们称为可靠的。只有该路由存在于路由器的路由表中（非 B），在 BGP 下 network 该网段才有可能使得该路由条目传递给 BGP 邻居。务必注意，network 只能有效注入那些在路由表中存在的非 B 的路由。如果我本地有个 loopback，4.4.4.0/24 而我 BGP 中 network 4.0.0.0，那么宣告不成功（BGP 默认不自动汇总，当然注意 IOS 版本）;

network 不加掩码就是有类宣告，加上掩码就是无类宣告



- 若 R1 关闭 auto-summary，且 network 1.1.1.0 mask 255.255.255.0 ,则 R2 能学到 BGP 路由 1.1.1.0/24
- 若 R1 关闭 auto-summary，且 network 1.1.1.0，则宣告不成功，因为不加 mask 为有类宣告
- 若 R1 **开启** auto-summary，且 network 1.1.1.0，则宣告不成功
- 若 R1 **开启** auto-summary，且 network 1.0.0.0，则 R2 能学习到汇总路由 1.0.0.0/8

3. **neighbor** *ip* **remote-as** *as-number*

neighbor 的这个地址，必须是 IP 可达的

4. **多跳 EBGP**

EBGP 邻居通常必须是直连的才能建立起 EBGP 会话。但

RA(as200)------RB(as100)--------RC(as100)

RC 和 RA 之间隔着 RB ,在这种情况下 ,EBGP 通过使用可以配置的 “多跳 EBGP”的特性穿过一台非 BGP 路由器运行。也就是说 RB 可以不支持 BGP，配置如下（RA）

```
router bgp 200
neighbor RC-ip remote-as 100
neighbor RC-ip ebgp-multihop   2      //他们之间跳数为 2
```

另，如果两个 EBGP 之间有多条链路，并使用各自的 Loopback 端口作为 source，那么在配置的时候也要注意。ebgp-multihop 2 （因为 loopback 跨越了两台路由器，其中包括自己这台),并且还必须指静态路由到对方的 loopback 接口。）

4.2 路由重发布

1. **BGP 重发布到 IGP**

   将 BGP 路由重发布到 IGP 时，默认不重发布 iBGP 路由，只重发布 EBGP 路由，

   需使用 bgp redistribute-internal 来让 iBGP 路由顺利被重发布

   注意，在 MPLS VPN 环境中的 PE 没有这个限制

2. **OSPF 重发布进 BGP**

```
router bgp x
  redistribute ospf 1
```

默认情况下，只会将 OSPF 中 O 及 O IA 的路由重发布进 BGP

```
redistribute ospf 1 match external 1 external 2
```

以上命令只重发布 OSPF 外部路由 E1、E2 进 BGP

```
redistribute ospf 1 match internal external 1
```

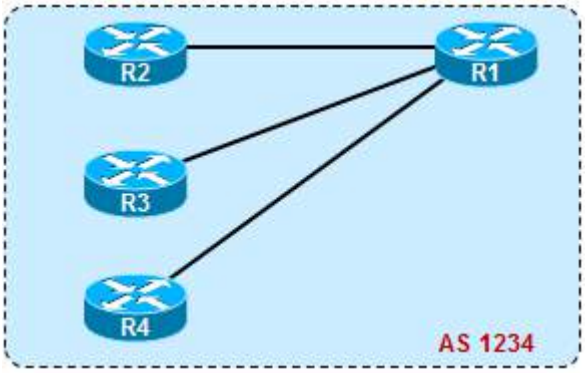
以上命令只重发布 OSPF 内部路由及 E1 路由进 BGP

```
redistribute ospf 1 match nssa-external 1 nssa-external 2
```

以上命令只重发布 OSPF NSSA 路由进 BGP

4.3 peer group

基本配置




```
router bgp 1234
   neighbor IBGP_peers peer-group
   neighbor IBGP_peers remote-as 1234
   neighbor IBGP_peers update-source loopback0
   neighbor IBGP_peers password cisco
   neighbor IBGP_peers send-community
!
   neighbor 2.2.2.2 peer-group IBGP_peers
   neighbor 3.3.3.3 peer-group IBGP_peers
   neighbor 4.4.4.4 peer-group IBGP_peers
```

无法对 peer-group 的 member 单独使用 out 方向的策略，只能对整个 group，in 方向是没有这个限制的。

同一个 Peer Group 中的所有邻居，必须全部为 iBGP 邻居，或者全部为 eBGP 邻居

4.4 配置查看及验证

1. **show ip bgp**

   显示 BGP 表，例如：

```
R1#sh ip b
BGP table version is 1, local router ID is 12.12.12.12
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric     LocPrf     Weight   Path
*  11.11.11.0/24    0.0.0.0                  0                 32768    i
```

   第一栏的可能取值如下：

   *　可用的路由(next-hop 可达)

s 被抑制的路由条目，例如做了路由汇总，抑制了明细

d 被惩罚（dampening）的路由，路由受到了惩罚，虽该路由当前可能正常，但惩罚期结束前不会被通告

h 被惩罚（dampening）的路由，路由可能出现了故障（down），有历史信息，但没有最佳路由

r 路由没有被装载进 RIB 表，例如由于 AD 值等原因导致

S 大写的 S，stale，表示过期的路由

第二栏 > BGP 算法选出的最优路径

第三栏 为空，或为 i。为空表示该路由从 EBGP 邻居获取，为 i 表示这是从 IBGP 学习到的路由

最后一栏 指出该 BGP 路由信息的起源

2. show ip bgp summary

```
BGP router identifier 12.12.12.12, local AS number 64512
BGP table version is 2, main routing table version 2
1 network entries using 117 bytes of memory
1 path entries using 52 bytes of memory
2/1 BGP path/bestpath attribute entries using 248 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 417 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 60 secs

Neighbor        V    AS  MsgRcvd  MsgSent   TblVer  InQ  OutQ  Up/Down  State/PfxRcd
10.1.12.2       4   200     1083     1082        2    0     0  17:57:27        0
```

BGP table version 每当 BGP 表发生变化，这个值+1

main routing table version 被注入到主路由表中的最后一个 table version

AS 邻居的 AS 号

MsgRcvd 从邻居那收到的消息数

MsgSent 发送给该邻居那的消息数

TblVer 发送给该邻居的最后一个 bgp table 的 version

inQ 来自该邻居的等待处理的消息数

outQ 队列中等待被发送到该邻居的消息数

State BGP 当前的回话状态基本的如 opensent 等，admin 表示邻居被管理型 shutdown，当处于 establish 状态时，不会显示状态提示，而显示一个表示 PfxRcd 的数字，也就是从该邻居处收到的前缀条目

3. **show ip b rib-failure**

   显示没有被加载进 RIB 中的 BGP 路由，以及没被加载的原因

4. **show ip bgp *路由前缀***

```
R5#sh ip b 30.30.30.0
BGP routing table entry for 30.30.30.0/24, version 2
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    3.3.3.3 (metric 65) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator: 3.3.3.3, Cluster list: 4.4.4.4
  Local
    3.3.3.3 (metric 65) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

5. **show tcp bri**

6. **sh ip bgp neighbors {address} received-routes**

   显示从指定邻居收到的所有 BGP 路由（本地入站策略执行前的原始路由）

7. **sh ip bgp neighbors {address} routes**

   显示从指定邻居那里收到的所有路由（上一条命令的子集，这里显示的是执行入站策略后剩余的路由）

8. **sh ip bgp neighbors {address} advertised-routes**

   显示通告给特定邻居的所有 BGP 路由

9.

4.5 邻居身份验证

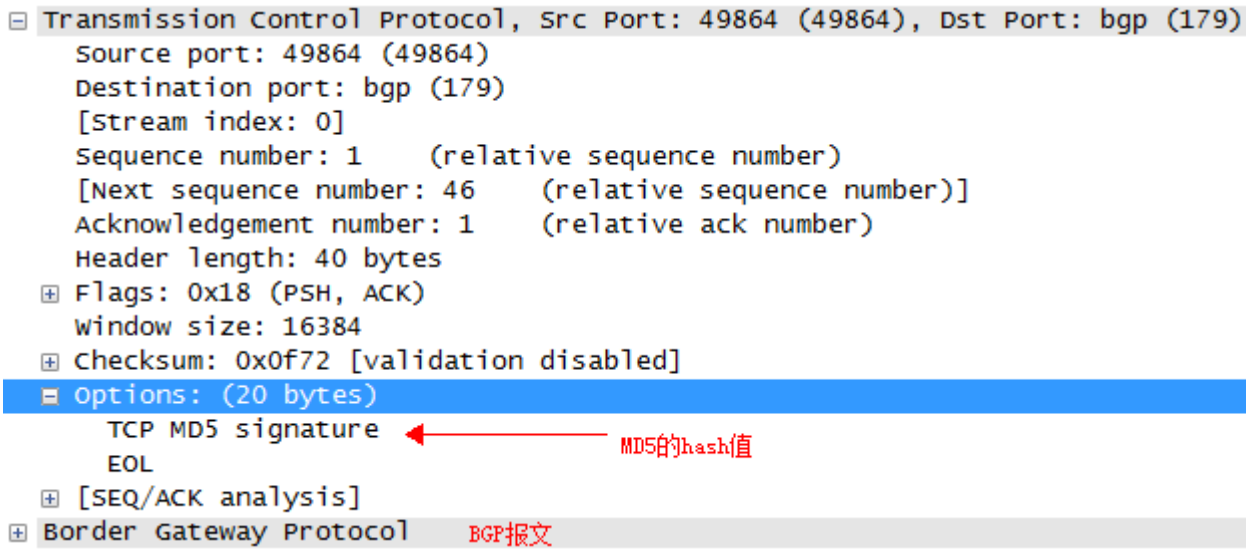
BGP 支持 MD5 身份验证，要在 BGP 对等体之间的 TCP 连接上启用 MD5 身份验证，使用命令 如下：

```
neighbor xxx password xxx
```

同一个 BGP 连接，密码必须一致；不同的邻居可设置不同的密码。配置认证后，将通过对等体之间的 TCP 连接传输的所有数据段进行验证。

以下是配置了 MD5 身份验证后报文的变化（所有 TCP 包增加了 MD5 HASH 字段）：

注意这只是基本的身份验证功能，并不是加密



4.6 邻居管理

1. **neighbor maximum-prefix xx**

   限制从邻居接受的前缀最大数 如果超出了这个数 路由器就会关闭与该邻居的 BGP 连接 ,在应用 clear ip bgp xxx 之前都不会再次建立 BGP 会话

2. **neighbor maximum-prefix xx restart 2**

   超过 xx 这个数，断开与邻居的 BGP 连接，2 分钟后才能重新连接

3. **Neighbor maximum-prefix 300 90% warning-only**

   当从邻居接受的前缀数量超过了最大数 300 的 90%时（默认 75%），生成一条日志消息

4. **重置 BGP 连接**

   - **硬重置**

     所有的 BGP 邻居的 TCPsession 及 BGP 邻居状态都重新复位

     clear ip bgp *

     clear ip bgp *{neighbor-address}*

   - **软重置**

     不拆除并重建 TCP 或 BGP 连接，而是仅触发更新操作以便让新的路由策略生效

     软重置可以仅由于出站或入站策略，也可同时用于出入站策略

     clear ip bgp soft out // 出站软重置，重新发送 update 消息给所有邻居

     clear ip bgp soft in // 入站软重置，本地发送 route-refresh 给所有 BGP 邻居

     clear ip bgp *{neighbor-address}* soft *in/out* 指定特定的邻居，建议

执行入站软重置的方法有两种：1、使用存储的路由更新信息的软重置 2、动态软重置

1> 使用存储的路由更新信息的软重置

```
neighbor x.x.x.x soft-reconfiguration inbound
```

首先在 BGP 进程中使用上述命令，这条命令会将 x.x.x.x 邻居发送过来的最原始的 BGP 路由存储在本地内存中，当配置入站软重置后（clear ip bgp soft in），路由器不再向邻居发送更新请求，而是直接在内存中存储的那些 BGP 路由中执行新配置的入站策略，以此来防止触发大批量的路由更新而造成资源的浪费，但是这种操作仍会耗费内存，因此在使用的时候要非常慎重。

2> 动态的入站软重置

CISCO IOS 12.1 开始全面支持入站路由的动态软重配置，这种软重置方法直接在本地向特定邻居发送 route-refresh 消息。

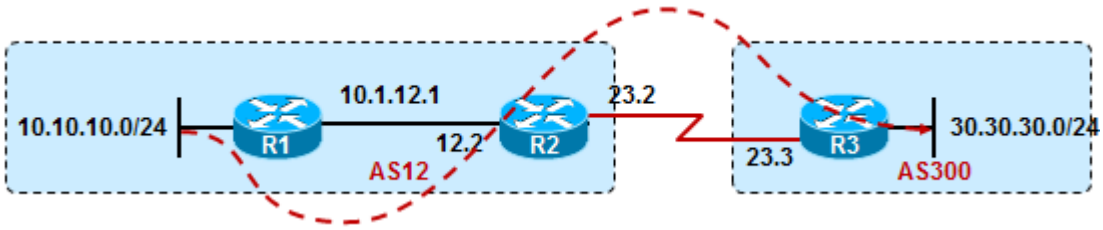
4.7 BGP Policy Accounting

1. 基本概念

BGP PA 能够对 BGP peer 发送来的 IP 流量进行度量或流量统计

Policy accounting is enabled on an input interface, and counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

2. 配置案例



配置的思路是，在 R2 上，想要抓取 fast0/0 口（也就是 R1 发过来的），去往 30.30.30.0/24 的流量

那么先在 R2 上对 30.30.30.0/24 的路由前缀进行标记，使用 community 标记，同时配置一个 route-map PA，将特定 community 标记的路由设置 traffic-index。配置如下：

```
ip prefix-list 30 permit 30.30.30.0/24
route-map setComm permit 10
  match ip address prefix-list 30
  set community 300:30
ip community-list 30 permit 300:30
route-map PA permit 10
 match community 30
 set traffic-index 30
```

```
router bgp 12
  table-map PA
  neighbor 10.1.23.3 route-map setComm in

interface fast0/0
 ip address 10.1.12.2 255.255.255.0
 bgp-policy accounting input
```

接下去在 R1 上向 30.30.30.0/24 去 ping 10 个 ICMP 包，那么就可以在 R2 上看到相应的流量：

```
R2#sh cef interface s0/0 policy-statistics input
Serial0/0 is up (if_number 4)
  Corresponding hwidb fast_if_number 4
  Corresponding hwidb firstsw->if_number 4
  BGP policy accounting on input is enabled.
  Index          Packets          Bytes
  …              …                …
  30             10               1000
```

如果嫌输出太多，可以用管道符：sh cef interface s0/0 policy-statistics input | in 30

5 BGP 决策

5.1 决策概述

BGP 路由信息数据库 RIB，包含三个部分：

ADJ-RIBs-IN　存储来自对等体的、未经处理的消息。所含的路由都是可用路由

Loc-RIB　BGP 路由器通过对 adj-RIBs-In 中的路由使用它的本地路由策略而选择的路由

ADJ-RIBs-OUT　公布给对等体的路由

BGP 的决策过程，是对 ajd-RIBs-IN 中的路由使用本地路由策略，同时将选定的或修改过的路由放到 Loc-RIB 和 Adj-RIBs-Out 中

1. **Weight 越大越优先**
2. **Local_Pref 越大越优先**
3. **起源于本地的路由优先（如本地 network 的，或 aggregate 的），即下一跳是 0.0.0.0(在 BGP 表中,当前路**

由器通告的路由的下一跳为 0.0.0.0)

- **规则详解**

  以下优先级依次降低：default-originate（针对每个邻居配置）、default-information-originate（针对每种地址簇配置）、network、redistribute、aggregate-address

4. AS-Path 越短越优先

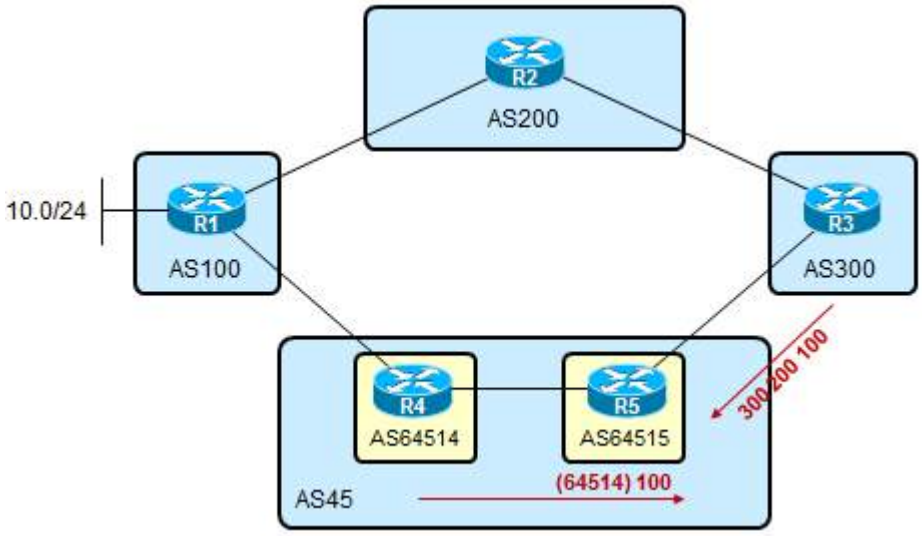
- **规则详解**

  在做聚合路由时，使用 as-set 后产生的 AS-Path 列表中的{}里的 AS 号长度只算一个 AS 号的长度

  在联盟内的 AS-Path 列表中的()的 AS 号长度不做计算依据

  bgp bestpath as-path ignore 这条命令如果配置了，则跳过此规则

- **规则验证（联邦内成员 AS 号不参与 AS_PATH 长度计算）**



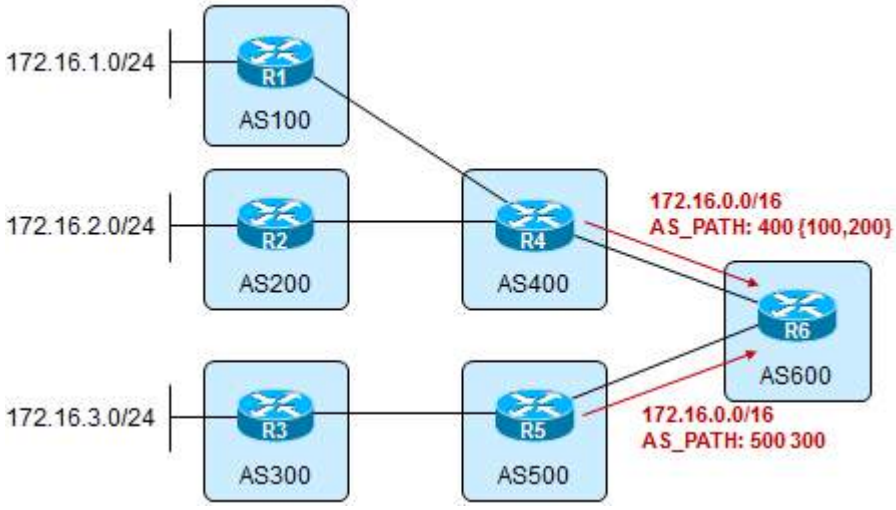
所有设备完成基本配置，建立 BGP 邻居关系。黄色区域为联邦成员 AS。

R5 将会同时从 R4 及 R3 收到关于 10.0 网络的 BGP 路由，R5 首先会优选 R3，因为 R4 传递给 R5 的路由保持了 NEXT_HOP，仍为 R1，因此 R5 上 R4 过来的这条路由下一跳不可达，通过在 R4 修改修改下一跳为自己后，R5 优选 R4。

此时在 R1 上对 R4 使用策略，将 10.0 的路由 AS_PATH 插入 100 的 AS 号，结果一边为 R4 过来的 (64514) 100 100 ，另一边为 R3 过来的 300 200 100，R5 仍优选 R4，因为(64512)不参与 AS_PATH 长度计算。

此时在 R1 上对 R4 使用策略，将 10.0 的路由 AS_PATH 插入 100 100 的 AS 号，R5 优选 R3。关于为什么 R5 会选择 R3，经过实验得出，这里 R5 收到的两条路由，一条来自联邦 EBGP 邻居，一条来自 EBGP 邻居，最终优选 EBGP 邻居的路由。

- **规则验证（AS_SET 类型的 AS_PATH，也就是{}中的 AS 号，在 AS_PATH 计算时作为一跳）**



BGP AS 规划如上，R1、R2、R3 各自在 BGP 进程中 network loopback 路由（不做任何策略）；

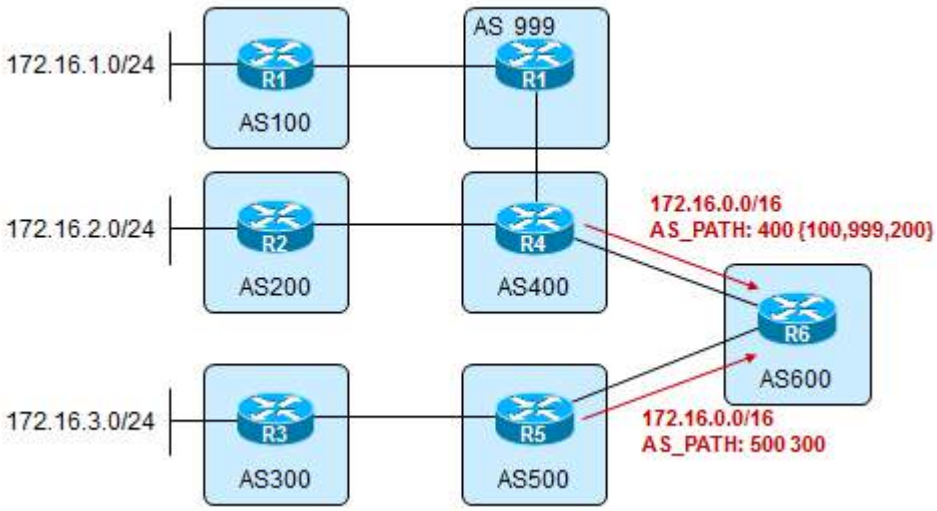
在 R4 及 R5 上分别进行路由汇总：

aggregate-address 172.16.0.0 255.255.0.0 as-set summary-only

如此一来，R6 将分别从 R4 及 R5 上收到汇总路由：172.16.0.0/16

注意到虽然 R4 传递过来的路由 400 {100，200}，大括号中的 AS，在 AS_PATH 计算中仅作为一个 AS 计算，因此 R6 上收到的这两条 EBGP 路由，AS_PATH 长度相等，最终，R6 将优选最老的 EBGP 邻居的路由。

再爽一把，下面的拓扑，现象和上面是一样的，要注意使用 as-set 关键字的汇总路由，底下的明细路由的 AS_PATH 都会被放入{}中，也就是 AS_SET 中。



5. **Origin 属性（优先：IGP > EGP > Incomplete）**
6. **MED 越小越优先**

- **规则详解**

  默认情况下仅有当所有的备选路由来自同一个 AS 的不同 EBGP 邻居时，才会比较 MED；

  bgp always-compare-med 这条命令，可以使得即使路由来源于不同的 AS 也进行 MED 的比较

7. **优选 EBGP 邻居发来的路由（相对于 IBGP 邻居），在联邦 EBGP 和 IBGP 中优选联盟 EBGP 路由**
8. **优选到 BGP NEXT_HOP 最近的路由，该路由是去往下一跳路由器 IGP 度量值最小的路由**
9. **如果有多条来自相同相邻 AS 的路由并通过 Maximum-paths 使多条路径可用，则将所有开销相同的路由加入 Loc-RIB**

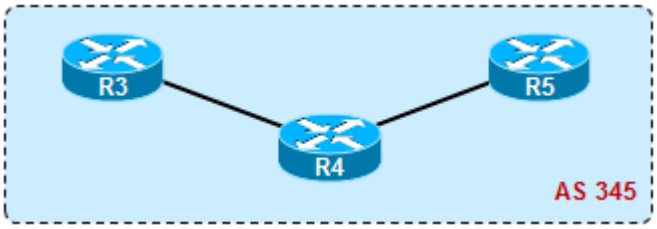
- **规则详解**

  当前面的 8 条选路原则都无法优选出最优路由时，并且在 BGP 进程下面配置了 maximum-paths [ibgp] n,n 的取值为 2-6,那么将执行等价负载均衡

  如果不关联 ibgp 关键字，那么只会对 EBGP 路由执行等价负载均衡（默认仅对 EBGP 路由）

  如果不配置 maximum-paths，那么将进行到下一条选路原则

- **实验验证（IBGP 等价负载均衡）**



  R3、R5 都向 R4 发布同一条网段的路由，在前 8 条规则都无法决策的情况下，如果 R4 配置了：Maximum-paths ibgp 2，则在 R4 上，ip 路由表里，将出现 BGP 路由的负载均衡，但是要注意，R4 在 BGP 表中，只会优选一条路由，应是 R3（因为 R3 的 RouterID3.3.3.3 小于 R5 的 5.5.5.5）

- **实验验证（EBGP 等价负载均衡）**

  EBGP 负载均衡，经过验证，只有当路由来源于同一个 AS 内的不同 EBGP 邻居时并且符合等价负载均衡的条件下，配置 maximum-paths x 才能实现负载均衡，如来源于不同的 AS，则配置了该命令也无效

- **非等价负载均衡请见 5.3 小节**

10. **如果路由都来自 EBGP 邻居，则优选最老的 EBGP 邻居传来的路由，降低滚翻的影响（此条主要对 EBGP 路由起效，但是现在基本不用该条，因不确定性太强）**

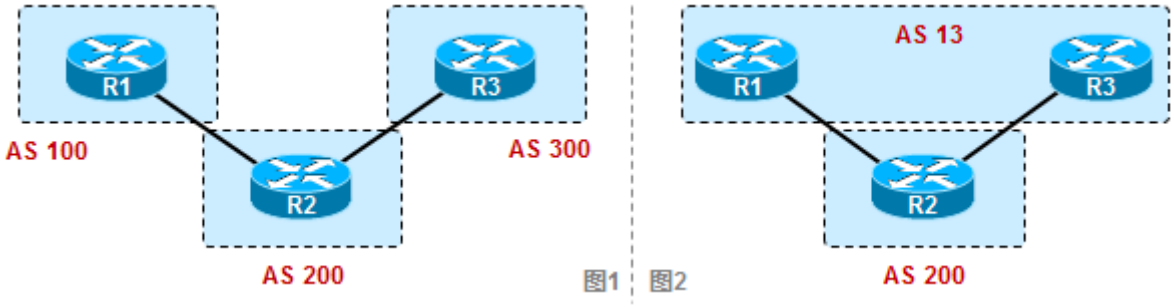
- **规则详解**

  最老的 EBGP 邻居，意味着有可能是最稳定的 BGP 邻居，因此这里做了优选，当然，这条规则可控性差，一般不作为选路策略使用。以下情况发生，将会跳过该条规则：

  1）配置了 bgp bestpath compare-routerid 命令后，将跳过该条原则，拥有最小 RID 的路由被选为最优

  2）多条路径具有相同的 RouterID，因为这些路由都是从同一台路由器接收过来的。

- **实验验证**



该实验是在 IOS C3640-IK9O3S-M Version 12.4(25)环境下测试，

图 1 R1、R3 同时向 R2 发布 13.13.13.0 的 BGP 路由，没有做任何的策略

由于实验中先配置的 R1，因此 R2 去往 13 网段，优选的是 R1，这时候在 R2 上 clear 一下 BGP 邻居 R1，则与 R1 BGP 连接 DOWN，R2 去往 13 的路由 又优选了 R3，待 R2 与 R1 恢复 BGP 连接并再次收到 R1 的 13 路由更新后，R2 仍优选 R3

图 2 R1、R3 为同一个 AS，也是同时向 R2 更新 13 网段的路由，实验效果也符合规则描述

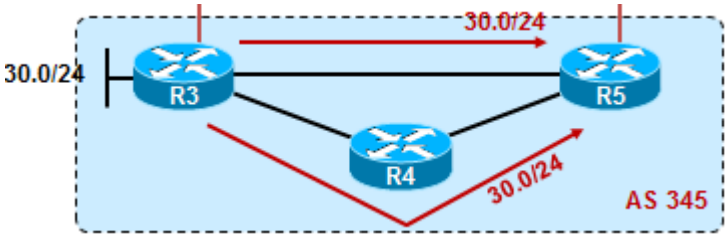
如果是在 IBGP 环境下，也就是 R1、R2、R3 都在一个 AS 内，那么该条规则不适用

11. BGP 邻居的 RID 越小越优先

- **规则详解**

RID 是路由器上的最大 IP 地址，倾向于赋给回环地址。也可以通过 bgp router-id 命令手工设置。如果一条路径包含 RR 属性，路由产生者 ID（originator）将在最优路径选择过程中代替 RID

- **实验验证**



R3-R4；R4-R5；R3-R5 维护 IBGP 邻居关系并且都是用各自的 LOOPBACK 口作为更新源及建立 BGP 邻居关系，地址分别为 3.3.3.3、4.4.4.4、5.5.5.5。AS 内运行 OSPF 协议，使得所有路由器都能获取到其他路由器的 LOOPBACK 网段。在 R3 上发布 30.0 网段的 BGP 路由，同时配置 R4 为 RR，R3 为 RR client。R5 会同时从 R3 及 R4 收到 30 网段的路由更新，它将如何选路？那一条选路规则生效？

首先规则 8 不适用 ,因为两条 BGP 路由 NEXT_HOP 相等 ,都是 3.3.3.3 ,不具有可比性(到达 3.3.3.3 的 metric 就一个值)

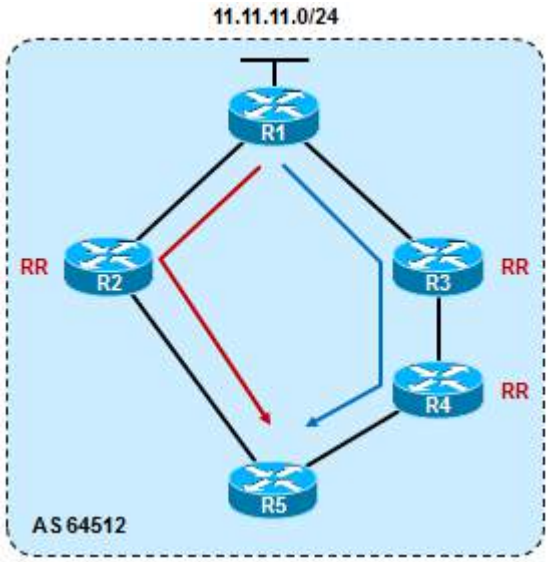
其次规则 9、10 略过

再次规则 11，似乎生效了，但是此时将 R3 的 BGP routerID 改的比 R4 大，发现 R5 仍然优选 R4，这是因为 “如果一条路径包含 RR 属性，产生者 ID 将在最优路径选择过程中代替 RID”

因此到规则 12 :如果多条路径始发路由器 ID 或路由器 ID 相同 ,那么优选 Cluster-List 最短的路径

12. 如果多条路径始发路由器 ID 或路由器 ID 相同，那么优选 Cluster-List 最短的路径

- **实验验证**



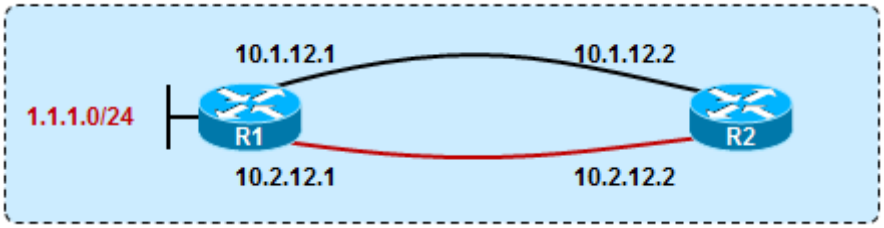

```
R5#show ip bgp 11.11.11.0
BGP routing table entry for 11.11.11.0/24, version 2
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    1.1.1.1 (metric 129) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator: 1.1.1.1, Cluster list: 4.4.4.4, 3.3.3.3
  Local
    1.1.1.1 (metric 129) from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 1.1.1.1, Cluster list: 2.2.2.2
```

13. 选择邻居 ip 地址最小的路由（BGP 的 neighbor 配置中的那个邻居的地址，也就是邻居的更新源 IP）

- **规则详解**

  注意这个地址，是邻居用来建立 BGP 邻居关系的地址。

- **实验验证**



R1R2 建立 IBGP 邻居关系，维护两条 BGP 连接，分别用本地的接口与对端的直连接口建邻居；

R1 宣告 1.1.1.0 进 BGP，R2 会分别从 10.1.12.1 及 10.2.12.1 学习到这两条路由

```
BGP routing table entry for 1.1.1.0/24, version 2
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Not advertised to any peer
  Local
    10.2.12.1 from 10.2.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, internal
  Local
    10.1.12.1 from 10.1.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

很明显 R2 优选了 10.1.12.1，这是因为前 12 条规则都无法决策，因此最后比较 neighbor IP，所以优选 10.1.12.1，当然，如果这时将 R1 的 10.1.12.1 地址改成 10.11.12.1，并且在 R2 上做 neighbor ip 的相应修改，那么 R2 去往 1.1.1.0 会切换到 10.2.12.1，因为这个 IP 小

补充说明：

1. 其中 3 意思是我自己 network 的

   该比较原则主要是指本地在进入一条 IGP 路由进去 BGP 表时，使用不同的方式比如 network 或 redistribute 等，那么这些方式之间是存在优先顺序的：network>redistribute>aggregate

   注意，该原则是不会作为 BGP 路由选路策略的

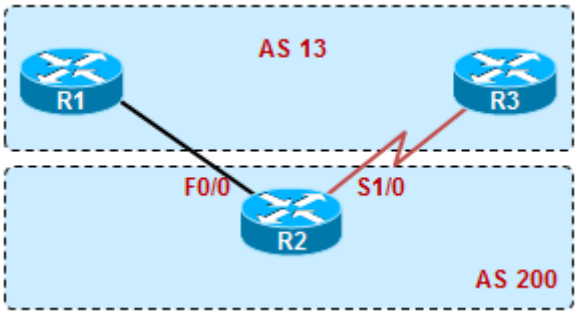
5.2 决策补充

如果一条路径满足下列任何一个条件，那么它在最佳路径选择过程中就不是有效的候选者

- 路径的下一跳不可达
- 路径未同步，但同步功能已开启
- 路径被入方向的 BGP 策略所拒绝，并且路由器配置了 soft reset
- 路由被惩罚 dampend

5.3 非等价负载均衡

1. EBGP 非等价负载均衡



在 R2 上，能分别从 R1 及 R3 上学习到 100.0 的 EBGP 路由，如果配置上 maximum-paths 2

则 R2 上，去往 100 网络就会同时将 R1、R3 作为下一跳进行等价负载均衡（虽然 R3 的上联口带宽更小，但是是直连）

R2 的配置如下：

```
router bgp 200
neighbor 10.1.12.1 remote-as 13
neighbor 10.1.23.3 remote-as 13
maximum-paths 2
```

R2#sh ip b 100.100.100.0

```
BGP routing table entry for 100.100.100.0/24, version 3
Paths: (2 available, best #1, table Default-IP-Routing-Table)
Multipath: eBGP
Flag: 0x860
  Advertised to update-groups:
     1
  13
    10.1.23.3 from 10.1.23.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, multipath, best
  13
    10.1.12.1 from 10.1.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, multipath
```

R2#sh ip route

```
B        100.100.100.0 [20/0] via 10.1.23.3, 00:02:57
                       [20/0] via 10.1.12.1, 00:02:57
```

如何进行非等价负载均衡呢？这个图中，R2 的上联接口带宽其实是不等的。

R2 的配置增加如下：

```
router bgp 200
 bgp dmzlink-bw
 neighbor 10.1.12.1 dmzlink-bw
 neighbor 10.1.23.3 dmzlink-bw
 maximum-paths 2
```

R2#sh ip b 100.100.100.0

```
BGP routing table entry for 100.100.100.0/24, version 5
Paths: (2 available, best #1, table Default-IP-Routing-Table)
Multipath: eBGP
Flag: 0x860
  Advertised to update-groups:
     1
  13
    10.1.23.3 from 10.1.23.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, multipath, best
      DMZ-Link Bw 193 kbytes
  13
    10.1.12.1 from 10.1.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, multipath
      DMZ-Link Bw 12500 kbytes
```

R2# sh ip ro 100.100.100.0

```
Routing entry for 100.100.100.0/24
  Known via "bgp 200", distance 20, metric 0
  Tag 13, type external
  Last update from 10.1.12.1 00:07:30 ago
  Routing Descriptor Blocks:
    10.1.23.3, from 10.1.23.3, 00:07:30 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
      Route tag 13
  * 10.1.12.1, from 10.1.12.1, 00:07:30 ago
```

```
Route metric is 0, traffic share count is 60
AS Hops 1
Route tag 13
```

F0/0 口带宽为 100M，S0/0 口带宽为 1.544M， 除一下，刚好比为将近 1:60

2. **IBGP 非等价负载均衡**

DMZ-Link Bw 193 kbytes 是一个扩展的 community 属性，因此，如果希望将该值传递给 AS 内的恰 IBGP 邻居，需加上 send-community，并且加扩展关键字；

DMZ-Link Bw 只能对本路由器的单跳 EBGP 邻居配置

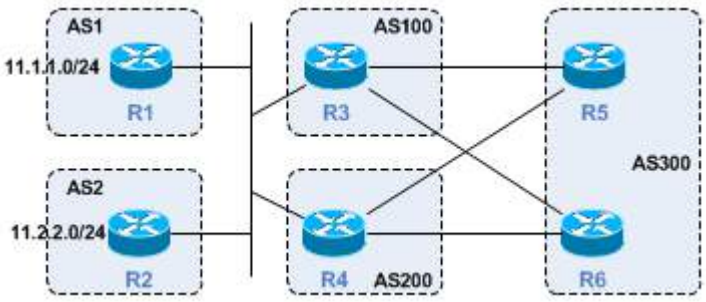
6 BGP 策略

6.1 Weight

neighbor 1.1.1.1 weight 2000 将邻居发送的路由均设置为 2000 weight（本地）

neighbor 1.1.1.1 filter-list 1 weight xxx 将邻居发送来的、且被 filter-list 1 匹配的条目，weight 设置为 xxx

- 注意 neighbor 1.1.1.1 filter-list x weight yyy 这条命令可以对一个邻居使用多次，也就是对不同的路由设置不同的 weight。但是 neighbor 1.1.1.1 filter-list 1 这条命令，在同一个邻居 in 或 out 方向，只能用一次，这两条命令是有很大区别
- 如果同时使用 neighbor 1.1.1.1 weight 和 neighbor 1.1.1.1 filter-list 1 weight xxx，则后者优先



R3、R4 同时向 R5 及 R6 更新 AS1 及 AS2 内的路由，如果要求 R5 去往 AS1 的路由走 R3，去往 AS2 的路由走 R4

那么在 R3 上的配置：

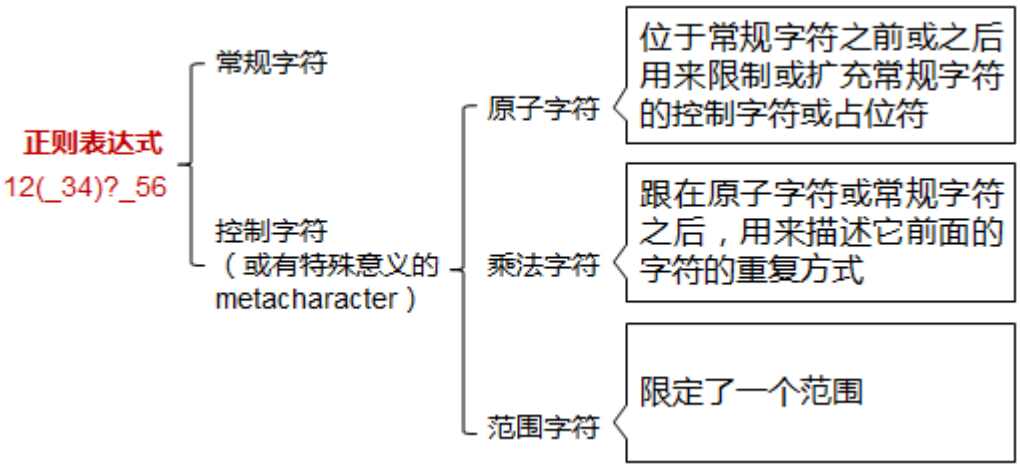
```
ip as-path access-list 1 permit   _1$                    //匹配 AS（100,1）及（200,1）
ip as-path access-list 2 permit   _2$                    //匹配 AS (100,2）及（200,2）
```

```
router b 300
    neighbor   R3   filter-list   1   weight   4000
    neighbor   R3   filter-list   2   weight   5000
```

某些 IOS 不支持 neighbor R3 filter-list 1 weight 命令

6.2 正则表达式

1. 正则表达式的组成



- **原子字符**

.	匹配任何单个的字符，包括空格
^	一个字符串的开始
$	一个字符串的结束
_	下划线，匹配任意的一个分隔符如 ^、$、空格、tab、逗号、{、}
|	管道符，逻辑或
\	转义符，用来将紧跟其后的控制字符转变为普通字符

原子字符示例：

^a.$	匹配一个以 a 开始，任意单一字符结束的字符串，如 a0，a!等
^100_	匹配 100、100 200、100 300 400 等
^100$	匹配 100
100$\|400$	匹配 100、1400、300 400 等
^\(65000\)$	仅仅匹配(65000)
[123].[7-9]	匹配 123 中的一个加上任意字符再加上 7 到 9 中的一个，如 167、3 空格 7 等

- **乘法字符**

*	匹配前面字符 0 次或多次出现
+	匹配前面字符 1 次或多次出现
?	匹配前面字符的 0 次或 1 次出现

一个乘法字符可以应用于一个单字符或多个字符，如果应用于多字符，需将字符串放入（ ）中。

乘法字符示例：

abc*d	匹配 abd、abcd、abccd、abcccd 等
abc+d	匹配 abcd、abccd、abcccd 等
abc?d	匹配 abd、abcd、abcdefg 等
a(bc)?d	匹配 ad、abcd、aaabcd 等

- **范围字符**

[ ]	表示一个范围。只匹配包含在范围内的字符之一。 可以在一个范围的开始使用 ^ 来排除范围内的所有字符，也可以使用下划线 _ 来指定一个区间。

范围字符示例：

[abcd]	匹配只要出现了 a、b、c、d 的内容
[a-c 1-2]$	匹配 a、a1、62、1b、xv2 等
[^act]$	匹配不以 a 或 c 或 t 结尾的内容
[123].[7-9]	159 220、91 70

2. 使用 as-path access-list 匹配路由

注意 as-path access-list 也是默认隐含拒绝所有

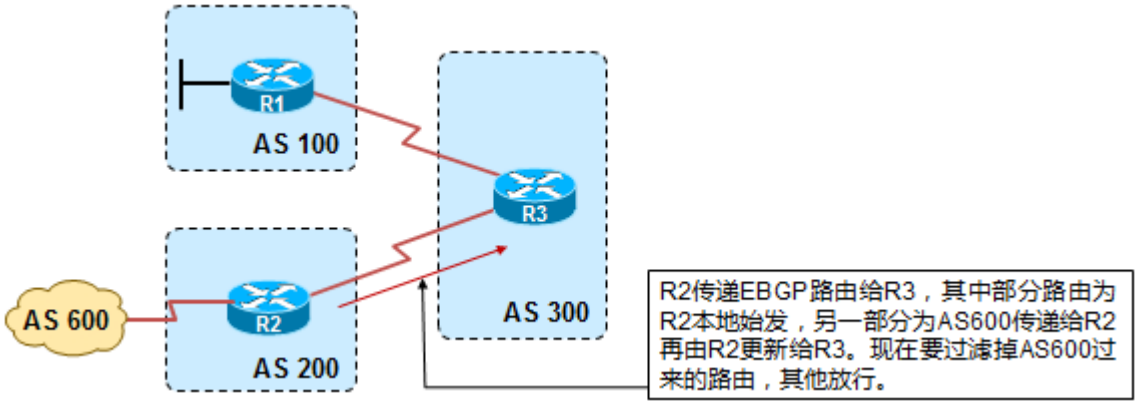
可使用 show ip bgp regexp 来检查所配置的正则表达式的结果。

正则表达式示例：

^$	匹配不包含任何 AS 号的 AS_PATH，也就是本 AS 内的路由
.*	一个点和一个星号，匹配所有，任何
^100$	就匹配 100 的这个 AS_PATH
_100$	以 100 结束的 AS_PATH，也就是路由起源于 100AS 的路由
^10[012349]$	匹配 100、101、102、103、104、109 这些 AS_PATH

^10[^0-6]$	匹配除了 100~106 外的 AS_PATH
^10.	匹配 100~109，以及 10，因为“.”也包含空格
^(100|200)$	匹配包含 100 及 200 的 AS_PATH
12(_34)?_56	匹配 12 56 及 12 34 56

- **配置示例 1：as-path access-list 搭配 filter-list**



在 R2 上开启 loopback，IP 分别为 172.16.10.0/24、172.16.11.0/24

R2 的配置如下：

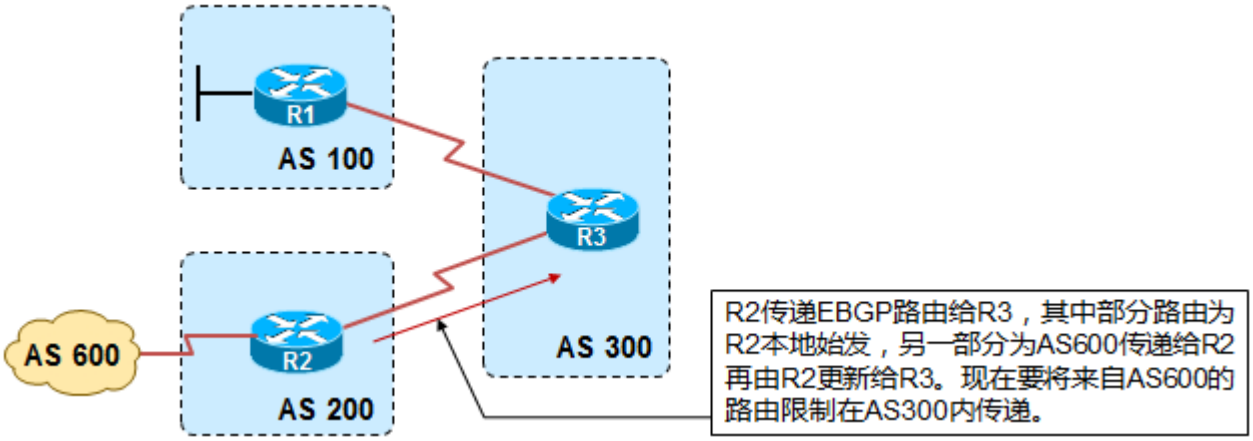
```
ip prefix-list 10 seq 5 permit 172.16.10.0/24
ip prefix-list 11 seq 5 permit 172.16.11.0/24
route-map test permit 10
  match ip address prefix-list 11
  set as-path prepend 600
route-map test permit 20
router bgp 200
  network 172.16.10.0 mask 255.255.255.0
  network 172.16.11.0 mask 255.255.255.0
  neighbor 10.1.23.3 remote-as 300
  neighbor 10.1.23.3 route-map test out
```

通过上述配置，模拟 R3 收到来自 AS200 及 AS600 的路由，此时在 R3 上要过滤掉来自 AS600 的路由，由于 AS600 的路由，AS_PATH 的尾端均为 600，因此正则表达式可用：_600$，下划线用于匹配空格。但是注意，as-path access-list 也是默认隐含 deny any 的，因此要注意 deny 掉了来自 600AS 的路由，其他的路由要放行。R3 配置如下：

```
ip as-path access-list 1 deny _600$
ip as-path access-list 1 permit .*
router bgp 300
```

```
neighbor 10.1.13.1 remote-as 100
  neighbor 10.1.23.2 remote-as 200
  neighbor 10.1.23.2 filter-list 1 in
```

- **配置示例 2：as-path access-list 搭配 route-map**



R2 的配置与上面大同小异。关键看 R3 的配置：

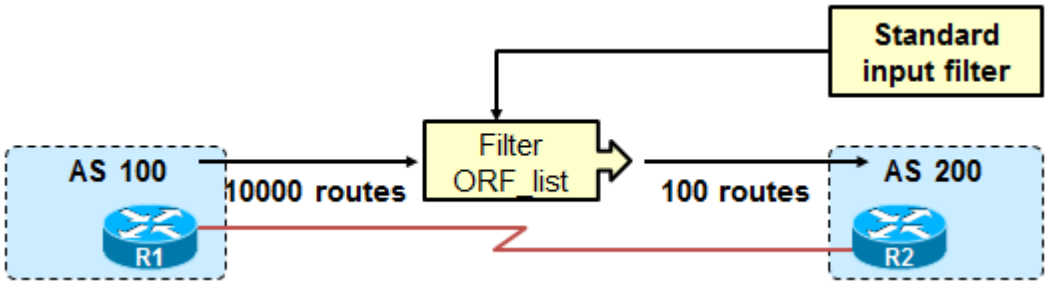
```
ip as-path access-list 1 permit _600$
route-map setCommu permit 10
  match as-path 1
  set community no-advertise
route-map setCommu permit 10
router bgp 300
  neighbor 10.1.23.2 route-map setCommu in
```

3. 正则表达式在 CISCO IOS 中的其他应用

Show 和 more 的输出信息中，可使用管道符 | 搭配正则表达式来过滤输出结果。

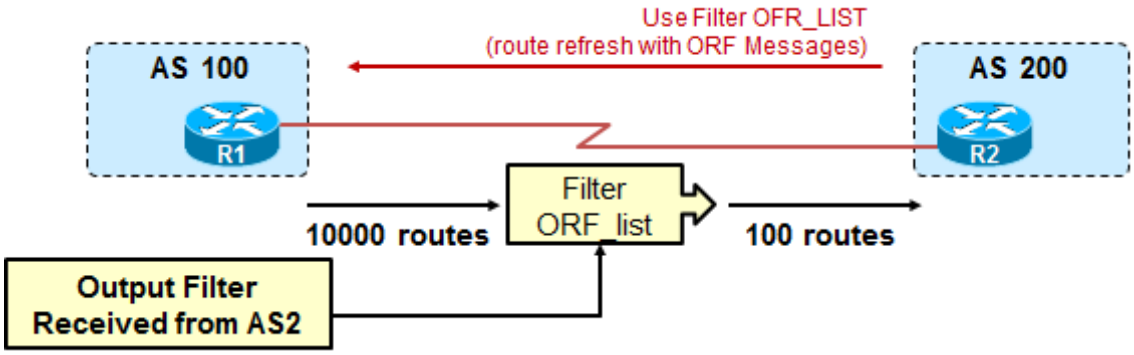
```
show run | in route
```

6.3 ORF

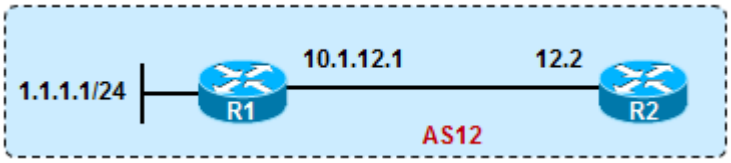


AS100 发了 10000 条路由给 R2，如果 R2 只希望收取其中 100 条路由，其余过滤，那么一般的做法是在 R2 上执行 in 方向的 filter，因为 AS100 对于 AS200 而言，是不可控的。但是这种方法，实际上对于链路带宽及网络资源的角度来说，是存在一定的浪费的。

ORF 的特性很简单，就是 R2 将这个 list 推送给 R1，让这个策略在 R1 这端就执行过滤，如下图。



配置如下：



Sender（R2）省去基本配置，如 BGP 邻居关系等

```
router bgp 12
   address-family ipv4 unicast
   neighbor 10.1.12.1 capability orf prefix-list send
   neighbor 10.1.12.1 prefix-list FILTER in

ip prefix-list FILTER deny 1.1.1.0/24
ip prefix-list FILTER permit
```

receiver（R1）

```
router bgp 100
  address-family ipv4 unicast
  neighbor 10.1.12.2 capability orf prefix-list receive
```

所以实际上 ORF 是借助 prefix-list 去实现路由过滤的一个特性，在本地将 prefix-list 推送给对端，让对端来执行路由前缀的过滤。

一旦部署了 ORF，那么 BGP peer 之间在建立 BGP 邻居关系的时候，在 open 消息里就会去进行 ORF 的能力协商，如果协商成功了，则使用 route-refresh 报文推送 ORF 内容。

下图是协商能力参数的 open 消息内容（注意 ORF 部分）

```
⊟ OPEN Message
    Marker: 16 bytes
    Length: 56 bytes
    Type: OPEN Message (1)
    Version: 4
    My AS: 12
    Hold time: 180
    BGP identifier: 2.2.2.2
    Optional parameters length: 27 bytes
  ⊟ Optional parameters
    ⊞ Capabilities Advertisement (8 bytes)
    ⊞ Capabilities Advertisement (4 bytes)
    ⊞ Capabilities Advertisement (4 bytes)
    ⊟ Capabilities Advertisement (11 bytes)
        Parameter type: Capabilities (2)
        Parameter length: 9 bytes
      ⊟ Cooperative route filtering capability (9 bytes)
          Capability code: Cooperative route filtering capability (130)
          Capability length: 7 bytes
        ⊟ Capability value
            Address family identifier: IPv4 (1)
            Reserved: 1 byte
            Subsequent address family identifier: Unicast (1)
            Number of ORFs: 1
            ORF Type: Cisco PrefixList ORF-Type (128)
            Send/Receive: Send (2)
```

下图是 sender 使用 route-fresh 报文推送 prefix-list

```
Border Gateway Protocol
⊟ ROUTE-REFRESH Message
    Marker: 16 bytes
    Length: 44 bytes
    Type: ROUTE-REFRESH Message (5)
    Address family identifier: IPv4 (1)
    Reserved: 1 byte
    Subsequent address family identifier: Unicast (1)
  ⊟ ORF information (21 bytes)
      ORF flag: Immediate
      ORF type: Cisco PrefixList ORF-Type
      ORF len: 17 bytes
    ⊟ ORFEntry-PrefixList (10 bytes)
        ACTION: Add MATCH:  deny
        Entry Sequence No: 5
        PrefixMask length lower bound: 0
        PrefixMask length upper bound: 0
      ⊞ 1.0.0.0/8
    ⊟ ORFEntry-PrefixList (9 bytes)
        ACTION: Add MATCH: Permit
        Entry Sequence No: 10
        PrefixMask length lower bound: 0
        PrefixMask length upper bound: 32
      ⊟ 0.0.0.0/0
          ORF prefix length: 0
          ORF prefix: 0.0.0.0
```

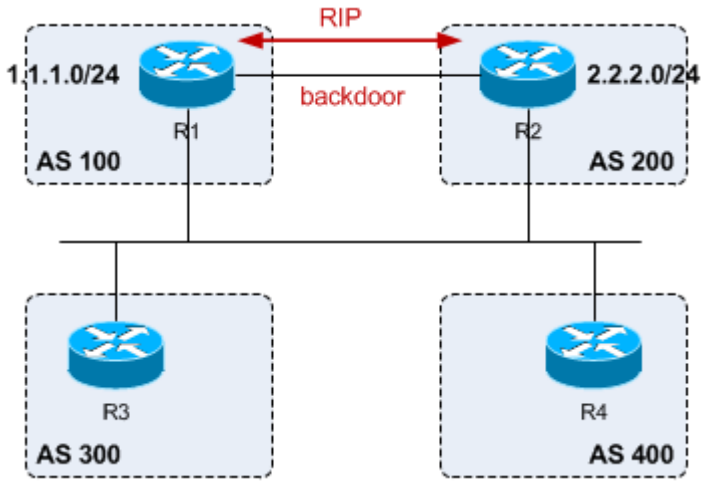
prefix-list 第一个条目

prefix-list 第二个条目

6.4 AD 值及后门

修改 BGP 的默认 AD 值，命令如下：

distance bgp x y z //x 为 EBGP 路由，y 为 IBGP 路由，z 为本地 network 宣告的 BGP 路由



R1，R2 之间，原先与 R3 和 R4 是通过一个多路访问网络更新路由，跑的是 BGP。

R1 及 R2 由于是私密合作伙伴，有些特殊的流量可能需要受保护，因此增加了 R1、R2 之间的专线，这时如果需保证 1.1.1.0 及 2.2.2.0 间的流量互访通过专线(跑了 RIP)走，当专线 DOWN 了，才走下面的多路访问网络。

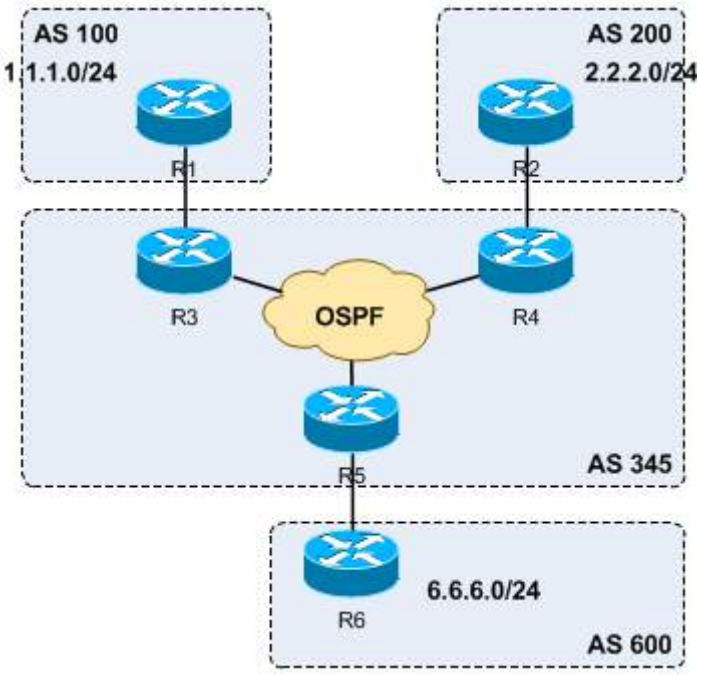
虽然 R1、R2 之间运行了 RIP 协议，互相更新 1.1.1.0 及 2.2.2.0 的路由，但是，由于 R1 同时从 RIP 及他的 EBGP 邻居学习到 2.2.2.0，RIP 的管理距离为 120，EBGP 为 20，所以优选 EBGP 的路由。如何使 R1 优选 RIP 路由呢？

- **解决方案 1，**修改 BGP 的默认 AD 值，使得其 EBGP 路由 AD 值大于 RIP 从而达到优选的目的，但是这个方案未必合适，因为他会对所有的 BGP 路由都产生印象，如果仅仅希望对 1.1.1.0 及 2.2.2.0 产生影响呢？
- **解决方案 2，**由于 BGP 本地路由（本地 network 引入的）默认的 AD 值为 200，比 RIP 大，因此如果在 R1 上，将其从 EBGP 邻居学习到的 2.2.2.0 路由(此时为 EBGP 路由，AD 为 20)，同时在本地 network 一下(变成 BGP 本地路由)，那么这条路由的 AD 将变为 200，比 RIP 大，所以此时优选 RIP 路由。R2 同理，在 BGP 进程中 network 1.1.1.0

  这个方案貌似可以解决问题，但是却带来了新的问题，首先以 R1 为例，R1 采用重发布的方式引入 1.1.1.0，这条路由通过 BGP 的方式被更新给了 R2、R3、R4（通过多路访问链路），这时候在这些路由器上，1.1.1.0 路由的 original 为 incomplete；而为了实现上面的需求，R2 在其本地 network 了 1.1.1.0，这一来，1.1.1.0 在 R2 当做本地路由引入 BGP，AD 值变为 200 的同时，R2 向它的 EBGP 邻居通告 1.1.1.0，而其通告的 1.1.1.0，original 属性为 i，如此 R3、R4 同时从 R1 及 R2 学习到 BGP 路由 1.1.1.0，经过决策，去往 1.1.1.0 的流量会从 R2 发往 R1，这样就次优路径了。
- **解决方案 3，**所以上面的问题在于，R2 不应该向其 EBGP 邻居通告 1.1.1.0，因此 CISCO 支持使用命令：network 1.1.1.0 backdoor 来解决该问题，一来该路由变成了本地路由，AD 值成了 200，而来路由器不会向其 EBGP 邻居通告该路由。

6.5 路由标记 TAG (待修订)

支持 TAG 的路由协议有：RIPv2、EIGRP、OSPF、ISIS、BGP



R1、R2、R6 都通过 EBGP 通告了各自 AS 的路由，在 AS345 中，这些路由被重发布进 OSPF，接着又在其他两台路由器重发布回了 BGP，然后被发布给他们的 EBGP 对等体。

这里由于中间隔了个 OSPF，而 OSPF 又不了解 BGP，因此在重发布过程中，BGP 的属性就会被丢失。

BGP 可以利用 OSPF 包中的路由标记在穿越 OSPF 时携带 AS_PATH 信息。实际上 CISCO 的 BGP 自动完成了该过程（注意，自动进入 OSPF TAG 字段的只有 AS_PATH 属性一值）。

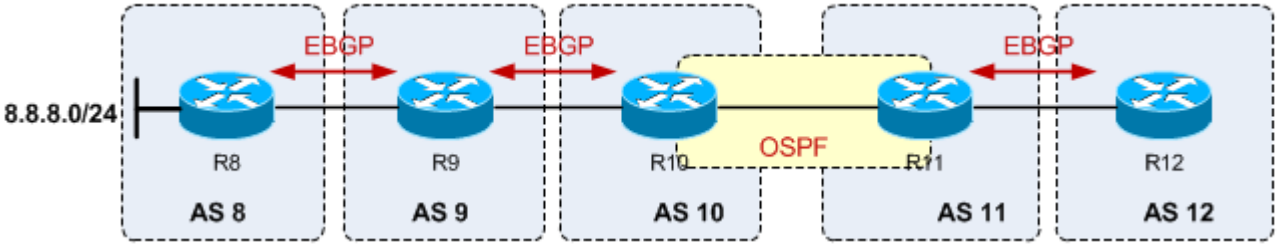
例如在R3上查看R5发过来的6.6.6.0 这条OSPF外部路由,可以看到tag字段为600(自动继承了AS_PATH)

而这个时候，R3 将 OSPF 重发布进 BGP 时，默认是不会自动恢复 AS_PATH 属性的，因此需要：

使用 set as-path tag，来恢复

例如本例在 R3 上，

```
Route-map test per 10
  Set as-path tag
router bgp 345
  redis ospf 1 route-map test
```



R8 重发布 8.8.8.0 进 BGP，一直传到 R10，在 R10 上，该路由的 AS_PATH 为 **9 8 ?**，

在 R10 上，将该路由重发布进 OSPF，R11 通过 OSPF 学习到了 8.8.8.0，

在 R11 上，该路由为：

```
R11#sh ip ro 8.8.8.0
Routing entry for 8.8.8.0/24
  Known via "ospf 1", distance 110, metric 1
  Tag 9, type extern 2, forward metric 64
  Last update from 192.168.101.10 on Serial0/0, 00:12:46 ago
  Routing Descriptor Blocks:
  * 192.168.101.10, from 10.10.10.10, 00:12:46 ago, via Serial0/0
      Route metric is 1, traffic share count is 1
      Route tag 9
```

因此 :该路由被重发布进 OSPF 时 ,自动将这条外部路由继承 BGP 的 A_PATH 属性 ,注意 ,只继承 AS_PATH 列表中，第一个 AS 号（也就是去往目的地离本 AS 最近的那个）

这时候继续在 R11 上将 OSPF 路由重发布进 BGP，以便 R12 能学习到，R12 上学到路由后，发现 BGP 属性（包括 AS_PATH）都丢失了。因为在 OSPF 重发布进 BGP 时，默认不会自动将 tag 值复制进 AS_PATH，需在 R11 上

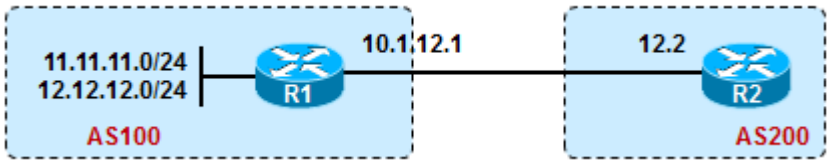
```
route-map test permit 10
 set as-path tag
router bgp 11
redistribute ospf 1 route-map test
```

配置后，发现软清路由没用，没办法，只能用 clear ip b *重置 bgp 连接了。之后再 R12 上看到的 8.8.8.0 路由 AS_PATH 为：11 9 ?

关于进一步的问题，请见 RFC 1403 - BGP OSPF Interaction

6.6 BGP 过滤器

6.7 Prefix-list

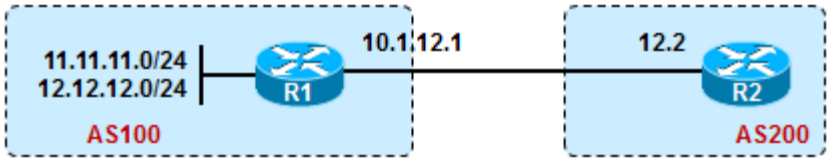


R1 与 R2 建立 EBGP，R1 上 network 宣告 11 及 12 网段，欲过滤掉 12 网段，不让其传递给 R2，可：

```
ip prefix-list 11 deny 12.12.12.0/24
ip prefix-list 11 permit 0.0.0.0/0 le 32
router bgp 100
 network 11.11.11.0 mask 255.255.255.0
 network 12.12.12.0 mask 255.255.255.0
 neighbor 10.1.12.2 remote-as 200
 neighbor 10.1.12.2 prefix-list 11 out
```

当然，在 R2 上做 in 方向的前缀列表进行过滤也是可以的

6.8 Distribute-list



R1 上，过滤掉 11.11.11.0/24 路由，其他放行

```
R1(config)# ip prefix-list 11 seq 5 deny 11.11.11.0/24
R1(config)# ip prefix-list 11 seq 10 permit 0.0.0.0/0 le 32
R1(config)# router bgp 100
R1(config-router)# network 11.11.11.0 mask 255.255.255.0
R1(config-router)# network 12.12.12.0 mask 255.255.255.0
R1(config-router)# distribute-list prefix 11 out
```

6.9 Route-map

- **可以在以下的 BGP 命令中使用 route-map**

  neighbor

bgp dampening

network

redistribute

- **可以为特定的目的在不同的命令中使用 route-map**

suppress-map

unsuppress-map

advertise-map

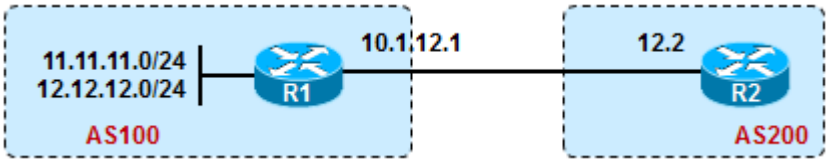
inject-map

exist-map

non-exist-map

tabel-map

- **示例 1：使用 route-map 关联 neighbor 过滤路由**




```
R1(config)# access-list 1 deny 11.11.11.0
R1(config)# access-list 1 permit any
R1(config)# route-map rp permit 10
R1(config-route-map) match ip address 1
R1(config)# router bgp 100
R1(config-router)# neighbor 10.1.12.2 route-map rp out
```

6.10 Policy-list

1. **policy-list 的概念**
   - 可预先将包含一组 match 语句的 route-map 定义成一个命令列表，这个列表称为 policy-list
   - 这些 policy-list 可以在 route-map 中被调用
   - 一个 policy-list 就像个只包含 match 语句的 route-map
   - 当 route-map 被执行，被其调用的 policy-list 中所包含的 match 语句将一并被遍历且执行 match 动作
   - 这是一种在大中型网络中运用、使得配置“模块化”的特性
2. **policy-list 的特性**
   - Ipv6 不支持
   - 12.0(22)S 和 12.2(15)T 之前的 CISCO IOS 版本不支持该特性，另外更老的 IOS 版本的路由器重启存在

路由策略配置丢失的风险

- Policy-list 中不能包含 set 语句，但是它被 route-map 调用后，该 route-map 中可以包含 set 语句
- Policy-list 只在 BGP 中支持，其他的 IP 路由协议并不支持这个特性

3. **policy-list 的配置**

```
ip policy-list as100 permit
  match as-path 1
  match community 1
```

上述命令创建一个 policy-list；Policy-list 只能包含 match 语句

```
route-map RP permit 10
  match policy-list as100
  match ip address prefix-list 100
  set local-preference 300
```

上述命令在 route-map 中调用定义好的 policy-list；一个 route-map 中可调用多个 policy-list

4. **配置示例**

```
ip prefix-list 1 permit 10.0.0.0/8
ip as-path access-list 1 permit ^100_
ip as-path access-list 2 permit ^200_
ip community-list 1 permit 300:105
```

```
ip policy-list as100 permit
 match as-path 1
 match community 1
```

```
ip policy-list as200 permit
 match as-path 2
 match community 1
```

逻辑的或

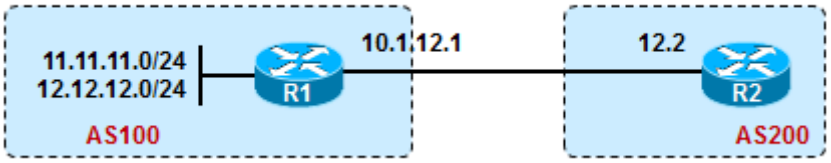
```
route-map Test permit 10
 match ip address prefix-list 1
 match policy-list as100 as200
 set local-preference 110
route-map Test permit 20
```

6.11 advertise-map

1. **neighbor x.x.x.x advertise-map xx exist-map xx**

   跟下面其实是一条命令，这么敲进去，show 一下又变成下面的东东了。

2. **neighbor xxx advertise-map yyy non-exist-map zzz**



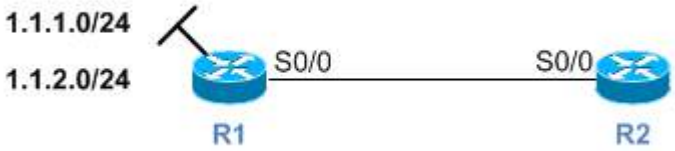
R1 上如若做如下配置：

```
ip prefix-list 1 permit 11.11.11.0/24
ip prefix-list 2 permit 12.12.12.0/24
route-map RP1 permit 10
  match ip address prefix-list 1
route-map RP2 permit 10
  match ip address prefix-list 2
router bgp 100
  neighbor 10.1.12.2 advertise-map RP1 non-exist-map RP2
```

如果 RP2 匹配的路由在 BGP 表中存在，则通告 RP2；如果 RP2 挂了，则通告 RP1

亲测这个特性相当不稳定，不建议使用。

6.12 Unsuppress-map

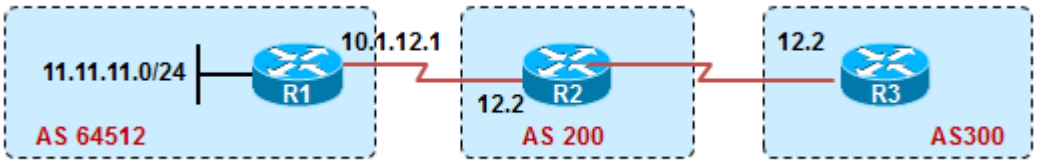


在 R1 上对 1.1.1.0 及 1.1.2.0 做汇总，成 1.0.0.0/8，如果加上 summary-only 关键字，则两个明细会被抑制掉

如果希望对某个邻居取消抑制某条明细，如 1.1.1.0，则

```
ip prefix-list 1 seq 5 permit 1.1.1.0/24
route-map test permit 1
 match ip address prefix-list 1
router bgp 1
  neighbor R2   unsuppress-map test
```

6.13 私有 AS 号

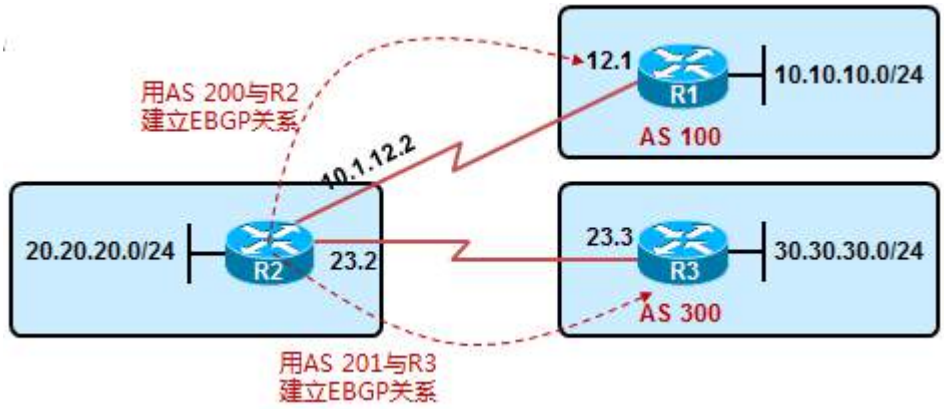


AS64512 为私有 AS 号，是不允许宣告到 Internet 的，如果不加限制，R3 收到 R1 的路由，AS_PATH 肯定包含 64512 这个私有 AS 号，假设 R3 为 Internet 路由器，如何在 R2 上，向 R3 传递路由时，屏蔽掉私有 AS 呢？只需一条非常简单的命令：neighbor 10.1.23.3 remove-private-as

6.14 DUAL AS

1. 基本概念

- 默认情况下，在单台 Router 上只能启动一个 BGP 进程，并且只能属于一个 AS。DUAL AS 允许我们在不终端现有 BGP 连接的情况下，在 primary AS 下同时运行一个 secondary AS，从而提供一种网络迁移的机制。
- 在迁移期间，运行 DUAL AS 的路由器可以同时使用 primary AS 及 secondary AS 与外部 AS 建立 EBGP 连接，并且都能进行 BGP 路由的更新和传递。
- 在不用断开现有连接的情况下，可缩短网络迁移的时间，且不用大批量的变更设备 BGP 配置

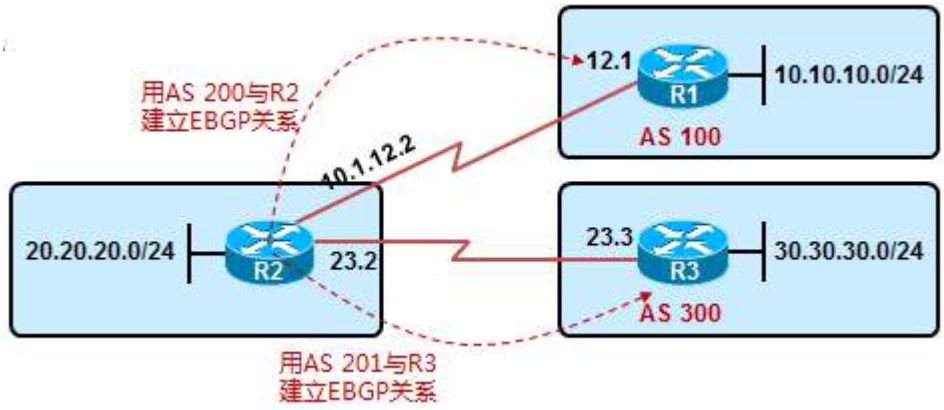


举个简单的例子：假设 R2（AS200）为一个 Customer Network，R1（AS100）是 R2 的 ISP，此时 BGP 邻居关系已经建立起来了，并且路由都已经收敛完毕，突然出现另一个 ISP-R3（AS300），并且它给 R2 分配的 AS 号为 AS201，这时候 R2 就很尴尬了，只能重新配置 BGP，断开 R1 的连接并与 R3 建立 BGP 连接。

DUAL AS 提供我们一种迁移的解决方案：允许 R2 的 BGP 进程同时兼顾两个 AS 号，200 及 201，其中 200 为 Primary AS，201 为 Secondary AS，R2 可以在保持原有的与 R1 的 BGP 连接及路由更新不受影响的情况下，

新增与 R3 的 BGP 连接，并且是使用 AS201 与 R3（AS300）建立的 EBGP 邻居关系。

2. 配置命令



R1 的配置如下：

```
router bgp 100
 network 10.10.10.0 mask 255.255.255.0
 neighbor 10.1.12.2 remote-as 200
```

R3 的配置如下：

```
router bgp 300
 network 30.30.30.0 mask 255.255.255.0
 neighbor 10.1.23.2 remote-as 201        // R3 指 R2 的 remote-as 为 201，也即是 secondary AS
```

R2 的配置如下：

```
router bgp 200                           // AS200 为 Primary AS，用于配置 BGP 进程
 network 20.20.20.0 mask 255.255.255.0
 neighbor 10.1.12.1 remote-as 100
 neighbor 10.1.23.3 remote-as 300
 neighbor 10.1.23.3 local-as 201  [no-prepend]  [replace-as]  [dual-as]
```

3. 命令详解

R1 发布 10.0、R2 发布 20.0、R3 发布 30.0 的路由，接下去看看 R2 上分别配置如下不同的命令后（关键字），这几条 BGP 路由被特定的路由器学习到之后的 AS_PATH 的变化：

- neighbor 10.1.23.3 **local-as 201**

学习者	10.10.10.0/24	20.20.20.0/24	30.30.30.0/24
R1	本地始发	200	**200 201 300**
R2	100	本地始发	300
R3	**201 200 100**	**201 200**	本地始发

- neighbor 10.1.23.3 local-as 201 **no-prepend**

学习者	**10.10.10.0/24**	**20.20.20.0/24**	**30.30.30.0/24**
R1	本地始发	200	**200 300**
R2	100	本地始发	300
R3	201 200 100	201 200	本地始发

注意这里的变化：R2 将 30.0 的路由传递给 R1 时，不再插入 seconday AS 号

所以，no-prepend 关键字的作用是：Do not prepend local-as to updates from ebgp peers，翻过一下：向 Primary AS 的 EBGP 邻居通告路由时，不附加 secondary AS 号

- neighbor 10.1.23.3 local-as 201 no-prepend **replace-as**

学习者	**10.10.10.0/24**	**20.20.20.0/24**	**30.30.30.0/24**
R1	本地始发	200	200 300
R2	100	本地始发	300
R3	**201 100**	**201**	本地始发

注意变化，replace-as 关键字让 R2 向 secAS 的 EBGP 邻居发送路由时，用 local-as201 替代真实 AS200

所以 replace-as 关键字的作用是：Replace real AS with local AS in the EBGP updates，翻译一下： 当路由器向 secondaryAS 的 EBGP 邻居发送更新时，用 secondaryAS 号替代 Pri AS 号。

- neighbor 10.1.23.3 local-as 201 no-prepend replace-as **dual-as**

Accept either real AS or local AS from the ebgp peer。翻译一下：EBGP 对等体既可以使用 PriAS 也可以使用 SecAS 对本地指 remote-as，例如在 R2 上配置上述命令后，R3 也就是 10.1.23.3，在其 BGP 进程中，即可使用 neighbor 10.1.23.2 remote-as 200，亦可使用 remote-as 201 去指 BGP 邻居 R2

6.15 默认路由

1. **方法一：静态默认路由 network**

   Ip route 0.0.0.0 0.0.0.0 null0

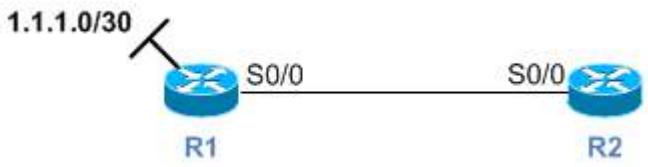
   然后在 BGP 进程中 network 0.0.0.0 即可将此默认路由注入 BGP 进程

   使用该方法在 BGP 中引入的默认路由会被传递给所有 BGP 邻居

2. **方法二：neighbor xxx default-originate**

   BGP 进程中： neighbor xxx default-originate 向特定的邻居传递默认路由

   此配置无需路由表中存在默认路由。有点类似 OSPF 的 default-originate always



可以有条件的下发默认路由

```
access-list 1 permit 1.1.1.0 0.0.0.3
route-map test permit 10
  match ip address 1
router bgp 100
   neighbor   R2   default-originate route-map test
```

如此只要 1.1.1.0 网络不 DOWN，R1 就始终会向 R2 下发默认路由

注意 access-list 1 的写法，实验的结果是，acl 必须匹配接口的 IP 和掩码，例如 如果写成 permit 1.1.1.0 0.0.0.255 就不行了

3. **方法三：default-information originate**

在本地配置一条静态的默认路由，如 ip route 0.0.0.0 0.0.0.0 null0

默认情况下 BGP 在重发布 static 的时候，不允许注入静态默认路由；

除非在同时在 BGP 进程中 default-information originate

```
ip route 0.0.0.0 0.0.0.0 null0
router bgp x
  default-information originate
  redistribute static
```

6.16 其他配置

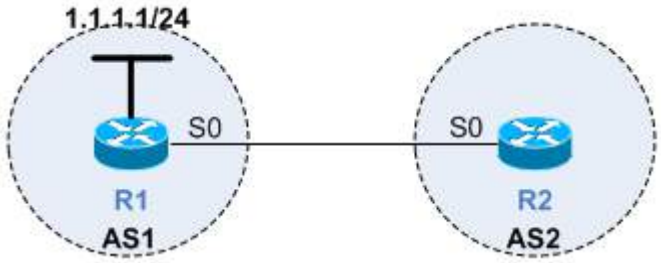
1. **bgp fast-external-fallover**

默认开启，如果一个接口 DOWN 了，BGP 回话立即中止。

如果一个接口处于 flapping 状态，那么将会给网络带来大量的 upadate 和 witdhdrawn 消息。因此对于一个翻动的接口，建议 no bgp fast-external-fallover，如此接口 shutdown 后，BGP 会话和邻居关系将仍在，直到 Holddown 计时器超时

7 故障案例分析

- **【案例】路由翻动**



如图，R1、R2 开设 LOOPBACK 接口，地址分别为 1.1.1.1、2.2.2.2/24

R1、R2 之间运行 EIGRP，与此同时，R1、R2 在 EIGRP 进程中宣告各自的 LOOPBACK 接口

R1、R2 之间形成 EBGP 邻居关系(使用各自 LOOPBACK 口)与此同时，R1 在 BGP 进程中 network 1.1.1.0 mask 255.255.255.0 引入 1.1.1.0 网络

观察一段时间，R2 会出现什么故障？为什么会出现这样的故障？

8 参考书目

TCP/IP 卷二	本笔记包含 TCPIP 路由技术卷二 BGP 部分的几乎所有内容，并做了详细扩展
CCNP ROUTE	包含书中所有内容
BGP 设计与实现	推荐阅读
RFC1771	A Border Gateway Protocol 4
RFC1403	BGP OSPF Interaction

IP routing

红茶三杯 CCIE 学习文档

文档版本： 2.0 更新时间： 2013-08-01

文档作者： 红茶三杯 http://weibo.com/vinsoney

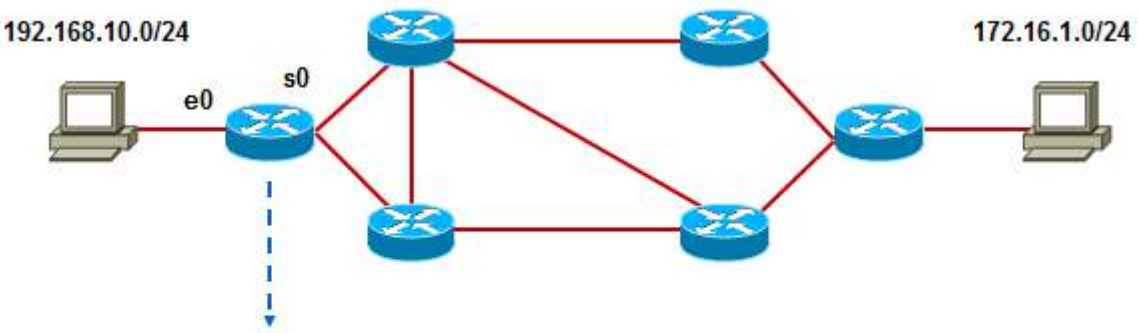
文档备注： 请关注文档版本及更新时间。关注 http://ccietea.com 以便获得文档的最新版本。

红茶三杯
网络工程 | 项目管理 | IT 服务管理 | CCIE 培训
学习 沉淀 成长 分享
微博：http://weibo.com/vinsoney
博客：http://blog.sina.com.cn/vinsoney
站点：http://ccietea.com

目录

1 IP 路由概述

1.1 关于 IP 路由



Protocol	Destination Network	Exit Interface
Connected	192.168.10.0	e0
RIP	172.16.1.0	s0

在一个 IP 网络中，路由（Routing）是个非常非常基本的概念。网络的基本功能，是使得处于网络中的两个 IP 节点能够进行通信，而通信实际上就是数据交互的过程，数据交互则需要网络设备帮助我们来将数据在两个通信节点之间进行传输。当路由器（或者其他三层设备）收到一个 IP 数据包，路由器会找出 IP 包三层头中的目的 IP 地址，然后拿着目的 IP 地址到自己的路由表中进行查找，找到“最匹配”的条目后，将数据包根据路由条目所指示的出接口或下一跳 IP 转发出去，这就是 **IP 路由（IP routing）**。而每台路由器都会在本地维护一个**路由表（Routing Table）**，路由表中装载着路由器获知的路由条目（Routes），路由条目由**路由前缀（路由所关联的目的地）、路由信息来源、出接口或下一跳 IP** 等元素构成。路由器通过静态的或者动态的方式获取路由条目并维护自己的路由表。

1.2 IP 路由表（IP Routing Table ）

每个路由表项最少必须包括下面三个项目：

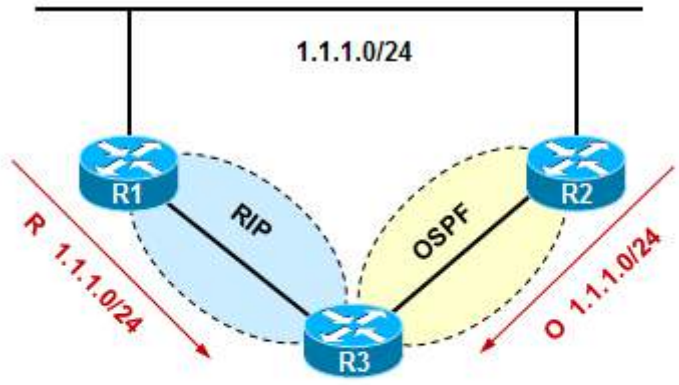
- **目标地址（路由前缀）：** 这是路由条目所关联的目的网络号。一条完整的路由前缀由：网络号+前缀长度构

成，两者缺一不可，例如 192.168.1.0/24 与 192.168.1.0/25，虽然两者的网络号相同，都是 192.168.1.0，但是两者绝对是两条不同的路由、两个不同的路由前缀，因为他们的前缀长度不相同。

- **指向目标的指针：** 指针不是指向路由器的直连目标网络就是指向直连网络内的另一台路由器地址,或者是到这个链路的本地接口。更接近目标网络一跳的路由器叫下一跳(next hop)路由器。
- **路由信息的来源：** 本条路由是通过什么途径学习到的，例如是静态的，或者是通过 OSPF、IS-IS、EIGRP、BGP 等动态路由学习到的。

1.3 管理距离（Administrative Distance，简称 AD）

路由器可以通过多种途径获知路由条目：如静态手工配置、各种动态路由协议等等。当路由器从两种不用的途径获知去往同一个目的地的两条路由，那么路由器会比较这两条路由的 AD 值，也就是管理距离，优选 AD 值小的路由。如果 AD 值相等，例如是同种路由协议，则进一步比较 metric 值，当然，这其中还牵涉到不同的路由协议内在的工作机制问题，这就要针对不同的路由协议具体讨论了。如下图，R3 与 R1 运行的是 RIP 协议，R3 又通过 OSPF 与 R2 建立邻接关系。于是 R3 同时从 RIP 及 OSPF 学习到了去往目的地 1.1.1.0/24 的路由，这两条路由分别以 R1 和 R2 作为下一跳。那么 R3 最终选择 OSPF 的路由装载进路由表，也就是将 R2 作为实际去往 1.1.1.0/24 的下一跳，因为 OSPF 协议的 AD 值比 RIP 要小要更优。



Protocol	Destination Network	Exit Interface
O	192.168.1.0/24	e0

针对不同的路由协议，对应的 AD 值见下表，这是个众所周知的约定：

Routing Protocols	AD	备注

直连接口	0	
关联出接口的静态路由	1	Metric =0
关联下一跳的静态路由	1	Metric =0
EIGRP 汇总路由	5	
外部 BGP	20	
内部 EIGRP	90	
IGRP	100	
OSPF	110	
RIPv1、v2	120	
外部 EIGRP	170	
内部 BGP	200	

PS：当同一台路由器，有两条去往同一目的地的静态路由、分别使用的是出接口和下一跳的方式，这两条路由负载均衡（因为 AD 值和 metric 值都相等）。而使用关联出接口的方式配置的静态路由，该路由条目将作为直连网络输入到路由表中，具体请见本文档静态路由部分。

1.4 动态路由协议端口号或协议号

Routing Protocols	基于协议	备注
RIP	UDP	端口号 520
RIPng	UDP	端口号 521
EIGRP	IP	IP 协议号 88
OSPF	IP	IP 协议号 89
BGP	TCP	端口号 179

1.5 ICMP 重定向

R3 的网关为 R1，R1 本身又有静态路由到 2.2.2.2，下一跳为 R2，注意这是个 MA 网络，三个接口都是同网段的，这个很关键，R3 将去往 2.2.2.2 时，将数据丢给自己的网关 192.168.123.1，这时 R1 经过路由表查找后，发现数据的下一跳是与本地入接口同一个网段的 123.2，因此他认为 123.2（也就是 R2）比自己距离目标更近，因此给源也就是 R3 发送了一个 **ICMP 重定向消息**（要求 R1 的以太网接口开启 ip redirects），告诉 R3 所 192.168.123.2 为更优的下一跳，那么 R2 后续的报文将发给 R2。

R1 的配置如下：

```
ip route 2.2.2.0 255.255.255.0 192.168.123.2
```

R3 的配置如下：

```
ip route 0.0.0.0 0.0.0.0 192.168.123.1
```

在 R3 上开启 debug ip icmp，现在 R3 去 ping 2.2.2.2：

```
*Mar  1 00:11:47.887: ICMP: redirect rcvd from 192.168.123.1- for 2.2.2.2 use gw 192.168.123.2
*Mar  1 00:11:47.891: ICMP: echo reply rcvd, src 2.2.2.2, dst 192.168.123.3
*Mar  1 00:11:47.951: ICMP: echo reply rcvd, src 2.2.2.2, dst 192.168.123.3
*Mar  1 00:11:47.999: ICMP: echo reply rcvd, src 2.2.2.2, dst 192.168.123.3
*Mar  1 00:11:48.023: ICMP: echo reply rcvd, src 2.2.2.2, dst 192.168.123.3
*Mar  1 00:11:48.043: ICMP: echo reply rcvd, src 2.2.2.2, dst 192.168.123.3
```

另一方面我们抓包得到：

```
Internet Protocol Version 4, Src: 192.168.123.1 (192.168.123.1), Dst: 192.168.123.3 (192.168.123.3)
  Version: 4
  Header length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Trans
  Total Length: 56
  Identification: 0x0000 (0)
⊞ Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
⊞ Header checksum: 0x446f [correct]
  Source: 192.168.123.1 (192.168.123.1)
  Destination: 192.168.123.3 (192.168.123.3)
Internet Control Message Protocol
  Type: 5 (Redirect)
  Code: 1 (Redirect for host)
  Checksum: 0x9b5d [correct]
  Gateway address: 192.168.123.2 (192.168.123.2)
⊞ Internet Protocol Version 4, Src: 192.168.123.3 (192.168.123.3), Dst: 2.2.2.2 (2.2.2.2)
⊞ Internet Control Message Protocol
```

这就是 R1 给 R3 发送的 ICMP 重定向消息，注意里头的 Gateway address 字段，填写的就是比自己距离目的地更近的下一跳 IP。

如果要关闭 ICMP 重定向，需在接口上，使用 no ip redirects。

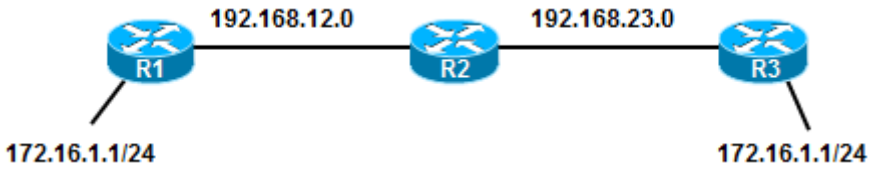
另外这里注意 ICMP 重定向的发送条件。同时注意，R3 上的这条默认路由，如果配置成关联出接口的方式，那么就无法观察到重定向的现象了，至于原因，你懂的，代理 ARP 嘛。

1.6 有类及无类路由查找方式

CISCO 路由器在路由的全局查找上有两种方式：有类（Classful）查找方式及无类（Classless）查找方式。

当路由器执行无类别路由查找时（默认，ip classless），它不会注意目的地址的类别，它会在目的地址和所有已知的路由之间逐位(bit by bit)执行最长匹配；

而如果是有类路由查找（no ip classless 且关闭 ip cef），那么收到一个数据包时，路由器先查找目的地址所属主类，如果路由表中有主类路由，则再去找子网，如果有子网路由，则查询被限定在这些子网中，并进一步查找，如果最终查找失败（没有任何子网匹配这条路由），则丢弃数据包，即使有默认路由存在；如果本地没有该主类路由，则看是否有默认路由，如果有，则按默认路由转发，如果无，则丢弃数据包。



在 R2 上做如下配置并做测试（**均在 no ip classless 且关闭 ip cef 环境下做的测试**）：

- **【实验 1】有主类路由；有默认路由，走主类路由**

```
ip route 0.0.0.0 0.0.0.0 192.168.23.3
ip route 172.16.0.0 255.255.0.0 192.168.12.1        !! 走主类路由
```

- **【实验 2】有子网路由（匹配）；有默认路由，走子网路由**

```
ip route 0.0.0.0 0.0.0.0 192.168.23.3
ip route 172.16.1.0 255.255.255.0 192.168.12.1      !! 走子网路由
```

- **【实验 3】无主类网络；有默认路由，走默认路由**

```
ip route 0.0.0.0 0.0.0.0 192.168.23.3               !! 走默认路由
ip route 172.17.0.0 255.255.255.0 192.168.12.1
```

- **【实验 4】有主类网络；有子网路由（匹配），按最长匹配**

```
ip route 172.16.1.0 255.255.255.0 192.168.12.1
ip route 172.16.0.0 255.255.0.0 192.168.23.3
```

- **【实验 5】有子网路由，子网路由前缀长度不一样（但都匹配），按最长匹配**

```
ip route 172.16.1.0 255.255.255.0 192.168.12.1
ip route 172.16.1.0 255.255.255.224 192.168.23.3    !! 按最长匹配，走这条
```

- **【实验 6】有子网路由，子网路由前缀长度不一样（匹配及不匹配均有），走匹配路由**

```
ip route 172.16.1.0 255.255.255.224 192.168.12.1        !! 匹配，走这条
ip route 172.16.1.32 255.255.255.224 192.168.23.3
```

- **【实验 7】有子网路由（不匹配）；有默认路由**

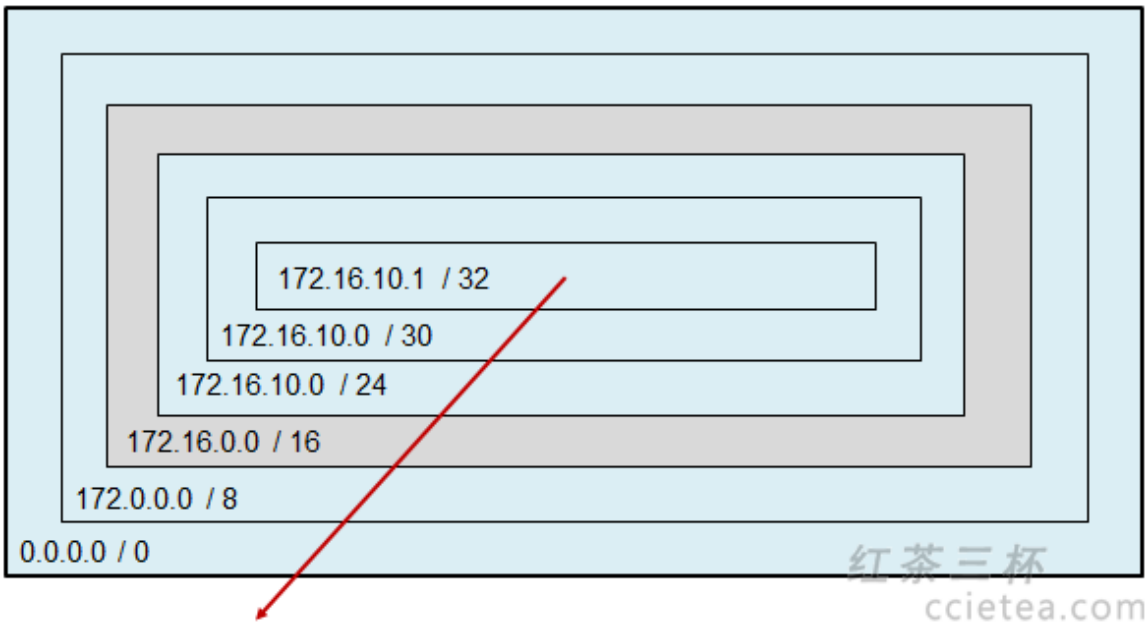
```
ip route 0.0.0.0 0.0.0.0 192.168.12.1
ip route 172.16.1.32 255.255.255.224 192.168.23.3
```

无法 ping 通，因为去往 172.16.1.1 查表后发现有子网路由，因此查找被限定在子网路由中，然后却发现这条路由不匹配，因此直接丢弃报文，而不会走默认路由。

PS：要注意将有类、无类路由查找方式，与有类、无类路由选择协议区分开来。

1.7 最长匹配原则

最长匹配原则是 CISCO IOS 路由器默认的路由查找方式。当路由器收到一个 IP 数据包时，会将数据包的目的 IP 地址与自己本地路由表中的表项进行 bit by bit 的逐位查找 ,直到找到匹配度最长的条目 ,这叫最长匹配原则。



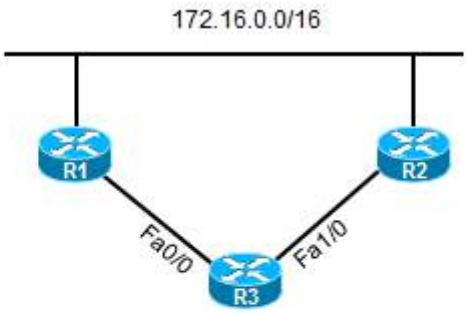
这里有几个概念要先搞清楚：

看上面的图，灰色的空间 172.16.0.0/16，这个网络号，我们称为**主类网络号**，所谓主类网络号，意思是该网络号，按照其所属的 IP 地址类别区分后，对应上的默认的子网掩码长度后得到的网络号。如 172.16.0.0 这是一个 B 类地址，B 类地址的默认子网掩码长度是 16 位，因此 172.16.0.0/16 本身就是一个主类网络号。再举过一个例子，10.1.12.0/24，首先 10 开头的，这是一个 A 类地址，A 类地址默认的掩码是 255.0.0.0，因此 10.1.12.0/24 它的主类网络号是 10.0.0.0/24。

我们首先顺着上面的图，从 172.16.0.0/16 开始往里走，下一个我们看到的网络号是 172.16.10.0/24，这很明显是应用了 VLSM 可变长子网掩码之后，得到的一个 172.16.0.0/16 这个主类网络的一个**子网**。所以所谓的子网，我们可以理解为是在网络号所属类别的默认掩码长度的基础上，将掩码“拉长”或者向主机位借位从而得到的一个网络号。实际上 172.16.0.0/16 是将 172.16.10.0/24 囊括在内的一个区间。那么在这里，如果我们有一个 IP：172.16.10.1，实际上这个 IP 既可以理解为在 172.16.0.0/16 网络内，也是在 172.16.10.0/24 网络内，当然，这里我们能看出来，谁更精确呢？很明显是 172.16.10.0/24 更精确，我们说，它的**匹配长度相比 172.16.0.0 更长**。

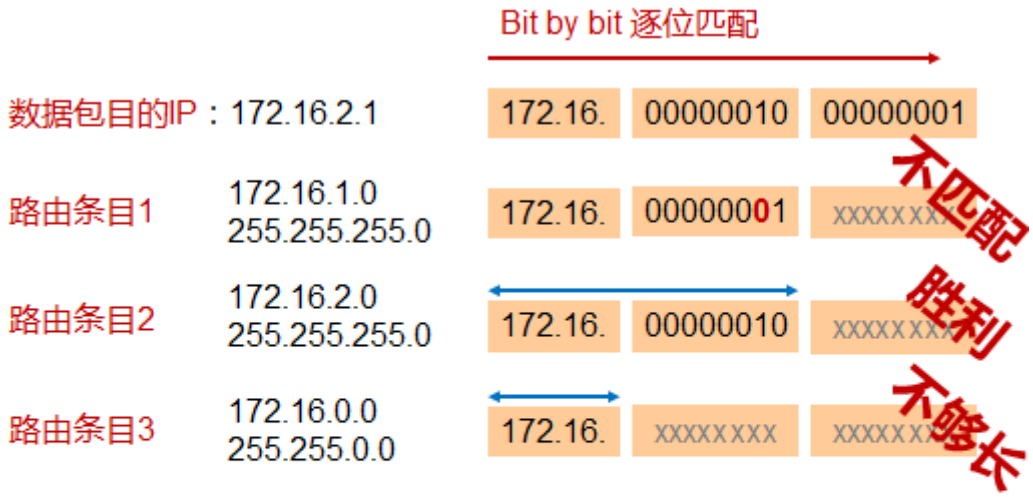
当然子网 172.16.0.0/16 还可以进一步划分子网，得到 172.16.10.0/30，甚至 172.16.10.1/32，那么如果这些前缀都存在的情况下，当我要去找 172.16.10.1，谁的匹配度最高呢？很明显，是 172.16.10.1/32 这条主机前缀，或者说，主机路由吧？这就是**最长匹配原则**。

OK，现在回到 172.16.0.0/16 这个主类网络号，然后我们向外走，看上图。172.0.0.0/8 实际上是将这个 B 类地址的掩码向左移了 8bits，这样一来得到的这个网络号实际上是囊括了 172.16.0.0/16 在内的一个大的网络号，我们称其为**超网**。




```
S  172.16.1.0/24 [1/0] via 192.168.13.1
S  172.16.2.0/24 [1/0] via 192.168.13.2
S  172.16.0.0/16 [1/0] via 192.168.23.1
```


因此，当路由器的路由查找方式为 classless 也就是无类路由查找方式时，路由器默认的查找动作是最长匹配原则。例如上图，当 R3 收到一个数据包，去往 172.16.1.1，那么实际上，172.16.1.1 是“掉落”在 172.16.1.0/24 及 172.16.0.0/16 网络中的，两者貌似皆可，但是 172.16.1.0/24 显然，匹配度要更长，因此，最终这个数据包被丢给了 R1。同理若有数据包去往 172.16.2.1 呢？由于根据最长匹配原则，172.16.2.0/24 这个条目匹配度最高，因此数据被扔给了 R2。这个过程有点类似下面的样子：




```
S  172.16.1.0/24 [1/0] via 192.168.13.1
S  172.16.2.0/24 [1/0] via 192.168.13.2
S  172.16.0.0/16 [1/0] via 192.168.23.2
```


并且，当 R2 挂掉之后，172.16.2.0/24 的条目失效，去往 2.0 子网的数据此时匹配的路由条目是 172.16.0.0/16 这条路由，因此被送往了 R1。

这就是利用最长匹配原则，实施的一种简单的数据分流及路径冗余的方法。

下面我们总结一下路由器关于路由查找的几个重点内容：

- 不同的前缀（网络号+掩码，缺一不可），在路由表中属于不同的路由
- 相同的前缀，通过不同的协议获取，先比 AD，后比 metric
  这是一般情况，当然有二般情况，这就要看特定的环境和特定的路由协议了
- 默认采用最长匹配原则，匹配，则转发；无匹配，则找默认路由，默认路由都没有，则丢弃
- 路由器的行为是逐跳的，到目标网络的沿路径每个路由器都必须有关于目的地的路由
- 数据是双向的，考虑流量的时候，要关注流量的往返。

2 CISCO 数据转发方式

2.1 数据转发方式

交换的概念：switching is the process of mapping layer 2 to layer 3 addresses andforwarding to a destination interface.

接下去我们来看看几种交换方式，本小节部分内容摘抄于网络，感谢原作者的分享。

1. **Process Switching(进程交换)**

   在这种模式下，一条数据流(flow)中的第一个包(packet)将被置入系统缓存(system buffer).其目的地址将会拿到路由表中去查询比对,路由器的处理器(CPU or Processer)同时将进行 CRC 校验,检查包是否正确.然后数据包的二层 MAC 地址将会被重写,替换为下一跳接口的 MAC 地址，这样的过程将会继续,对这条数据流(flow)中的第 2 个、第 3 个数据包.....相同的操作,包括查询路由表、重写 MAC 地址,CRC 校验等。这种方式无疑是延迟最大的,因为它要利用 system buffer 以及 processor 去处理每个收到的包.但是我们仍然有机会使用这种交换方式,比如在进行基于每个包的负载分担时,或是 debug ip packet 时。

   【启动进程交换】

   默认情况下,思科路由器会启用 fast switching 或 optimum switching 或是 cef switching,而不是 process switching,所以我们只能通过:no ip route-cache 来禁用 fast switching,这在另一种意义上正是开启 process switching.

2. **Fast Switching**

   快速交换要优于 process switching,它采用了 route cache(路由缓存)来存储关于某条数据流(flow)的特定信息,当然会包括诸如目的 MAC 地址,目的接口等内容.这时我们只需要对一条数据流(flow)中的第一个包做 process

switching,并把信息存入 cache,所有后续数据包,可以不必再中断 system processor 去执行查询等操作,直接从 cache 中提取目的接口,目的 MAC 地址等,这样大大加速了包转发速度。

【启动快速交换】

fast switching 在某些资料上可能被称为 route-cache switching

思科 1600、1700、2500、2600 系列路由器的 ethernet、fast ethernet、serial 接口默认采用的就是 fast switching.

我们可以用 ip route-cache 命令,在接口上启用 fast switching

show ip cache 来检查 fast switching 的相关信息.

3. Optimum and Distributed Switching

这两种交换模式,从原理上来讲都与 fast switching 极为相似,比如 optimum switching 其实采用了一种经过优化的交换缓存(optimumed switching cache),它的速度要较平常 cache 要快.distributed switching mode 需要使用 Versatile Interface Card 这种硬件卡,又称 VIP card.它会自己保存一份 route cache,这样在查询时就不必要等待使用共享的系统缓存了(shared system buffer),无论相对于 fast switching 还是 optimum switching 来讲,都是比较快的.

这两种模式一般只在思科高端设备上有所应用,比如 7200 系列路由器.或者 12000 系列路由器.

命令:ip route-cache optimum show ip cache optimum

4. Netflow switching

这种模式是最值得参考的,它完全基于其它 switching mode,重点在于对流经的数据包进行计费、监控、网管.但不得不提的是,这种模式因为也要存储相关信息,经过统计,大致 65536 条数据流(flow)会耗费 4MB 的 system buffer.

相关命令:

ip route-cache flow

show ip cache flow

ip flow-export 将 NETFLOW 审计的数据包转发到指定设备.

5. Cisco Express Forwarding

思科 CEF 是最为高效的一种三层协议。CEF 采用了基于硬件的平台,它不仅仅是将数据都存入 system buffer,而是将整个路由表、拓扑表,以及所有的下一跳地址、MAC 地址全部进行"预存",只要路由表、拓扑表中存在的条目,无论是否有数据请求发往其目的地址,都会提前预读取,预设置缓存.这样,当有新的数据请求发送时,就不需要 CPU 去查询目的接口,目的 MAC 地址等等信息,而是直接从缓存中读取,从而使转发速度得以大大提高.

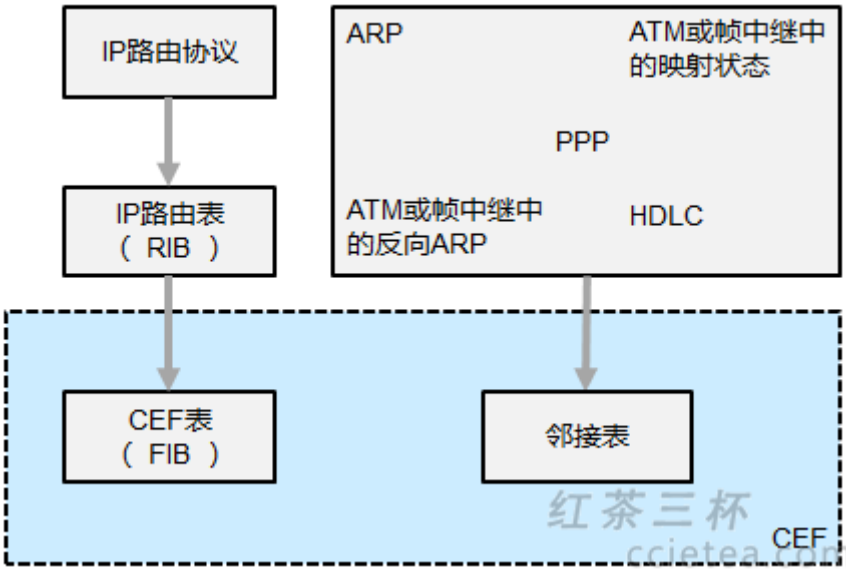
相关命令:ip route-cache cef

show ip cef

show ip cef detail

路由器通常根据入站接口和源与目的地址类型确定是否使用 CEF 交换. 对于考虑使用 CEF 的路由器来说,出站接口必须配置为 CEF 交换模式,如果接口上配置了 CEF,那么 CEF 将尝试交换数据包。否则,CEF 将会把数据包交付给仅次于最好的可用交换方法去处理。

CEF 有哪些组件：



- **邻接表**

CEF 组件中的邻接表用于 MAC 或者第二层重写信息。

- 当路由器与主机邻接的话，他们通过某些方式学习。
- 当路由器通过点对点链路直连，每一个接口上只有一个邻居存在。
- 当路由器通过以太网这样的多路访问介质连接，需要一种动态的机制来发现对方，例如 ARP，ARP 可以将二层地址映射到 IP。如果是帧中继多点链路，那么邻接关系可以用过地址解析协议、映射表等学习到。

尽管决策如何转发报文的工作由 FIB 来实现，但是第二层的帧重写工作却是根据邻接表中的信息来完成，第二层需要重写的字段包括用于帧转发的第二层头部。对于以太网来说，就是新的源、目 MAC，以及类型字段。例如：

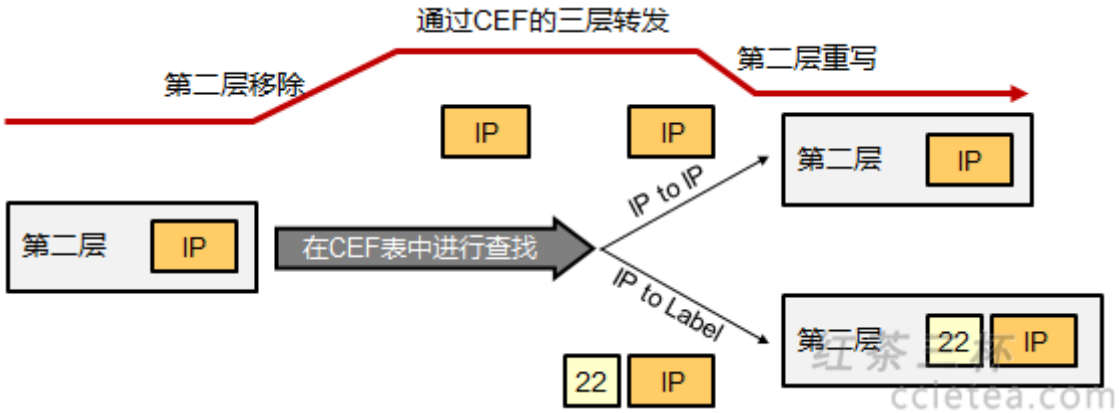
```
R2#show adjacency detail
Protocol Interface                 Address
IP       FastEthernet0/0           10.1.12.1(10)
                                   14 packets, 7640 bytes
                                   epoch 0
                                   sourced in sev-epoch 1
                                   Encap length 14
                                   CA004FEC0008CA014FEC00080800
                                   ARP
```

上面输出中的红色字体部分就是二层的 MAC 信息等等。

- **CEF 表**

  进行三层转发决策。CEF 表维护着从路由表中提炼的核心信息，这些信息用于对接收到的报文执行转发决策，这些信息包括 IP 前缀、递归的下一跳地址和出站接口。CEF 表的一个重要的功能是它能够立即决策出递归前缀。

- **CEF 的实施**



当报文进入路由器的时候，路由器将第二层信息剥掉，随后在 CEF 表中查找目的 IP 地址，以进行转发。

转发决策的结果将会指向邻接表中的一条邻接条目，在邻接表中可以找到第二层需要重写的字符信息，路由器使用这些信息为数据帧构造一个新的第二层头部，然后再将该报文转发到指向下一跳的出站接口。

2.2 负载均衡方式

1. **基于 CEF**

   在 CEF 中，有两种主要的负载均衡方式，基于报文和基于目的。

- **（接口）ip load-sharing per-packet**

  基于报文的负载均衡 ,所有报文在出战链路上进行轮循。如果你想要使用基于报文的 CEF 负载均衡的话，就需要在所有出站接口上都配置该命令。哥们实验验证过了，确实可以 per-packet 负载均衡。

- **（接口）ip load-sharing per-destination**

  默认的负载均衡方式是基于目的地的负载均衡。CEF 的基于目的地负载均衡实际上是通过对目的和源 IP 地址进行 HASH 后实现的。相反，在快速转发中，基于目的地址负载均衡是严格按照目的 IP 地址来进行的。CEF 默认使用基于目的地址的负载均衡方式，因为基于报文的负载均衡方式可以持续性地在

不同路径中发送同一数据流的报文，那么在 IP 报文到达目的地的时候可能需要进行重新排队的工作。这可能会降低对诸如 VoIP 这样的流量的转发性能，或者报文如果不按顺序到达的话，服务质量也会降低，报文可能被丢弃，另外还会增加延迟抖动。

你可以使用命令 show cef interface 来检查使用了哪一种负载均衡模式，该命令可以给出在这个接口上配置的 CEF 信息。

```
R2#show cef interface f0/0
FastEthernet0/0 is up (if_number 4)
  Corresponding hwidb fast_if_number 4
  Corresponding hwidb firstsw->if_number 4
  Internet address is 10.1.12.2/24
  ICMP redirects are always sent
  Per packet load-sharing is disabled
  IP unicast RPF check is disabled
  Inbound access list is not set
  Outbound access list is not set
  IP policy routing is disabled
  BGP based policy accounting on input is disabled
  BGP based policy accounting on output is disabled
  Hardware idb is FastEthernet0/0
  Fast switching type 1, interface type 18
  IP CEF switching enabled
  IP CEF switching turbo vector
  IP CEF turbo switching turbo vector
  IP prefix lookup IPv4 mtrie 8-8-8-8 optimized
  Input fast flags 0x0, Output fast flags 0x0
  ifindex 3(3)
  Slot    Slot unit 0 VC -1
  Transmit limit accumulator 0x0 (0x0)
  IP MTU 1500
```

在 CISCO IOS 中，CEF 可以对源和目的 IP 地址进行哈希，将哈希的结果导入一张负载均衡表来实现负载均衡。这张表有 16 个哈希桶 bucket，每一个哈希桶都指向一个邻接关系，而多个桶可以同时指向一个邻接关系。可以使用 show ip cef 路由前缀 internal 这条隐藏命令来看：

R1#show ip cef 2.2.2.2 internal

```
2.2.2.2/32, version 55, epoch 0, per-destination sharing
0 packets, 0 bytes
  tag information set
    local tag: 100
  via 10.1.13.3, FastEthernet1/0, 0 dependencies
    traffic share 1
    next hop 10.1.13.3, FastEthernet1/0
    valid adjacency
    tag rewrite with Fa1/0, 10.1.13.3, tags imposed: {}
  via 10.1.12.2, FastEthernet0/0, 0 dependencies
    traffic share 1
    next hop 10.1.12.2, FastEthernet0/0
    valid adjacency
    tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {}

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
                internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface          Address         Packets  Tags imposed
  1     Y   FastEthernet1/0    10.1.13.3       0        none
  2     Y   FastEthernet0/0    10.1.12.2       0        none
  3     Y   FastEthernet1/0    10.1.13.3       0        none
  4     Y   FastEthernet0/0    10.1.12.2       0        none
  5     Y   FastEthernet1/0    10.1.13.3       0        none
  6     Y   FastEthernet0/0    10.1.12.2       0        none
  7     Y   FastEthernet1/0    10.1.13.3       0        none
  8     Y   FastEthernet0/0    10.1.12.2       0        none
  9     Y   FastEthernet1/0    10.1.13.3       0        none
  10    Y   FastEthernet0/0    10.1.12.2       0        none
  11    Y   FastEthernet1/0    10.1.13.3       0        none
  12    Y   FastEthernet0/0    10.1.12.2       0        none
  13    Y   FastEthernet1/0    10.1.13.3       0        none
```

```
14      Y   FastEthernet0/0        10.1.12.2          0     none
15      Y   FastEthernet1/0        10.1.13.3          0     none
16      Y   FastEthernet0/0        10.1.12.2          0     none
refcount 6
```

要在基于目的的负载均衡中检验特定的流量使用的是哪一个出站接口，使用

Show ip cef exact-route src-addr dest-addr

例如前往一个目的地可能有多条等价路径，那么实际上可能在转发数据的时候，只是用了其中一条，用这条命令可以查看实际用于流量转发是哪一个接口。

2. 基于快速转发

快速交换意味着所有去往指定目的地的数据包都从相同的接口被发送出去,因此交换时间和处理器的占用率会大大降低。当去往相同网络内不同主机的数据包进入路由器且还一存在条可选路由时,路由器会在另一条路径上发送数据包到目的地。因此路由器能够做得最好的就是基于目标网络的均衡负载。

3. 过程交换

基于数据包的负载均衡和过程交换。过程交换就是对于每个数据包,路由器都要进行路由表查询和接口选择,然后再查询数据链路信息。因为每一次为数据包确定路由的过程都是相互独立的,所以不会强制去往相同目标网络的所有数据包使用相同的接口。为了在接口上打开过程交换功能,可以在接口下使用命令 no ip route-cache。对于 IPv6 什么也不需要做,因为缺省情况下该功能是打开的。

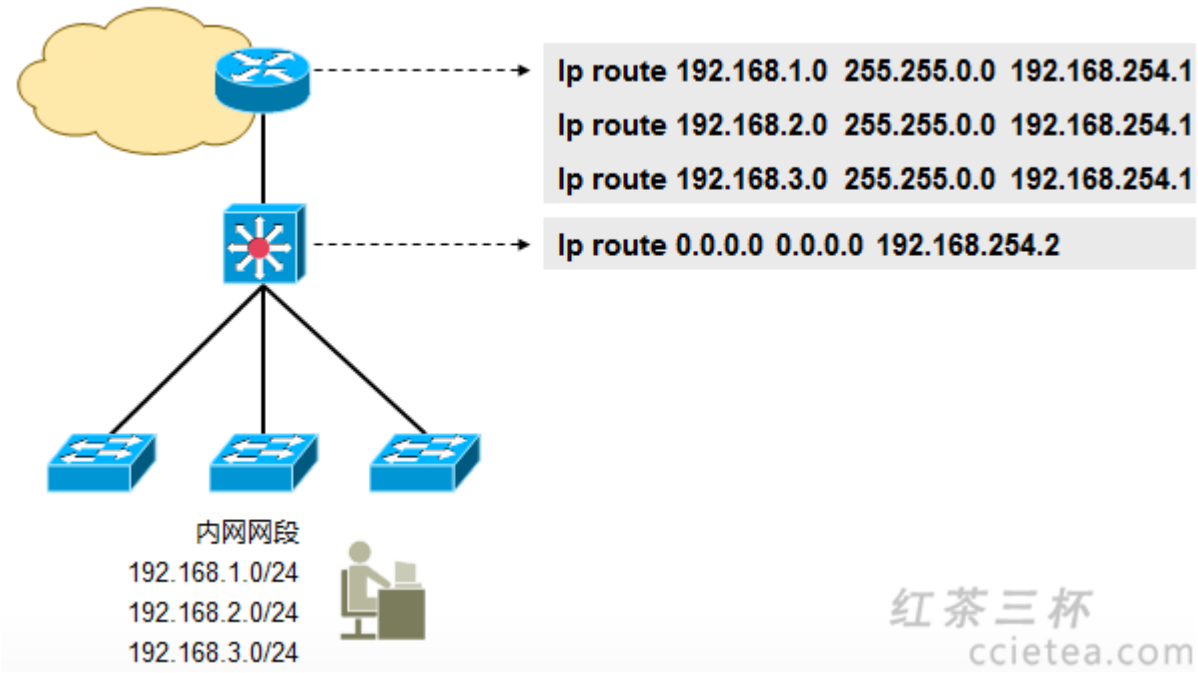
4. 哪一种交换方式会被用到

表 3-1 IOS 根据入站和出站接口的配置确定交换方法

入站配置	出站配置	所用的交换方法
CEF	过程	CEF
CEF	快速	CEF
过程	CEF	快速（如果是 IPv6 为过程）
过程	快速	快速
快速	CEF	快速（如果是 IPv6 为过程）
快速	过程	过程

3 静态路由

3.1 概述



路由器的天职，就是维护路由表以及利用路由表进行数据转发。而路由表中包含通过各种途径学习到的路由表项或路由条目，其中最简单最直接的方法，就是使用静态手工配置的方式，为路由器创建路由条目，这种方式最直接，可控性最高，配置也最简单。在小型的网络中，全网静态路由似乎没有什么问题，但是在一个大型网络中，如果纯用静态路由来做，工作量就非常大了，不仅仅工作量大，另外一个更重要的缺陷是静态路由无法根据网络拓扑结构的变更而做出调整，因此，在大规模网络中，我们往往采用静态+动态路由协议的方式来完成路由的部署。

3.2 静态路由的配置

```
Ip route 192.168.10.0 255.255.255.0 192.168.1.1
```

- 使用指向下一跳的静态路由

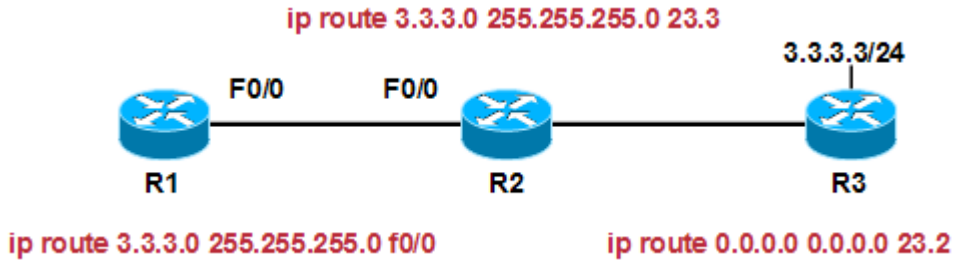
```
Ip route 192.168.10.0 255.255.255.0 fast 0/0
```

- 使用关联出接口的方式配置静态路由
- 该条目将作为直连网络输入到路由表中（如下）
- 如果出接口为广播型接口，可能会给接口下的节点造成额外的负担（ARP）或者，造成潜在的问题

```
Router#show ip route
Gateway of last resort is not set
     9.0.0.0/24 is subnetted, 2 subnets
C       9.9.12.0 is directly connected, FastEthernet0/0
S       9.9.23.0 is directly connected, FastEthernet0/0
```

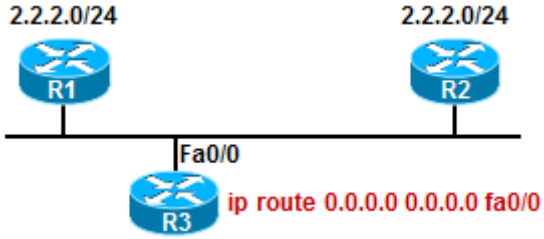
3.3 注意事项

1. 关联出接口的静态路由



正常情况下，R1 是能够 ping 通 3.3.3.3 的，但是如果把 R2 f0/0 口的代理 arp 关掉的话，就 ping 不通了使用出口关联静态路由的话，路由器会将路由目的地址认为是本地链路，因此发 arp 请求 3.3.3.3 的 mac，而 R2 的 F0/0 口配置了代理 ARP，3.3.3.0 的网络 R2 本身又可达，因此 R2 会用自己 F0/0 口的 MAC 来回应这个 ARP 请求，实际上，这是一种 ARP 欺骗行为：)

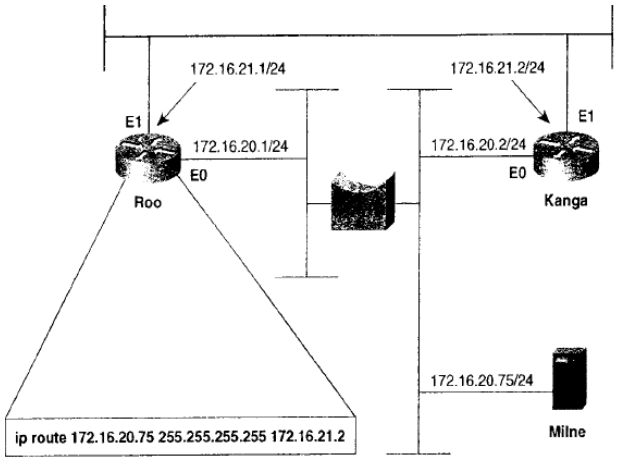
关联出接口的静态路由的另一个 CASE：



R1、及 R2 上均连接到 2.2.2.0 网络，此时 R3 去 ping2.2.2.2，由于使用的使出接口的静态路由，R3 认为 2.2.2.0 是本地连接的网络，因此发 arp 请求，R1、R2、均能收到请求报文，且均回复请求报文（前提两

者的 f0/0 口都开了 arp 代理），对于 R3，会 R1、R2 中，较晚到达的 arp 回应报文中的目的 MAC 放入 arp 表中，关联 2.2.2.2（架设为 R1），此后但凡去往 2.2.2.2 的数据包，均直接使用 R1 的 f0/0 口的 MAC 地址进行封装，不再发 ARP 请求。而当 R1 的 f0/0 口 DOWN 时，在 R3 的 2.2.2.2 的 ARP 条目超时之前，R3 均无法 ping 通 2.2.2.2，直到 arp 条目超时，或手工清除 2.2.2.2 的 arp 条目。

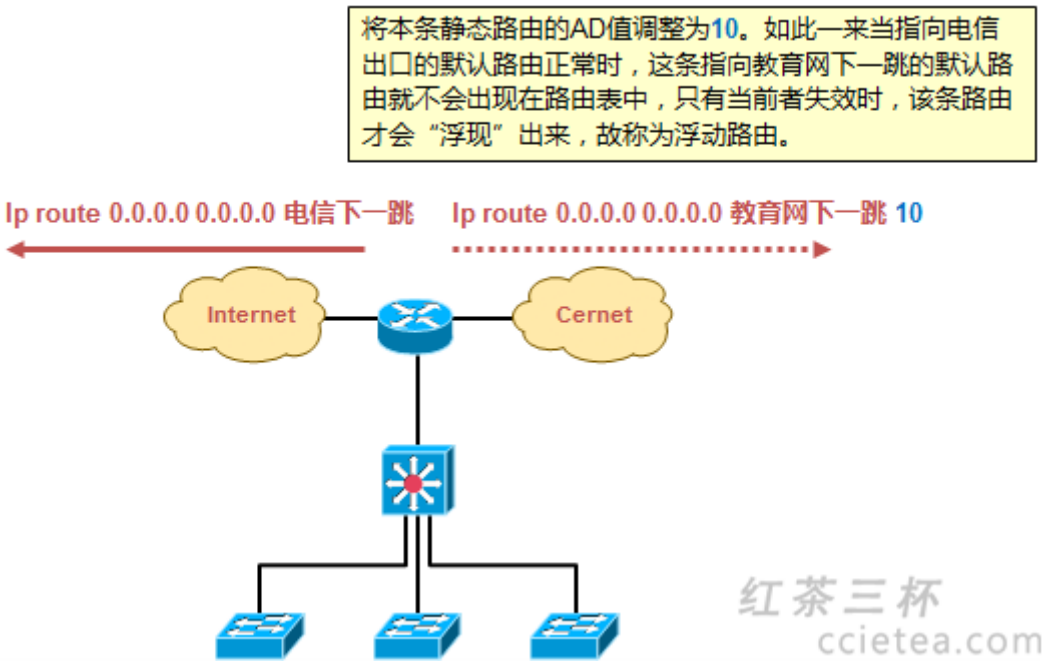
2. 代理 ARP 的另一个问题（来自 TCP/IP 路由技术卷一）



ROO 上添加一条指向主机 Milne 的静态路由，下一跳为 Kanga 的 E1 口，目的是为了通过时常拥塞的网桥。但是却发现 Kanga 将发向主机 Milne 的数据包发给 Roo，原因是学习到错误的 MAC 地址。

Kanga 发 arp 请求，请求 Milne 的 mac，这时 Roo 也收到该请求，而其从 172.16.20.0 网段收到的请求，同时查表发现自己通向主机 Milne 的路由所在的网络（172.16.21.0）与收到 arp 请求包的网络不一样，并且接口开了 arp 代理，因此回复自己的接口 mac.

3.4 浮动静态路由



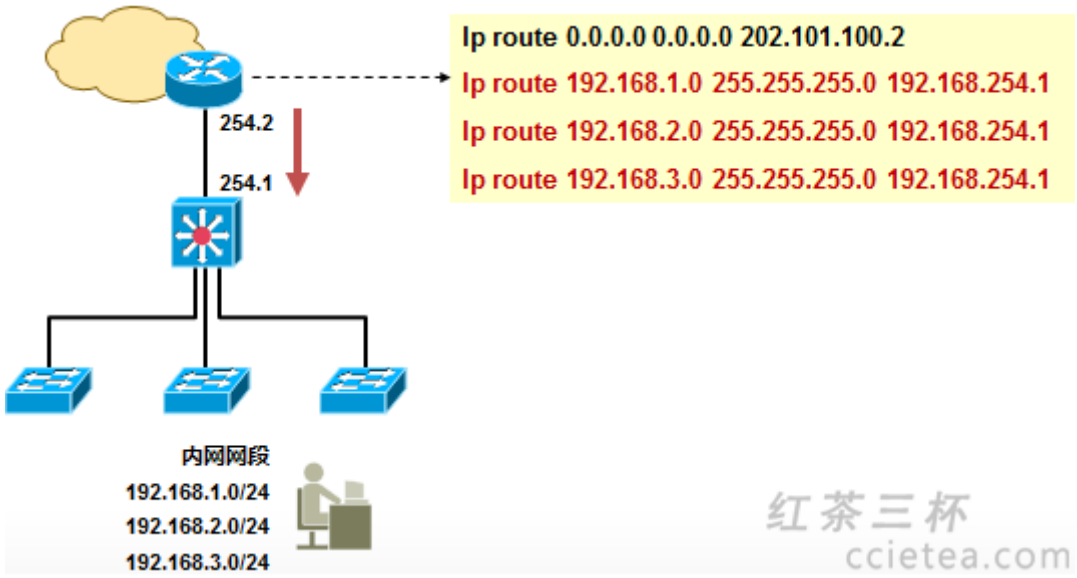
我们看上面的例子，在一个典型的教育园区网中，我们往往有两条以上的外网出口线路，假设一条为电信，一条为教育网出口。那么我们一般为出口路由器添加一条默认路由，指向电信的下一跳地址，为的是让内网用户能够通过电信线路访问 Internet 资源。然而 ,如果电信出口出现故障呢？我们可以在路由器上增加一条默认路由，我们知道，如果你配置两条默认路由，分别关联两个不同的下一跳，那么这两条路由将会在路由表中进行负载均衡，但是这里我们并不希望出现这个现象，我们希望一主一备，那么这条新增的默认路由就可以这么来配置：

```
Ip route 0.0.0.0 0.0.0.0 教育网下一跳 IP 10
```

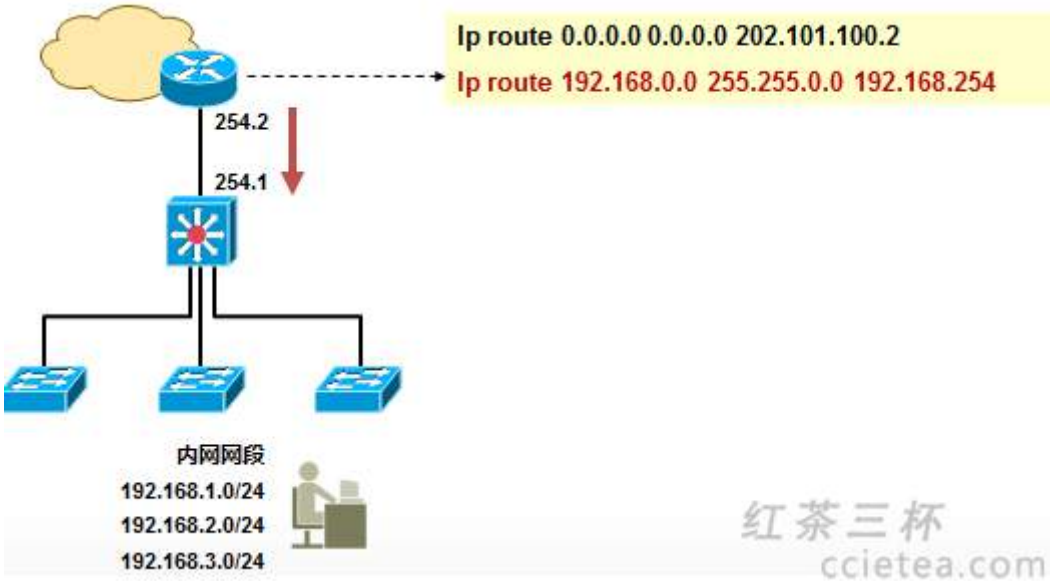
我们知道静态路由的 AD 值是 1，上面的配置方法，实际上是将该条静态路由的 AD 值修改为 10。这样一来我们有两条默认路由，一条指向电信出口，AD 值为默认的 1；另一条指向教育网出口，AD 值为 10。那么经过 PK 之后，毫无疑问指向电信的默认路由出现在了路由表里，而指向教育网这条默认路由，“猥琐”的躲了起来。当指向电信的默认路由失效的时候它就从路由表里消失了，那么这时候，指向教育网的这条默认路由，就“浮”了出来。

3.5 路由汇总

1. 路由汇总技术背景



我们看上面的场景，重点看出口路由器，这台出口路由器由于和三层交换机之间是三层链路，因此需配置到内网的回程路由，也就是上图中红色字体部分，这个场景中内网只有三个网段，因此配置了三条静态路由，但是如果有 100 个网段呢？岂不是要配 100 条路由？如此一来路由表就变的非常庞大和臃肿，维护和管理和非常不方便，更重要的是，这无疑浪费了设备的资源。因此从网络优化的角度，不管是何种网络场景何种网络模型，我们都需时刻关心网络中路由器路由表里的路由条目数量，是否足够优化，是否有可优化的空间。咋办？

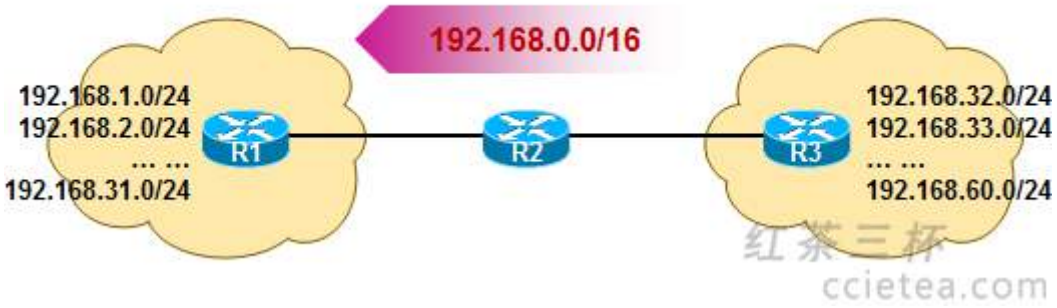


用路由汇总！看上图，前一个场景，我们需使用 3 条精细路由，而这里，我们却仅仅使用一条路由即可实现相同的效果，这条路由是上一个场景中三条明细路由的**汇总路由**。这样配置的一个直接好处就是，路由器的路由表条目大大减少了。这种操作方式我们称为路由汇总。路由汇总是一个非常重要的网络设计思想，通常在一个大中型的网络设计中，必须时刻考虑网络及路由的可优化性，路由汇总就是一个我们时常需要关

注的工具。这里实际上是部署了静态路由的汇总，当然除此之外我们也可以在动态路由协议中进行路由汇总，几乎所有的动态路由协议都支持路由汇总。

2. 路由精确汇总的算法

路由的汇总实际上是通过对子网掩码的操作来完成的。对于下面的例子来说：



在 R2 上，为了到达 R1 下联的网络，R2 配置了使用路由汇总的工具，指了一条汇总路由：192.168.0.0/16 到 R1，虽然这确实起到了网络优化的目的，但是，这条汇总路由太“粗犷”了，它甚至将 R3 这一侧的网段也囊括在内，我们称这种行为不够精确。因此，一种理想的方式是，使用一个“刚刚好”囊括这些明细路由的汇总路由，这样一来就可以避免汇总不够精确的问题。

这里不得不强调一点，网络可以部署路由汇总的前提是 IP 子网及网络模型设计具备一定的科学性和合理性，因此路由汇总和网络的 IP 子网及网络模型的设计是息息相关的。

那么如何进行汇总路由的精确计算呢?下面我们来看一个例子：

假设我们有这么几个子网：

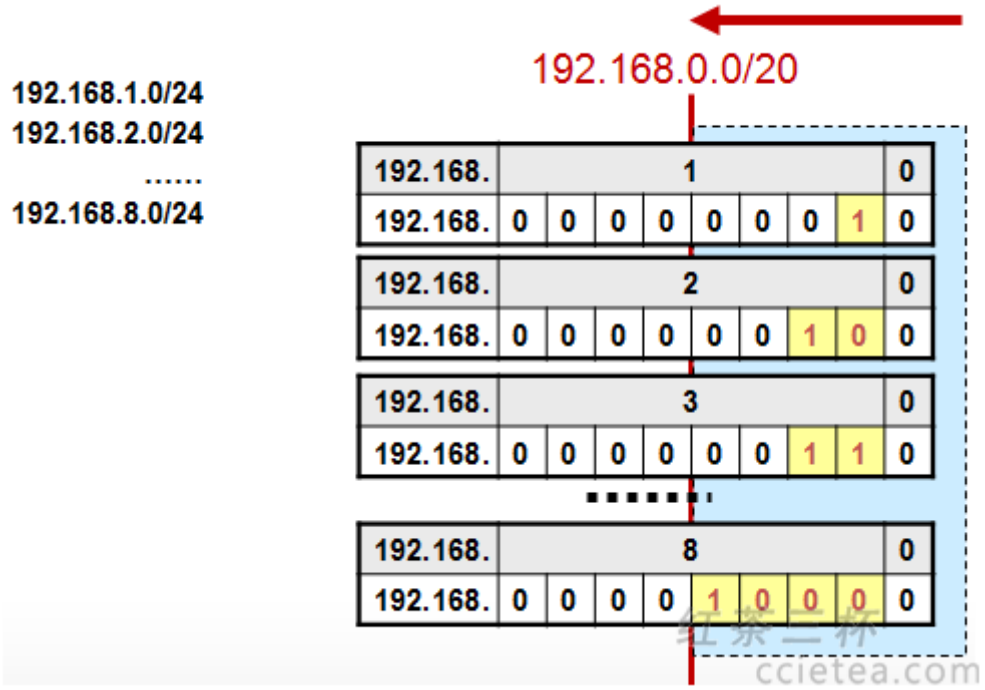
192.168.1.0/24

192.168.2.0/24

……

192.168.8.0/24

现在，我们要经过计算，得出刚刚好“囊括”这几个明细的汇总网段。



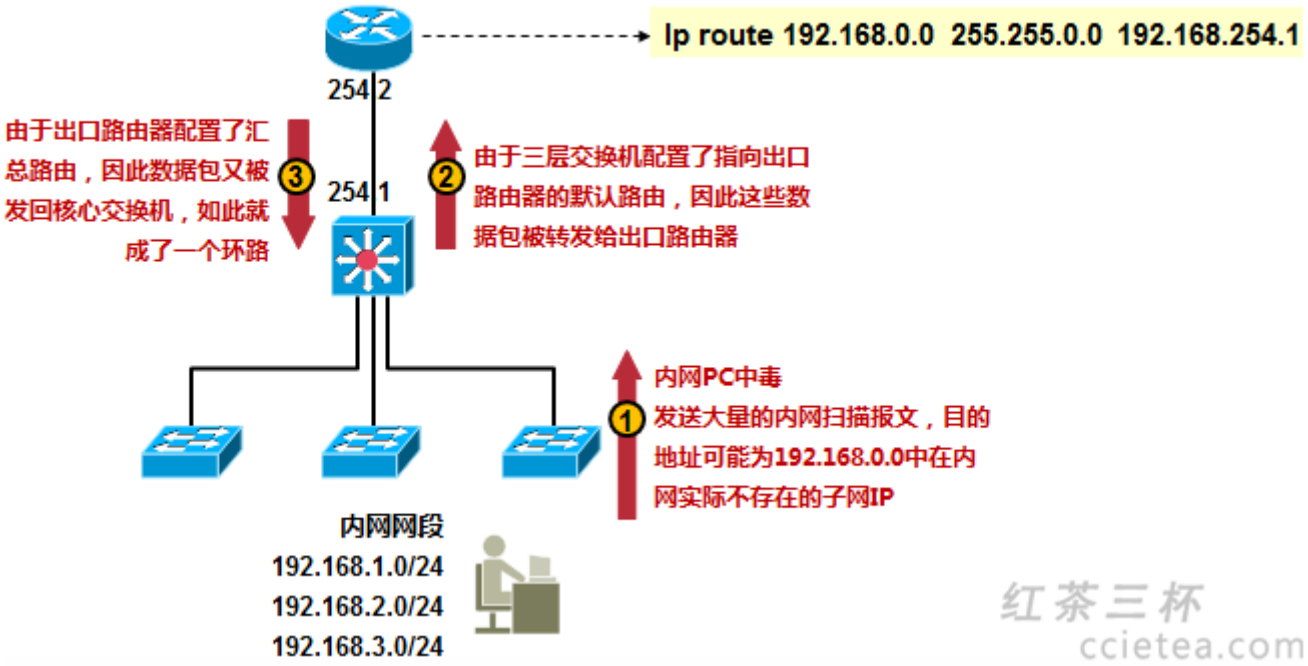
我们要做的事情非常简单 ,这些个明细子网是连续的 ,我们只要挑出首位的两到三个网络号来计算就足够了：

1）将他们写成二进制形式，实际上，我们只要考虑第三个 8 位组即可，因为只有它是在变化的

2）现在，我们要画一根竖线，这根线的左侧，每一个列二进制数都是一样的，线的右侧则无所谓，可以是变化的，注意这根竖线，可以从默认的掩码长度，也就是/24 开始，一格一格的往左移，直到你观察到线的左端每一列数值都相等，即可停下，这时候，这根线，所处的位置就是刚刚好。

3）如上图 ,线的位置是 16+4=20 ,所以我们得到汇总地址 :192.168.0.0/20 ,这就是一个最精确的汇总地址。

3. 路由汇总需注意的问题



路由汇总虽然确实是一个非常非常重要的思想和工具，但是使用起来要持谨慎态度，毕竟减少路由条目的同时，也降低了路由的颗粒度和精确性。看上图，在出口路由器上配置了静态汇总路由，下一跳是三层交换机。而三层交换机为了将访问外网的流量送到出口路由器，配置了一条默认路由，下一跳是出口路由器。

这个网络在流量正常的情况下不会有问题，但是，现在内网用户中毒了，于是这些 PC 开始疯狂的发送内网的扫描报文，这些报文的目的地址是一些 192.168 打头的不知名地址，甚至根本不存在的地址。数据包被送到了网关也就是三层交换机上，由于三层交换机配置了默认路由，因此这些数据包目的地被默认路由匹配并被引导到了出口路由器上，而出口路由器上部署了汇总路由，这些数据包的目的地址虽然在内网中不存在，但是却是这个汇总路由里的一个 IP，因此又被出口路由器转发回给三层交换机，接下去三层交换机又根据默认路由，将数据包转发回出口路由器，这就形成了数据的环路。

因此，从这里我们可以看出来，路由汇总，是有产生环路的风险的，解决上述问题的一个办法就是，我们在三层交换机上，增加一条静态路由：ip route 192.168.0.0 255.255.0.0 null 0，这样一来，当它收到访问 192.168 开头的、不存在的目的地址的数据包，就会直接丢弃。而正常的访问 192.168 内网其他子网的流量会根据最长匹配原则被正常转发。

这个思想被应用到了诸如 OSPF 等这类动态路由协议上，细心的童鞋会发现，你在 OSPF 中部署了路由汇总后，它会自动在本地产生一条指向 null0 的汇总路由，道理跟上面讲解的是一样的。

4 距离矢量路由协议

4.1 Basic

1. 距离矢量名称的由来是因为路由以矢量（距离，方向）的方式被通告出去的，其中距离是根据度量定义的，方向是根据下一跳路由器定义的
2. 定期更新、广播更新、路由表更新；依照传闻；距离矢量路由协议并不了解网络拓扑
3. 定义一个最大值：定义跳数的最大值 15 跳，16 跳为不可达，以避免路由选择环路
4. 通过水平分割消除路由选择环路
   简单水平分隔的规则是，从某接口发送的更新消息不包含从该接口收到的更新所包含的网络，换成人话说，就是我从这个接口收到的路由，就不会再从这个接口发回去。
5. 路由中毒

   当 10.4.0.0 挂掉的时候，C 会立即发一条路由中毒消息（10.4.0.0　16 跳）然后通告出去；
   B 收到这条中毒消息后，将 10.4.0.0 从路由表里抹去，但仍存在在 rip database 里，状态是 possible down，垃圾收集时间（Garbage colletion CISCO 默认 60S）到后，路由被从 B 的 database 抹去。
6. 毒性逆转
   同上图，4.0 挂掉后，C 会发送中毒消息消息，理论上 4.0 的路由是 C 通告给 B 的，根据水平分割原则，B 不能向 C 通告 4.0 的信息，但是带毒性逆转的水平分割打破了这个原则，B 会定期向 C 发送 4.0 的毒性逆转消息，以让 C 知道，他的邻居晓得了 4.0 挂掉的消息并且在眼巴巴的等着 4.0 原地满血复活，这样做的另一个好处是避免环路，至少 C 不会从 B 再去访问 4.0 了。
7. 用触发更新避免路由选择环路
   新的路由表一般是定期发送给邻居路由器的，而触发更新(triggered update) 则是立即发送以响应路由表的变化。
8. 用抑制定时器(hold-down timer)防止路由选择环路
   当一个 router 从邻居 router 收到一条更新，指示以前可达的网络现在不可达了，或有一个更大跳数的路由，则这个 router 标该路由为不可达并启动一个抑制定时器，如果在定时器满以前收到该路由又可达的更新，或者比以前的记录有更好的度量值，则该 router 标识这个路由可达并删除定时器。

4.2 RIPv1

4.2.1 Summary

1. V1 使用广播，默认每 30s 发送一次，采用 UDP 520 端口（源、目的端口）
   V2 使用组播 224.0.0.9
2. RIP 定义了请求消息和响应消息两种类型
   请求消息：可以请求整张路由表，也可以请求具体路由信息。
   - 请求整个路由表：请求消息含有一个地址族标识字段为 0（地址为 0.0.0.0），度量值为 16 的单条路由，接收到这个请求的设备将通过单播方式向发出请求的地址回送它的整个路由表。
   - 请求具体路由信息：当需要获知某个或某些具体路由的信息，请求消息将与特定地址的路由条目一起发送。接受到请求的设备将根据请求消息逐个处理这些条目，并构成一个响应消息。

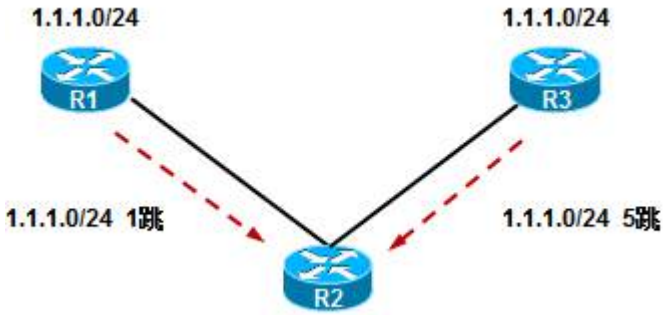
   更新消息：包含了整张路由表，周期性地更新
3. RIP V1 支持最大 6 条，默认 4 条等价负载均衡。并且不支持验证。 V2 支持验证。

4.2.2 Timers

Update timer 更新计时器	默认 30s，RIP 更新周期
Invalidation timer 无效计时器	默认 180s，也可称为 expiration timer 限时计时器或 timeout timer 用来限制停留在路由表中的路由未被更新的时间。 这个时间到了后，路由条目会变成 16 跳，标记不可达路由 possibly down。 Gateway of last resort is not set R 3.0.0.0/8 is **possibly down**, routing via 9.9.12.2, FastEthernet0/0 这个时候，他自己到如果收到数据包去 3.0.0.0 依然转发，但是不会向其他 rip 路由器去更新 3.0.0.0 的路由
Garbage colletion/flush timer	一般为比无效计时器多 60-240 秒，如果连这个时间也超时了，那么路由条目彻底删除。思科默认 60s（注意是比 invalidtimer 多 60s），RFC 默认 120s
Holddown timer	默认 6 个更新周期，即 180s。当收到一个更大跳数的条目时，该路由条目标记为不可达，同时启动抑制计时器，如果计时器超时后，同一个邻居仍然通告该

	路由，则接受更新。

- **计时器验证**



如上图，R1 向 R2 更新 1.1.1.0，1 跳，当 R1 DOWN 掉，而 R3 向 R2 更新 1.1.1.0 为 5 跳时， R2 上去往 R1 的路由会随着 invalid 计时器的到期变成 pdown 状态，随后进 holddown timer 的计时器

```
C       192.168.12.0/24 is directly connected, Serial0/0
        1.0.0.0/24 is subnetted, 1 subnets
R           1.1.1.0/24 is possibly down,
              routing via 192.168.12.1, Serial0/0
```

另外，如果 R1 更新 1.0.0.0 给 R2，如果 R1passive 掉与 R2 互联的接口，invalid 到期后，R2 上关于 1.0.0.0 的路由会 pdown，这时 R2 仍然可以去往 1.0.0.0，但是不会将本路由传递给其他邻居，相反是仍然不停的发送中毒消息。这时候如果 no passive 掉 R1 的这个接口 ,R2 继续收到更新 ,但它不会马上恢复该路由 ,而是进入 holddown 计时器，直到 holddown 计时器超时后，才会接受该路由的更新。

Holddown timers 在 IOS 某些版本中被忽略 ,也就是即使设置了也不生效。

4.2.3 RIP database

每个运行 RIP 的路由器管理一个路由数据库，该路由数据库包含了到网络所有可达信宿的路由项，这些路由项包含下列信息：

- 目的地址：主机或网络的地址。
- 下一跳地址：为到达目的地，需要经过的相邻路由器的接口 IP 地址。
- 接口：转发报文的接口。
- Metric 值：本路由器到达目的地的开销，是一个 0～15 之间的整数。
- 路由时间：从路由项最后一次被修改到现在所经过的时间，路由项每次被修改时，路由时间重置为 0。

- 路由标记：区分内部路由协议路由和外部路由协议路由的标记。

4.2.4 RIP 消息格式

	8	8	8	8
	命令	版本	未使用	
路由条目最大 25 条	地址族		未使用	
	IP 地址			
	未使用（设置为全 0）			
	未使用（设置为全 0）			
	METRIC			
	.....（最大 25 条）			

字段名	长度	含义
Command	8 比特	标识报文的类型： 1：Request 报文，向邻居请求全部或部分路由信息； 2：Reponse 报文，发送自己全部或部分路由信息，一个 Response 报文中最多包含 25 个路由表项。
Version	8 比特	RIP 的版本号： 1：RIP-1； 2：RIP-2。
Must be zero	16/32 比特	必须为零字段。
AFI(Address family identifier)	16 比特	地址族标识，其值为 2 时表示 IP 协议。
IP Address	32 比特	该路由的目的 IP 地址，只能是自然网段的地址。
Metric	32 比特	路由的开销值。

每个 RIP 消息包含路由条目，最多 25 条，超过则需多条消息。

4.2.5 Ip rip trigger

在接口上配置了该命令后，便不在该方向周期性更新路由表，而是触发更新。路由表的更新将会变得最少,仅仅包括路由表最初的交换信息和路由表发生变化时的更新信息。这条命令仅仅在串行链路上有效，并且必须在链路的两端同时配置才会产生效果。

因为 RIP 出现的比较早，早期的路由器之间使用串行链路互联（点到点链路），所以 RIP 触发更新只支持在点到点链路上开启 ,对于 Frame-Relay 和以太网这样的多路访问接口中 ,不支持 RIP 触发更新 ,但是 Frame-Relay 点到点子接口被 RIP 认为是点到点链路，可以开启触发更新。

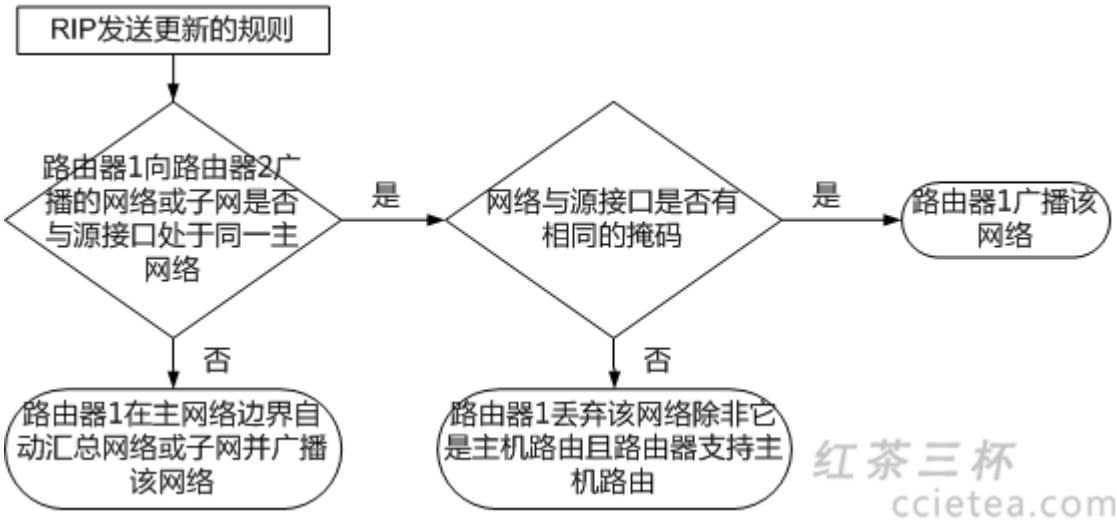
当开启触发更新后，链路两端的路由器之间不再周期性的发送路由更新，当然也带来另一个问题，路由表里的路由如果生存时间超时了也就挂了，因此要注意触发更新必须在链路两端接口上都进行配置，如此一来，路由更新会被标注 permanent 永久有效。

配置命令：接口模式下，在链路两端的接口上都要配置，完成后可 show ip rip database 查看路由详情

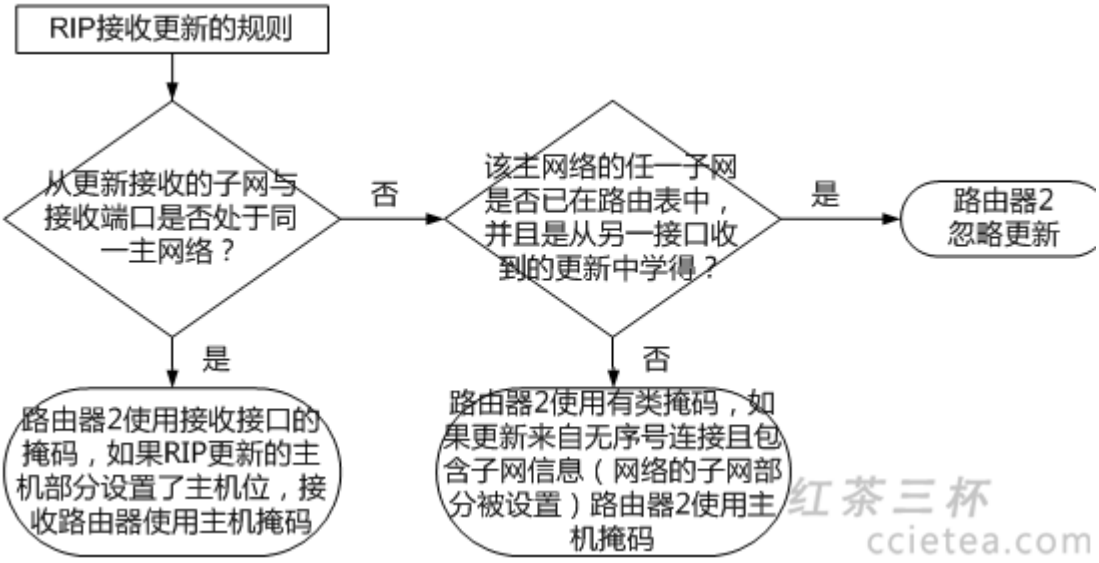
```
ip rip triggered
```

4.2.6 RIPV1 操作行为

1. RIP 更新行为

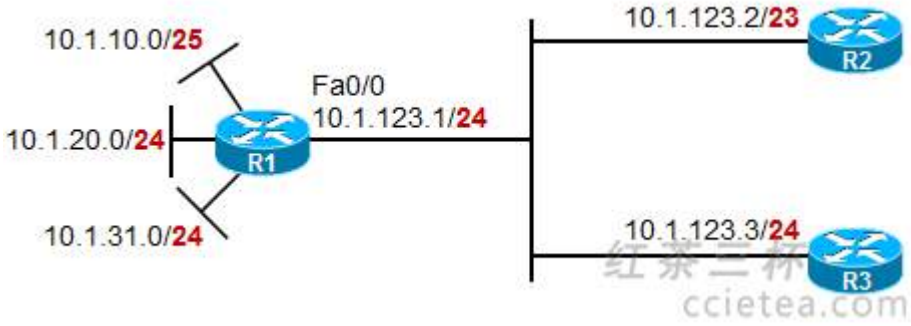


2. 接收行为



4.2.7 疑难解析

1. RIPv1 更新及接收规则验证



- **更新行为**

  10.1.10.0/25 这个子网，不能被更新，这是因为子网与更新源接口属于同一主类网络，但掩码不同。

  10.1.20.0/24 这条路由被 R1 以“10.1.20.0”更新出去，这是因为子网掩码与更新接口掩码/24 一致

  注意，v1 的更新包不含掩码，更新的只有子网号。

  10.1.31.0/24 这个子网，由于和 R1 的更新源接口同主类同掩码，因此被顺利更新“10.1.31.0”

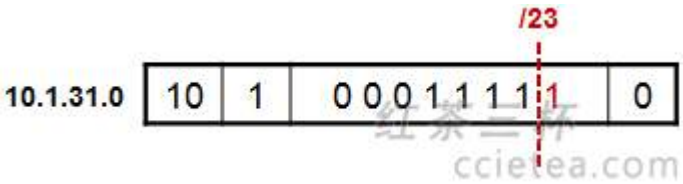
- **接收行为**

  “10.1.20.0”在传递给 R2 后，由于更新的子网与 R2 接口主网一致，因此 R2 接收更新，并且使用接口的掩码/23 作为接收的 RIP 子网的掩码，那么 R2 本地装载的这条 RIP 路由就是 10.1.20.0/23。这里其实路由的前缀长度就错误了，我们可以在 R2 接口上再配置个 secondary address，这样一来，RIP 将使用 secondary address（如果配置了的话）的掩码进行路由接收操作。例如给 R2 配置个 10.1.123.222/24，

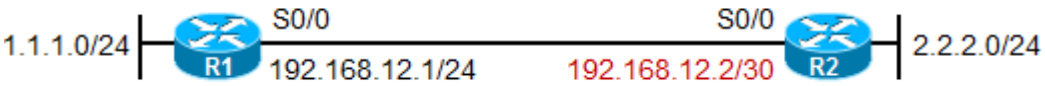
那么收到的路由就变成了 10.1.20.0/24，掩码就对了。

“10.1.20.0”在传递给 R3 后 ,由于 RIP 更新的子网与 R3 接口子网属同一主类网络 ,因此 R3 接收更新，并且用接口的掩码作为 RIP 路由的掩码，于是 R3 路由表中装载的路由就是：10.1.20.0/24

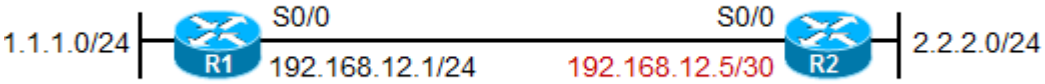
“10.1.31.0” 在传递给 R2 后 ,由于 R2 拿接收接口的掩码去运算 ,发现 10.1.31.0 在主机部分有位置 1，因此将该子网存进路由表，并将其视为主机路由：10.1.31.0/32



2. RIP 更新源问题



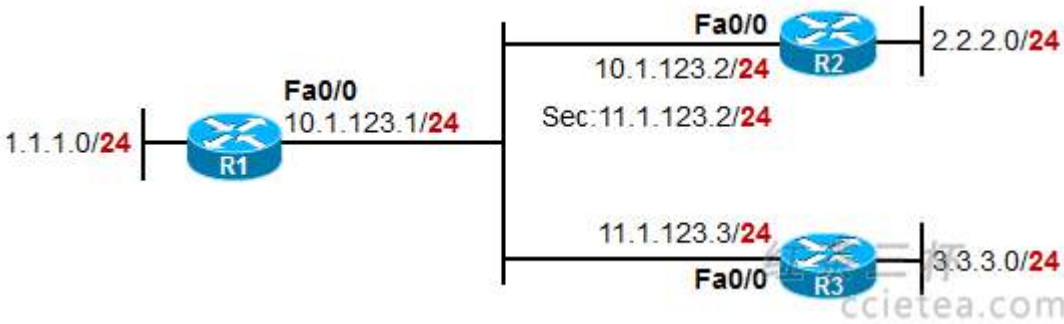
R1、R2 之间跑 RIP (宣告直连及各自的 LOOPBACK) , 这时 R1 发出来的更新包 , 源地址为 192.168.12.1，R2 收到包检查后发现，这是与自己接受更新包的接口同一子网的 IP，因此接受这个更新。所以这个测试，虽然两边接口掩码不同，但是互相之间都能接受对方的更新，同时 1.1.1.1 及 2.2.2.2 之间也都能 ping 通。



变更拓扑后，R2 的接口 IP 变了。这时候 R1 发过来的 RIP 更新，源地址仍是 192.168.12.1，R2 收到后，发现与本地接受更新的接口 IP 不在一个子网，因此 R2 忽略 R1 的更新消息。另外，R2 发给 R1 的消息，被 R1 接受了，因为 R1 认为 R2 发过来的 RIP 更新消息源地址 192.168.12.5 与自己的接口在同一个子网。

当 R2 上配置：no validate-update-source 后，R2 则不会对 RIP 更新消息源进行校验，因此 R2 会接受 R1 发来的更新，这时候双方各有对方的路由，但是 1.1.1.1 无法与 2.2.2.2 ping 通。

3. RIP secondary address 问题



IP 配置如上，R2 接口配置了两个 IP 地址，并都做了宣告全网跑 RIP。

- R2 会同时用主地址和辅助地址进行 validate-update-source ,因此 R2 能学习到 2.2.2.0 及 3.3.3.0 路由。
- 当同时存在主地址和辅助地址，RIPV1 路由器将使用本地“接收路由更新包”的接口的辅助地址的掩码进行接收操作（这点在上面的实验已经验证过了）。
- 当同时存在主地址和辅助地址，RIP 将同时使用主、辅助地址作为源来发送路由更新。所以 R1 和 R3 都会学习到 2.2.2.0 的路由更新。但是 R3 学习不到 1.1.1.0，一方面是因为 R1 发给 R3 的更新被 R3 忽略（更新源校验失败）,另一方面 R2 也不会将从 R1 收到的 1.1.1.0 的路由 ,再经过相同的接口、以 11.1.123.2 作为更新源，发给 R3。
- 另一点，R2 会将 10.0.0.0、11.0.0.0（辅助地址所在网段）、以及 1.0.0.0、3.3.3.0 都从自己的 loopback 更新出去 ;然而 ,它不会将自己的辅助地址 11.0.0.0 网段从 fast0/0 发送出去给 R1(除非关闭水平分割)。

4.3 RIPv2

4.3.1 改进

在 RIPv1 的基础上增加了

1. **外部路由标记**
   通过在 RIP 中使用路由标记，就能在其他协议中，控制相关路由的重发布。当重发布到其他协议时，RIP 路由只需比较赋予他们的标记而不用比较整个路由。
2. **VLSM 支持**
3. **组播能力**
   使用 224.0.0.9
4. **认证**
5. **下一跳**
   下一跳(Next Hop)—如果存在的话,它标识一个比通告路由器的地址更好的下一跳地址。换句话说,它指出的下一跳地址,其度量值比在同一个子网上的通告路由器更靠近目的地。如果这个字段设置为全 0(0.0.0.0),说明通告路由器的地址是最优的下一跳地址。

4.3.2 消息类型

字段名	长度	含义
Command	8 比特	标识报文的类型： 1：Request 报文，向邻居请求全部或部分路由信息； 2：Reponse 报文，发送自己全部或部分路由信息，一个 Response 报文中最多包含 25 个路由表项。
Version	8 比特	RIP 的版本号： 1：RIP-1； 2：RIP-2；
Must be zero	16 比特	必须为零字段。
AFI（Address Family Identifier）	16 比特	地址族标识，其值为 2 时表示 IP 协议。
Route Tag	16 比特	外部路由标记。
IP Address	32 比特	该路由的目的 IP 地址，可以是自然网段的地址，也可以是子网地址或主机地址。
Subnet Mask	32 比特	目的地址的掩码。
Next Hop	32 比特	提供一个更好的下一跳地址。如果为 0.0.0.0，则表示发布此路由的路由器地址就是最优下一跳地址。
Metric	32 比特	路由的开销值。

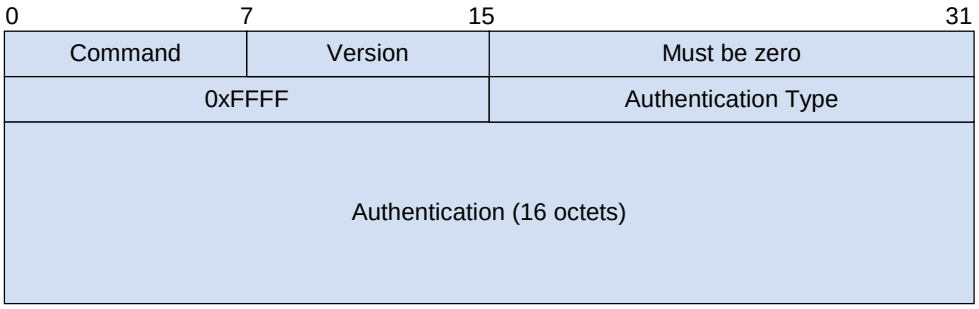
其中地址族消息默认总是 2，当请求整个路由表时为 0

关于下一跳及路由标记，详见卷一 189 页

4.3.3 认证

RIPv2 是通过更改 RIP 消息中正常情况下应该是第一个路由条目的字段来支持认证的,在含有认证的单个更新消息中,最大可以携带的路由条目被减少到了 24 个。

认证是通过设置地址族标识字段为全 1（0xFFFF）来标识的。



字段名	长度	含义
Command	8 比特	标识报文的类型： 1：Request 报文，向邻居请求全部或部分路由信息； 2：Reponse 报文，发送自己全部或部分路由信息，一个 Response 报文中最多包含 25 个路由表项。
Version	8 比特	RIP 的版本号： 1：RIP-1； 2：RIP-2；
Must be zero	16 比特	必须为零字段。
0xFFFF	16 比特	验证项标识，表示整个路由报文需要验证。
Authentication Type	16 比特	验证类型： 2：明文验证； 3：MD5 验证。
Authentication	16 字节	验证字，当使用明文验证时包含了密码信息。

4.3.4 兼容性

默认情况下 CISCO 路由器上运行的 RIP 发送 v1，接收 v1 v2， 如果显示声明为 v1（在 RIP 进程中使用 version 1 命令）则只发送和接受 v1 报文，v2 默认只发送和接收 v2 报文

RFC1723 定义了几个兼容性开关

- RIP-1　　只发送 RIPv1 更新
- RIP-1 兼容性　　RIPv2 使用广播的形式发送更新，使得 V1 能收到
- RIP-2　　RIPv2 使用组播更新
- None　　不发送更新 （passive）

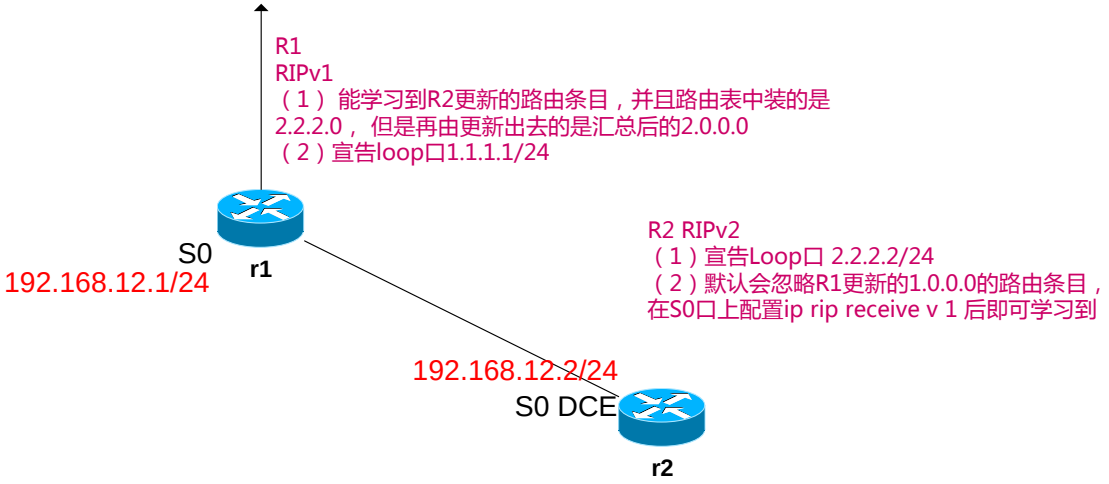
RFC1723 定义了一个接受控制开关

- RIP-1 Only

- RIP-2 Only
- Both
- None 使用分发列表或访问控制列表 deny 521 端口。

v1 默认发送 v1 接收 v1,v2

v2 默认发送 v2 接收 v2



4.3.5 高级特性

1. **被动接口**

   Passive-interface Fa0/0

   接口将只收更新，不发更新

2. **单播更新**

   Passive-interface Fa0/0

   neighbour xxxxxx

3. **偏移列表**

   手动修改 RIP 的跳数。首先配置匹配的路由

   Offset-list Acl 号 in/out 偏移值 接口

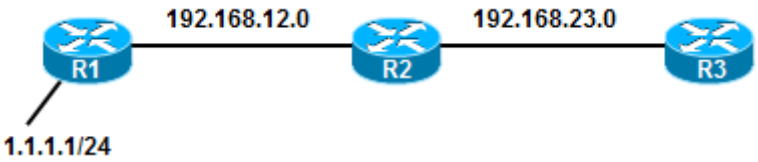
   //该偏移值是增加在正常路由条目中显示的值上，是在原来的基础上增加而不是替换原来的跳数。

4. **RIPv2 手工汇总**

   Ip summary-address rip xxxx yyyy //注意，这个汇总不支持 CIDR 超网。

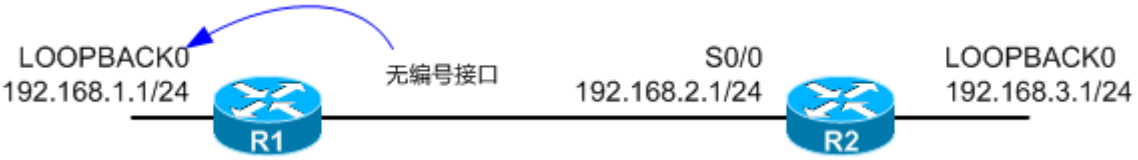
4.3.6 疑难解析

1. RIP 基于传闻的更新



运行 RIPv2，默认 R1 更新 1.0.0.0/8 的汇总给 R2，在 R1 上关闭自动汇总，则 R2 学习到 1.1.1.0/24，同时由于自己仍然开启自动汇总，于是更新给 R3 的是 1.0.0.0/8。这时候如果在 R2 上 ip route 1.1.1.0 255.255.255.0 null0，则 R2 上路由表关于 1.1.1.0 就变成了 static 静态路由了，则 R2 不再向 R3 更新 RIP 的 1.0.0.0/8 的路由了

2. 使用无编号地址传递 RIP 更新(无效源)



R2 会忽略 R1 发送过来的更新，因为“有源检查”，从 R1 发送过来的 RIP 更新与 R2 的 S0/0 口不在同一子网，所以 R2 忽略 R1 发来的路由更新。这个时候只要在 R2 的 RIP 进程下 no validate-update-source 即可。

4.4 RIPng

请看红茶三杯《IPv6 笔记》，访问 ccietea.com 获取最新版本

4.5 参考资料

文档编号	描述
RFC1058	Routing Information Protocol
RFC1723	RIP Version 2 Carrying Additional Information
RFC1721	RIP Version 2 Protocol Analysis

文档编号	描述
RFC1722	RIP Version 2 Protocol Applicability Statement
RFC1724	RIP Version 2 MIB Extension
RFC2082	RIP-2 MD5 Authentication
RFC2080	RIPng for IPv6

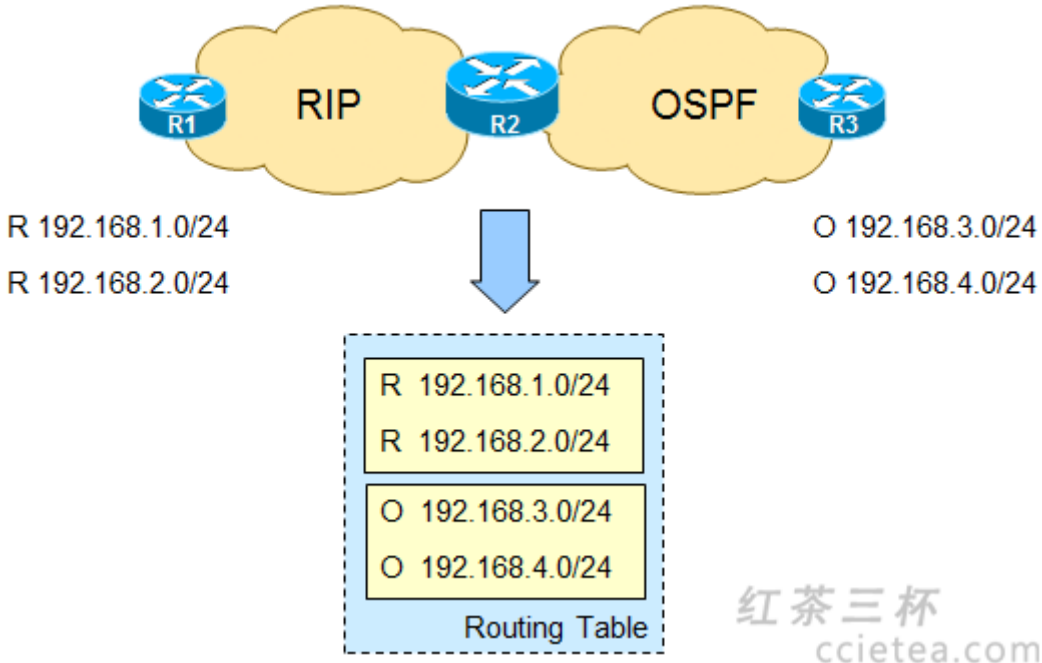
5 路由重发布（Redistributing Routing Protocols）

5.1 技术背景

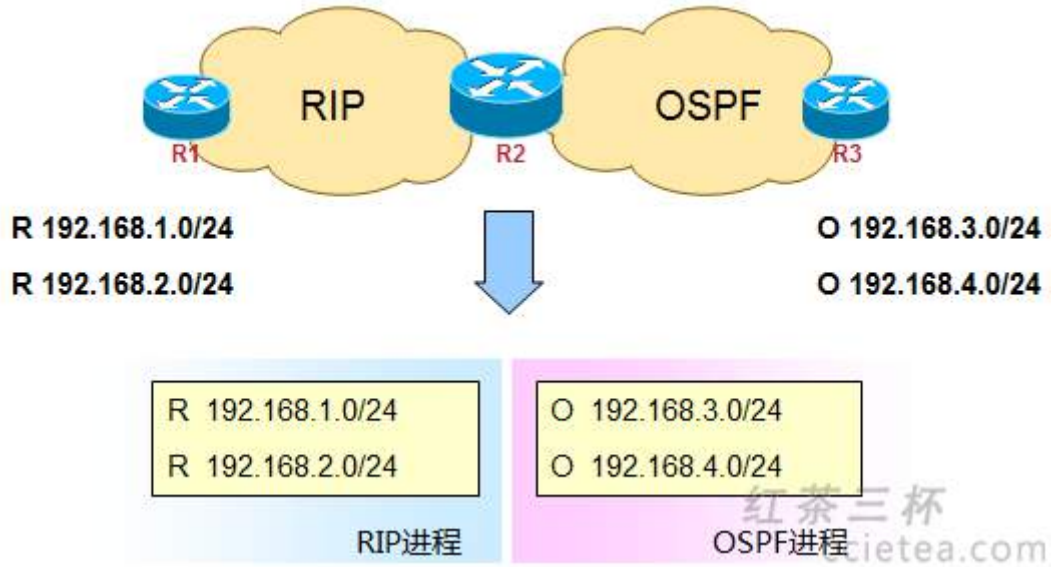
在现实的网络环境中，我们可能会遇到，一个网络环境中，同时存在两种或两种以上的路由协议的情况，例如：

- 多厂商的路由环境
- 网络合并（同一协议或是不同协议）
- 从旧的路由协议过渡到新的路由协议
- 路由策略的需要（可靠性、冗余性、分流模型等）

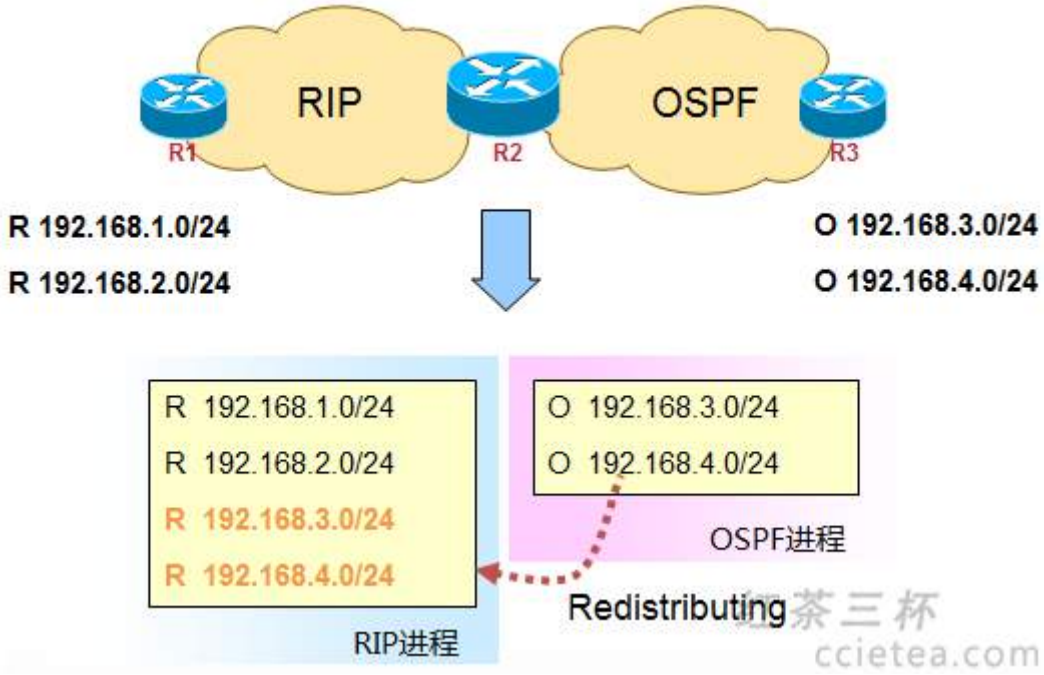
在同一个网络拓扑结构中，如果存在两种不同的路由协议，由于不同的路由协议的机理各有不同，对路由的理解也不相同，这就在网络中造成了路由信息的隔离，然而由于这很有可能是同一个自治系统内的网络，全网需要互通，这时候咋办？这就需要使用路由重发布了。



我们看上图，R1 与 R2 之间，运行 RIP 来交互路由信息，R2 通过 RIP 学习到了 R1 发布过来的 192.168.1.0/24 及 2.0/24 的路由信息，装载进路由表，标记为 R。同时 R2 与 R3 又运行 OSPF，建立起 OSPF 邻接关系，R2 也从 R3、通过 OSPF 学习到了两条路由：3.0 及 4.0/24，也装载进了路由表，标记为 O。那么这样一来，对于 R2 而言，它自己就有了去往全网的路由，但是，在 R2 内部，我们可以这么形象的理解：它不会将从 RIP 学习过来的路由，变成 OSPF 路由告诉给 R3，也不会将从 OSPF 学习来的路由，变成 RIP 路由告诉给 R1。对于 R2 而言，虽然它自己的路由表里有完整的路由信息，但是，就好像冥冥之中，R 和 O 的条目之间有道鸿沟，无法逾越。而 R2 就也就成了 RIP 及 OSPF 域的分界点。那么如何能够让 R1 学习到来自 OSPF 的路由，让 R3 学习到来自 RIP 的路由呢？关键点在于 R2 上，通过在 R2 上部署**路由重发布**，可以实现路由信息在不同路由选择域间的传递。



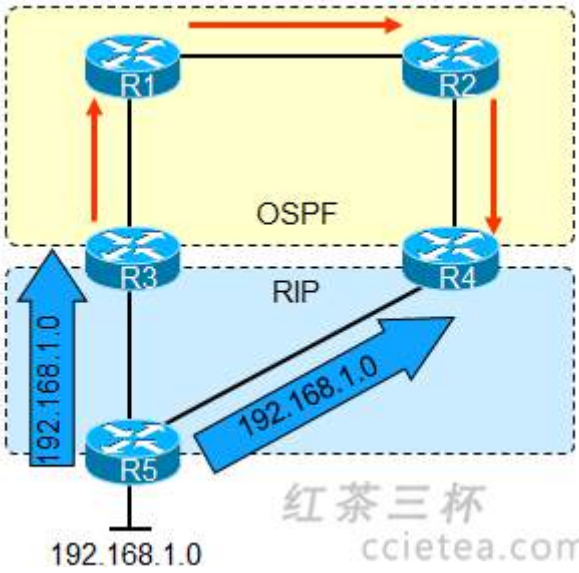
上图是初始状态。



上图中，我们开始在 R2 上执行重发布的动作，我们将 OSPF 的路由“注入”到了 RIP 进程之中，如此一来，R2 就会将 3.0/24 及 4.0/24 这两条 OSPF 路由“翻译”成 RIP，然后传递给 R1。R1 也就能够学习到 3.0 和 4.0 路由了。注意重发布的执行地点，是在 R2 上，也就是在路由选择域的分界点上执行，另外，路由重发布是有方向的，例如刚才我们执行完相关动作后，R3 还是没有 R1 的路由的，需进一步在 R2 上，将 RIP 路由重发布进 OSPF，才能让 R3 学习到 1.0/24 及 2.0/24 路由。

5.2 实施要点

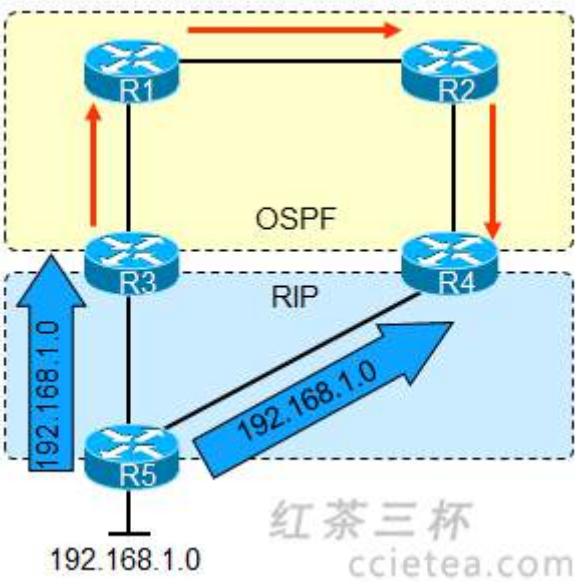
5.2.1 路由 feedback



路由的 Feedback（回馈）是一个在部署路由重发布时需要非常注意的一个现象。如上图所示，R5 将 192.168.1.0 宣告进了 RIP，R3 及 R4 都能够学习到这条路由，并且装载进自己的全局路由表。那么如果我们在 R3 上部署 RIP 到 OSPF 的双向重发布，会发生什么事情呢？我们假设在 R3 上先完成的配置，192.168.1.0 这条路由将被 R3 注入到 OSPF 中，并被更新给 R1，再由 R2 更新给 R4，此刻，R4 同时从 OSPF 及 RIP 都学习到了这条路由，它会作何优选？当然是优选 OSPF 的，因为 AD 小，所以它的路由表里，关于 192.168.1.0 的路由是 OSPF 的。这样一来，对于 R4 而言，它去往 192.168.1.0，就存在**次优路径**，也就是说，绕远路了，走 R2-R3-R3-R5 这条路径。并且由于路由表里没了 RIP 路由，自然 RIP 向 OSPF 重发布就失败了，更糟糕的是，R4 上关于 192.168.1.0 的 OSPF 路由更会被重新注入 RIP（因为我们部署的是双向重发布），这就是路由 Feedback，路由被灌回来了。

因此在部署重发布时，这个问题是需要格外注意的，至于问题如何规避，在后面的内容中，我再做介绍。

5.2.2 管理距离问题

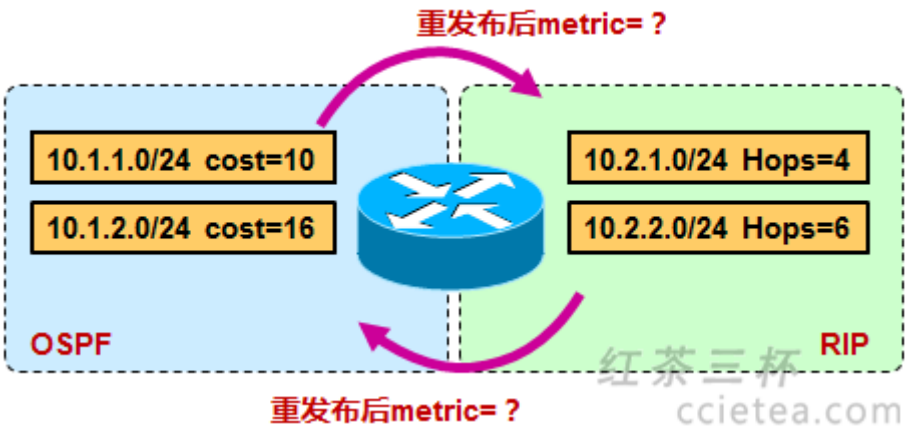


在上面这个例子中，我们提到的现象，R4 会同时从 OSPF 及 RIP 都获知到 192.168.1.0/24 的路由，最终 R4 会选择 OSPF 的路由。这是我们在这个环境中，不愿看到的现象，因为这样一来就造成了次优路径。几种常见的路由协议其 AD 值列举如下：

路由来源	管理距离
直连接口	0
静态路由	1
EIGRP汇总路由	5
外部BGP	20
内部EIGRP	90
OSPF	110
IS-IS	115
RIPv1、RIPv2	120
外部EIGRP	170
内部BGP	200
未知	255

值得注意的是我们可以通过在 R4 上，特定的协议进程中手工修改该路由协议的管理距离，从而达到影响路由器本身路由选择的目的，例如在 R4 上，我们将 OSPF 针对外部路由的默认管理距离 110，修改为 130，比 RIP 的管理距离 120 更大，这样一来，R4 在这个环境中，针对 192.168.1.0 就会优选 RIP 的路由，就可以规避次优路径以及路由 feedback 的问题了。

5.2.3 Metric 问题



要知道，每一种路由选择协议，对于路由 Metric 度量值的理解是不同的，OSPF 是用 cost 开销来衡量一条路由，RIP 是用跳数，EIGRP 是用混合的各种元素，那么当我将一些路由，从某一中路由协议重发布到另一种路由协议中，这些路由的 metric 会作何变化呢？方式之一是，你可以在执行重发布的动作的时候，手工进行修改，具体改成什么值，要看具体的环境需求，这个我们后面会举例别担心，哥们就这么实诚 :)。 方式之二是，采用默认的动作，也就是在路由协议之间重发布时所定义的种子度量。

所谓种子度量，指的就是当，我将一条路由，从外部路由选择协议重发布到本路由选择协议中时，如果没有手工指定路由的 metric，而使用的默认的 metric。看下表（下表是一个公认的默认值，可在路由进程中使用 default-metric 修改）:

将路由重分发到该协议	默认种子度量值
RIP	0，视为无穷大
IGRP/EIGRP	0，视为无穷大
OSPF	BGP 进来为 1，其他路由为 20，OSPF 之间度量值保持不变
IS-IS	0
BGP	BGP 度量值被设置为 IGP 度量值

注意，以上是从其他动态路由协议重发布进本路由路由协议时的默认 metric。

而如果是重发布本地直连或静态路由，则情况就有变化了，如下：

- EIGRP 请见红茶三杯 EIGRP 笔记（访问 ccietea.com）
- RIP 重发布直连如果没有设置 metric，则默认 1 跳传给邻居（邻居直接使用这个 1 跳作为 metric）；重发布静态路由默认 metric=1 ,使用 default-metric 可以修改这个默认值 ,这条命令对重发布直连接口的 metric 无影响。
- OSPF 重发布直连接口默认 cost20；重发布静态路由默认 cost20；使用 default-metric 可以修改重发布静态路由以及其他路由协议的路由进 OSPF 后的默认 cost，只不过这条命令对重发布直连接口无效。

5.2.4 有类、无类路由选择协议的重发布

这个话题，个人觉得没必要研究了，对有类路由协议的研究个人感觉意义不大。无论是实际应用或是 LAB 考试，都不再涉及，有兴趣自己弄弄吧。

5.3 配置示例

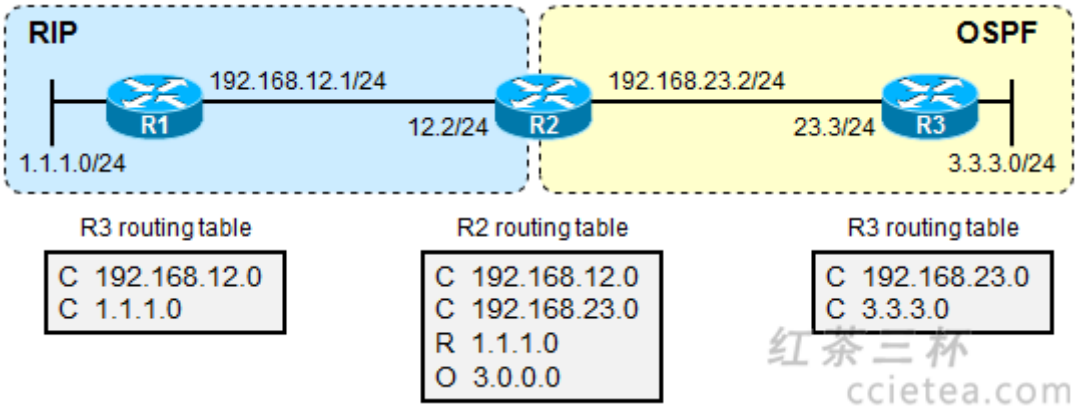
5.3.1 配置命令

路由重发布是有方向的，将 A 路由选择域的路由信息注入到 B 路由选择域中，我们要在 B 路由协议的进程中进行配置，例如，要将其他路由协议重发布到 RIP，那么配置如下（重发布到其他路由协议大同小异）：

```
Router(config)#router rip
Router(config-router)#redistribute ?
  bgp          Border Gateway Protocol (BGP)
  connected    Connected
  eigrp        Enhanced Interior Gateway Routing Protocol (EIGRP)
  isis         ISO IS-IS
  iso-igrp     IGRP for OSI networks
  metric       Metric for redistributed routes
  mobile       Mobile routes
  odr          On Demand stub Routes
  ospf         Open Shortest Path First (OSPF)
  rip          Routing Information Protocol (RIP)
  route-map    Route map reference
  static       Static routes
```

5.3.2 配置示例

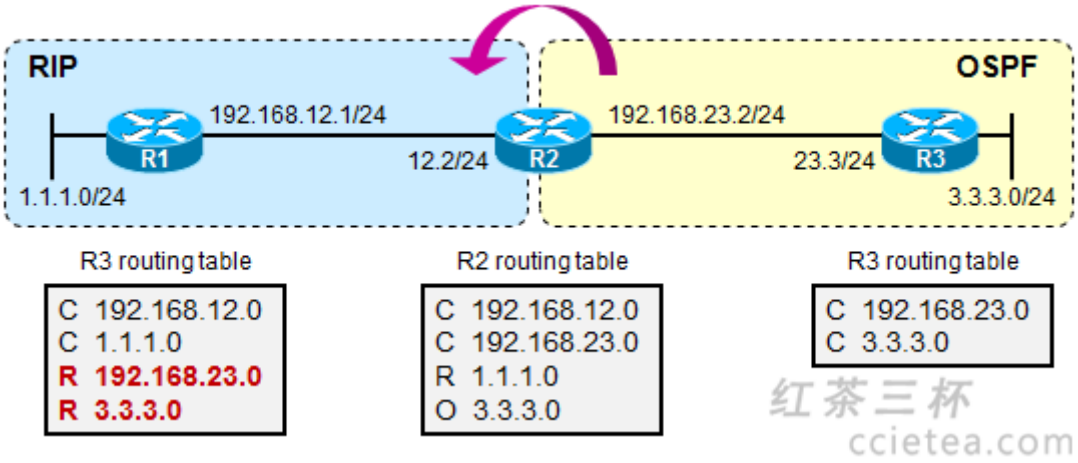
1. OSPF 与 RIP



R1 与 R2 运行 RIP；R2 与 R3 建立 OSPF 邻居关系。初始化情况下，R2 的路由表中有四个条目，如上图所示，而 R1 的路由表中，只有 2 个条目，也就是两个直连链路。那么现在，我们在 R2 上做重发布动作，将 OSPF 路由重发布到 RIP，那么配置如下：

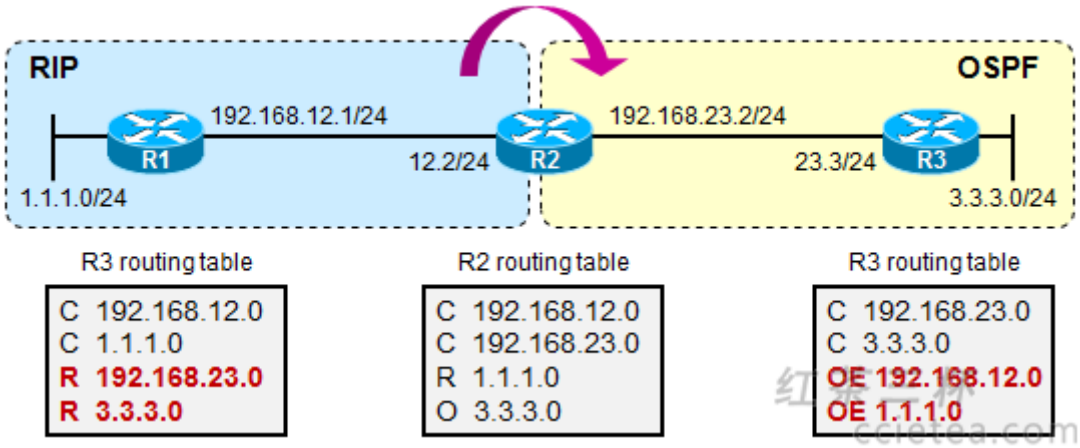
```
router rip
  redistribute ospf 1 metric 3
```

如此一来，R2 上，路由表中的 OSPF 路由 1.1.1.0，以及宣告进 OSPF 进程的 192.168.12.0 直连网段，都被宣告进了 RIP，而 R1 通过 RIP 就能够学习到这两条路由，如下图示红色粗体部分。



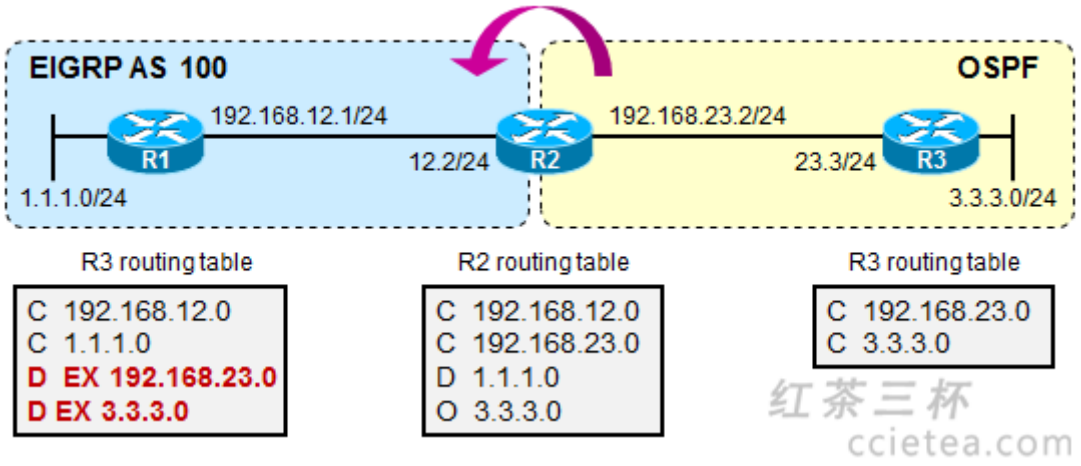
当然，这个时候 1.1.1.0 是无法访问 3.3.3.0 的，因为 R3 并没有 RIP 路由选择域中的路由（也就是说回程路由有问题，数据通信永远要考虑来回路径，记住了），所以如果要实现全网互通，那么需在 R2 上：

```
R2(config)#router ospf 1
R2(config-router)#redistribute rip subnets
```



如此一来，就实现了全网互通。注意，当重发布路由到 OSPF 时，redistribute rip subnets，这个 **subnets** 关键字要加上，否则只会重发布主类路由。

2. OSPF 与 EIGRP 重发布



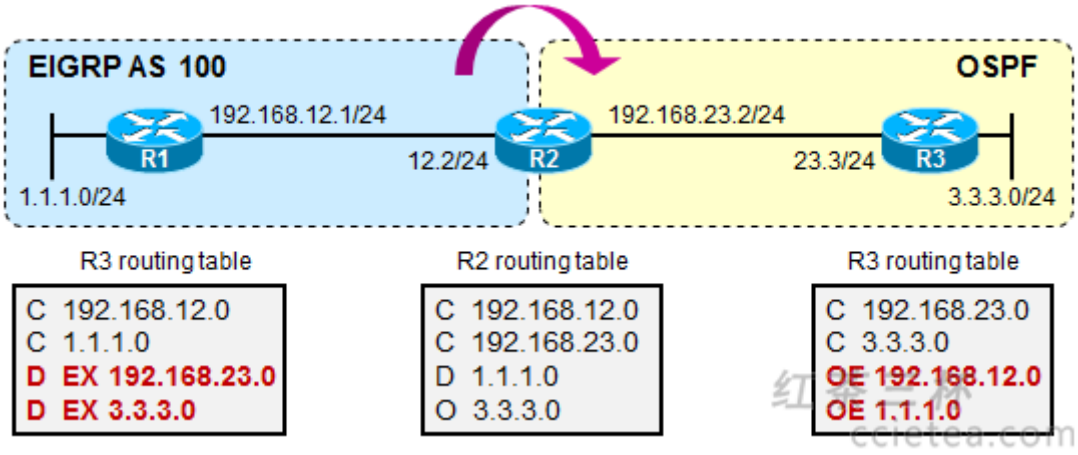
初始情况同上，我们先看看将 OSPF 路由重发布进 EIGRP AS 100，配置当然还是在 R2 上进行，进入 R2 的 EIGRP 路由进程：

```
R2(config)# router eigrp 100
R2(config-router)# redistribute os 1 metric 100000 100 255 1 1500
```

注意，EIGRP 的 metric 是混合型的，**metric 100000 100 255 1 1500** 这里指定的参数，从左至右依次是带宽、延迟、负载、可靠性、MTU。可根据实际需要灵活的进行设定。上述配置完成后，R2 就会将路由表中 OSPF 的路由：包括 3.3.3.0，以及宣告进 OSPF 的直连网段 192.168.23.0/24 注入到 EIGRP 进程。这样 R1 就能够学习到这两条外部路由。

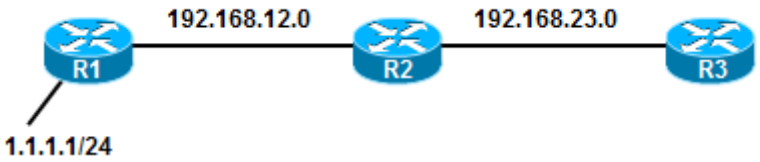
接下去是 EIGRP 到 OSPF 的重发布：

```
R2(config)# router ospf 1
R2(config-router)# redistribute eigrp 100 subnets
```



5.3.3 重发布的常见问题

1. **关联出接口的静态路由 在被 network 时的问题**



R1、R2、R3 **跑 RIP**，R1 上 1.1.1.0/24 没有直接宣告，在 R2 上：

```
Ip route 1.1.1.0 255.255.255.0    serial 0/0
```

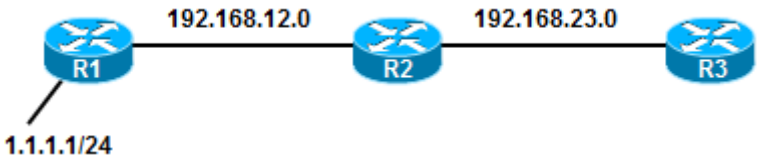
同时 network 1.0.0.0，这时 1.0.0.0 会被宣告出去

使用关联出接口的方式配置的静态路由，路由器会将目的网段视为**本地直连**，因此 RIP 在 network 的时候，会宣告出去。

如果以上换成 EIGRP，则现象与 RIP 一样，R1 会将 1.1.1.0 宣告进 EIGRP

如果以上换成 OSPF，则无效，即关联出接口的静态路由，在 OSPF 中 network 该路由的网络号时，并不会被宣告进 OSPF。

2. **关联出接口的静态路由 在重发布时的问题**

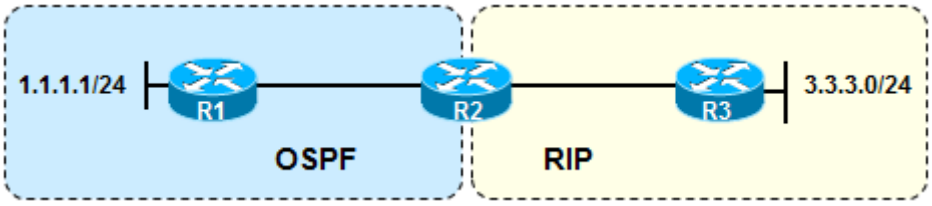


R1、R2、R3 跑 RIP，R1 上 1.1.1.0/24 没有直接宣告，在 R2 上：

Ip route 1.1.1.0 255.255.255.0 serial 0/0

此时在 R2 上重发布直连接口，则发现 1.1.1.0 并没有被重发布进 RIP

3. 重发布只会将路由表中有的路由执行重发布动作



在 R2 上进行双向重发布，正常情况下 R1 能够学习到 3.3.3.0、R3 能学到 1.1.1.0

如果在 R2 上 ip route 1.1.1.0 255.255.255.0 null0 ,这个时候路由表中没有了 OSPF 的 1.1.1.0 路由了因此重发布不成功，所以 R3 无法学习到 1.1.1.0/24 的路由；

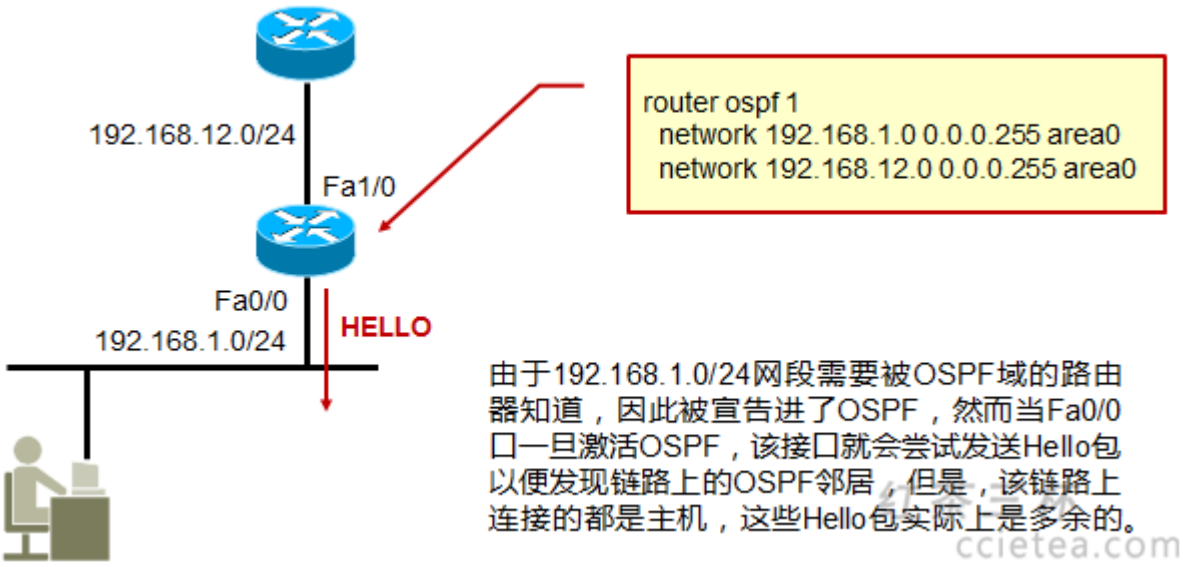
同样，如果 ip route 3.3.3.0 255.255.255.0 null0 也是一样的道理。

重发布是看路由表的，也就是说，例如我将 OSPF 重发布到 EIGRP，那么首先路由必须得在我路由表里有，而且必须是 OSPF 的路由，才能够被注入到 EIGRP。另外，这里有个小问题，R1、R2 之间的链路，虽然在 R2 的路由表中没有看到关于它的 OSPF 路由，但是却成功地被重发布进 RIP 且被 R3 学习到了，这是因为这个**直连链路(接口)已经被 R2 的 OSPF 进程 network 了** ,也即通过 OSPF 学习到了 ,且是直连链路，因此能被重发布。

6 路由策略

6.1 Passive-interface

6.1.1 特性概述



针对上面的问题，我们可以将路由器上的 Fa0/0 口设置为 passive-interface，如此一来，该接口将不再发送 OSPF Hello 消息，同时，该接口关联的网段 192.168.1.0/24 仍然会被宣告进 OSPF，这样就能够避免不必要的 OSPF 组播包在 LAN 中泛洪。这就是 passive-interface 特性的一个最常见的应用，需要注意的是，不同的路由协议对 passive-interface 的操作有所不同。

6.1.2 相关要点

- RIP 和 IGRP 的 passive-interface 不发送路由更新，但是接受路由更新
- EIGRP 的 passive-interface 既不发送也不接收路由更新，并且也不发 HELLO 包
- OSPFpassive-interface 既不发送也不接收路由更新，并且也不发 HELLO 包

【注意】接口如果被 passive 掉，但是同时在路由进程里 network 这个接口，则该接口虽不会尝试去发送更新或建立邻居关系，但是其所在网段，仍会被宣告进路由选择进程。

6.1.3 Passive-interface 的配置

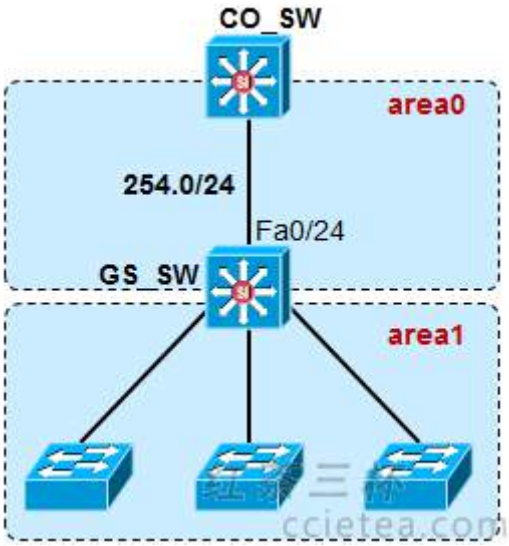
- **将某个接口配置为被动接口：**

  Router(config-router)# **passive-interface** int-*type int-num*

- **将所有接口配置为被动接口，并手动激活特定接口：**

```
Router(config-router)# passive-interface default
Router(config-router)# no passive-interface int-type int-num
```

- **典型配置示例：**



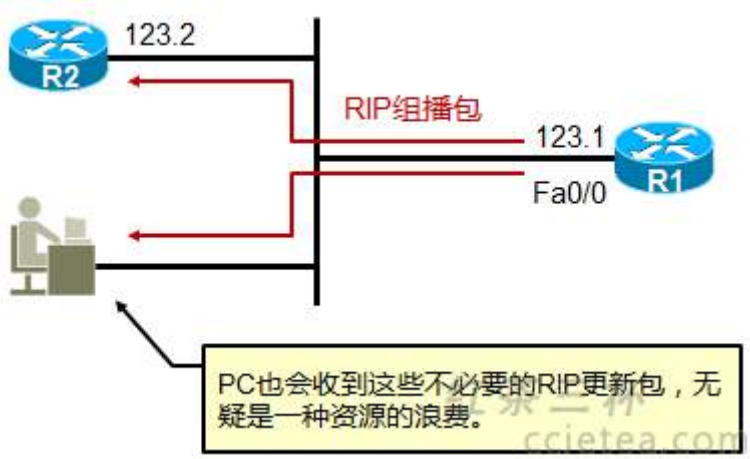
这是一个非常典型的场景，GS_SW 是汇聚层交换机，与核心交换机之间跑个 OSPF。GS_SW 需要通告下挂的 VLAN 所对应的网段，以便核心交换机能够获知相关的路由。然后一旦 GS_SW 在 OSPF 进程中宣告这些 VLAN 对应的网段，相应的 SVI 接口就会向 VLAN 中去泛洪 OSPF HELLO 消息，而这些消息，实际上是多余的。因此我们可以做如下配置：

```
interface vlan 10
  ip address 192.168.10.254 255.255.255.0
interface vlan 20
  ip address 192.168.20.254 255.255.255.0
router ospf 1
  network 192.168.10.0 0.0.0.255 area 1
  network 192.168.20.0 0.0.0.255 area 1
  network 192.168.254.0 0.0.0.255 area 1
  passive-interface defualt
  no passive-interface fast 0/24
```

由于实际部署的时候，SVI 口可能比较多，如果一个个的去 no passive-interface 配置量可能会比较大，因此可以选择先 passive-interface defaut 将所有接口全部 passive 掉，然后再单个接口去 no passive-interface。

6.2 单播更新

1. **配置 RIP 单播更新：**



上图中，R1 及 R2 两台路由器，互相之间通过 RIP 交互路由信息。但是，由于 RIPv2 基于组播发送路由更新及相关报文，一旦两台路由器在接口上激活 RIP，PC 将不得不耗费资源处理这些它并不需要的组播消息。因此这里就可以使用到单播更新这个特性，**R1 配置如下：**

```
Router(config) router rip
Router(config-router)# passive-interface fast 0/0
Router(config-router)# neighbor 192.168.123.2
```

R2 的配置类似。配置完成后，R1、R2 之间交互 RIP 消息，就采用单播进行，这样 PC 就不会收到影响。

2. **配置 EIGRP 单播更新：**

注意：如果是 EIGRP 环境，需实现单播更新，那么路由更新接口不能被 PASSIVE（这与 RIP 不一样），直接使用 neighbor 命令去指定邻居即可。如果接口一旦被 PASSIVE，则即使手工指定了 neighbor，也是无法正常建立 EIGRP 邻居关系的。

3. **配置 OSPF 单播更新：**

OSPF 的 neighbor 命令使用上又与 EIGRP 不太一样，经测试，在以太网环境下，直接互指 neighbor，仍然会发组播 hello

6.3 调整路由协议的管理距离

1. **配置如下：**
   - **修改 OSPF 的 AD 值**

     Router(config)# router ospf 1

     Router(config-router)# **distance** *AD ip-src wildmask acls*

     或者

     Router(config-router)# **distance ospf external** *ad1* **inter-area** *ad2* **intra-area** *ad3*

     上述两条命令，都能起到调整路由协议管理距离的作用，第一条命令 distance ad ip-src ，可以针对特定的路由更新源及特定的路由前缀调整管理距离，例如，我将某个 OSPF 邻居发给我的某些路由，AD 值调整为 130。

     第二条命令，是针对外部路由、区域间或者区域内的路由进行 AD 值的调整。
   - **修改 EIGRP 的 AD 值**

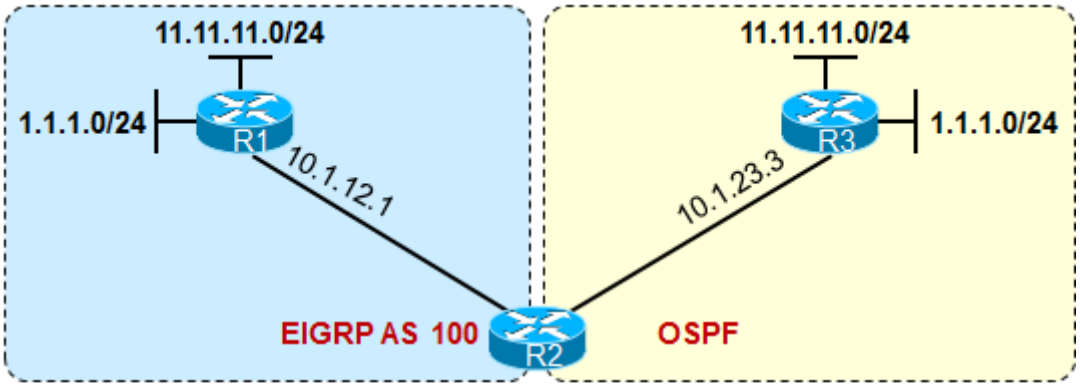
     Router(config)# router eigrp 100

     Router(config-router)# **distance** *AD ip-src wildmask acls*

     或者

     Router(config-router)# **distance eigrp** *internal-distance external-distance*

2. **配置示例**



实现需求（R2的路由表）：

```
O 11.11.11.0
D 1.1.1.0
```

实现方式（R2的配置）：

```
access-list 1 per 11.11.11.0
!
router eigrp 100
distance 130  10.1.12.1 0.0.0.0 1
```


上图中，R1、R3 都会更新路由 1.1.1.0 及 11.11.11.0 给 R2，当然，默认情况下，R2 肯定是优选 EIGRP

的路由。因此去往这两个网段，都走 R1。那么如果我们希望，去往 11.11.11.0 走 R3，从而使得这两个目的地能够起到分流的效果呢？我们就可以选择在 R2 的 EIGRP 进程中，将来源于 10.1.12.1 这个更新源的路由 11.11.11.0/24 的 AD 值调整为比 OSPF 大，例如 130。因此配置如下：

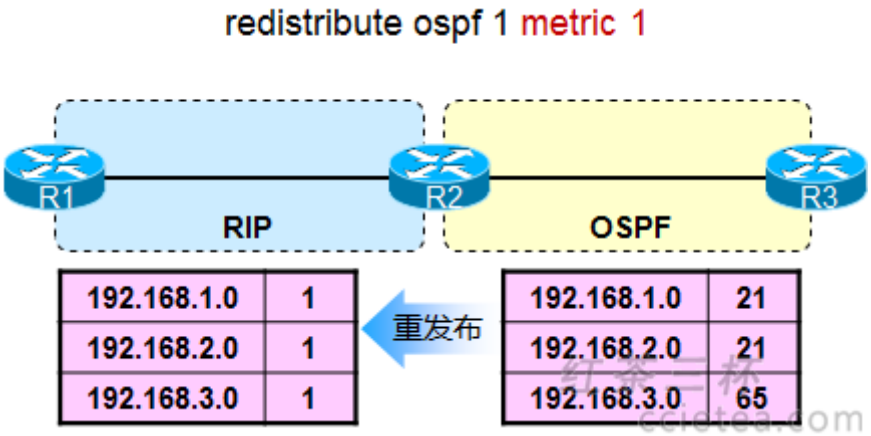
```
access-list 1 permit 11.11.11.0
router eigrp 100
  distance 130 10.1.12.1 0.0.0.0 1
```

更详细的关于各种路由协议调整 AD 值的内容，请见红茶三杯的 OSPF、EIGRP、BGP 等技术文档。

6.4 Route-map

6.4.1 Route-map 概述

1. **技术背景**



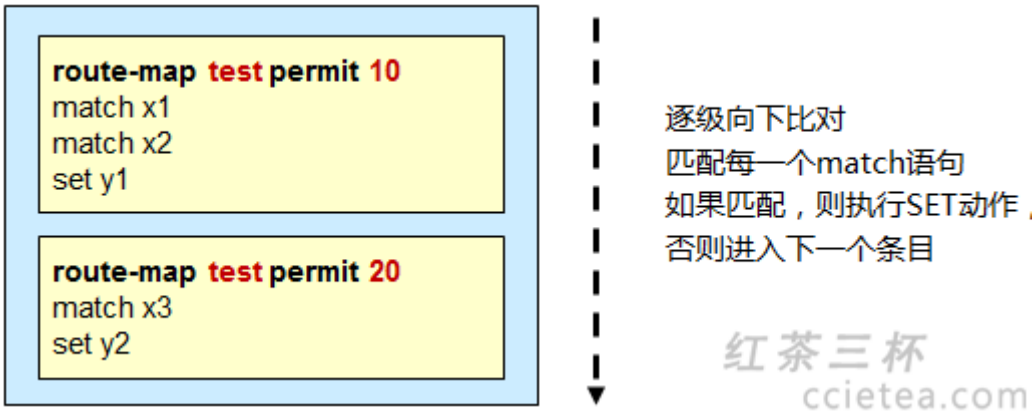
首先来初步认识一下 route-map。看上图，我们在 R2 上，将 OSPF 路由重发布进 RIP，前面已经说过了，在重发布时，可以使用 metric 关键字来设置路由被重发布进 RIP 后的 metric，这里设置为 1，那么直接的结果是，所有被注入到 RIP 的 OSPF 路由，metric 都是 1。那么如果我希望做些灵活性的调整呢？例如我希望在路由被注入 RIP 后，192.168.1.0 路由的 metric 为 1，2.0 的 metric 为 2 如此这般呢？传统的重发布是没办法做到的。

那么就可以使用 route-map 这个工具，也就是说，我们可以在执行重发布的时候，关联一个 route-map，来实现我刚才所说的这个功能。

2. Route-map 的使用场景

- **重分发期间进行路由过滤或执行策略**
- **PBR（策略路由）**
- **NAT（网络地址转换）**
- **BGP 中的策略部署**
- **其他用途**

3. Route-map 初相识



首先明确一下，route-map 是一个非常重要的工具，使用的范围非常广泛。在定义 route-map 的时候，我们采用 route-map 关键字，关联一个自定义的参数，例如 test 来创建。一个 route-map 列表，由这个 test 字符串统一表示，你可以在一个 route-map 下定义多个序列，用十进制的序列号来表示，例如上图中的，10、20。

那么在每一个序列中，我们就可以来定义供策略部署的两个元素：**匹配条件、执行动作**。你可以定义多个条件，当条件被匹配时，就会去执行 set 指定的相关动作。这里稍微提一下，set 语句并不是必须，例如如果该 route-map 仅仅为了匹配感兴趣流量，那么可能就只有 match 语句而没有 set 语句。

在 route-map 被调用后，匹配动作将会从最小的序列号开始执行，如果该序列号中的条件都被匹配了则执行 set 命令，如果条件不匹配，则切换到下一个序列号继续进行匹配动作。

4. Route-map 的特点

- 使用 match 命令匹配特定的分组或路由，set 修改该分组或路由相关属性。
- Route-map 中的每个序列号语句相当于于访问控制列表中的各行。按照序列号的顺序自上而下的处理，一旦找到匹配的序列则不会继续往下继续查找。
- Route-map 默认为 permit，默认序列号为 10，序列号不会自动递增，需要指定序列号
- 末尾隐含 deny any

- 单条 match 语句包括多个条件时，使用逻辑 or 运算；多条 match 语句时，使用逻辑 and 运算。

6.4.2 配置命令

1. **创建 route-map**

   route-map

   - 这个全局配置命令创建一个 route-map，使用自定义的字符串来表示这个 route-map，你可以在一个 route-map 下定义多个序列号。序列号在进行匹配动作时具有优先顺序。
   - Permit/deny 关键字在不同的部署场合中作用有所不同

```
route-map test permit/deny 10
  match x1
  match x2，x3
  set Y
route-map test permit/deny 20
  match x4
  set Y
```

2. **定义匹配条件**

```
match ip address 匹配访问列表或前缀列表
match length 根据分组的第三层长度进行匹配
match interface 匹配下一跳出接口为指定接口之一的路由
match ip next-hop 匹配下一跳地址为特定访问列表中被允许的那些路由
match metric 匹配具有指定度量值的路由
match route-type 匹配指定类型的路由
match community 匹配 BGP 共同体
match tag 根据路由的标记进行匹配
```

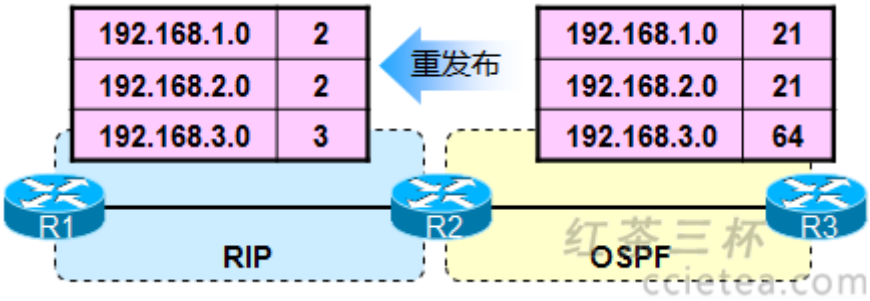
3. **定义 set 动作**

```
set metric 设置路由协议的度量值
set metric-type 设置目标路由协议的度量值类型
set default interface 指定如何发送这样的分组
```

```
set interface  指定如何发送这样的分组
set ip default next-hop 指定转发的下一跳
set ip next-hop  指定转发的下一跳
set next-hop  指定下一跳的地址，指定 BGP 的下一跳
set as-path
set community
set local-preference
set weight
set origin
set tag
default  关键字优先级低于明细路由
```

6.4.3 配置示例

1. **路由重发布时关联 route-map**



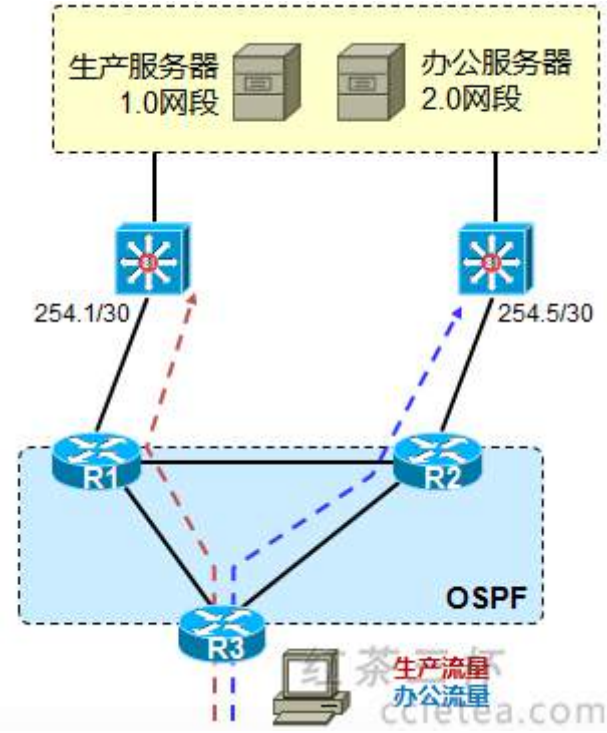
在上图中，我们将 OSPF 路由注入到 RIP，传统的做法，你只能够对所有注入进来的路由统一设置 metric，但是有了 route-map，我们可以在配置重发布命令时，关联一个已经定义好的 route-map，在 route-map 中，我们可以通过创建多个序列号语句，进而对不同的路由，设置不同的属性或动作。

例如这个例子，我们希望注入进来后，192.168.1.0 和 192.168.2.0 这两条路由的 metric 变为 2 跳，3.0 变为 3 跳。

```
access-list 1 permit 192.168.1.0
access-list 1 permit 192.168.2.0
access-list 2 permit 192.168.3.0
!                                   !! 上面创建了两个 ACL，分别匹配需要差分对待的路由
route-map test permit 10
  match ip address 1                !! 当路由匹配 ACL1 时
```

```
  set metric 2                          !! 将 metric 修改为 2
route-map test permit 20
match ip address 2
  set metric 3
router rip
  redistribute ospf 1 route-map test
```

2. 路由重发布时关联 route-map （典型案例）



这是一个非常典型的案例，上图中，网络环境是这样的，假设我们有 R1、R2 两台路由器，连接到了服务器群，服务器群使用两台三层交换机下挂着网络的服务器，服务器中我们规划了两个子网分别是生产的 10.1.1.0/24，以及办公 10.1.2.0/24。R3 是接入路由器。R1、R2、R3 跑 OSPF。

R1、R2 与三层交换机之间，假设是静态路由环境。那么现在，我们希望 R3 下的用户，在访问生产服务器到时候，流量往红色虚线箭头所指示的方向流动，访问办公服务器的时候往蓝色箭头方向流动。

那么首先 R1 及 R2 上，为了让他们自己能够到达服务器 10.1.1.0 及 2.0 网段，需要配置两条静态路由：

```
Ip route 10.1.1.0 255.255.255.0 10.1.254.1
Ip route 10.1.2.0 255.255.255.0 10.1.254.1
```

接着为了让 R3 能够动态学习到生产及办公服务器的路由，现在需要将这两条静态路由重发布进 OSPF，当然，在重发布的时候就有技巧了。

R1 的配置如下：

```
access-list 1 permit 10.1.1.0
```

```
access-list 2 permit 10.1.2.0
route-map cisco permit 10
      match ip address 1
      set metric 10
route-map cisco permit 20
      match ip address 2
      set metric 20
router ospf 100
   redis static route-map cisco
```

R2 的配置如下：

```
access-list 1 permit 10.1.1.0
access-list 2 permit 10.1.2.0
route-map cisco permit 10
      match ip address 1
      set metric 20
route-map cisco permit 20
      match ip address 2
      set metric 10
router ospf 100
   redis static route-map cisco
```

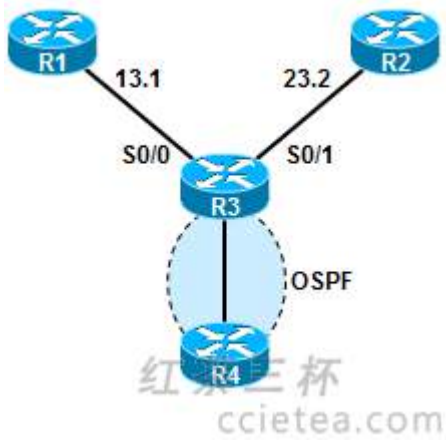
这样就实现了需求。

6.4.4 难点案例

1. **验证 match interface 的作用 1**

   一个 route-map 语句中，如果没有 match 语句，则匹配所有

   Match interface ：To distribute any routes that have their next hop out one of the interfaces specified, use the match interface command in route-map configuration mode 中文上的理解是，match interface 匹配的是下一跳出接口是这个接口的路由条目



在 R3 上：

Ip route 1.1.1.0 255.255.255.0 serial 0/0

Ip route 2.2.2.0 255.255.255.0 192.168.13.1

Ip route 3.3.3.0 255.255.255.0 serial 0/1

!

Route-map test permit 10

Match interface s 0/0

Router os 1

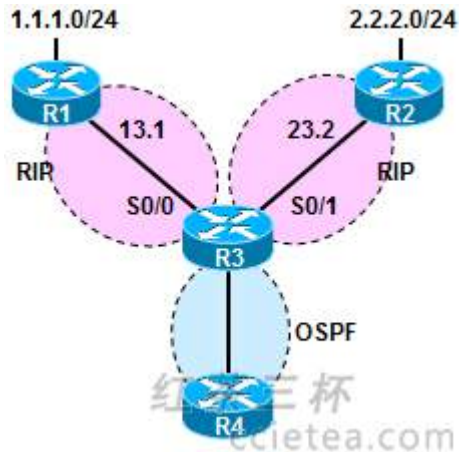
Redis static route-map test

那么在 R4 上，只能学习到 1.1.1.0。

2.2.2.0 路由虽然下一跳是指向 R1，但是并没有用出接口的方式创建路由条目，从实验现象看没有被 match 住。3.3.3.0 就不用说了，关联的接口是 s0/1。

所以 match interface，是 match 具有出接口属性的路由，并且这条路由的下一跳出接口是 match 的这个端口。

2. 验证 match interface 的作用 2



R1、R2、R3 之间跑 RIP，R1、R2 分别注入各自的 loopback 口

R3、R4 跑 OSPF ,在 R3 上能分别学习到来自 R1 和 R2 的 looback 路由，接下去：

R3 上配置如下：

route-map test permit 10

match interface Serial0/0

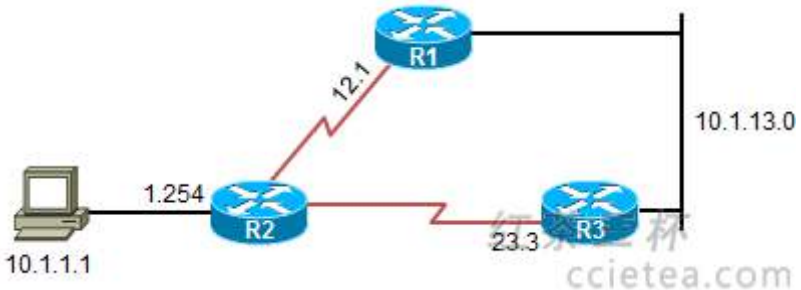
router ospf 1

redistribute rip subnets route-map test

此时在 R4 上，只能学习到 1.1.1.0/24 以及 192.168.13.0/24

此时 R3 上看到 1.1.1.0 这条路由，是包含下一跳属性的，而前一个例子中，关联下一跳的静态路由是没有这个属性，所以不被匹配

3. 验证 set ip default next-hop



先保证 R1、R3 到 10.1.1.0 是有路由的，在 R2 上做测试：

- **如果 R2 上没有任何的动、静态路由，且配置如下：**

```
access-list 1 permit 10.1.1.0 0.0.0.255
route-map test permit 10
 match ip address 1
 set ip default next-hop 10.1.12.1
```

则 PC ping 10.1.13.0，数据走 R1；

- 如果在上述基础上，R2 增加到 R3 的默认路由，则 PC 到 10.1.13.0 网络的数据仍被丢给 R1，也就是说 ip defaut-next-hop 的优先级高于默认路由。

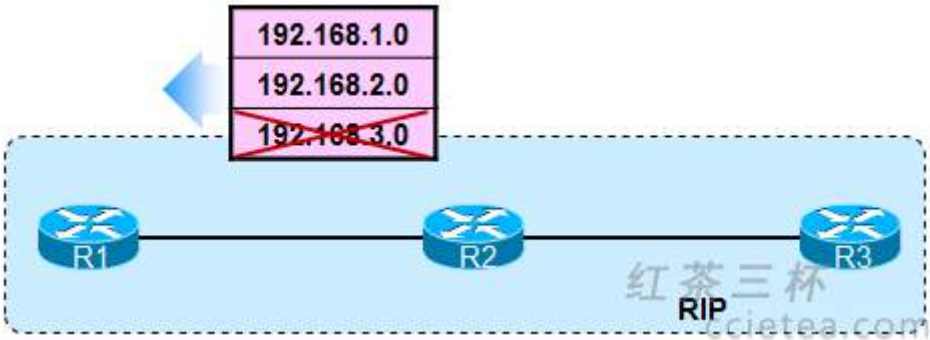
- No 掉上面配置的默认路由，再配一条去往 13.0 网络的路由，下一跳为 R3，则 PC 到 13.0 网络的数据切换到 R3 证明 ip default next-hop 的优先级低于明细路由，高于默认路由 。

- 再次验证 ,不用明细路由 ,而是用一条 ip route 10.0.0.0 255.0.0.0 的汇总路由 ,下一跳为 R3 ,效果同上，也走 R3。因此只要不是默认路由，只要路由表中存在这么一条匹配的路由，则优先走路由，没有路由的情况下走 route-map。

6.5 distribute-list

6.5.1 工具概述

用于控制路由更新的一个工具，只能过滤路由信息，不能过滤 LSA。



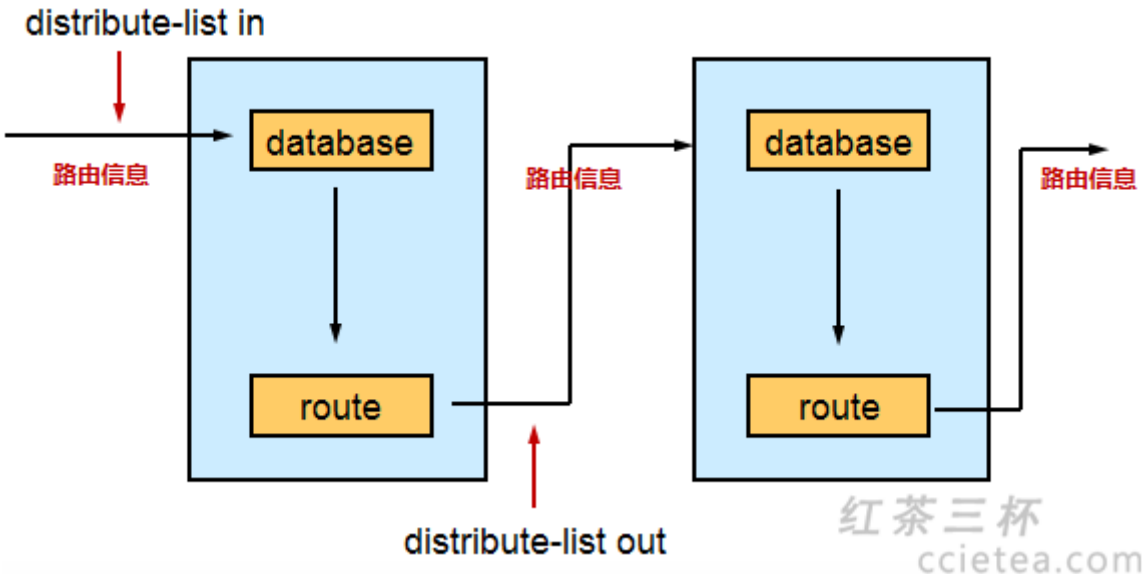
如上图，R1、R2、R3 运行 RIP。R2 在初始情况下，会将自己的路由表更新给 R1，其中假设包含三条路由 1.0、2.0 及 3.0。现在我们可以通过在 R2 上部署分发列表 distribute-list，使得 R2 在更新给 R1 的路由信息中过滤掉 3.0 这条路由。这就是分发列表的一个使用示例。当然，它还有更加广泛的应用。

6.5.2 部署要点

分发列表是**用于控制路由更新的一个工具，只能过滤路由信息，不能过滤 LSA。因此：**分发列表在距离矢量路由协议中使用，无论是 in 或者是 out 方向，都能正常的过滤路由。但是在链路状态路由协议中的工作就有点问题了。

The command distribute-list out works only on the routes being redistributed by the Autonomous System Boundary Routers (ASBRs) into OSPF. It can be applied to external type 2 and external type 1 routes, but not to intra-area and interarea routes.

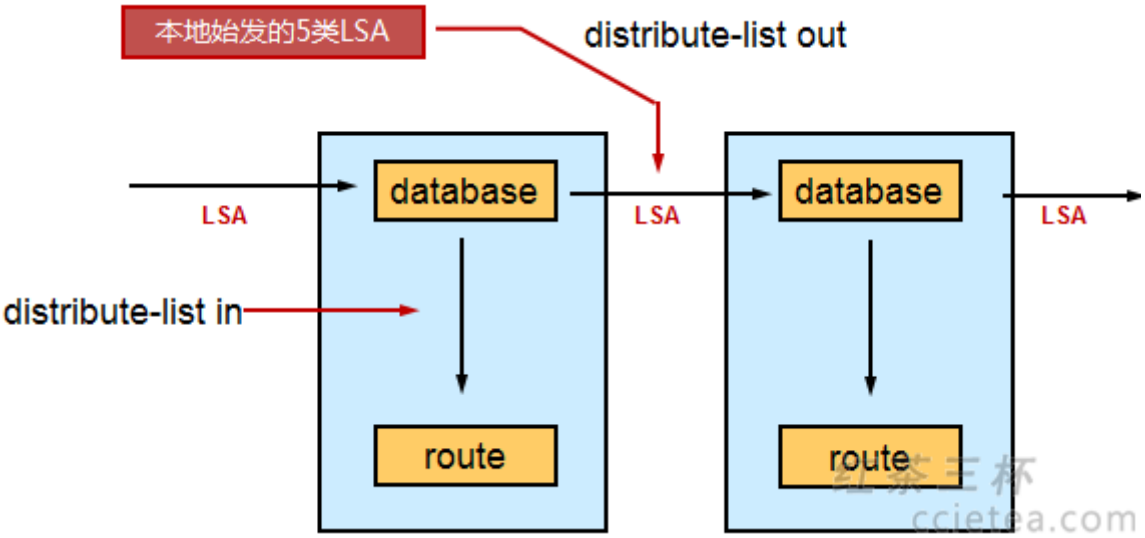
1. 对于距离矢量路由协议



路由器之间，传递的是路由信息，分发列表对路由信息是有绝对的控制权的。因此如果是 in 方向，那么通过部署分发列表，可以过滤特定的路由，使得执行分发列表的本地路由路由表发生变化，同时，本地路由器在更新路由信息给下游路由器的时候，实际上更新的内容是受分发列表影响之后的条目。

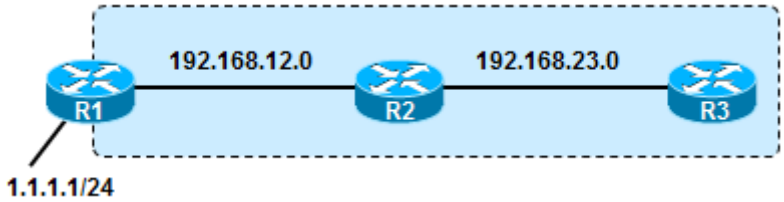
同时在 out 方向，也是没有问题的。

2. 对于链路状态路由协议，如 OSPF



值的注意的是 ,对于 OSPF 这样的链路状态路由协议 ,路由器之间传递的消息不再是路由信息了 ,而是 LSA，而分发列表是无法对 LSA 进行过滤的。因此，在链路状态协议中部署分发列表，就需要留意了：

- in 方向，分发列表只能在本地收到 LSA 后，生成路由的那一刹那进行路由的过滤，执行分发列表的路由器自己路由表会被分发列表影响（但是本地 LSDB 仍然是有 LSA 的）,而且该路由器仍会将 LSADB 中的 LSA 发送给邻居，因此本地被过滤的路由，邻居还有（因为邻居已经收到 LSA 了）。
- out 方向，分发列表只能工作在执行路由重发布动作的那个 ASBR 上，且只能针对外部引入的路由起作用。因为 OSPF 执行重发布时，其实这些外部路由是以路由的形式引入进来的，因此分发列表在这个场合下能够正常工作，但是如果不是本地始发的外部路由，或者是内部的 OSPF 路由，out 方向的分发列表均束手无策。



例如在 R1 上重发布直连进 OSPF，用 out 方向的分发列表可过滤掉 1.1.1.0 这条外部路由。但 R1 重发布进来的路由，如果在 R2 上用 out 方向的分发列表试图阻挡 R3 接受路由或 LSA，则无法，因为这不是本地始发的外部路由。

6.5.3 配置命令

1. **In 方向**

```
R1(config-router)#distribute-list 1 in ?      // 都是接口
  Async              Async interface
  BVI                Bridge-Group Virtual Interface
  CDMA-Ix            CDMA Ix interface
  Dialer             Dialer interface
  FastEthernet       FastEthernet IEEE 802.3
  Multilink          Multilink-group interface
  Port-channel       Ethernet Channel of interfaces
  Tunnel             Tunnel interface
  Vif                PGM Multicast Host interface
  Virtual-PPP        Virtual PPP interface
  Virtual-Template   Virtual Template interface
  ......
```

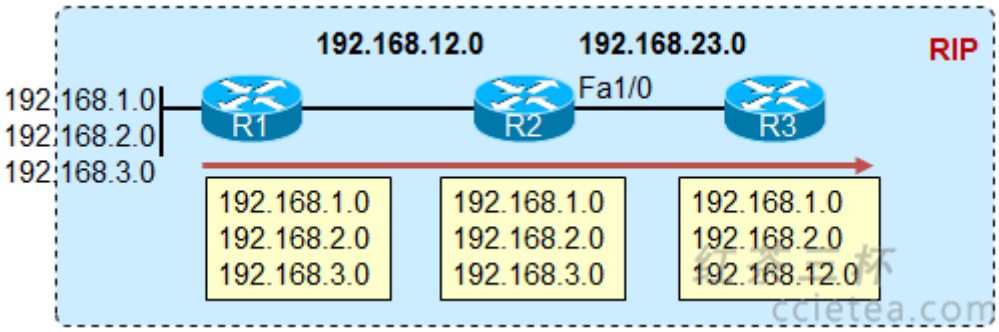
2. **OUT 方向**

```
R1(config-router)#distribute-list 1 out ?              // 接口或协议
  Async              Async interface
  BVI                Bridge-Group Virtual Interface
  Dialer             Dialer interface
  FastEthernet       FastEthernet IEEE 802.3
  Loopback           Loopback interface
  Multilink          Multilink-group interface
  Port-channel       Ethernet Channel of interfaces
  Tunnel             Tunnel interface
  Virtual-PPP        Virtual PPP interface
  Virtual-Template   Virtual Template interface
  Virtual-TokenRing  Virtual TokenRing
```

bgp	Border Gateway Protocol (BGP)
connected	Connected
eigrp	Enhanced Interior Gateway Routing Protocol (EIGRP)
ospf	Open Shortest Path First (OSPF)
rip	Routing Information Protocol (RIP)
static	Static routes
.....	
<cr>	

6.5.4 应用场合

1. **配置示例 1（单一路由协议环境下-RIP）**

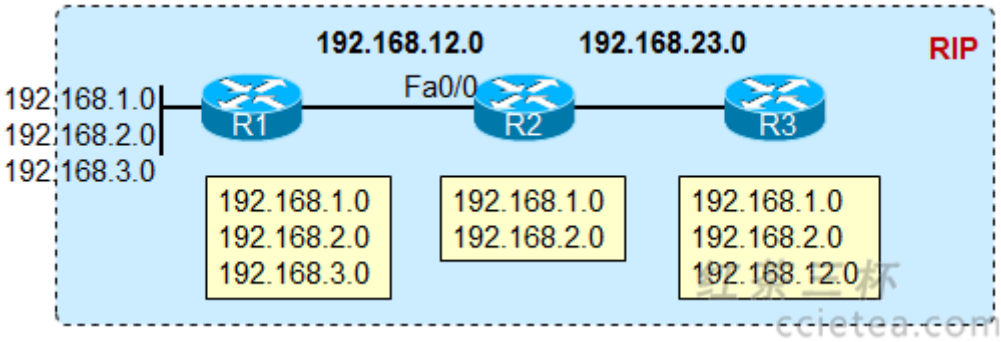


初始情况下，R3 能够学习到 R1 的三条 loopback 路由，以及 192.168.12.0/24 路由。现在我们不希望 R3 学习到 192.168.3.0/24 的路由，那么可以在 R2 上如下配置：

```
R2(config)# access-list 1 deny 192.168.3.0
R2(config)# access-list 1 permit any
R2(config)# router rip
R2(config-router)# distribute-list 1 out fa 1/0
```

当然，在 R3 上，用 in 方向的分发列表也可以达到同样的效果。

2. **配置示例 2（单一路由协议环境下-RIP）**

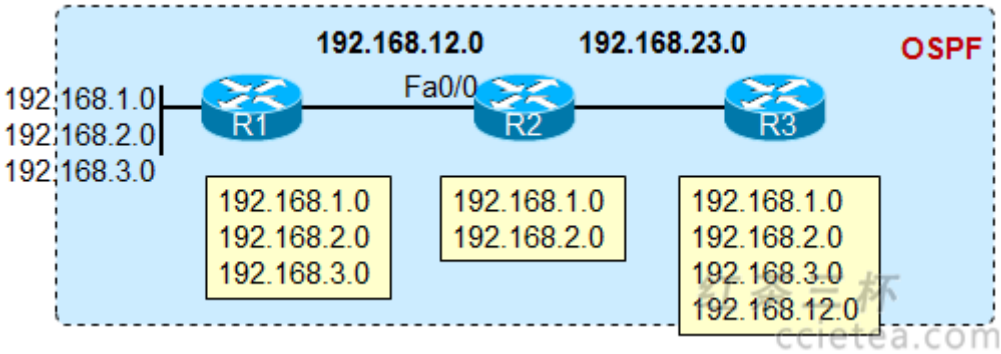


在 R2 上如果做如下配置：

```
R2(config)# access-list 1 deny 192.168.3.0
R2(config)# access-list 1 permit any
R2(config)# router rip
R2(config-router)# distribute-list 1 in fa0/0
```

那么，首先 R2 自己的路由表会发生改变，3.0 的路由被过滤掉了，同时 R3 也就是下游 RIP 路由器，3.0 也学不到。

3. 配置示例 3（单一路由协议环境下-OSPF）

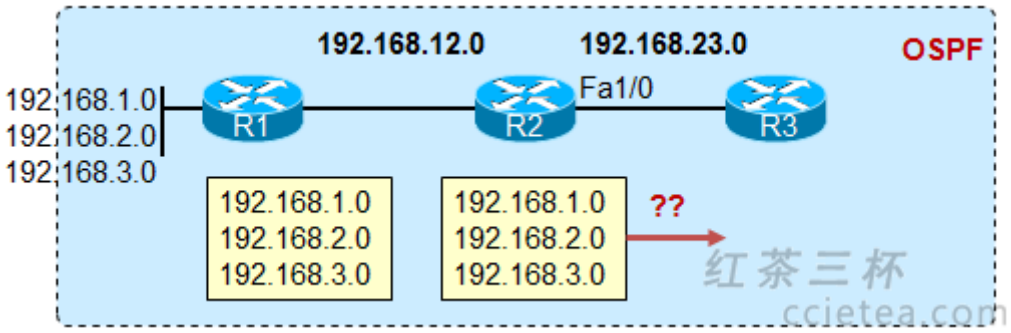


R2 的配置如下：

```
R2(config)# access-list 1 deny 192.168.3.0
R2(config)# access-list 1 permit any
R2(config)# router ospf 1
R2(config-router)# distribute-list 1 in fa0/0
```

注意这时候，首先在 R2 的路由表里，3.0 的路由就被干掉了。注意，这时候实际上，area 内 OSPF 路由器产生的 LSA 已经是装载到了 R2 的 OSPF database 之中，而在 R2 从 OSPF database 中计算路由，并准备将路由条目装载进路由表之前，in 方向的分发列表发生作用了，将 3.0 的路由过滤掉了，因此 R2 的路由表中，是没有 3.0 的 OSPF 路由的。但是，虽然 R2 自己路由表里没 3.0 路由，这不妨碍 R2 将相关 LSA 泛洪给 R3，因此，R3 仍然是有 1.0、2.0、3.0 以及 12.0 的 OSPF 路由的。

4. **配置示例 4（单一路由协议环境下-OSPF）**

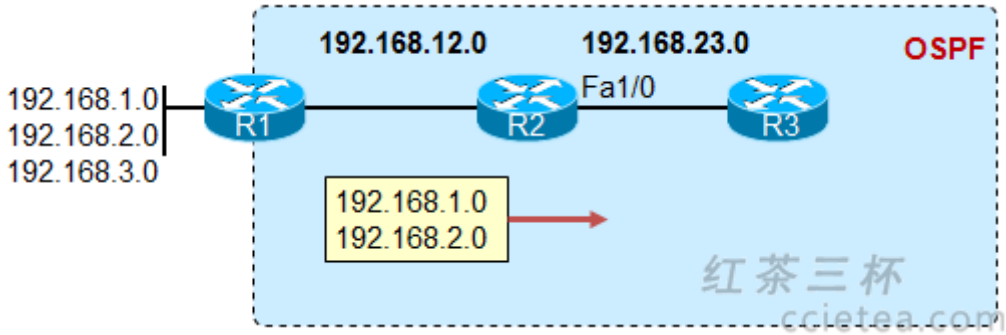


现在我们在 R2 上做如下配置：

```
R2(config)# access-list 1 deny 192.168.3.0
R2(config)# access-list 1 permit any
R2(config)# router ospf 1
R2(config-router)# distribute-list 1 out
```

R3 的路由表会是什么情况？实际上，没有任何影响，R3 能学习到全网的路由。至于为什么，我相信前面已经解释的非常清楚了。

5. **配置示例 5（单一路由协议环境下-OSPF out 方向分发列表）**



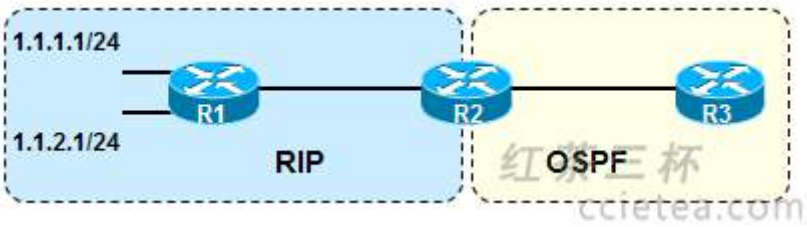
分发列表，部署在 OSPF 这样的链路状态路由协议中，如果要用 out 方向，则只能用在这样的场合。

如上图，在 R1 上部署，R1 使用重发布直连的方式引入这三条外部路由，那么 out 方向的分发列表，只能在 R1 上部署，且对这三条路由产生作用。

```
R1(config)# access-list 1 deny 192.168.3.0
R1(config)# access-list 1 permit any
R1(config)# router ospf 1
R1(config-router)# redistribute connected subnets
R1(config-router)# network 192.168.12.1 0.0.0.0 area 0
R1(config-router)# distribute-list 1 out
```

上述配置实现后，R1 将过滤掉 3.0 路由。

6. 配置示例 6 协议间重发布时部署分发列表



RIP 重发布进 OSPF

- **情况 1**

  R2 的配置如下：

```
access-list 1 permit 1.1.1.0
router ospf 1
 redistribute rip metric 10 subnets
 distribute-list 1 out rip
```

  这里这条命令的意思是，从 RIP 路由协议重分发过来的路由中，只允许 1.1.1.0 出去（到 OSPF 协议，没有方向，只要是运行了 OSPF 的接口）

  R3 的路由表里，只有 1.1.1.0 的路由

- **情况 2**

  在 R2 上开设 loopback 接口 2.2.2.0/24，R2 既重发布 RIP 进 OSPF，又重发布直连进 OSPF

```
access-list 1 permit 1.1.1.0
router ospf 1
 redistribute connected subnets
 redistribute rip metric 10 subnets
 network 192.168.23.0 0.0.0.255 area 0
 distribute-list 1 out
```

  // 在 R3 上只有 1.1.1.0 的路由，也就是说 distribute-list 1 out 此处这条命令，对所有从外部注入进 OSPF 的路由都生效，最终只有 1.1.1.0 路由存活下来。而不断路由的来源是直连路由，还是 RIP。

- **情况 3**

  在 R2 上开设 loopback 接口 2.2.2.0/24，R2 既重发布 RIP 进 OSPF，又重发布直连进 OSPF

```
access-list 1 permit 1.1.1.0
router ospf 1
```

```
redistribute connected subnets
redistribute rip metric 10 subnets
distribute-list 1 out rip
```

// R3 的路由表中有路由：1.1.1.0 、 2.2.2.0 、 192.168.12.0

// 也就是屏蔽掉了从 RIP 重发布进来的除了 1.1.1.0 以外的路由，并重发布本地直连接口

6.6 路由匹配工具

6.6.1 ACL

1. **标准 ACL**

   标准 ACL 只能匹配路由前缀，无法匹配路由的前缀长度，例如，如果想抓取 192.168.1.0/24 这条路由，用 access-list 1 permit 192.168.1.0，则该条路由被匹配，但是同时，192.168.1.0/25、/26.....也都被匹配了，因为 ACL 无法匹配掩码，或者说，前缀长度。再者，在使用标准 ACL 抓取路由的时候，建议不加反掩码，否则被匹配的路由条目范围将更大更不精确。

2. **扩展 ACL**

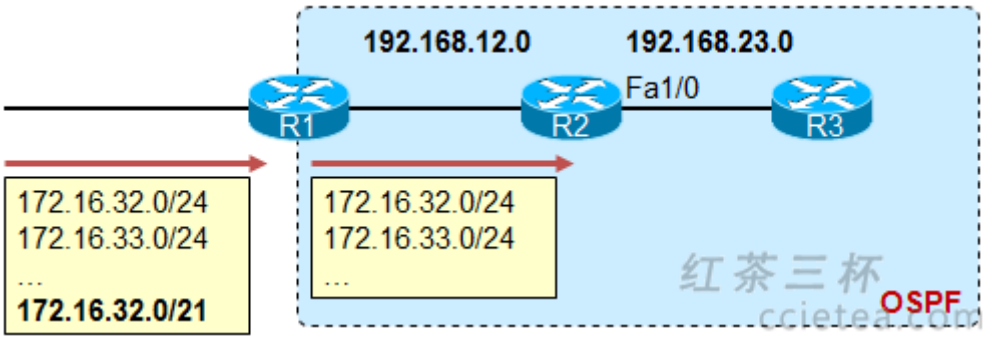
   标准 ACL 有源部分、目的部分，使用源匹配路由前缀，使用目的部分匹配路由掩码。

   例如，要抓取 192.168.1.0/24 这条路由，则 access-list 100 permit ip 192.168.1.0 0.0.0.0 255.255.255.0 0.0.0.0。

   所以其实很简单，只要把路由的前缀和掩码部分，分别使用 ACL 的源和目的部分进行匹配即可。

6.6.2 Prefix-list

1. **技术背景**



我们看上图，外部明细路由 172.16.32.0 – 39.0/24，以及汇总路由 32.0/21 被 R1 引入 OSPF，现在需在 R1 上，仅将汇总路由 32.0/21 过滤，而所有明细路由放行，如果使用标准 ACL 匹配路由，该如何写？

```
R1(config)# access-list 1 deny 172.16.32.0
R1(config)# access-list 1 permit any
```

仔细思考一下会有什么问题，实际上 access-list 1 deny 172.16.32.0 是一并把汇总路由 172.16. 32.0/21 和明细路由 172.16.32.0/24 给匹配上了，因此最终这两条路由都会被过滤掉，这就与我们的需求不符了。

其实这就是用标准的 ACL 去匹配路由的弊端，你只能匹配路由的网络号，而无法进一步匹配路由的前缀长度，或者说掩码长度。在配上上面这条 ACL 的时候，有些同学甚至会写成 access-list 1 deny 172.16.32.0 0.0.0.255 实际上，这就更有问题了，因为这种写法，实际上最后一个 8 位组不管是什么，都会被匹配住，他就更不精确了。

2. 关于前缀列表

- **可匹配路由前缀中的网络号及前缀长度，增强了匹配的精确度**
- **前缀列表的可控性比访问列表高得多，支持增量修改，更为灵活**
- **前缀列表包含序列号，从最小的开始匹配**
- **如果前缀不与前缀列表中的任何条目匹配，将被拒绝**

3. 前缀列表的配置

```
router(config)# ip prefix-list {list-name [seq number] {deny | permit} network/length [ge ge-value] [le le-value]
```

参数	描述
ge ge-value	要匹配的前缀范围，范围为 ge-value 到 32
le le-value	要匹配的前缀范围，范围为 length 到 le-value

输入条件：length < ge-value < le-value <= 32

4. 配置示例

```
ip prefix-list ABC seq 5 permit 172.0.0.0/8
```

路由前 8bits 必须与 172.0.0.0 的前 8bits 完全匹配，其他位不关心，并且掩码必须是/8 的

```
ip prefix-list ABC seq 5 permit 172.0.0.0/8 le 24
```

路由前 8bits 必须与 172.0.0.0 的前 8bits 完全匹配，其他位不关心，并且掩码必须是/8，/9，/10，.....，/24

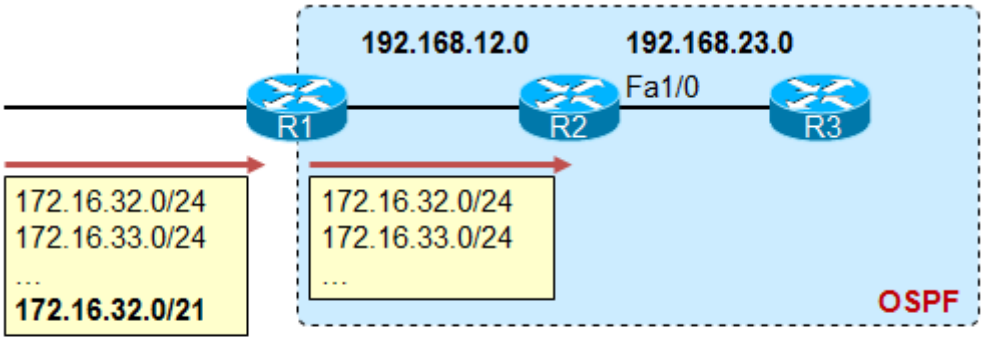
```
ip prefix-list ABC seq 5 permit 172.0.0.0/8 ge 24
```

路由前 8bits 必须与 172.0.0.0 的前 8bits 完全匹配，其他位不关心，并且掩码长度必须大于 24，注意，这里 le 关键字没写，那么默认是 大于 24 小于 32

```
ip prefix-list ABC seq 5 permit 0.0.0.0/0 le 32
```

路由的前 0bits 必须与 0.0.0.0 的前 0 个 bit 匹配，实际上就是所有 bits 都无所谓了，而且对于掩码，以为没有写 ge 关键字 ,所以隐含的是 ge 0 le 32，也就是掩码长度大于 0 小于等于 32。所以这条前缀列表就是 permit any

5. 应用示例



外部路由172.16.32.0 – 39.0/24，以及汇总路由32.0/21被R1引入OSPF
现在需在注入过程中，仅将汇总路由32.0/21过滤，所有明细放行。

```
R1(config)# ip prefix-list list1 deny 172.16.32.0/21
R1(config)# ip prefix-list list1 permit 0.0.0.0/0 le 32
R1(config)# route-map test permit 10
R1(config-route-map)# match ip address prefix-list list1
```

6.6.3 路由标记 Tag

1. **OSPF TAG 概述**

   TAG 字段（32bits）只在外部 LSA 中存在。

   在 ASBR 重发布时可以使用多种方式修改外部 LSA 的 TAG 值。

   - **修改 ASBR 产生的所有外部 LSA 的 TAG**

     ```
     r1(config-router)#redistribute rip subnets tag ?
       <0-4294967295>  32-bit tag value
     ```

     当然，也可以在重发布命令后关联 route-map，在 route-map 中设置 tag

   - **修改 ASBR 通告的汇总 LSA 的 TAG**

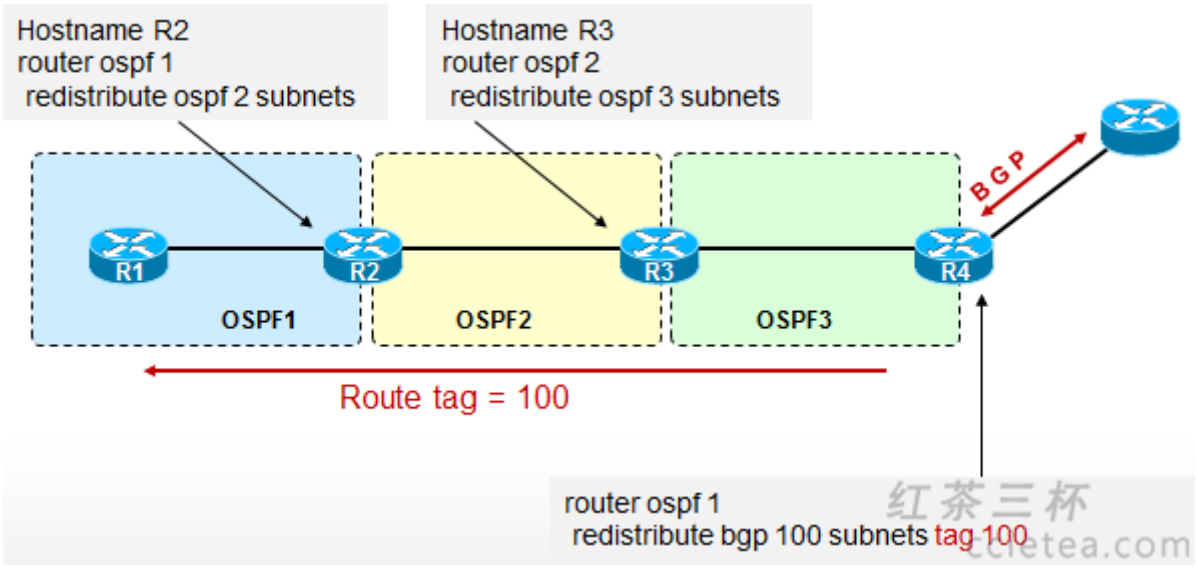
     ```
     r1(config-router)#summary-address 10.0.0.0 255.0.0.0 tag ?
       <0-4294967295>  32-bit tag value
     ```

   下面是 5 类 LSA 报文中，TAG 字段的位置：

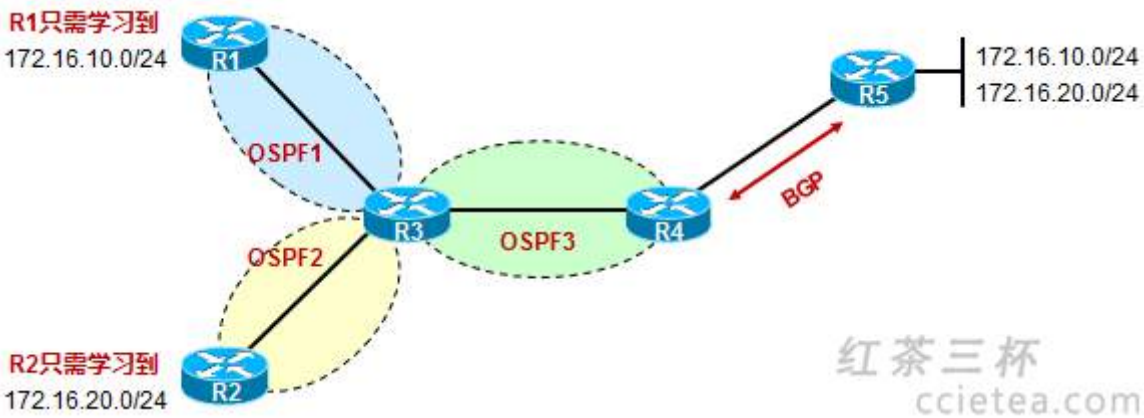
   ```
   ⊞ OSPF Header
   ⊟ LS Update Packet
       Number of LSAs: 1
     ⊟ LS Type: AS-External-LSA (ASBR)
         LS Age: 2 seconds
         Do Not Age: False
       ⊞ Options: 0x20 (DC)
         Link-State Advertisement Type: AS-External-LSA (ASBR) (5)
         Link State ID: 3.3.3.0
         Advertising Router: 3.3.3.3 (3.3.3.3)
         LS Sequence Number: 0x80000001
         LS Checksum: 0x216a
         Length: 36
         Netmask: 255.255.255.0
         External Type: Type 2 (metric is larger than any other lir
         Metric: 20
         Forwarding Address: 0.0.0.0
         External Route Tag: 0
   ```

2. **外部 LSA 中的 TAG 值的传递范围**

   在始发的 ASBR 产生外部 LSA 时设置了 TAG 值，如果该外部 LSA 重发布进其他 OSPF 自治系统，TAG 值为默认携带。



3. 实验示例：重发布时调用



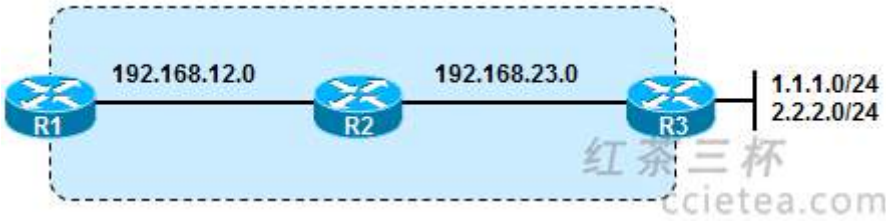
R4 的配置如下：

```
access-list 1 permit 172.16.10.0
access-list 2 permit 172.16.20.0
route-map BGP2OSPF permit 10
   match ip address 1
   set tag 10
route-map BGP2OSPF permit 20
   match ip address 2
   set tag 20
router ospf 3
 redistribute BGP 1 subnets route-map BGP2OSPF1
```

R3 的配置如下：

```
route-map OSPF3to1 permit 10
  match tag 10
route-map OSPF3to2 permit 20
  match tag 20
router ospf 1
  redistribute ospf 3 subnets route-map OSPF3to1
router ospf 2
  redistribute ospf 3 subnets route-map OSPF3to2
```

4. 实验示例：在分发列表中调用



R3 在注入路由的时候，设置上 tag

R3 的配置如下

```
access-list 1 permit 1.1.1.0
access-list 2 permit 2.2.2.0
router ospf 1
  redistribute connected metric 10 subnets route-map test
  network 192.168.23.3 0.0.0.0 area 0
route-map test permit 10
 match ip address 1
 set tag 10
route-map test permit 20
 match ip address 2
 set tag 20
```

R2 的配置如下：

```
router ospf 1
  network 192.168.12.2 0.0.0.0 area 0
```

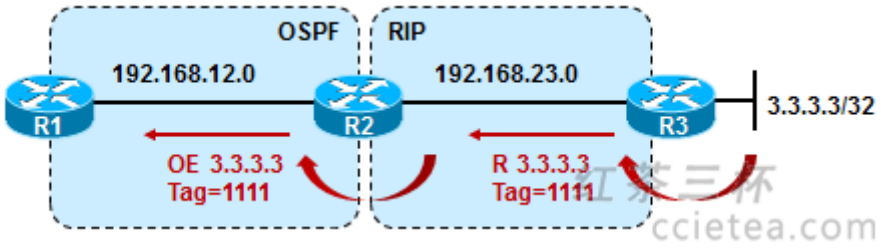
```
 network 192.168.23.2 0.0.0.0 area 0
 distribute-list route-map test in
route-map test permit 10
 match tag 10
```

【结果】R2 上路由表中只有 1.1.1.0 的路由。

R1 上有 1.1.1.0、2.2.2.0 的路由

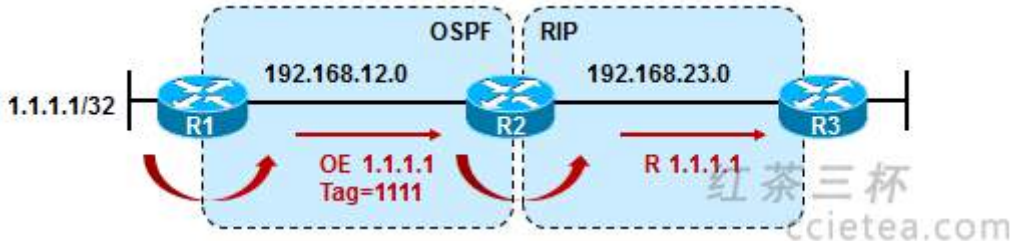
5. Tag 值的传递

RIP 也是支持 tag 的，但是必须是 version 2。

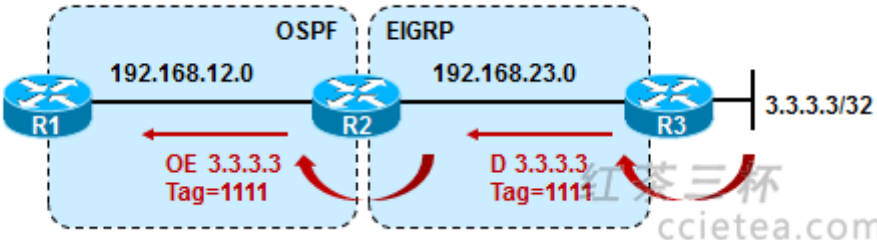


R3 在重发布直连路由 3.3.3.3/32 的时候关联 route-map，打上 tag1111，这条路由传递到了 R2。在 R2 上，部署 RIP 到 OSPF 的重发布，那么 3.3.3.3 的外部路由注入 OSPF 后，tag 值是默认携带的。

然而如果在 R1 上发布一条携带 tag 的外部路由，并且在 R2 上重发布 OSPF 进 RIP，tag 丢失：

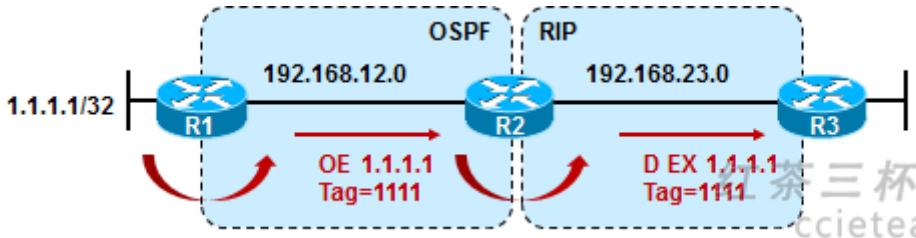


EIGRP 也是支持 tag 的。



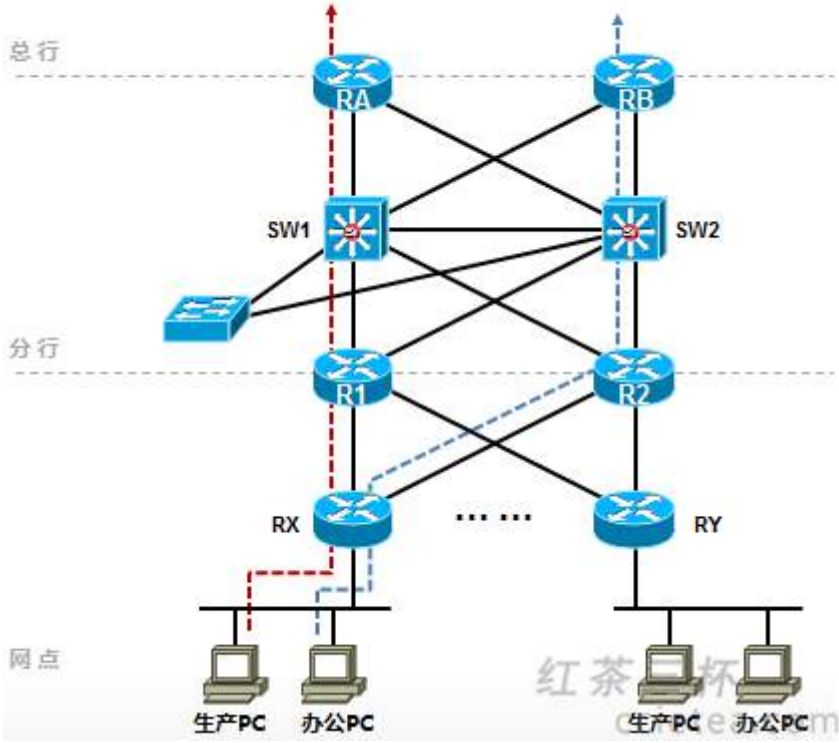
R3 在重发布直连路由 3.3.3.3/32 的时候关联 route-map，打上 tag1111，这条路由传递到了 R2。在 R2 上，部署 RIP 到 OSPF 的重发布，那么 3.3.3.3 的外部路由注入 OSPF 后，tag 值是默认携带的。而如果在 R1 上发布一条携带 tag 的外部路由，并且在 R2 上部署 OSPF 到 EIGRP 的重发布，那么这条路由

在注入到 EIGRP 后，tag 也依然携带。



7 路径控制

7.1 路径控制概述



严格的说，路径控制是一个非常大的课题，在一个大型网络的部署中，往往需要费尽心思考虑对数据流量访问路径的控制，为的是更加合理的、科学的利用和分配网络资源，同时增强网络的可靠性、冗余性和健壮性。

而实现数据访问路径控制的需求，方法和工具往往有非常非常多，如果一个网络前期的 IP、VLAN 等基本元素规划的非常科学和合理，那么在策略部署这块就更加的轻松和选择多样。本章不会详细讨论各种路径控制的工具，仅仅列举几个常见的工具和方法做个讨论。其实利用前面所学的工具，诸如 route-map 等，通过在路由控制

层面执行多样化的策略，已经能够起到非常不错的路径控制的效果，而且，通过控制路由来控制数据流走向，是一个非常科学也非常建议的方法。除此之外，还有许多，如：

再次强调一下，这里只是简单的各种工具的罗列，在实际网络的部署中，需要根据实际的情况，灵活的挑选最经济、最科学、最可靠的工具和方法来部署。

7.2 Offset-list 偏移列表

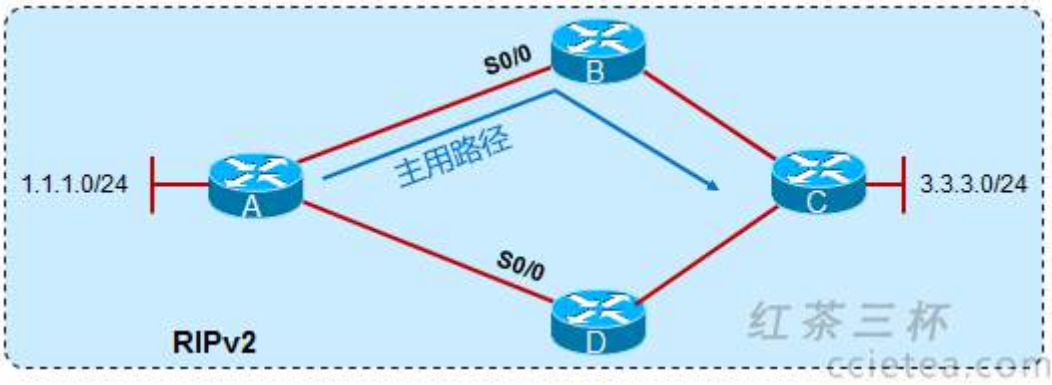
1. **技术概述**

   用于在入站或出站时增大通过 EIGRP 或 RIP 获悉的路由度量值。

2. **配置命令**

```
router(config-router)#
  offset-list {access–list-number | name} {in|out} offset [interface-type interface-number]
```

3. **配置示例 RIP 环境**

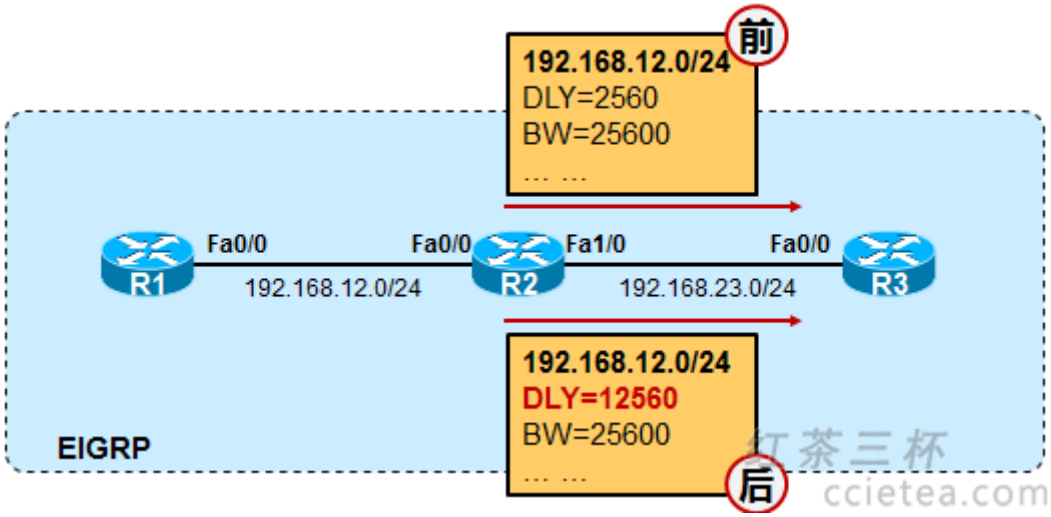


1.1.1.0 到 3.3.3.0 实际上有两条路径可走，如果网络中运行 RIP 协议，实际上对于 A 而言去往 3.3.3.0/24 是可以通过 B 和 D 负载分担的。但是在某些情况下，我们更希望数据的走向是可控的，例如，我们希望 1.1.1.0 访问 3.3.3.0 的流量主走 B，当 B 挂掉了，则切换到 D 上。那么我们只要简单的在 D 上部署 offset-list，在其向 A 通告 3.3.3.0 路由时，增加 1 跳，那么这条路由就会相比 B 通告给 A 的路由 metric 大 1 跳，A 自然会优选 B。

```
access-list 1 permit 3.3.3.0
router rip
  offset-list 1 out 1 serial 0/0          !! 红色字体部分为要增加的跳数
```

这里补充一句，既然关注了流量，就要注意流量的往返，一般情况下，我们希望 1.1.1.0 访问 3.3.3.0 的流量走 ADC，那么往返流量一般都是需要路径一致的。也就是说 3.3.3.0 到 1.1.1.0 的数据走向是 CDA，所以要进一步在 D 上，将更新给 C 的 1.1.1.0 的路由的 metric 加 1 跳。

4. 配置示例 EIGRP 环境



初始情况下 R2 更新 192.168.12.0/24 路由给 R3 的时候，我们看一下，updata 报文中，该路由前缀携带的各项属性：DLY=2560，因为 R2 F0/0 接口的 DLY 是 100 微妙，而报文中的单位是 10us，因此，10*256=2560，这个值就是报文中 DLY 字段携带的值。带宽的值也类似。这都是可以算出来的，没啥问题。

那么现在，我们在 R2 的配置上增加如下：

```
access-list 1 permit 192.168.12.0
router rip
 offset-list 1 out 10000 fastEthernet 1/0
```

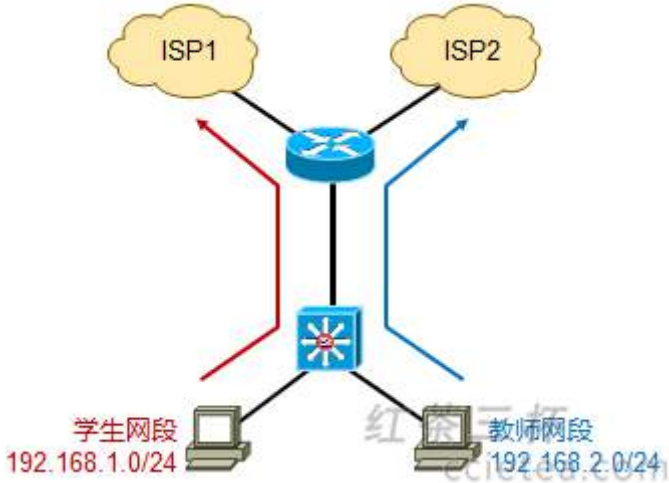
这条命令的直接结果是，R3 原本路由表中 192.168.12.0 的 metric 为 30720，而现在则加了 10000，变成 40720。我们看到，配置了 offset 命令后，实际上变化的是更新给邻居的 DLY。因为确实 DLY 最好算，默认的 metric=BW+DLY，而其中 DLY 是路由沿途入接口的 DLY 的累加，而 BW 则是最小接口带宽。显然通过 DLY 来控制，最简单方便。

7.3 Policy-based Routing（PBR）策略路由

7.3.1 关于 PBR

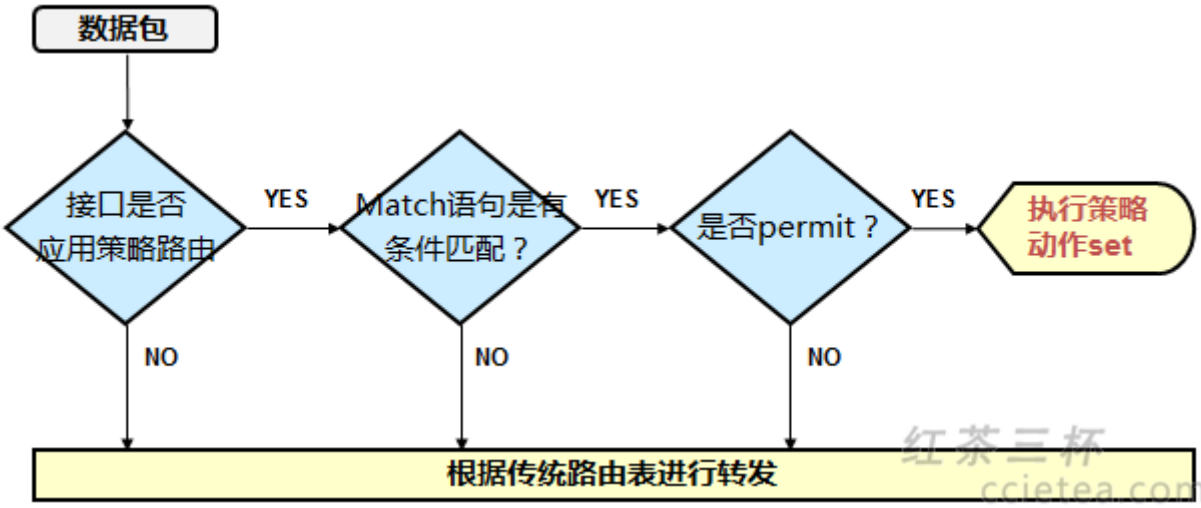
传统的 IP 路由的概念我们再回顾一下：当路由器收到一个 IP 数据包的时候，查看数据包的 IP 包头，将包头的目的 IP 拿到路由表中按最长匹配原则进行比对，最后，将数据包转发出去。因此，传统的 IP 路由只能根据数据的目的进行选路控制。

基于策略的路由比传统路由能力更强，使用更灵活，它使网络管理者不仅能够根据目的地址而且能够根据协议类型、报文大小、应用或 IP 源地址来选择转发路径。



例如上图中，网络有两条出口线路，分别连接到 ISP1 及 ISP2，如果希望让内网学生用户，访问外网的时候走 ISP1 的线路，而教师网段访问外网的时候走 ISP2 线路，这是传统路由无法实现的，因为传统的 IP 路由选择不会去关心数据的源地址。这就必须借助 PBR 了。

当我们在一个接口上部署了 PBR 后，如果这个接口收到一个数据包，它将：



7.3.2 命令汇总

1. **Route-map xx permit 10**

   这个就不多说了吧？对于 PBR 而言，在创建 route-map 的时候，都是 permit 的。

2. **match ip address**

   后面跟上 ACL，用于匹配流量。

3. **set interface**

   设置数据的出接口

4. **set ip next-hop** { ip-address [...ip-address] | **recursive** ip-address }

   允许写多个下一跳 IP，但这些 IP 必须是直连路由器的接口 IP。

   如果定义了多个下一跳 IP ,则当第一个下一跳关联的本地出接口 DOWN 掉 ,则自动切换到下一个 next-hop。

   - **recursive next-hop（递归下一跳）** 特性突破了传统下一跳必须是直连路由器下一跳接口 IP 的限制。Recursive next-hop 可以不是直连网络，只要路由表中有相关的路由可达即可。一般 recursive next-hop 不可达，数据将交由路由处理（一般就被默认路由匹配走了）
   - 如果在一个 route-map 列表的同一个序列中同时使用 ip next-hop 及 ip next-hop recursive ,则 ip next-hop 有效。如果 ip next-hop 挂了 ,则启用 ip next-hop recursive ,如果 ip next-hop recursive 和 ip next-hop 都挂了，则丢给路由表处理。注意：一个 route-map 序列（如 route-map test permit 10），只允许配置一个

ip next-hop recursive

5. **set ip next-hop verify-availability** [next-hop-address sequence track object]

   检测下一跳的可达性，默认是关闭的

   Sequence of next hops. The acceptable range is from 1 to 65535.

   此条命令可以下列方式使用：

   - **在 PBR 环境下使用 CDP 检测下一跳 IP 可达性（不加后面的可选参数）**

     使用该特性可能会一定程度上降低设备性能，另外必须保证自己以及邻居路由器接口 CDP 都是开启的，过程交换及 CEF 都支持该特性，但 dCEF 不支持。

     该特性借助设备的 CDP 表来判断下一跳的可达性，

     如果本端开启了该特性，next-hop 设备不支持 CDP，则切换至下一个 next-hop，如果没 next-hop 了，则跳过 PBR

     如果本端没开启该特性，那么数据包要么被成功策略路由，要么永远无法正常路由出去（被丢弃）

     如果仅仅想检测部分 next-hop 设备的可达性，则可以配置不同的 route-map 序列号，来选择性的使用该特性（在同一个 route-map 中）。

   - **结合 object tracking 来检测一个远端设备（或 IP）的可达性**

     使用 object tracking,PBR 可以做的更加灵活，可依据 ICMP、HTTP、路由表中某条路由的存在与否、接口的 up/DOWN 等来进行决策。

     注意： 如若基于 CDP 的检测及基于 object tracking 的检测都应用了，则后者优先

6. **set ip next-hop 与 set ip default next-hop 的区别比较简单，这里就不解析了**

   在本文档的 route-map 章节有介绍

7. **ip policy route-map *x***

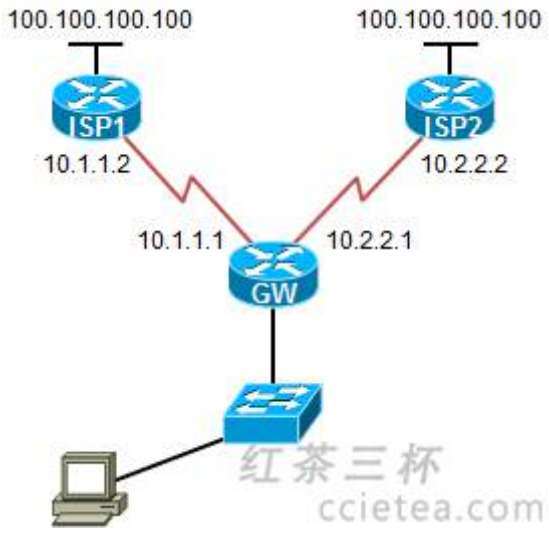
   在接口上定义好的 route-map。这种方式，将只会对进入该接口的流量执行 PBR 动作。而对本设备始发的流量无效。

8. **ip local policy route-map *x***

   这条全局配置命令，可以使得 PBR 对本地始发的流量生效。

7.3.3 实验验证

7.3.3.1 set ip next-hop



GW 的配置如下：

```
access-list 1 permit any
route-map PBR permit 10
 match ip address 1
 set ip next-hop 10.1.1.2 10.2.2.2
interface fast 1/0
  ip policy route-map PBR
!! GW 并无其他关于路由的配置
```

实验现象：

1. 当网络正常时，数据强制走 ISP1，ping 100 的远程网络数据到 ISP1
2. 当 ISP1 宕机时，GW 连接 ISP1 的接口 DOWN 掉，则 PC 访问 100 的流量自动切换至 ISP2
3. 当 ISP1 宕机时，且 GW 检测不到时（也就是 GW 连接 ISP1 的接口没 DOWN），PC 访问 100 的流量仍然被扔给 ISP1，这就断网了

补充：

Set ip next-hop ip1 ip2 ip3，这个知识点已经没问题了吧？match 住相关条件后，数据包首先被送到第一个 next-hop ip address，如果这个 ip 地址所关联的直连接口 DOWN 了，则切换至下一个 next-hop ip address，如此反复，可以配置多个 next-hop。但是，如果直连的 next-hop（对端路由器或其接口）自己挂了，而本地直连接口没感知到（如中间串了台 switch），则无法自动切换，路由器仍然会一股脑的把数据丢给这个 next-hop。

另外，如果配置的时候命令这么写的话：

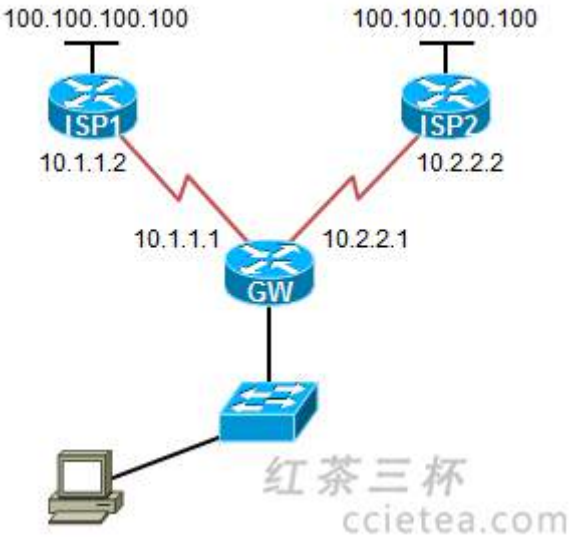
Set ip next-hop ip1

Set ip next-hop ip2

.....

则 IOS 会自动将命令变成 Set ip next-hop ip1 ip2 .....

7.3.3.2 set ip next-hop verify-availability



GW 的配置如下：

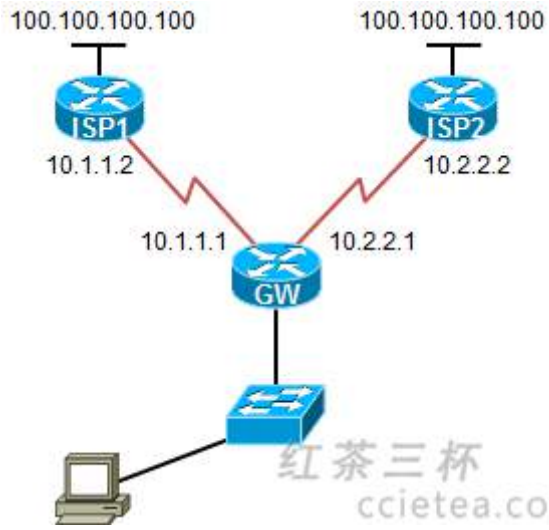
```
access-list 1 permit any
route-map PBR permit 10
 match ip address 1
 set ip next-hop 10.1.1.2 10.2.2.2
 set ip next-hop verify-availability
interface fast 1/0
   ip policy route-map PBR
!! GW 并无其他关于路由的配置
!! ISP1、ISP2 及 GW 都需开启 CDP
```

实验现象：

1. 当网络正常时，数据强制走 ISP1，ping 100 的远程网络数据到 ISP1
2. 当 ISP1 宕机时，且 GW 连接 ISP1 的接口 DOWN 掉，则 PC 访问 100 网络的数据切换至 ISP2

3. 当 ISP1 宕机时，且 GW 连接 ISP1 的接口没 DOWN（如关闭 ISP1 的 CDP），由于 GW 丢失了 ISP1 的 CDP 信息，因此认为 ISP1 挂了，于是 PC 访问 100 网络的数据切换至 ISP2，网络不断

7.3.3.3 set ip next-hop verify-availability 基于 object tracking




```
ip sla monitor responder
ip sla monitor 1
 type echo protocol ipIcmpEcho 10.1.1.2 source-ipaddr 10.1.1.1
 frequency 10
exit
ip sla monitor schedule 1 life forever start-time now
track 1 rtr 1 reachability

ip sla monitor 2
 type echo protocol ipIcmpEcho 10.2.2.2 source-ipaddr 10.2.2.1
 frequency 10
exit
ip sla monitor schedule 2 life forever start-time now
track 2 rtr 2 reachability
```

```
access-list 1 permit any
route-map PBR permit 10
 match ip address 1
 set ip next-hop verify-availability 10.1.1.2 10 track 1
 set ip next-hop verify-availability 10.2.2.2 20 track 2
```

实验现象：

1. 当网络正常时，数据强制走 ISP1，ping 100 的远程网络数据到 ISP1
2. 当 ISP1 故障，GW 通过 tracking 感知到，于是数据切换至 ISP2
3. 回复 ISP1，GW 通过 tracking 感知到，数据又切换回 ISP1

技术解析：

1. **首先定义 track object，关联到一个 ip sla monitor**

   使用 ICMP 协议去探测 10.1.1.2 的可达性（使用源地址 10.1.1.1 去 ping10.1.1.2）

   Track object 的 ID 为 1，关联到 ip sla monitor 1

   当 10.1.1.2 可达，则 track 对象为 true

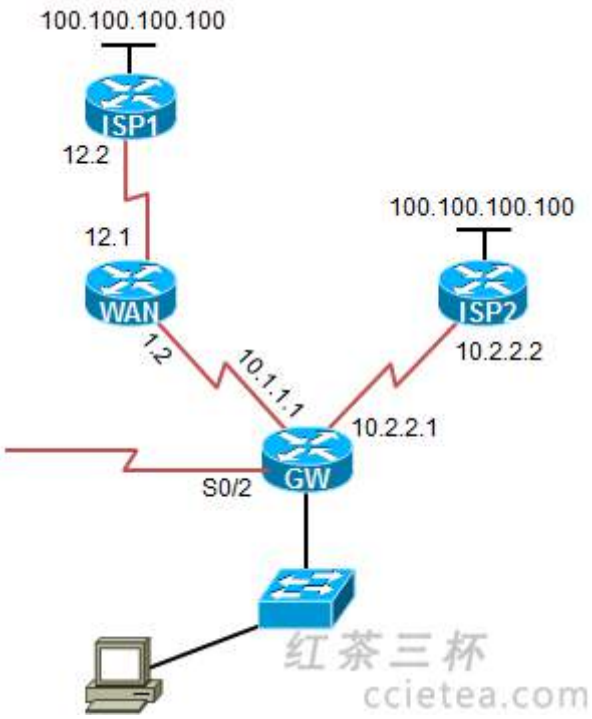
```
ip sla monitor 1
 type echo protocol ipIcmpEcho 10.1.1.2 source-ipaddr 10.1.1.1
 frequency 10
exit
ip sla monitor schedule 1 life forever start-time now
track 1 rtr 1 reachability
```

2. **然后在 route-map 中调用该 track object**

```
route-map PBR permit 10
 match ip address 1
 set ip next-hop verify-availability 10.1.1.2 10 track 1
```

   当 track1 为 true，也就是 10.1.1.2 可达，则 PC 访问 100 网络的数据被丢给 10.1.1.2，如果 track 1 挂了，则切换至下一个 next-hop

7.3.3.4 set ip next-hop recursive



GW 的配置如下：

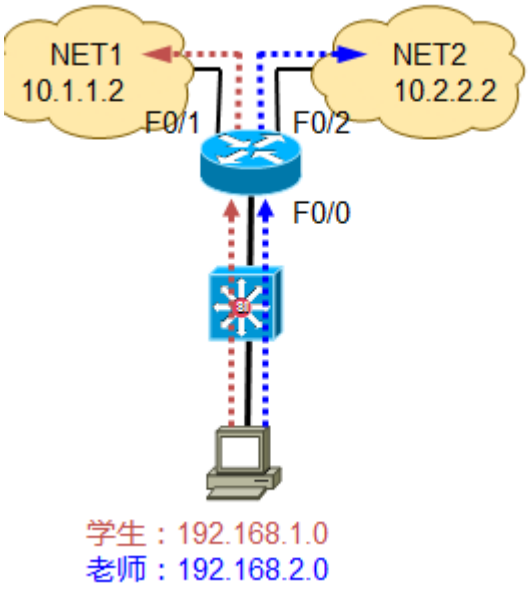
```
access-list 1 permit any
route-map PBR permit 10
 match ip address 1
 set ip next-hop 10.2.2.2
 set ip next-hop recursive 10.1.12.2

ip route 10.1.12.0 255.255.255.0 10.1.1.2
ip route 0.0.0.0 0.0.0.0 serial s0/2
```

实验现象：

1. 正常情况下，数据优先走 ip next-hop，也就是走 ISP2
2. 当 GW 连接 ISP2 的出接口 DOWN 掉（也就是 ISP2 挂了），则切换至 ip next-hop recursive,也就是 ISP2
3. 注意，这个时候是才用路由表递归找到去往 10.1.12.2 的路由的，因此路由表里必须有可达路由
4. 当 GW 丢失了去往 10.1.12.2 的路由，并且连接 ISP2 的连接也丢失了，则走默认路由

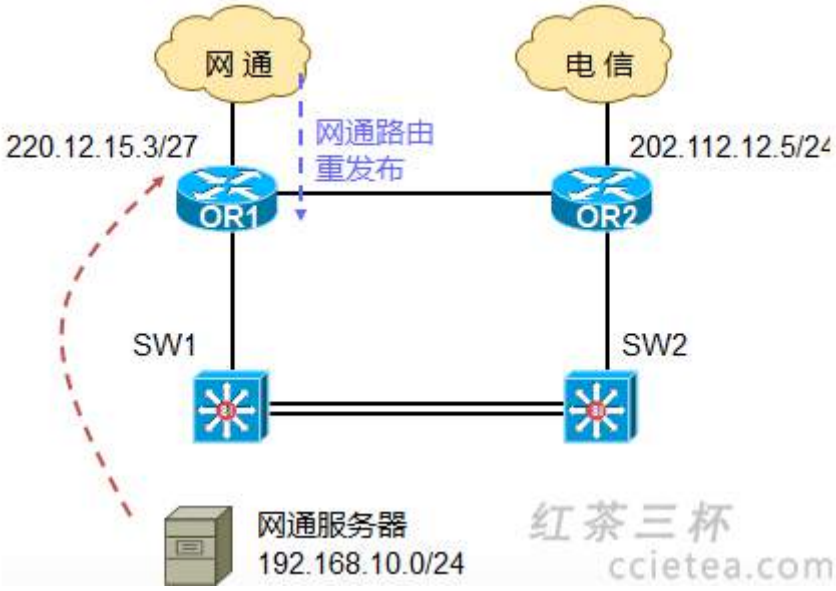
7.3.4 PBR 案例

1. **案例：通过 PBR 实现内网出口数据分流**




```
access-list 1 permit 192.168.1.0 0.0.0.255
access-list 2 permit 192.168.2.0 0.0.0.255

route-map test permit 10
match ip address 1
set ip next-hop  10.1.1.2

route-map test permit 40
match ip address 2
set ip next-hop  10.2.2.2

int f0/0
 ip policy route-map test

ip route 0.0.0.0 0.0.0.0 10.1.1.2
ip route 0.0.0.0 0.0.0.0 10.2.2.2 10
```

2. **案例：通过 PBR 规避教育、电信双出口 NAT 网络环境中存在的问题**



哥们先来介绍下上面的网络环境，这尼玛是一个相当经典的案例。OR1、OR2 是网络的出口路由器，分别连接到网通及电信的出口线路，两台出口路由器上都做了 PAT，使得内网用户能够访问外网。另外 OR1、OR2、SW1、SW2 跑 OSPF，OR2 作为电信的出口路由器，向 OSPF 域注入一条默认路由。OR1 则将本地

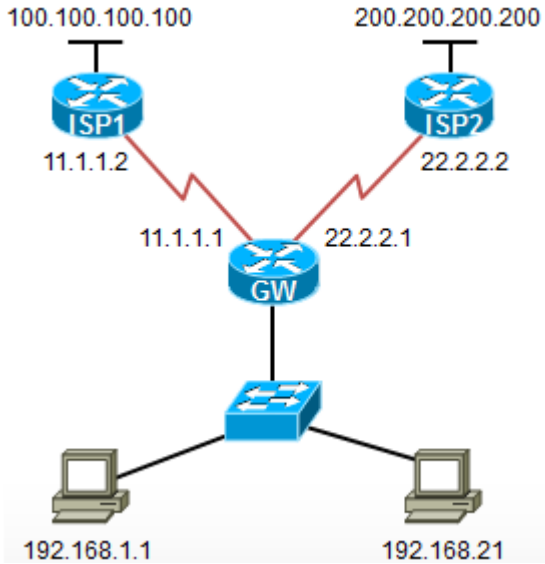
配置的静态网通路由重发布进 OSPF。这样一来，内网用户需访问网通资源时，由于有网通路由的明细，则走 OR1 出去。而访问其他资源，则走默认路由到 OR2 出去。

内网有一个网段，192.168.10.0/24，这是服务器网段，在 OR1 上部署了静态 NAT 映射，将特定的服务器映射到了 OR1 的出口上，这样一来，网通的用户能够从外网访问到这些服务器资源。但是问题来了，电信的外网用户，却无法通过映射出来的外网地址访问这些服务器资源，为什么？

仔细分析一下数据流的走向。当外网电信用户访问这些服务器资源的时候，数据的源地址是，电信公网 IP，目的地址是服务器映射出来的公网地址，于是数据从电信 WAN，绕到网通的 WAN，然后到达 OR1 的外网口，由于 OR1 上做了静态 NAT 映射，因此，数据包的目的 IP 被转换成服务器的内网 IP，数据包最终被送到了服务器。然后，服务器要回包吧？ 服务器的回包，源地址是服务器的内网 IP，目的地址是电信公网 IP，数据包被送到了三层交换机，三层交换机现在要做路由了，查路由表发现，只能走默认了，因此通过默认路由转发到 OR2，到了 OR2，它做了 PAT，因此将数据包的源地址，也就是服务器的内网 IP 转换成 OR2 的出口 IP，也就是一个电信的公网 IP，然后将数据包发回给那个电信的访问者，这哥们一收傻逼了，因为数据的源地址它压根不知道啊，它是访问的 OR1 上的地址啊。

这就是问题所在，怎么解决呢？很简单，在 OR2 上，部署 PBR，只要是来自网通的那些服务器数据，全部强制丢给 OR1，就解决了。

3. 案例：双出口 NAT



- 192.168.1.0用户访问外网强制走ISP1的线路，当ISP1线路DOWN掉，则自动切换至ISP2
- 192.168.2.0用户访问外网强制走ISP2的线路，当ISP2线路DOWN掉，则自动切换至ISP1
- 内网为私有IP，访问公网需使用公网地址

重点看 GW 的配置：

```
access-list 1 permit 192.168.1.0 0.0.0.255
access-list 2 permit 192.168.2.0 0.0.0.255
route-map PBR permit 10
  match ip address 1
```

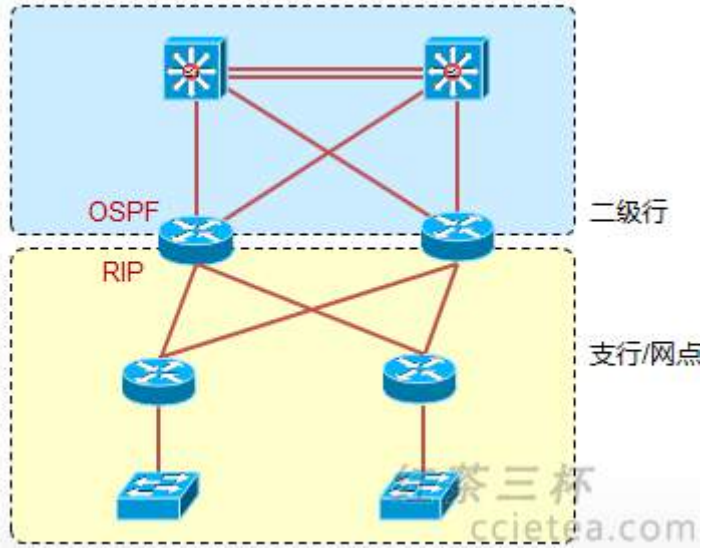
```
   set ip next-hop 11.1.1.2
exit
route-map PBR permit 20
   match ip address 2
   set ip next-hop 22.2.2.2
exit
```

上面的配置只是解决数据的走向问题，但是 NAT 的问题呢：

```
route-map nat1 permit 10
   match ip address 1
   match interface serial0/0                    !! 匹配数据包的出口
route-map nat2 permit 10
   match ip address 1
route-map nat3 permit 10
   match ip address 2
   match interface serial0/1
route-map nat4 permit 10
   match ip address 2
ip nat inside source route-map nat1 interface serial0/0 overload
ip nat inside source route-map nat2 interface serial0/1 overload
ip nat inside source route-map nat3 interface serial0/1 overload
ip nat inside source route-map nat4 interface serial0/0 overload
```

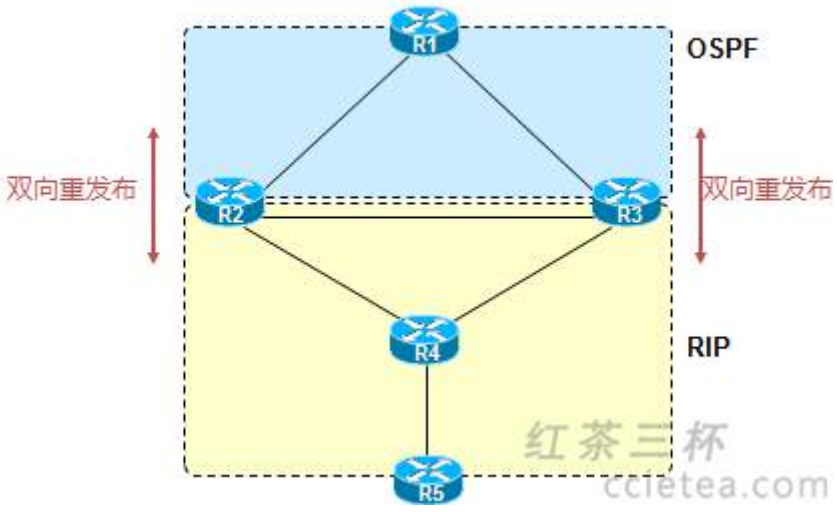
8 双点双向路由重发布

8.1 概述



“双点双向”路由重发布是一个在大型网络中时常能简单的模型。譬如金融网络中，二级行与网点路由器之间，如果运行动态路由协议，那么就有可能涉及到这个模型。所谓双点双向，指的是两个路由选择域的边界上有两个路由重发布的节点，并且重发布的方向是双向的。例如上图中，二级行有两台汇聚路由器，用于下面网点或支行路由器的接入。那么这两台路由器就是双点，另外，为了让全网的路由可达，因此在这两台路由器上部署双向路由重发布，也就是 OSPF 向 RIP 重发布，同时 RIP 也向 OSPF 来做重发布。

双点双向路由重发布的过程中，可能会发生许多问题，例如次优路径、潜在路由环路等等，本章对这个模型以及存在的问题将做深入探讨。



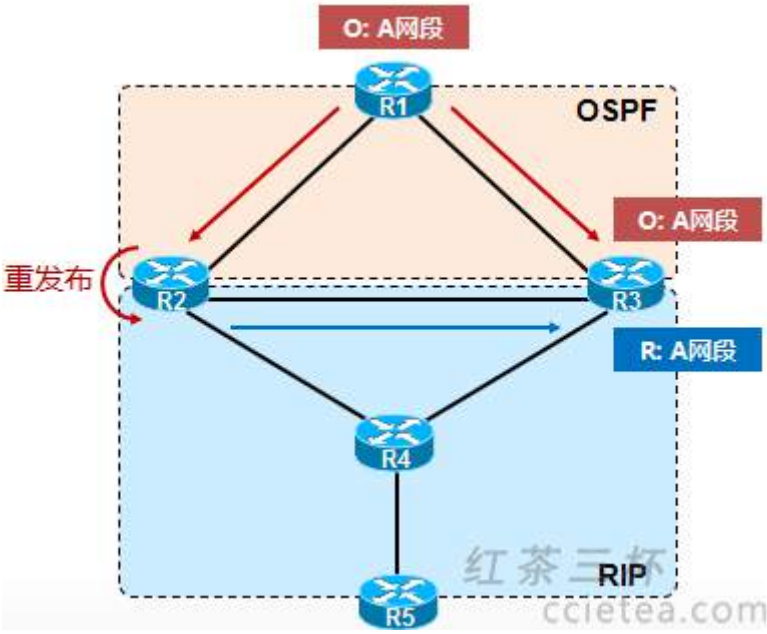
我们主要分析：

- RIP 与 OSPF
- OSPF 与 OSPF
- OSPF 与 BGP

这三个有代表性的路由重发布模型。

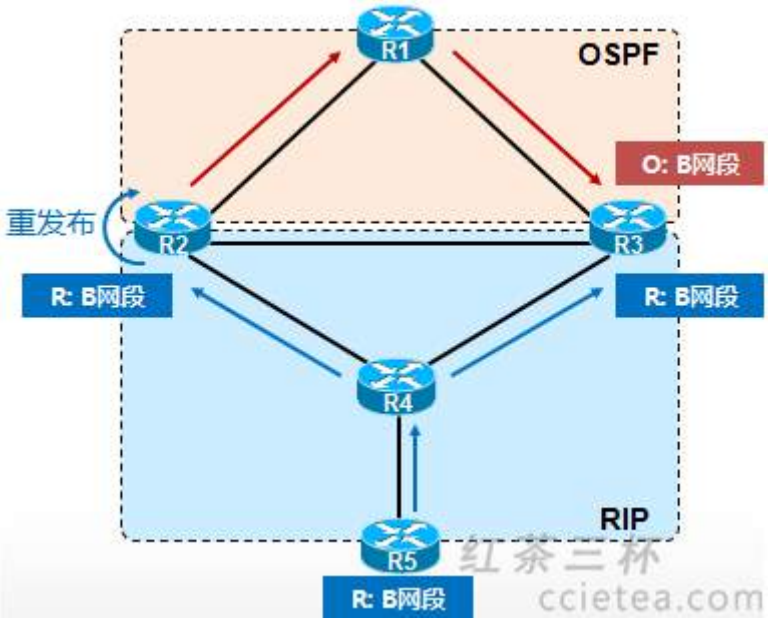
8.2 双点双向路由重发布存在的问题

1. **OSPF 与 RIP 的双向重发布**
   - **OSPF 向 RIP 重发布路由：**

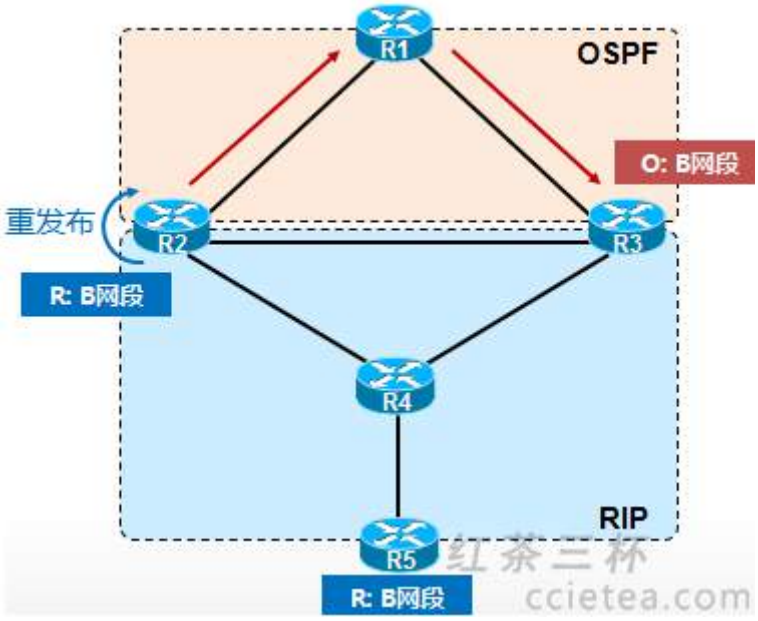


R1 发布 A 网段路由，R3 会学习到这条 OSPF 路由；另外，R2 也学到了，假设 R2 先部署 OSPF 到 RIP 的路由重发布，这条路由最终 R3 会通过 RIP 也学习到，那么 R3 将同时从 RIP 及 OSPF 学习到这条路由，R3 会优选 OSPF 路由，因为 OSPF 的 AD 小，所以 OSPF 向 RIP 重发布，不会造成次优路径问题。

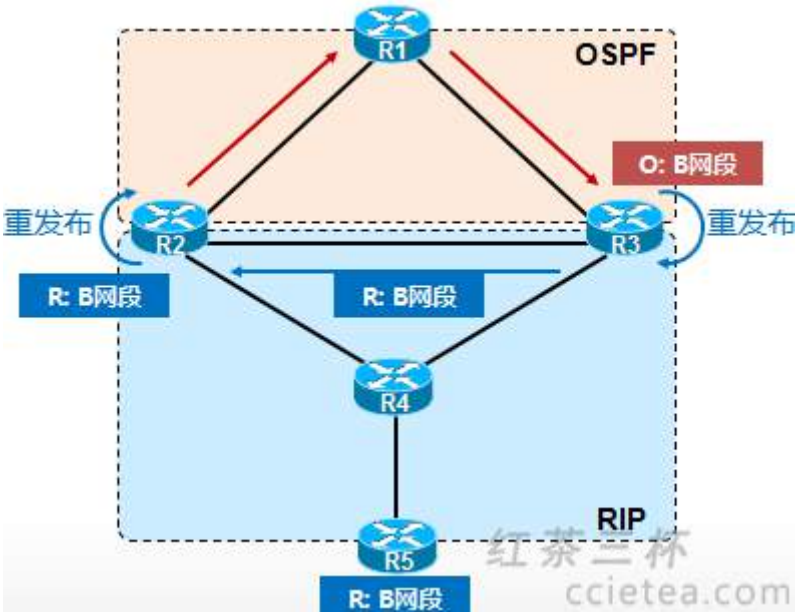
   - **RIP 向 OSPF 重发布路由：**



R5 发布 B 网段的 RIP 路由，R2 及 R3 都能学习到，此刻我们在 R2 及 R3 上都向 OSPF 来注入 RIP 路由，假设 R2 先注入的，那么 B 路由将被 R1 更新给 R3，R3 这时候同时从 RIP 及 OSPF 都学习到了 B 路由，那么它会：

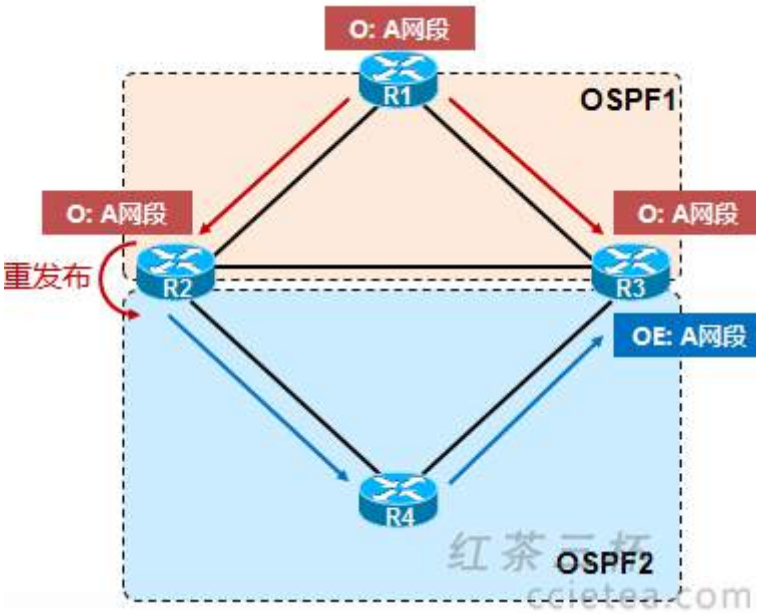


毫无疑问 ,R3 会优选 OSPF 的 B 路由 ,因为 OSPF AD 小 ,这样 R3 就形成了次优路径 ,它去往 B 网段，会选择 R1-R2 –R4-R5 这条路径。再者在 R3 的路由表中，关于 B 网段的路由是 O 的，那么即使 R3 做了从 RIP 到 OSPF 的重发布，重发布肯定是失败了的，因为前面我们说过了，路由协议重发布只会住入路由表里有的路由 ,而此刻路由表了关于 B 的路由是 OSPF 的 ,所以当然重发布失败。而且问题还没完，我们接下去看，如果此刻 R3 是双向重发布呢？也就是 R3 又部署了 OSPF 到 RIP 的重发布：



由 R3 路由表里是有 OSPF 的 B 网段路由，那么这条 B 网段路由会被重发布回 RIP 域，并且最终更新给 R4 和 R2，我们看 R2，R2 此刻是从 R4 已经学习到这条 RIP 路由，如果它又从 R3 也学到这条路由并且这条路由的 metric 更小的话，R2 就会优选来自 R3 的这条路由，这样一来，R1、R2、R3 就形成了一个路由环路。

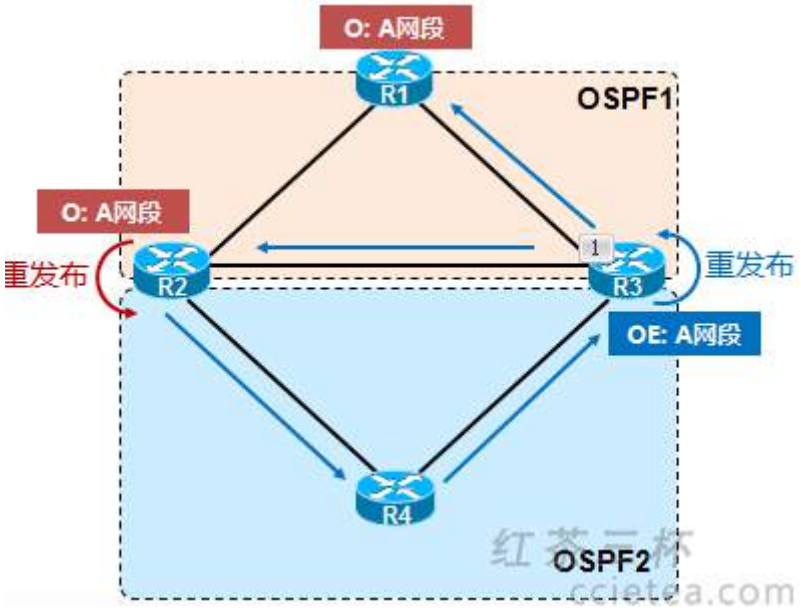
2. 不同的 OSPF 进程之间重发布路由



R2、R3 上，各有两个 OSPF 进程，如果所示，一个为 OSPF1，一个为 OSPF2，注意，其实这个双进程，主要是体现在 R2 和 R3 这两台 ASBR 上。

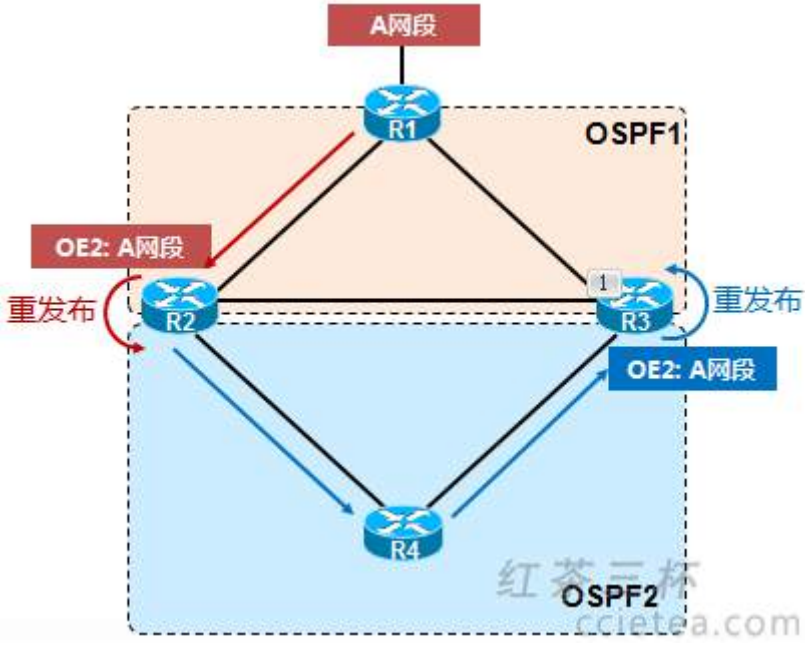
现在我们考虑当 R1 始发一条 OSPF 内部路由，R2、R3 都能学习到这条路由。那么在 R2 上，将路由从进程 1 重发布到进程 2 ,最终 R3 也会学习到这条从 R4 传递过来的路由。那么对于 R3 而言 ,双 OSPF 进程，

同时从两个进程都学习到同一条 OSPF 路由，采用先到先得的原则录用。所以，如果 R3 先学习到 R1 传来的路由，那么自然就会忽略 R2 重发布进来的那条。但是，如果 R1、R3 之间的邻居关系是在 R3 已经获得了 R2 重发布进来的这条 A 网段路由之后才起来的呢？那么 R3 将忽略 R1 更新来的路由，对于 R3 而言，去往 A 网段的下一跳就变成了 R4，这样一来就产生次优路径了。并且，R3 上，进程 1 向进程 2 的重发布也就失败了。



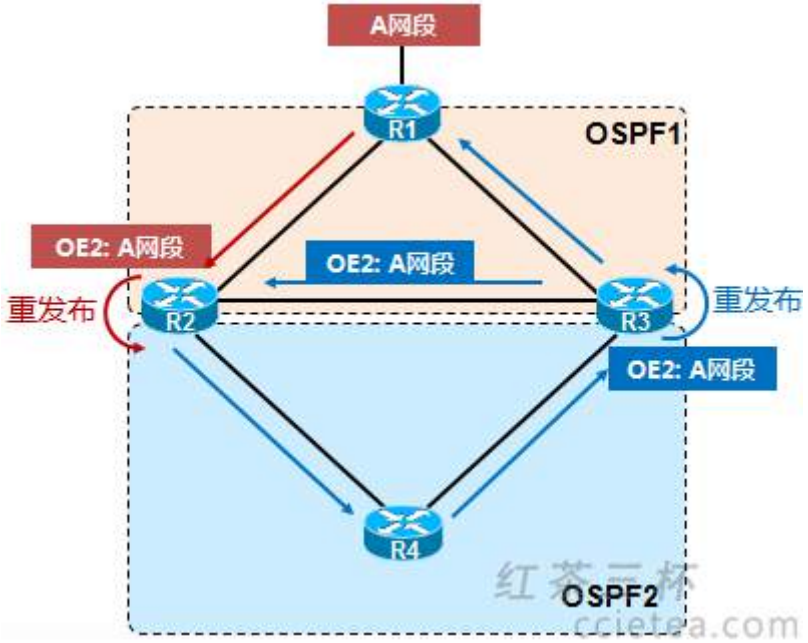
接着 ,R3 由于部署了双向重发布 ,因此会将进程 2 的路由注入进程 1 ,那么 A 网段的路由会被注入回进程 1，好在 O 的优先级大于 OE，因此 R1 及 R2 直接忽略这条路由。不会造成其他影响。

但是，如果 A 路由为 OSPF 外部路由呢？



假设现在环境是这样的，R1 重发布 A 路由进 OSPF，那么 R2 会在自己的 OSPF 进程 1 学习到这条 OE2 的

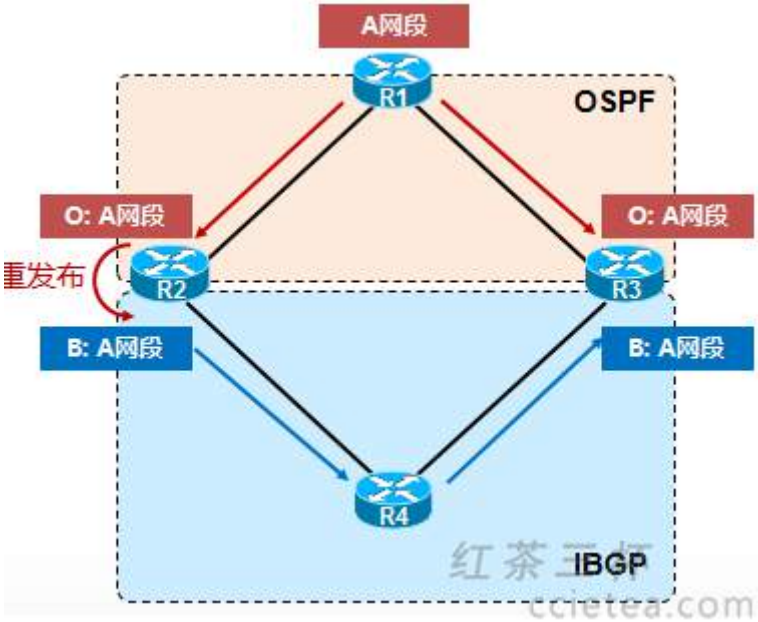
路由，并且将路由重发布进自己的 OSPF 进程 2，那么 R3 会从进程 2 学到这条路由，假设 R3 先学到这条 R2 重发布的 A 路由，那么根据先入为主的规则，R3 将忽略 R1 后来的关于 A 路由的更新。与此同时，由于部署了双向重发布，R3 又会进一步的将学习到的 A 路由重发布回进程 1。



对于 R2，它此刻是在自己的进程 1 中，同时从 R1 和 R3 学习到这条 OE2 的路由，它会如何选择？回顾一下 OE2 的比较原则，先比 OE2 的外部 metric，如果相等，进一步比较内部 metric 也就是到达 ASBR 的 metric（当然，是 FA=0.0.0.0 的情况下），那么，在这个图中，如果 R2 到 ASBR-R3 的开销更小，最终 R2 将优选 R3 重发布进来的 A 路由，于是乎，R2、R3、R4 就构成了路由环路。

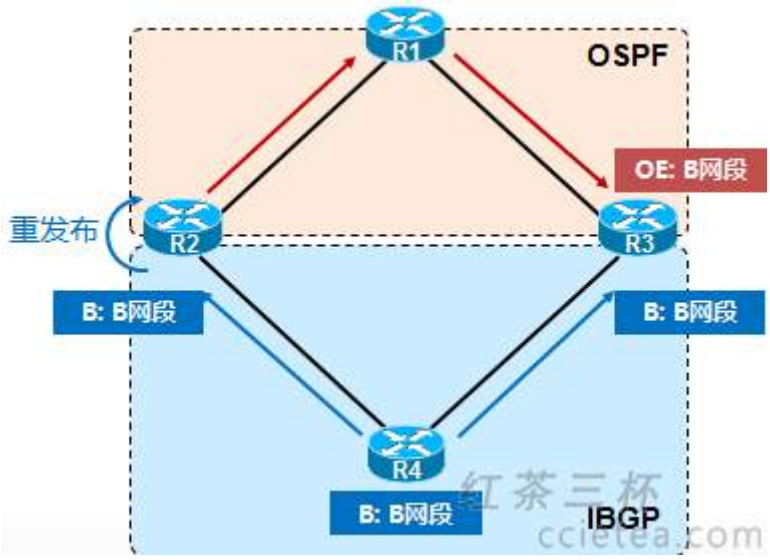
3. OSPF 与 IBGP 的路由重发布

- **OSPF 向 IBGP 重发布路由**



当 OSPF 向 IBGP 重发布路由时，由于 OSPF 的 AD 比 IBGP 的 AD 要小，因此不会出现次优路径或是路由环路的问题：

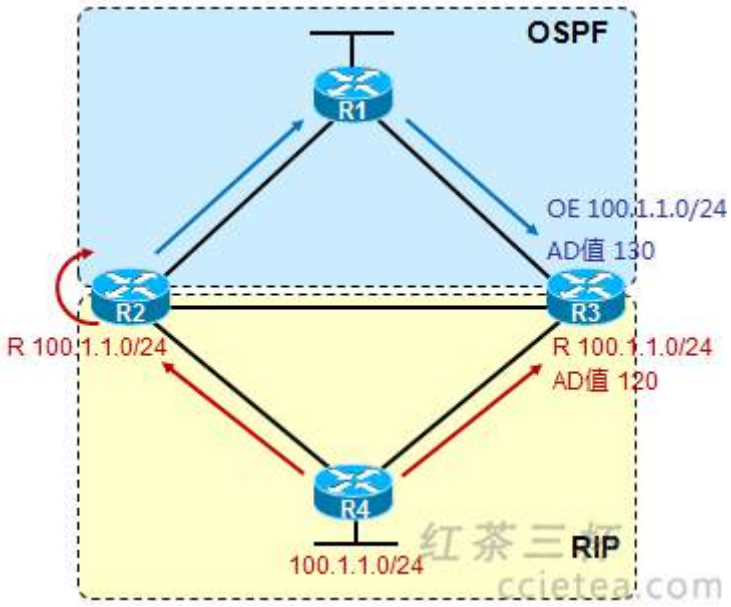
- **IBGP 向 OSPF 重发布路由**



问题和前面说到的类似，R2 及 R3 都能学习到更新自 R4 的 BGP B 网段路由。假设 R2 先做的重发布，那么 B 路由会注入到 OSPF 中，最终被 R3 学习到，那么 R3 上，同时从 OSPF 及 IBGP 学习到这条路由，会优选 OSPF，因为 OSPF 的 AD 要小。这样一来，R3 上，IBGP 向 OSPF 的重发布就失败了，因为 B 网段的路由在路由表中并不是 BGP 的。再者，R3 上做了双向重发布，因此 B 网段的路由，又会被重发布回 IBGP，自此，就有可能造成次优路径、环路等一系列问题。

8.3 双点双向路由重发布的实现

8.3.1 解决方案：修改路由协议管理距离



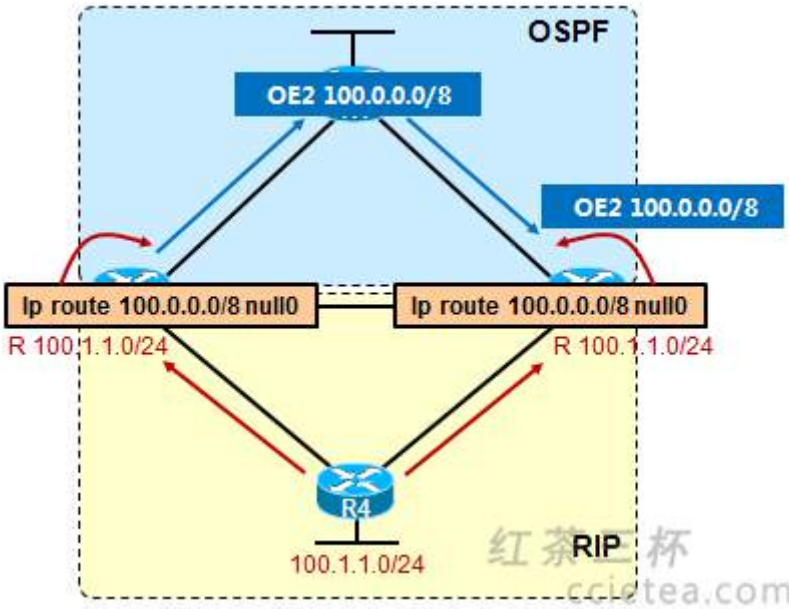
前面分析得很透彻了，当 RIP 向 OSPF 做双点重发布的时候，由于路由被注入到 OSPF 后，AD 就变成了 110，那么路由传递到 R2、R3 之后，由于 OSPF AD 比 RIP 要小，因此 OSPF 路由会覆盖 RIP 的由此造成次优路径。那么，我们可以在 R2、R3 上，用 ACL 等工具去抓取 RIP 域中的路由，然后在 OSPF 进程中，将这些路由的 AD 值调得比 RIP 要大，那么这样一来，这些路由就不会将 RIP 路由覆盖，次优路径问题也就可以规避。

举例 R3 的配置如下：

```
access-list 1 permit 100.1.1.0
router ospf 1
    distance 130   2.2.2.2 0.0.0.0 1          !!其中 2.2.2.2 为 R2 的 router-ID ，注意由于 100.1.1.0 在 R3 的 OSPF 进程学习到的是外部路由，因此这里的更新源是 R2。
```

8.3.2 解决方案：采用重发布静态汇总路由的方式规避次优路由等问题

- **RIP 与 OSPF 的重发布**



R4 发布 100.1.1.0/24 的 RIP 路由，两台 ASBR 都能够学习到。

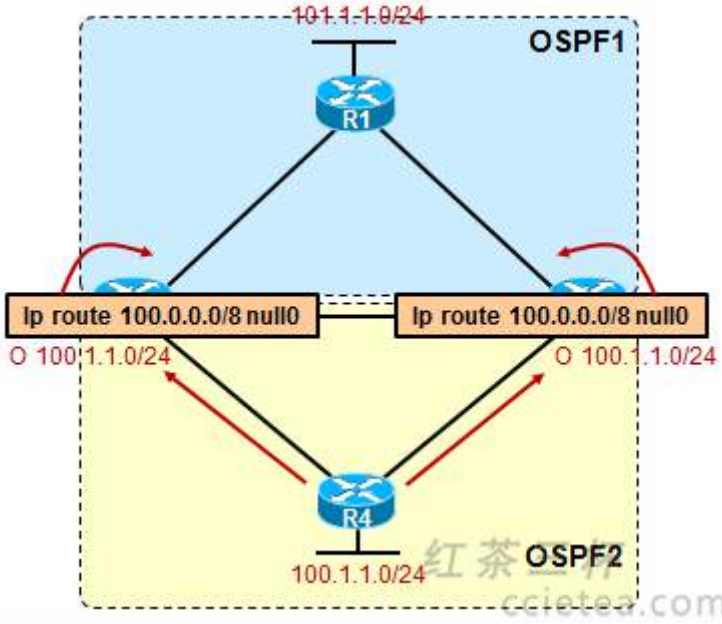
解决的办法是，在两台 ASBR 上配置静态汇总路由：

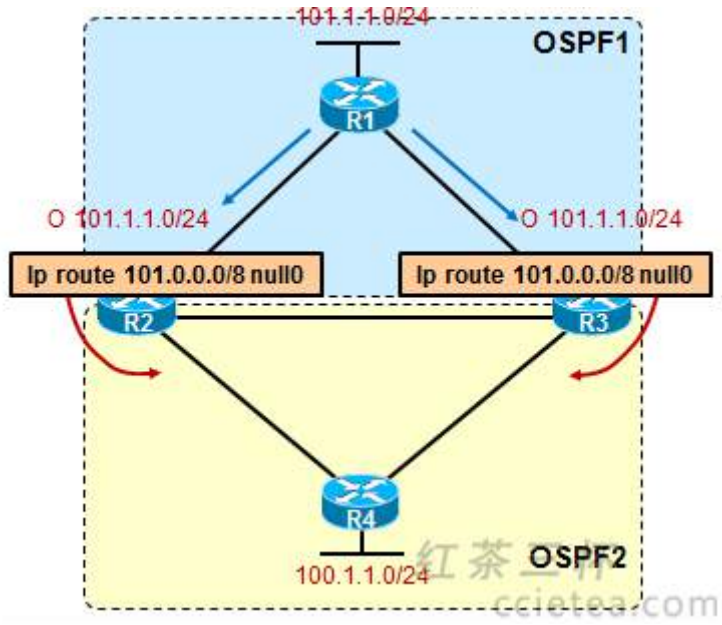
```
Ip route 100.0.0.0 255.0.0.0 null0
```

然后仅仅将这条静态汇总路由重发布进 OSPF。这样一来 OSPF 内的路由器都能通过这条汇总路由到达 100.1.1.0/24，同时，这条重发布后变成 OSPF 的汇总路由，就算传回了两台 ASBR，由于本地有静态的汇总路由，因此直接忽略 OSPF 的汇总路由，而不会产生次优路径，这条路由也不会被重发布回 RIP 而导致其他的什么问题。

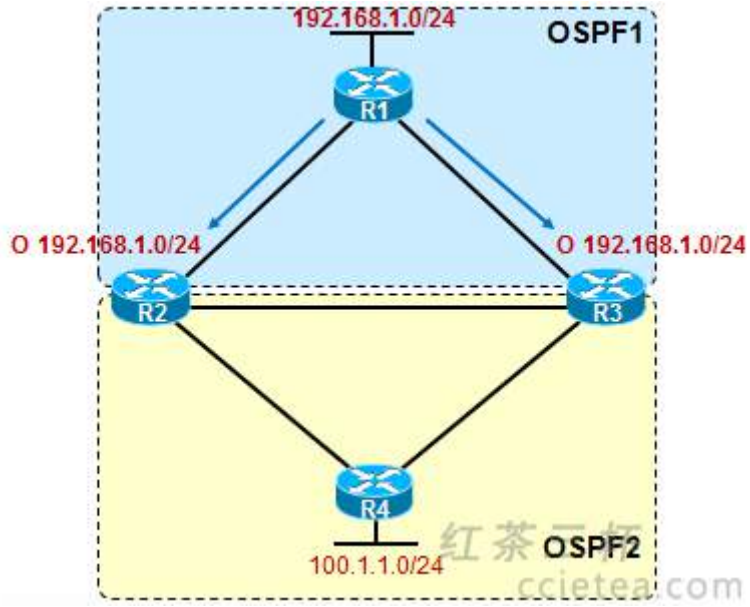
- **不同的 OSPF 进程之间的重发布**

不同的 OSPF 进程之间双点双向重发布，同样可以使用静态汇总路由重发布的方式，分别在两台 ASBR 上面创建不同的静态汇总路由，在相应的进程下进行重发布静态汇总路由。





那么，如果某个 OSPF 进程中（这句话可能不大严谨，大家理解我的意思就成），路由无法汇总咋办，例如下图。



如果某个 OSPF 进程内路由无法做汇总那么：

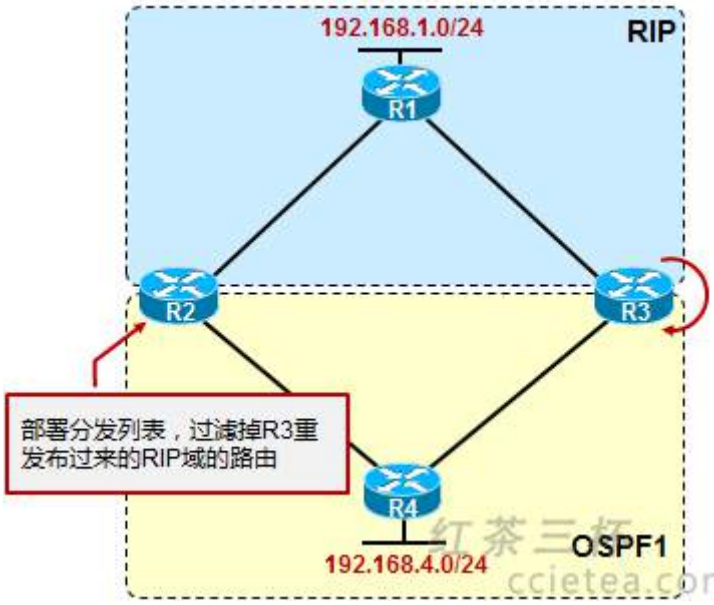
为 OSPF2 内的路由创建静态汇总路由（Null0），并将静态路由重发布进 OSPF1 中

修改 OSPF1 的管理距离，使 OSPF1 的优先级高于 OSPF2 的管理距离

以 R2 的配置为例：

```
ip route 100.1.0.0 255.255.0.0 null0
!
access-list 1 permit 192.168.1.0
!
router ospf1
    redistribute static subnets
    distance 100 192.168.12.1 0.0.0.0 1
```

8.3.3 解决方案：使用分发列表规避次优路径问题

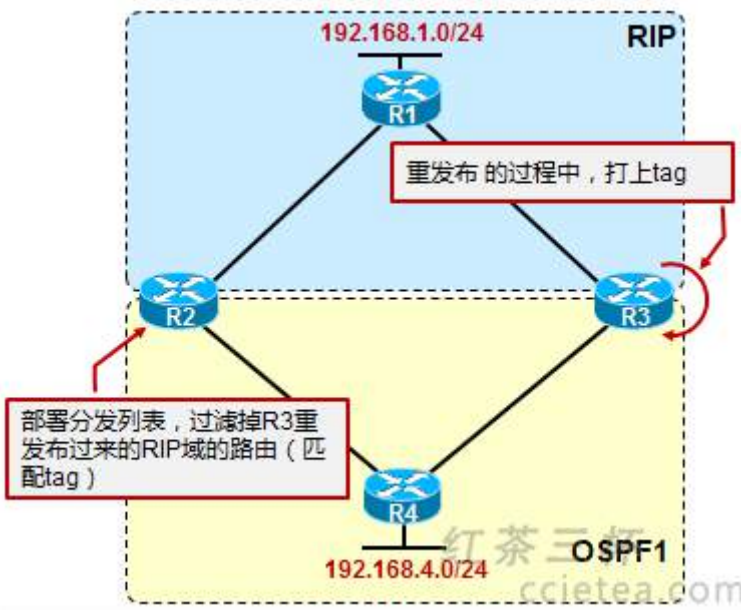


一个非常简单的方法，假设在 R3 上，先做的 RIP 到 OSPF 的路由重发布，前面已经说过了，一个直接的问题是在 R2 上，由于 192.168.1.0/24 这条路由，同时从 OSPF 及 RIP 学习到，而 OSPF 的 AD 值较小，从而导致次优路径的产生，因此为了规避这个问题，我们可以直接在 R2 上部署分发列表，使得 OSPF 路由不加载进路由表，从而规避次优路径问题。

在 R2 上配置如下：

```
access-list 1 deny 192.168.1.0
access-list 1 deny 192.168.13.0                    !!这是 R1-R3 之间的直连链路
access-list 1 permit any
!
router ospf 1
  distribute-list 1 in
```

当然 R3 上也要做类似的配置。



上面的规避方案，我们是直接用 ACL 来匹配路由，再在分发列表里进行调用，这种方法可扩展性不高，如果重发布进 OSPF 的路由条目太多，ACL 条目就会写疯。因此我们尝试用一种更具有扩展性的方法，举例来说，在 R3 上，将 RIP 重发布进 OSPF 的时候，将注入的路由打上 tag，然后呢在 R2 上，部署分发列表的时候，就可以关联一个 route-map 来“抓取”这些带了 tag 的路由。

R3 的配置如下：

```
router ospf 1
  redistribute rip subnets tag 1111
```

完成这个配置后，我们可以 show 一下：

R4#sh ip ro 192.168.1.0

```
Routing entry for 192.168.1.0/24
  Known via "ospf 1", distance 110, metric 20
  Tag 1111, type extern 2, forward metric 64
  Last update from 192.168.34.3 on Serial0/1, 00:09:32 ago
  Routing Descriptor Blocks:
  * 192.168.34.3, from 192.168.3.1, 00:09:32 ago, via Serial0/1
      Route metric is 20, traffic share count is 1
      Route tag 1111
```

由于我们在 R3 上，将 RIP 路由注入 OSPF 后打上了 tag 1111，那么接下去

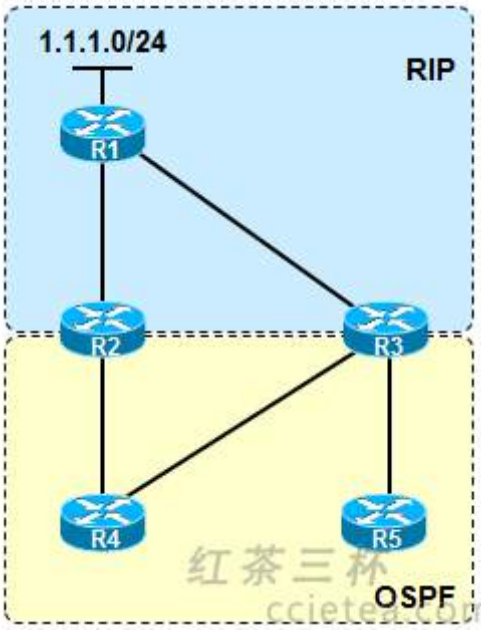
在 R2 上：

```
route-map test deny 10
 match tag 1111
!
```

```
route-map test permit 20
router ospf 1
 network 192.168.24.2 0.0.0.0 area 0
 distribute-list route-map test in
```

8.3.4 思考题

这个综合思考题帮助大家梳理一下几种解决方案：

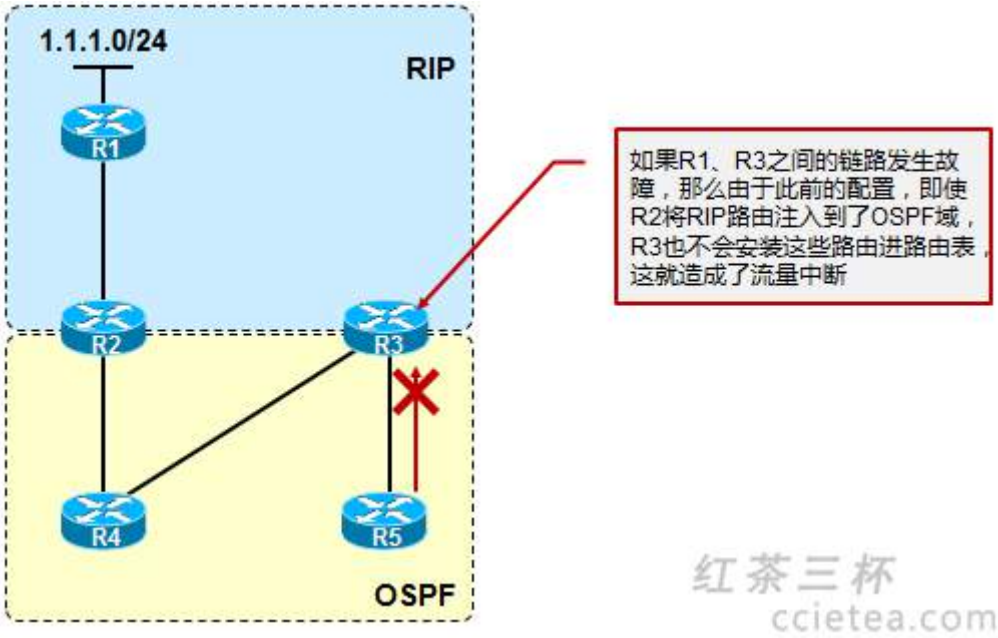


我们看上图，在 R2、R3 上部署了双向重发布。

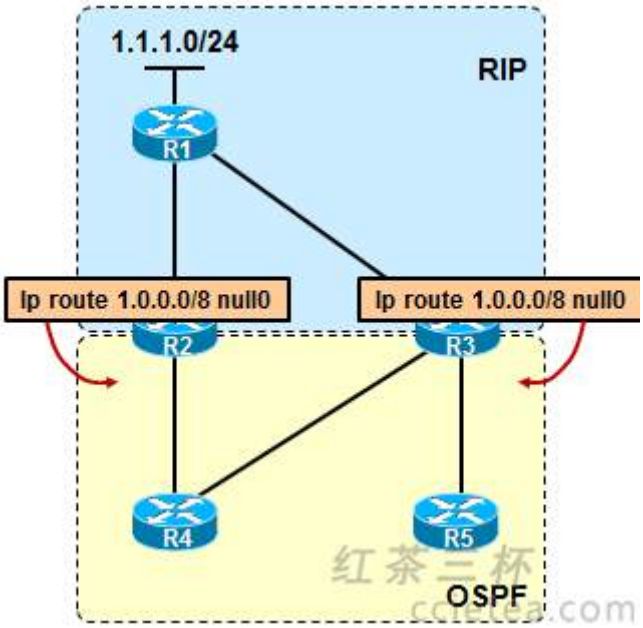
首先为了规避次优路径问题，我们在 R2 及 R3 上，使用 in 方向的分发列表，将 1.1.1.0 过滤掉。

```
access-list 1 permit 1.1.1.0
router ospf 1
  distribute-list 1 in
```

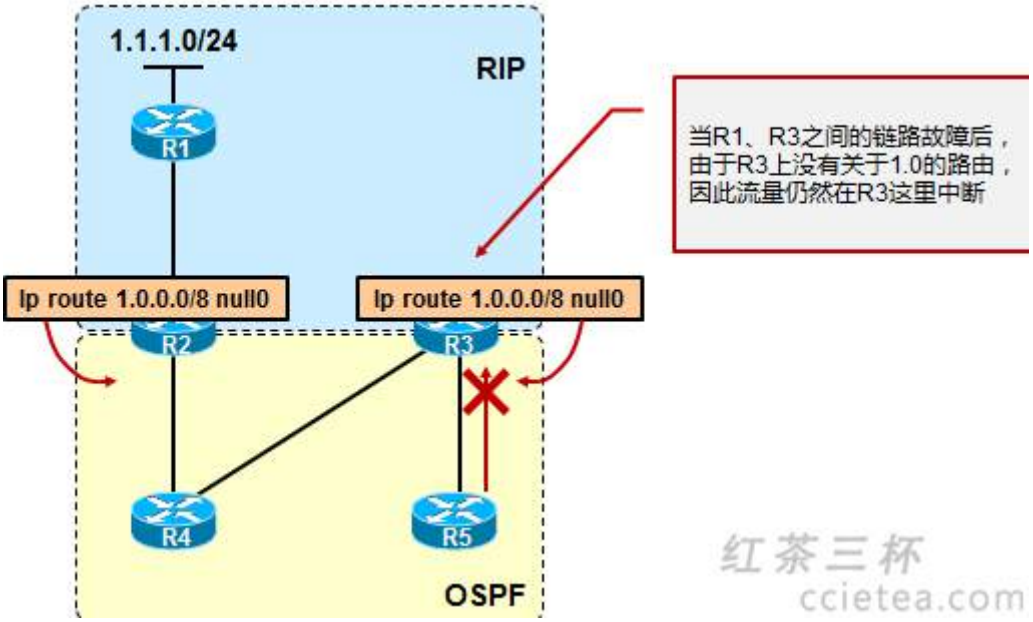
这样的确可以规避次优路径问题，但是，却留下了一个隐患：



那么如果我们用重发布静态汇总路由的解决方案呢：

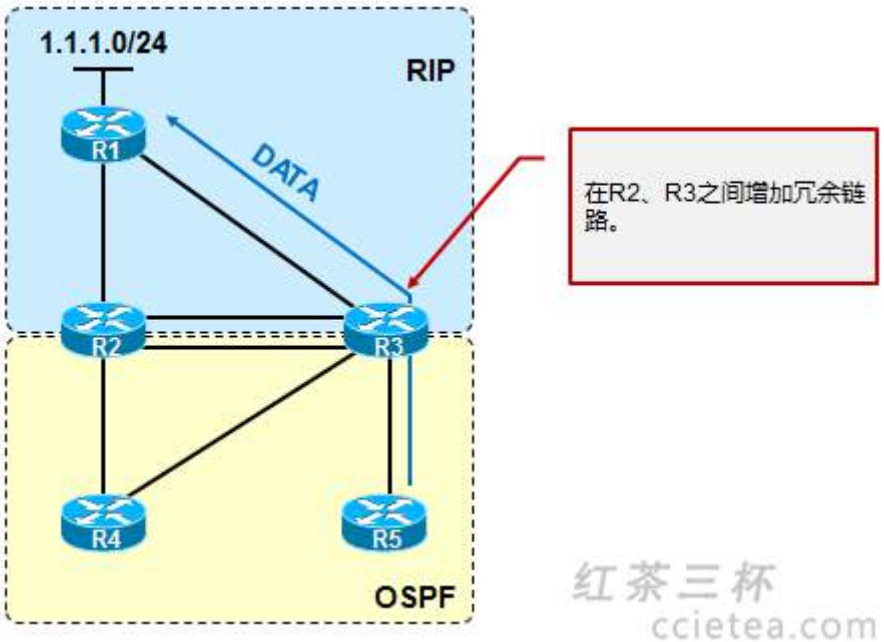


R2、R3 上，针对 RIP 域内的路由创建静态汇总路由，指向 null0，同时只将这些静态路由重发布进 OSPF，也确实可以起到规避次优路径的问题，但是：

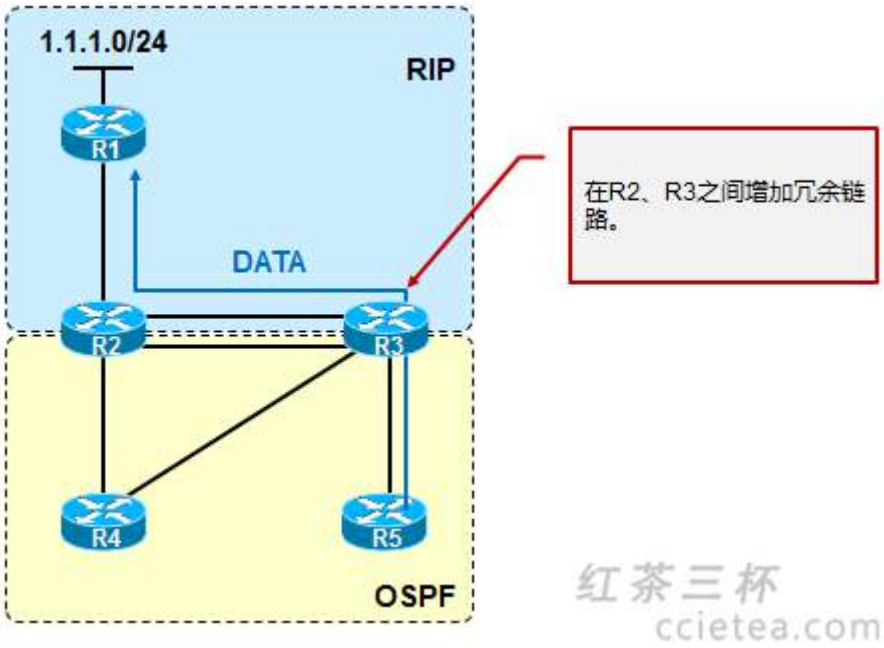


当 R1、R3 之间链路发生故障时，同样网络还是会出问题。因为虽然 R2 将本地配置的静态汇总路由重发布进了 OSPF，但是 R3 是忽略这条路由的（本地配置了静态的指向 null0 的汇总路由），虽然 LSA 还是被传递给了 R5，也就是说虽然 R5 还是有 1.0.0.0/8 的路由 ,但是数据包丢给 R3 后 ,最终在 R3 这里被丢弃 ,因此这里还是有问题。

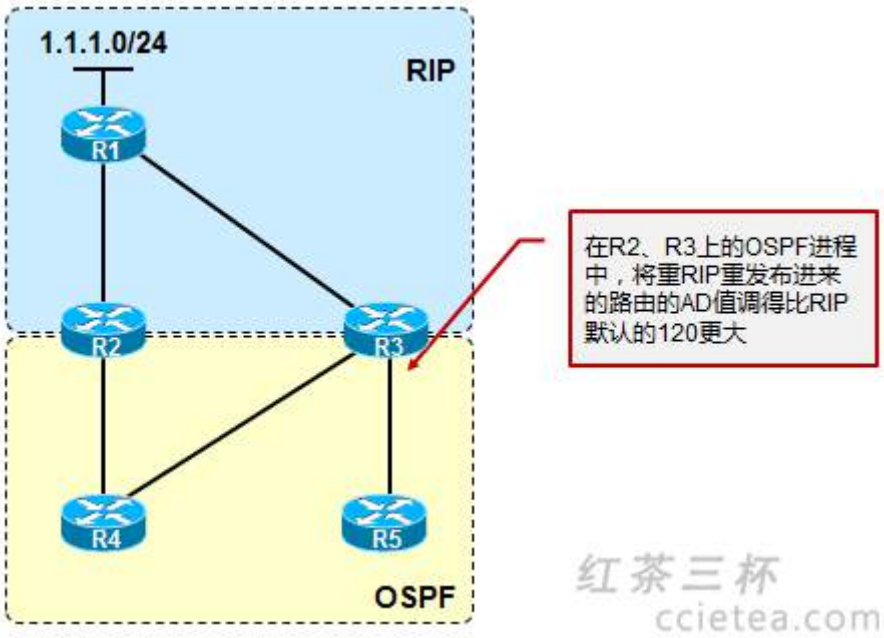
比较理想的解决方案之一是，改造网络拓扑：



这样一来，无论是采用重发布静态汇总路由的方式，还是分发列表过滤路由的方式，都可以在网络出现故障的情况下，保证网络不中断，如下：



另一个推荐的解决办法是，调整路由协议的管理距离：



9 缺省路由

9.1 ip default-gateway

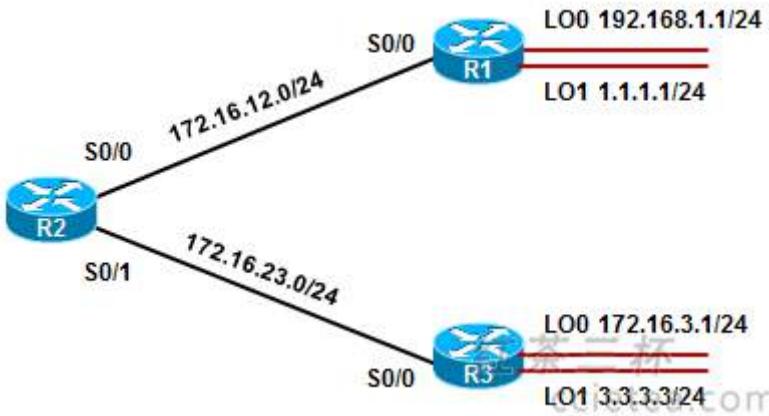
仅在路由器（或三层交换机）关闭路由功能的时候（no ip routing）有效；

如果开启了 ip routing，则忽略该命令（而是看默认路由 Ip route 0.0.0.0 0.0.0.0 x）;

路由器在 boot 模式下，做软件升级的时候，由于 ip routing 也是关闭的，因此在必要时该命令也会用到。

9.2 ip default-network

当使用 Ip default-network 在本地指一个网络时，这个网络号如果在路由表中存在，那么路由器会将该网络指定为缺省网关。



- **实验 1**

  完成基本的接口 IP 配置，R2 上增加如下配置：

```
ip route 192.168.1.0 255.255.255.0 172.16.12.1        // 路由表中有了标记为 S 条目
Ip default-network 192.168.1.0                        // 将路由表中 192.168.1.0 标记为缺省网络
```

  完成配置后，R2 show ip route

```
Gateway of last resort is 172.16.12.1 to network 192.168.1.0
S*     192.168.1.0/24 [1/0] via 172.16.12.1
```

  这时 R2ping 1.1.1.1 就能通了（相当于将 172.16.12.1 作为最后一跳求助对象）

- **实验 2**

完成基本的接口 IP 配置，R2 上增加如下配置：

```
ip route 172.16.3.0 255.255.255.0 172.16.12.3
ip default-network 172.16.3.0
```

完成配置后，R2 show ip route

```
S    172.16.0.0/16 [1/0] via 172.16.3.0                    // 出来一条汇总路由，而不是缺省路由
S    172.16.3.0/24 [1/0] via 172.16.23.3
```

R2 show run 后发现：

```
ip default-network 172.16.3.0  变成了：ip route 172.16.0.0 255.255.0.0 172.16.3.0
```

这是因为 ip default-network 是有类的，因此如果使用该命令标记一个主类网络的某个子网，实际上路由器会安装该子网对应的主类网络路由进路由表而不会产生任何缺省路由。所以这时候就在上面的基础上，由于产生了 172.16.0.0 的路由，因此可以再使用一次 ip default-network 命令

```
ip default-network 172.16.0.0
这样一来路由表：
Gateway of last resort is 172.16.3.0 to network 172.16.0.0
*     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
S*    172.16.0.0/16 [1/0] via 172.16.3.0
S     172.16.3.0/24 [1/0] via 172.16.23.3
```

如此 R2 就能 ping 通 3.3.3.3

- **实验 3**

前面是使用静态路由，如果使用动态路由协议，那么情况就不太一样了，例如 IGRP 或 EIGRP，ip default-network 命令指定的网络就必须是通过 IGRP 或 EIGRP 获取到的（宣告、学习或重发布），该条缺省路由才能被动态的传递给其他协议路由器。

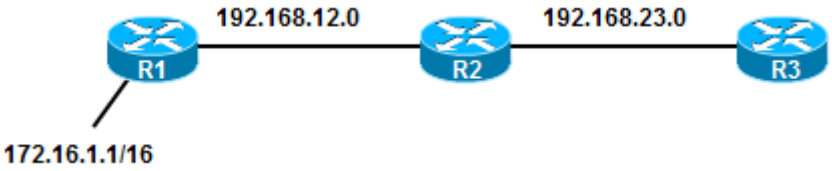
如果是 RIP，ip default-network 命令指定的网络则无需被显示宣告进 RIP，且一旦缺省网关被指定，RIP 会发布一条默认路由进 RIP，传递给其他路由器。例如 R1、R2、R3 运行 RIP，在 R1 上开 loopback 192.168.1.0/24，RIP 只宣告直连链路（而无需在进程中宣告这个 Loopback），在 R1 上 ip defaut-network 192.168.1.0，则会传递 0.0.0.0 的默认路由进 RIP，R2R3 都会学习到。

总结

如果使用 ip default-network 指定了多个候选缺省路由，那么拥有最低 AD 的将会成为缺省路由，并且设定为缺省网关（gateway of last resort），如果 AD 都相等，那么 show ip route 第一个显示的，就作为缺省网关。如果同时使用 ip defaut-network 及 ip route 0.0.0.0 0.0.0.0，且 ip defaut-network 指定的网络为静态路由配置的，那么 ip

defaut-network 的优先，并且成为缺省网关 gateway of last resort。但如果 ip defaut-network 指定的网络是学习自动态路由协议，则 ip route 0/0 的优先。

9.3 RIP 重发布默认路由



- **实验 1**

  R1、R2、R3 运行 RIP，只宣告三者之间的直连链路。在 R1 上 ip default-network 172.16.0.0

  则 R2 的路由表如下：

```
Gateway of last resort is 192.168.12.1 to network 0.0.0.0
C       192.168.12.0/24 is directly connected, Serial0/0
C       192.168.23.0/24 is directly connected, Serial0/1
R*      0.0.0.0/0 [120/1] via 192.168.12.1, 00:00:03, Serial0/0
```

- **实验 2**

  在 R1 上 ip route 0.0.0.0 0.0.0.0 null 0　　然后路由进程里重发布静态，也成

- **实验 3**

  在 R1 上，ip route 1.0.0.0 255.0.0.0 null 0　然后 ip defaut-network 1.0.0.0，则 R2R3 会学习到缺省路由

- **实验 4**

  在 R1 上，default information originate　　也成，R2 及 R3 会学习到 0/0 的默认路由。 这条命令对于 RIP 而言，不要求 R1 自己本地有默认路由。

10 参考书目

TCP/IP 卷一

IP 路由疑难解析

IPv6 技术笔记

红茶三杯 CCIE 学习文档

文档版本： 3.0 更新时间： 2013-07-02

文档作者： 红茶三杯 文档地址： http://ccietea.com

文档备注： 访问 ccietea.com 以获得最新的文档版本。文档的编撰倾注了作者大量的心血、时间和精力，供广大技术爱好者交流分享，请勿用于商业用途，转载请保留作者信息。

1 IPv6 概述

1. IPv6 特点

- 128bits 的地址方案，为未来数十年提供了巨大的 IP 地址空间
- 多等级层次有助于路由聚合，提高了因特网网络路由的效率及可扩展性
- 自动配置过程允许 IPv6 网络中的节点更加便捷的接入 IPv6 网络
- 重新编址机制使得 IPv6 提供商之间的转换对最终用户是透明的
- 无需 NAT
- 不再有广播、不再有 ARP
- IPv6 的包头比 IPv6 更有效率，数据字段更少，去掉了包头校验和。更简单的报头提高了路由器的处理效率。新的扩展包头替代了 IPv4 的选项字段，并且提供了更多的灵活性
- 更有效的支持移动性和安全性
- v4v6 过渡方式丰富多彩

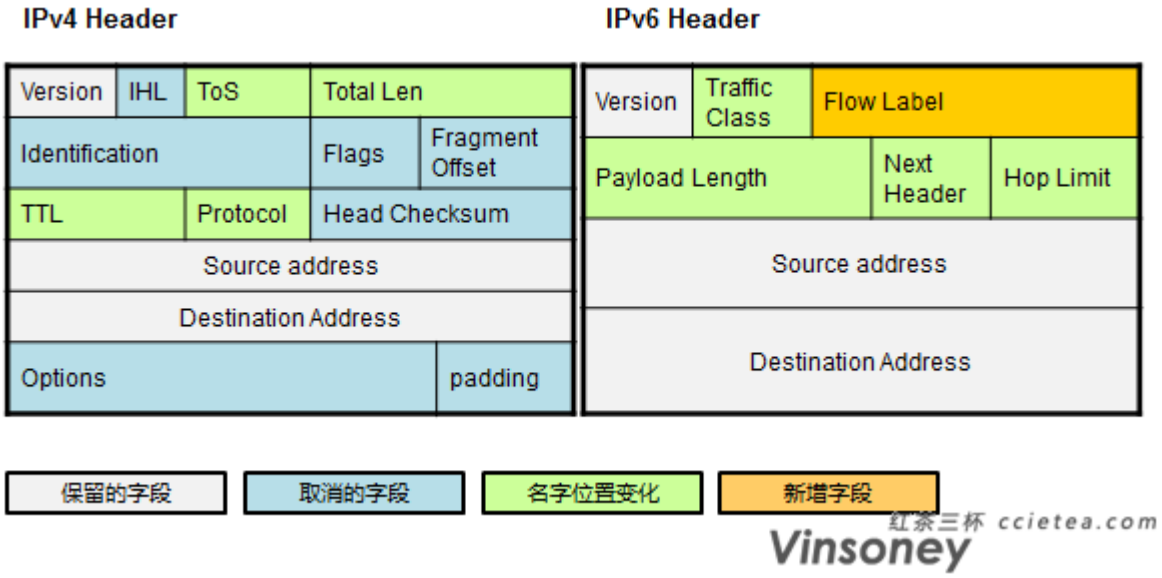
2. 6bone

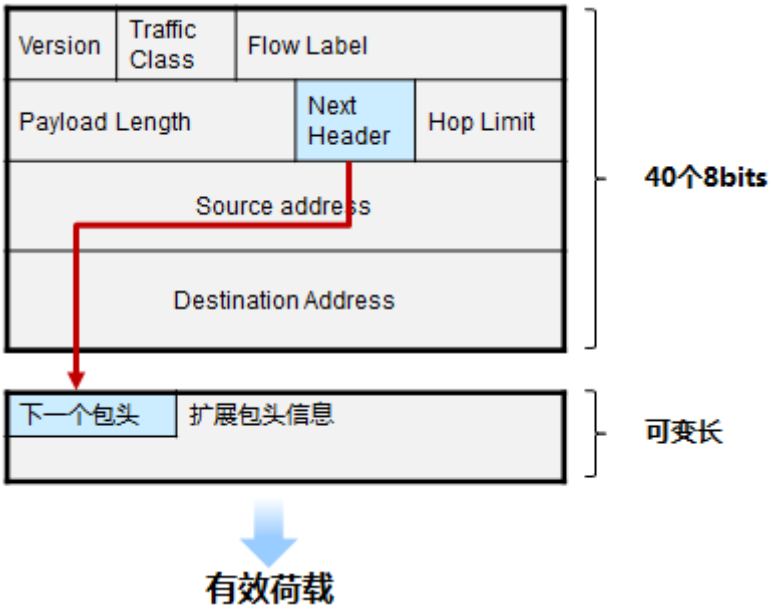
6bone 是一个世界范围的非正式合作的 IPv6 试验床。也是 IETF IPng 项目的成果。始于 1996 年，是运行在因特网上的由 IPv6 的 IPv4 隧道组成的虚拟网络，网络在缓慢的向纯 IPv6 链路迁移。更多的消息，请关注

2 IPv6 Header

2.1 IPv6 Header

1. **包头格式**





- **基本的 IPv4 报头包含 12 的字段，20 个字节长。options 和 padding 字段在需要时添加。**

  IHL 包头长度；

  Total Length 总长度；

- **基本的 IPv6 报头 40 个 8 位 bit，即 40 个字节长，共 8 个字段。**

  IPv6 数据包由一个基本报头加上 0 个或多个扩展报头再加上上层协议单元构成。

  几个字段的含义：

  下一个报头(Next Header)： 该字段定义了紧跟在 IPv6 报头后面的第一个扩展报头(如果存在)的类型，或者上层协议数据单元中的协议类型。

  跳限制(Hop Limit)： 类似于 IPv4 中的 TTL 字段。它定义了 IP 数据包所能经过的最大跳数。每经过一个路由器，该数值减去 1，当该字段的值为 0 时，数据包将被丢弃

  FlowLabel： **流标签，用于 QoS**

2. IPV6 包头的改进

- 取消了 IP 的校验

  第二层和第四层的校验已经足够健壮了，因此 IPv6 直接取消了 IP 的三层校验。

- 取消中间节点的分片功能

  分片重组功能由源目端自己进行，通过 PMTU 机制来发现路径 MTU

- 定义最长的 IPv6 报头

  有利于硬件的快速处理，如此一来中间节点可以避免处理而节省大量的资源

- 安全选项的支持

  IPv6 提供了对 Ipsec 的完美支持，如此上层协议可以省去许多安全选项，例如 OSPFv3 就取消了认证

- 增加流标签

  提高 QoS 效率

3. 扩展报头

在 IPv6 中，v4 中的选项被移到了扩展报头中。一个 IPv6 数据包可能包括 0 个或多个扩展包头，当使用多个扩展包头时，通过前面的包头的 Nexthead 字段指明该扩展包头后的扩展包头。有了扩展包头，中间路由器就不需要处理每一个可能出现的选项，提高了路由器处理数据包的速度，提高了其转发性能。在扩展报头链的最后就是有效负载。

扩展报头可选，只有需要该扩展报头对应的功能，发送主机才会添加相应扩展报头

- **逐跳选项报头 Hop-by—Hop Options Header（协议 0）**

  传送路径上每个路由器都要处理，用于巨型数据包和路由器警报。如 RSVP 资源预留协议

- **目标选项报头 Destination Options Header（协议 60）**

  该包头承载特别针对数据包目的地的可选信息，例如用在移动节点和家乡代理之间交换注册信息，移动

IP 是这样的一个协议，即使移动节点改变了连接点，仍允许它保持永久的 IP 地址

- **路由包头 Routing Header （协议 63）**

  在数据包发往目的地的途中，该包头能够被 IPv6 源节点用来强制数据包经过特定的路由器，当路由类型字段为 0 时，在路由包头中可以指定中间路由器列表，这个功能类似 IPv4 中的松散源路由选项。

- **分段报头 Fragment Header （协议 44）**

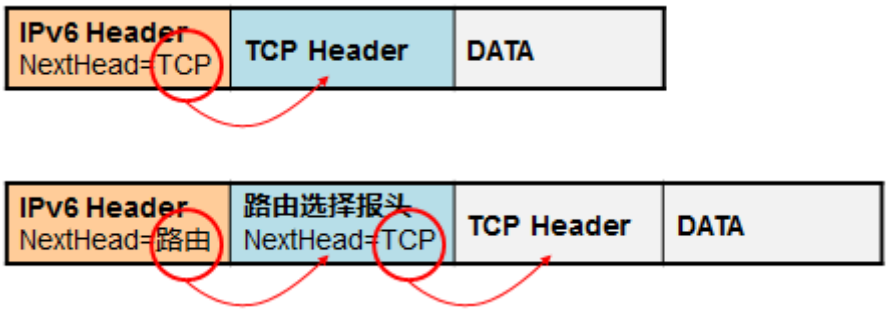
  在 IPv6 中，建议所有的节点使用 PMTUD 机制来发现链路的 MTU，这样就不需要对数据进行分段了。但是如果 IPv6 节点不支持 PMTUD，但又必须发送比路径 MTU 还大的数据包时，就不得不分段了，这时候需使用分段包头，节点将数据分段，使用分段包头发送每个分段。

- 认证报头 Authentication Header （协议 51）

  AH 头，这个很熟悉了

- 封装安全有效载荷报头 Encapsulating Security Payload Header（协议 51）

  ESP 头



3 IPv6 编址

3.1 基础

1. **IPv6 地址简写**

   IPv6 地址在简写的时候多个前导 0 可以省略成一个 0；

   如：2001：**00**a8：**0**207：0000：0000：0000：0000：8207

   可缩写成：2001：a8：207：0：0：0：0：8207

   一个或多个连续的 16 比特字段为 0 时，可用：：表示，但整个缩写中只允许有一个：：

   如上面的例子，可以进一步缩写成：2001：a8：207：：8207

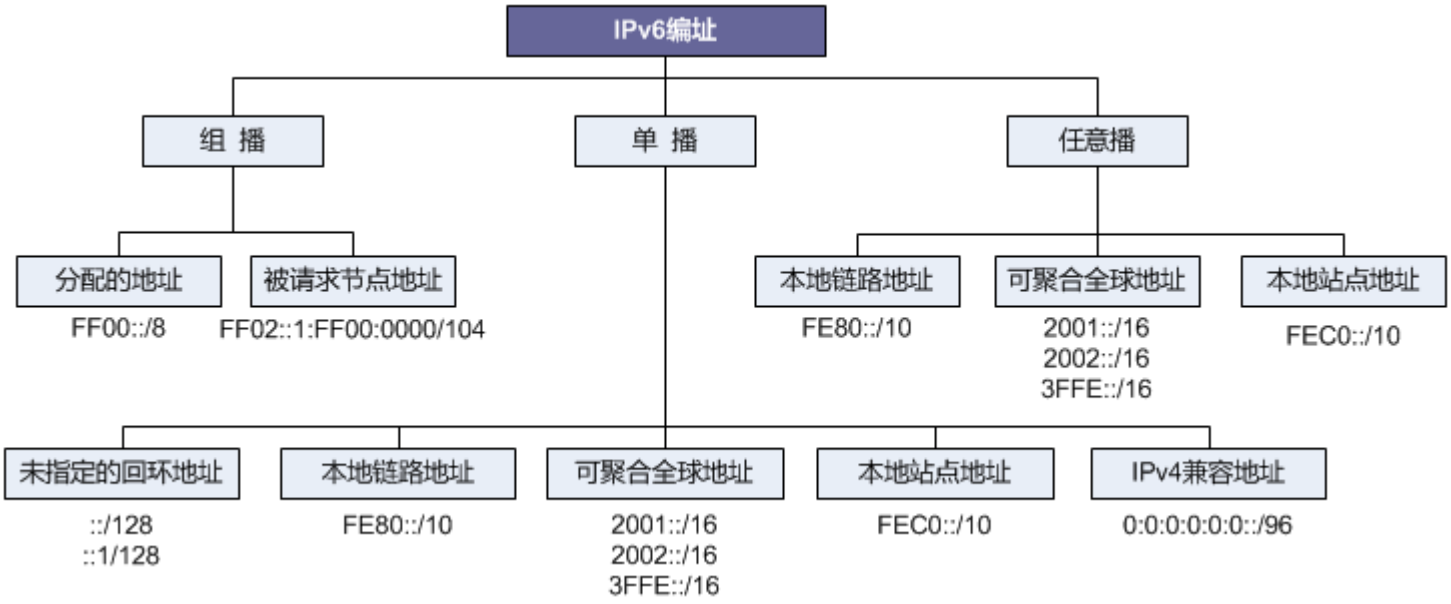
2. IPv6 地址的 URL 表示

http://[ipv6 地址]：8080　　!! 必须用 [ ] 括起来

3. IPV6 和子网划分

IPv6 与 v4 的不同之处是网络前缀范围内没有保留广播号，在 v4 中，第一个和最后一个地址是被保留给网络号和广播号的。而在 IPv6 中，不再有广播了。而且事实上 IPv6 的地址空间如此之大，以至于 VLSM 都不是特别有必要了。

3.2 地址分类



整个 IPv6 空间的已分配部分：

二进制的前缀（高 bits）	Hex	Size	Allocation
0000 0000 xxxx xxxx	0000 – 00FF	1/256	Reserved 未指定、回环、IPv4 兼容地址
0000 0001 xxxx xxxx	0100 – 01FF	1/256	Unassigned
0000 001x xxxx xxxx	0200 – 03FF	1/128	NSAP
0000 010x xxxx xxxx	0400 – 05FF	1/128	IPX -> moving to Unassigned
0000 011x xxxx xxxx	0600 – 07FF	1/128	Unassigned
0000 1xxx xxxx xxxx	0800 – 0FFF	1/32	Unassigned
0001 xxxx xxxx xxxx	1000 – 1FFF	1/16	Unassigned
001x xxxx xxxx xxxx	2000 – 3FFF	1/8	可聚合全球单播地址（IANA to registers）
010x xxxx xxxx xxxx	4000 – 5FFF	1/8	Unassigned

011x xxxx xxxx xxxx	6000 – 7FFF	1/8	Unassigned
100x xxxx xxxx xxxx	8000 – 9FFF	1/8	Unassigned
101x xxxx xxxx xxxx	A000 – BFFF	1/8	Unassigned
110x xxxx xxxx xxxx	C000 – DFFF	1/8	Unassigned
1110 xxxx xxxx xxxx	E000 – EFFF	1/16	Unassigned
1111 0xxx xxxx xxxx	F000 – F7FF	1/32	Unassigned
1111 10xx xxxx xxxx	F800 – FBFF	1/64	Unassigned
1111 110x xxxx xxxx	FC00 – FDFF	1/128	Unassigned
1111 1110 0xxx xxxx	FE00 – FE7F	1/512	Unassigned
1111 1110 10xx xxxx	FE80 – FEBF	1/1024	本地链路
1111 1110 11xx xxxx	FEC0 – FEFF	1/1024	本地站点
1111 1111 xxxx xxxx	FF00 – FFFF	1/256	组播

几点说明：

- 200::/7 保留用于网络业务接入点 NSAP ,当前没有使用 NSAP 保留空间 ,NSAP 地址主要用于 ATM 技术中。
- 2000::/3 (2000::到 3FFF::)是可聚合全球单播地址空间，

3.2.1 单播地址

1. **可聚合全局单播地址（Aggregatable global unicast address）**

   目前已经分配的全球单播地址前缀都是以 001 开头的，(2000：： 到 3FFF:FFFF:….FFFF)

   以下是这个区间地址的分配情况：

Unicast Global [001]		
2001::/16	0010 0000 0000 0001	IPv6 InternetARIN, APNIC, RIPE NCC, LACNIC
2002::/16	0010 0000 0000 0010	6 to 4 transition mechanisms
2003::/16	0010 0000 0000 0011	IPv6 InternetRIPE NCC
2400:0000::/19 2400:2000::/19 2400:4000::/21	0010 0100 0000 0000	IPv6 InternetAPNIC
2600:0000::/22 2604:0000::/22 2608:0000::/22 260C:0000::/22	0010 0110 0000 0000 0010 0110 0000 0100 0010 0110 0000 1000 0010 0110 0000 1100	IPv6 InternetARIN
2A00:0000::/21 2A01:0000::/23	0010 1010 0000 0000 0010 1010 0000 0001	IPv6 InternetRIPE NCC
3FFE::/16		6bone

一般从运营商处申请到的 IPv6 地址空间为/48，再由自己根据需要进一步规划：

如下图：

2. 本地站点地址 (Site-local address)

类似 IPv4 中的私有地址。

以 FEC0 : : /10 为前缀。其中前 10 bits 固定为 1111111011，紧跟在后面的是连续 38 bits 的 0。因此，对于站点本地地址来说，前 48bits 总是固定的。在接口 ID 和高位 48bits 特定前缀之间有 16bits 子网 ID 字段，供机构在内部构建子网。站点本地地址不是自动生成的，是手工配置的。

本地站点地址永远不会用于与全球 ipv6 因特网通信。一般用于内网通信。当然，其实 IPv6 地址空间如此之庞大，以至于根本不需要用到 Site-local 地址，直接用全局单播地址即可。

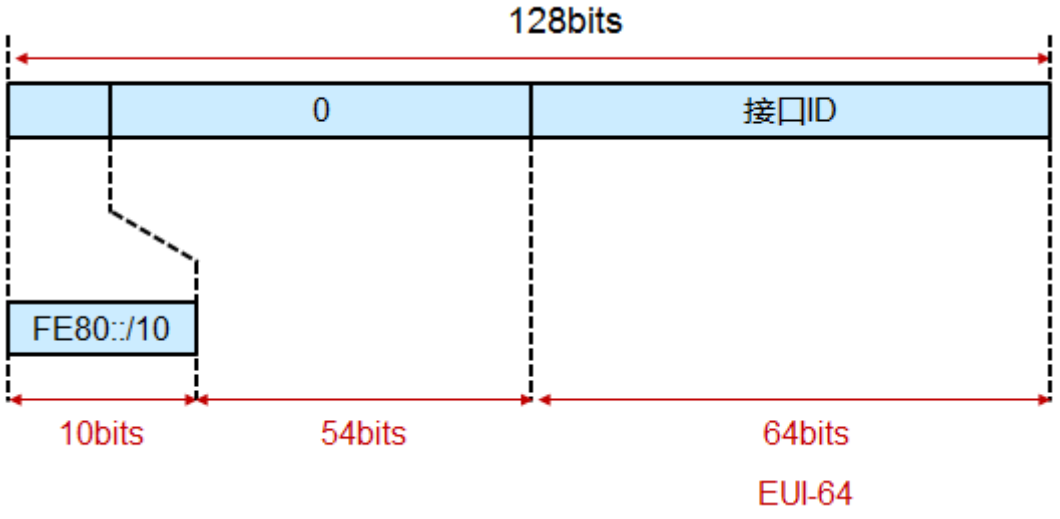
3. 链路本地地址 (Link-local address)

以 FE80::/10 为前缀，11-64 位为 0，外加一个 64bits 的接口标识（一般是 EUI-64）。

只能在连接到同一本地链路的节点之间使用，也就是链路范围内有效。

用于自动地址配置、邻居发现、路由器发现等。

当一个节点启动 IPv6 协议栈时，节点的每个接口会自动配置一个链路本地地址。这种机制使得两个连接到同一链路的 IPv6 节点不需要做任何配置就可以通信 。缺省网关建议使用链路本地地址，因为这个地址是最稳定的。



4. **特殊的地址**

- **未指定地址：全 0 地址 ::　　0:0:0:0:0:0/128　或者 :/128**

  该地址可以表示某个接口或者节点还没有 IP 地址，可以作为某些报文的源 IP 地址（比如作为 DAD 报文的 DHCP 初始化过程中客户端的源 IP）。用于 IPv6 节点在没有获取 IPv6 地址时的场景

- **回环地址：::1　　0:0:0:0:0:1/128　或者::1/128**

  用于 IPv6 节点发送报文给自己

- **IPv4 兼容地址**

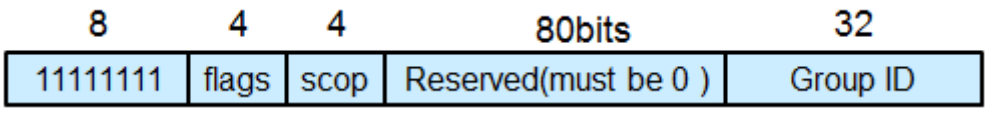
  在过渡技术中会使用到一些包含 IPv4 地址的 IPv6 地址，为了让 IPv4 地址显得更加突出一些，定义了内嵌 IPv4 地址的 IPv6 地址格式。内嵌 IPv4 地址格式是过渡机制中使用的一种特殊表示方法。在这种表示方法中，IPv6 地址的部分使用十六进制表示，IPv4 地址部分可用十进制格式。

  0:0:0:0:0:0:192.168.1.2 或者::192.168.1.2（96 个 0）

  IPv4 兼容地址用于过渡机制，如自动 ipv4 兼容隧道 及 NAT-PT（详细请见本文档 IPv6 过渡技术章节）。

3.2.2 Multicast

1. **地址结构**



组播地址最高位前 8 位固定为全 1　　FFXX：：/8

Flags　4bits，0000：永久分配或众所周知的；0001：临时的

Scop　用来限制组播数据流在网络中发送的范围。

- 0：预留；
- 1：节点本地范围；　单个接口有效，仅用于 Loopback 通讯
- 2：链路本地范围；　如 FF02::1 表示链路上的所有节点；FF02::9：表示链路上的所有 RIP 路由器
- 3：本地子网范围；
- 4：本地管理范围；
- 5：本地站点范围；
- 8：组织机构范围；
- E：全球范围；

- F：预留。

Group-ID 该字段长度可以为 112 位，用来标识组播组，而 112 位最多可以生成 2112 个组 ID ，RFC2373 并没有将所有的 112 位都定义成组标识 ,而是建议仅使用该 112 位的最低 32 位组 ID，将剩余的 80 位都置 0。

2. IPv6 有一些特殊的组播地址，这些地址有特别的含义

FF01::1 节点本地范围所有节点组播地址

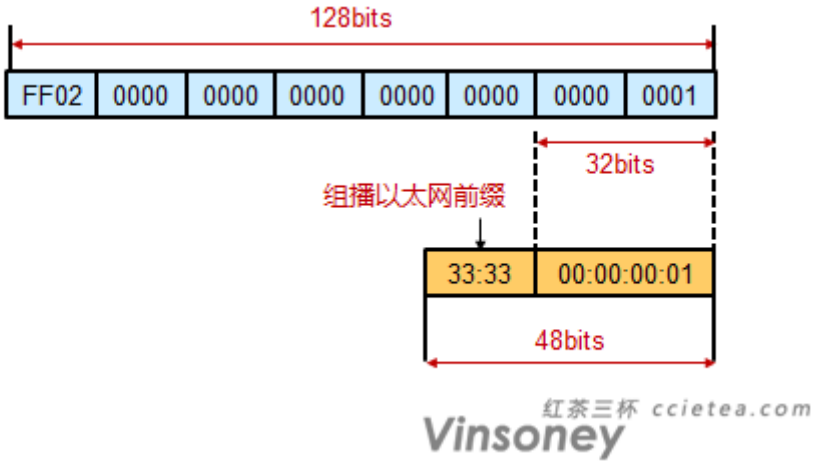
FF01 ::2 节点本地范围所有路由器组播地址

FF02::1 链路本地范围所有节点组播地址

FF02::2 链路本地范围所有路由器组播地址

.....

3. IPv6 组播地址的 MAC 地址映射



4. 被请求节点组播地址 Solicited-node

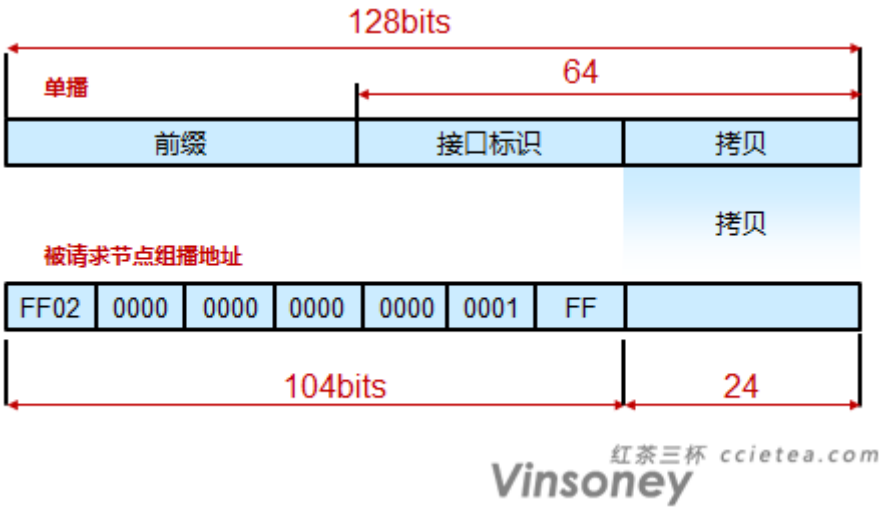
在 IPv6 组播地址中，有一种特别的组播地址，叫做 Solicited-node 地址(被请求节点组播地址)。

Solicited-node 地址是一种特殊用途的地址．主要用于重复地址检测（DAD）和替代 IPv4 中的 ARP。

Solicited-node 地址由前缀 FF02::1:FF00:0 / 104 和 ipv6 单播地址的最后 24 位组成。

一个 IPv6 单播地址对应一个 Solicited-node 地址。Solicited-node 地址有效范围为本地链路范围。

地址格式： FF02:0:0:0:0:1:FFXX:XXXX，具体的对应关系如下：



3.2.3 Anycast

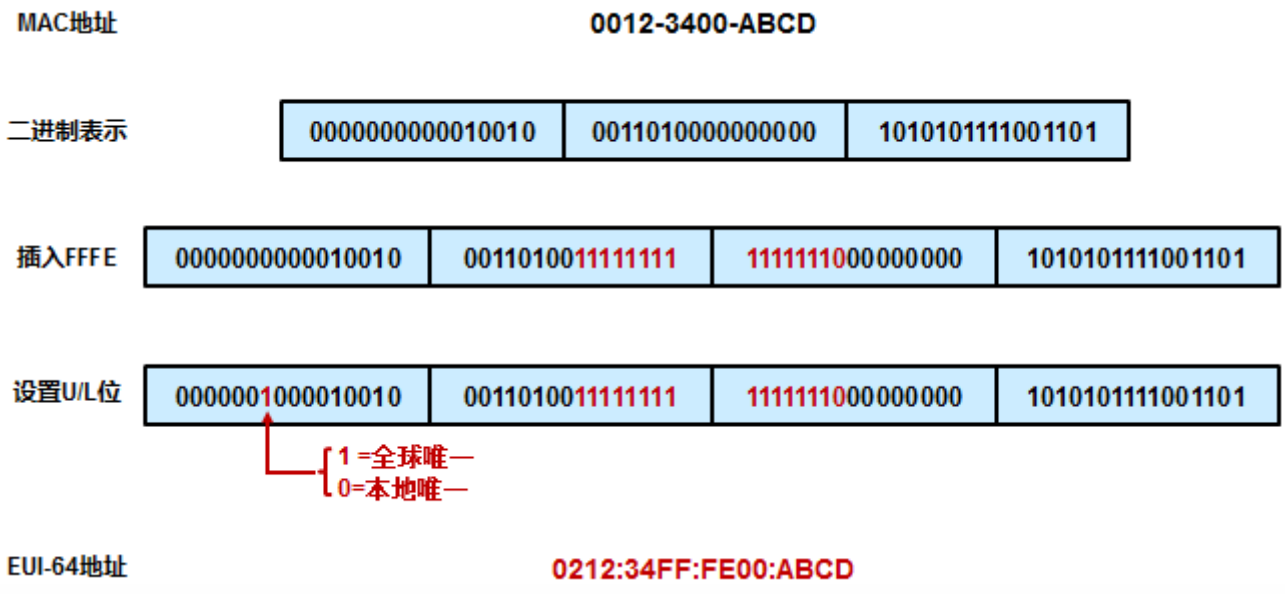
任播地址没有专门的地址空间，使用的是单播的地址空间。一个 IPv6 地址被分配给多个接口，仅用于路由器，发往任播地址的数据包被路由给分配了任播地址中的最近的一个（由路由协议判断远近）。适合于 One-to-One-of-Many(一对一组中的一个)的通信场合。接收方只需要是一组接口中的一个即可，如移动用户上网就需要因地理位置的不同而接入离用户最近的一个接收站，这样才可以使移动用户在地理位置上不受太多的限制。任播地址只能作为目的 IP，不能作为源。

3.3 接口 ID

接口 ID 为 64bits，可以根据 IEEE 的 EUI-64 规范将 48 比特的 MAC 地址转化为 64 比特的接口 ID。

MAC 地址的唯一性保证了接口 ID 的唯一性。

设备自动生成，不需人为干预，由接口的 MAC 地址转换得到 64bits 的 EUI-64 格式接口 ID 的过程如下：



将 48bits 的 MAC 对半劈开，插入 FFFE，然后设置第 7 位，也就是 U/L 位，该比特位确定 48bits 的 MAC 地址的唯一性与否。一个以太网地址可以有两种含义，地址可被全球管理或本地管理。全球管理指使用如 0800-2bxx-xxxx 之类的厂商 MAC，本地管理指使用自己的值重写 MAC 地址，在这种情况下，这个特殊的位=1 表示本地管理；为 0 表示全球管理。但是在 EUI-64 格式中，U/L 的含义正好相反，0 表示本地管理，1 表示全球管理，所以使用 EUI-64 格式的地址 IPv6 地址，U/L 位为 1，则地址是全球唯一的，如果为 0，则为本地唯一。

3.4 必须具备的 IPv6 地址

- **节点必须具备的 IPv6 地址：**

链路本地地址	FE80::/10
回环地址	::1
所有节点组播地址	FF01::1；FF02::1
分配的可聚合全球单播地址	2000::/3
所使用的每个单播和任意播地址对应的被请求节点组播地址	FF02::1:FFxx:xxxx 其中 xx:xxxx 是每个单播或任意播地址的低 24 比特
主机所属的所有组的组播地址	FF00::/8

以上是主机节点必须具备的地址，路由器的有所差异，多了以下地址：

所有路由器组播地址：FF01::2 ； FF02::2 ； FF05::2

子网路由器任意播地址：UNICAST_PREFIX :0:0:0:0

其他任意播配置地址：2000::/3

3.5 IPv6 地址配置方法

1. 手工配置

```
R1(config)# interface fast0/0
R1(config-if)# ipv6 enable
R1(config-if)# ipv6 address 2001:0001::/64 eui-64
```

上面这条命令的意思是使用 2001:0001::/64 作为前缀，并且追加 64bits 的 EUI-64 格式接口 ID，构成接口的全局唯一 IPv6 地址。

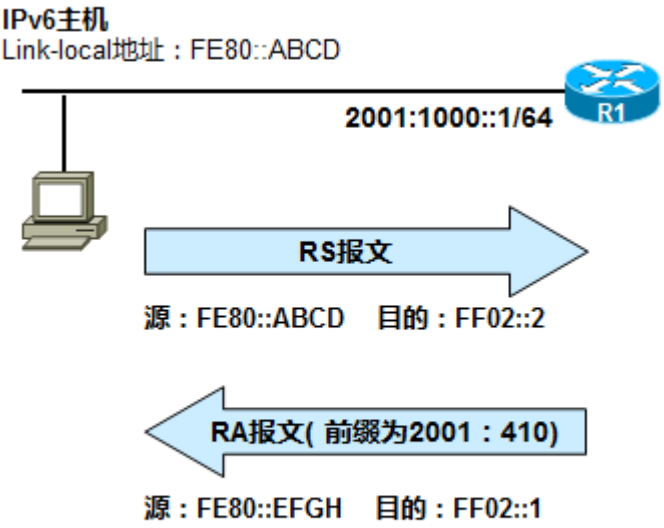
2. 自动配置

IPv6 地址的自动配置一般有两种方式：

- **有状态地址自动配置（DHCPv6）**
- **无状态地址自动配置**

  为了自动获得这个前缀，只要在路由器和主机之间运行一个协议即可。使用 NDP 协议的 Router Solicitation 恳求和 Router Advertisement 通告消息。前者用于发现路由器，并促使路由器发送 Router Advertisement 消息通报前缀信息。RA 消息中包含前缀、生存期、缺省网关等信息。大致过程如下图，更详细的内容见本文档相关章节。

1. 主机发送router Solicitation报文
2. 路由器回应Router Advertisement报文
3. 主机获得前缀及其它参数
4. 路由器周期性地向外发送RA报文



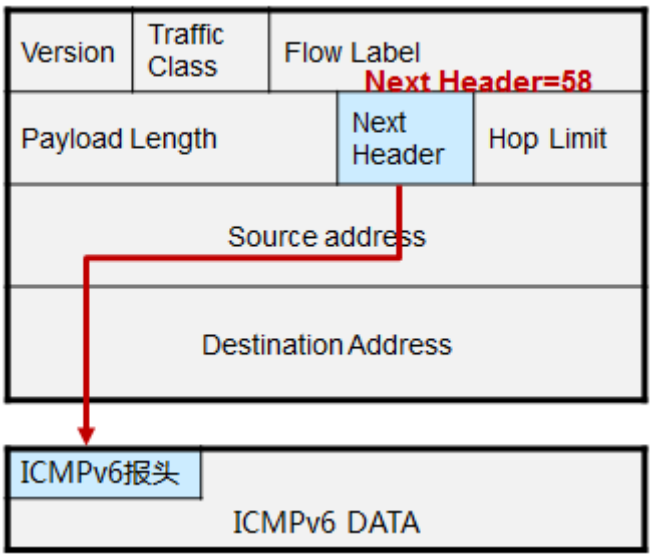
获取到的地址：2001:1000::ABCD

4 ICMPv6（RFC2463）

4.1 Foundation

ICMPv6 是 IPv6 的基础协议之一。协议号 58，该协议号在 IPv6 报头的“下一个包头”字段中。

ICMP 报文有两种：差错消息及信息消息



4.2 消息类型

消息类型	TYPE	名称	CODE
差错消息	1	目的不可达	0 无路由
			1 因管理原因禁止访问
			2 未指定
			3 地址不可达
			4 端口不可达
	2	数据包过长	0
	3	超时	0 跳数到 0
			1 分片重组超时
	4	参数错误	0 错误的包头字段
			1 无法识别的下一包头类型
			2 无法识别的 ipv6 选项
信息消息	128	Echo request	0
	129	Echo reply	0

还有一些其他报文，在下面介绍

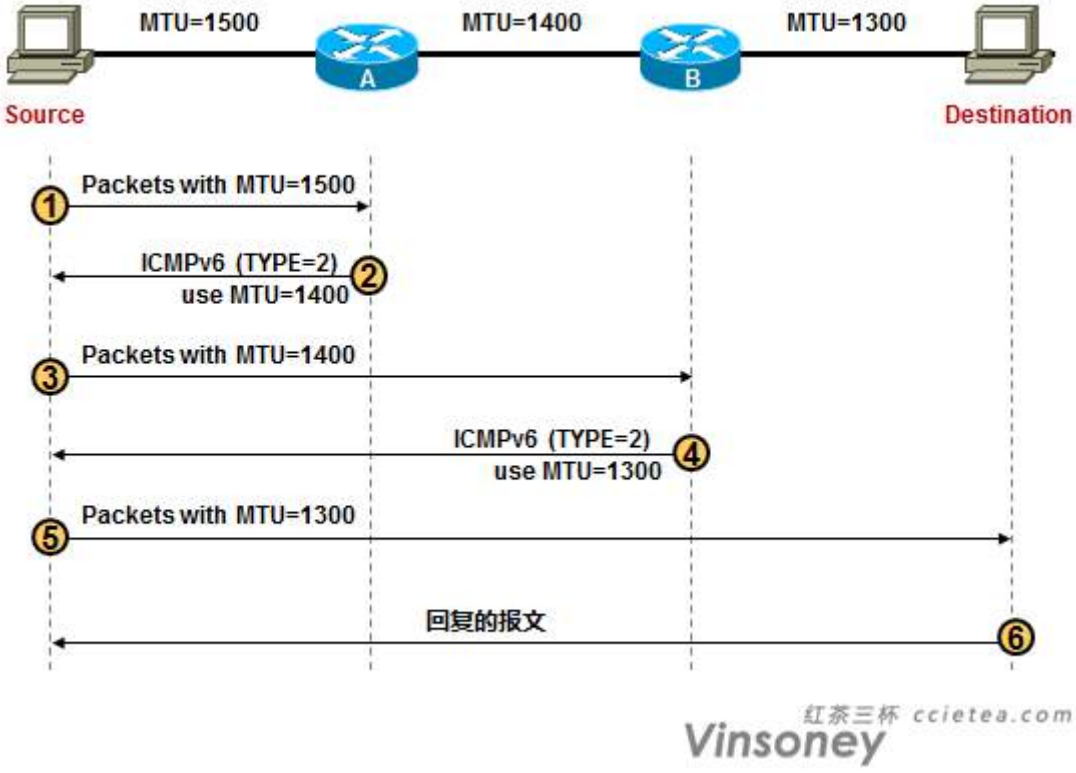
4.3 PMTUD

PMTU 就是路径上的最小接口 MTU。

PMTUD 的主要目的是发现路径上的 MTU，当数据包被从源转发到目的地的过程中避免分段。依赖 PMTUD 源节点可以使用所发现到的最大 PMTU 与目的地节点进行通信，这样可以避免数据包在从源传输到目的的过程之中，被中途的路由器分片，而导致性能的下降。

因此 IPv6 的分片不是在中间路由器上进行的，仅当路径 MTU 比欲传送的数据包小的时候，由源节点自己对数据包进行分片。

在 RFC1981 中定义了 PMTU 发现协议。IPv6 PMUTD 使用 ICMPv6 Type 2 消息。典型过程如下：

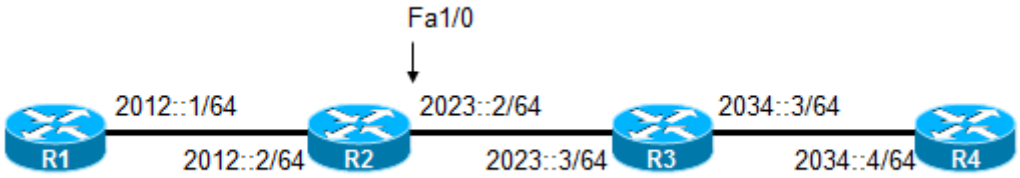


1. 首先 Source 用 1500 字节作为 MTU 向目标节点发送一个 IPv6 数据包
2. 中间路由器 A 意识到数据包过大，MTU 为 1400，于是回复一个 ICMPv6 type=2 消息向 Source 应答，在该 ICMPv6 消息中指定较小的 MTU=1400
3. Source 开始使用 MTU=1400 发送 IPv6 数据包，该数据包到了 B
4. 然而 B 意识到本地接口 MTU 为 1300，于是回复一个 ICMPv6 type=2 消息向 Source 应答
5. Source 开始使用 MTU=1300 发送 IPv6 数据包，该数据包顺利到达了目的地。
6. Source 和 Destination 之间的会话被建立起来。

注意：

- 这里的 PMTU，指的是单向的，沿路上的数据包出接口 MTU 中最小的。
- IPv6 要求的链路层所支持的 MTU 最小为 1280bytes
- 用 IPv6 PMTUD 发现的 MTU 值被源节点缓存，使用 CISCO IOS，可以用 show ipv6 mtu 来显示每个缓存的目的地 PMTUD 值

测试：



1. 准备工作

完成环境的搭建及设备的基本配置，保证全网互通。

R1 去 ping R4，能够 ping 通。

2. 实验测试

现在，将 R2 的 Fa1/0 口的接口 ipv6 MTU 修改为 1400 字节（上图中默认接口 IPv6 MTU 都是 1500 字节）：

```
Interface fast1/0
  Ipv6 mtu 1400
```

完成后，在 R2 上简单的 show 一下：

```
R2#show ipv6 interface f1/0
FastEthernet1/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE01:CFF:FEF0:10
  Global unicast address(es):
    2023::2, subnet is 2023::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::5
    FF02::6
    FF02::1:FF00:2
    FF02::1:FFF0:10
  MTU is 1400 bytes
.....
```

现在，我们再到 R1 上，

```
R1#ping 2034::4 repeat 1 size 1500          !! 让 R1 产生一个报文大小为 1500 的包，发给 R4
Type escape sequence to abort.
Sending 1, 1500-byte ICMP Echos to 2034::4, timeout is 2 seconds:
B
```

结果不通，因为报文在传递到 R2 时，超出了 R2 Fa1/0 口的 MTU，因此 R2 回送一个 TYPE=2（packet too big）的 ICMPv6 消息给 R1。在这个回送给 R1 的 ICMPv6 消息中，包含允许的 MTU=1400，以便告知 R1，后续发送的报文不要超过 1400 字节。这个过程抓包如下：

Time	Source	Destination	Protocol	Length	Info
62.942000	2012::1	2034::4	ICMPv6	1514	Echo (ping) request id=0x02
62.963000	2012::2	2012::1	ICMPv6	1294	Packet Too Big

其中，R2 回送给 R1 的 ICMPv6 差错消息如下：

```
Ethernet II, Src: cc:01:0c:f0:00:00 (cc:01:0c:f0:00:00), Dst: cc:00:0c:f0:00:
Internet Protocol Version 6, Src: 2012::2 (2012::2), Dst: 2012::1 (2012::1)
Internet Control Message Protocol v6
  Type: Packet Too Big (2)
  Code: 0
  Checksum: 0xbb6f [correct]
  MTU: 1400
⊞ Internet Protocol Version 6, Src: 2012::1 (2012::1), Dst: 2034::4 (2034::4)
⊞ Internet Control Message Protocol v6
```

此时，在 R1 上进一步查看一下：

```
R1#sh ipv6 mtu
MTU        Since       Source Address            Destination Address
1400       00:00:27    2012::1                   2034::4
```

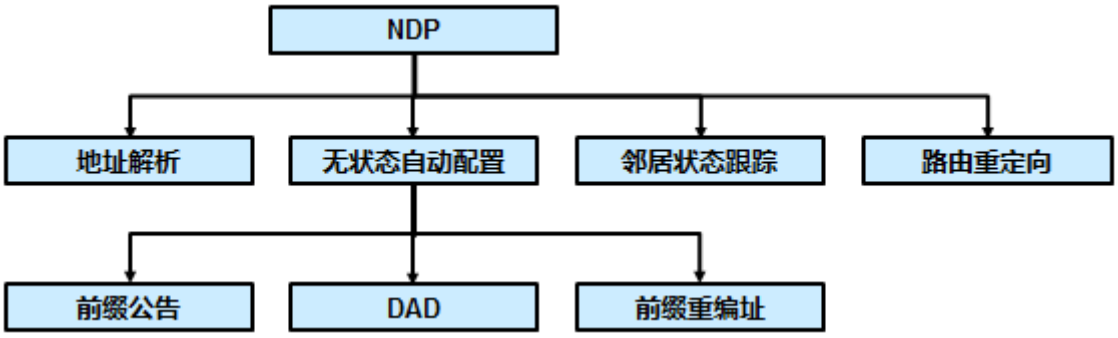
我们发现，R1 创建了一条缓存，标识从 2012::1 到 2034::4，PMTU=1400，因此 R1 在后续发送给 2034::4 的报文中，如果超出 1400 字节，R1 自己就会进行分片，例如我们再做测试，在 R1 上继续 ping 2034::4，报文大小为 1500 字节，则：

```
2012::1   2034::4   IPv6      1414 IPv6 fragment (nxt=ICMPv6 (0x3a) off=0 id=0x1)
2012::1   2034::4   ICMPv6     170 Echo (ping) request id=0x06c8, seq=0
2034::4   2012::1   ICMPv6    1514 Echo (ping) reply id=0x06c8, seq=0
```

上图中，前面两个包就是 R1 发送给 R4 的 ICMPv6 报文的两个分片，第三个报文是 R4 的 reply 包

所以这样 R1 发送大包给 R4 的时候，就不会有问题了。直接在 R1 源头这就分片。当然，我们可以继续试验，在 R3 的 Fa1/0 接口上将 IPv6 MTU 设置为 1300 字节，然后继续 R1 去 ping 大包给 R4，最终 R1 上的缓存条目，PMTU 会变成 1300 字节。

5 NDP 邻居发现协议（RFC2461）



NDP 可以帮助我们实现以下功能：

1. **地址解析(代替 ARP 协议，使用 ICMP 完成地址解析)**

   IPV6 取消了 arp 使用 NS 和 NA 来做，利用的是 solicited address

   通过邻居请求（NS）和邻居通告（NA）报文来解析三层地址对应的链路层地址。

2. **邻居的状态跟踪**

   Show ipv6 neighbor

3. **无状态自动配置**
4. **重复地址检测（DAD）**
5. **前缀重编址**
6. **路由器重定向**

   路由器向一个 Ipv6 节点发送 icmpv6 消息，通知它在相同的本地链路上才能在一个更好的到达目的地望楼的路由器地址

5.1 为 NDP 定义的 icmpv6 消息

ICMPv6 TYPE	消息名称
133	路由器请求（RS）
134	路由器通告（RA）
135	邻居请求（NS）
136	邻居通告（NA）
137	重定向消息

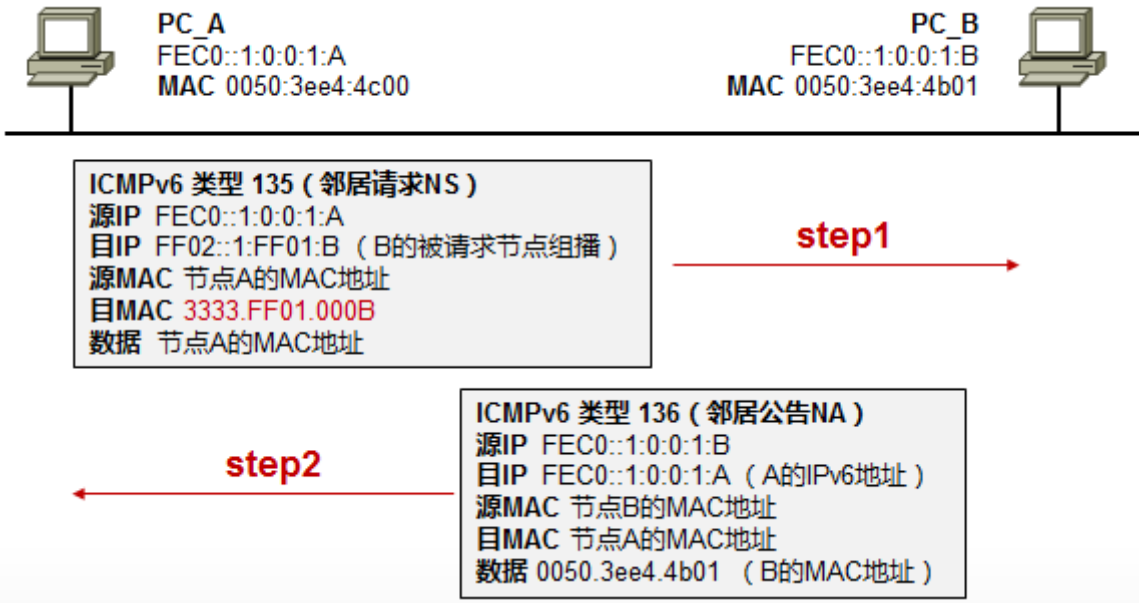
NDP 机制使用的 ICMPv6 消息：

机制	RS 133	RA 134	NS 135	NA 136	重定向消息 137
报文介绍	主机可以发送 RS 要求路由器立即产生 RA	包含 MTU、前缀信息等	用来判断邻居的链路层地址，也用于 DAD 等		
替代 ARP			X	X	
前缀公告	X	X			
前缀重新编制	X	X			
DAD			X		
路由重定向					X

RS 133 指的是使用 ICMPv6 TYPE133 Code0 的 RS 报文，其他报文类似。

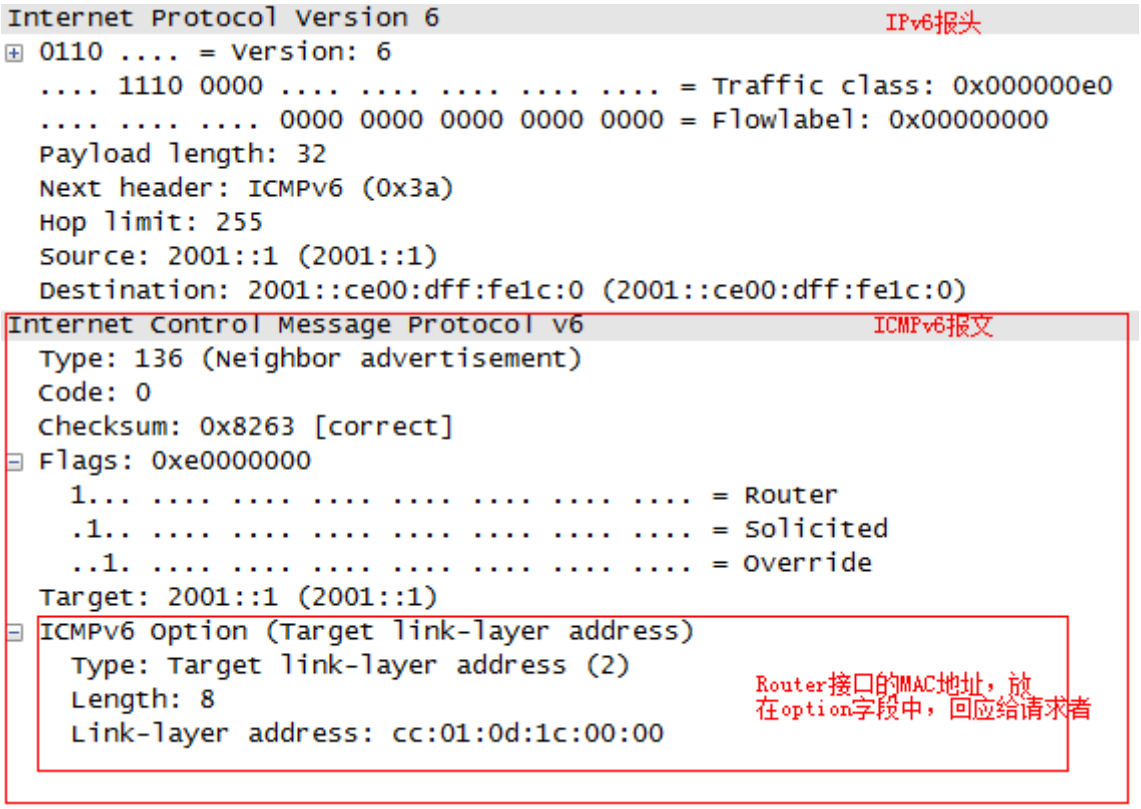
5.2 地址解析

在 IPv6 中，对节点链路层地址的确定，使用 NS、NA 和被请求节点组播地址的组合来完成。这比 IPv4 的 ARP 要高效得多。我们来看一下典型的地址解析过程：



其中 33:33:FF:01:00:0B 是 ipv6 目的地址 FF02::1:FF01:B 的多播映射。

以下是一个 NA 报文的示例：



注意到其中的 Flag 字段，

R 位置 1 表示这是台路由器，

S 位置 1 表示这是响应某个邻居请求的通告，置 0 则为主动发送的，

O 位置 1 表示 NA 消息中的条目是否覆盖已有，收到该消息的邻居会覆盖已有条目

- **使用 show ipv6 neighbors 可以查看邻居表项**

R1#show ipv6 neighbors

IPv6 Address	Age	Link-layer Addr	State	Interface
2012::2	11	cc01.0cf0.0000	STALE	Fa0/0
FE80::CE01:CFF:FEF0:0	11	cc01.0cf0.0000	STALE	Fa0/0

其中 State 如果为 REACH 表示邻居可达；为 STALE 意味着这些邻居在最后的 30s 内是不可达的

- **使用“ipv6 neighbor ipv6 地址 接口编号 mac 地址”，可添加一个静态表项到邻居发现表，例如：**

Router(config)# ipv6 neighbor 2012::2 fastEthernet 0/0 cc01.0cf0.0000

R1# show ipv6 neighbors

IPv6 Address	Age	Link-layer Addr	State	Interface
2012::2	-	cc01.0cf0.0000	REACH	Fa0/0

- **使用“clear ipv6 neighbors”，会清楚邻居发现表中的动态表项**

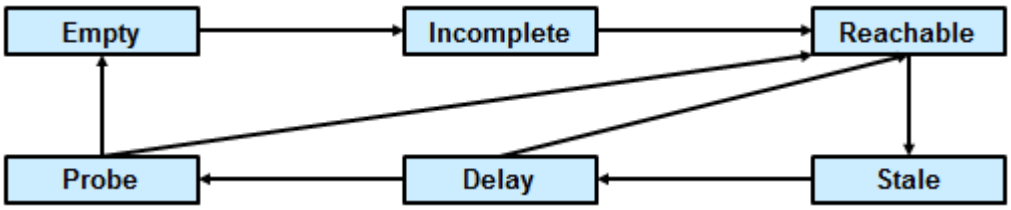
- **调整 NDP 消息的参数**

  Ipv6 nd ns-interval !! 用于设置邻居请求消息的时间间隔，默认是 1000ms 也就是 1S

  Ipv6 nd reachable-time !! 一个邻居在由某个事件证实它的可达性后，在这段时间内这个邻居被认为是可达的，默认 30S（在 CISCO IOS 上 show 出来的，书上说的是 30min，估计是写错了）

5.3 邻居状态

Incomplete	邻居请求已经发送到目标节点的请求组播地址，但没有收到邻居的通告
Reachable	可达，收到确认，不续再发包确认
Stale	从收到上一次可达性确认后过了超过 30s。
Delay	在 stale 状态后发送过一个报文，并且 5s 内没有可达性确认
Probe	每隔 1s 重传邻居请求来主动请求可达性确认，直到收到确认



1. A 发送 NS，并生成缓存条目，A 上条目的状态为 Incomplete
2. 若 B 回复 NA，则 A 上关于 B 的邻居状态就由 Incomplete ->Reachable。但若 A 发出 NS 消息后一定时间内仍没有收到任何的回复，则由 Incomplete->Empty，即删除条目
3. 如果在 reachable 状态上经过 ReachableTime（默认 30S），A 路由器上关于 B 的条目状态 Reachable->stale 或如果在 reachable 状态上，收到 B 的非请求 NA，且链路层地址不同，则马上->stale
4. 在 Stale 状态若 A 要向 B 发送数据，可直接发送，并从 A 上关于 B 的条目由 Stale->Delay，同时会等待应用层的提示信息，提示邻居是否可达
5. 如果在 Delay_First_Probe_Time（默认 5S）内，有 NA 应答或者应用层的提示信息（例如我发了 ICMP 包给对端，对端回复我 ICMP 了，那就是上层可达），则 Delay->Reachable，如果无应用层提示信息，Delay->Probe
6. 在 Probe 状态，每隔 RetransTimer（默认 1S）发送单播 NS，发送 MAX_Unicast_Solicit 个后再等 RetransTimer，有应答则 Reachable，无则进入 Empty，即删除条目

测试的时候可开 debug ipv6 nd

5.4 无状态自动配置

涉及机制

- 前缀公告
- DAD
- 前缀重新编址

5.4.1 路由器公告 RA

```
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x4a68 [correct]
  Cur hop limit: 64
⊟ Flags: 0x00
    0... .... = Not managed
    .0.. .... = Not other
    ..0. .... = Not Home Agent
    ...0 0... = Router preference: Medium
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊟ ICMPv6 Option (Source link-layer address)
    Type: Source link-layer address (1)
    Length: 8
    Link-layer address: cc:01:0d:1c:00:00
⊟ ICMPv6 Option (MTU)
    Type: MTU (5)
    Length: 8
    MTU: 1500
⊟ ICMPv6 Option (Prefix information)
    Type: Prefix information (3)
    Length: 32
    Prefix length: 64
  ⊟ Flags: 0xc0
      1... .... = Onlink
      .1.. .... = Auto
      ..0. .... = Not router address
      ...0 .... = Not site prefix
    Valid lifetime: 2592000
    Preferred lifetime: 604800
    Prefix: 2001::
```

- Cur Hop Limit： 表示当 PC 使用该 RA 通告的前缀构建 IPv6 地址，那么该 PC 发送的 IPv6 报文 hoplimit 都是该值（64）
- Flags 如果 not managed 位=1 表示主机使用有状态地址自动配置（如 DHCPv6）；not other 位为 1 则表示主机使用有状态地址自动配置来获取除了地址以外的参数
- RouterLifetime 该 RA 关联的前缀发给 PC 后，PC 将 Router 视为网关，这个 lifetime 就是这个意思
- Reachable time 及 retrans timer 如果为 0 则表示为指定

PC 收到一个 RA 报文，会做一个检测：

1. RA 数据包的源地址，是否为一个 linklocal 地址
2. RA 数据包的 IPv6 报头的 hop limit 是否为 255。因为只有 255 才表示是本地网络中路由器产生的
3. 如果 RA 数据包包含认证信息，那么认证是否正确
4. ICMPv6 的 checksum 是否正确
5. ICMPv6 code 是否为 0
6. ICMPv6 长度大于等于 16
7. 所有的 options 长度大于等于 0

当然，如果一台 PC 初始化，也可以主动发送一个 RS 请求关于前缀信息

5.4.2 前缀公告

1. **前缀公告概述**

   前缀公告是无状态自动配置的初始机制。利用 RA（ICMPv6 Type134）和所有节点的组播地址（FF02::1），路由器 RA 消息在本地链路上周期性的发送到 FF02::1 上。

   在无状态自动配置中，前缀长度为 64 比特

   Show ipv6 interface xxx perfix !!显示接口上公告的前缀参数

2. **在 CISCO IOS 路由器上公告 IPv6 前缀**

   只要在网络接口上配置了一个本地站点或者全局可聚合单播 IPv6 地址及其前缀长度，就启用了 CISCO 路由器上的 IPv6 前缀公告。

   以下是自动配置过程期间和之后使用的参数：

   - **IPv6 前缀**

     默认情况下，无状态自动配置公告的前缀长度为 64 位，节点收到 IPv6 前缀，将自己的 EUI-64 地址附加在这 64 位的前缀后面，构成 128 位的 IPv6 地址。
   - **生存期**
   - **有效生存期（Valid lifetime）**
   - **首选生存期（preferred lifetime）**

   【前缀的两个时间】

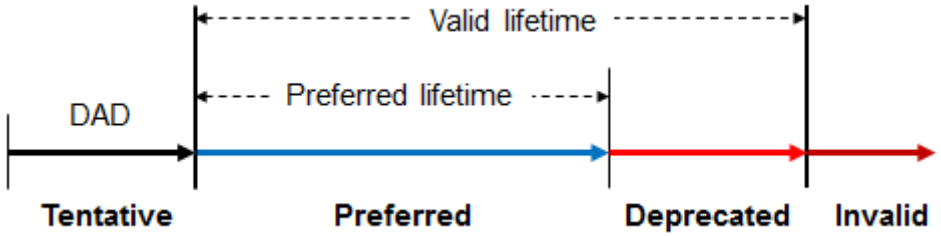
   IPv6 地址可以通过 state configuration 和 stateless configuration 获得到，简单的说

   状态化配置是通过手工添加或是通过 dhcpv6 获得，无状态化是通过 prefix advertisement 获得。不管哪种方式获得的地址，都会有几个时间状态。

其实 IPv6 地址的 lifetime 主要有两个，一个是 **Valid lifetime** ，另一个是 **preferred lifetime**。首先在 DAD 检查阶段。在 DAD success 前，地址一直处于 tentative (试验状态)状态，这时的地址是不可用状态，直到 DAD success。然后进入 Preferred 状态，也就是首选状态，这个状态过程中的 IPv6 address 的有效性是由 preferred lifetime 决定的。

Valid lifetime 是一个类似于 dhcp 中的 lease time。如果地址超过了 valid lifetime，节点就会把这个地址置为 inactive 状态，然后 delete。在 preferred time 和 valid lifetime 之间叫做 deprecated 状态，这种状态算是一种 buffer，大概意思叫做不赞成使用的状态，当地址达到这个时间段的时候，地址不能主动的发起连接只能是被动的接受连接，这也是为了保证上层应用而设计的，但是过了 valid lifetime 时间地址就变为 invalid，这时任何连接就会 down 掉。

在 CISCO IOS 上，默认 valid time 为 30 天（2592 000s），prefer time 为 7 天（604 800s）。



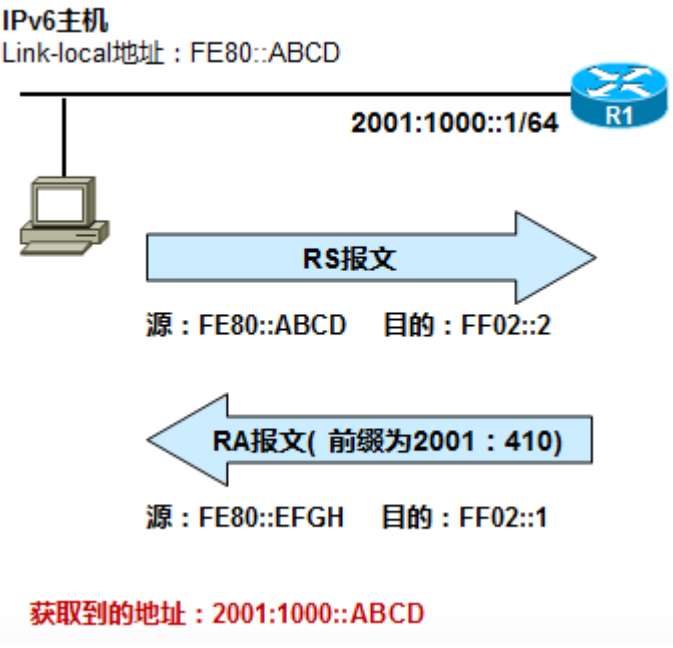
- **默认路由器信息**

  提供关于默认路由器 IPv6 地址的存在和生存期信息。在 IPv6 中，节点使用的默认路由器地址是路由器的本地链路地址（linklocal address），这样一来，即使前缀被重新编址，默认网关也是可达的（linklocal address 不会变）

- **标志/选项**

3. 前缀公告是如何工作的

1. 主机发送router Solicitation报文
2. 路由器回应Router Advertisement报文
3. 主机获得前缀及其它参数
4. 路由器周期性地向外发送RA报文



4. 调整前缀公告的参数

```
R1(config-if)#ipv6 nd prefix 2001::/64 ?
  <0-4294967295>   Valid Lifetime (secs)
  at               Expire prefix at a specific time/date
  infinite         Infinite Valid Lifetime
  no-advertise     Do not advertise prefix
  no-autoconfig    Do not use prefix for autoconfiguration
                   // 当在特定前缀后开启该参数后，该前缀不能用于无状态自动配置
  no-rtr-address   Do not send full router address in prefix advert
  off-link         Do not use prefix for onlink determination
```

举例：

```
Ipv6 nd prefix 2012::/64 30 15
!! 配置 validtime 及 prefered time，这是一个相对时间
```

配置该命令后，邻居 R2 收到 prefix，显示如下：

```
R2#sh ipv int f 0/0
FastEthernet0/0 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::CE00:DFF:FE48:0
Global unicast address(es):
```

2001::CE00:DFF:FE48:0, subnet is 2001::/64 [PRE]　　!! 状态为 prefer

valid lifetime 25 preferred lifetime 10　　!! 两个时间在不断递减

当 preferred lifetime 先到 0 时，状态变为[DEP]，当 valid time 变 0 时，地址抹去

Ipv6 nd prefix 2012::/64 at xxx yyyy

!! xxx 为 validtime，yyy 为 prefered time，这是绝对时间

Ipv6 nd prefix 2012::/64 890000 720000 off-link

!! off-link 和 L 比特有关，这个比特位在 RFC2461 中定义。当可选的 off-link 关键字在 CISCO IOS 中被配置时，L 比特被关闭。而如果 L 比特被打开（默认打开），他表示在 RA 消息中的前缀是分配给本地链路的。因此，向包含这个指定前缀的地址发送数据的节点认为目的地是本地链路可达的。

Ipv6 nd prefix 2012::/64 890000 720000 no-autoconfig

!! no-autoconfig 和 A 比特有关，RFC2461 中定义。A 比特也成为自治地址配置标志。当这个可选关键字 no-autoconfig 在 CISCO IOS 中被配置（就是配了上面的命令），A 比特被置 0。如果 A 比特被置 1（默认就是 1），它指示本地链路的主机可以使用该前缀进行无状态自动配置。

ipv nd prefix 2012::/64 no-advertise

该前缀将不包含在 RA 消息中。默认情况下，CISCO IOS 关闭这个 no-advertise，也就是前缀都会被包含在 RA 消息中。

ipv nd prefix default ?

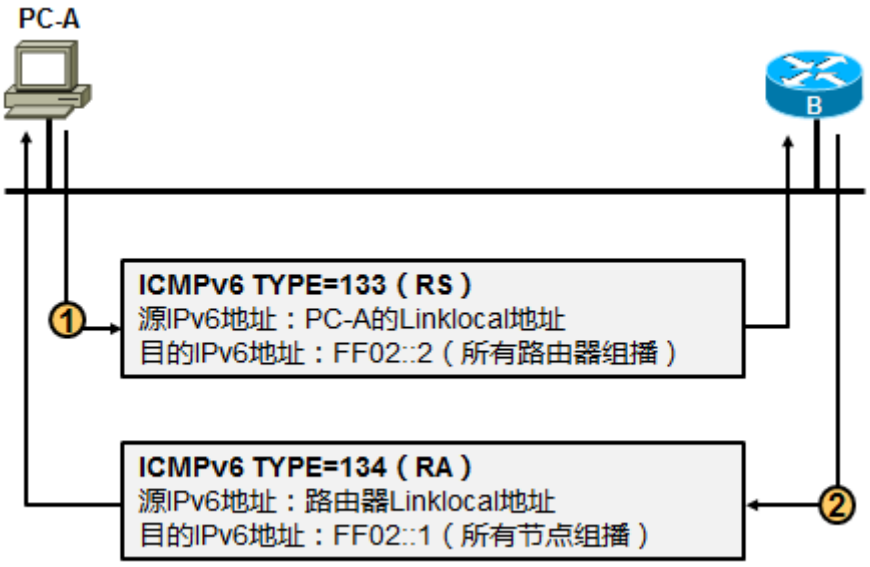
可以修改接口下，nd 前缀通告的默认参数

Ipv6 nd suppress-ra

在接口上关闭路由器公告。默认情况下，当全局命令 ipv6 unicast-routing 被启用，CISCO 设备的以太网口、FDDI 和令牌环接口上具有 RA 功能

5. **使用 RS 来请求 RA**

PC 可以主动发送 RS 来触发路由器发送 RA，而不用等 RA 的周期性发送时间到了才收 RA。



为了避免路由器请求消息在本地链路上泛滥，在启动时每个节点只能发送 3 个 RS 消息。

5.4.3 调整 IPv6 nd 参数

- **Ipv6 nd ra-lifetime**

  RA 消息的生存时间，默认 1800s

- **Ipv6 nd ra-interval**

  RA 消息的通告间隔，这个时间小于或等于 ra-lifetime。默认 200S。

- **ipv6 nd managed-config-flag**

  配置这条命令后，该接口上的前缀信息将不能被链路上的主机用于无状态自动配置，主机必须使用有状态自动配置例如 DHCPv6 来获取地址。默认情况下这命令没配，也就是说，managed-config-flag 为 0，也就是说，主机可以通过无状态配置获取地址。

```
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x4a68 [correct]
  Cur hop limit: 64
⊟ Flags: 0x00
    0... .... = Not managed
    .0.. .... = Not other
    ..0. .... = Not Home Agent
    ...0 0... = Router preference: Medium
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊞ ICMPv6 Option (Source link-layer address)
⊞ ICMPv6 Option (MTU)
⊞ ICMPv6 Option (Prefix information)
```

- **ipv nd other-config-flag**

  Hosts should use DHCP for non-address config

  如果配置了该命令，则主机需使用 dhcp 配置除了 ipv6 地址外的其他信息，如 DNS，域名什么的

  这个标志也与有状态自动配置有关，当它没有置位（默认情况下），节点不应该使用有状态自动配置机制来配置除了 IPv6 地址以外的其他参数。

5.4.4 DAD（Duplicate Address Detection）

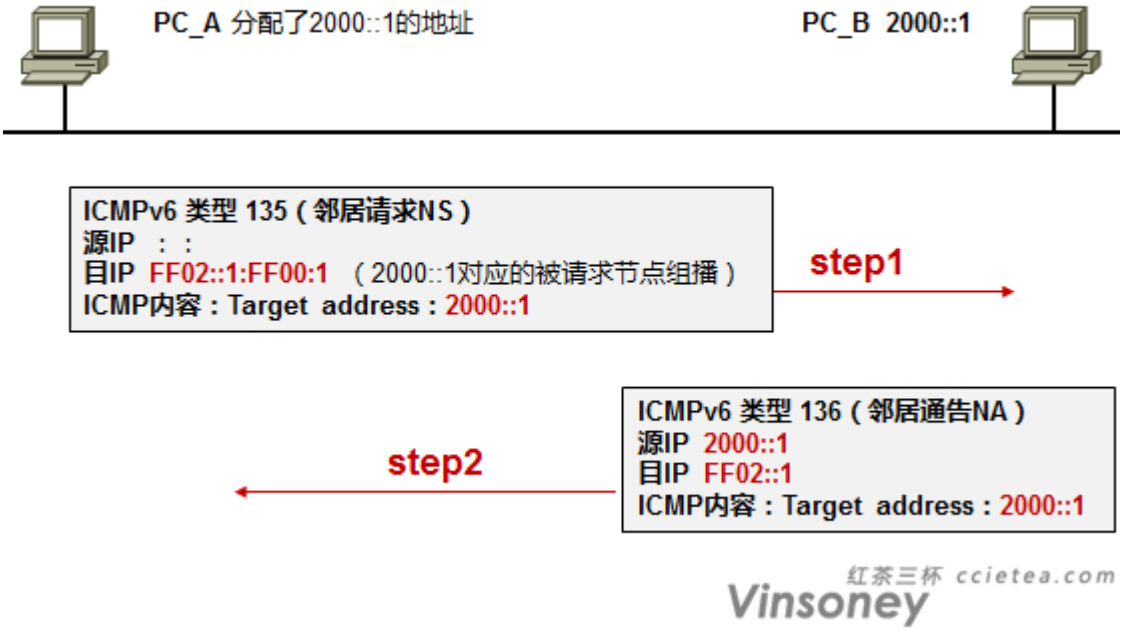
1. **机制概述**

   无状态配置，和节点启动时的一个 NDP 机制。用于保证节点准备启用的 IPv6 单播地址在链路上的唯一性。这个机制使用 NS 消息（ICMP 135），使用源地址（ ::）、目的地址为获取到的 v6 地址对应的被请求节点组播地址的 NS 报文来完成这个任务。

2. **机制原理**

   一个地址在通过重复地址检测（DAD）之前称为“tentative 地址”，试验地址。接口还暂时不能使用这个试验地址进行正常单播通讯，但是会加入和 tentative 地址所对应的 Solicited-Node 组播组。

   重复地址检测 :节点向一个自己将使用的 tentative 地址所对应的 Solicited-Node 组播地址发送一个 NS，如果收到某个其他站点回应的 NA，就证明该地址已被网络上使用，节点将不能使用该 tentative 地址通讯。



如果 1S 后没有检测到冲突，A 就会发送 non-solicited advertisement（一个 NA 消息），宣告大家我将正式使用这个 IPv6 地址。

3. 调整 DAD

- **Ipv6 nd dad attempts x**

  默认情况下，CISCO 路由器启用 DAD，在确定一个地址的唯一性之前，在本地链路上发送 NS 消息的个数为 1。使用上述命令，能修改 NS 消息的个数。为 0 则关闭 DAD。

5.4.5 前缀重新编址

前缀重新编址允许从以前的网络平稳过渡到新的前缀，站点内节点使用无状态自动配置（或者其他重编址方法，但是不如无状态自动配置的方法透明），这样可以使得在前缀重新编址过程对站点内的节点完成透明，也就是说，站内节点完全“不知情“或”无感知 “，部署的动作在路由器上完成，切换过程平滑。

路由器接口配置新、老两个前缀，并且都进行公告，老前缀的生存期较短，这样站内的节点可以同时使用两个地址，当老前缀失效后，仅剩下新的前缀被使用，即可实现切换。

首选 ,站点中所有路由器继续公告当前的前缀(老前缀),但是有效和首选生存期被减小到接近于 0 的一个值，然后，路由器开始在本地链路公告新的前缀，因此每个本地链路上至少有两个前缀存在。当旧的前缀完全被废止时（生存期已过），路由器公告消息仅包括新前缀。

配置前缀重新编制

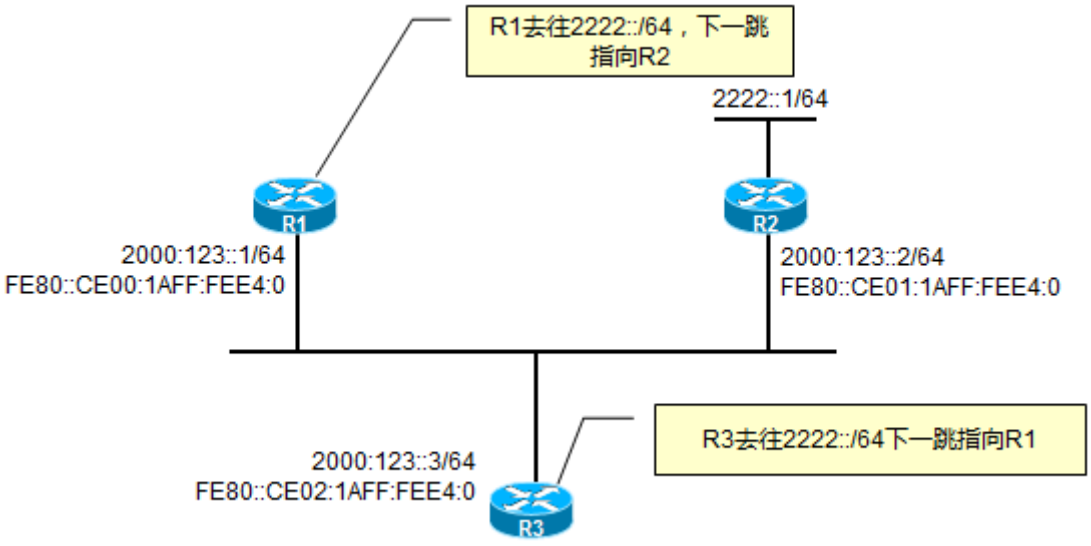
```
Ipv6 nd prefix 2001:0001::/64  43200  0                    // 老前缀
Ipv6 nd prefix 2001:0002::/64  43200  43200                // 新前缀
或者
Ipv6 nd prefix 2001:0001::/64  at Jul 31 2012 23:59  Jul 20 2012 23:59    // 老前缀
Ipv6 nd prefix 2001:0002::/64  43200  43200                // 新前缀
```

5.4.6 路由器重定向

路由器使用 ICMPv6 重定向消息（ICMP TYPE 137）通知链路上的节点，在链路上存在一个更好的前转数据包的路由器。接收到这个 ICMPv6 重定向消息的节点可以根据重定向消息中新的路由器地址修改它的本地路由选择表。基本上从机制上来将跟 IPv4 的 ICMP 重定向没啥两样。

在 IPv6 规范中，**不推荐使用可聚合全球单播或本地站点地址作为下一跳地址，如果这样做，ICMPv6 重定向消息就不会工作**。因此使用 Linklocal 地址作为下一跳，在某些场合可能更为推荐，毕竟 linklocal 地址稳定且长久不变。在配置 linklocal 地址作为下一跳 IP 时，必须关联路由器上相应的接口。

接下去做一个简单的测试：



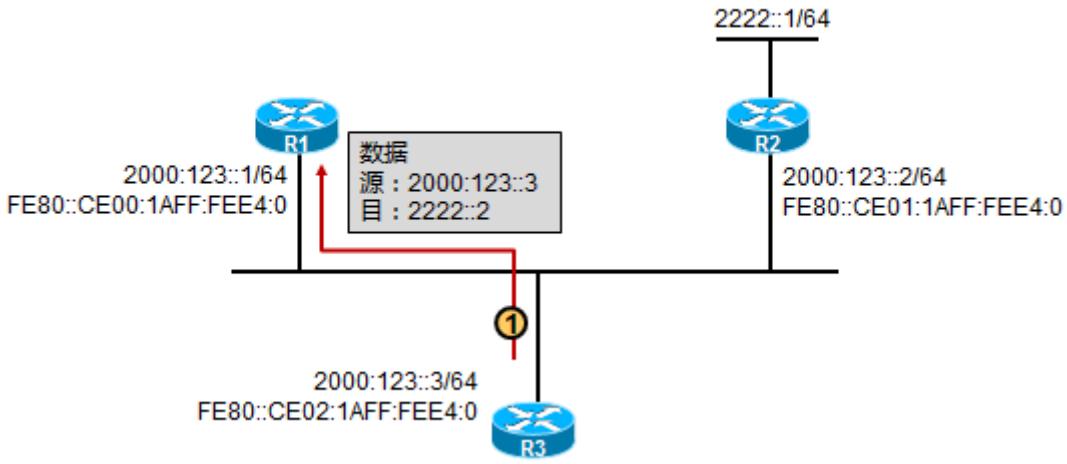
实验环境如上，配置如下：

R3 的关键配置：

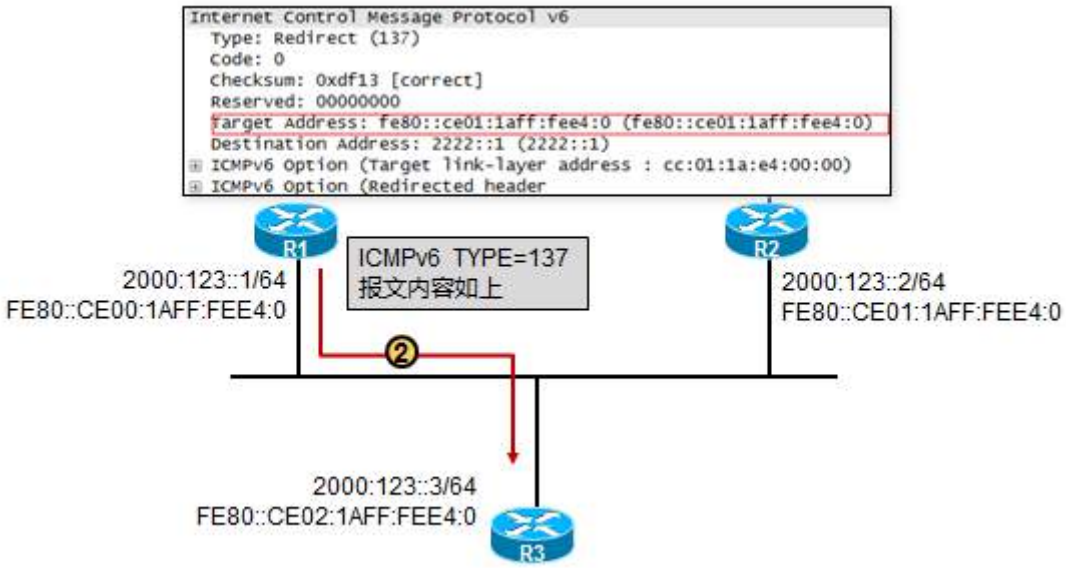
```
ipv6 route 2222::/64 FastEthernet0/0 FE80::CE00:1AFF:FEE4:0
```

R1 的关键配置：

```
ipv6 route 2222::/64 FastEthernet0/0 FE80::CE01:1AFF:FEE4:0
```



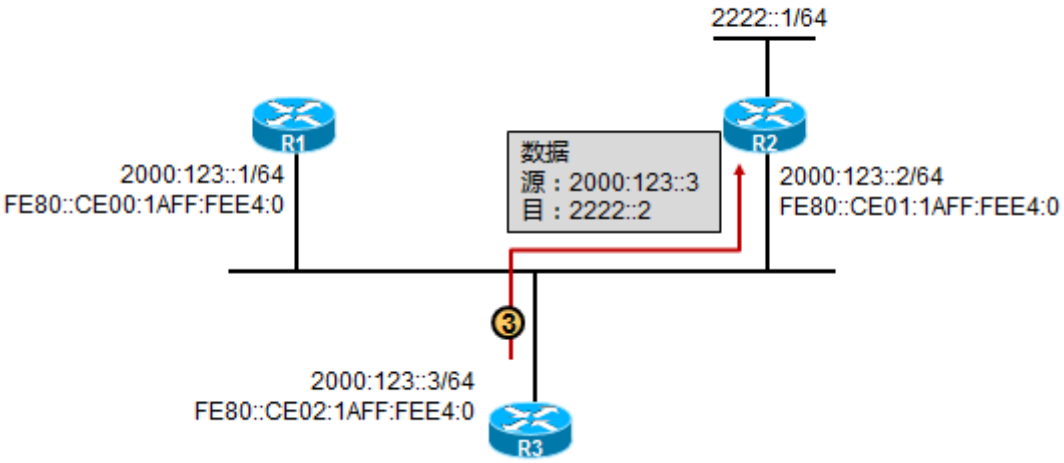
接下去，我们在 R3 上去 ping 2222::2，这个动作将使得 R3 发送一个 ICMPv6 报文，源为 2000:123::3，目的为 2222::2，报文将被发送到 R1，当 R1 接收后，他发现，这个数据包将从自己收到该报文的接口再转发出去，而下一跳 IP 与该接口在同一个网段，证明这个下一跳比自己更优（距离目的地更近）。



因此，R1 将发送一个 ICMPv6 Type=137 的重定向报文给 R3，以便告知 R3，要去往目的地 2222::3，有人比我更优，而这个 ICMPv6 消息中，包含的关键信息，就是 R2 的 Linklocal address。报文如下：

```
Internet Protocol Version 6, Src: fe80::ce00:1aff:fee4:0 (fe80::ce00:1aff:f
Internet Control Message Protocol v6
  Type: Redirect (137)
  Code: 0
  Checksum: 0xdf13 [correct]
  Reserved: 00000000
  Target Address: fe80::ce01:1aff:fee4:0 (fe80::ce01:1aff:fee4:0)
  Destination Address: 2222::1 (2222::1)
⊟ ICMPv6 Option (Target link-layer address : cc:01:1a:e4:00:00)
    Type: Target link-layer address (2)
    Length: 1 (8 bytes)
    Link-layer address: cc:01:1a:e4:00:00 (cc:01:1a:e4:00:00)
⊞ ICMPv6 Option (Redirected header
```

上面就是R1发送给R3的ICMPv6的type137报文 ,我们关键看target address ,这个就是R2的linklocal address，也就是 R1 想告知给 R3 的、距离目标 2222::1 比自己更近的下一跳。



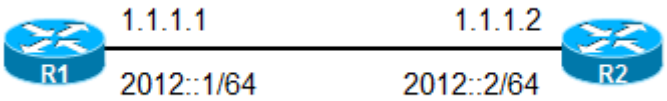
R3 收到这个 ICMPv6 重定向消息后，后续的数据包将直接发给 R2，就不用再绕过 R1 了。

6 IPv6 应用

6.1 域名系统

IPv4 中 A 记录用来将主机名称映射到一个 V4 地址，AAAA 资源记录将主机名映射到一个 IPv6 地址。

我们可以做一个测试：



R1 及 R2 为双栈路由器，R1 的配置如下（省去接口 IP 的配置）：

```
ipv6 unicast-routing
ip name-server 1.1.1.2
ip name-server 2012::2
ip domain-lookup
```

现在做一个测试，在 R1 上 ping www.ccietea.com，我们会发现：

Source	Destination	Protocol	Length	Info
1.1.1.1	1.1.1.2	DNS	72	Standard query AAAA www.sina.com
2012::1	2012::2	DNS	92	Standard query AAAA www.sina.com
2012::2	2012::1	ICMPv6	140	Destination Unreachable (Port unreachable)
1.1.1.1	1.1.1.2	DNS	72	Standard query A www.sina.com
1.1.1.1	1.1.1.2	DNS	72	Standard query A www.sina.com
2012::1	2012::2	DNS	92	Standard query A www.sina.com

R1 首先会通过 IPv4 网络去查找 AAA 记录，随后又会通过 IPv6 网络去 AAAA 记录，发现都没有回应，那么又用 IPv4 网络去查找 A 记录，没回应，又用 IPv6 网络查找了一次 A 记录。

因此对于双栈的情况下，PC DNS 解析的时候，一般会先请求 AAAA 记录。

在 CISCO IOS 路由器上，使用如下的方式配置一条主机名到 IPv6 地址的映射：

```
Ipv6 host xxx 2001::1/64
```

6.2 ACL

Ipv6 access-list x	进入 acl 配置模式
Ipv6 traffic-filter	在接口下应用

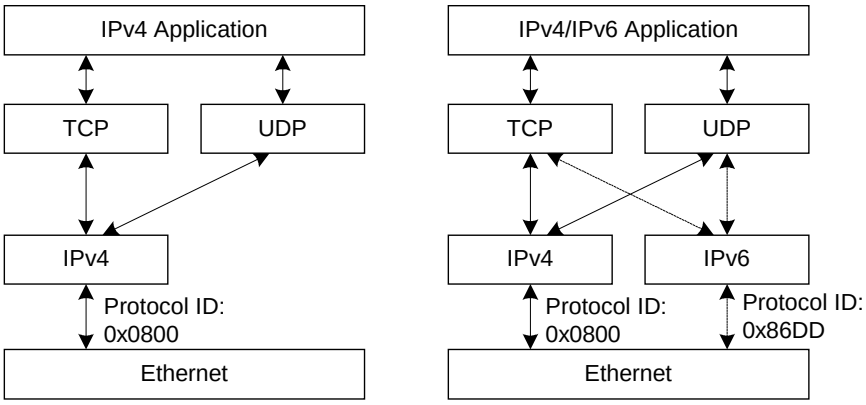
7 IPv6 过渡技术

7.1 Foundation

由于现今无论是广域网如 Internet，还是全球范围内的各种 LAN，都是以 IPv4 居多。要切换到 IPv6 不可能一气呵成，这是一个综合政治、经济、政策、方法、环境等等各种因素的大问题。IPv4 切换到 V6 需要一个相当长的时间。那么在这个过程中，就势必出现 V4 和 V6 共存的网络环境。接下去就研究研究 V4 到 V6 的平稳过渡技术。

1. 双栈（dual stack）

网络中的主机、路由器或服务器等设备如果支持双栈（IPv4 及 IPv6 协议栈），那么可以同时使用 V4 和 V6 的协议栈。在双栈设备上，上层应用会优先选择 IPv6 协议栈，而不是 IPv4。 比如，一个同时支持 v4 和 v6 的应用请求地址，会先请求 AAAA 记录，如果没有，则在请求 A 记录。



2. 隧道 tunnel

用于在现有网络（v4）中传输不兼容的协议（v6）或者特殊的数据，

- 手工隧道技术：手工 IPv6 over IP 隧道、GRE 隧道
- 自动隧道技术：6to4 隧道、IPv4 兼容 IPv6 自动隧道、ISATAP 隧道
- 6PE：6PE 技术依赖于 BGP，BGP 的 Peer 是需要手工指定的，可以算是一种半自动隧道技术。

3. v6 v4 协议转换

NAT-PT（Network Address Translation-Protocol Translation）

7.2 双栈（Dual Stack）

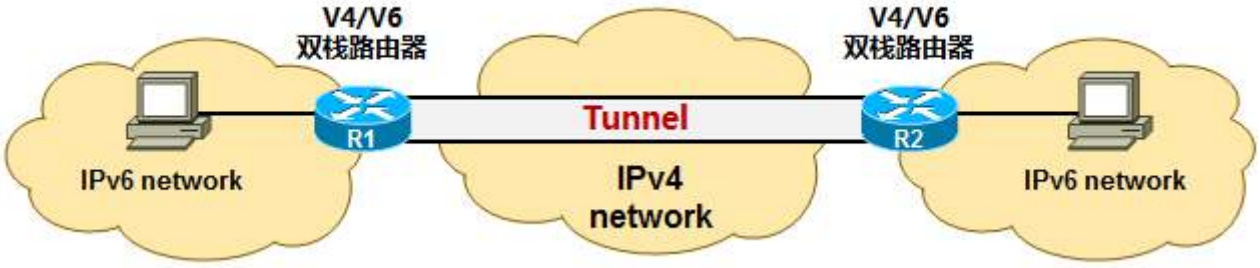
- 节点有 IPv4 及 IPv6 两个协议栈
- 缺点：每台设备都需要配置两种协议，需要占用资源，设备需要存储两个路由表（如果是路由器），需要独立处理每种协议。

```
router(config)#  ipv6 unicast-routing
router(config)#  interface fast0/0
router(config-if)#  ipv6 enable
router(config-if)#  ip address 192.168.1.254 255.255.255.0
router(config-if)#  ipv6 address 2001::0001::FFFE/64
```

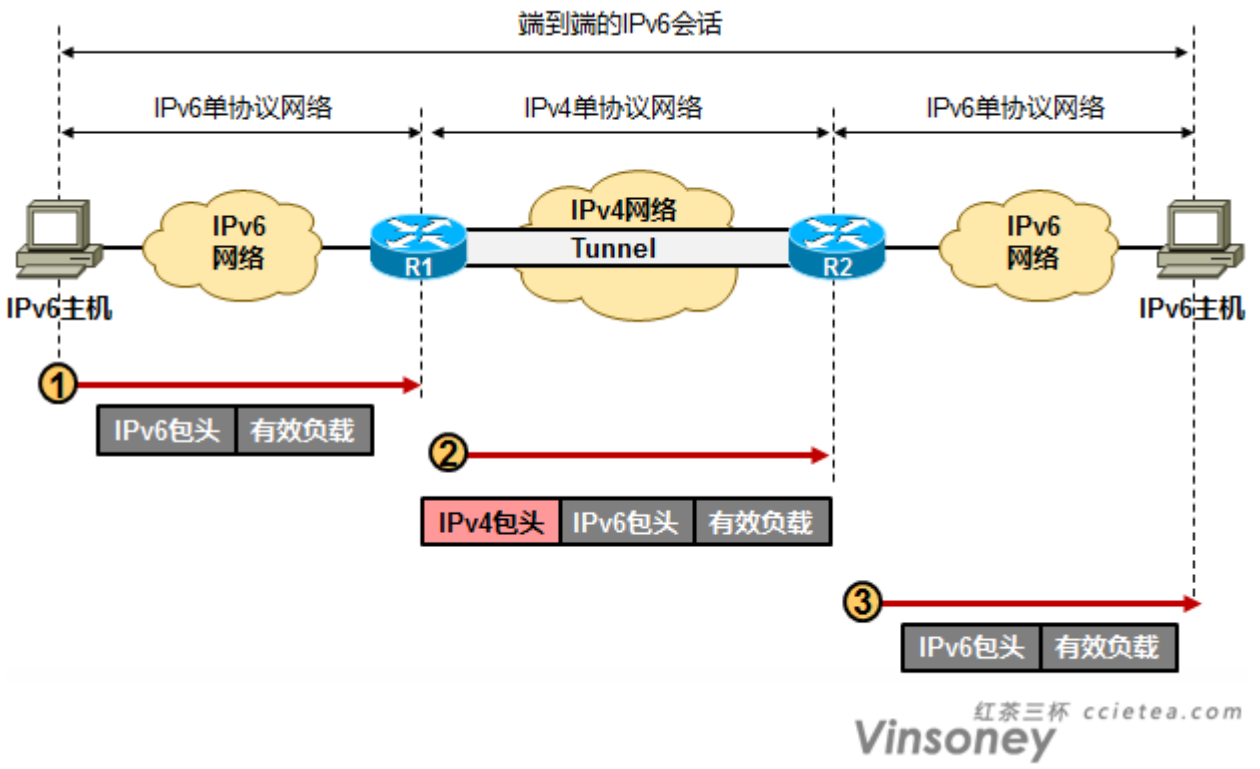
7.3 隧道（Tunnel）

7.3.1 隧道机制概述

隧道技术一般用于在现有网络中传输不兼容的协议或特殊的数据。在因特网上无处不在的现有 IPv4 基础设施中，使用隧道技术可以使得 IPv6 孤岛得以连接。当然，相对于任何过渡和并存的策略（如隧道技术），我们还是首选由纯 IPv6 连接组成的网络、链路和基础设施。实际上，只有当在网络、连接和基础设施中不可能获得纯 IPv6 连接性时，IPv4 基础设施上的 IPv6 隧道机制才被认为是一种可选择的方法。



接下来简单的了解一下最基本的隧道技术来实现在 IPv4 网络中连接 IPv6 孤岛：



- 左边的 IPv6 主机欲与右边的 IPv6 主机通信，发出原始的 IPv6 数据包
- 这些数据包到达 R1 后，R1 路由器支持 IPv4 及 IPv6 也就是双栈路由器，R1 与 R2 建立了一个隧道以连接分别挂在 R1、R2 两端的两个 IPv6 孤岛。PC 发出来的原始 IPv6 报文被 R1 添加了一个 IPv4 的隧道包头，形成了一个“外表看似 IPv4 报文的”报文。此后，这个报文在 IPv4 网络中被路由，最终到达 R2。R2 收到这个“隧道报文”后，将报文的 IPv4 头部去除，然后还原成原始的 IPv6 报文，再转发给 IPv6 网络。

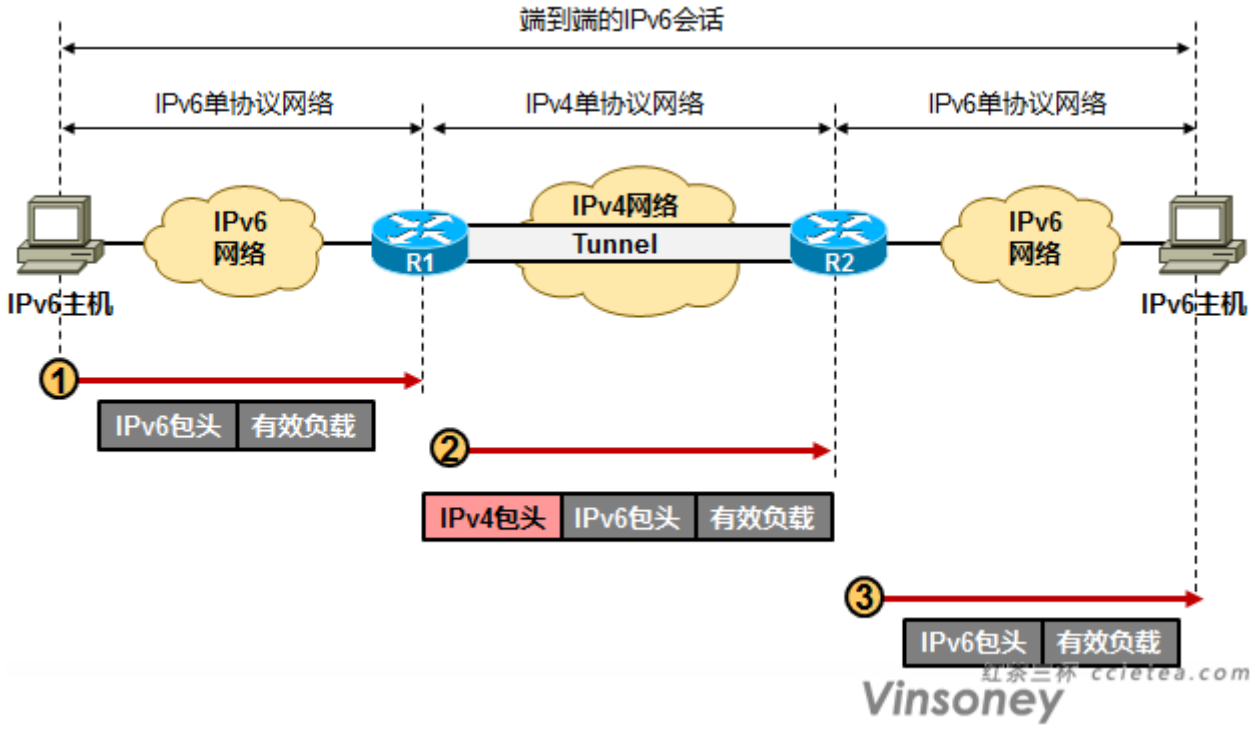
  【插嘴】　　在 R1 所添加的这个 IPv4 包头中 ,protocol 字段用于指示上层封装的是 IPv6 的报文 ,protocol 字段值为 41：

```
Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
  Version: 4
  Header length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 120
  Identification: 0x0033 (51)
⊞ Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: IPv6 (0x29)
⊞ Header checksum: 0x8424 [correct]
  Source: 10.1.12.1 (10.1.12.1)
  Destination: 10.1.23.3 (10.1.23.3)
Internet Protocol Version 6                IPv6头
Internet Control Message Protocol v6       有效负载，这个是直接ping产生数据
```

- R2 将 R1 添加的 IPv4 包头去除，还原成那个原始的 IPv6 数据，然后转发给右边的 IPv6 PC。

7.3.2 手工 IPv6 over IP 隧道

1. 机制介绍



R1、R2 为双栈路由器，同时连接到 IPv6 及 IPv4 网络。

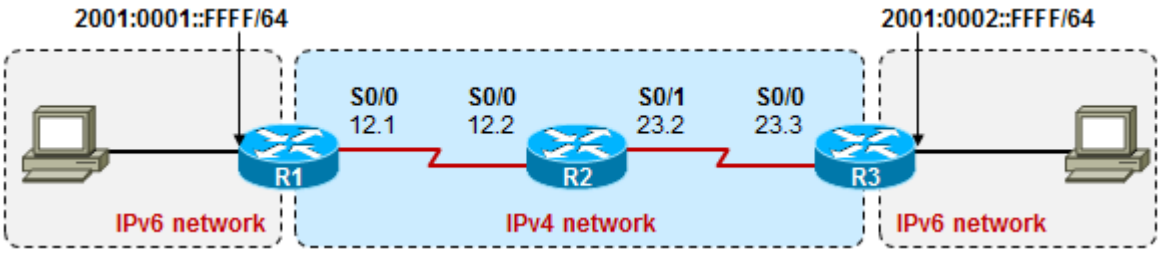
在 R1、R2 下分别挂着两个 IPv6 的孤岛，现在，通过在 R1 及 R2 间建立一个 IPv6 over IP 的 tunnel，使得两个 IPv6 的孤岛得以穿越 IPv4 的因特网而进行互联。原始的 IPv6 报文前面，被添加上一个隧道的 IPv4

头，从而形成一个外层的 IPv4 报文的报文，这个报文在 IPv4 网络中被路由。

关于配置，这里贴个简单的示例：

```
Ipv6 unicast-routing
Interface serial0/0
   ip address 10.1.12.1 255.255.255.0
Interface fa1/0
   Ipv6 enable
   Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
   Ipv6 enable
   tunnel mode ipv6ip
   tunnel source serial 0/0
   tunnel destination 10.1.23.3
ip route 0.0.0.0 0.0.0.0 10.1.12.2        !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0   tunnel 0                !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

2. 基础实验



R1 的配置如下：

```
Ipv6 unicast-routing
Interface serial0/0
   ip address 10.1.12.1 255.255.255.0
Interface fa1/0
   Ipv6 enable
   Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
   Ipv6 enable
   tunnel mode ipv6ip
   tunnel source serial 0/0
   tunnel destination 10.1.23.3
```

```
ip route 0.0.0.0 0.0.0.0 10.1.12.2          !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0    tunnel 0                 !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

R3 的配置类似；R2 只需配置接口 IP 即可；PC1 及 PC2 各自配好 IPv6 地址，指网关即可。

Tunnel 接口未必一定需要 ipv6 address，当然，也可以配置 v6 地址，这不影响穿越路由器的流量

配置完成后，PC1 即可 ping 通 PC2，抓包如下：

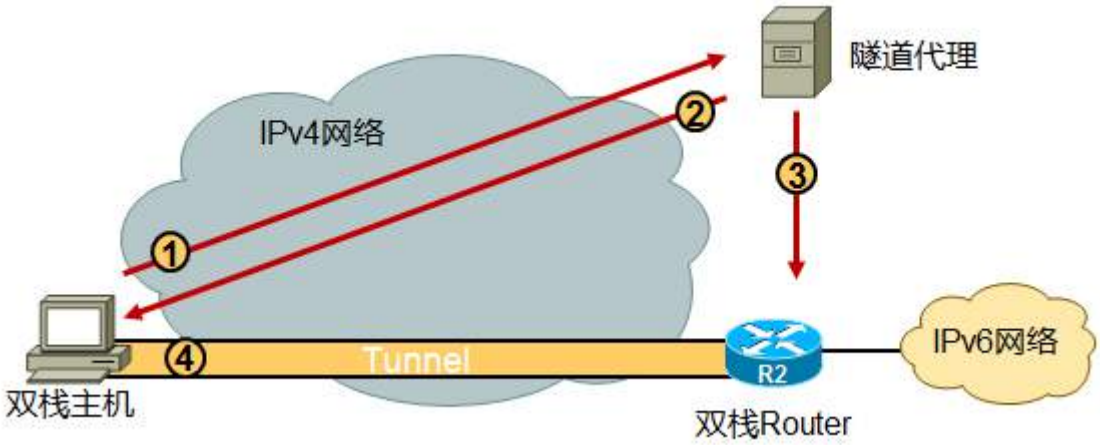
```
⊞ Cisco HDLC
⊟ Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 120
    Identification: 0x0019 (25)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: IPv6 (0x29)
  ⊞ Header checksum: 0x843e [correct]
    Source: 10.1.12.1 (10.1.12.1)
    Destination: 10.1.23.3 (10.1.23.3)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source: 2001:1::1 (2001:1::1)
    Destination: 2001:2::1 (2001:2::1)
⊞ Internet Control Message Protocol v6
```

无论是 GRE 隧道 ,或是 IPv6 over IP 隧道 ,在隧道两头都可以跑动态路由协议。例如在上面 ,可以为 tunnel 两端接口都配置上 ipv6 地址，然后激活动态路由协议即可。从封装效率来说，还是 IPv6 over IP 会比 GRE 更好，毕竟少了一层 GRE 的封装，但是当跑动态路由协议的时候，在某些场合下，GRE 封装的通用性或者兼容性好比 IPv6 over IP 的封装要好一些，例如跑 ISIS 协议，用 IPv6 over IP 隧道就会有点问题。

3. 隧道代理

IPv6 over IP 手工隧道，由于需要在隧道的两端，也就是两台双栈路由器上配置本地隧道、对端隧道的地址，因此扩展性比较差。为了方便在 IPv4 网络上手工隧道的部署，IETF 定义了隧道代理机制。

如 RFC3053 所定义，隧道代理是一个外部系统，而不是路由器。它在 IPv4 网络中作为服务器，并接受双栈节点的隧道请求。



如上图所示：

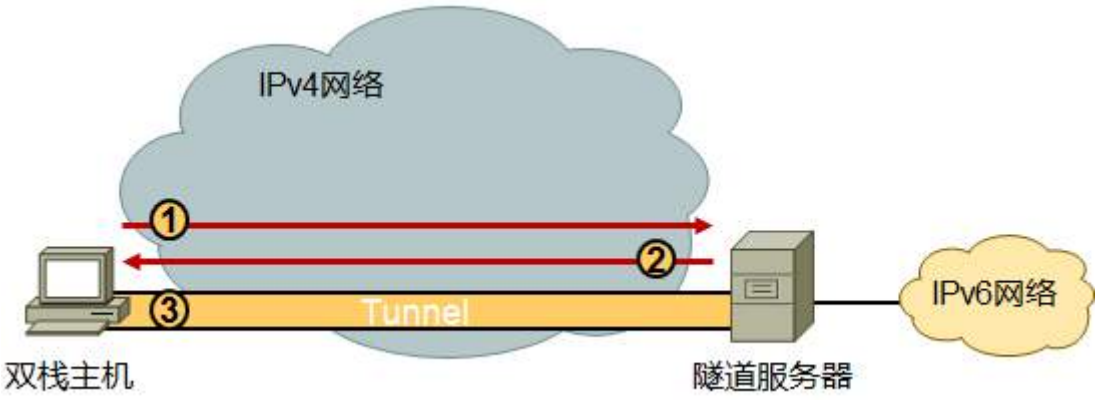
1) 双栈主机通过 IPv4 使用 HTTP 访问隧道代理。这个双栈主机可能是个终端用户，它填写一个网页。
2) 双栈主机通过 HTTP 从隧道代理处获得 IPv4 和 IPv6 地址 然后终端用户使用这些地址配置并启用隧道，其中获得到的 IPv4 地址就是隧道的 IPv4 地址。
3) 与此同时，隧道代理自动地在一个连接到 IPv6 网络的双栈路由器上应用配置隧道的远端配置，一旦配置在双栈主机和双栈路由器上得到了应用，隧道就会被正确的建立起来
4) 端到端的 IPv6 隧道建立起来之后，IPv6 流量就可以进行通讯了。

一般来说，隧道代理和双栈 Router 同属一个管理方，例如同一家公司。因为隧道代理要有双栈 Router 的配置权限。

另外，CISCO IOS 不支持隧道代理。但是其实现在公网上有许多免费的隧道代理服务器。

- **隧道服务器**

隧道服务器是隧道代理的简化模型。将隧道代理和双栈 Router 进行了整合。



1) IPv4 网络上的双栈主机首先通过 IPv4 使用 HTTP 访问隧道服务器，终端用户通过填写网页并从隧道服务器获得 IPv4 及 IPv6 地址
2) 终端用户通过双栈主机获得的地址配置隧道，而隧道服务器在本地应用隧道的远端配置
3) 最终，隧道建立起来

CISCO IOS 还不支持隧道服务器。

7.3.3 6to4 自动隧道

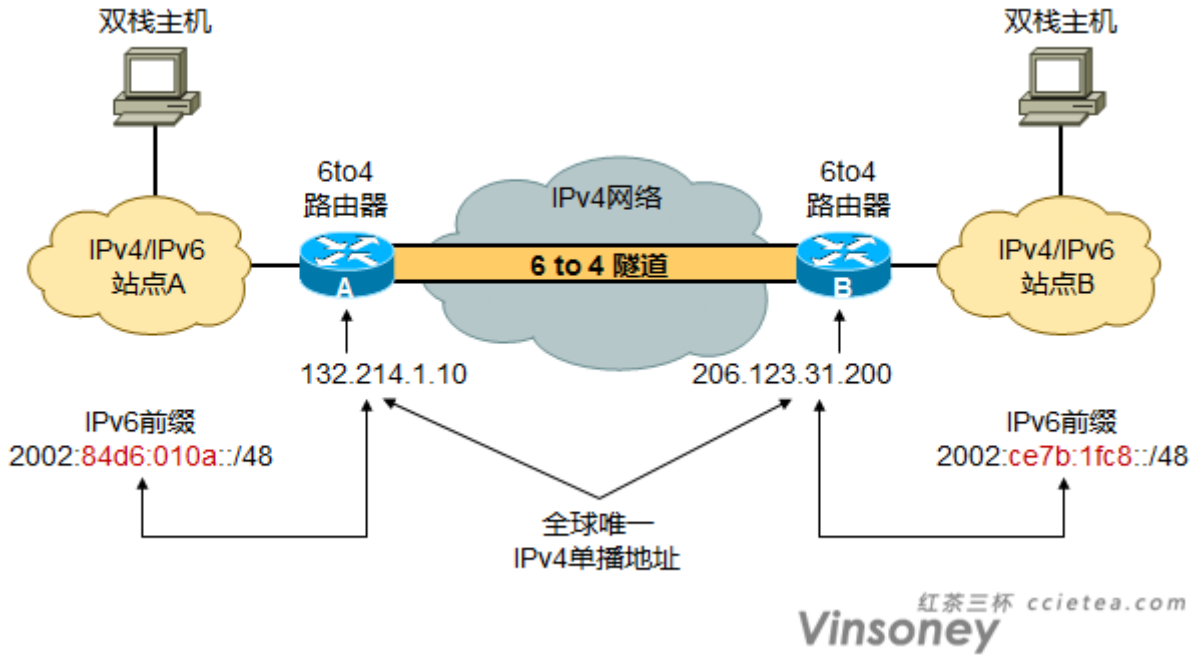
1. 机制介绍

我们来回顾一下 IPv6 over IP 手工隧道（也叫做配置隧道）的配置：

```
Ipv6 unicast-routing
Interface serial0/0
  ip address 10.1.12.1 255.255.255.0
Interface fa1/0
  Ipv6 enable
  Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
  Ipv6 enable
  tunnel mode ipv6ip
  tunnel source serial 0/0
  tunnel destination 10.1.23.3
ip route 0.0.0.0 0.0.0.0 10.1.12.2
Ipv6 route ::/0   tunnel 0
```

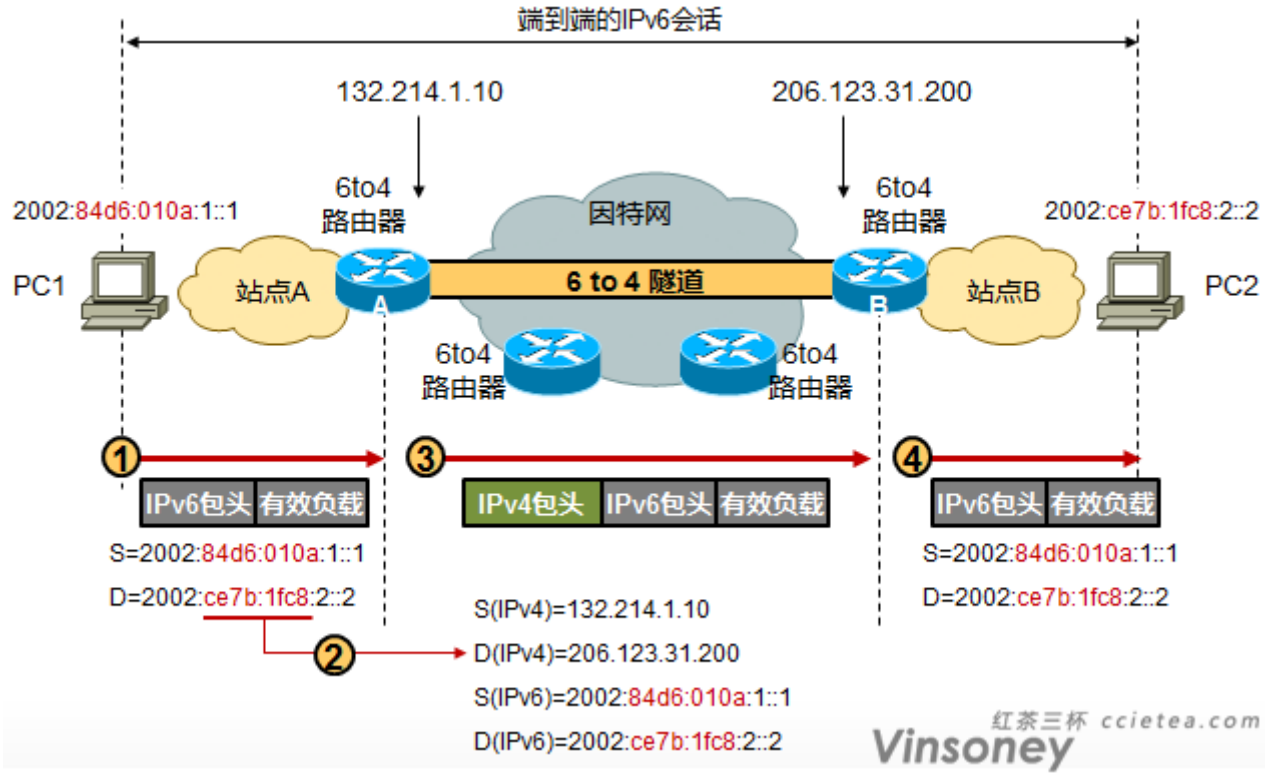
我们注意到，对于 tunnel 的配置，需要制定 tunnel 的 destination，这样一来，如果隧道比较多，这种配置就变的非常笨重了，而且可扩展性太差。因此，我们又有另外一个机制：6to4 自动隧道。

2. 机制详解



看上图：路由器 A 及 B 分别是两个站点。两边都申请到一个 IPv4 公网地址，使用这个公网 IPv4 地址进行映射得到全球唯一的 6to4 IPv6 地址，这个 6to4 IPv6 地址/48 位，空间非常大，用于站点内的 IPv6 用户。如此一来，当站点 A 内的 IPv6 用户访问站点 B 的 IPv6 用户时，IPv6 数据包发送到 A，那么 A 根据 IPv6 数据报头中的目的 IPv6 地址，得到对应的 IPv4 全局 IP，那么就将原始的 IPv6 数据包进行封装，套上 6to4 隧道的 IPv4 的头，然后将这个数据包放入 IPv4 网络进行路由，直到到达 B。

接下去，我们来看一下数据包交互的详细过程：



A 拿到的 IPv4 公网地址是 132.214.1.10，而 B 是 206.123.31.200。

A 和 B 分别使用这个 IPv4 公网地址来映射得到 6to4 地址空间，6to4 地址空间使用 2002::/16，而 6to4 地址的形成如下：

2002：**IPv4 地址**：子网 ID：：接口 ID

上面的 IPv4 地址，就是 A 或 B 拿到的那个公网 IPv4 地址，注意，这个公网 IP 非常重要，此公网 IP 的变化，有可能导致 IPv6 站点内 IPv6 不得不重新编址。而且，当然，这个 IPv4 地址不能使用私有 IPv4 地址。

得到 6to4 IPv6 地址空间后，这个地址空间是 16+32=48bits，也就是/48 的，因此空间非常大，你可以进一步的进行子网的划分，这个 IPv6 的地址空间就将用于站点内网。

1) 上面的例子，A 拿到的 IPv4 公网地址：132.214.1.10，映射得到 2002:**84d6:010a**::/48，这个地址空间的一部分最终被用在了 PC1 上。现在，PC1 要给 PC2 发送一个 IPv6 的数据。这个数据的源 IP 是 PC1 的 IPv6 全局唯一地址 2002:84d6:010a:1::1，目的 IP 为 PC2 的 IPv6 全局唯一地址 2002:ce7b:1fc8:2::2，这个 IPv6 报文被送到了 PC1 的默认网关也就是路由器 A。
2) A 是一台 6to4 路由器，它会查看这个 IPv6 数据包的包头里的目的 IPv6 地址，它发现这个 IPv6 目的地地址是一个 6to4 地址，因此，它计算得出这个 6to4 IPv6 地址对应的 IPv4 公网 IP，然后，为这个原始的 IPv6 数据进行封装，加上一个新的 IPv4 的头，而这个 IPv4 的头源地址是 132.214.1.10，目的地址刚才通过计算得出的那个 IPv4 公网地址，正是 206.123.31.200。然后，这个新的 IPv4 数据包进入因特网，最终被传送到 B。
3) B 将报文的 IPv4 头除去，得到原始的 IPv6 数据，转发给 PC2。

3. 机制补充

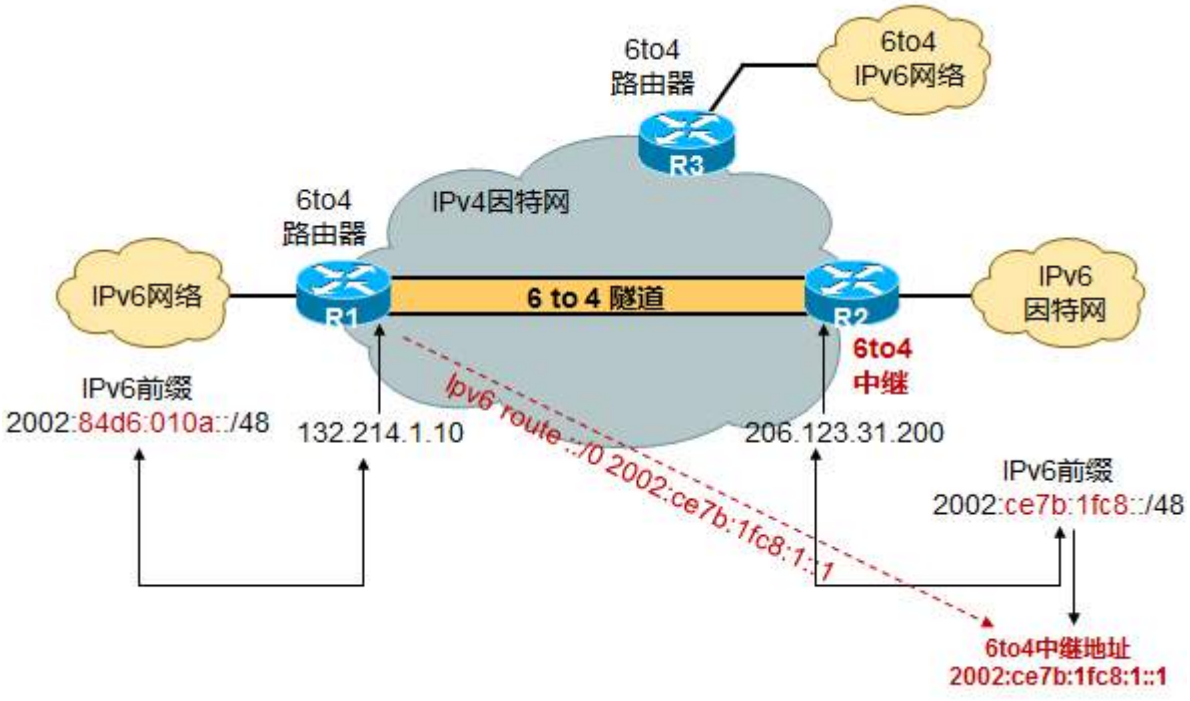
上面提到的 **IPv4 地址**是为 IPv6 孤岛申请的公有地址，在 IPv6/IPv4 边界路由器（6to4 路由器）上配置该 v4 地址，可以直接为 6to4 路由器连接公网接口的 IP，也可以是 6to4 路由器 Loopback 接口的 IP，这个 IP 同时是 6to4 tunnel 的 source 或 destination。

隧道的源 IPv4 地址手工指定（就是前面提到的 **IPv4 地址**），隧道的目标 IPv4 地址不需要显式配置（路由器动态建立隧道），根据通过隧道转发的 IPv6 报文决定，如果 IPv6 报文的目的地是 6to4 地址，则从目的 IP 中提取 IPv4 地址，如果目的地不是 6to4 地址，那么就需要使用 6to4 中继。

所以 6to4 隧道是自动完成查找和建立的，无需手工配置 tunnel 的 destination，相比配置隧道，可扩展性要更高。

4. 6to4 中继

之前我们所讲的，都是两个 IPv6 孤岛都使用的是 6to4 的 IPv6 地址空间。6to4 边界路由器在收到发送往 6to4 网络的 IPv6 数据，能够从目的 IPv6 地址中得到对应的公网 IPv4 地址从而进行转发。但是，如果 IPv6 孤岛使用的不是 6to4 地址空间，而是像 2001::/16 这样的常规可聚合全球单播地址呢？这就需要使用 6to4 中继了。



我们看上面的图，其实利用 6to4 自动隧道技术，IPv6 孤岛既可以使用 6to4 IPv6 地址空间，也可以使用普通的 IPv6 地址空间，上图中，R1 上挂的 IPv6 孤岛使用 6to4 地址空间，R3 上的 IPv6 孤岛也是 6to4 空间，而 R2 这台因特网上的 6to4 路由器同时也连接到了 IPv6 的因特网和 IPv4 因特网，因此他具有 IPv6 公网的路

由。所以，R1、R3 下的两个 6to4 孤岛的互相访问，解决方案我们前面已经讲过了，但是 R1 或 R3 如果需要访问非 6to4 的 IPv6 网络呢？就需要 R2 了，R2 此时是一台 **6to4 中继路由器**。此刻，R2 通过其自身的一个 IPv4 公网 IP，得到一个 6to4 的 IPv6 全球可聚合单播地址，这个 IPv6 地址用于响应其他 6to4 路由器的隧道建立请求。

我们拿 R1 举例，在 R1 上，配置可能如下：

```
Interface loopback0
  Ip address 132.214.1.10 255.255.255.0
Interface fast0/0
  Ipv6 enable
  Ipv6 address 2002:84d6:010a:0001:/64 eui-64        !!这是 IPv6 站点内网的网关
Interface tunnel1
  Ipv6 enable
  Ipv6 unnumbered fast 0/0
  tunnel source loopback0
  tunnel mode ipv6ip 6to4
!
Ipv6 route 2002::/16 tunnel1                !!当本地访问远端的 6to4 网络时，走 tunnel
Ipv6 route ::/0 2002:ce7b:1fc8:1::1         !!当访问的远端 IPv6 网络使用的地址空间不是 6to4 时 ,走默认路由，这时候会找下一跳。
```

经过上述配置，当 R1 所连接的 IPv6 岛屿中用户要访问其他 6to4 网络时，那么 R1 就从目的 6to4 IPv6 地址中得到 IPv4 公网 IP，然后将隧道 IPv4 数据包转发到目的地的 6to4 路由器。而如果 R1 所连接的 IPv6 岛屿中用户要访问非 6to4 的 IPv4 网络时，就需要求助于 6to4 中继，也就是 R2，那么怎么找到 R2 呢？

首先 R2 自己通过自己的 IPv4 公网地址映射得到一个 6to4 地址空间，R2 给自己分配了这个 6to4 IPv6 地址空间中的一个地址（2002:ce7b:1fc8:1::1），以便其他 6to4 路由器能够找到自己。而对于 R1 这样的 6to4 路由器，只需添加一条默认路由：Ipv6 route ::/0 2002:ce7b:1fc8:1::1 即可，这样一来 R1 所连接的 IPv6 岛屿中用户要访问非 6to4 的 IPv4 网络时，数据被这条默认路由所匹配，下一跳为 2002:ce7b:1fc8:1::1，这是一个 6to4 地址，于是 R1 首先根据这个地址得到其对应的 IPv4 地址：206.123.31.200，然后将对 2002:ce7b:1fc8:1::1 再进行路由表的递归查找，发现要从 tunnel1 接口扔出去，于是给原始的 IPv6 数据压上新的 IPv4 隧道头，源地址为隧道的 IPv4 源地址 132.214.1.10，目的地址为 206.123.31.200。这个数据包就这么到了 R2。

在因特网上，像 R2 这类 6to4 中继路由器还是有不少的。有一些公共的 6to4 中继。例如：6to4.ipv6.microsoft.com 等。

有一点要注意的是，如果这个 6to4 中继路由器距离你的网络“很远 “，那么这可能会造成你和 IPv6 因特网之间的 IPv6 流量性能低下。为了帮助一个 6to4 站点在因特网上找到可用的 6to4 中继并且给 6to4 机制更多可扩展性，RFC3068 引入了一个任意播前缀。这样 6to4 数据包可以被自动路由到 IPv4 因特网上最近的 6to4 中继。IANA 分配了 6to4 中继任意播前缀 192.88.99.0/24 专门用于自动路由 6to4 数据包到最近的 6to4 中继。在这个任意播前缀中，所定义的到达最近的 6to4 中继的 IPv4 地址若是 192.88.99.1，那么这个地址的 IPv6 表示就是 2002.c058.6301::，在你的 6to4 配置中，可以使用 ipv6 roiute ::/0 2002.c058.6301::来配置默认路由。

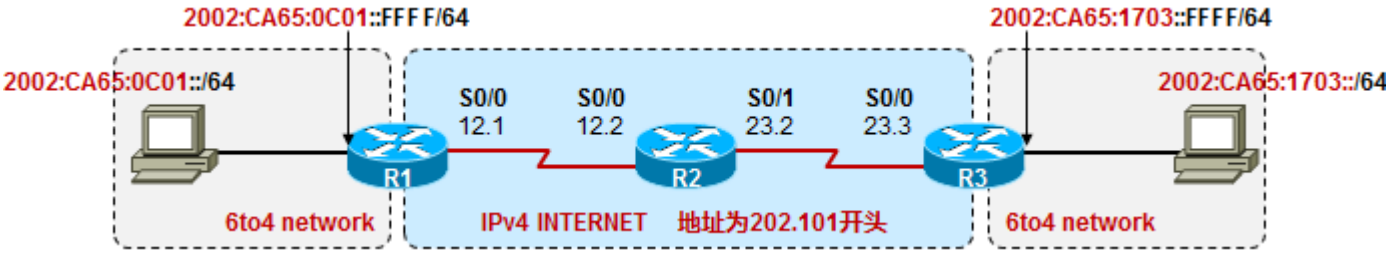
你可以配置一台 CISCO 路由器作为一个具有任意播 IPv4 前缀的 6to4 中继：

```
Interface tunnel1
   Ipv6 enable
   Ipv6 unnmbered fast0/0
   tunnel source fast0/0
   ipv6 address 2002.c058.6301::/128 anycast
   tunnel mode ipv6ip 6to4
interface fast0/0
   ipv6 enable
   ip address 132.214.1.10 255.255.255.0
   ip address 192.88.99.1 255.255.255.0 secondary
   ipv6 address 2002:84d6:010a:0001::/64 eui-64
!
Ipv6 route 2002::/16 tunnel1
```

注意，这里的 IPv4 前缀 192.88.99.0/24 应该被发布到 IPv4 因特网上的路由选择协议中。否则这个使用任意播前缀的 6to4 中继在 IPv4 因特网中是不可达的。

5. 6to4 自动隧道的配置

【实验 1】两边内网都是用 6to4 地址（双方都是 6to4 网络）



R1 路由器申请的 ipv4 公网地址为 202.101.12.1；R3 申请的公网地址为 202.101.23.3

有了公网地址，我们也就有了根据公网地址计算得来的 6to4 的地址空间：

2002:IPV4 地址映射:子网 ID::/48　　前面 48 位是 2002+ipv4 的公网地址，后面 64 位是接口 ID

例如 R1 公网地址为：202.101.12.1，那么对应的 6to4 地址空间就是：2002:CA65:0C01::/48

这个地址空间加上子网 ID ,就可以分配给内网用户了 ,这个实验我们内网用户采用无状态自动配置获取地址。

这时候，如果 R1 下有用户要访问 R3 下的 v6 网络，那么目的地址肯定是 R3 的 6to4 地址空间，数据到了 R1 后，R2 一看，发现是 2002 的 ipv6 地址，于是它就直接去读 2002 后面的 32 位，将其转换成 ipv4 地址，并作为 6to4 的 tunnel destination。

R1 的配置如下：

```
ipv6 unicast-routing
!
interface Tunnel0
  ipv6 enable
  tunnel source Serial0/0                    !! 注意，不用指 destination，因为是动态的
  tunnel mode ipv6ip 6to4
interface fast1/0
  ipv6 address 2002:CA65:0C01::FFFF/64
  ipv6 enable
interface Serial0/1
  ip address 202.101.12.1 255.255.255.0
!
ipv6 route 2002::/16 Tunnel0
```

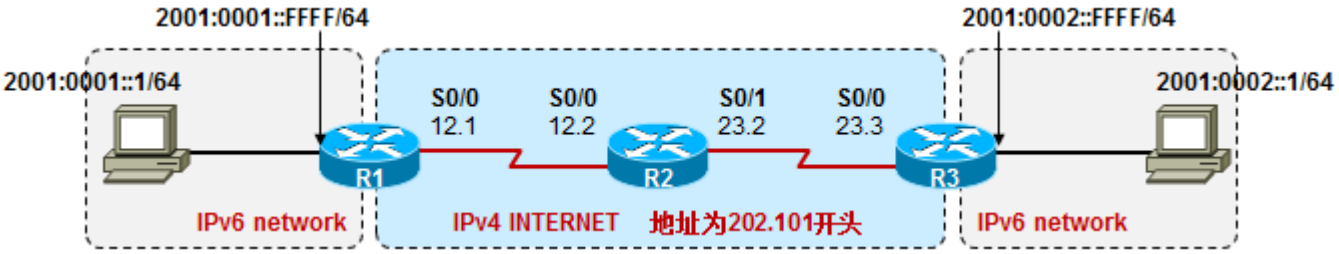
R2 的配置如下：

```
ipv6 unicast-routing
!
interface Tunnel0
  ipv6 enable
  tunnel source Serial0/0
  tunnel mode ipv6ip 6to4
interface fast1/0
  ipv6 address 2002:CA65:1703::FFFF/64
  ipv6 enable
interface Serial0/1
```

```
 ip address 202.101.23.3 255.255.255.0
!
ipv6 route 2002::/16 Tunnel0
```

PC（用路由器模拟）使用无状态自动获取地址： ipv6 address autoconfig default

【实验 2】两边的 IPv6 孤岛都是使用普通的 IPv6 全局地址



两端均为普通 IPv6 网络，使用常规的全局 IPv6 地址。

R1 及 R3 分别申请公网 Ipv4 地址，配置动态 6to4 隧道。以 R1 为例，配置 tunnel0，分配一个 IPv6 地址（6to4 地址），这个地址正是 R1 的 V4 外网地址对应的 IPv6 6to4 地址，也即 202.101.12.1 对应 2002:**CA65:0C01**::1/48 ,R3 同理。接下来其实就是静态路由的把戏了。这时 ,内网有数据去往 2001:0002::/16 网络时，数据被送到 2002:CA65:1703::FFFF（下一跳，也就是 R3 的 tunnel 6to4 地址）

路由器通过递归查询到，下一跳 2002:CA65:1703::FFFF 应该扔到 tunnel0，而 tunnel0 是 6to4 隧道，于是将 2002:CA65:1703::FFFF 翻译成对应的 V4 地址，也就是 202.101.23.3

R1	R3
ipv6 unicast-routing	ipv6 unicast-routing
interface Tunnel0	interface Tunnel0
ipv6 address 2002:CA65:C01::FFFF/64	ipv6 address 2002:CA65:1703::FFFF/64
ipv6 enable	ipv6 enable
tunnel source Serial0/0	tunnel source Serial0/0
tunnel mode ipv6ip 6to4	tunnel mode ipv6ip 6to4
interface fast1/0	interface fast1/0
ipv6 address 2001:0001::FFFF/64	ipv6 address 2001:0002::FFFF/64
ipv6 enable	ipv6 enable
interface Serial0/0	interface Serial0/0
ip address 202.101.12.1 255.255.255.0	ip address 202.101.23.3 255.255.255.0

ip route 0.0.0.0 0.0.0.0 202.101.12.2	ip route 0.0.0.0 0.0.0.0 202.101.23.2
ipv6 route 2001::/16 2002:CA65:1703::FFFF	ipv6 route 2001::/64 2002:CA65:C01::FFFF
ipv6 route 2002:CA65:1703::/48 Tunnel0	ipv6 route 2002:CA65:C01::/48 Tunnel0

要注意的是 6to4 的隧道技术，不支持动态路由协议。

7.3.4 GRE 隧道

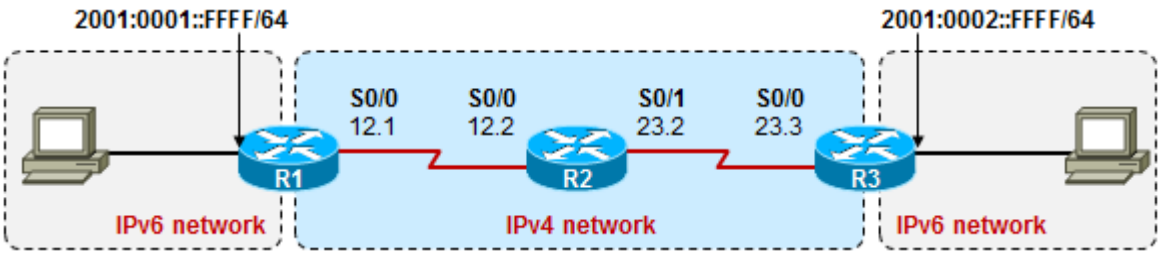
1. 机制介绍

- GRE 隧道是一种众所周知的能够保证稳定和安全的端到端链路的标准隧道技术。
- GRE 隧道同样需要在隧道的两端手工互相指定隧道的目的地
- GRE 隧道给在域内使用 IS-IS 作为 IPv6 路由选择协议的组织提供了方便。如果是 IPv6 over IP 的手工隧道，跑 IS-IS 是会出问题的，因为 IS-IS 需要在网络上相邻的路由器之间发送链路层信息，GRE 是仅有的能够在 IP 基础设施上携带这种类型流量的隧道协议。所以，同一条 GRE 隧道可以用来在广域网中同时传输 IPv6 数据包及 IS-IS 路由器间的 IS-IS 链路层信息。

2. GRE 隧道配置

```
Interface tunnel 0
Ipv6 enable
tunnel mode gre ip                    !! 注意隧道模式
tunnel source serial 0/0
tunnel destination 10.1.23.3
```

3. GRE 隧道基础实验



R1 的配置如下：

```
Ipv6 unicast-routing
Interface serial0/0
```

```
 ip address 10.1.12.1 255.255.255.0
Interface fa1/0
 Ipv6 enable
 Ipv6 address 2001:0001::FFFF/64
!
Interface tunnel 0
 Ipv6 enable
 tunnel mode gre ip                    !! 注意隧道模式
 tunnel source serial 0/0
 tunnel destination 10.1.23.3
!
ip route 0.0.0.0 0.0.0.0 10.1.12.2     !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0   tunnel 0             !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

R3 的配置大同小异，只不过隧道接口的 destination 修改一下即可；

如果需要在 R1、R3 之间运行动态路由协议，则在 tunnel 接口上激活路由选择协议即可。

GRE 隧道报文抓包如下：

```
⊞ Cisco HDLC
⊟ Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 124
    Identification: 0x0030 (48)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: GRE (0x2f)
  ⊞ Header checksum: 0x841d [correct]
    Source: 10.1.12.1 (10.1.12.1)
    Destination: 10.1.23.3 (10.1.23.3)
⊞ Generic Routing Encapsulation (IPv6)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source: 2001:1::1 (2001:1::1)
    Destination: 2001:2::1 (2001:2::1)
⊞ Internet Control Message Protocol v6
```

7.3.5 ISATAP 隧道

1. **机制介绍**

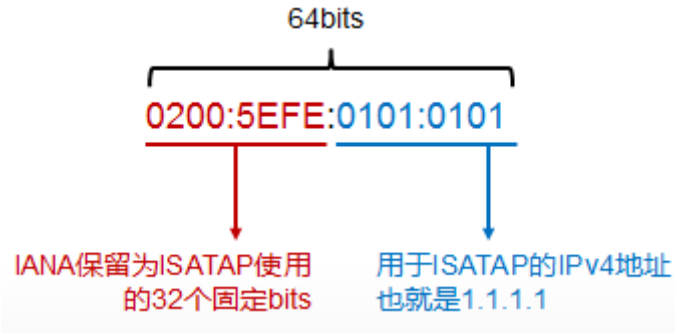
ISATAP（Intra-Site Automatic Tunnel Addressing Protocol）

ISATAP 是一种非常容易部署和使用的 IPv6 过渡机制。在一个 IPv4 网络中，我们可以非常轻松的进行 ISATAP 的部署，首先你的 PC 需是 V4/V6 双栈 PC，然后，需要有一台支持 ISATAP 的路由器，ISATAP 路由器可以在网络中的任何位置，只要 PC 能够 ping 通它（当然，你要知道路由器的 IPv4 地址）。那么接下去，我们可以通过在路由器上部署 ISATAP，这样网络中支持 ISATAP 的双栈主机，在需要访问 IPv6 资源时，可以与 ISATAP 路由器建立起 ISATAP 隧道，ISATAP 主机根据 ISATAP 路由器下发的 IPv6 前缀构造自己的 IPv6 地址（这个 IPv6 地址是被自动关联到 ISATAP 主机本地产生的一个 ISATAP 虚拟网卡上），并且将这台 ISATAP 路由器设置为自己的 IPv6 默认网关，如此一来，后续的这台主机就能够通过这台 ISATAP 路由器去访问 IPv6 的资源。

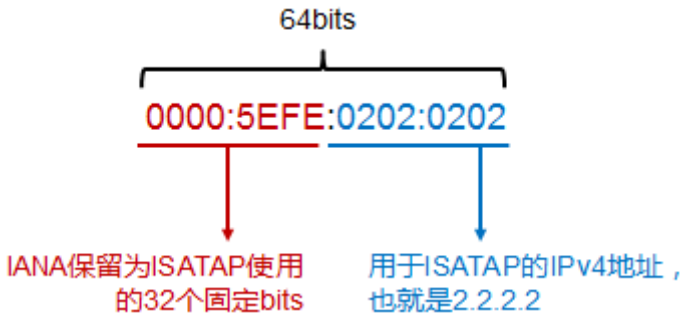
这种方法部署起来非常简单，在许多场合，客户为了节省成本，又希望网络中的 IPv6 主机能够访问 V6 资源，同时又不愿意对现有网络做大规模的变更及设备升级，那么就可以采用这种方法，购买一台支持 ISATAP 的路由器，甚至可以将 ISATAP 路由器旁挂在网络上，只要它能够访问 V6 资源并且响应 ISATAP PC 的隧道建立请求。

2. **ISATAP 的功能组件如下：**

- **自动隧道：** ISATAP 的隧道机制也是自动的，隧道在主机和 ISATAP 路由器之间被创建。主机首选需要知道 ISATAP 路由器的 IPv4 地址。
- **ISATAP 地址格式：** 分配给 ISATAP 路由器的 IPv6 地址是全局单播地址，该地址的前缀将被 ISATAP 主机用于自己的 IPv6 地址构造。ISATAP 主机通过在 IPv4 建立起来的 ISATAP 隧道从 ISATAP 路由器发送的消息中接收/64 的 IPv6 前缀，并且使用这个前缀结合“特殊的接口标识”来构造自己的 IPv6 地址。
- **接口标识：** ISATAP 在主机上启用后，会产生一个 ISATAP 虚拟网卡，该虚拟网卡会产生一个 64bits 的特殊接口标识，有点类似 EUI-64，但是产生机制不同，它是由专为 ISATAP 保留的 32 位的 **0200:5EFE** 加上主机上配置的 IPv4 地址构成，如下图，假设 ISATAP 主机配置的 IPv4 地址为 1.1.1.1，那么 ISATAP 虚拟网卡的 64bits 接口标识就是：



另一方面，在路由器上部署 ISATAP 后，路由器也会产生一个 tunnel 接口，用于响应 ISATAP 主机的隧道建立请求，这个 tunnel 接口同样会产生接口标识。地址的格式是 IANA 保留给 ISATAP 的 32 比特的 **0000:5EFE** 后追加 32 比特的 IPv4 地址。如下图，假设给 ISATAP 路由器配置的 IPv4 地址（用于隧道的）是 2.2.2.2，那么 ISATAP tunnel 的接口标识就是：

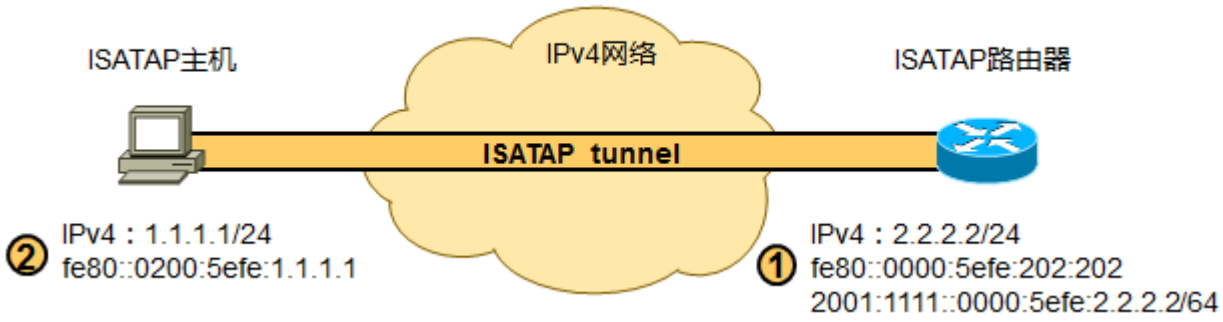


这里关于 64bits 的接口标识中“为 ISATAP 保留的”高阶 32bits 在维基百科上有这么一段描述：“The link-local address is determined by concatenating fe80:0000:0000:0000:**0200:5efe:** for global unique and fe80:0000:0000:0000:**0000:5efe**: for private addresses with the 32 bits of the host's IPv4 address.”。貌似有全局唯一和私有之分，不过在 IETF 的相关草案上找到的更多是 0200:5efe 的描述，在我所作的测试环境中，windows 主机上系统使用的是 0200:5ede，而 CISCO 路由器上用的是 0000:5efe。

ISATAP 主机和 ISATAP 路由器产生的这个 64btis 的接口标识，可进一步用于构造隧道接口的 Linklocal 地址，以及 IPv6 全局单播地址。这个下面会描述到。

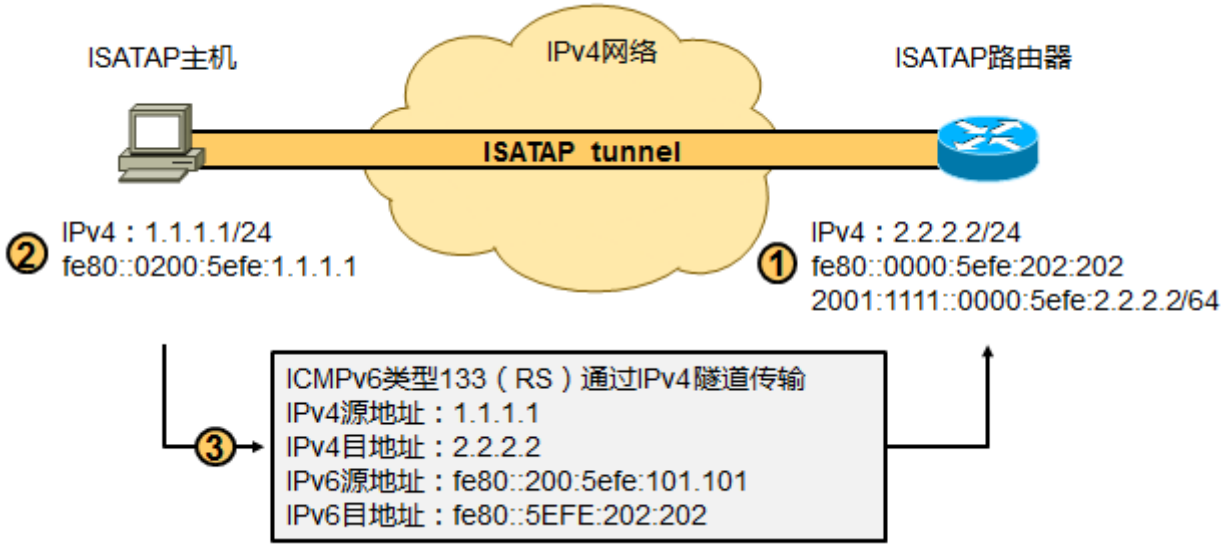
另外，因为 ISATAP 的操作范围在站点内，所以 ISATAP 主机和 ISATAP 路由器的 IPv4 地址可以是私有 IP，也可以是公有 IP。

3. 工作机制



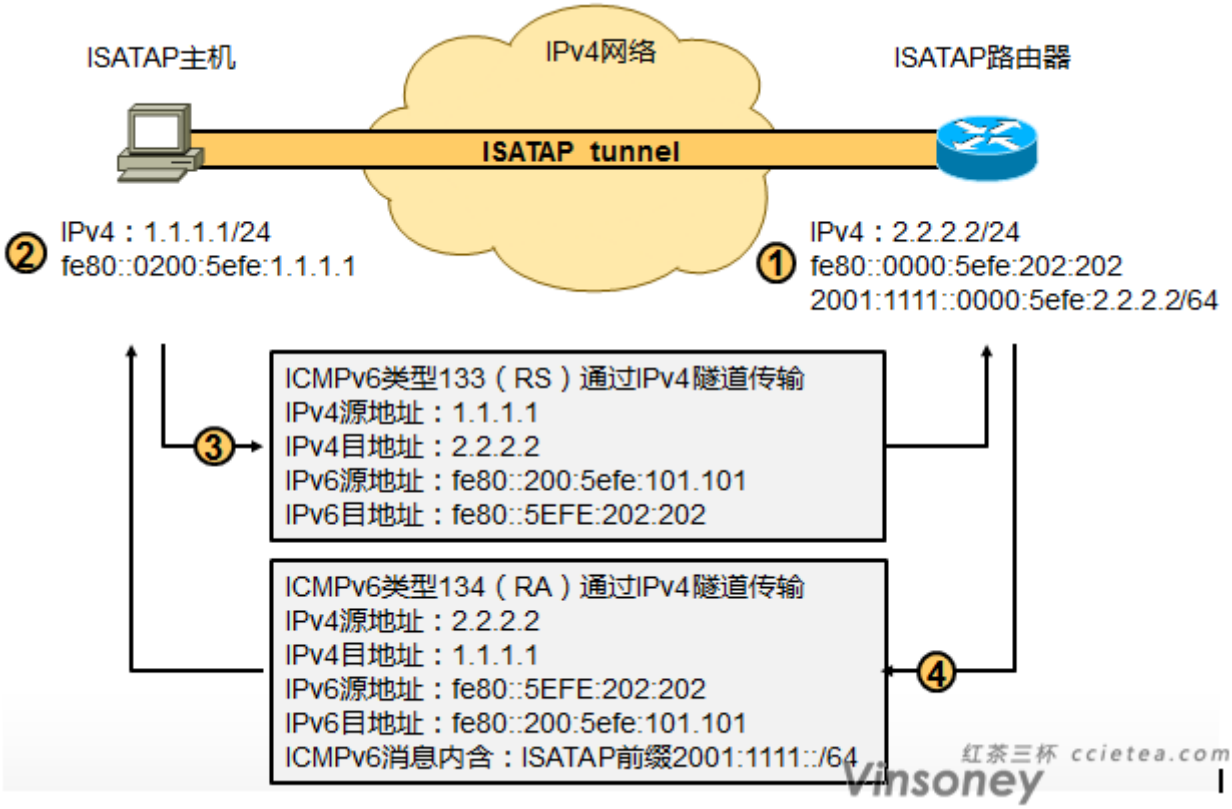
首先，我们有一个 IPv4 的网络，IPv4 网络中绝大部分网路设备都不支持 IPv6，除了终端主机，以及一台路由器，这台能够访问我们需要的 IPv6 资源。现在，一种最廉价的方式是，在这台路由器上部署 ISATAP，终端 ISATAP 主机与路由器之间建立一个 ISATAP tunnel，这样一来 PC 可以直接将 IPv6 流量放进 tunnel 传到 ISATAP 路由器从而穿越整个 IPv4 网络。

1) 现在我们在 ISATAP 路由器上进行相应的配置，给路由器分配的 IPv4 地址是 2.2.2.2/24，同时建立一个 tunnel 接口用于 ISATAP，此时 tunnel 接口会根据 IPv4 地址产生一个 64bits 的接口标识。这个接口标识搭配上高位的 fe80::就形成了 tunnel 接口的 Linklocal 地址：fe80::0000:5efe:202:202。另外，还需给 ISATAP tunnel 接口配置一个全局单播 IPv6 地址，这里可以手工配置，也可以通过前缀+EUI64 的方式来构建，这里的 EUI-64 就是上面所述的特殊的 64bits 接口标识。如上图，构建出来的 IPv6 地址就是 2001:1111::0000:5efe:0202.0202/64，因此 IPv4 的前缀为 2001:1111::/64，这个前缀稍后会通过 tunnel 下发给 ISATAP 主机，从而使它能够构建自己的 IPv6 地址。
2) 现在我们在 ISATAP 主机上，配置 ISATAP，一般来说，在 WIN7 系统上默认安装了 IPv6 协议栈，默认就会有一个 ISATAP 的虚拟网卡。在我们给 PC 的物理网卡配置 IPv4 地址如 1.1.1.1/24 后，ISATAP 虚拟网卡就会自动根据这个 IPv4 地址计算出上面所讲的特殊的接口标识：0200:5efe:1.1.1.1，注意这种格式等同与 0200:5efe:0101.0101，在 windows 系统上我们可以看到前者的简便写法。
3) 当我们在主机上配置了 ISATAP 路由器之后（指向的是 ISATAP 路由器的 IPv4 地址），ISATAP 主机开始向 ISATAP 路由器发送 RS 消息，如下图：



这个 RS 消息是通过 IPv4 隧道传输的，外层是 IPv4 的头，源地址是 ISATAP 的 IPv4 地址 1.1.1.1，目的地址是 2.2.2.2，也就是 ISATAP 的 IPv4 地址。IPv4 头里面裹着 IPv6 的报文，源地址是 ISATAP 主机的 ISATAP 虚拟网卡的 Linklocal 地址，目的地址是 ISATAP 路由器的 Linklocal 地址。

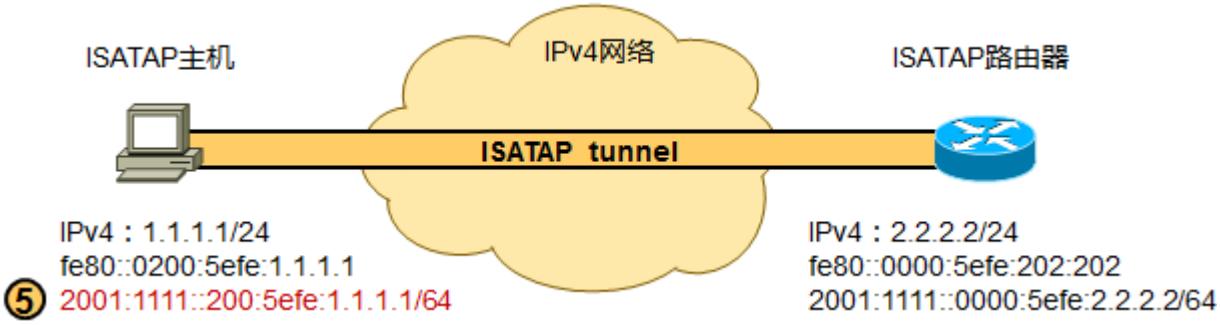
4) ISATAP 主机发出的这个 RS 消息，会在 IPv4 网络中被路由，最终转发到 ISATAP 路由器。这将使得路由器立即以一个 RA 进行回应：



而这个回应的 RA 消息里，就包含 ISATAP 上所配置的那个 IPv6 全局单播地址的/64 前缀。

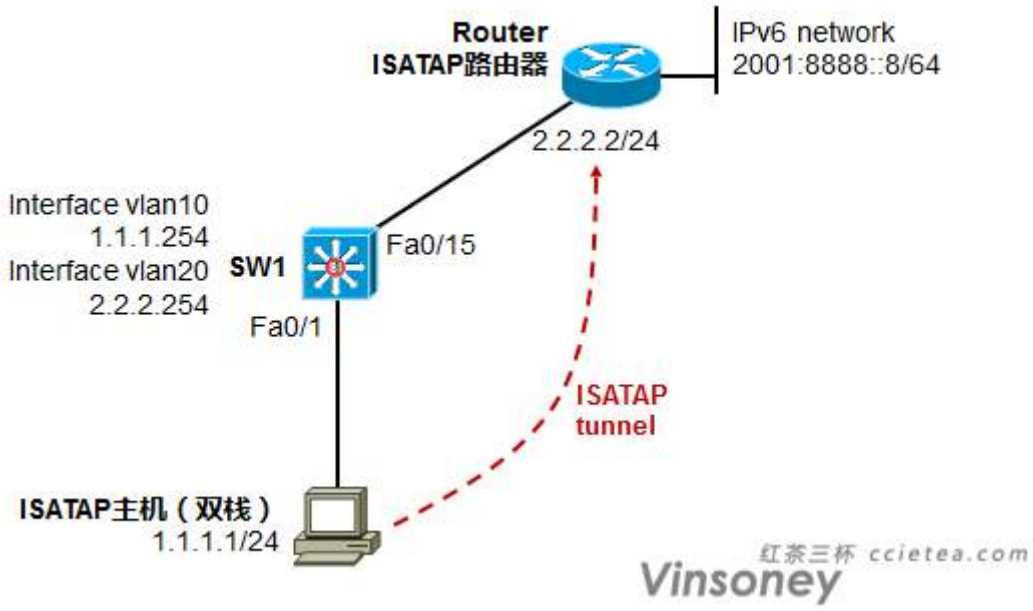
5) ISATAP 主机收到这个 RA 回应后，会拿出里头的 IPv6 前缀，随后在后面加上自己 ISATAP 虚拟网卡的

64bits 的接口标识地址，构成 128bits 的 IPv6 全局单播地址，同时会产生一条默认路由，指向 ISATAP 路由器的 Linklocal 地址：



6) 从现在起，ISATAP 主机需要访问 IPv6 资源的时候，将 IPv6 数据包封装在 IPv4 的隧道里，也就是说，套上 ISATAP 隧道的 IPv4 头，然后传给 ISATAP 路由器，再由 ISATAP 路由器解封装，再帮忙转发 IPv6 数据。

4. 典型实验



- **环境描述**

1) PC 是 ISATAP 主机，它是一台双栈 PC，这里我们使用的是一台 win7 系统的电脑做测试。电脑网卡的 IP 地址为 1.1.1.1/24，网关为 1.1.1.254，网关是 SW1 的 interface vlan10。

2) SW1 创建两个 VLAN：VLAN10 及 20，分别对应 PC 及 ISATAP 路由器。VLAN20 的 SVI 口 IP 为 2.2.2.254，是 ISATAP 路由器的默认网关。

3) ISATAP Router 的接口 IP 为 2.2.2.2。该 IPv4 地址在后续的 ISATAP 配置中使用到，ISATAP 主机就是通过这个 IP 找到 ISATAP Router 并与之建立 ISATAP tunnel。ISATAP Router 同时连接到了一个 IPv6 网络，这里我们用 loopback 模拟：2001:8888::8/64，用于后续的测试。

4) 最终的实验结果是首先 PC 要能够 ping 通 ISATAP Router 的 IPv4 地址也就是 2.2.2.2。然后 PC 与 ISATAP router 建立隧道并拿到 IPv6 地址，而且要能够 ping 通 20001:8888::8

- **设备配置**

PC1 的配置：

网卡配置 IP 地址 1.1.1.1/24，网关为 1.1.1.254

安装 IPv6 协议栈，此时 Win7 会自动产生一个 ISATAP 隧道虚拟接口：

```
隧道适配器 isatap.{0DB7233C-89B7-49DB-A8C0-D1AA005F4E6A}:
```

SW1 的配置：

```
vlan 10
vlan 20
interface fast0/1
  switchport access vlan 10
interface fast0/15
  switchport access vlan 20
interface vlan 10
  ip address 1.1.1.254 255.255.255.0
interface vlan 20
  ip address 2.2.2.254 255.255.255.0
```

Router 的配置：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ip address 2.2.2.2 255.255.255.0
  no shutdown
!
interface Tunnel1
  ip unnumbered   fastEthernet 0/0        !! 这个 IPv4 地址就是 ISATAP 隧道的目的地址
  ipv6 enable
  ipv6 address 2001:1111::/64 eui-64      !! 这个 IPv6 地址的前缀会被通告给 ISATAP 主机
  no ipv6 nd suppress-ra
  tunnel source fastEthernet 0/0
```

```
 tunnel mode ipv6ip isatap
!
interface loopback0
 ipv6 enable
 ipv6 address 2001:8888::8/64
!
ip route 0.0.0.0 0.0.0.0 2.2.2.254
```

注意 ISATAP 路由器的配置，关键部分在于 tunnel 的配置，tunnel 模式是 ipv6ip isatap 的，同时注意在 tunnel 这里配置的 IPV4 地址，就是对应的 ISATAP 主机上配置的那条 CMD 命令里 ISATAP 路由器的地址。我们这个实验演示的是 tunnel 直接用 fa0/0 的地址，当然，tunnel 也可以有自己的 IPv4 地址，只要保证 ISATAP 主机到这个 IPv4 地址路由可达就行。另外 tunnel 的 IPv6 地址，对应的前缀就是稍后要下发给 ISATAP 主机的前缀，这个实验中，我们 tunnel 的 IPv6 全局单播地址使用的是前缀+eui-64 的配置方式，这里的 eui-64 实际上指的就是前面我们介绍的那个特殊的 64bits 接口标识。

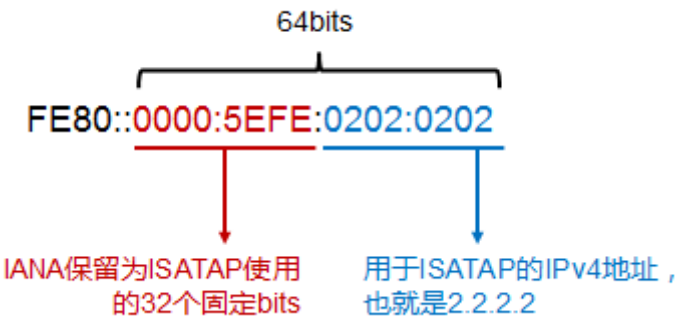
- **实验测试**

我们首先在路由器上查看一下：

R2#show ipv6 interface brief

```
FastEthernet0/0          [up/up]
Tunnel0                  [up/up]
    FE80::5EFE:202:202
    2001:1111::5EFE:202:202
```

注意，这里的 Linklocal 地址：FE80::5EFE:202:202 就是一个 ISATAP 格式的地址，最后的 64bits 是由 32bits 的 0000:5EFE 加上 32bits 的接口 IPv4 地址（这里是 2.2.2.2）构成的，如下图。而 IPv6 全局单播地址，也是使用 64bits 的接口标识构成的，当然，你也可以手工配置 IPv6 全局单播地址，不一定要使用接口标识。



接下去，我们在 ISATAP 主机上，CMD 模式下输入：

```
netsh interface ipv6 isatap set router 2.2.2.2
```

PC 就会开始发送 RS，报文如下：

```
Internet Protocol, Src: 1.1.1.1 (1.1.1.1), Dst: 2.2.2.2 (2.2.2.2)
Internet Protocol Version 6
⊞ 0110 .... = Version: 6
  .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
  .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 8
  Next header: ICMPv6 (0x3a)
  Hop limit: 255
  Source: fe80::200:5efe:101:101 (fe80::200:5efe:101:101)
  Destination: fe80::5efe:202:202 (fe80::5efe:202:202)
Internet Control Message Protocol v6
  Type: 133 (Router solicitation)
  Code: 0
  Checksum: 0xb7b8 [correct]
```

我们看到这个 RS 的 ICMPv6 报文外是 IPv6 的头，IPv6 的头外是 IPv4 的头。

注意外层 IPv4 的头，源是 1.1.1.1，目的是 2.2.2.2

然后内层 IPv6 的头，源是 ISATAP 主机的 Linklocal 地址，目的是 ISATAP 路由器的 Linklocal 地址

在路由器收到 RS 后回回应一个 RA：

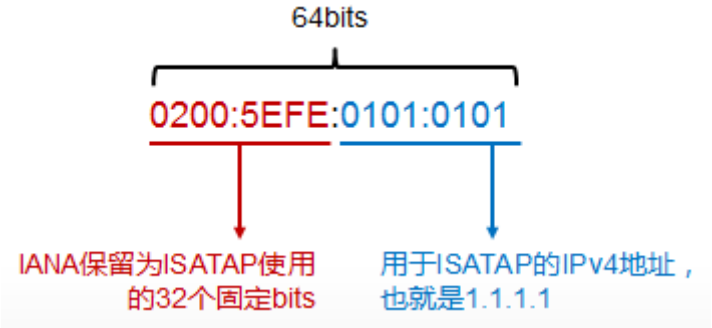
```
Internet Protocol, Src: 2.2.2.2 (2.2.2.2), Dst: 1.1.1.1 (1.1.1.1)
Internet Protocol Version 6
⊞ 0110 .... = Version: 6
  .... 1110 0000 .... .... .... .... .... = Traffic class: 0x000000e0
  .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 56
  Next header: ICMPv6 (0x3a)
  Hop limit: 255
  Source: fe80::5efe:202:202 (fe80::5efe:202:202)
  Destination: fe80::200:5efe:101:101 (fe80::200:5efe:101:101)
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x2830 [correct]
  Cur hop limit: 64
⊞ Flags: 0x00
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊞ ICMPv6 Option (MTU)
⊞ ICMPv6 Option (Prefix information)
```

路由器回应的这个 RA 里，就有一个 ICMPv6 的 Option，其中就包含着 ISATAP 路由器的 IPv6 前缀。而 ISATAP 主机就可以根据这个前缀，结合自己的接口标识构建 IPv6 地址。

最终 PC 获取到的 IPv4 地址如下：

```
隧道适配器 isatap.{0DB7233C-89B7-49DB-A8C0-D1AA005F4E6A}:
   连接特定的 DNS 后缀 . . . . . . . :
   IPv6 地址 . . . . . . . . . . . . : 2001:1111::200:5efe:1.1.1.1
   本地链接 IPv6 地址. . . . . . . . : fe80::200:5efe:1.1.1.1%40
   默认网关. . . . . . . . . . . . . : fe80::5efe:2.2.2.2%40
```

我们看到 PC 首先根据自己本地配置的 IPv4 地址：1.1.1.1，生成 64bits 的接口 ID：



这个 64bits 的接口 ID ,与从 ISATAP 路由器获取到的 IPv6 全局单播地址前缀 2001:1111::的前 64bits，构成了 PC 的 IPv6 全局单播地址：2001:1111::200:5efe:1.1.1.1。

这个 64bits 的接口 ID，与 FE80::/10 构成了 PC 的 Linklocal 地址：fe80::200:5efe:1.1.1.1

同时，PC 将 ISATAP 路由器的 Linklocal 地址 fe80::5efe:2.2.2.2 设置为默认网关

当主机与其它 *IPv6* 主机进行通讯时，从隧道接口转发，将从报文的下一跳 *IPv6* 地址中取出 *IPv4* 地址作为 *IPv4* 封装的目的地址。如果目的主机在本站点内，则下一跳就是目的主机本身，如果目的主机不在本站点内，则下一跳为 *ISATAP* 路由器的地址。

我们最后再做一个测试，就是 ISATAP 主机去 ping 2008:8888::1。

```
⊞ Internet Protocol, Src: 1.1.1.1 (1.1.1.1), Dst: 2.2.2.2 (2.2.2.2)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x0000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: 2001:1111::200:5efe:101:101 (2001:1111::200:5efe:101:101)
    Destination: 2001:8888::8 (2001:8888::8)
⊞ Internet Control Message Protocol v6
```

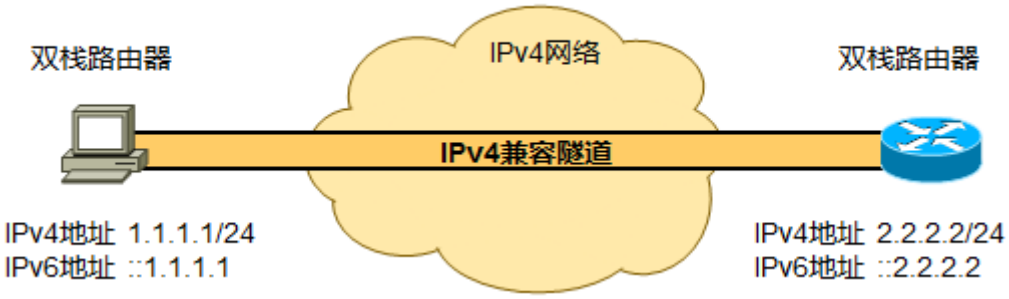
这个到达 ISATAP 的 loopback 的 IPv6 数据包，被套上一个 ISATAP 的 IPv4 隧道头，然后传给 ISATAP 路由器，由路由器进行下一步的 IPv6 转发。

7.3.6 自动 IPv4 兼容隧道

这种技术使用的比较少，是 IETF 的第一批转换机制之一。自动 IPv4 兼容隧道机制只允许两个双栈主机之间的 IPv6 数据在 IPv4 网络上进行自动隧道（无需手动配置）传输。这个机制允许 IPv4 网络上的孤立主机和另一个 IPv4 网络上的孤立主机之间自动启用隧道，从而实现 IPv6 数据在隧道中的传输。源和目的的 IPv6 地址的低 32 位地址表示隧道终点的源和目的 IPv4 地址。

自动 IPv4 兼容隧道机制使用的 IPv6 地址是“IPv4 兼容 IPv6 地址”

：：a.b.c.d/96（前面 96 个 0，后面跟上的 a.b.c.d 是 IPv4 地址）



适用于不经常性的 IPv6 节点连接需求。IPv4 兼容隧道是通过 Tunnel 虚接口实现的，如果一个 Tunnel 口的封装模式是 IPv4 兼容隧道，则只需配置隧道的源地址，而目的地址是在转发报文时，从 IPv6 报文的目的地址中取得的。从 IPv4 兼容隧道转发的 IPv6 报文的目的地址必须是 IPv4 兼容的 IPv6 地址，隧道的目的地址就是 IPv4 兼容地址的后 32 位。如果一个 IPv6 报文的目的地址不是 IPv4 兼容地址，则不能从 IPv4 兼容隧道转发出去。

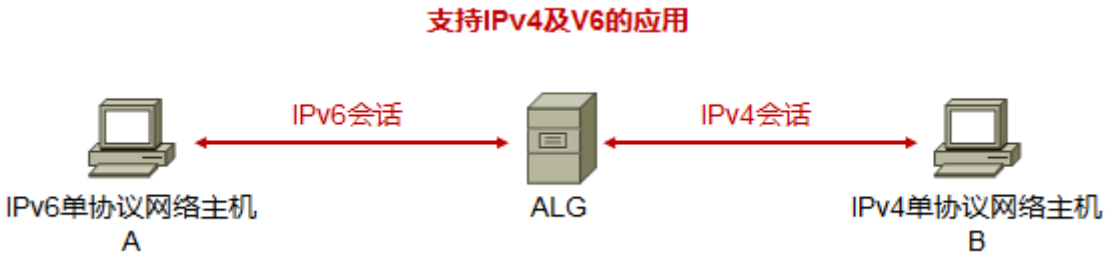
这种机制现在都不用了。

7.3.7 Teredo 隧道机制

8 IPv6 单协议网络到 IPv4 单协议网络的过渡机制

前面我们讨论的 V4 到 V6 的过渡机制，基本都是存在 V4、V6 共存的情况。那么如果是 V4 单协议和 V6 单协议网络有互访需求呢？

8.1 应用层网关 ALG



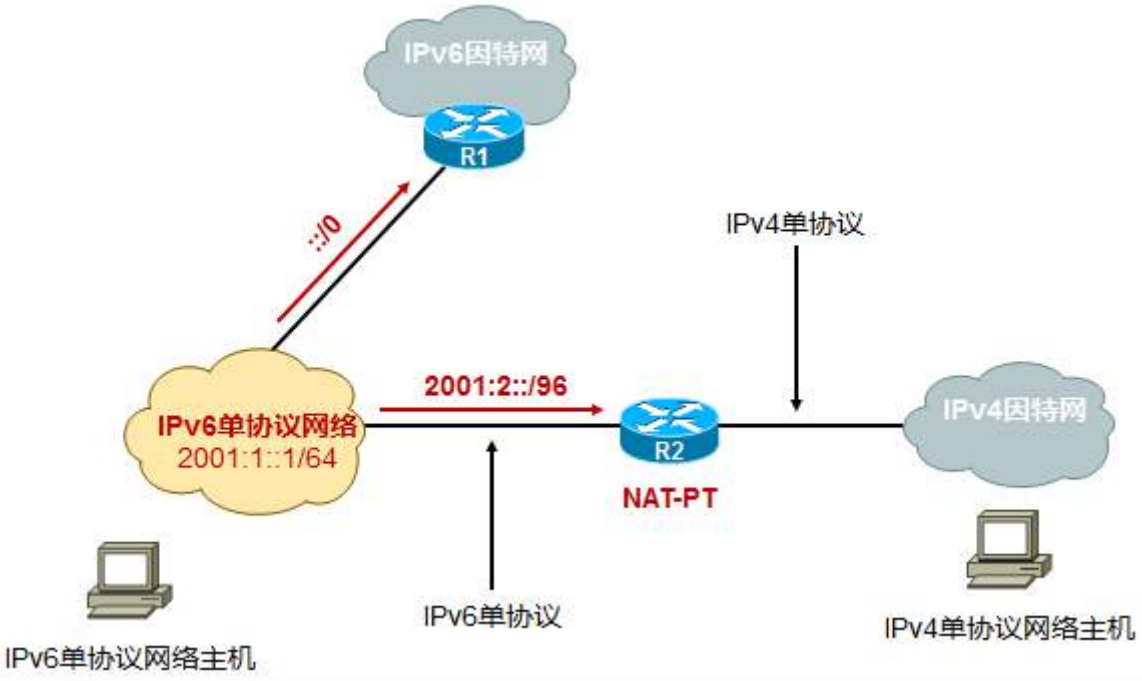
IPv6 单协议网络主机 A 与 ALG 之间维护着一条 IPV6 的会话，主机 A 可以向 ALG 发送 IPV6 数据。如此同时，ALG 与 IPV4 单协议网络主机 B 维护着一条 IPV4 会话。ALG 负责 IPV4 与 IPV6 的会话转换。这个 ALG 具有双栈的支持能力。举个例子，假设 ALG 是一台双栈，且可充当 SMTP　ALG 的服务器。那么 A 可以基于 IPV6 的 SMTP 发送电子邮件到这台邮件服务器。当然，B 可以使用 IPV4 会话去读取。

8.2 NAT-PT

8.2.1 机制概述

RFC2766、RFC2765。NAT-PT（网络地址转换-协议转换）是一种地址转换技术，它可以把 IPv6 地址转换成 IPv4 地址，反之亦然。NAT-PT 基于 RFC2766 中定义的无状态 IP/ICMP 转换器（SIIT）算法。SIIT 算法互译 IPv4 和 IPv6 数据包头部，也包括 ICMP 头部。

需要注意的是，在 IPv6 环境中，不建议像 IPv4 对待 NAT 的态度哪样，去使用 NAT。仅仅在 V4 单协议与 V6 单协议网络需要互相通信的时候，才建议使用 NAT-PT。



我们看上面的例子，对于 IPv6 单协议网络而言，首先它有访问 IPv6 因特网的需求，因此默认的 IPv6 流量全部交给 R1，另外，它可能还有访问 IPv4 因特网的需求，这时候，就需要借助 R2 这台 NAT-PT 设备。这台 NAT-PT 设备首先肯定是双栈，其次它能够将来自 IPv6 网络的流量，转换成 IPv4 流量放进 V4 网络，反之亦然。

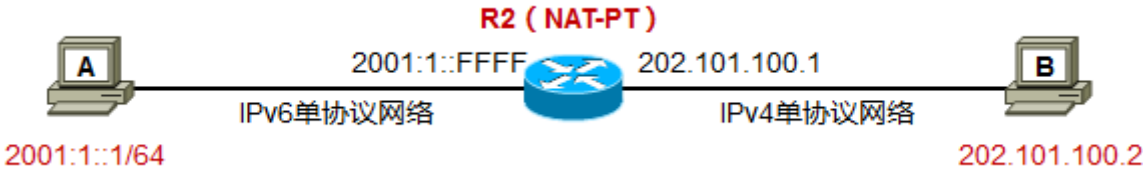
2001:2::/96，这个长度为 96 位的前缀是我们为了 NAT-PT 操作预定义的前缀，前缀可以自定义，但是长度必须是 96bits。这个预定义的前缀非常关键，相当于为 R2 右侧的 IPv4 网络预定义一个 IPv6 的空间。举个非常简单的例子，IPv6 单协议网络中，如果有 PC 想访问 IPv4 单协议网络中的 PC，那么咋访问？目的地址是啥？总不能让一个 IPv6 only 的 PC 去 ping 一个 IPv4 地址吧？因此这个/96 的预定义前缀就派上用场了，可以理解为它就是在这种场景中，为右侧的 IPv4 only 网络所服务的一个预定义的 IPv6 地址空间。

在 IPv6 单协议网络中产生的，去往 2001:2::/96 这个目的地的流量被路由到 R2 也就是 NAT-PT 设备，然后数据包中的 IPv6 地址被转换为 IPv4 地址并传送给 IPv4 因特网中的 IPv4 单协议节点。

8.2.2 NAT-PT 配置及原理

8.2.2.1 静态 NAT-PT

1. 静态 NAT-PT（单向）

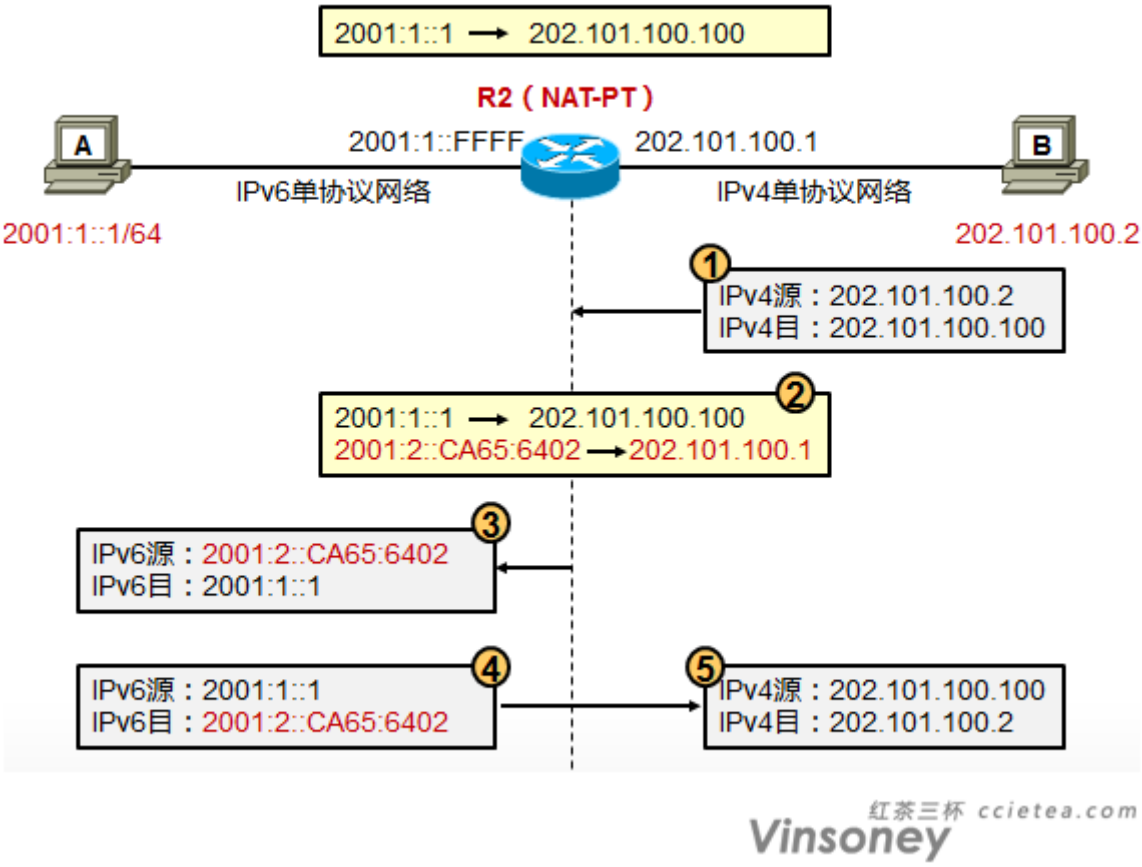


A 和 B 的配置都极其简单，这里就不说了

R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable                                  !! 连接 A 的接口
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
ipv6 nat prefix 2001:2::/96                     !! 是一个为 NAT-PT 预留的池
ipv6 nat v6v4 source 2001:1::1 202.101.100.100      !! 相当于将 2001:1::1 这个 IPv6 的节点，“告知”给 IPv4 单协议网络中的用户知道，可以以 202.101.100.100 的方式访问。
```

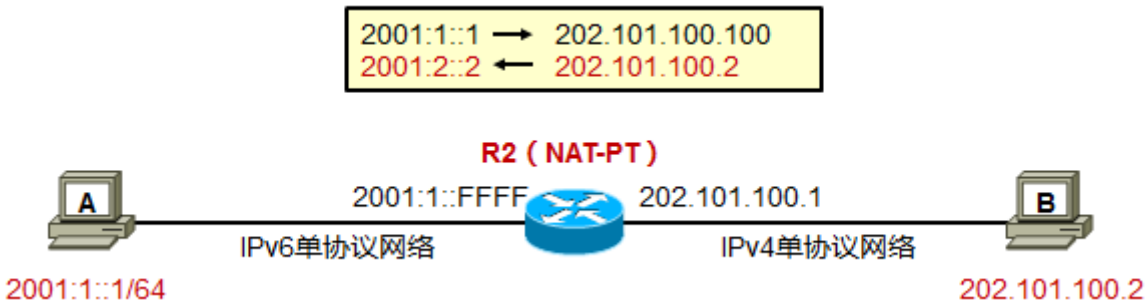
在上述配置中 ,我们将 A 节点 ,也就是 2001:1::1 映射到了 IPv4 网络 ,映射到 202.101.100.100 这个 IPv4 地址。这样一来 B 可以主动去访问 A，例如 B 去 ping 202.101.100.100，能够 ping 通，这个过程如下：



1) 首先 B 去 ping 202.101.100.100，数据包如图所示。
2) 数据包到达 R2 后，R2 本地是已经存在了一个映射：2001:1::1 映射到 202.101.100.100
   由于存在这个映射，因此 R2 将 B 发过来的这个数据包的目的地址 202.101.100.100 替换成 2001:1::1，同时 B 的源地址是个 IPv4 地址 ,咋办呢？还记得我们配置了 NAT-PT 预留的前缀么？也就是 2001:2::/96 这个玩意儿 ,R2 将 B 的 IPv4 地址映射到这个前缀上 ,构成一个临时的 IPv6 地址 :2001:2:CA65 :6402，同时在本地生成一条新的映射条目：**2001:2::CA65:6402 映射到 202.101.100.1**，如上图所示。
3) R2 将原始的 IPv4 数据包的包头替换成 IPv6 的包头，然后转给 A。
4) A 回包，数据包送给 R2，R2 由于已经有了 2001:2::CA65:6402 到 202.101.100.1 的映射，因此 R2 将 IPv6 包头替换成 IPv4 包头，然后再转发给 B。

注意在此时，A 是可以去主动访问 B 的，也就是说 A 可以主动 ping 2001:2::CA65:6402 这个临时的地址来达到访问 B 的目的。但是如果我们在 R2 上去 clear ipv6 nat translation * ，如此 2001:2::CA65:6402 到 202.101.100.1 的映射条目就被清空了，A 就无法主动访问 B 了，只能 B 先主动访问 A。

2. 静态 NAT-PT（双向）



R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
ipv6 nat prefix 2001:2::/96
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
ipv6 nat v6v4 source 2001:1::1 202.101.100.100
```

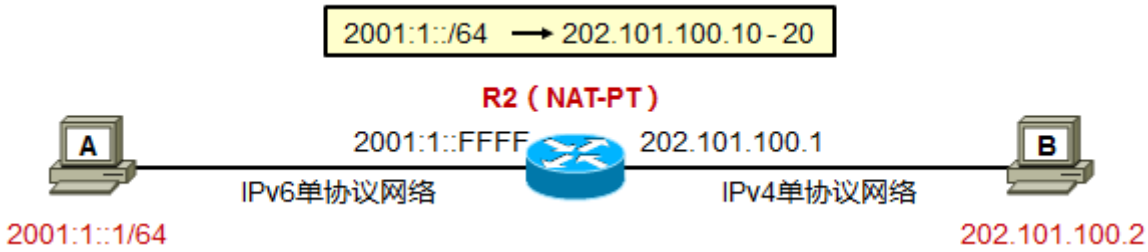
如此一来，A 主动发起访问连接到 B，或者 B 主动发起访问到 A 都可以。

NAT-PT#show ipv nat translations

Prot	IPv4 source IPv4 destination	IPv6 source IPv6 destination
---	---	---
	202.101.100.2	2001:2::2
---	202.101.100.100 202.101.100.2	2001:1::1 2001:2::2
---	202.101.100.100 ---	2001:1::1 ---

8.2.2.2 动态 NAT-PT

1. **V6 可以主动访问 V4（使用 V4 地址池）**



R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
ipv6 nat prefix 2001:2::/96
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
```

来分解一下关键命令：

```
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
```

上面的命令是定义允许被 IPv6 nat 的源地址，用一个 IPv6 ACL 进行匹配。

```
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
```

上面的命令是，创建一个供 v6tov4 使用的地址池，这个地址池当然是 IPv4 的地址池，当 IPv6 only 的用户，如 A 要访问 IPv4 only 网络的时候，就从池中取一个空闲的 IPv4 地址。

```
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
```

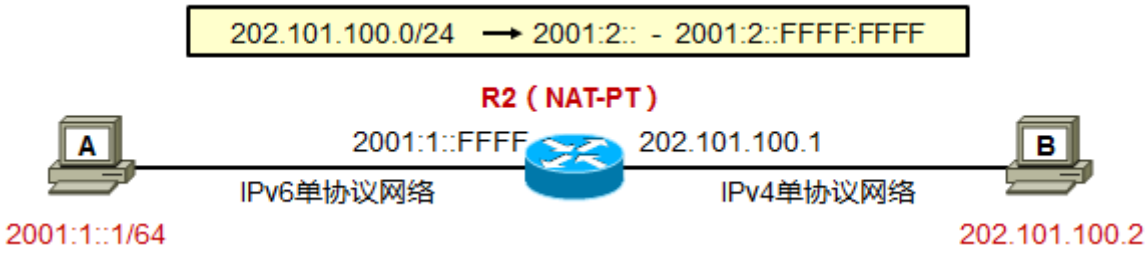
将 ACL 所允许的 IPv6 网络与这个 v6v4 地址池进行关联

```
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
```

最后这条命令是将 202.101.100.2 这个 IPv4 网络的主机“放进来”到 IPv6 网络，使得 IPv6 only 的用户有访问目标，否则，你让 IPv6 only 的用户用什么目标地址去访问 IPv4 only 网络？

完成上述配置后，A 即可主动发起访问到 B 了， 使用目标地址 2001:2::2 即可访问 B

2. V4 可以主动访问 V6（使用 V6 地址池）



R2 的配置如下：

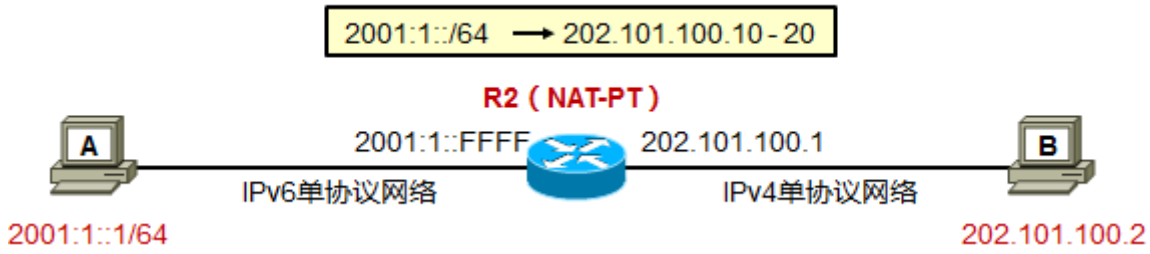
```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:1::FFFF/64
  ipv6 nat
!
interface FastEthernet1/0
  ip address 202.101.100.1 255.255.255.0
  ipv6 nat
!
access-list 1 permit 202.101.100.0 0.0.0.255
ipv6 nat v4v6 pool v4v6-pool 2001:2:: 2001:2::FFFF:FFFF prefix-length 96
ipv6 nat v4v6 source list 1 pool v4v6-pool
ipv6 nat prefix 2001:2::/96
ipv6 nat v6v4 source 2001:1::1 202.101.100.111
```

思路和 V6 访问 V4 是一样的。

这里我们用一个 access-list 1 匹配 202.101.100.0/24 网络，同时将其与 v4v6-pool 这个 IPv6 地址池做了捆

绑。使得 B 访问 IPv6 网络的时候，可以从池中拿 IPv6 地址。另外，为了让 V4 网络访问 V6 网络有目标，还做了一条静态映射，将 2001:1::1 映射到 202.101.100.111，这样，B 就能够使用 202.101.100.111 这个 IP 来访问 IPv6 主机 A。

3. IPv4-Mapped NAT-PT



回顾一下前面 IPv6 only 网络访问 IPv4 only 网络使用 V4 地址池的情况。

V4 地址池使得 V6 用户在穿越 NAT-PT 设备之后，可以从池中拿一个地址当做源地址。但是，V6 用户如何访问 V4 网络呢？ 总不能让 V6 only 主机去 ping 202.101.100.2 吧？因此，不得不在配置上增加命令：

ipv6 nat v4v6 source 202.101.100.2 2001:2::2

这条命令，将 202.101.100.2 这个 V4 主机映射到 V6 网络，使得 V6 网络用户能够使用地址 2001:2::2 去访问这个 V4 主机。那么如果我们这样的 V4 主机有很多呢？难道需要手工去添加静态映射么？

当然不用，使用 IPv4-mapped 功能即可，机制非常简单，我们在 R2 上修改配置，关键配置如下：

ipv6 access-list v4map permit 2001:1::/64 2001:2::/96

ipv6 nat prefix 2001:2::/96 v4-mapped v4map

首先用一条 IPv6 ACL 来匹配需要进行 IPv4-mapped 的流量，上面的 ACL 源是 IPv6 only 网络，目的是保留给 NAT-PT 的 IPv6 前缀。然后使用 **ipv6 nat prefix 2001:2::/96 v4-mapped v4map 命令**关联这条 ACL。这样一来当 A 访问 B 时，可以直接使用 **2001:2::CA65:6402** 这个地址来访问处于 V4 网络的 B，注意，这里前缀 2001:2::/96 是保留给 NAT-PT 的 IPv6 前缀，后面的 CA65:6402 将被翻译成 IPv4 地址，也就是 202.101.100.2。因此 R2 收到这个 IPv6 数据包的时候，发现数据包匹配上了 v4map 这个 IPv6ACL，因此将目的 IPv6 地址的最后 32bits 翻译成 IPv4 格式（这就是 IPv4 目的地址），然后从 V4 地址池中取出一个地址替换掉 IPv6 地址。完美。

R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:1::FFFF/64
  ipv6 nat
!
```

```
interface FastEthernet1/0
  ip address 202.101.100.1 255.255.255.0
  ipv6 nat
!
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
ipv6 access-list v4map permit 2001:1::/64 2001:2::/96
ipv6 nat prefix 2001:2::/96 v4-mapped v4map
```

9 IPv6 路由选择

9.1 IPv6 路由选择基础

```
Router# Show ipv6 route
```

使用该命令查看路由器 IPv6 路由表

9.2 静态路由

```
Router(config)# ipv6 route  目的 IPv6 网络/前缀长度 { 出接口 | 下一跳 IP }
```

配置 IPv6 静态路由

在 IPv6 规范中，不推荐使用可聚合全球单播或本地站点地址作为下一跳地址，如果这样做，ICMPv6 重定向消息就不会工作。因此使用 Linklocal 地址作为下一跳 ,在某些场合可能更为推荐 ,毕竟 linklocal 地址稳定且长久不变。在配置 linklocal 地址作为下一跳 IP 时，必须关联路由器上相应的接口。例如：

```
ipv route 2222::/64 fastEthernet 0/0 FE80::CE00:1AFF:FEE4:0
```

9.3 RIPng

9.3.1 基础知识

- 使用 UDP521（源 UDP 端口及目的 UDP 端口都是 521）
- 最大跳数依然是 15 跳
- 携带的网络前缀是 128bis，而不是 32bits
- 下一跳地址是 128bits
- 使用 Linklocal 地址作为协议数据包的源地址，发送 RIPng 消息到邻接 RIPng 路由器。使用 FF02::9 作为 RIPng 更新的目标地址
- Distance 默认 120

9.3.2 配置命令

1. **激活 RIPng 进程及接口**

```
router(config)# ipv6 router rip name
router(config)# interface serial0/0
router(config-if)# ipv6 router rip name enable
```

【注意】进程名本地有效，若 Router 两接口，分别启用 RIPng 用的是两个不同的进程名，则两进程互相独立。

2. **默认路由的传递**

```
Ipv6 rip 1 default-information originate    //接口模式下，重发布默认路由进 RIP 进程( 本地无需静态默认路由 )
Ipv6 rip 1 default-information only          //接口模式下，只发布默认路由，禁止发布其他路由信息
```

3. **调整 RIPng 进程**

```
Router(config-rtr)# distance x
!! 修改 RIPng 管理距离，默认 120

Router(config-rtr)# distribute-list prefix-list xxx {in | out} [interface]
```

!! 对某个 RIPng 接口发送或接收路由更新动作执行分发列表。必须跟 in/out 方向，接口可选，如果不配置接口，则为所有接口生效。

```
Router(config-rtr)# poison-reverse
```

执行毒性逆转，默认关闭。

如果同时启用水平分割和毒性逆转，则只有水平分割有效。

```
Router(config-rtr)# split-horizon
```

执行水平分割处理

```
Router(config-rtr)# port  x  multicast-group  X:X:X:X::X
```

修改 RIP 使用的 UDP 端口及组播组地址。默认情况下是 UDP521 及 FF02::9

```
Router(config-rtr)# timers update expire holddown garbage-collect
```

修改计时器。默认 update 30S，expire 是超时时间默认 180S。

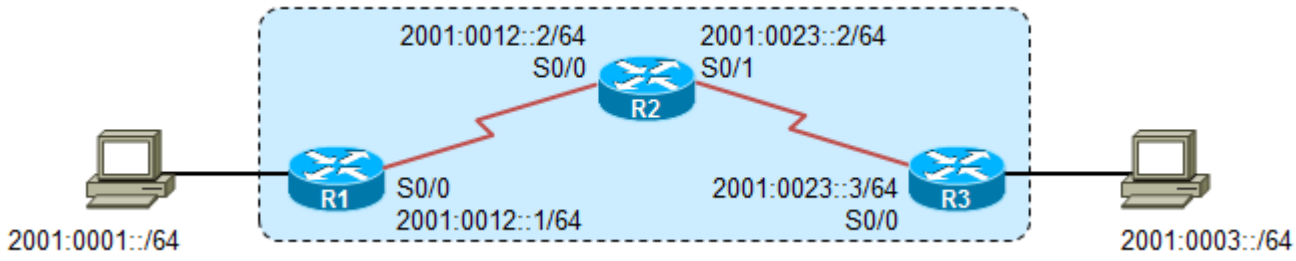
holddown 默认为 0，也就是不使用抑制

garbage-collect 默认 120S

```
Router(config-if)# ipv6 rip xxx metric-offset  ?
```

!! 注意，是在接口模式下配置的，用来设置 metric 累加量。可选值 1-16

9.3.3 配置示例



1. **实验需求**
   - 完成基本 IP 配置
   - 三台路由器完成 RIPng 的配置，要求全网互通
   - PC 无需收到多余的 RIPng 组播消息

- 假设 R1 为网络结构中的末梢节点，因此，在 R2 上，将多余的路由过滤掉，仅下放 R3 的 Fa1/0 口路由

2. 实验配置

R1 的配置如下：

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
  ipv6 rip RIPprocess enable              !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0001::FFFF/64
  ipv6 rip RIPprocess enable
```

R2 的配置如下：

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::2/64
  ipv6 rip RIPprocess enable
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:23::2/64
  ipv6 rip RIPprocess enable
```

R3 的配置如下：

```
Ipv6 unicast-routing
!
```

```
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:23::3/64
  ipv6 rip RIPprocess enable              !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0003::FFFF/64
  ipv6 rip RIPprocess enable
```

完成上述配置后，全网路由就通了。

PC 的配置非常简单，使用无状态自动配置获取地址，例如 PC1（用路由器模拟）：

```
interface FastEthernet0/0
  no ip address
  ipv6 enable
  ipv6 address autoconfig default
```

加上 default 关键字会使得该接口在通过无状态自动配置拿到地址后，安装一条默认路由。

PC1#sh ipv int

```
FastEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE03:1AFF:FEC4:0
  Global unicast address(es):
    2001:1::CE03:1AFF:FEC4:0, subnet is 2001:1::/64 [PRE]
      valid lifetime 2591944 preferred lifetime 604744
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FFC4:0
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
```

```
Default router is FE80::CE00:1AFF:FEC4:10 on FastEthernet0/0
```

好那么继续 ,现在 R1 及 R3 的以太网接口都激活了 RIPng ,于是这两接口都会向 LAN 中去组播 RIPng 报文，而这个动作实际上是没有意义的，因此，我们可以将这个以太网口 passive 掉。但是很可惜，RIPng 里没有定义 passive-interface 的机制。

因此，我们可以在 R1、R3 上，将以太网口（所在网段）重发布进 RIPng。

R1 的配置修改如下（R2 的配置类似）:

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
  redistribute connected                !! 重发布直连
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
  ipv6 rip RIPprocess enable            !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0001::FFFF/64
  no ipv6 rip RIPprocess enable         !! 不激活 RIP，而是采用重发布的方式
```

假设 R1 为网络结构中的末梢节点，因此，在 R2 上，将多余的路由过滤掉，仅给 R1 通告 R3 的 Fa1/0 口路由。我们在 R2 上过滤掉特定的路由：

R2 的配置增加如下：

```
ipv6 prefix-list test seq 5 permit 2001:3::/64
Ipv6 router rip RIPprocess
  distribute-list prefix-list test out serial 0/0
```

为了让 R1 能够正常访问网络其他地方，一劳永逸，给 R1 下发一条 RIPng 的默认路由：

R2 的配置增加如下：

```
Interface serial0/0
  ipv6 rip RIPprocess default-information originate
```

9.4 IPv6 IS-IS

9.4.1 基础知识

现有的 CISCO IOS 的 IS-IS 支持 IPv6，相关 RFC：1195

更新内容主要是新添加的承载有关 IPv6 路由信息的两个 TLV。

- **IPv6 可达性：** 这个 TLV 定义了如 IPv6 路由选择前缀、度量值、及一些选项比特等。分配给 IPv6 可达性 TLV 的十进制值是 236,
- **IPv6 接口地址：** 这个 TLV 包含一个 IPv6 接口地址(128bits),分配给 IPv6 接口地址 TLV 的十进制值是 232，对于 IS HELLO PDU ,这个 TLV 必须包含 Linklocal 地址 ,但是对于 LSP ,TLV 必须包含非 Linklocal 地址。

9.4.2 IPv6 IS-IS 网络设计

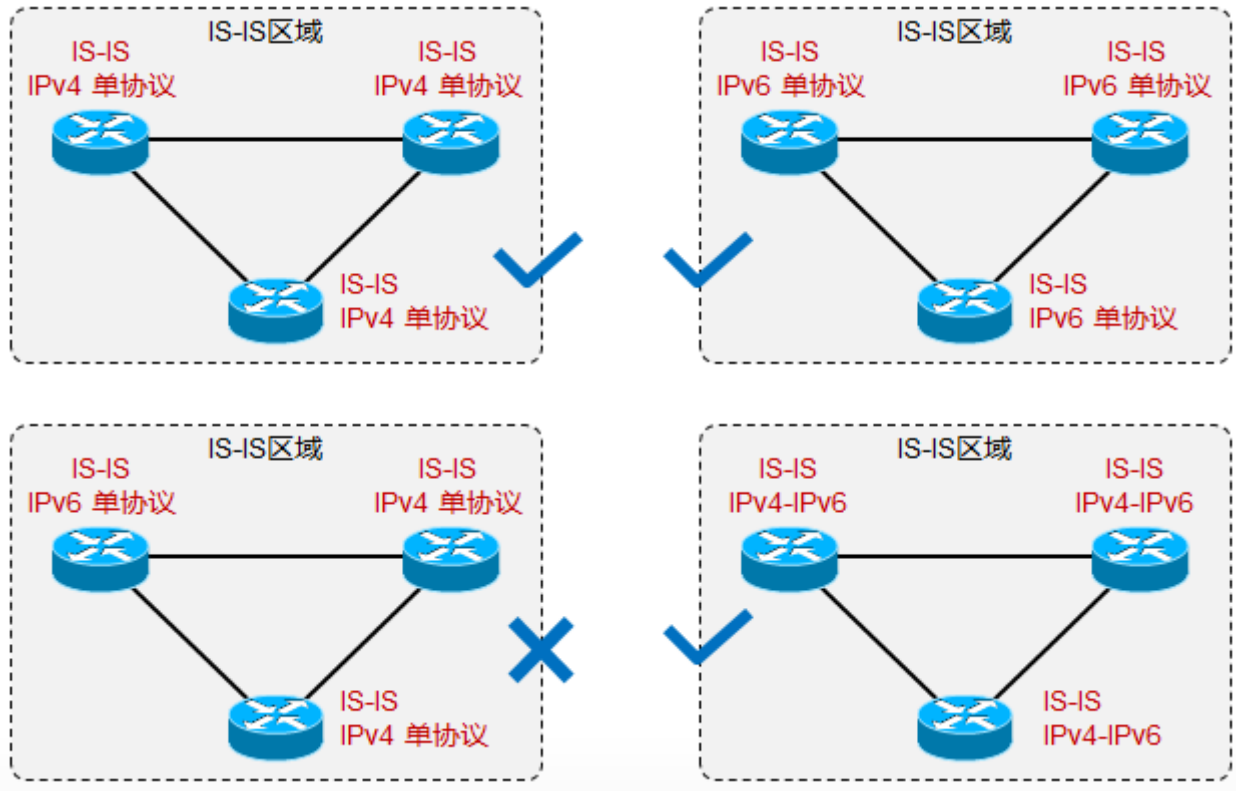
- **层次结构的考虑**

  IS-IS 域基于两层结构 , Level1 及 Level2 , 同时 L1-router、L2-router 及 L1L2-router 的概念 , 通 IPv4 IS-IS。

- **IS-IS 邻接关系的考虑**

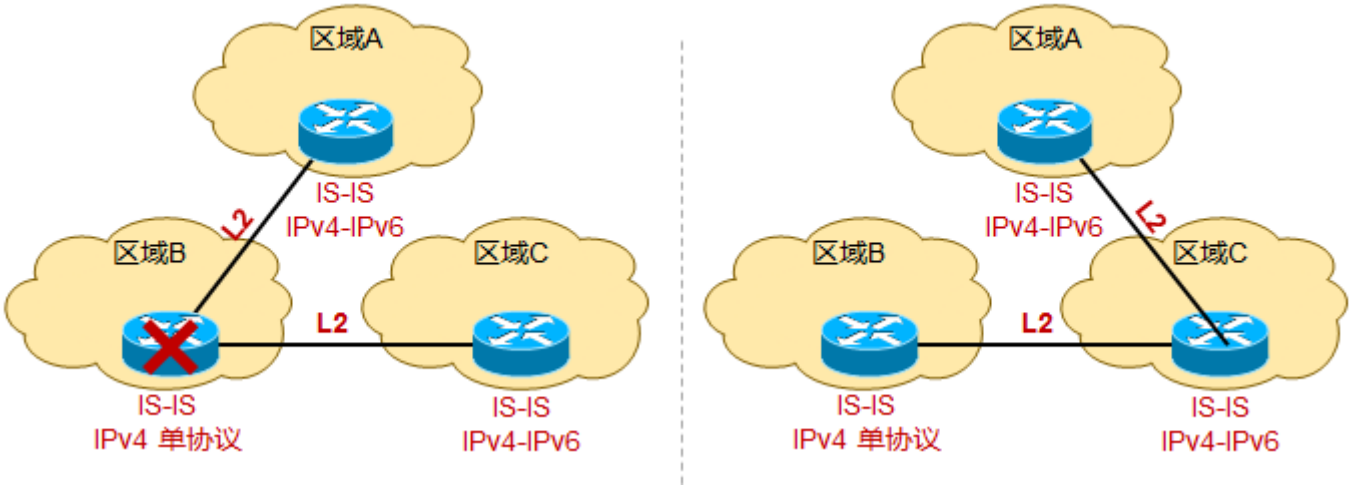
  另外 IS-IS 邻接路由器有个非常重要的因素要考虑。IPv4 或 IPv6 单协议环境下，IS-IS 的邻接关系及信息的交互都不会有问题。但是在所有邻接 IS-IS 路由器上启用两种协议（IPv4 及 IPv6）时，就要谨慎了。在这种特定的情况下，在所有的 IS-IS 邻接路由器上必须同时启用两种协议，否则 IS-IS 路由器将不再维持与其所有 IS-IS IPv4 邻居的邻接关系。事实上，当 Level1 或 Level1-2 IS-IS router 在启用邻接关系检查时，会检查从其 IS-IS 邻居发来的 Hello 消息，如果邻居使用不同的协议集，就拒绝与其形成邻接关系。

  IPv4 单协议网络 IS-IS 路由器向 IPv6-IPv4 路由器的过渡是非常关键的，如果在过渡过程中没有关掉邻接关系检查(关掉邻接关系检查 ,用 no adjacency-check),IPv4 单协议王丽萍 IS-IS 路由器将拒绝与 IPv6-IPv4 路由器形成邻接关系。成功过渡之后，当所有 IS-IS 路由器同时支持两种协议时，就可以去掉 no adjacency-check 命令了。



- **Level2 router 因素**

  在任何 IS-IS 网络中，负责区域间路由选择的 L2 router 必须是连续的，同样，对于 IPv4 单协议环境、IPv6 单协议环境或者是 IPv4-IPv6 环境，L2 router 都必须是连续的。否则会产生路由黑洞。



上面这个图，左侧的网络就有问题，IPv4 单协议的 L2 router 不支持 IPv6，因此破坏了 IPv6 L2 router 的连续性。右侧的网络则是正确的结构。

9.4.3 配置命令

```
Router(config)# router isis [tag]
```

定义 IS-IS 进程

```
Router(config-router)# address-family ipv6 [unicast]
```

进入 IPv6 地址簇

```
Router(config-router)# default-information originate [route-map x]
```

产生一条默认路由

```
Router(config-router)# no adjacency-check
```

在网络从 IPv4 单协议网络过渡到 IPv4-IPv6 IS-IS 路由器的过程中，该命令维持使用不同协议集的 IS-IS 路由器之间的 IS-IS 邻接关系。它可以防止使用不同协议集的 IS-IS 路由器执行 hello 检查而丢失 IS-IS 邻接。在网络切换或者过渡完成之后，可以将 adjacency-check 配置回去。

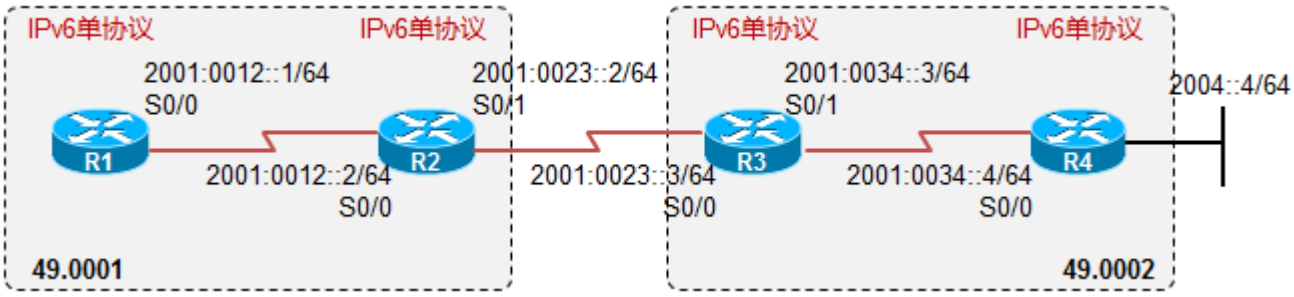
```
Router(config-router)# net XX.XXXX. ... .XXX.XX
```

为本路由器配置一个 NET 地址，这与 IPv4 IS-IS 的概念完全一样

```
Router(config-if)# ipv6 router isis
```

在接口上激活 IPv6 IS-iS

9.4.4 配置示例



1. **实验需求**
   - IS-IS 网络的规划如拓扑所示
   - R4 上的 Loopback 口使用重发布的方式引入 IS-IS
   - 实现全网互通

2. **实验配置**

注意，如果使用虚拟机配置，要主机在 serial 接口上先手工配置 linklocal 地址

R1 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0001.0000.0000.0001.00                !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:12::1/64
 ipv6 address FE80::FFFF:FE10:1 link-local      !!GNS 模拟器环境下，serial 口的 Linklocal 地址冲突，因此手工指定
 ipv6 router isis                               !! 接口激活 IS-IS
 isis circuit-type level-1
```

R2 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0001.0000.0000.0002.00                  !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:12::2/64
 ipv6 address FE80::FFFF:FE10:2 link-local
 ipv6 router isis
 isis circuit-type level-1
interface Serial0/1
 ipv6 enable
 ipv6 address 2001:23::2/64
 ipv6 address FE80::FFFF:FE10:3 link-local
 ipv6 router isis
 isis circuit-type level-2-only
```

R3 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0002.0000.0000.0003.00                !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:23::3/64
 ipv6 address FE80::FFFF:FE10:4 link-local
 ipv6 router isis
 isis circuit-type level-2-only
interface Serial0/1
 ipv6 enable
 ipv6 address 2001:34::3/64
 ipv6 address FE80::FFFF:FE10:5 link-local
 ipv6 router isis
 isis circuit-type level-1
```

R4 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0002.0000.0000.0004.00                !! NET 实体地址
 address-family ipv6
    redistribute connected level-1
 exit-address-family
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:34::4/64
 ipv6 address FE80::FFFF:FE10:6 link-local
 ipv6 router isis
```

```
 isis circuit-type level-1
interface Loopback0
 ipv6 address 2004::4/64
 ipv6 enable
```

在 R2 上简单的查看一下：

R2#sh isis neighbors

```
System Id      Type Interface IP Address      State Holdtime Circuit Id
R3             L2   Se0/1                     UP    27       00
R1             L1   Se0/0                     UP    25       00
```

R2#sh isis neighbors detail

```
System Id      Type Interface IP Address      State Holdtime Circuit Id
R3             L2   Se0/1                     UP    22       00
  Area Address(es): 49.0002
  SNPA: *HDLC*
  IPv6 Address(es): FE80::FFFF:FE10:4
  State Changed: 00:10:04
  Format: Phase V
R1             L1   Se0/0                     UP    29       00
  Area Address(es): 49.0001
  SNPA: *HDLC*
  IPv6 Address(es): FE80::FFFF:FE10:1
  State Changed: 00:11:03
  Format: Phase V
```

继续在 R4 上进行简单的查看：

R4#sh isis database

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R3.00-00              0x0000000E   0xA440        835               1/0/0
R4.00-00            * 0x0000000B   0xF895        729               0/0/0
IS-IS Level-2 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R4.00-00            * 0x00000005   0x7768        850               0/0/0
```

R4#sh isis da R4.00-00 l1 detail

```
IS-IS Level-1 LSP R4.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R4.00-00              * 0x0000000B   0xF895        696               0/0/0
  Area Address: 49.0002
  NLPID:        0x8E
  Hostname: R4
  IPv6 Address: 2001:34::4
  Metric: 10         IPv6 2001:34::/64
  Metric: 10         IS R3.00
  Metric: 0          IPv6 2004::/64
```

9.4.5 高级配置

1. **在 GRE 隧道之上配置 IPv6 IS-iS**

   因为 IS-IS 运行在数据链路层之上，也就是说，运行在链路层协议之上，并且需要 CLNP 的支持。所以如果在通过配置隧道连接的远端 IPv6 网络上不能使用 IPv6 IS-IS。配置隧道就是静态的 IPv6 over IPv4 隧道，在 IPv4 的数据包之上承载 IPv6 数据包。

   如果希望用隧道技术连接两个 IPv6 IS-IS 岛屿，那么需使用 GRE 隧道。

9.5 OSPFv3（RFC2740）

9.5.1 基础知识

OSPF V2 与 V3 有相似之处：

- 使用相同的基本数据包类型，如 Hello、DBD、LSR、LSU、和 LSA
- 邻居发现和邻接关系形成机制是相同的
- 支持在遵循 RFC 的 NBMA 和点到多点拓扑模式上的 OSPV3 操作。OSPFv3 也支持 CISCO 的其他模式，例如点到点等

- LSA 泛洪和衰老机制是相同的

OSPF V2 与 V3 的不同之处：

- OSPFv3 运行在链路上，每条链路有可以有多个 OSPFv3 实例
- OSPFv3 的 RouterID 还是 32bits 的，如果路由器上没有配置 IPv4 地址，则在 OSPFv3 进程中需手工指定 RouterID
- LinkID 在 OSPFv3 中仍然是 32bits 的
- OSPFv3 使用 IPv6 Linklocal 地址标识 OSPFv3 邻接的邻居
- 新的 LSA 类型：
  - 链路 LSA（LSA 类型 0x0008）: 每条链路都有一个链路 LSA，这个新类型提供了路由器的本地链路地址，并列出了链路的所有 IPv6 前缀
  - 区内前缀 LSA（LSA 类型 0x2009）:
- OSPFv3 使用组播地址：FF02::5 及 FF02::6
- 在认证这块，不使用 OSPFv2 中定义的认证方法。而是使用 AH 及 ESP 扩展头部作为安全机制

9.5.2 OSPFv3 报文

IPv6 报头中的协议号仍然是 89；

OSPFv3 的报头将 v2 中的认证相关字段都去除了，这是因为 ipv6 报头已经有了很好的认证功能。

1. **Option 字段**

变成了 24 位，出现在 HELLO 包、DBD 包，以及 router LSA、network LSA、interAreaRouter LSA、Link LSA 中。这些字段都有特殊的用途。

```
Options: 0x000033 (DC, R, E, V6)
  .... .... .... 0... .... .... = F: F is NOT set
  .... .... .... .0.. .... .... = I: I is NOT set
  .... .... .... ..0. .... .... = L: L is NOT set
  .... .... .... ...0 .... .... = AF: AF is NOT set
  .... .... .... .... ..1. .... = DC: DC is SET
  .... .... .... .... ...1 .... = R: R is SET
  .... .... .... .... .... 0... = N: N is NOT set
  .... .... .... .... .... .0.. = MC: MC is NOT set
  .... .... .... .... .... ..1. = E: E is SET
  .... .... .... .... .... ...1 = V6: V6 is SET
```

DC	是否支持按需电路
R	路由器位，表示通告着是否为一台路由器，如果为 0 则该通告者不能路由数据。经过该通告者的路由不能纳入路由计算

N	一个接口所属区域为 NSSA 时，为 1
MC	描述路由器是否运行了 MOSPF
E	是否能处理 AS-external-lsa，一般是 ASBR
V6	表示路由器是否在运行 IPV6，运行 OSPFv3 的路由器发出来的报文，该字段一般都置 1

更多的 IPv6 OSPF 全面解析，请见《红茶三杯 OSPF 技术笔记》

9.5.3 OSPFv3 配置

Router(config)# ipv6 router ospf *process-id*

启动一个 OSPFv4 进程

Router(config-router)# router-id x.x.x.x

指定一个 32bits 的 Router-ID

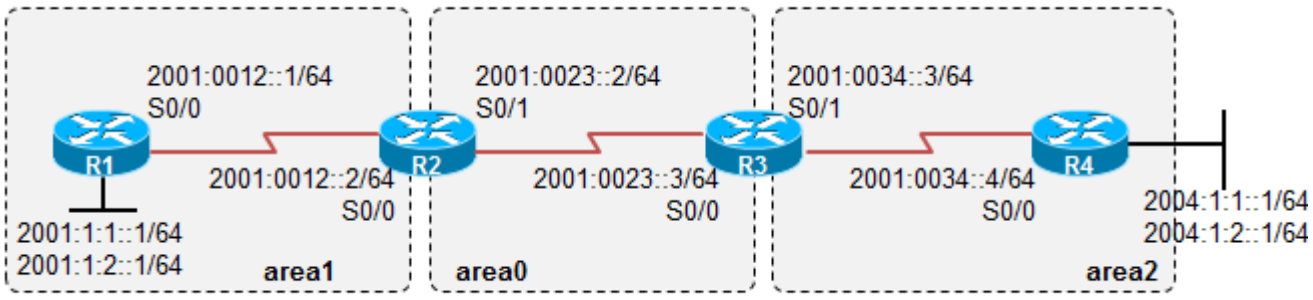
Router(config-router)# area *area-id* **range** *xxxxx*

路由汇总

Router(config-if)# ipv6 ospf *process-id* **area** *area-id*

在接口上激活 OSPFv3

9.5.3.1 基本配置示例



1. **实验需求**
   - 完成基本 IP 配置
   - 根据如果所示运行 OSPFv3
   - 对 area1 内的 IPv6 前缀进行汇总，减少路由条目

- 对 R4 重发布进来的 IPv6 前缀进行汇总，减少路由条目

2. 实验配置

R1 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
Interface loopback0
  ipv6 enable
  ipv6 address 2001:1:1::1/64
  ipv6 address 2001:1:2::1/64
  exit
!
Ipv6 router ospf 100
  router-id 1.1.1.1
  exit
interface serial0/0
  ipv6 ospf 100 area 1
interface loopback0
  ipv6 ospf 100 area 1
```

R2 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::2/64
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:23::2/64
!
Ipv6 router ospf 100
  router-id 2.2.2.2
```

```
  exit
interface serial0/0
  ipv6 ospf 100 area 1
interface serial0/1
  ipv6 ospf 100 area 0
```

R3 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:23::3/64
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:34::3/64
!
Ipv6 router ospf 100
  router-id 3.3.3.3
  exit
interface serial0/0
  ipv6 ospf 100 area 0
interface serial0/1
  ipv6 ospf 100 area 2
```

R4 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:34::4/64
interface loopback0
  ipv6 enable
  ipv6 address 2004:1:1::1/64
  ipv6 address 2004:1:2::1/64
!
Ipv6 router ospf 100
```

```
 router-id 4.4.4.4
 redistribute connected                !!重发布直连路由
 exit
interface serial0/0
 ipv6 ospf 100 area 2
```

到此为止，初步的配置都已完成了。

R1 上看看：

R1#show ipv6 ospf neighbor detail

```
 Neighbor 2.2.2.2
     In the area 1 via interface Serial0/0
     Neighbor: interface-id 4, link-local address FE80::FFFF:FE10:2
     Neighbor priority is 1, State is FULL, 6 state changes
     Options is 0x6423B8FD
     Dead timer due in 00:00:31
     Neighbor is up for 00:05:46
     Index 1/1/1, retransmission queue length 0, number of retransmission 0
     First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
     Last retransmission scan length is 0, maximum is 0
     Last retransmission scan time is 0 msec, maximum is 0 msec
```

R1#sh ipv6 route ospf

```
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI   2001:23::/64 [110/128]
      via FE80::FFFF:FE10:2, Serial0/0
OI   2001:34::/64 [110/192]
      via FE80::FFFF:FE10:2, Serial0/0
OE2   2004:1:1::/64 [110/20]
      via FE80::FFFF:FE10:2, Serial0/0
OE2   2004:1:2::/64 [110/20]
```

```
        via FE80::FFFF:FE10:2, Serial0/0
```

R4#show ipv6 route ospf

```
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI   2001:1:1::1/128 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:1:2::1/128 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:12::/64 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:23::/64 [110/128]
       via FE80::FFFF:FE10:5, Serial0/0
```

接下去在 R2 上对 R1 的 Loopback 路由进行汇总：

R2 的配置增加如下：

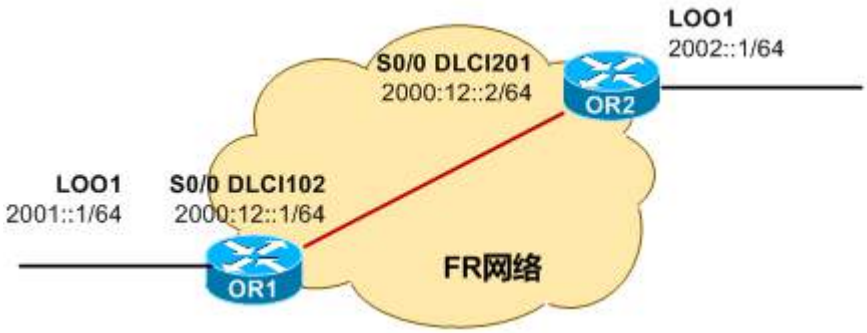
```
ipv6 router ospf 100
area 1 range 2001:1::/32
```

接下去在 R4 上，对其自身重发布进来的外部路由做汇总：

R4 的配置增加如下：

```
Ipv6 router ospf 100
  summary-prefix 2004:1::/32
```

9.5.3.2 帧中继环境下的问题



OR1 的配置

```
ipv6 unicast-routing
interface Serial0/0
encapsulation frame-relay
  no frame-relay inverse-arp
  ipv6 enable
  ipv6 address 2000:12::1/64
  ipv6 ospf network point-to-point
  ipv6 ospf 1 area 0
  frame-relay map ipv6 2000:12::2 102 broadcast
```

OR2 的配置

```
ipv6 unicast-routing
interface Serial0/0
encapsulation frame-relay
  no frame-relay inverse-arp
  ipv6 enable
  ipv6 address 2000:12::2/64
  ipv6 ospf network point-to-point
  ipv6 ospf 1 area 0
  frame-relay map ipv6 2000:12::1 201 broadcast
```

完成此步骤后，OR1 及 OR2 能建立起 OSPF 邻居关系，互相之间的全局 IPV6 单播地址也都能 ping 通

此时分别在 OR1 及 OR2 上开启 loopback 接口，并配置 Ipv6 地址，然后宣告进 OSPFv3

OR1 的配置

```
interface Loopback0
  ipv6 address 2001::1/64
```

```
 ipv6 enable
 ipv6 ospf 1 area 1
```

OR2 的配置

```
interface Loopback0
 ipv6 address 2002::1/64
 ipv6 enable
 ipv6 ospf 1 area 2
```

完成后，**OR1 及 OR2 都能学习到对方的 Loopback 接口路由，但是却无法 ping 通。**

查看 OR1 的路由表：

```
OI    2002::1/128 [110/64]
       via FE80::5C10:8CFF:FEE0:FE89, Serial0/0
```

发现去往 OR2 loopback 接口的路由，下一跳是 FE80::5C10:8CFF:FEE0:FE89，也就是 OR2 接口的链路本地地址，原来 Ipv6 环境下，一个接口往往具有多个 Ipv6 地址，而 OSPF 邻居关系的维护又以稳定为前提，每个接口都必备的链路本地地址，就成了建立邻居关系最好的一句，那么为什么 ping 不通呢？**正是由于这是个帧中继的环境，FE80::5C10:8CFF:FEE0:FE89 这个 IPV6 地址，OR1 上并没有做映射，OR2 上同理，因此分别添上各自对端的链路本地地址的帧中继映射即可。**

如 OR1，OR2 同理

```
frame-relay map ipv6 FE80::5C10:8CFF:FEE0:FE89 102 broadcast
```

9.6 MP-BGP

9.6.1 多协议 BGP 概述

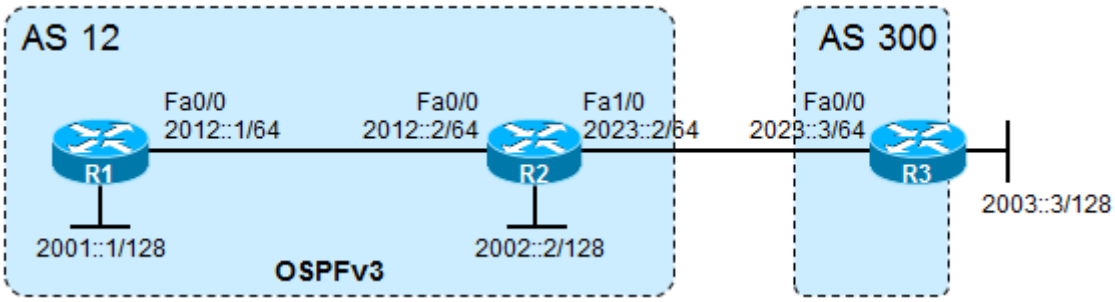
BGP4+，也叫做多协议 BGP，扩展 BGP-4 规范以支持诸如 IPv6、IPX、VPN 等这样的新地址簇。因此 BGP4+能够为 IPv6 和包括 IPv4 在内的其他协议携带路由选择信息。RFC2858 和 RFC2545 定义了在 BGP4+中处理 IPv6 而更新的属性。

下面是 BGP-4 规范中为了支持 IPv6 而进行了更新的属性：

- NEXT_HOP：表示为 IPv6 地址，或者是一个可聚合全球单播地址，或者是一个可聚合全球单播地址及其下一跳的本地链路地址
- NLRI： 是一组目的地，表示为一个 IPv6 前缀。

9.6.2 多协议 BGP 的基础配置

9.6.2.1 使用 IPv6 全局单播地址建 BGP4+邻居、传递 IPv6 路由



实验环境：

- R1、R2 处于 AS12，AS 内运行 OSPFv3，使得 R1、R2 能够学习到对方的 Loopback 路由
- R1、R2 建立 IPv6 的 IBGP 邻居关系（基于 LOOPBACK 接口）
- R2、R3 建立 IPv6 的 EBGP 邻居关系，R3 重发布直连路由，使得 R1、R2 能够学习到

实验配置：

R1 的配置如下（基本配置及 OSPFv3）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2012::1/64
   ipv6 nd suppress-ra                    !!由于 Fa0/0 连接的是路由器而非主机，为减少不必要的 ra 的
   no shutdown
interface loopback 0
   ipv6 enable
   ipv6 address 2001::1/128
!
Ipv6 router ospf 1
   router-id 1.1.1.1
!
interface fastEthernet 0/0
   ipv6 ospf 1 area 0
```

```
interface loopback 0
   ipv6 ospf 1 area 0
```

R2 的配置如下（基本配置及 OSPFv3）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2012::2/64
   ipv6 nd suppress-ra
   no shutdown
interface fastEthernet 1/0
   ipv6 enable
   ipv6 address 2023::2/64
   ipv6 nd suppress-ra
   no shutdown
interface loopback 0
   ipv6 enable
   ipv6 address 2002::2/128
!
Ipv6 router ospf 1
   router-id 2.2.2.2
!
interface fastEthernet 0/0
   ipv6 ospf 1 area 0
interface loopback 0
   ipv6 ospf 1 area 0
```

R3 的配置如下（基本配置）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2023::3/64
   ipv6 nd suppress-ra
   no shutdown
```

```
interface loopback 0
  ipv6 enable
  ipv6 address 2003::3/128
```

接下去开始配置 BGP4+：

R1 的 BGP4+配置如下：

```
router bgp 12
  bgp router-id 1.1.1.1
  no bgp default ipv4-unicast                    !!如果不配置此命令，则一旦指定 BGP 邻居后，默认 ipv4 邻居就会建立起来，本实验中路由器之间不传递 IPv4 前缀因此无需 IPv4 的连接。
  neighbor 2002::2 remote-as 12
address-family ipv6
  neighbor 2002::2 activate
  neighbor 2002::2 update-source loopback 0
  exit-address-family
```

R2 的 BGP4+配置如下：

```
router bgp 12
  bgp router-id 2.2.2.2
  no bgp default ipv4-unicast
  neighbor 2001::1 remote-as 12
  neighbor 2023::3 remote-as 300
address-family ipv6
  neighbor 2001::1 activate
  neighbor 2023::3 activate
  neighbor 2001::1 update-source loopback 0
  exit-address-family
```

R3 的 BGP4+配置如下：

```
router bgp 300
  bgp router-id 3.3.3.3
  no bgp default ipv4-unicast
  neighbor 2023::2 remote-as 12
address-family ipv6
```

```
 neighbor 2023::2 activate
 exit-address-family
```

简单的查看一下：

R1#show ip bgp ipv6 unicast summary

```
BGP router identifier 1.1.1.1, local AS number 12
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2002::2         4    12       3       3        1    0    0 00:00:39        0
```

R2#show ip bgp ipv6 unicast summary

```
BGP router identifier 2.2.2.2, local AS number 12
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001::1         4    12      22      22        1    0    0 00:19:44        0
2023::3         4   300       6       6        1    0    0 00:01:32        0
```

现在，在 R3 上，将直连 loopback 路由重发布进 BGP4+

```
ipv6 prefix-list loopb permit 2003::3/128
route-map test permit 10
 match ipv6 address prefix-list loopb
!
router bgp 300
 address-family ipv6
  redistribute connected route-map test
```

现在 R2 能学习到这条 IPv6 路由：

R2#sh ip bgp ipv6 unicast

```
BGP table version is 2, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 2003::3/128      2023::3                  0             0 300 ?
```

但是在 R1 上：

R1#sh ip bgp ipv6 unicast

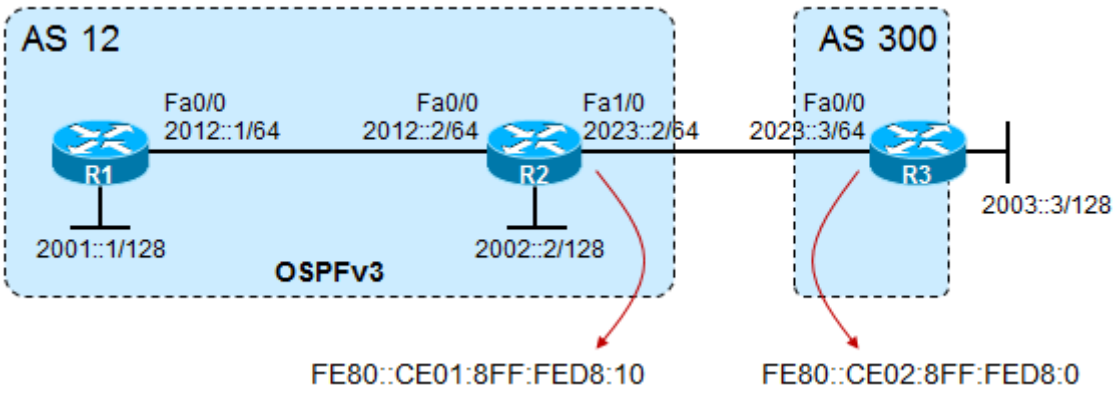
```
BGP table version is 1, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i2003::3/128      2023::3                  0    100      0 300 ?
```

我们发现在 R1 上，路由并不 best，很简单，因为 NH 不可达。解决的办法很简单，在 R3 上对 R2 做 next-hop-self 即可：

```
router bgp 12
  address-family ipv6
    neighbor 2001::1 next-hop-self
    exit-address-family
```

9.6.2.2 使用 IPv6 Linklocal 地址建 BGP4+邻居关系、传递 IPv6 路由

在某些环境下，可以使用 Linklocal 地址来建立 BGP 邻接关系，这样做的好处之一是，无需为这些链路分配可聚合全球 IP 地址。在 BGP 中使用 Linklocal 地址有一些注意事项，例如要确定目的 Linklocal 地址相对应的路由器物理接口，因为 Linklocal 地址只是本地有效。在一个是结合 route-map 修改 NEXT_HOP 属性。



如上图，我们在上一个实验的基础上，在 R2、R3 之间使用 Linklocal 地址来建立邻居关系：

其中 R2 的配置如下：

```
router bgp 12
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 neighbor 2001::1 remote-as 12
 neighbor 2001::1 update-source Loopback0
 neighbor FE80::CE02:8FF:FED8:0 remote-as 300
 neighbor FE80::CE02:8FF:FED8:0 update-source FastEthernet1/0      !!注意这条命令一定要配，因为linklocal 地址只是 link 范围内有效，因此需要指定本地与这个 linklocal 邻居地址对接的接口。
 !
 address-family ipv6
 neighbor 2001::1 activate
 neighbor 2001::1 next-hop-self
 neighbor FE80::CE02:8FF:FED8:0 activate
 exit-address-family
```

其中 R3 的配置如下：

```
 router bgp 300
 bgp router-id 3.3.3.3
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor FE80::CE01:8FF:FED8:10 remote-as 12
 neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
 !
 address-family ipv6
 neighbor FE80::CE01:8FF:FED8:10 activate
 redistribute connected route-map test
 no synchronization
 exit-address-family
```

注意，在 R2 上我们是对 R1 做了 next-hop-self，因此从 AS300 传递过来的路由 NH 发生了改变。但是在某些网络环境中，使用 Linklocal 建立 BGP 邻接关系可能导致 NH 的一些问题，毕竟 Linklocal 地址只是链路内有效，因此，可以搭配 route-map，使用 set ipv6 next-hop 来修改这些 BGP 路由的 NH 属性。

9.6.3 其他配置

1. 应用前缀列表

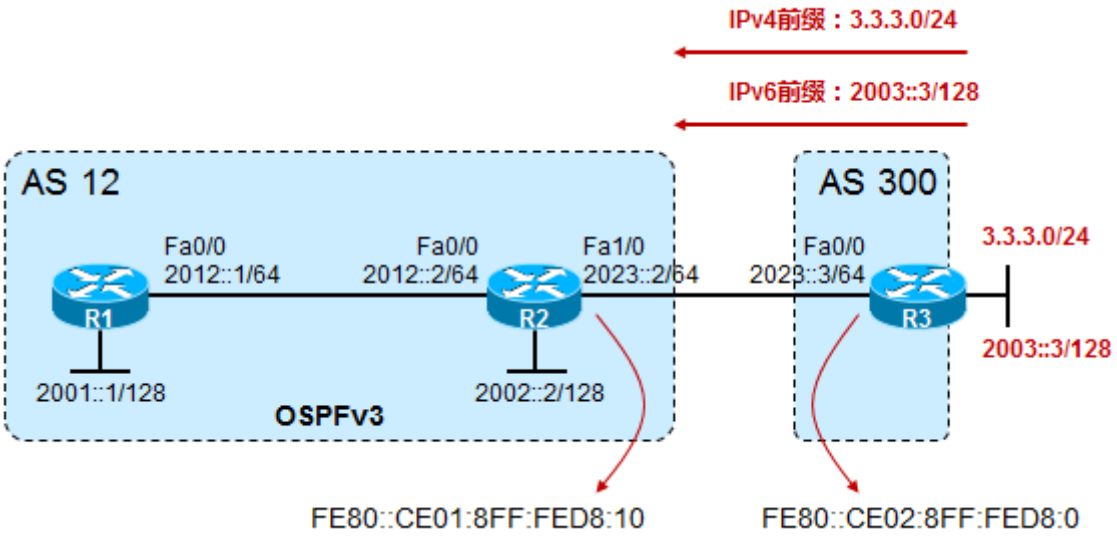
```
ipv6 prefix-list loopb permit 2003::3/128
router bgp 12
  address-family ipv6
    neighbor 2023::2 prefix-list loopb out
  exit-address-family
```

注意，过滤的命令是在 address-family ipv6 中进行配置

2. 在 IPv6 BGP 对等体之间传递 IPv4 路由

有这么一种网络环境：两个 IPv4 网络通过纯 IPv6 提供商连接，那么这时就不得不通过 BGP IPv6 对等体交换 IPv4 路由。因为 BGP4+支持 IPv4 及 IPv6 协议簇，因此这是完全没有问题的。

但是需格外关注 next-hop 的可达性问题。



上面的实验环境中，R2、R3 是使用 Linklocal 地址建立 BGP 邻居关系。R3 所在的 AS300 内 IPv6 的路由前缀，也有 IPv4 的路由前缀，要传给 R2。

那么 R2 的配置如下：

```
router bgp 12
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 2001::1 remote-as 12
```

```
neighbor 2001::1 update-source Loopback0
neighbor FE80::CE02:8FF:FED8:0 remote-as 300
neighbor FE80::CE02:8FF:FED8:0 update-source FastEthernet1/0
!
address-family ipv4
neighbor FE80::CE02:8FF:FED8:0 activate
no auto-summary
no synchronization
exit-address-family
!
address-family ipv6
neighbor 2001::1 activate
neighbor 2001::1 next-hop-self
neighbor FE80::CE02:8FF:FED8:0 activate
exit-address-family
```

R3 的配置如下：

```
router bgp 300
 bgp router-id 3.3.3.3
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor FE80::CE01:8FF:FED8:10 remote-as 12
 neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
 !
 address-family ipv4
 neighbor FE80::CE01:8FF:FED8:10 activate
 no auto-summary
 no synchronization
 network 3.3.3.0 mask 255.255.255.0
 exit-address-family
 !
 address-family ipv6
 neighbor FE80::CE01:8FF:FED8:10 activate
 redistribute connected route-map test
 no synchronization
```

```
 exit-address-family
```

完成配置后，R2 及 R3 之间的 BGP 连接就能够同时传递 IPv4 及 IPv6 的路由前缀。

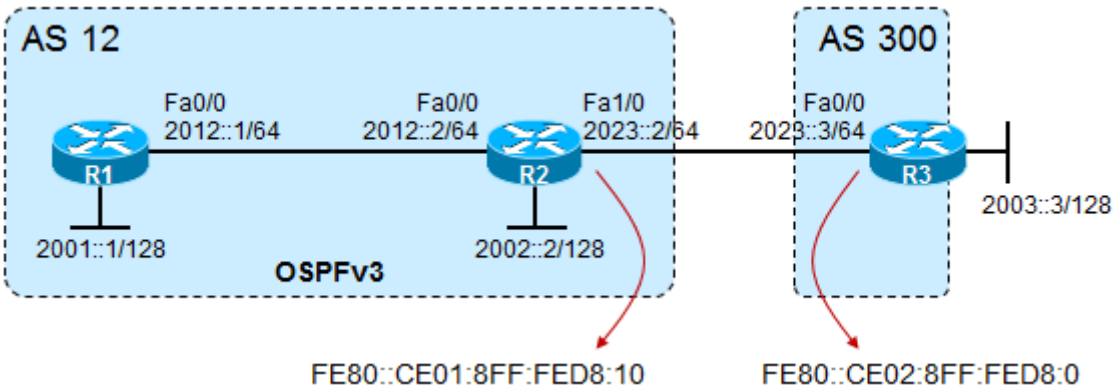
但是我们发现，R2 上，已经能够学习到 R3 传递过来的 IPv6 前缀，但是 IPv4 前缀始终没有收到。在 R2 上，还能看到如下报错：

```
*Mar  1 17:40:49.491: %BGP-6-NEXTHOP: Invalid next hop (254.128.0.0) received from FE80::CE02:8FF:FED8:0: martian next hop
R2#
*Mar  1 17:40:49.495: BGP(0): FE80::CE02:8FF:FED8:0 rcv UPDATE w/ attr: nexthop 254.128.0.0, origin i, metric 0, originator 0.0.0.0, path 300, community , extended community
*Mar  1 17:40:49.499: BGP(0): FE80::CE02:8FF:FED8:0 rcv UPDATE about 3.3.3.0/24 -- DENIED due to: martian NEXTHOP;
```

原来，R3 传递过来的 3.3.3.0/24 的 IPv4 前缀，NH 为 254.128.0.0，是一个保留的地址，该地址当然是不合法的，因此 R2 直接忽略这条路由。解决的办法，可以在 R3 上对 R2 做 out 方向的 route-map，并且设置 IPv4 前缀的 NH，如：

```
route-map setNH permit 10
 set ip next-hop 10.1.23.3
 !
router b 300
address-family ipv4
neighbor FE80::CE01:8FF:FED8:10 route-map setNH out
```

3. 和 BGP4+一起使用 MD5



R2 和 R3 之间用 MD5 认证，

R3 的配置如下：

```
router bgp 300
 bgp router-id 3.3.3.3
```

```
no bgp default ipv4-unicast
bgp log-neighbor-changes
neighbor FE80::CE01:8FF:FED8:10 remote-as 12
neighbor FE80::CE01:8FF:FED8:10 password cisco
neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
!
address-family ipv4
.....
exit-address-family
!
address-family ipv6
neighbor FE80::CE01:8FF:FED8:10 password cisco
neighbor FE80::CE01:8FF:FED8:10 activate
redistribute connected route-map test
no synchronization
exit-address-family
R3#
```

10 IPv6 基础实验

10.1 Windows 主机篇

1. **安装 IPv6 协议栈**

   Windows7 的系统，默认就已经安装了 IPv6 协议栈。

   WindowsXP 的系统，需要手工安装，方法很简单，进入 CMD 界面操作：

```
Ipv6 install                          !! 安装完后重启，默认开启无状态自动获取地址
ipv6 if                               !! 查看 ipv6 接口配置信息
ping IPv6Address [%ZoneID]            !! ping 链路本地地址和站点本地地址可能要加索引号，ping 全局单播地址不需要
```

2. **常用 IPv6 命令**

   netsh 可以用来配置、管理、重置 IPV4 V6 协议栈

netsh interface ipv6 show ?

可以查看 IPv6 配置的各项参数。

netsh interface ipv6 show interface

查看接口（网卡，包括虚拟网卡）索引及名称，等同于 WindowsXP 下的 ipv6 if x

netsh interface ipv6 show address

查看网卡 ipv6 地址信息，如有效期、prefered 生存期等等 （ipconfig 也可）

netsh interface ipv6 show neighbors

相当于看 ARP 表，其实是邻居缓存表

netsh interface ipv6 show route

查看 IPv6 路由

netsh interface ipv6 add address *接口索引号 ipv6 地址*

为接口添加 ipv6 地址

例如：

netsh interface ipv6 show interface !!得到目标接口索引号

netsh interface ipv6 add address 21 3ffe::1 unicast !!假设要配置的接口索引号为 21，则这条命令是为该接口配置一个 IPv6 单播地址。

netsh interface ipv6 add route ::/0 “本地连接” 2010::1

添加路由 或 ipv6 rtu

例如：

netsh interface ipv6 add route 3333::/64 21 nexthop=fe80::1 !!添加一条目的为 3333::/64 的 IPv6 路由，关联到索引号为 21 的接口，路由的下一跳是 fe80::1

netsh interface ipv6 reset

删除用户已配置的所有设置。需要重新启动计算机之后默认设置才能生效

3. IPv6 基本应用

IPv6 主机（xp 系统 ie6.0）访问 IPv6 WEB 站点

IE6.0 暂不支持地址栏输入 rfc2732 定义的 literal ipv6 address 的方式访问 IPv6 站点。

解决办法，通过域名来访问，修改 hosts 文件 **C:/windows/system32/drivers/etc/hosts**

```
# 102.54.94.97      rhino.acme.com          # source server
# 38.25.63.10       x.acme.com              # x client host

127.0.0.1           localhost
::1   test.com
```

10.2 路由器基本配置

常用基本配置

```
Router(config)#ipv6 unicast-routing          !! 打开接口之间的 IPv6 数据包转发功能
Router(config)#intertface f0/1
Router(config-if)#ipv6 enable                !! 接口激活 IPv6
Router(config-if)#ipv6 adress 2001::1/64     !! 配置一个 IPv6 地址
Router(config-if)#ipv6 adress 2002::/64 eui-64    !! 又配置了一个 IPv6 地址，并且使用 EUI-64 填充
```

10.3 无状态自动配置

RFC2462 所定义，即插即用

利用 RS 及 RA 报文。

RS 报文由主机主动发起，RA 报文由主机默认以 200S 周期性发送（收到 RS 也会立刻发送）。

GW 的配置

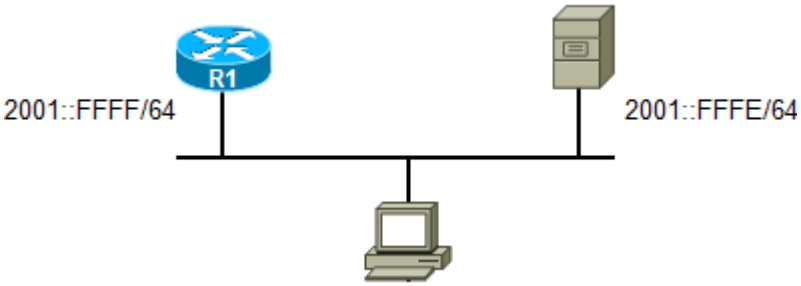
```
ipv6 unicast-routing
interface fast0/0
  ipv6 address 2001::1/64
```

```
 no ipv6 nd suppress-ra                    !! 接口地址 2001::1/64， 同时开启路由器通告（默认关闭）
```

客户端的配置

```
ipv6 address autoconfig [default]          !! 使用 EUI64 地址构建 Ipv6 地址；
                                           !! 如果加 default 关键字，则会在获取到地址的接口上添加一个默认
网关（默认路由）
```

10.4 有状态自动配置



这里的示例是 DHCPv6，使用 Router 作为 DHCPserver

DHCPserver 的配置

```
ipv6 unicast-routing                                  !! 这条必须敲
ipv6 dhcp pool DHCP-pool
 prefix-delegation pool dhcppool lifetime 1800 600    !! 调用 ipv6 local pool 并设定 lifetime
 dns-server 2000::8
 domain-name HelloWord
ipv6 local pool dhcppool 2001::/64 64                 !! 定义准备通告的 Ipv6 前缀

interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFE/64
 ipv6 nd other-config-flag
 ipv6 nd managed-config-flag
 ipv6 dhcp server DHCP-pool      !! 在接口上开启 ipv6 DHCP，并调用池
```

PC 的配置

```
interface FastEthernet0/0
```

```
ipv6 enable                          !! 接口激活 IPV6
ipv6 dhcp client pd test             !! 配置 DHCP client，并且指定获取到的前缀名称为 test（本地有效）
ipv6 address test ::/64 eui-64       !! 使用获取到的前缀（test），加上本接口的 EUI64，构成接口全局 ipv6 地址
```

当一个主机收到 RA 路由器通告的时候，它会去查看 RA 里 M 位，如果M位为 1，则开始在 LAN 上寻找 DHCP 服务器。主机将使用 linklocal 地址作为源发送 DHCP-solicit message，目的是 ALL_DHCP_Agents，

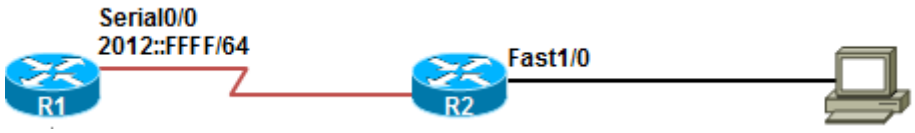
以下是 DHCPv6 可能使用到的地址及端口：

FF02::1:2 = ALL DHCP Agents（servers or relays,link-local scope）

FF05::1:3 = ALL DHCP servers（Site-local scope）

DHCP 报文 client 侦听 UDP546；servers 及 agents 侦听 UDP547

10.5 DHCP PD



R1 的配置如下：

```
ipv6 dhcp pool DHCP-pool
   prefix-delegation pool dhcppool lifetime 1800 600
   dns-server 2000::8
   domain-name HelloWord
ipv6 local pool dhcppool 2001::/64 64
interface Serial0/0
   ipv6 address 2012::FFFF/64
   ipv6 dhcp server DHCP-pool
```

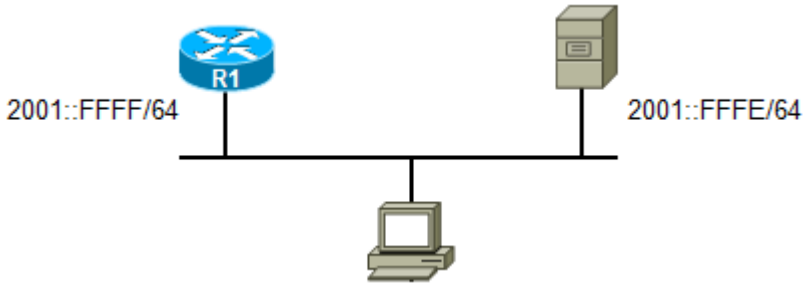
R2 的配置如下：

```
interface Serial0/0
   ipv6 address autoconfig default
   ipv6 dhcp client pd test
Interface fast1/0
   ipv6 address test    ::FFFF/64
```

R1 上有个地址池，如果 R2 希望使用 R1（IPv6 DHCPserver）的地址池分配给自己内网的用户，那么可以使

用 DHCP PD 特性。在 R2 上，这个 test，作为一个 PD 标示符本地有效。

10.6 混合的自动配置实验



R1 的配置如下

```
interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFF/64
 no ipv6 nd suppress-ra
 ipv6 nd other-config-flag
```

DHCPserver 的配置

```
ipv6 dhcp pool DHCP-pool
 dns-server 2000::8
 domain-name HelloWord
 exit
interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFE/64
 no ipv6 nd suppress-ra
 ipv6 dhcp server DHCP-pool
```

11 IPv6 Multicast

11.1 基本配置

ipv6 multicast-routing

全局命令，开启 IPv6 multicast-routing，打开该开关后，所有激活 IPv6 的接口自动打开 IPv6 PIM

Ipv6 mld join-group ff04::1

接口级命令，用于加入组播组，也就是说该接口成为该组播组的成员

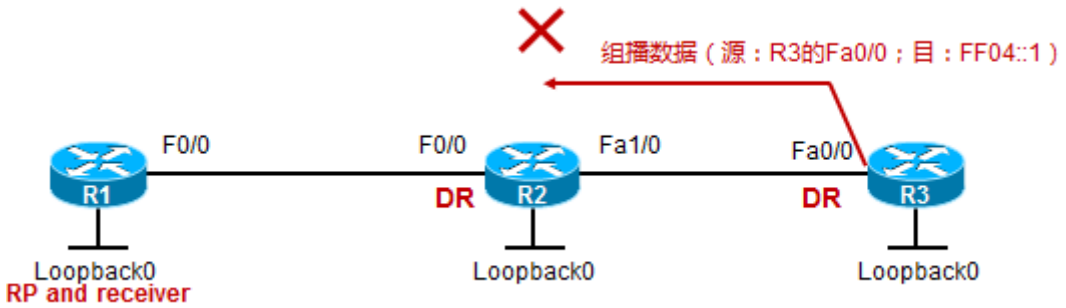
Ipv6 pim rp-address X:X:X:X::X

全局命令，静态指定 RP

关于 IPv6 multicast 的全面详解，请见《红茶三杯 Multicast 技术笔记》

11.2 关于 DR 理解的一个小实验

【测试1】



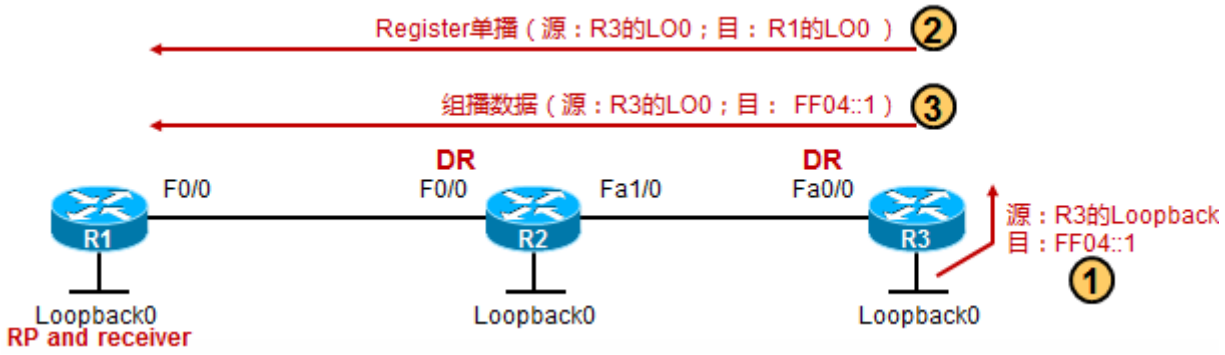
测试描述： 在 R3 上 ping FF04::1，出接口为 Fa0/0

测试现象： 无法 ping 通。

现象分析： R3 直接产生组播数据，这些组播数据的源 IP 为 R3 Fa0/0 的接口 IP，目的 IP 为组播组地址 FF04::1。这些组播数据是直接发送给了 R2，此时此刻，R2 是 First-hop 路由器。那么当 R2 收到这些组播数据的时候，由于是在一个多路访问网络中收到的（在自己的 Fa1/0 口），而自己的 Fa1/0 口又不是该 LAN 内的 DR，既然不是 DR，就无权向 RP 发起 register。因此，R2 只是简单的创建一个 IPv6 组

播转发表项而已（且 outgoing interface list 为空），不会做进一步的处理。于是这些组播数据在 R2 处被丢弃。

【测试2】



测试描述： 在 R3 上 ping FF04::1，出接口为本地的 Loopback0

测试现象： 能够 ping 通

现象分析： 这次环境跟前面就不大一样了。首先组播数据是来自 R3 的 Loopback0 口，那么 First-hop 路由器就变成了 R3，而不是 R2 了。那么 R3 作为第一跳路由器，在收到组播源（也就是自己的 LO0）发出的组播数据后，R3 将触发 Register 过程，register 过程的详细内容，大家都非常熟悉了，这里就不罗嗦了，最终的结果是，R3 到 R1 构建了一个 SPF，于是乎来自 R3 Loopback0 口、发向组播组 FF04::1 的组播数据可以沿着这可途径 R2 到 R1 的 SPF 进行转发。

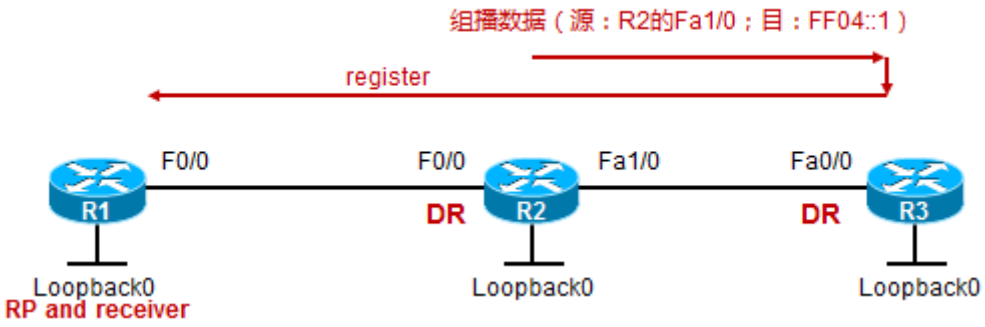
R2 的表项（2033::3 是 R3 的 Loopback0 口 IP 地址）：

```
(2033::3, FF04::1), 00:00:04/00:03:24, flags: ST
  Incoming interface: FastEthernet1/0
  RPF nbr: FE80::CE02:1BFF:FE54:0
  Immediate Outgoing interface list:
    FastEthernet0/0, Forward, 00:00:04/00:03:24
```

R3 的表项：

```
(2033::3, FF04::1), 00:00:16/00:03:13, flags: SFT
  Incoming interface: Loopback0
  RPF nbr: FE80::CE02:1BFF:FE54:0
  Immediate Outgoing interface list:
    FastEthernet0/0, Forward, 00:00:15/00:03:14
```

【测试3】

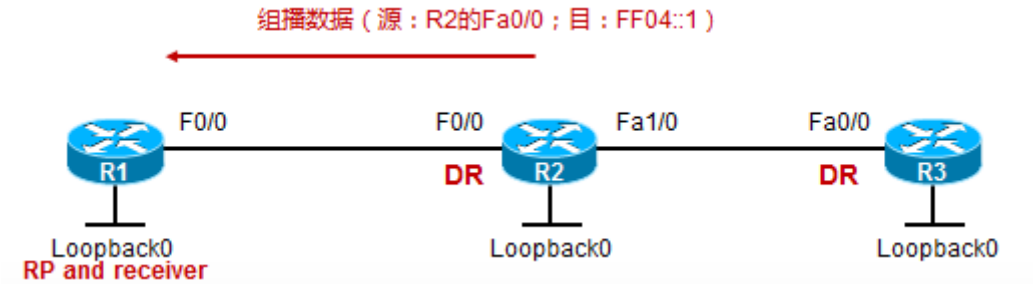


测试描述： 在 R2 上 ping FF04::1，出接口为 Fa1/0

测试现象： 能够 ping 通

现象分析： R2 产生的这些组播数据包，源地址为 R2 的 Fa1/0 口地址，目的地就不说了铁定是组播组的 IP。这些组播数据到了 R3，由于 R3 是第一跳路由器，并且 R3 的 Fa0/0 口又是本 LAN 的 DR，因此 R3 将向 RP 发起 Register，组播数据被送到了 R1。这时候实际上组播数据是到了 R1 也就是到了接收者了，但是实际上，底下还在发生着一些更复杂的事情。R1 在收到 Register 之后，会有意愿与源形成 SPT，因此它向 R2 发了一个 SPT 的 join 消息，R2 收到之后将自己的 Fa0/0 口添加到对应组播表项的 outgoing interface list 中。但是，由于组播数据始终是从 R2 的 Fa1/0 口发出（因为我们测试的时候，指定的 outgoing 接口为 Fa1/0），因此 R1 不断的收到 R3 发来的、封装了组播数据的 Register，却没有收到沿着 SPT 传下来的组播数据，因此，R1 不会向 R3 发送 register-stop，R3 的注册过程只能反复的进行，组播数据将一直被封装在 Register 消息中传给 R1。

【测试4】



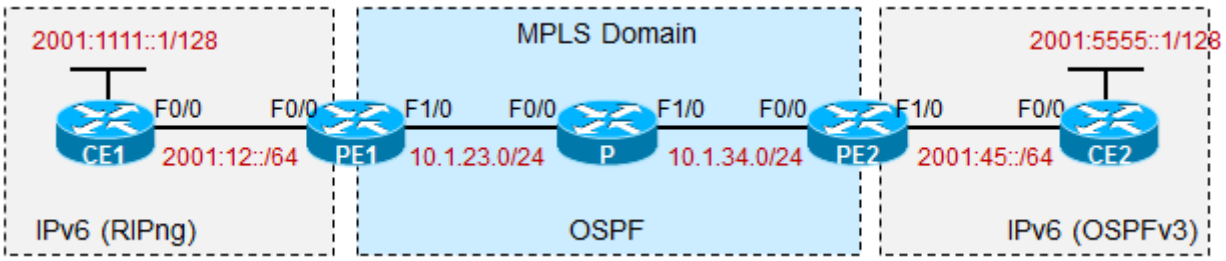
测试描述： 在 R2 上 ping FF04::1，出接口为 Fa0/0

测试现象： 能够 ping 通

现象分析： 这个应该说没任何难度了，此时此刻 R2 就是一个组播源，将组播数据直接发送给 R1。

12 6PE 及 6VPE

12.1 6PE 基础实验



1. 实验环境

1) IPv4 及 IPv6 地址规划如上图所示。

2) MPLS Domain 由三台路由器：PE1、P、PE2 构成，域内运行的 IGP 协议是 OSPF，进程号用 100。三台路由器各自将自己的 Loopback 口宣告进 Core 的 OSPF。Loopback 的编址为 x.x.x.x/32，x 为路由器编号，如 PE1 为 2.2.2.2、P 为 3.3.3.3、PE2 为 4.4.4.4。

3) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。为了方便观察实验现象，将 PE1 的标签范围定为 200-299；P 的标签空间为 300-399；PE2 的为 400-499。

2. 实验需求

1) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。

2) PE1 及 PE2 之间激活 MPBGP 的邻居关系，为传递 IPv6 路由做准备。

3) CE1 将 IPv6 路由放给 PE1，经由 PE1、PE2 间的 MPBGP 邻接传递给 PE2，最终传递给 CE2。CE2 到 CE1 的 IPv6 路由传递类似。实验的最终目的就达到了，通过 MPLS Backbone 为我们传递 IPv6 的路由。那么两个 CE 都能学习到对方的 IPv6 路由 我们的 IPv6 数据能否穿越 Core 呢？毕竟 Core 是非 IPv6 的而且更加没有 IPv6 路由的。这时候，MPLS 的威力就发挥出来了。站点之间的数据在 MPLS Backbone 内传输的时候，多是两层标签，外层是 LDP 生成的，内层是 MPBGP 为 IPv6 路由生成的。

3. 实验步骤及配置

1) 完成 CE1、CE2 的配置

CE1 的配置如下：

```
ipv6 unicast-routing
!
```

```
ipv6 router ospf 1
   router-id 1.1.1.1
!
interface Loopback0
   ipv6 enable
   ipv6 address 2001:1111::1/64
   ipv6 ospf 1 area 0
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:12::1/64
   ipv6 ospf 1 area 0
```

CE2 的配置如下：

```
ipv6 unicast-routing
!
ipv6 router ospf 1
   router-id 5.5.5.5
!
interface Loopback0
   ipv6 enable
   ipv6 address 2001:5555::5/64
   ipv6 ospf 1 area 0
interface FastEthernet0/0
   ipv6 address 2001:45::5/64
   ipv6 ospf 1 area 0
```

2) 完成 Core 区域内 IGP 协议的配置

PE1 的配置如下：

```
ip cef
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
interface FastEthernet0/0
```

```
 ipv6 enable
 ipv6 address 2001:12::2/64
 ipv6 ospf 1 area 0
!
mpls label range 200 299
mpls ldp router-id loopback0
!
interface FastEthernet1/0
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100                          !! Core 内的 OSPF
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
```

P 的配置如下：

```
ip cef
mpls label range 300 399
mpls ldp router-id Loopback0
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
interface FastEthernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
interface FastEthernet1/0
 ip address 10.1.34.3 255.255.255.0
 mpls ip
router ospf 100
 router-id 3.3.3.3
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip cef
!
mpls label range 400 499
mpls ldp router-id Loopback0
!
interface Loopback0
   ip address 4.4.4.4 255.255.255.255
interface FastEthernet0/0
   ip address 10.1.34.4 255.255.255.0
   mpls ip
interface FastEthernet1/0                    !! 连接 CE2 的接口
   ipv6 enable
   ipv6 address 2001:45::4/64
   ipv6 ospf 1 area 0
 !
router ospf 100                              !! core 内的 IGP
   router-id 4.4.4.4
   network 4.4.4.4 0.0.0.0 area 0
   network 10.1.34.4 0.0.0.0 area 0
```

3) 完成 PE1 及 PE2 上，PE-CE 路由协议的配置，以及 MP-BGP 的配置

PE1 的配置如下：

```
ip cef
ipv6 unicast-routing          !! 注意，要支持 IPv6 单播路由协议，该开关必须打开
ipv6 cef                      !! 注意，支持 IPv6 的 PE 上，IPv6 cef 必须打开
!
ipv6 router ospf 1            !! PE-CE 间的 OSPFv3
   router-id 2.2.2.2
   redistribute bgp 234
!
interface FastEthernet0/0
   ipv6 ospf 1 area 0
!
router bgp 234
```

```
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 neighbor 4.4.4.4 remote-as 234              !! 4.4.4.4 也就是 PE2
 neighbor 4.4.4.4 update-source Loopback0
!
 address-family ipv6
    neighbor 4.4.4.4 activate                !! 激活与 PE2 的 IPv6 地址族
    neighbor 4.4.4.4 send-label              !! 向 PE2 发送为 IPv6 前缀分配的标签
    redistribute connected                   !! 重发布直连
    redistribute ospf 1                      !! 将从 CE1 学习到的 IPv6 路由前缀重发布进 BGP
 exit-address-family
```

PE2 的配置如下：

```
ip cef
ipv6 unicast-routing                 !! 注意，要支持 IPv6 单播路由协议，该开关必须打开
ipv6 cef                             !! 注意，支持 IPv6 的 PE 上，IPv6 cef 必须打开
!
interface FastEthernet1/0
 ipv6 ospf 1 area 0
 !
router ospf 100
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
 !
router bgp 234
 no bgp default ipv4-unicast
 neighbor 2.2.2.2 remote-as 234
 neighbor 2.2.2.2 update-source Loopback0
 !
 address-family ipv6
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 send-label
    redistribute connected
    redistribute ospf 1
```

```
  exit-address-family
 !
ipv6 router ospf 1
  router-id 4.4.4.4
  redistribute bgp 234
```

4. 实验分析

首先在 PE1 上，看一下：

PE1# show bgp all neighbors 4.4.4.4

```
…部分省略. ..
For address family: IPv6 Unicast
BGP neighbor is 4.4.4.4,   remote AS 234, internal link
  BGP version 4, remote router ID 4.4.4.4
  BGP state = Established, up for 01:05:47
  Last read 00:00:44, last write 00:00:47, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv6 Unicast: advertised and received
    ipv6 MPLS Label capability: advertised and received
…部分省略. ..
```

可以看出，PE1 及 4.4.4.4 的 PE2 之间是具有 IPv6 MPLS 的标签分配和分发能力的。

PE1#show bgp ipv6 unicast

```
BGP table version is 5, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 2001:12::/64	::	0		32768	?
*>i2001:45::/64	::FFFF:4.4.4.4	0	100	0	?
*> 2001:1111::1/128	::	1		32768	?
*>i2001:5555::5/128	::FFFF:4.4.4.4	1	100	0	?

我们看到，PE1 上已经学习到来自 CE2 的 IPv6 路由了，继续看看

PE1#show bgp ipv6 unicast labels

Network	Next Hop	In label/Out label
2001:12::/64	::	204/nolabel
2001:45::/64	::FFFF:4.4.4.4	nolabel/404
2001:1111::1/128	::	203/nolabel
2001:5555::5/128	::FFFF:4.4.4.4	nolabel/403

我们重点看 2001:5555::5/128 这条路由，下一跳是个奇怪的地址，不过从这个地址的最后 32bits 也就是 4.4.4.4 不难看出这条路由是谁传给我的，另外，前缀的 out label 是 403，而这个 403 正是 PE2 传递给我的。也就是说，PE2 为 2001:5555::5/128 这条 IPv6 前缀分配了 403 标签，并且将这个映射传递给了 PE1，这个标签其实是 MP-BGP 产生的。那么当 PE1 有数据要去往 2001:5555::5 是不是直接打上标签 403 就行了呢？当然不是，因为标签 403 对于 MPLS Backbone 内的 P 路由器来说，根本不理解，因此，我们还需要一个外层标签，这个标签由 LDP 产生，使得我们的数据包能够在 MPLS Backbone 内传输。

进一步查看这条路由：

PE1#show bgp ipv6 unicast 2001:5555::5/128

```
BGP routing table entry for 2001:5555::5/128, version 3
Paths: (1 available, best #1, table Global-IPv6-Table)
  Not advertised to any peer
  Local
    ::FFFF:4.4.4.4 (metric 3) from 4.4.4.4 (4.4.4.4)
      Origin incomplete, metric 1, localpref 100, valid, internal, best
      mpls labels in/out nolabel/403
```

我们发现路由的下一跳是 4.4.4.4。

PE1#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	10.1.34.0/24	0	Fa1/0	10.1.23.3
201	301	4.4.4.4/32	0	Fa1/0	10.1.23.3
202	Pop tag	3.3.3.3/32	0	Fa1/0	10.1.23.3
203	Untagged	2001:1111::1/128	570	Fa0/0	FE80::CE00:10FF:FE58:0
204	Aggregate	2001:12::/64	2204		

进一步查看，发现 P 为 4.4.4.4 分配的了标签 301，因此，当 PE1 收到数据包，去往 2001:5555::5 的时候，数据包会被压上两层标签，外层标签值为 301，内层标签值为 403。

为了再次证明这一点，我们在 PE1 上：

PE1#sh ipv cef 2001:5555::5

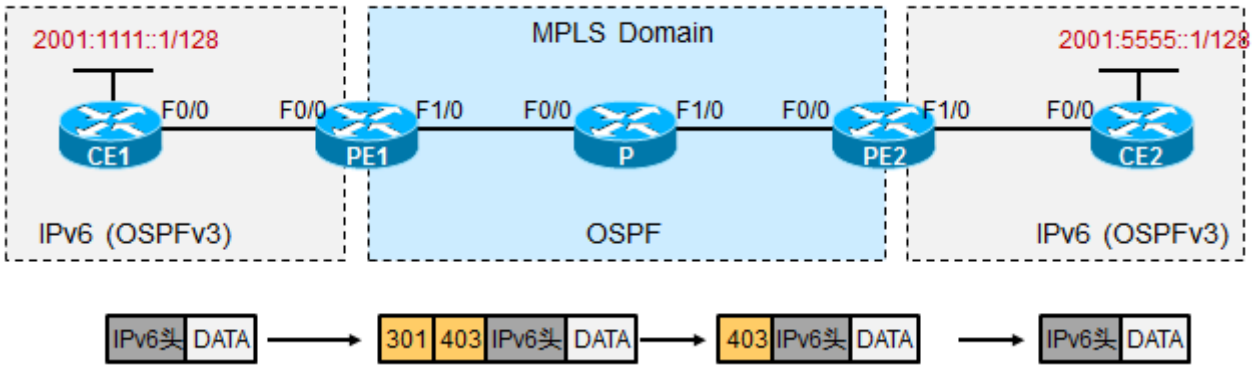
```
2001:5555::5/128
```

```
    nexthop ::FFFF:4.4.4.4
  fast tag rewrite with Fa1/0, 10.1.23.3, tags imposed: {301 403}
```

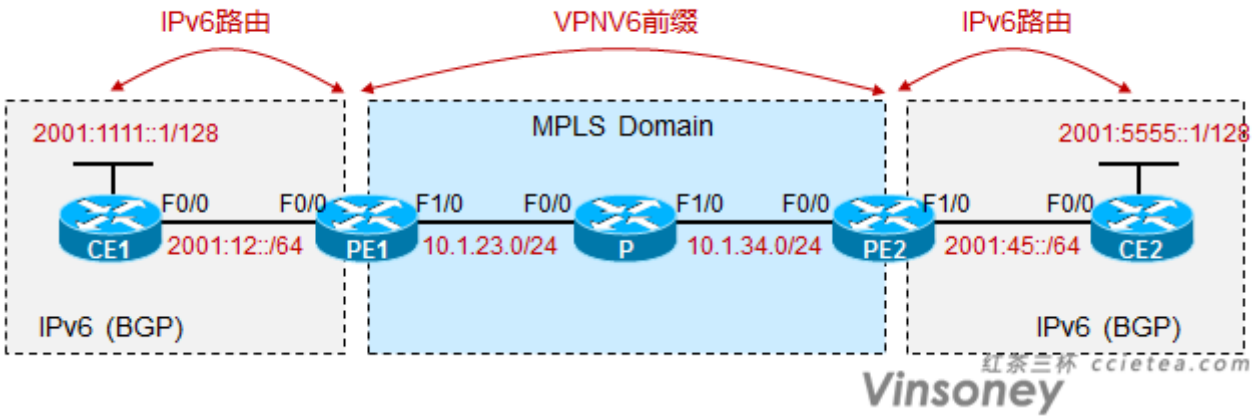
以下是我们在 CE1 上，ping 2001:5555::5，然后在 PE1 的 Fa1/0 口上抓包：

```
Ethernet II, Src: cc:01:10:58:00:10 (cc:01:10:58:00:10), Dst: cc:02:10:58:00
MultiProtocol Label Switching Header, Label: 301, Exp: 0, S: 0, TTL: 63
  MPLS Label: 301
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 0
  MPLS TTL: 63
MultiProtocol Label Switching Header, Label: 403, Exp: 0, S: 1, TTL: 63
  MPLS Label: 403
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 1
  MPLS TTL: 63
Internet Protocol Version 6, Src: 2001:12::1 (2001:12::1), Dst: 2001:5555::5
Internet Control Message Protocol v6
```

从抓包的结果能看到，有两层 label，这个标签包到了 P 路由器后，由于有 PHP 机制，顶层的 LDP 标签被弹出，接着这个标签包又被传到 PE2，PE2 弹出 MP-BGP 的标签，还原成原始的 IPv6 数据然后转发给 CE2。



12.2 6VPE 基础实验



1. 实验环境

4) IPv4 及 IPv6 地址规划如上图所示。

5) MPLS Domain 由三台路由器：PE1、P、PE2 构成，域内运行的 IGP 协议是 OSPF，进程号用 100。三台路由器各自将自己的 Loopback 口宣告进 Core 的 OSPF。Loopback 的编址为 x.x.x.x/32，x 为路由器编号，如 PE1 为 2.2.2.2、P 为 3.3.3.3、PE2 为 4.4.4.4。

6) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。为了方便观察实验现象，将 PE1 的标签范围定为 200-299；P 的标签空间为 300-399；PE2 的为 400-499。

2. 实验需求

4) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。

5) PE1 及 PE2 之间激活 MPBGP 的邻居关系，**为传递 VPNv6 前缀做好准备**

6) CE1 将 IPv6 路由放给 PE1，PE1 上创建 IPv6 的 VRF，将 Fa0/0 口放进 VRF。PE1 通过 BGP 学习到 CE1 的 IPv6 前缀，并形成 VPNv6 前缀，通告给自己的 MP-BGP 邻居 PE2，PE2 将 VPNv6 前缀对应的 IPv6 前缀最终传递给 CE2。

3. 实验步骤

1) CE1 及 CE2 的配置

CE1 的配置如下：

```
ipv6 unicast-routing
interface Loopback0
  ipv6 enable
  ipv6 address 2001:1111::1/128
```

```
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
router bgp 100
  bgp router-id 1.1.1.1
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 2001:12::2 remote-as 234
 !
  address-family ipv6
    neighbor 2001:12::2 activate
    network 2001:1111::1/128
  exit-address-family
```

CE2 的配置如下：

```
ipv6 unicast-routing
interface Loopback0
  ipv6 enable
  ipv6 address 2001:5555::5/128
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:45::5/64
router bgp 100
  bgp router-id 5.5.5.5
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 2001:45::4 remote-as 234
 !
  address-family ipv6
    neighbor 2001:45::5 activate
    network 2001:5555::5/128
  exit-address-family
```

2) 完成 MPLS Backbone 的 IGP 及 LDP 配置

PE1 的配置如下：

```
ip cef
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface FastEthernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100
 router-id 2.2.2.2
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
```

P 的配置如下：

```
ip cef
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 300 399
!
interface FastEthernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
interface FastEthernet0/1
 ip address 10.1.34.3 255.255.255.0
 mpls ip
!
```

```
router ospf 100
    network 3.3.3.3 0.0.0.0 area 0
    network 10.1.23.3 0.0.0.0 area 0
    network 10.1.34.3 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip cef
interface Loopback0
  ip address 4.4.4.4 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 400 499
!
interface FastEthernet0/0
  ip address 10.1.34.4 255.255.255.0
  mpls ip
!
router ospf 100
  router-id 4.4.4.4
  network 4.4.4.4 0.0.0.0 area 0
  network 10.1.34.4 0.0.0.0 area 0
```

3) 完成 VRF 的配置、MP-BGP 的配置

PE1 的配置如下：

```
ip cef
ipv6 unicast-routing                !! 再次强调下这几天命令的必要性
ipv6 cef
!
vrf definition cisco                !! 定义 vrf
  rd 1:1
  !
  address-family ipv6               !! 在 IPv6 地址族中定义 RT 值
    route-target export 234:2
    route-target import 234:4
```

```
  exit-address-family
!
interface FastEthernet0/0
  vrf forwarding cisco                    !!接口关联到 VRF cisco
  ipv6 enable
  ipv6 address 2001:12::2/64
!
router bgp 234
  bgp router-id 2.2.2.2
  no bgp default ipv4-unicast
  neighbor 4.4.4.4 remote-as 234
  neighbor 4.4.4.4 update-source Loopback0
  !
  address-family vpnv6                    !!邻居 4.4.4.4 也就是 PE2，激活 VPNv6 支持能力
    neighbor 4.4.4.4 activate
    neighbor 4.4.4.4 send-community extended
  exit-address-family
  !
  address-family ipv6 vrf cisco           !! 用于从 CE1 学习 IPv6 路由前缀
    neighbor 2001:12::1 remote-as 100
    neighbor 2001:12::1 activate
  exit-address-family
```

PE2 的配置如下：

```
ip cef
ipv6 unicast-routing                      !! 再次强调下这几天命令的必要性
ipv6 cef
!
vrf definition cisco
  rd 1:1
  !
  address-family ipv6
    route-target export 234:4
    route-target import 234:2
```

```
  exit-address-family
!
interface FastEthernet0/1
  vrf forwarding cisco
  ipv6 enable
  ipv6 address 2001:45::4/64
!
router bgp 234
  bgp router-id 4.4.4.4
  no bgp default ipv4-unicast
  neighbor 2.2.2.2 remote-as 234
  neighbor 2.2.2.2 update-source Loopback0
  !
  address-family vpnv6
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 send-community extended
  exit-address-family
  !
  address-family ipv6 vrf cisco
    neighbor 2001:45::5 remote-as 500
    neighbor 2001:45::5 activate
  exit-address-family
```

4. **实验分析**

我们在 PE1 来看看：

PE1#show bgp vpnv6 unicast all 2001:5555::5/128

```
BGP routing table entry for [1:1]2001:5555::5/128, version 32
Paths: (1 available, best #1, table cisco)
  Advertised to update-groups:
        1
  500
    ::FFFF:4.4.4.4 (metric 3) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:234:4
```

```
        mpls labels in/out nolabel/403
```

查看 PE1 上的 VPNv6 前缀 2001:5555::5 ,可以看到 out label 是 403 ,很明显是 R4 也就是 PE2 分配的标签，这个标签其实就是 VPNv6 的标签，为了让 PE2 知道，这个数据所归属的 VRF。也可以使用如下命令查看 VPNV6 前缀关联的标签：

PE1#show bgp vpnv6 unicast all labels

```
   Network          Next Hop        In label/Out label
Route Distinguisher: 1:1 (cisco)
   2001:1111::1/128 2001:12::1        204/nolabel
   2001:5555::5/128 ::FFFF:4.4.4.4    nolabel/403
```

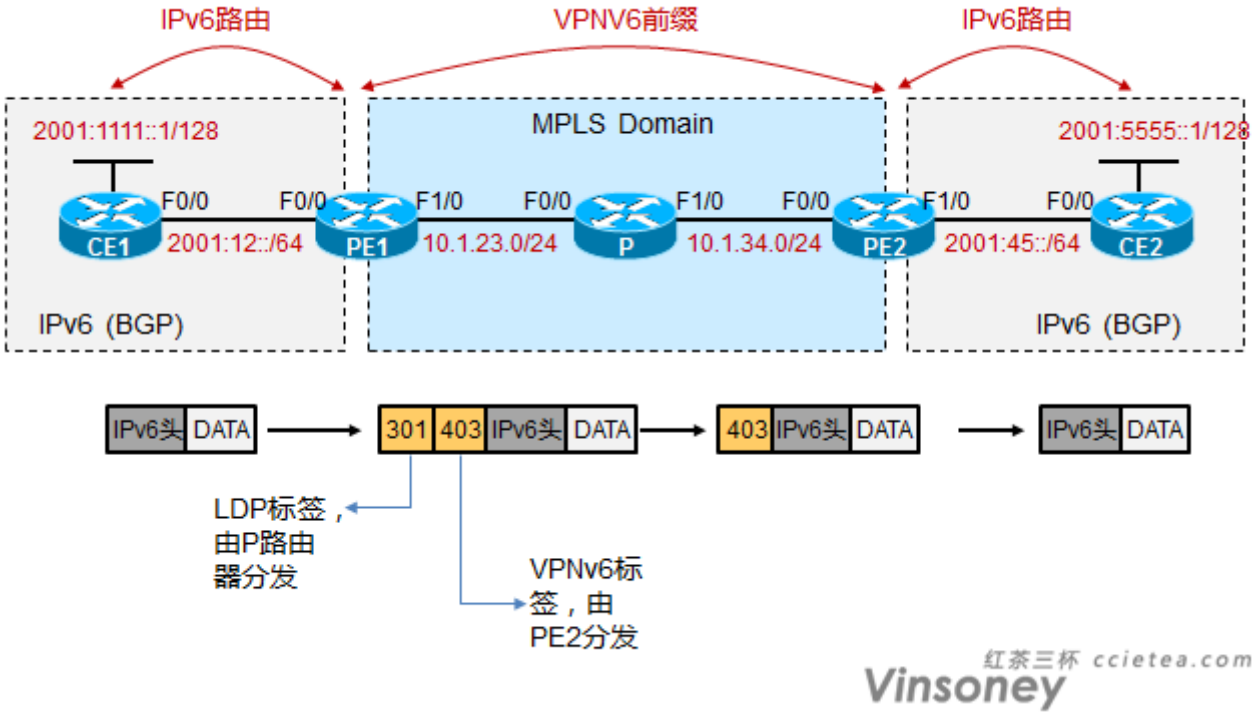
现在 2001:5555::5 有了 VPNV6 的标签，但是，还数据层面是无法穿越 MPLS BACKBONE 的，因为 P 路由器都不识别这个标签。因此，还是需要再增加一个标签，使得内网标签及数据负载能够在 MPLS BACKBONE 中传输。这个外层标签由 LDP 分配。

PE1#show ipv6 cef vrf cisco 2001:5555::5/128

```
2001:5555::5/128
  nexthop 10.1.23.3 FastEthernet0/1 label 301 403
```

从 CEF 表可以看出，当 PE1 收到 CE1 去往 CE2 的 2001:5555::5 的流量时，标签是压两层的，外层是 301，是 P 路由器分发的，内层是 403，是 PE1 分发的。

因此，当 CE1 从源 2001:1111::1 访问 2001:5555::5 时，数据层面上是这样的：



5. 其他命令

Show vrf ipv6	查看 vrf ipv6
Show vrf ipv6 detail	查看 vrf 详细信息

13 参考书籍

- CISCO IPv6 网络实现技术：CISCO self-study : Implementing Cisco IPv6 Networks(IPv6)

IS-IS 技术笔记

红茶三杯 CCIE 学习文档

文档版本：	2.0	更新时间：	2013-03-20
文档作者：	红茶三杯	文档地址：	http://ccietea.com
文档备注：	请关注文档版本及更新时间		

1 基础知识

1.1 协议概述

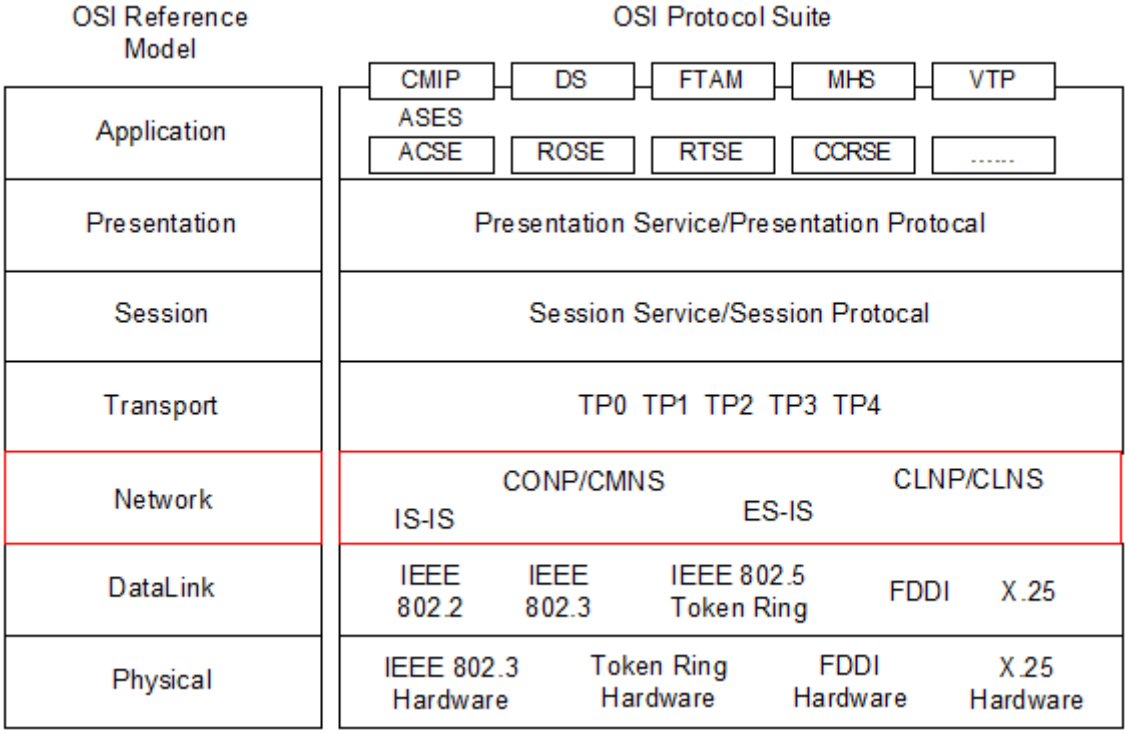
1. IS-IS 是一个链路状态的、内部网关协议。它被设计成适用在 OSI 无连接网络服务（CLNS：Connctionless Network Service）的环境中。为了提供对 IP 的路由支持，IETF 在 RFC1195 中对 IS-IS 进行了扩充和修改，使它能够同时应用在 TCP/IP 和 OSI 环境中，称为集成化 IS-IS（Integrated IS-IS 或 Dual IS-IS）
2. 集成型 IS-IS 和 OSPF 都是 20 世纪 80 年代后期定义的，大约是 1988 年。OSPF 实际上是从 IS-IS 的早期版本进化而来的，不过它采用 IP 作为前提。
3. IS-IS 是路由选择协议，传递 CLNP 的路由信息。在 ISO 协议族，我们可以把 ES 终端系统理解为主机，把 IS 中间系统理解为路由器。因此 IS-IS 的出生 ,其实是为 CLNS 服务的 ,往后发展成了集成性 IS-IS 就做了扩展，也就可以支持 IP 路由了。
4. IS-IS 协议管理距离 115

1.2 OSI 协议栈

1. OSI 协议簇及模型

OSI(Open System Interconnect)参考模型是一个国际化标准 ,用于增强不同厂商设备之间的互操作性。它定义了一个 7 层的模型，并且详细规定了各层的功能，同时也确定了计算机网络的标准。制定 OSI 七层参考模型的是 ISO（International Organization for Standardization，国际标准化组织）。对于数据通信和信息技

术的发展来说，OSI 参考模型起到了重要的作用。它提供了开放式的标准架构，使不同厂商生产的通信设备之间可以进行互联和互操作。ISO 七层模型的每一层都定义了单一的功能，可以将相关功能组合成功能层，从而简化和方便了协议的设计。



OSI 参考模型中的网络服务规范定义了网络设备之间使用无连接通信的功能，也就是 CLNS（Connectionless Network Service，无连接网络服务）。顾名思义，使用 CLNS，无需在发送数据之间建立端到端的路径。下图中展示的是 CLNS 中所包括的协议组件，这些协议组件都由 ISO 所定义。**OSI CLNS** 类似于 TCP/IP 中的**网络层服务。**

CLNP（Connectionless Network Protocol，无连接网络协议）、IS-IS、ES-IS（End System—Intermediate System，终端系统—中间系统）都是 ISO 定义的独立的 OSI 第三层（网络层）的协议，这些协议分别在不同的 ISO 标准中定义。

2. OSI CLNS 包含以下三个协议：

- CLNP　　等价于 TCI/IP 模型中的 IP，他提供尽力而为的传输。
- ES-IS　　终端系统到中间系统的协议，类似 TCP/IP 中的 ARP、ICMP 等协议
- IS-IS　　中间系统到中间系统，路由选择协议。IS-IS 是路由选择协议，传递 CLNP 的路由信息。

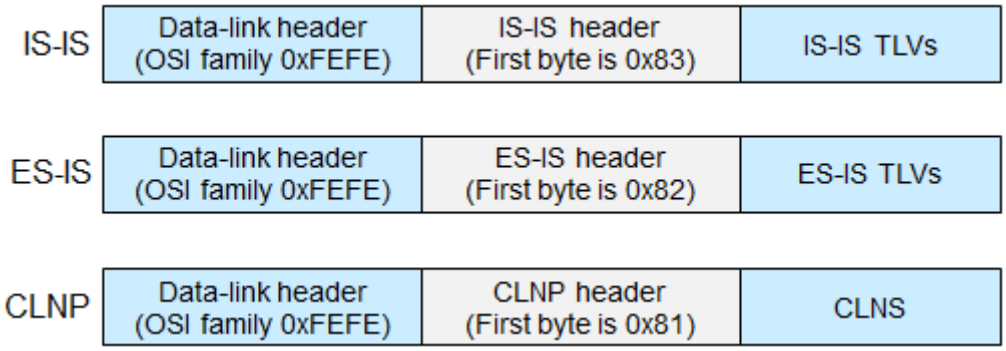
CLNP 这个名词可能很多人都比较陌生，它是一个 OSI 网络层协议。打个比方来说，它就相当于我们所熟悉的 IP 协议，而 IP 定义为用来为 TCP/IP 协议栈提供网络层服务。与 IP 一样，CLNP 也是一个无连接的协议，不提供可靠的数据连接，而且也独立于下层（数据链路层）协议。我们都知道，IP 是 TCP/IP 协议栈中唯一的网络层协议，高层的协议和数据全都封装在 IP 数据包中进行传输。这不同于 CLNS 网络环境，在

CLNS 中，CLNP、IS-IS、ES-IS 都是独立的网络层协议，它们都直接被封装到数据链路层的帧中进行传输。

在 ISO 协议族，我们可以把 ES 终端系统理解为主机，把 IS 中间系统理解为路由器。因此 IS-IS 的出生，其实是为 CLNS 服务的，往后发展成了集成性 IS-IS 就做了扩展，也就可以支持 IP 路由了。

3. 关于 CLNP

CLNP 类似 IP 协议，只不过它是为 ISO 传输层提供服务的。IS-IS、ES-IS、CLNP 都是网络层协议，都是直接封装在数据链路层帧内的。相比于 TCP/IP 中的 OSPF 报文是躲在 IP 头后的，前者的协议报文的封装效率就要高一些了。



4. ES-IS

OSI 已经为用于路由选择的两种网络协议制定了标准：ES-IS 和 IS-IS：

在 OSI 术语中，主机（例如 PC）被称为 ES（终端系统），路由器被称为 IS（中间系统）。ES-IS 可以说是一种终端系统和路由器之间的“语言”或路由协议。它用来使同一网段或链路的终端系统和路由器之间可以彼此发现对方，并可以让 ES 能够获悉其网络层地址。总结来说，**ES-IS 主要有以下几种功能**：

1) 使 ES 获悉其所在的区域，即区域前缀
2) 在 ES 与 IS 之间建立邻接关系
3) 建立数据链路层地址到网络层地址（CLNP 地址）的映射

可以看出，ES-IS 在 CLNS 网络环境中的作用就好像 IP 网络中的 ICMP、ARP 与 DHCP 协议的协同工作。

在 ES-IS 工作过程中，终端系统通过发送 ESH（ES Hello）报文到特定的地址，目的是向路由器通告自己的存在。路由器通过监听 ESH 报文，以发现网络中存在的 ES，以便后续将到达特定 ES 地址的数据包转发给 ES。在 ES-IS 中，路由器通过发送 ISH（IS Hello）报文到特定地址，也向 ES 通告其自身的存在。ES 也监听 ISH，如果收到多个 IS 发送的 ISH，ES 将随即进行选择，并将所有数据都发送给这个 IS。

需要注意的是，通常我们现网环境中的终端系统，例如 PC，都不使用 ES-IS，因为这些 PC 都运行的是 TCP/IP 协议栈，类似 ES-IS 的工作都由 TCP/IP 协议栈中的 ARP、ICMP、DHCP 协议来完成。

每一个 ES 都属于一个特定的区域，当 ES 通过侦听中间系统的 HELLO（ISH）分组发现最近的 IS 的时

候，OSI 路由就开始了。当一个 ES 想发一个分组给另一个 ES 的时候，它就把分组发给它的直连网络上某个 IS（第 0 层 routing），然后这个 router 就会查找目的地址并沿着最佳路径转发分组。

5. IS-IS

IS-IS 是 CLNS 中一个重要的组成部分 ,它是一个用来在 CLNS 网络环境中使路由器与路由器(IS 与 IS ）之间动态的交换路由信息的协议，IS-IS 在 ISO 10589 中进行了定义。IS 与 IS，即路由器与路由器之间的通信使用 IIH（IS-IS Hello）报文。IS-IS 的设计主要是为了满足 CLNS 网络中的如下需求：

1) 在路由域内执行路由选择协议功能
2) 为网络提供最佳路由
3) 当网络出现故障后，能够快速的收敛
4) 提供无环路的网络
5) 提供网络的稳定性
6) 提供网络的可扩展性
7) 合理利用网络资源

为了满足如上需求，IS-IS 被设计成一种链路状态路由协议，并且使用 SPF 最短路径优先算法以实现快速的收敛和无环路网络。

6. 集成 IS—IS

之前所提到的 IS-IS，它仅支持 CLNS 网络环境，而不支持 IP 网络环境中的路由信息交换。后来，IETF 在 RFC 1195 中对 IS-IS 进行了修改和扩展 ,称之为集成 IS-IS(Integrated IS-IS)或双重 IS-IS(Dual IS-IS)。集成 IS-IS 的制定是为了使其能够同时应用在 TCP/IP 网络和 OSI 网络中，使其能够为 IP 网络提供动态的路由信息交换。

集成 IS-IS 是一个能够同时处理多个网络层协议（例如 IP 和 CLNP）的路由选择协议。相反，OSPF 只支持 IP 一种网络层协议，即 OSPF 仅支持 IP 路由。而集成 IS-IS 可以支持纯 CLNP 网络或纯 IP 网络，或者同时支持 CLNP 和 IP 两种网络环境，并为其提供路由功能。集成 IS-IS 协议经过多年的发展，已经成为一个可扩展的、功能强大的、易用的 IGP 路由选择协议，并且在运营商网络中得到了更多的应用和部署，主要用来实现域内的 IP 路由选择。

- 集成 IS-IS（integrated IS-IS）使得 IS-IS 协议可以传播除 CLNP 之外的其他协议的路由信息。
- IS-IS 能在混合模式下同时路由 CLNP 和 IP。
- IS-IS 可以纯粹地用做 IP 路由选择，也可以纯粹地用做 ISO 路由选择，或同时用于两者。
- 即使只为 IP 提供路由选择功能，也需要 CLNS 地址。

1.3 OSI 协议栈术语

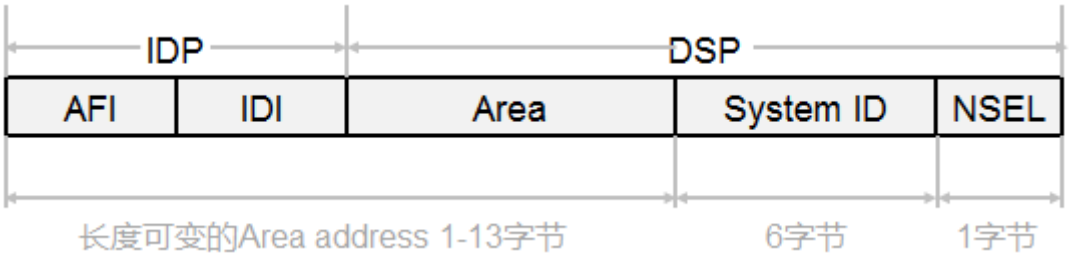
OSI 术语	OSI 术语解释	IP 中“类似”的概念
IS	Intermediate System 中间系统	Router 路由器
ES	End System 终端系统	Host 主机
PDU	Packet data unit 数据单元	Packets 包
NSAP	Network Service Access Point。是在网络层和传输层之间的边界上的概念性的点，它是 OSI 网络层为传输层提供服务的位置，每个传输层实体都会分配到一个唯一的 NASP 地址。NSAP 的最后一个字节用来标识在同一个设备上的程序，类似于 TCP/IP 中的端口。	NSAP 可以理解为类似 IP+端口号的这么一个东西。
NET	Network Entity Title 是最后一个字节为 0 的 NSAP，它就是标识一个设备（NSAP 中的端口号为 0 嘛）。因此每个 router 都有唯一的 NET	-
SNPA	subnet point of attachments 是提供子网服务的点，它等价于对应的第三层地址（NET 或 NSAP）的第二层地址。通常是 LAN 上的 MAC 或者是 X.25，FR 或 ATM 中虚电路的 ID	MAC 地址等
SysID	System ID 系统 ID	OSPF 的 routerID
LSP	Link state Protocol Data Unit 链路状态数据单元	OSPF 的 LSA
LSPDB	LSP Database LSP 数据库	OSPF 的 LSADB
DIS	Designated Intermediate System 指定中间系统	OSPF 的 DR

2 ISO 编址

OSI 网络层编址是通过使用两类层次化地址：NSAP 和 NET 来实现的

2.1 NSAP

每一个传输层的实体都会分配到一个 NSAP 地址。 **NSAP 地址是 CLNS 分组的网络层地址**。它用来标识设备。它由初始域部分（IDP）和域内自定义部分（DSP）组成，这两部分下面又做了详细的设定，在咱们理解这个 IDP 和 DSP 的时候，可以借 IP 里头的概念来套，IDP 有那么点像主网络号的意思，DSP 有那么点像子网号和主机 IP，当然，这个 IDP 的含义相对要复杂一点，见下图：



- AFI　　Authority and Format Identifier　，1 字节的授权和格式标识符。AFI 字段标识与 NSAP 相关的高层寻址域和 DSP 部分的语法。AFI 字段的取值范围为 0-99 的十进制数。高层地址域提供各种各样的子域，其值由 IDI 字段分配。每一个高层域定义自己的 IDI 字段格式。下面就是地址域 AFI 和 DSP 语法类型。

Address Domain	DSP Syntax		
	Decimal	Binary	Character
X.121	36	37	
ISO DCC	38	39	
F.69	40	41	
E.163	42	43	
E.164	44	45	
ISO 6523 ICD	46	47	
Local	48	49	50

- IDI　　Initial Domain Identifier，可变长的初始域标识符，标识 AFI 下的子域
  AFI+IDI　　用于标识 Domain
- Area　　2 字节的 area 标识符，也称为 Domain 内自定义部分的高位部分（HODSP）用来把 Domain 细分为 area， 大致类同 IP 中的子网
- System ID　6 字节的系统 ID，ES 或 IS 的标识符，类似于 OSPF 的 router ID，每台设备都有一个系统 ID，而在 IP 网络中每个接口都有一个 IP，这是区别之一。要注意：SystemID 必须在整个 Area 和主干（Level2）上保持唯一。
- N-SEL　　1 字节的选择符 ,英文 :NSAP-Selector 类似 TCP/IP 中的端口， 识别设备上的进程(或服务)，在 NET 中为 00

IS-IS 使用简单的两层结构。把 IDP 和 HODSP 合在一起用作第 2 层路由的区域 ID，而剩下的系统 ID 就做第一层路由选择。

对于 IP 应用程序而言，在 NSAP 地址中，1 字节定义 AFI，最少 2 字节定义实际区域信息，6 字节定义系统 ID 和 1 字节定义 NSEL。因此 NSAP 地址最少为 10 字节。在 CISCO IOS 中，NSAP 配置为点分 16 进制形式。AFI 值为 49 的只能本地使用，是 RFC1618 定义的预留私有地址空间。

2.2 NET

NSEL 值为 0 的 NSAP 地址用来标识设备，这就是这个设备的网络地址 NET。因此 NET 由区域 Id 和系统 ID 所决定。总的来说，NSAP 编址风格和 IP 编址风格之间的最大区别**就是 NSAP 仅使用一个地址标识一台 router，**而 IP 则是每个端口都分配一 Ip 地址。

分配 NET 的一些指导方针：

1. 一个中间系统（可以理解为路由器）至少有一个 NET（最多可有 254 个）且系统 ID 必须相同。Cisco 路由器默认支持最多三个 NET 地址，可使用如下命令修改：

```
Router(config-router)# max-area-address xx
```

2. 在一个路由选择区中的全部 IS 和 ES 必须有相同长度的系统 ID
3. 在一个区域中的所有 router 必须有相同的区域 ID
4. 所有的 2 层 router 必须有域内唯一的系统 ID
5. 所有的 1 层 router 必须有区域内唯一的系统 ID
6. 如果 ES 和 IS 有相同的区域 ID，那么同一个区域的所有 ES 都会和它同在一段共享媒介质上的 1 层 router 建立毗邻关系。
7. 如果一个 router 上分配了多个 net，则这些 NET 的系统 ID 必须是相同的。

2.3 Cisco Router 的 ISO 编址

前面说了，NSAP 地址有 Area adddress 部分是变长的，那么当我们拿到一个地址，我怎么知道这个地址的哪一部分对应的是什么？Cisco IOS 软件从右边开始解释 NSAP 地址（这里使用 NSAP 的 OSI 格式）。首先从右边数起，第一个 B 是 NSEL，往左的 6B 是 SystemID，剩下的部分是 AreaID。

如 39.0f01.0002.0000.0c00.1111.00

NSEL 是 **00**（右边起 1B）

系统 ID 是 **0000.0c00.1111**

区域 ID 是 **39.0f01.0002**

对于 SystemID 的设定，习惯上把 router 的接口 MAC 或 IP 地址编码成系统 ID 或它的一部分。在集成 IS-IS 中普遍将回环 IP 用于这一目的，例如我有配置一个 loopback：192.168.1.24/32，然后我将这个 IP 补全，不够 3 位数的在前面加 0 形成 192.168.001.024 然后再形成一串儿：192168001024，再根据 4 位数一组形成 SystemID：1921.6800.1024.

2.4 SNPA

NSAP 或 Net 地址用来区分设备。在 ISO 术语中，数据链路层地址（LAN MAC，帧中继 DLCI 等）被称为子网连接点 SNPA（subnet point of attachments）。由于一个网络设备可能连接多个链路，因此需要多个 SNPA 地址，但只需要一个 OSI 网络地址。

ES-IS 协议的主要功能就是为网络设备提供 NSAP 地址到 SNPA 地址的映射。

SNPA 相当于 NSAP 或 NET 的二层地址。

SNPA 地址可以通过下列几种方法获得：

- Lan 接口的 MAC 地址
- X.25 或 ATM 的虚电路 ID
- FR 的 DLCI
- 对于 HDLC（高级数据链路控制）接口，SNPA 被设置为“HDLC”

术语电路（circuit）相当于接口。由于 NET 地址用于标识整个设备，所以 Circuit ID 用来表示不同的接口。路由器按照如下方式为接口指定 1 字节的 Circuit ID。

- 对于点到点接口，SNPA 是电路的唯一标识符。例如在 HDLC 点到点链路上，电路 ID 为 0x00
- 对于 LAN 接口，将 1 字节的电路 ID 附加到 6 字节的指定中间系统（DIS）的 SysID 的后面，形成一个 7 字

节的 LAN ID 如 1921.6811.1001.**03** 。“the routers connected to the multiaccess network (Ethernet) all have the **same circuit ID**. The circuit ID is a one-octet number that the router uses to uniquely identify the IS-IS interface. If the interface is attached to a multiaccess network, the circuit ID is concatenated with the system ID of the DIS”

术语链路，是位于两个 IS（路由器）之间的路径，当两个相邻的 SNPA 可以通信是，链路就处于 UP 状态

2.5 NSAP 地址到主机名的映射

这 NSAP 地址呢，还是长了点，用起来呢，也不方便，这其中 SystemID 主要是用来标识几个属性，例如邻接信息、IS-IS 数据库中的 LSP 标示符等等，大多数情况下，我们比较适应那种有字面含义的符号，例如 R1、R2 神马的，而不是用一串数字。因此 CISCO IOS 为我们提供了一个非常好的方法：路由器 hostname 到 NSAP 地址的映射。本质上就是 hostname 到 SystemID 的映射。有静态映射和动态映射两个方法。

1. **静态主机名映射**

RouterA 的配置如下：

```
router isis
  net 49.0001.1111.2222.3333.00
  exit
clns host RouterA   49.0001.1111.2222.3333.00
clns host RouterB   49.0001.4444.5555.6666.00
```

RouterB 的配置如下：

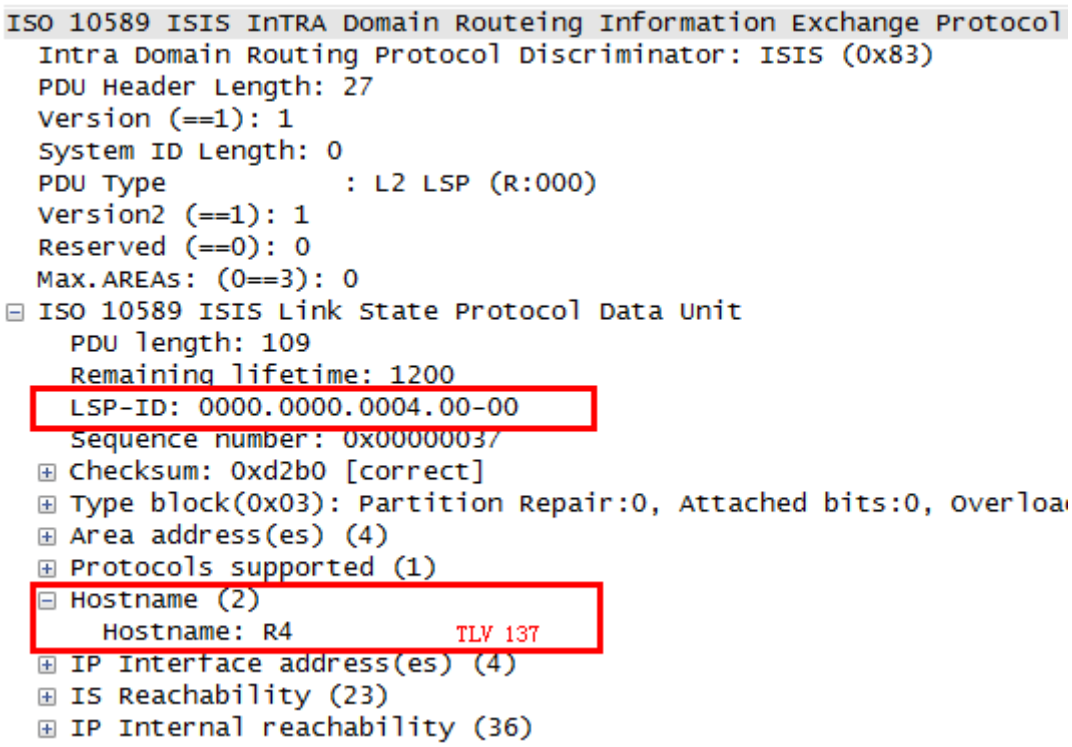
```
router isis
  net 49.0001.4444.5555.6666.00
  exit
clns host RouterA   49.0001.1111.2222.3333.00
clns host RouterB   49.0001.4444.5555.6666.00
```

在全局配置模式下使用 clns host 命令来建立主机名到 NSAP 地址的映射。这样一来在链路状态数据库中 LSP ID 中可以使用 hostname 来替代 SysID 部分。从而为我们的网络维护和排错带来极大的便利。

2. **动态主机名映射**

静态映射，当网络规模大起来了，配置量就大得惊人，RFC2763 中定义了 IS-IS 的增强特性，其中就有动态主机名的映射定义。RFC2763 引入了新的 TLV，类型 137，在参与动态映射的路由器发送的 LSP 中包含着

这个 TLV，这样一来就提供了一个简单可靠的通告主机名信息的机制。如下图，这是一个 LSP 的报文：



使用 show isis hostname 可以查看到主机名的映射

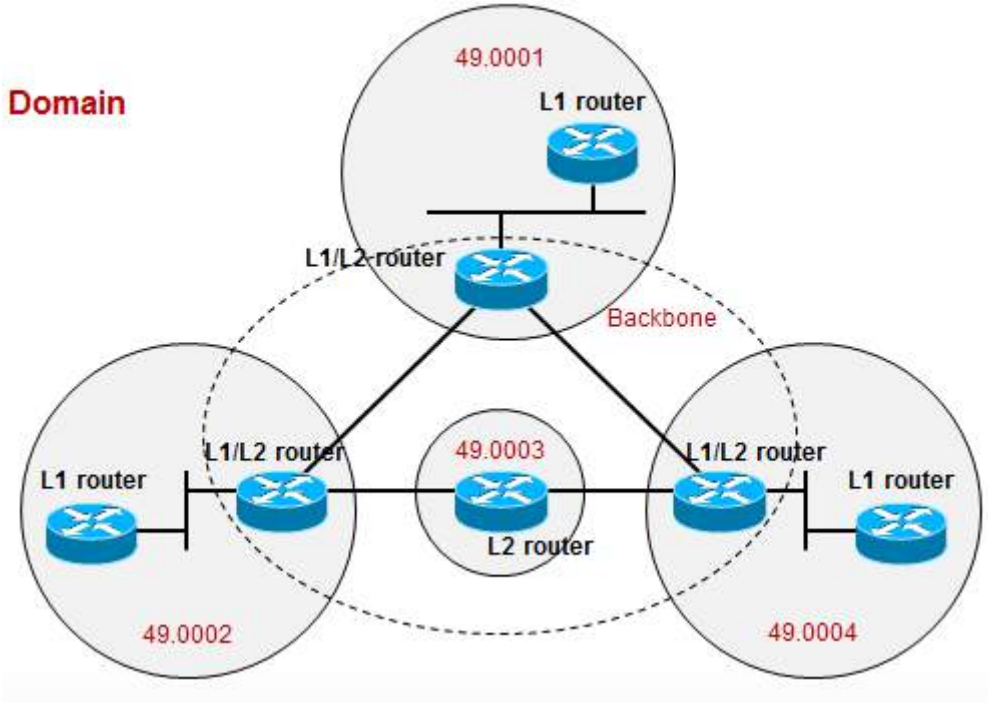
3 IS-IS 基础模块

3.1 IS-IS 路由选择层次

1. **IS-IS area**
   - IS-IS 允许将整个 Domain 划分为多个区域，采用两级的分层结构
   - 区域之间只能通过 L2 或 L1/L2 路由器进行互联
   - 一台路由器最多可有 254 个 area ID。当然，一般一台路由器就属于一个区域，多区域的情况，可能用于区域的合并、分割或变更等（请见下文）
   - 和 OSPF 不同的是，一台路由器必须整台属于一个区域，区域的边界不能在路由器上，也就说，不能一个接口属于某个区域，另一个接口属于其他区域
   - 区域内的所有路由器（除边界路由器外）只能和本区域内的路由器建立邻居关系

2. **Nodes level and area level**

- **L1 router**，维护着第 1 层的 LSBD，默认情况下，L2 的 LSP 不会被泛洪过来，因此 L1 router 的 LSDB 比较轻量，只有本区域内的 LSP。
- **L2 router**，连接了不同的 Leve1 area。储存着一个独立的 Level2 LSDB。
- **L1/L2 router**，同时保存着两套独立的 LSPD。同时运行着第 1 层和第 2 层的路由选择进程。在第 1 层的LSDB中维护着第1层的LSP信息 同时向其他L1 router通告它们是该区域的一个出口(这些L1 router 根据这个通告来形成一条指向给区域边界路由器的默认路由);它们也支持第 2 层的功能来与主干中的其他 router 通信，并维护一个独立于第 1 层 LSDB 的第 2 层拓扑数据库。
- Level1 area 是 L1 routers 和 L1/L2routers 的集合，Level2area 是 L2 routers 和 L1/L2 routers 的集合。



3. L1 router

- 位于普通 area 内部
- L1 router 只与本区域内的 L1 router（或 L1/L2 router）形成邻接关系
- L1 router 只有本区域内 Level1 的链路状态数据库，包含本区域内所有的 L1 router 的路由信息
- 通过与自己最近的 L1/L2 router 的 ATT bit 生成指向此台设备的默认路由作为出口路由
- 在转发时，如果目的地址在本 area 内，就直接利用 L1 LSDB 生成的路由转发报文，如果目的地址不在本区域内，则利用本 area 最近的 L1/L2router 作为区域外网络的出口，由此可能造成次优路由

4. L2 router

- 位于骨干 area
- 可以和其他的 L2（或 L1/L2）router 形成邻接关系
- L2 router 有 Level2 链路状态数据库，它包含所有区域间的路由信息

- 接收来自本区域内其他 L2 router 的报文，并按照目的地址将报文转发给其他 area 的 L2 router（或者转发到同一 area 的 L2 router）。接收来自其他区域的 L2 路由器的报文，并按照目的地址将报文转发

5. L1/L2 router

- L1/L2 router 通常位于 area 边界上
- 可以和本 area 内任何级别的路由器形成邻接关系；可以和其他 area 相邻的 L2 或 L1/L2 router 形成 L2 邻接关系
- 可能有两个级别的链路状态数据库：
  - L1（区域内）及 L2（区域间）
- 完成它所在的 area 和骨干之间的路由信息交换，既承担 L1 的职责也承担 L2 的职责
- 注意：一台 L1/L2 router 如果和其他 area 的路由器形成邻接关系，那么它将向本 area 的 L1 router 通告自己是个出口，具体方法是在生成本区域的 L1 LSP 时将报文中的 ATT bit 置 1

6. IS-IS 的层次性

ISIS 由两个层次：

a) Level2： Backbone，连续的 L2 router 的集合（含 L1/L2 router）；Backbone 是由所有的 L2（含 L1/L2 router）组成，Backbone 必须是连续的。
   注意：IS-IS 的 Backbone 不是某个特定的什么区域

b) Level1： 相对于单个区域的概念，由本区域中的 L1 router 构成，其路由信息发布到 Backbone 中。

注意：一个 IS-IS 路由域（routing domain）并不一定需要有两个层次，如果只部署一个区域的话，可能全都是 L1 或者全部是 L2，推荐用 L2，以得到比较好的扩展性。

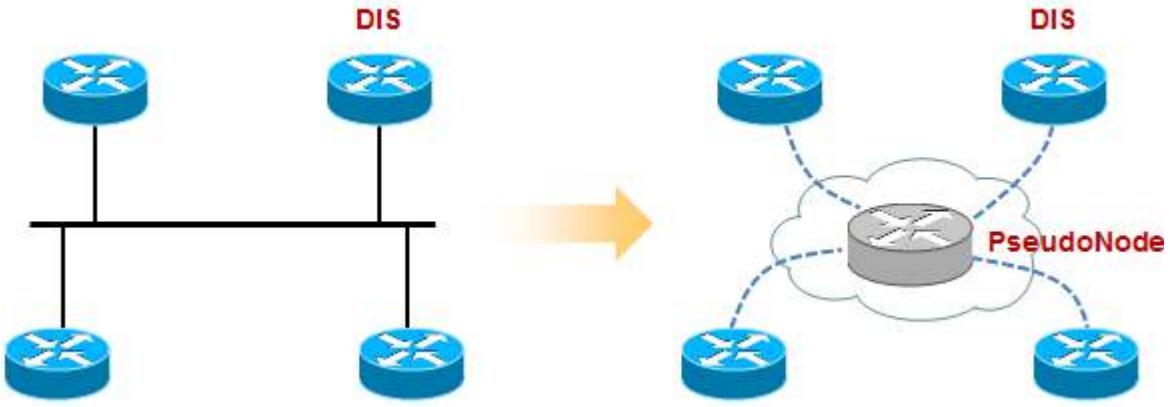
3.2 Designated IS and PseudoNode

1. Designated IS（DIS）

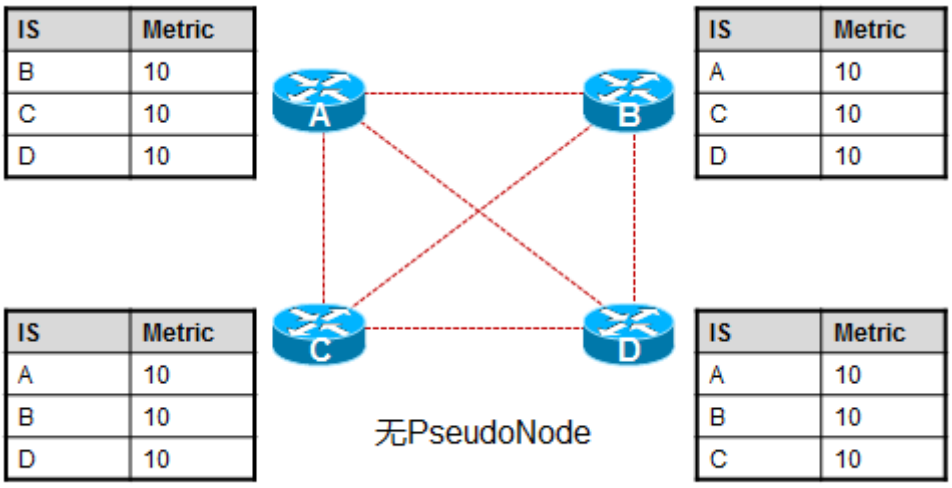
- 类似 OSPF 中 DR 的概念
- 在广播多路访问网络中，一台 router 会被选举为 DIS，点对点网络不需要 DIS
- Level1 有 level1 的 DIS，Level2 有 Level2 的 DIS，选举结果有可能不一样
- 与 OSPF 不同的是，DIS 是可抢占的，并且不存在备份 DIS。当一个 DIS 挂掉了，直接再选举。
- DIS 发送 HELLO 数据包的时间间隔是普通 router 的 1/3（默认是 3.3S），这样可以确保 DIS 出现故障的时候能够被更快速的被发现。

2. PseudoNode（PSN）

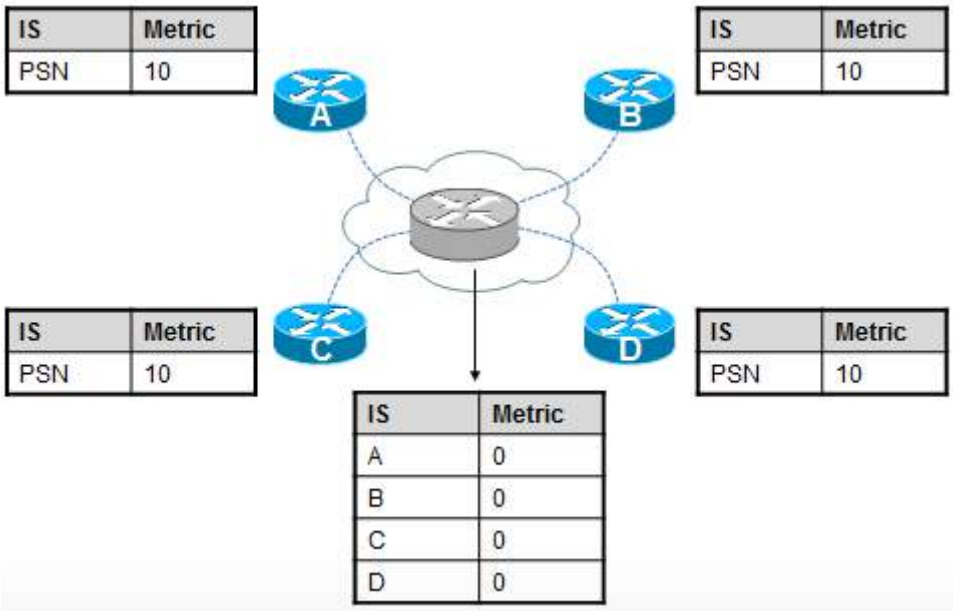
- 也叫伪节点，是广播多路访问网络中的一台虚拟路由器，由 DIS 创建
- DIS 在“伪节点 LSP”（类似于 OSPF 中的 LSA）中通告 LAN 中的所有邻居
- LAN 中的所有路由器在它们的 LSP 中通告自己与伪节点的连接性（自己和伪节点的邻接关系）
- PSN 可以减少泛洪扩散和数据库同步的数量。



伪节点代表了一个广播多路访问网络（包括连接到这个 LAN 上的所有 IS），实际上就是将整个广播多路访问网络“当成“一台路由器（如上图，灰色的路由器），在网络的其他地方看来，这就是一个整体。那么这个整体是不是得产生一个 LSP？这个 LSP 就由 DIS 来产生。



如果没有 PSN 伪节点，情况就像上图，LAN 上所有路由器的 LSPDB 如图所示，图中红色虚线为邻接关系，我们看到邻接关系多而且 LSPDB 庞大，那么有了 PSN 伪节点呢？看下图：



那么这样一来，LSPDB 的体积就大大减小了，注意，上面这张图只是一个抽象化、模拟化的概念，ABCD 四台路由器之间实际还是都有 ISIS 邻接关系的（在 IS-IS 广播网中，同一网段上的同一级别的路由器之间都会形成邻接关系，包括所有的非 DIS 路由器之间也会形成邻接关系，这一点与 OSPF 是不同的），只不过有了 DIS 之后，大家产生的 LSP 体积减小了。也就是说，虽然 IS-IS 广播网上所有的路由器之间都形成邻接关系，但 LSDB 的同步仍然依靠 DIS 来保证。而 LSP 的稳定性和可靠性也得到了保证。SPF 算法的运算自然也就快了。

包括 DIS 在内的 LAN 中的所有 router 都跟 PSN 建立毗邻关系，依靠伪节点在 LAN 上形成的这种毗邻关系。一但 DIS 在 LAN 中产生了伪节点，可靠性就得到了保证。它将周期性发送第 1 层和第 2 层的 Hello PDU 及 CSNP。Hello PDU 指明他是某一层的 DIS，CSNP 描述了所有 LSP 的汇总信息，包括 LSP ID、序列号、检验和剩余生存时间。它用组播来泛洪。CSNP 只是用来更正任何丢失的 PDU。

【补充】　非伪节点 LSP 代表一台 router，包括连接到这台 router 的所有 IS 和 LAN。

3. Designated IS（DIS）的作用

- **在广播子网中创建并向子网中的所有路由通告“伪节点 LSP”（LSP 就是类似 LSA 的东东）**
- **在广播子网中每隔 10S 周期性的发送 CSNP 来泛洪 LSP**

DIS 发送 HELLO 数据包的时间间隔是普通 router 的 1/3（默认是 3.3S）

DIS 发送 CSNP 数据包的时间间隔默认是 10S

当我们选定一个 DIS 后，DIS 就代表了这个广播网络，包括连接到这个广播网络上的所有的 IS

4. 选举 DIS 的顺序

1) 接口优先级 （默认 64），注意与 OSPF 不同的是，优先级为 0 的 IS 也参与 DIS 的选举
2) SNPA（Subnetwork point of attachment），比大
   - 在 LAN 中，SNPA 指的是 MAC 地址
   - 在帧中继网络中，SNPA 是 DLCI
3) 系统 ID，比大

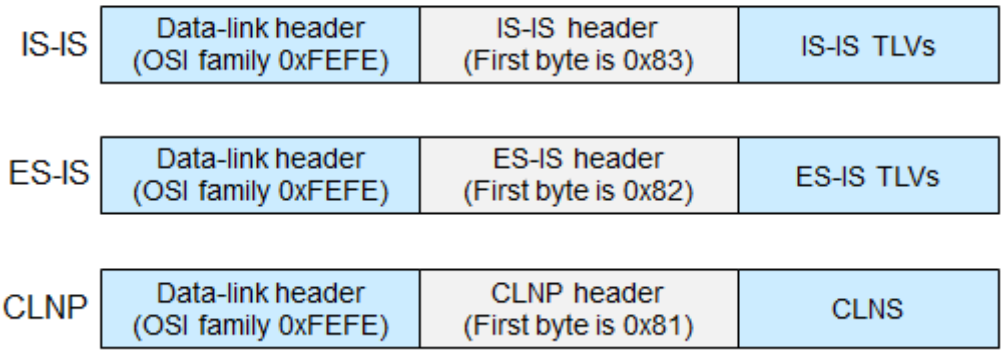
DIS 是可抢占的，而且没有备份的 DIS，也就是说当 DIS 挂掉了，立即重选。

3.3 OSI PDU

1. OSP PDU

- OSI 协议栈定义数据的单元为协议数据单元 PDU，因此 OSI 视数据 Frame 为数据链路层的 PDU.
- 有三种类型的 PDU：IS-IS 的 ES-IS 的和 CLNP
- IS-IS 和 ES-IS 不将路由信息放在 IP 或 CLNP 分组中，而是直接放在数据链路层 Frame 中（这样封装效率更高）。
- 真正的 CLNP 数据会被封装在 CLNP 头后，然后进行二层的封装。

- Network PDU = datagram , packets
- Data-link PDU = frame



2. IS-IS 的 PDU 有 4 种:

- **HELLO PDU(ESH 、ISH 和 IIH)**

  用于建立和维护毗邻关系

  ESH 是 ES 发送到 IS 的

  ISH 是由 IS 发送到 ES

  IIH 则是 IS 之间传送的。

注意 ESH 和 ISH PDU 都是 ES-IS 的 PDU 而 不是 IS-IS 的 PDU

- **LSP**

  用来发布链路状态信息，就是有点类似 OSPF 的 LSA 的东东

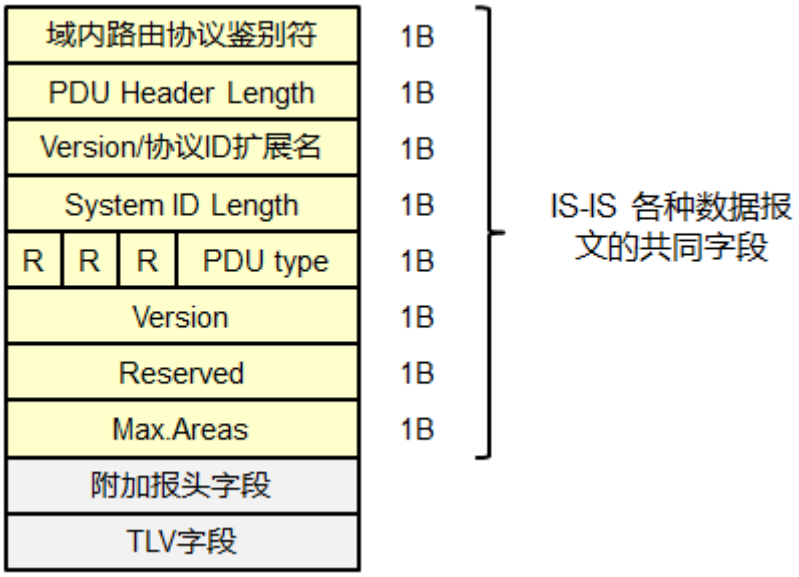
- **CSNP（完全序列号 PDU）**

  用来发布一台 router 上的完整的链路状态数据库，CSNP 用来告知其他 router 他们自己的数据库中可能过时或者丢失的 LSP。

- **PSNP（部分序列号 PDU）**

  用来**确认和请求**链路状态信息

3. IS-IS 报文格式



上图为 IS-IS 报文的格式，黄色背景色部分为 IS-IS 各种类型报文的共同字段。接下来每种数据包会有自己的附加报头字段，然后就是 TLV。

- 域内路由协议鉴别符： 这是分配给 IS-IS 的网络层标示符，值为 0x83
- PDU header Length： 数据报头字段的字节数（这个字节总数包括了附加报头字段的字节数在内）
- Version/协议 ID 扩展名： 值为 1
- SystemID Length： 标识系统 ID 的长度，值为 0 表示长度为 6B，值为 255 表示长度为 0，其他系统 ID 字段长度可能值为 1-8 字节。
- PDUtype： PDU 类型。这是一个 5bit 的字段，用于标识 IS-IS 数据包的类型。值为 15 表示 L1 LAN IIH；值为 16 表示 L2 LAN IIH；值为 18 表示 L1 LSP；值为 20 表示 L2 LSP；值为 24 表示 L1 CSNP；值为 25 表示 L2 CSNP；值为 26 表示 L1 PSNP；值为 27 表示 L2 CSNP。
- Version： 值为 1
- MAX.Areas 最多区域地址。表示我们可以为一个路由器配置多少个不同的区域前缀。值

为 0 表示最多支持 3 个区域地址数，默认情况下值为 0，取值范围为 1-254 的实际数字。

IS-IS 的数据包都有上面的共同 8B，接下去附加报头字段各有不同，然后就是 TLV 字段。在抓包的时候，TLV 中的三元：类型、长度、可变长的内容，中的类型及长度，抓包软件可能不呈现，但是可以从抓包结果的二进制编码中找到类型和长度的内容。

4. **IS-IS Hello 消息（IIH PDU）**
   - Hello 数据单元：定期发送，用来确定相邻的其他 IS 是否在运行 IS-IS，已建立邻接关系，交换 LSP，达到 LSDB 同步
   - 在 IS-IS 里有三种 HELLO 包：一种是点对点接口的（Point-to-Point IS to IS hello PDU）、一种是 LAN 上对 L1 router（Level1 LAN IS to IS Hello PDU）的、一种是在 LAN 上对 L2 router 的（Level2 LAN IS to IS Hello PDU）
   - 在 LAN 上 L1 和 L2 IIH PDU 发送到不同的组播 MAC 地址 L1 为 0180-C200-0014，L2 为 0180-C200-0015

   更详细内容，见下文。

5. **IS-IS LSP 消息及 CSNP、PSNP 请见 IS-IS 链路状态数据库章节**

3.4 IS-IS 毗邻关系及网络类型

1. **对不同的网络类型，IS-IS 的邻接关系建立方式有所不同，IS-IS 支持如下两类网络：**
   - 点对点网络
   - 广播网络

2. **IS-IS 邻接关系建立需要遵循的基本原则**
   - 只有同一层次的相邻路由器才有可能建立邻接关系
   - 对于 Level1 router 来说要求 area 号一致
   - 要进行同一网段检查（邻接必须在同一个网段）
   - MTU 隐含检查

     IS-IS 的 IIH 会打上许多 padding，hello 报文填充为出接口 MTU。这种机制在低宽带链路上会比较伤，可以在接口上 no isis hello padding 取消 padding，这条命令，如果要打，建议在链路两端都加上。实验亲测的结果是，如果一边 no 掉 hello padding，邻接仍然能建立。

3. IS-IS 的 IIH 概述

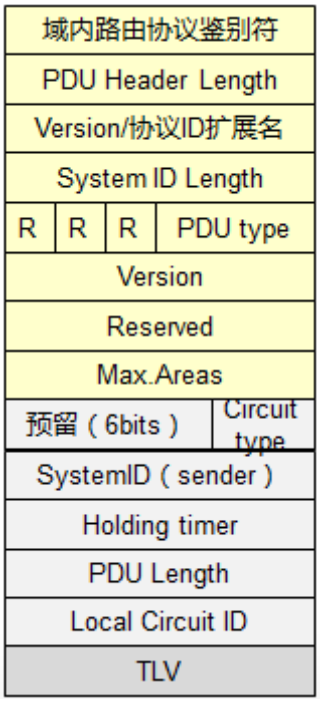
- Hello 数据单元：定期发送，用来确定相邻的其他 IS 是否在运行 IS-IS，已建立邻接关系，交换 LSP，达到 LSDB 同步
- 在 IS-IS 里有三种 HELLO 包：
  一种是点对点接口的（Point-to-Point IS to IS hello PDU）、
  一种是 LAN 上对 L1 router（Level1 LAN IS to IS Hello PDU）的、
  一种是在 LAN 上对 L2 router 的（Level2 LAN IS to IS Hello PDU）
- 在 LAN 上 L1 和 L2 IIH PDU 发送到不同的组播 MAC 地址： L1 为 0180-C200-0014，L2 为 0180-C200-0015
- 和所有的 IS-IS 数据包一样，HELLO 包也由报头和 TLV 组成。

4. HELLO 间隔、HELLO 乘数和 HELLO 保持间隔

- HELLO interval 是 hello 数据包的发送周期，有 25%的抖动来减少网络中 IIH 同步传输的可能性。在 CISCO IOS 中，普通路由器 hello interval 默认 10S，DIS 默认 3.3S。
- HELLO multiplier（HELLO 乘数），是丢失多少 HELLO 包后，邻接路由器宣告邻居失效，默认为 3
- HELLO holdtime 是 IS-IS 路由器收到 2 个连续 hello 数据包的最大允许时间间隔。Hello holdtime=hello interval × hello multiplier。因此 hello holdtime 默认为 30S。如果一台路由器在 hello holdtime 超时了还没收到任何 hello，那么宣告邻居失效。

5. 点对点链路上的 IS-IS 邻接

在点对点链路上默认 10S 发送一次 IIH。



路由器会检查收到的 IIH，确认里面的各项参数。

- Circuit type： 01 表示 L1 路由器，10 表示 L2 路由器，11 表示 L1/2 路由器
- SystemID： 产生该 hello 的 IS 的 systemID（有的书上写的是 SourceID，同义）
- Holding timer： 保持时间，默认 30S
- PDU length： 整个 PDU 的长度，包括头
- Local Circuit ID 本地电路 ID ,是一个链路标示符。由发送 Hello PDU 的路由器分配给这条电路的标识，并且在路由器的接口上是唯一的。在点到点链路的另一端，Hello 报文中的本地电路 ID 可能或也可能不为同样的值。
- TLV 部分包含了系统特性的字段，如下图：

```
⊞ Restart Option (3)
⊞ Point-to-point Adjacency State (1)
⊞ Protocols Supported (1)
⊞ Area address(es) (4)
⊞ IP Interface address(es) (4)
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (169)
```

详细考虑一下邻接关系建立过程中的细节：

- 当路由器在一个点对点链路上收到 ISH 报文，路由器会检查本地的邻接数据库看看报文的源 systemID 是否存在，如果存在，则忽略该 ISH，如果不存在，则接收路由器创建新的邻接关系，设为“初始化”状态，系统类型为“未知”。随后发送 IIH 回应。如果这台路由器又从这个新的邻居收到一个 IIH 报文，路由器将邻接关系设为 UP，系统类型为 IS，但是这里有个问题，就是本地的 IIH 发出去后，我并不知道对方是否收到，那么就有可能报文在中途丢失而出现一边 UP 一边 DOWN 的情况。因此我们引入了三次握手机制，请看下文。
- 在默认的实现模式中，IIH 报文会被填充至出接口 MTU 大小（使用增加 padding 的方式，如上图），路由器会对比本地接口 MTU 和收到的 IIH 大小，确保在形成邻接前能够处理来自邻接点的最大可能的数据包。
- 路由器始发 IIH 时，根据我们所做的配置来设置报头中的 circuit type：L1、L2 或 L1-2。
- 路由器采用 8bits 的 local circuit ID 给每个点对点链路分配一个本地唯一的链路标示符。
- IS-IS 要求整个域内的 SystemID 长度一致。如果收到的 IIH 中，“System ID Length”字段与本地的 sysID 长度不一致，则忽略该 IIH。
- 单台路由器配置的区域地址最大数目必须与邻接邻居一致，默认情况下 CISCO IOS 路由器最多支持 3 个区域地址。最新的 IOS 版本可以设置为 255.如果两台路由器支持的最大区域地址数目不一致，也就是 MAX.Areas 字段不匹配，则忽略 IIH 报文。

三次握手机制：

我们引入一个新的 TLV（类型为 240）,称作“point-to-point adjacency state”，如果不支持该新特性的设备收到包含该 TLV 的 IIH，将忽略这个 TLV，这样可以保证向后兼容。

```
Point-to-point Adjacency State (1)
    Adjacency State: Up
```

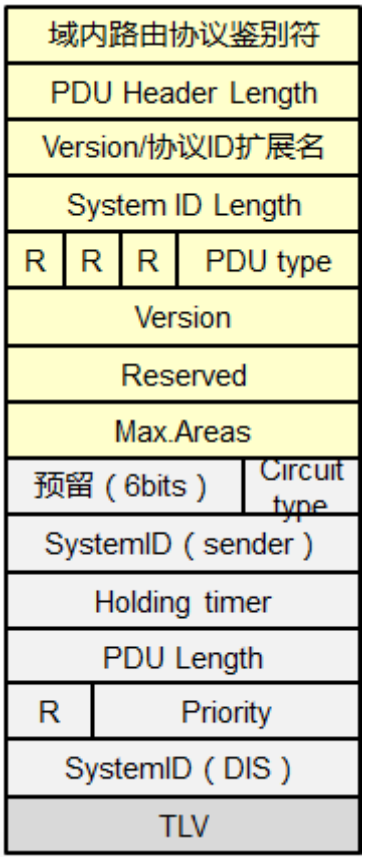
这个 state 有三种取值：down、init、up

6. 多路访问链路上的 IS-IS 邻接

LAN 上的 IS-IS 邻接与 P2P 链路还是些不同的：

- 不是通过 ISH 触发，一旦接口激活了，路由器就在接口上泛洪 IIH 报文
- 产生伪节点，伪节点作用由选举的 DIS 扮演
- Lan 中的 router 与同一 LAN 中的所有具有相同区域 ID 和层次的其他 router 都建立毗邻关系。
- 在 LAN 上 L1 和 L2 IIH PDU 发送到不同的组播 MAC 地址，L1 为 0180-C200-0014，L2 为 0180-C200-0015

以下就是 LAN 中的 IIH 的报文格式：



- **Priority：** 优先级。接口的 DIS 优先级，用来在广播 LAN 中选举 DIS。优先级数值越高，路由器成为 DIS 的可能性越大。

- **SystemID(DIS)：** 局域网 ID。由 DIS 的 SysID 与 1 字节的伪节点 ID 组成，LAN ID 用来区分同一台 DIS 上的不同 LAN。

```
⊟ ISIS HELLO
    Circuit type              : Level 1 and 2, reserved(0x00 == 0)
    System-ID {Sender of PDU} : 0000.0000.0002
    Holding timer: 30
    PDU length: 1497
    Priority                  : 64, reserved(0x00 == 0)
    System-ID {Designated IS} : 0000.0000.0002.02
  ⊞ Protocols Supported (1)
  ⊞ Area address(es) (4)
  ⊞ IP Interface address(es) (4)
  ⊞ Restart Signaling (3)
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (163)
```

用到的 TLV 有：

```
⊟ Protocols Supported (1)
    NLPID(s): IP (0xcc)
⊟ Area address(es) (4)
    Area address (3): 49.0001
⊟ IP Interface address(es) (4)
    IPv4 interface address: 10.1.123.1 (10.1.123.1)
⊟ Restart Signaling (3)
  ⊟ Restart Signaling Flags: 0x00
      .... .0.. = Suppress Adjacency: False
      .... ..0. = Restart Acknowledgment: False
      .... ...0 = Restart Request: False
⊟ IS Neighbor(s) (12)
    IS Neighbor: cc:02:2d:0c:00:00
    IS Neighbor: cc:01:2d:0c:00:00
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (255)
  Padding (149)
```

其中，IS neighbor 放的是邻接的 MAC

Lan 上的路由器与其他所有路由器建立毗邻关系，而 OSPF 中，LAN 上的路由器仅与 DR 建立毗邻关系。

形成 LAN 邻接关系：

- 当 LAN 接口激活 IS-IS，路由器立即发送 IIH，其中 IIH 包含一个 LANID，这个 LANID 在报文中显示为

DIS，初始的时候是本地的 SystemID+唯一的本地 circuit ID 构成。路由器同时开始侦听 ESH、ISH、IIH 等报文从而视图发现邻接体。接下来开始 DIS 选举进程。

- **路由器收到 IIH 的时候，会进行一系列的校验动作，例如 SystemID 长度是否匹配啊，max area address 是否匹配啊，验证密码是否正确啊，区域 ID 什么的。**
- **一旦路由器收到 IIH，会检查 hello 包发送者的邻接情况，如果邻接关系已经建立，则重置该邻接的 holdtimer，如果邻接关系没有建立，接收路由器创建邻接关系并标识邻接类型（L1、L2），设置为初始化状态，直到后续收到 hello 数据包确认邻接关系才算建立。**
- **路由器发送的 hello 包里包含整个 LAN 上发送 hello 的所有邻居的 MAC 地址，路由器使用一种简单的机制确认二步通信，也就是在收到的 hello 包中，如果看到自己的 MAC 那就算二步通信正常。如果没有看到自己的 MAC，那么证明二步通信不正常，邻接关系则维持在初始化状态。**

3.5 IS-IS 链路类型

1. **IS-IS 支持的链路类型**
   - **点对点链路**

     如 PPP、HDLC

     不选举 DIS，链路建立起来后用 CSNP 来触发数据库同步。
   - **广播型多路访问链路**

     选举 DIS。在该网络中该 DIS 会在它所参与的每一路由选择层上（无论是 L1 还是 L2）以及它连接着的每一个 LAN 上都生成并泛洪扩散新的伪节点 LSP。LAN 上每台路由器都与其他所有路由器和 DIS 建立毗邻关系，不选举备份 DIS 路由器。选举出的 DIS 并不能保证一直为 DIS。
   - **非广播型多路访问链路（实际上 IS-IS 并不理解 NBMA 介质）**

     IS-IS 在 NBMA 网络中运行的时候，有一些问题。

     如果是帧中继 P2P 子接口，那么不会有问题，这是推荐的部署方法

     如果是帧中继主接口或者是 P2MP 子接口，则要求是全互联的 PVC，具体请见本文档相关章节

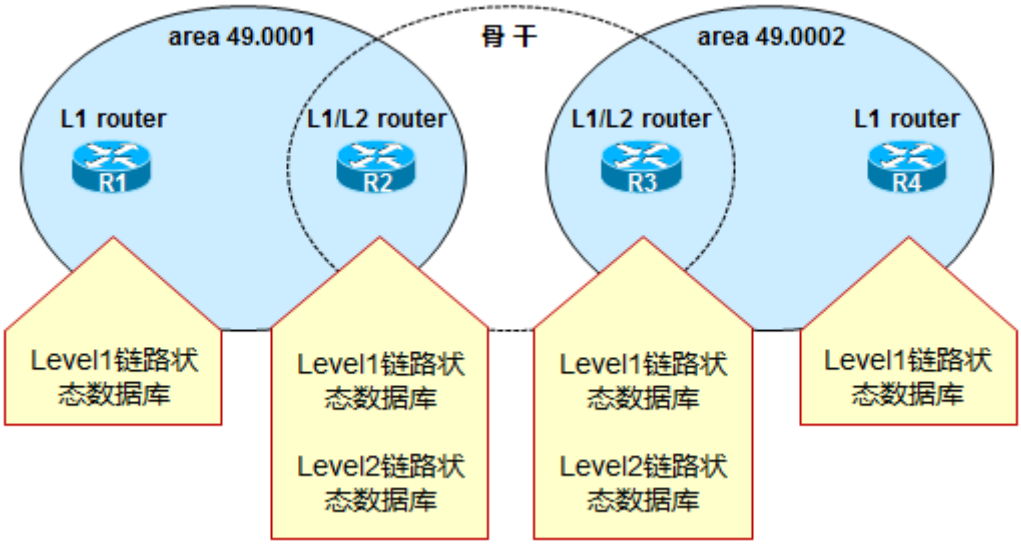
4 IS-IS 链路状态数据库

4.1 概述

区域内的路由器交换 LSP 的过程称为 flooding，从区域内的邻接路由器收到的 LSP 存储在本地路由器的 Level1 链路状态数据库里。IS-IS 域中的每一个区域都有一个唯一的 Level1 链路状态数据库。通过 SPF 算法再计算出来最短路径。

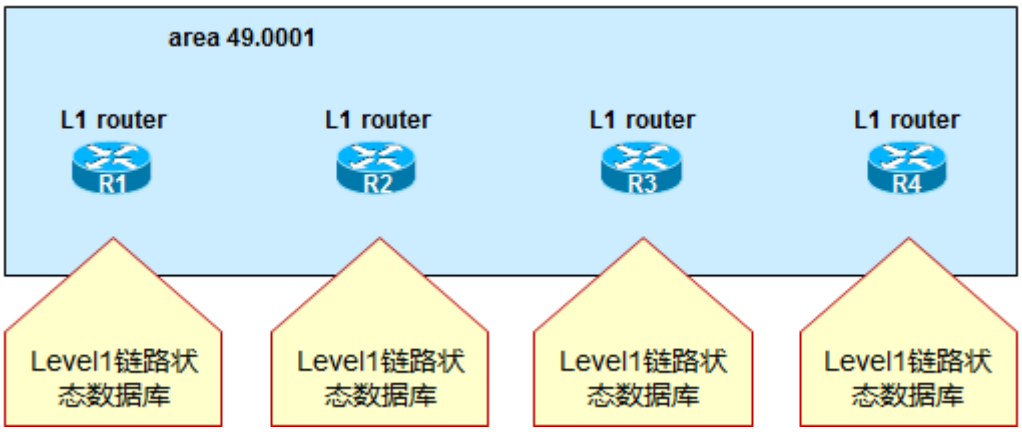
IS-IS 支持两个层次的路由选择：Level1 及 Level2。Level1 支持区域内的路由、Level2 支持区域间的路由。Level2 LSP 中封装了从骨干网络相连的路由器中获取的描述 Level2 网络拓扑结构信息。完整的 Level2 网络拓扑结构可以通过在 Level2 链路状态数据库中使用 SPF 算法获得。

在一个典型的网络设计中，多个独立的 Level1 区域通过骨干网相连，骨干网由具有 Level2 路由选择功能的路由器构成。Level1-2 路由器拥有分别支持 Level1 及 Level2 路由选择的两个独立的链路状态数据库。

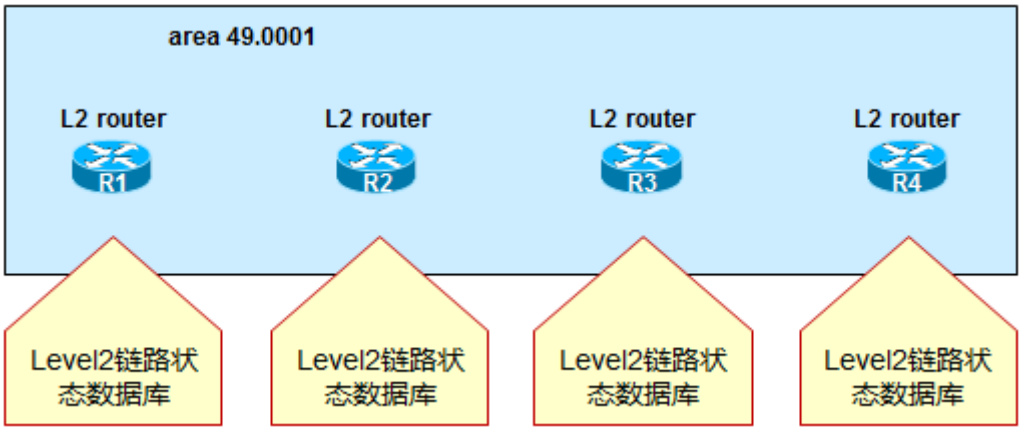


从上面这个图，我们可以看出，R1、R4 作为 L1 router，只维护 Level1 的链路状态数据库，R2、R3 是 L1/L2 router，因此维护两个互相独立的 L1 链路状态数据库及 L2 链路状态数据库。那么，如果骨干中，若有一台路由器并不与 L1 router 相连，可以配置为 L2 router。

当然，如果是单区域运行，你可以采用全 L1 router，也可以采用全 L2 router 的解决方案。



上图是单区域的 Level1 路由选择域



上图是单区域的 Level2 路由选择域

4.2 LSP（LinkState Protocol Data Unit）格式

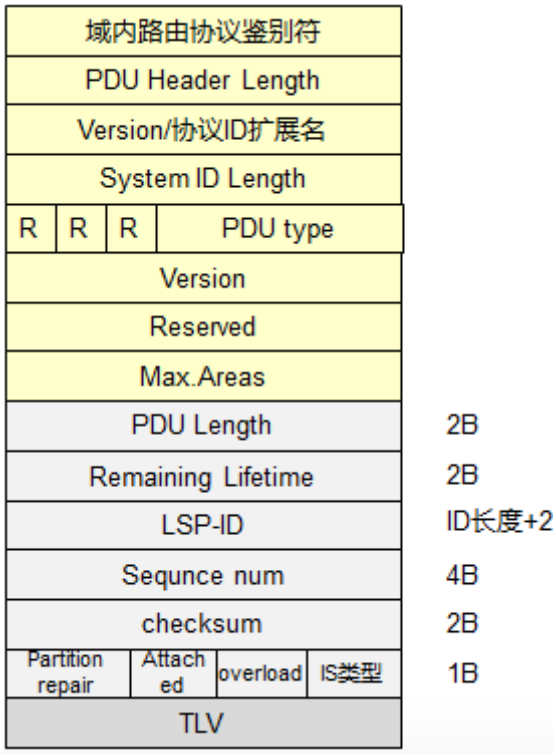
1. LSP 概述

- LSP 链路状态数据单元
- 链路状态报文用来在区域中传播链路和节点信息（描述本地网络的拓扑结构）
- LSP 的泛洪和交互最终构成 IS-IS 链路状态数据库
- LSP 分为两种：Level1 LSP PDU 和 Level2 LSP PDU
- Level2 LSP 包含在 IS-IS 里所有可能到达的前缀信息。Level1 LSP 只用于本地区域

2. LSP 主要包含如下信息

- **区域信息**
- **邻接路由器**
- **IP 子网**
- **度量值**
- **认证信息**

3. LSP 报文结构

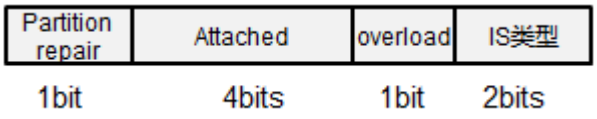


Remaing lifetime LSP 到期前的生存时间

LSP ID LSP 的身份标识

Sequence num LSP 的序列号

Type block 这是 1B，里头包含不少重要的位：



```
Type block(0x03): Partition Repair:0, Attached bits:0, Overload bit:0, IS type:3
    0... .... = Partition Repair: Unsupported
  .000 0... = Attachment: 0
    0... = Error metric: Unset
    .0.. = Expense metric: Unset
    ..0. = Delay metric: Unset
    ...0 = Default metric: Unset
  .... .0.. = Overload bit: Unset
  .... ..11 = Type of Intermediate System: Level 2 (3)
```

Partition repair: 第 8 位如果 LSP 始发路由器支持区域划分修复则设置该字段

Attached bits： 第 4-7 位，用于设置该 LSP 与带有如下可用度量的另一个区域的关联关系：第 4 位-默认；第 5 位-延迟；第 6 位-开销；第 7 位-误差

Overload： 第 3 位 如果设置则表示 LSP 始发路由器的链路状态数据库已超载，内存空间和 CPU 资源已受到限制

IS type： 第 1、2 位，用于指示路由器的类型是 L1router 还是 L2router

不同的 TLV 字段值可以包含在 LSP 中用来通告各种不同的路由选择信息。

4.3 LSP header 详解

1. Remaning timer

LSP remaining time 字段中两个重要的阀值是：LSP maxage 及 LSP refresh interval

LSP maxage，定义了 LSP 的生存时间，默认为 20 分钟，也就是 1200S，可使用 lsp-max-lifetime 命令修改。当一条 LSP 从一台路由器始发，其 remaning timer 就被设置为 LSP maxage，然后在区域内泛洪。同时在路由器本地，也会为这个 LSP 的 remaining time 进行倒计时，当倒计时达到 LSP refresh interval 时，也就是减少的时间等于 LSP refresh inverval 时，LSP 的始发路由器会重新产生 LSP 并泛洪出去。否则该 LSP 将继续生存下去，知道剩余生存时间字段值减少到 0，然后被清楚出网络。

LSP refresh inverval，是一个 LSP 周期性的泛洪时间，默认 15 分钟，也就是 900S，可通过 lsp-refresh-inverval 来修改。

如果某台路由器在 LSP refresh interval 来临后没有刷新其产生的 LSP，并且 LSP remaining timer 字段已经减少到 0，那么所有拥有该 LSP 拷贝的路由器，将在 60S 宽限期后将该 LSP 拷贝从自己的链路状态数据库中清楚。这个 60S 的时间是 ZeroAgeLifetime，零寿命生存时间，不可配置。

2. LSP ID（链路状态数据包标示符）

用来唯一地标识一个 LSP 及鉴别源路由器

如 R1.01-00 其中 R1 为 SystemID（这是基于主机名的 SystemID），01 为 PseudonodeID 伪节点 ID，如果是伪节点发的，则为非 00，00 为分片号。

我们的每个 IS 产生的 LSP 一般就一个，LSP 的第一段编号为 0。但是如果我这 LSP 特别大，超过了接口 MTU，那么就有可能先进行分片然后再传，再用这个分片号来表示（0 为第一段，1 位第二段，依此类推），单个 LSP 最大长度是 1492 字节，最多可以分 255 片。IS-IS 通过在产生 LSP 的时候就进行分片，而不是先产生大 LSP，然后在网络层（例如 IP）分片，从而达到降低 CPU 资源浪费的负面效应。

ISO 10589 中要求把 hello 数据填充到输出接口的 MTU 大小 ,这意味着邻接关系的两端通常使用同一个 MTU 来构成邻接关系。

3. LSP 序列号

4B，是一个无符号整数。从 0 开始，每次增加 1。当一个路由器最先连接到网络中，它产生的第一个 LSP 的序列号就为 1。当网络发生变化时而产生的 LSP 序列号也加 1。LSP 序列号在数据同步过程中起到关键作用，主要体现在用于区分 LSP 的新旧。

4. 链路状态数据包校验和

Checksum 的计算从 LSP 中剩余生存时间字段之后的字段开始知道数据包末尾。LSP 拷贝在网络中路由器之间传播时其 checksum 字段不被修改。

一般来说，任何一个检查到被破坏的 LSP 的路由器都会通过发送一个剩余生存时间重置为 0 的被破坏的 LSP 的拷贝来启动一个清除进程。这种方式有效的保证了网络中的其他路由器清除这个被破坏的 LSP。被破坏的 LSP 不参与路由计算，当然也不会在网络中被泛洪。

如果一条 LSP 在传输过程中不断被破坏，它会被其他路由器不断清除，同时源路由器不断重发。这就产生了 LSP 破坏风暴。CISCO IOS 允许路由器忽略被破坏的 LSP 并只在本地将错误写进日志文件。通过 lsp-ignore-errors 命令开启。

5. Type Block

- **Partition repair**

  ISO10589 中定义了如何通过建立一条穿过 Level2 骨干的 Level1 路由来修复一个被隔离的 Level1 区域。这主要通过在每个区域中选举出一台具有 level2 功能的路由器作为区域指定的 level2 IS 并在区域间建立一条被称作虚邻接或虚链路的特殊邻接关系来实现。

  虚链路实现了穿过骨干的 level1 修复路径。区域指定路由器通过设置其 level1 LSP 的 partition repair

位来通告虚链路，从而把虚链路存在的信号通知给 level1 router 以便实现区域间的数据转发。当然，这个特性极其少用，在 CISCO IOS 上的支持，还得查查相关的文档。

- **Attached**

区域关联。L1/L2 路由器在其生成的 L1 LSP 中设置该字段以通知同一区域中的 L1 路由器自己与其他区域相连。通常来说就是 L2 骨干区域相连。当 L1 区域中的路由器收到 L1/2 路由器发送的 ATT 位被置位的 L1 LSP 后，它将创建一条指向 L1/2 路由器的默认路由，以便数据可以被路由到其他区域。虽然 ATT 位同时在 L1 LSP 和 L2 LSP 中进行了定义，但是它只会在 L1 LSP 中被置位，并且只有 L1/2 路由器会设置这个字段。

ISO 10589 中定义的 IS-IS 区域是 stub 区域，Attached 定义了四种 metric 类型，CISCO IOS 只支持其中的 default metric。

这里就有一个隐含的问题，因为 Level1 router，并不知道 level2 的（本区域外的）路由信息，取而代之的是寻找最近的 L1/L2 router 前往目的地，那么就有可能会出现次优路径的问题，为了解决这个问题，IS-IS 引入了一个叫做路由泄露的机制。

6. Overload

表示一台路由器资源的可用状态。如果该 bit 置位了，则表明该路由器发生了超载。超载是指路由器的性能因为 CPU 与内存资源不足而受到抑制。被设置了 overload 字段的那些 LSP 不会在网络中扩散并且在计算通过 overload 路由器的路由时也不会采用这些 LSP。换句话说，overload 路由器被传输路绕开了，那些 overload 路由器作为最后一跳的路由才参与计算。

使用 set-overload-bit 来手动设置 overload 位。

7. IS-type

表明了该 LSP 是来自 L1 router 还是 L2 router，这个字段 2bits。

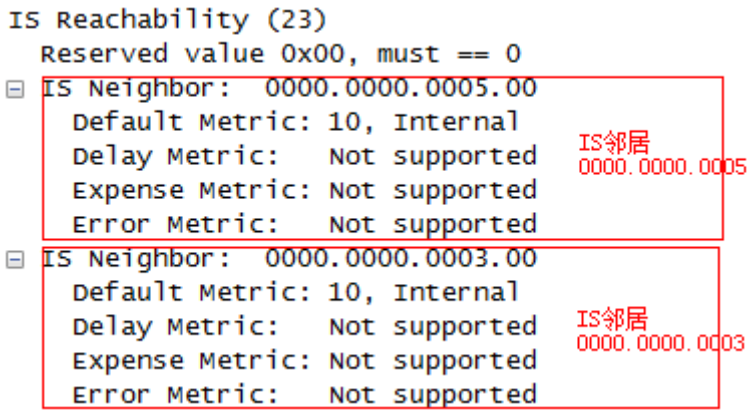
00-未使用；01-level1；10-未使用；11-level2

4.4 LSP 的各类 TLV

根据不同的 IS-IS PDU 类型和特定的网络环境，紧跟在各种类型 IS-IS PDU 之后的是 TLV（Type/Length/Value）字段，PDU 报头与 TLV 字段构成了一个完整的 IS-IS PDU。在 ISO10589 和 RFC1195 这二种当前 IS-IS 标准中，使用代码（code）这个词，而不是类型（type），但由于 TLV 用于其他协议标准中，故 TLV 比 CLV 在网络文献中使用的多，在这里我们也使用 TVL 代替 CLV。在 IS-IS PDU 所使用的各种 TLV 中，既有 ISO 10589 中定义的，也有 RFC 1195 中定义的。ISO 中定义的 TLV 用于 CLNP 网络环境，但是其中的大多数也用于 IP 网络环境。RFC 中定义的 TLV 只用于 IP 环境。对于一个 IS-IS PDU，后面既可以携带支持 CLNP 协议的 TLV，又可以携带支持

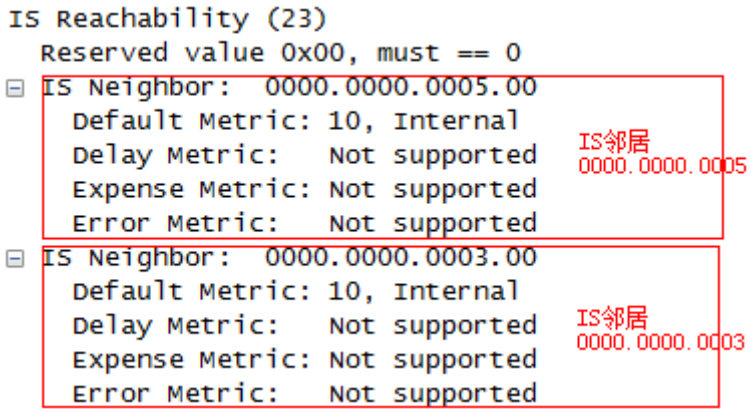
IP 协议的 TLV。如果一个路由器不能识别一个 TLV，那么将忽略它。

1. Level1 路由的 LSP TLV 字段

TLV	类 型值	定义来源	备注
Area address	1	ISO10589	列出了路由器上配置的区域地址组。它只出现在非伪节点的 LSP 中，并且如果 LSP 被划分成不同的片段，那么该 TLV 字段出现在第一段。
中间系统邻接路由器 IS reachability	2	ISO10589	 列举邻接的 L1-router，注意啊，我们这小节说的是 L1 router LSP 的 TLV 字段啊。这里头除了包含邻接 IS 的系统 ID，还有一堆 metric
终端系统邻接路由器	3	ISO10589	只在 L1 router 的 LSP 中出现。列举了邻接的 L1-router 和终端主机。例如通过 ES-IS 协议搜寻到的 ISO CLNP 工作站。
认证信息	10	ISO10589	用于认证
IP 内部可达性信息	128	RFC1195	列举了直连的 IP 地址前缀列表 ,它只在非伪节点 LSP 中使用。每个 IP 前缀分配了一堆的 metric。
支持协议 Protocols support	129	RFC1195	列举了集成型 IS-IS 协议支持的第三层协议，目前只有 CLNP（NLPID 0x81）和 IP（0xCC）
IP 接口地址	132	RFC1195	包含源路由器上配置的一个或多个 IP 地址 ,在 CISCO IOS 中为最高的 loopback 口 IP

2. Level2 路由的 LSP TLV 字段

TLV	类 型值	定义来源	备注
Area address	1	ISO10589	列出了路由器上配置的区域地址组。

中间系统邻接路由器 IS reachability	2	ISO10589	
分离指定层 2 中间系统	4	ISO10589	这种 TLV 通过在 L2 router 之间建立一条穿过骨干网的虚链路来支持区域修复
前缀邻接路由器	5	ISO10589	搜集可达的 NSAP 前缀信息，它只用于区域间 ISO CLNP 路由选择（level2 路由）
认证信息	10	ISO10589	用于认证
IP 内部可达性信息	128	RFC1195	列举了直连的 IP 地址前缀列表
支持协议 Protocols support	129	RFC1195	列举了集成型 IS-IS 协议支持的第三层协议，目前只有 CLNP（NLPID 0x81）和 IP（0xCC）
IP 外部可达性信息	130	RFC1195	收集通过其他路由协议得到的 IP 路由信息
域间路由选择协议信息	131	RFC1195	暂不支持
IP 接口地址	132	RFC1195	包含源路由器上配置的一个或多个 IP 地址，在 CISCO IOS 中为最高的 loopback 口 IP

4.5 IS-IS metric

1. **你看我们在 LSP 中的许多地方都出现了 IS-IS metric，例如：**

```
IP Interface address(es) (4)
  IPv4 interface address: 4.4.4.4 (4.4.4.4)
IS Reachability (23)
  Reserved value 0x00, must == 0
  IS Neighbor:  0000.0000.0005.00
    Default Metric: 10, Internal
    Delay Metric:   Not supported
    Expense Metric: Not supported
    Error Metric:   Not supported
  IS Neighbor:  0000.0000.0003.00
IP Internal reachability (36)
  IPv4 prefix: 4.4.4.0/24
  IPv4 prefix: 10.1.34.0/24
  IPv4 prefix: 10.1.45.0/24
    Default Metric: 10, Internal, Distribution: up
    Delay Metric:   Not supported
    Expense Metric: Not supported
    Error Metric:   Not supported
```

2. 一共有四种 metric 类型：

- Default metric　　所有 IS-IS 路由器都支持，经常被解释成跟带宽成反比的度量方式，越小越优先
- Delay metric　　可选，链路的传输延迟，不支持
- Expense metric　　可选，链路的传输开销，不支持
- Error metric　　可选，链路的出错概率，不支持

3. 每一种 metric 类型，都是 8bits，形式如下：

1bit	1bit	6bits
0	I/E	Default metric
1	I/E	Delay metric
1	I/E	Expense metric
1	I/E	Error metric

如果最高位（最左边的位）为 0，则表示 metric 类型为 default metric，如果为 1，则表示不支持该 metric 类型。第 7bit，是内部/外部标志位，设置为 0 表示是内部度量，设置为 1 表示是外部度量。

在 CISCO IOS 路由器中，IOS 无法自动分配基于带宽的链路（接口）度量，无论链路的带宽多少，所有的接口默认 IS-IS 度量都是 10，当然你可以修改。一完整路径的最大度量值是 1023。当然 IETF 对 default metric 做了扩展。

4. IS-IS 度量值的扩展

前面说了 IS-IS 四种 metric 类型，每种 metric 其实真正能用的 bits 就是低 6 位，等于说，接口 metric 最大也就是 63，我们提出几个新的 TLV，来弥补这个缺陷。

扩展的中间系统可达性 TLV（类型 22）

是类型 2TLV 的扩展形式

扩展的 IP 可达性 TLV（类型 135）

是类型 128TLV 的扩展形式

流量工程 RouterID TLV（类型 134）

主要用于 TE

- **扩展的中间系统可达性 TLV（类型 22）**

主要用于代替类型 2TLV 以提供更大的度量值。另外，它还支持基于 IS-IS 的 MPLS TE

Type	Length	Value
22	Value 的长度	3B 的 default metric，够你臭屁的吧？ 1B 的 sub-TLV 长度 6B 的 systemID+1B 的 PSN ID 0-244B 的 sub-TLV 主要用于 MPLS TE

- **扩展的 IP 可达性可达性 TLV（类型 135）**

被用来代替 TLV128。

全局配置命令 metric-style wide 可以让 cisco 路由器运行适当的 IOS 版本来发送携带 宽度量值 的 LSP 同时也能接受并理解解释携带 TLV22 和 TLV135 的 LSP

全局配置命令 metric-style transition 可以让原来的窄度量平滑扩展到新的 宽度量 ,这条命令允许一台路由器同时接受和发送窄、及宽度量。

MPLS TE 配置需要使用宽度量值

全局配置命令 metric-narrow 可以恢复到最初的默认配置

4.6 SNP 序列号数据包

- Complete sequence number packets （CSNP）完全序列号数据包
- Partial sequence number packets （PSNP）部分序列号数据包

两种数据包具有相同的格式并且各自携带了 LSP 摘要信息的集合。区别在于 CSNP 携带了该路由器链路状态数据库中所有已知 LSP 的摘要信息，而 PSNP 携带的是是其中一个子集。为了区分 Level1、Level2 链路状态数据库，路由器产生 L1、L2 各自独立的 SNP 报文。

1. **CSNP**

- 对于第一层的 LSP 数据库 ,使用的是第 1 层的 CSNP 和 PSNP ,对于第 2 层的使用的则是第 2 层的 CSNP 和 PSNP.
- 描述了 LSP 数据库里所有的 LSP 信息。
- 用于两种情况：DIS 每隔 10s 周期性组播；链路刚刚建立起来时的点对点链里上。
- 如果 LSP 数据库过大，则要发送多个 csnp
- 在点对点链路上，一但毗邻关系建立起来了， 两个 IS 都会发送 CSNP 分组。 当发现有丢失的 LSP 时会用 PSNP 来请求。
- 在 LAN 中存在着 DIS ,DIS 生成并更新伪节点信息 ,并在 LAN 中扩散 LSP。DIS 每隔 10s 发送 CSNP，其中列出其链路状态数据库中保存的 LSP。这些 CSNP 通过组播发送到 LAN 上所有 IS-IS router。

```
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
  Intra Domain Routing Protocol Discriminator: ISIS (0x83)
  PDU Header Length  : 33
  Version (==1)      : 1
  System ID Length   : 0
  PDU Type           : L2 CSNP (R:000)
  Version2 (==1)     : 1
  Reserved (==0)     : 0
  Max.AREAs: (0==3)  : 0
⊟ ISO 10589 ISIS Complete Sequence Numbers Protocol Data Unit
    PDU length: 67
    Source-ID:    0000.0000.0003.00
    Start LSP-ID: 0000.0000.0000.00-00
    End LSP-ID: ffff.ffff.ffff.ff-ff
  ⊟ LSP entries (32)
    ⊞ LSP-ID: 0000.0000.0002.00-00, Sequence: 0x0000000a, Lifetime:  1198s, Checl
    ⊞ LSP-ID: 0000.0000.0003.00-00, Sequence: 0x00000008, Lifetime:  1196s, Checl
```

- Start LSP ID：起始 LSP ID。表示 TLV 字段中描述的 LSP 范围的第一个 LSP ID。
- End LSP ID：结束 LSP ID。表示 TLV 字段中描述的 LSP 范围的最后一个 LSP ID。

2. **PSNP**

主要是两个功能：

- 路由器使用 PSNP 在点对点链路上确认多个 LSP（中的下一个接收）
- 路由器使用 PSNP 来请求 LSP 的最新版本或者被丢失的 LSP。这点上同时适用于点对点和广播链路

在点到点链路上，**邻居用 PSNP 来确认新的 LSP。** 当源 router 收到了邻居发送过来的 PSNP 确认时，它会停止向这个邻居发送新的 LSP。在 LAN 中，没有明确的 PSNP 确认。丢失了的 LSP 可以被检测到。如果任何 LSP 丢失或过期了， router 都会**以 PSNP 的形式发送请求**来要求重传这些 LSP。

```
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
  Intra Domain Routing Protocol Discriminator: ISIS (0x83)
  PDU Header Length  : 17
  Version (==1)      : 1
  System ID Length   : 0
  PDU Type           : L2 PSNP (R:000)
  Version2 (==1)     : 1
  Reserved (==0)     : 0
  Max.AREAs: (0==3)  : 0
⊟ ISO 10589 ISIS Partial Sequence Numbers Protocol Data Unit
    PDU length: 35
    Source-ID: 0000.0000.0002.00
  ⊟ LSP entries (16)
    ⊞ LSP-ID: 0000.0000.0003.00-00, Sequence: 0x00000008, Lifetime:  1194s, Checks
```

4.7 LSP 洪泛扩散和同步

1. 关于泛洪

LSP 报文的“泛洪”指当一个路由器向相邻路由器报告自己的 LSP 后，相邻路由器再将同样的 LSP 报文传送到自己相邻的路由器，并这样逐级将 LSP 传送到整个层次内的一种方式。通过这种“泛洪”，整个层次内的每一个路由器就都可以拥有相同的 LSP 信息，并保持 LSDB 的同步。每一个 LSP 都拥有其自己的一个 4 字节的序列号。在路由器启动时所发送的第一个 LSP 报文中的序列号为 1 以后当需要生成新的 LSP 时，新 LSP 的序列号在前一个 LSP 序列号的基础上加 1。更高的序列号意味着更新的 LSP。

每个 LSP 在 LSDB 中都有一个最大经历时间（MaxAge），当这个时间到达后如果没有接收到新的 LSP 来更新 LSDB，则这个 LSP 会从 LSDB 中清除。在旧的 LSP 被从 LSDB 中清除后，它还会再保留一段时间（ZeroAgeLifetime，这期间只保留这个 LSP 的报文头），当这个时间也达到时它将会被真正删除。

一般缺省的 MaxAge 是 1200 秒，ZeroAgeLifetime 是 60 秒。当一个 IS 所发出的 LSP 的序列号达到 0XFFFFFFFF 时，这个路由器会将 IS-IS 进程暂停 MaxAge + ZeroAgeLifetime 秒，以便在整个路由域内和这个路由器对应的 LSP 都被删除掉。随后 IS-IS 进程再重新启动，并从序列号 1 开始发送 LSP。（这种情况其实不会出现，因为即使每秒钟产生一个 LSP，也要到 136 年后序列号才能从 1 达到 0XFFFFFFFF。）

2. LSP 产生的原因

IS-IS 路由域内的所有路由器都会产生 LSP，以下事件会触发一个新的 LSP：

- 邻接关系 Up 或 Down；
- IS-IS 相关接口 Up 或 Down；

- 引入的 IP 路由发生变化；
- 区域间的 IP 路由发生变化；
- 接口被赋了新的 metric 值；
- 周期性更新。

3. 路由选择信息扩散和数据库同步

- **SRM（Send Routing Message）：发送路由信息消息**
- **SSN（send Sequence Number）：发送序列号标志**

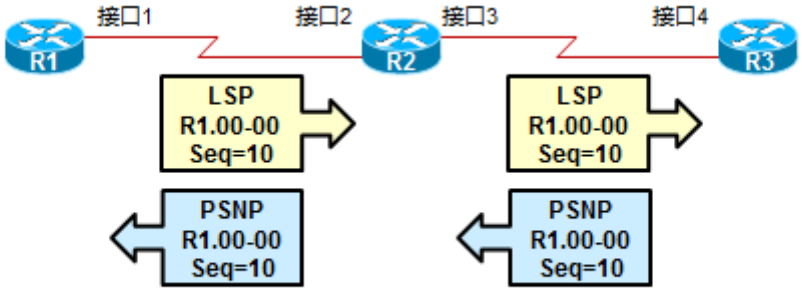
SRM：在更新过程中用来控制 LSP 传递到邻接路由器

SSN：主要用在如下两个方面：

- 在点对点链路上确保 LSP 可靠的扩散
- 广播链路上数据库同步时用于请求完整的 LSP 信息

SRM 和 SSM 用来协调扩散和数据库同步

4. 点对点链路中路由选择信息扩散



R2：
- R2从接口2收到R1发送的LSP R1.00-00，R2检查数据库，发现此LSP是自己没有的
- 在接口2为R1.00-00设置SSN
- 在接口3为R1.00-00设置SRM
- 在接口3扩散R1.00-00
- 在接口2为R1.00-00发送PSNP进行确认
- 在接口2清除R1.00-00的SSN设置
- 在接口3收到R1.00-00的PSNP确认
- 在接口3清除R1.00-00的SRM设置

R3：
- R3从接口4收到R2发送的LSP R1.00-00，R3检查数据库，发现此LSP是自己没有的
- 在接口4为R1.00-00设置SSN
- 在数据库中添加R1.00-00
- 在接口4为R1.00-00发送PSNP进行确认
- 在接口4清除R1.00-00的SSN设置

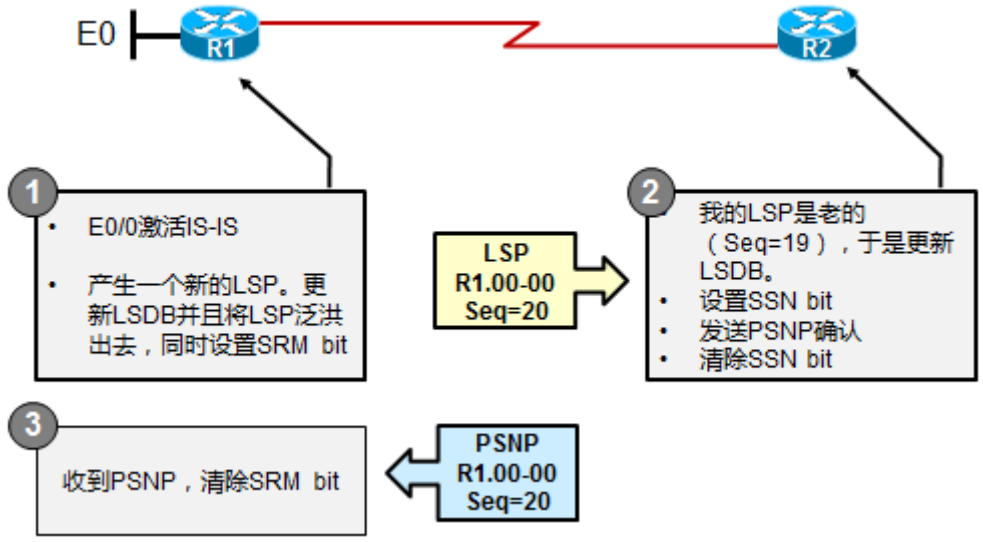
- ISs 将新的 LSP 泛洪给所有的邻居
  - LSP 发出后，SRM 被设置
  - 邻居要使用 PSNP 确认这个 LSP

如果 retransmission timout 后仍然没有收到 PSNP 确认，则始发路由器将重传这个 LSP

- 在收到关于这个 LSP 的 PSNP 确认后，SRM 被清除

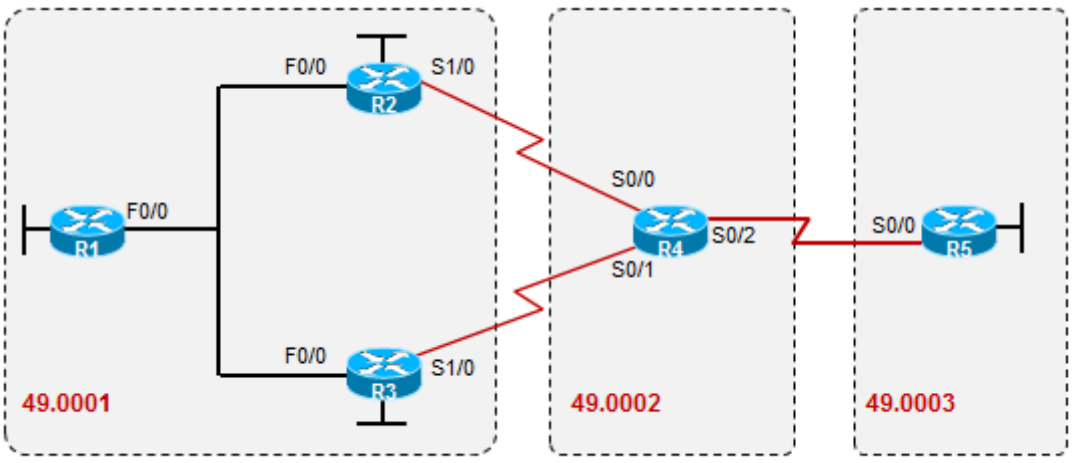
- 在邻接关系建立过程中，邻接双方会互相发送 CSNP（邻接关系建立完成后不再发送 CSNP）
  - 如果收到 CSNP 且发现其中丢失了 LSPs，则路由器会对该丢失的 LSP 进行重传
  - 丢失的 LSPs 可以通过 PSNP 进行请求

再来看一个例子：



5 集成 IS-IS 的配置

5.1 基础实验



R1 的配置如下：

```
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
 ip router isis                          !! 在接口上激活 IS-IS
interface FastEthernet0/0
 ip address 10.1.123.1 255.255.255.0
 ip router isis
router isis
 net 49.0001.0000.0000.0001.00           !! 为设备配置 net 地址，可以配置多个，默认最多 3 个
 is-type level-1                         !! 配置设备的 is-type，默认为 level1-2
```

R2 的配置如下：

```
interface Loopback0
 ip address 2.2.2.2 255.255.255.0
 ip router isis
interface FastEthernet0/0
 ip address 10.1.123.2 255.255.255.0
 ip router isis
interface Serial1/0
 ip address 10.1.24.2 255.255.255.0
 ip router isis
router isis
 net 49.0001.0000.0000.0002.00
```

R3 的配置如下：

```
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
 ip router isis
interface FastEthernet0/0
 ip address 10.1.123.3 255.255.255.0
 ip router isis
interface Serial1/0
 ip address 10.1.34.3 255.255.255.0
 ip router isis
router isis
 net 49.0001.0000.0000.0003.00
```

R4 的配置如下：

```
interface Serial0/0
 ip address 10.1.24.4 255.255.255.0
 ip router isis
interface Serial0/1
 ip address 10.1.34.4 255.255.255.0
 ip router isis
interface Serial0/2
 ip address 10.1.45.4 255.255.255.0
 ip router isis
router isis
 net 49.0002.0000.0000.0004.00
 is-type level-2-only                    !! R4 为 L2 router，因此修改 is-type 类型为 level2-only
```

R5 的配置如下：

```
interface Loopback0
 ip address 5.5.5.5 255.255.255.0
 ip router isis
interface Serial0/0
 ip address 10.1.45.5 255.255.255.0
 ip router isis
router isis
 net 49.0003.0000.0000.0005.00
 is-type level-2-only
```

接下去就 show 一下：

R1#show ip route

```
      1.0.0.0/24 is subnetted, 1 subnets
C        1.1.1.0 is directly connected, Loopback0
      2.0.0.0/24 is subnetted, 1 subnets
i  L1    2.2.2.0 [115/20] via 10.1.123.2, FastEthernet0/0
      3.0.0.0/24 is subnetted, 1 subnets
i  L1    3.3.3.0 [115/20] via 10.1.123.3, FastEthernet0/0
      10.0.0.0/24 is subnetted, 3 subnets
i  L1    10.1.24.0 [115/20] via 10.1.123.2, FastEthernet0/0
```

```
i      L1        10.1.34.0 [115/20] via 10.1.123.3, FastEthernet0/0
C                10.1.123.0 is directly connected, FastEthernet0/0
i*     L1    0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
                        [115/10] via 10.1.123.2, FastEthernet0/0
```

我们看到 R1 已经学习到了 2.2.2.0、3.3.3.3.0 以及 10.1.24.0、10.1.34.0。

R2 及 R3 的 loopback 接口来自本区域，因此为 L1 的路由这个没有问题，但是为啥 R2-R4、R3-R4 直连链路也进来了呢？那是因为这些是 R2 和 R3 的直连接口，可以通过在 R2 的 s1/0 及 R3 的 S1/0 口上，将接口配置为 level2，使用接口级别命令：isis circuit-type level2，那么这样一来 R2 及 R3 在区域 49.0001 里通告 LSP 的时候，就不会通告这些这直连链路了。因此 R1 的路由表变成：

R1#show ip route

```
           1.0.0.0/24 is subnetted, 1 subnets
C             1.1.1.0 is directly connected, Loopback0
           2.0.0.0/24 is subnetted, 1 subnets
i      L1        2.2.2.0 [115/20] via 10.1.123.2, FastEthernet0/0
           3.0.0.0/24 is subnetted, 1 subnets
i      L1        3.3.3.0 [115/20] via 10.1.123.3, FastEthernet0/0
           10.0.0.0/24 is subnetted, 3 subnets
C             10.1.123.0 is directly connected, FastEthernet0/0
i*     L1    0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
                        [115/10] via 10.1.123.2, FastEthernet0/0
```

接下来再看看 R1 的 isis database：

R1#show isis database

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x0000000B   0xA538        687               0/0/0
R2.00-00              0x0000000F   0x7056        906               1/0/0
R3.00-00              0x0000000E   0x3B83        1096              1/0/0
R3.02-00              0x0000000C   0x4F65        880               0/0/0
```

在区域 49.0001 我们可以看到所有路由器都泛洪了自己的 LSP，LSP 使用 LSPID 来表示，例如 R1.00-00，这个 R1 就是设备的 hostname，这里我们有默认的 hostname 映射机制，请看上文。另外 R1 后面的 00 是伪节点标示符，为 00 则表示该 LSP 的始发路由器不是 DIS，再后头的 00 则是分片标记。

我们看到上面的输出，R1、R2、R3 都产生了 LSP，打星号的条目为 R1 自己发出的。另外，还有个特殊的 LSP：R3.02-00，这里伪节点标记为非 0，则表示 R3 是一台 DIS，而这个 LSP，是一个伪节点的 LSP。

接下去我们看看这些 LSP 的明细：

R1#show isis da R1.00-00 detail

```
IS-IS Level-1 LSP R1.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x0000000C   0xA339        1174              0/0/0
  Area Address: 49.0001
  NLPID:        0xCC
  Hostname: R1
  IP Address:   1.1.1.1
  Metric: 10         IP 10.1.123.0 255.255.255.0
  Metric: 10         IP 1.1.1.0 255.255.255.0
  Metric: 10         IS R3.02
```

可以看到其中包含的主要两个内容：一是 R1 的直连网段，这里是 10.1.123.0 及 1.1.1.0，另一是 R1 的直连 IS，由于这是一个 LAN 的环境，因此会选举 DIS，并且每一台非 DIS 路由器通告自己与 DIS 的邻接关系，而不通告与其他物理 IS 的邻接关系。R3\R3 的 LSP 类似，这里就不再赘述。看看这个伪节点的 LSP 吧：

R1#show isis da R3.02-00 detail

```
IS-IS Level-1 LSP R3.02-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R3.02-00              0x0000000C   0x4F65        539               0/0/0
  Metric: 0          IS R3.00
  Metric: 0          IS R2.00
  Metric: 0          IS R1.00
```

我们看到伪节点 LSP 主要的内容就是连接在这个伪节点上的 IS。

再查看一下 isis 的邻居：

R1#sh isis neighbors detail

```
System Id      Type     nterface    P Address       State   oldtime  Circuit Id
R2             L1       a0/0        0.1.123.2       UP      3        R3.02
  Area Address(es): 49.0001
  SNPA: cc02.1ab0.0000              !! 邻居的 SNPA，在 LAN 中为对方的接口 MAC
  State Changed: 00:20:27
  LAN Priority: 64                  !! 用于选举 DIS 的接口优先级
  Format: Phase V
R3             L1       Fa0/0       10.1.123.3      UP      9        R3.02
```

```
  Area Address(es): 49.0001
  SNPA: cc03.1ab0.0000
  State Changed: 02:23:31
  LAN Priority: 64
  Format: Phase V
```

再看一下 R4 的 isis database：

R4#sh isis da

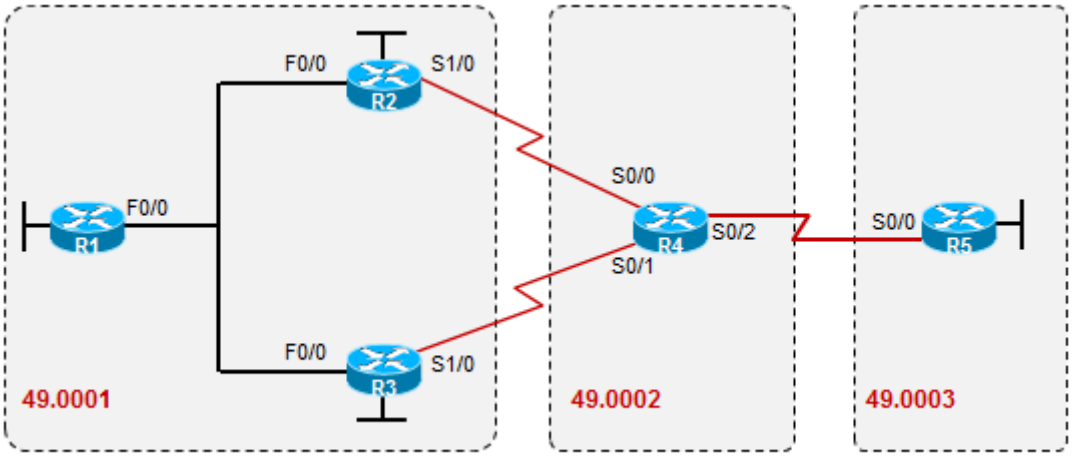
IS-IS Level-2 Link State Database:

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
R2.00-00	0x00000011	0x2B7C	558	0/0/0
R3.00-00	0x00000011	0x1B7C	562	0/0/0
R3.02-00	0x0000000C	0x10BA	908	0/0/0
R4.00-00	* 0x0000000F	0xD3FB	569	0/0/0
R5.00-00	0x0000000D	0x26DC	1144	0/0/0

R4 的路由表：

```
      1.0.0.0/24 is subnetted, 1 subnets
i L2     1.1.1.0 [115/30] via 10.1.34.3, Serial0/1
                 [115/30] via 10.1.24.2, Serial0/0
      2.0.0.0/24 is subnetted, 1 subnets
i L2     2.2.2.0 [115/20] via 10.1.24.2, Serial0/0
      3.0.0.0/24 is subnetted, 1 subnets
i L2     3.3.3.0 [115/20] via 10.1.34.3, Serial0/1
      5.0.0.0/24 is subnetted, 1 subnets
i L2     5.5.5.0 [115/20] via 10.1.45.5, Serial0/2
      10.0.0.0/24 is subnetted, 4 subnets
C        10.1.24.0 is directly connected, Serial0/0
C        10.1.45.0 is directly connected, Serial0/2
C        10.1.34.0 is directly connected, Serial0/1
i L2     10.1.123.0 [115/20] via 10.1.34.3, Serial0/1
                    [115/20] via 10.1.24.2, Serial0/0
```

5.2 Attached-bit



在完成基本配置后，由于 R2、R3 为 L1\L2 router，因此对于他两来说，都会向本地的 level1 area 通告自己为出去的一个口子，方法就是在自己产生的 LSP 中，ATT 置位，我们在 R1 上看 isis database 已经能看出来了。

R1#show isis database

IS-IS Level-1 Link State Database:

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
R1.00-00	* 0x0000000B	0xA538	687	0/0/0
R2.00-00	0x0000000F	0x7056	906	**1/0/0**
R3.00-00	0x0000000E	0x3B83	1096	**1/0/0**
R3.02-00	0x0000000C	0x4F65	880	0/0/0

这将直接导致 R1 会从 R2、R3 中选择出最近的一个出口，并且生成一条默认路由指向这个出口，由于这个实验中，到达 R2、R3 等代价，因此 R1 的路由表中，有两条默认路由负载均衡。这种默认的、自动的机制对于 level1 area 来说是福利，因为他可以减小路由表，同时又提供了一个出去的口子，但是这也带来一个问题，例如，如果 R2 或者 R3 连接 backbone 的链路 DOWN 掉了，那么 R1 是否会傻乎乎的仍然将数据包丢给 R2 或 R3 呢？庆幸的是，在新的 CISCO IOS 中，对 IS-IS 做了优化：在正常情况下，R2、R3 都会对在 49.0001 区域中泛洪的 LSP 进行 ATT 置位，拿 R2 举例，如果 R2 的 S1/0 口 DOWN 掉了，那么它就丢失了 Backbone 的连接，于是 R2 立即触发一个新的 LSP 并且 ATT 没有置位，如此一来，R1 将原先指向 R2 的默认路由撤销。

当然，上面的方法不保险。所以，在特殊的情况下，我们可能希望 L1\L2 router 有条件性的去设置这个 ATT 位、更加可控的去设置，那么可以关联 route-map，并且在 route-map 中去 match 某条特定的路由，只要这条路由在我路由表里，route-map 才满足，我才会设置 ATT。

例如上图，我们可以在 R2 上做测试，配置如下：

```
ip prefix-list test seq 5 permit 5.5.5.0/24
route-map test permit 10
```

```
 match ip address prefix-list test
router isis
 net 49.0001.0000.0000.0002.00
 set-attached-bit route-map test
```

这样一来，只要 5.5.5.0 在 R2 的路由表中，那么 R2 就会在其向区域 49.0001 中泛洪的 LSP 中进行 ATT 置位，我们可以尝试将 R5 的 loopback shutdown，这样 5.5.5.0 就挂了，当 R2 检测到这个变化，发现路由表中没了 5.5.5.0/24 的路由，那么 R2 将触发一个 LSP 更新，这个 LSP 中 ATT 就没有置位了，这样一来 R1 就只会产生一条指向 R3 的默认路由。当 5.5.5.0 恢复了，那么 R2 又会产生一个 LSP，又将 ATT 置位（这条 5.5.5.0 的路由未必必须是 ISIS 路由，你完全可以耍耍，配一条指向 null0 的路由，丫就 ATT 置位了）。这里有一点要注意的是：用于匹配路由的工具建议采用前缀列表 ,我在 12.4 的 IOS 上测试过用标准的 ACL 去匹配路由 ,结果貌似不成功；

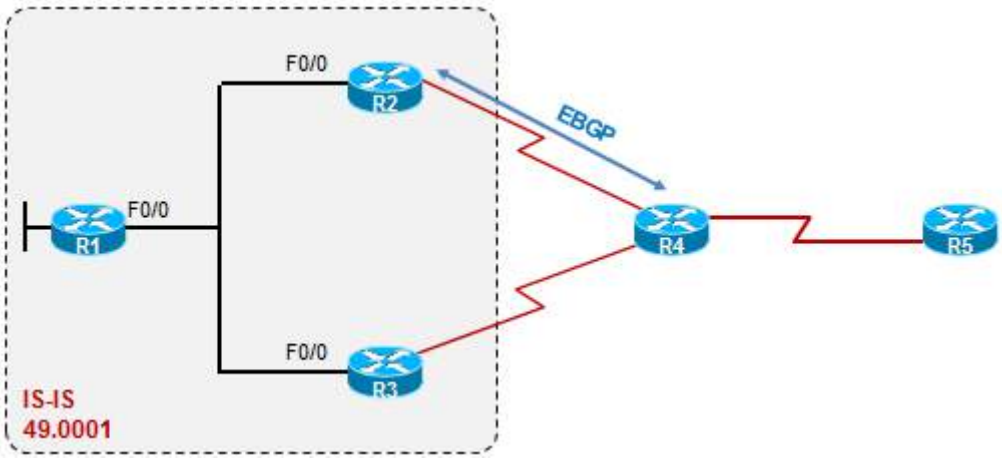
另一点要注意的是，只有 L1/L2 router 在保有 L2 连接的情况下，才会自动将 ATT 置位，如果上图中，R2 的 S1/0 口挂掉了，那么 R2 将没有任何 Level2 的邻接关系，也就是说，R2 与 backbone 或者 level2 area 断开了，R2 将立刻触发一个新的 LSP，并且将 ATT 清除，这就导致 R1 不在使用 R2 作为默认路由的下一跳。

不希望 L1-router 产生这条默认路由，或者更严谨点说，我们不希望 L1\L2 router 压 att 位，方法就是在 L1/L2router 上用 set attached-bit route-map 关联一个 route-map test deny 的 route-map，就可以了。

5.3 Overload-bit

一台路由器如果资源不足，它可能就无法维护完整的链路状态数据库，数据流量到达此台设备可能就无法得到正常的转发，因此它有告知或者告警其他路由器的能力，具体的做法是在其自己产生的 LSP 中将 Overload-bit 置位。被设置了 overload 字段的那些 LSP 不会在网络中扩散并且在计算通过 overload 路由器的路由时也不会采用这些 LSP。换句话说，overload 路由器被传输路绕开（Bypass）了，那些以 overload 路由器作为最后一跳的路由（本地直连的网段）才参与计算。再说白点，就是接收这个 LSP 的 IS-IS 路由器，仍然可以产生到达 overload 路由器直连的路由，但是不会通过这台路由器去往其他的目的地。

单纯的由于设备资源不足而去设置 overload 的动机现如今已经不多了，毕竟现在设备的性能越来越好，而且 overload 这个特性毕竟是在设备性能不够强劲的时代的产物。然而 overload 的扩展应用却是非常值得关注的。我们来看一下下面这个图：



假设这是一个运营商内部的网络，R4 是国干，R2、R3 往左是省干，省干内部跑了 IS-IS 用于承载 Core 内的路由前缀信息，并且作为省干出口 R2、R3 都下发了特定的国干的路由前缀或缺省路由，我们这里拿缺省路由举例。R2、R3，都与 R4 维护 EBGP 邻居关系，那么这时候假设 R2 要做设备升级，然后重启，重启之后由 IS-IS 的收敛速度比 BGP 要快，率先收敛完毕后，R2 向 IS-IS 区域内泛洪 LSP，R1 将使用 R2 作为某些路由前缀的下一跳，或默认路由的下一跳，然而这时候 R2 的 BGP 并未收敛完毕，这样一来，到达 R2 的数据就有可能被丢弃，黑洞就出现了。

我们可以借助 overload-bit 来避免这个问题，命令的解释如下（配置在 router isis 进程下）：

Set-overload-bit

本路由器产生的 LSP overload-bit 置位。这将导致这台路由器将不会被其他路由器作为去往任何目的地的下一跳，除了其本地直连的网段。这一般用在路由器是叶节点的时候，也就是说是网络的最末梢，下面直接连的就是用户。

Set-overload-bit on-startup <5-86400 sencondes>

后面跟一个秒数，用来说明设备重启需要设置 overload-bit 的时间，超过该时间后，overload-bit 被清除

set-overload-bit on-startup wait-for-bgp

只有等到 BGP 收敛完成后，overload-bit 才会被清除，这个特性不是特别的推荐。感觉不太靠谱。

set-overload-bit suppress external

不通告从其他路由协议重发布进来的路由

set-overload-bit suppress interlevel

不通告从其他 level 泄露进来的路由

1. 测试 1：set-overload-bit

在 R2 上配置如下：

```
Router isis
set-overload-bit
```

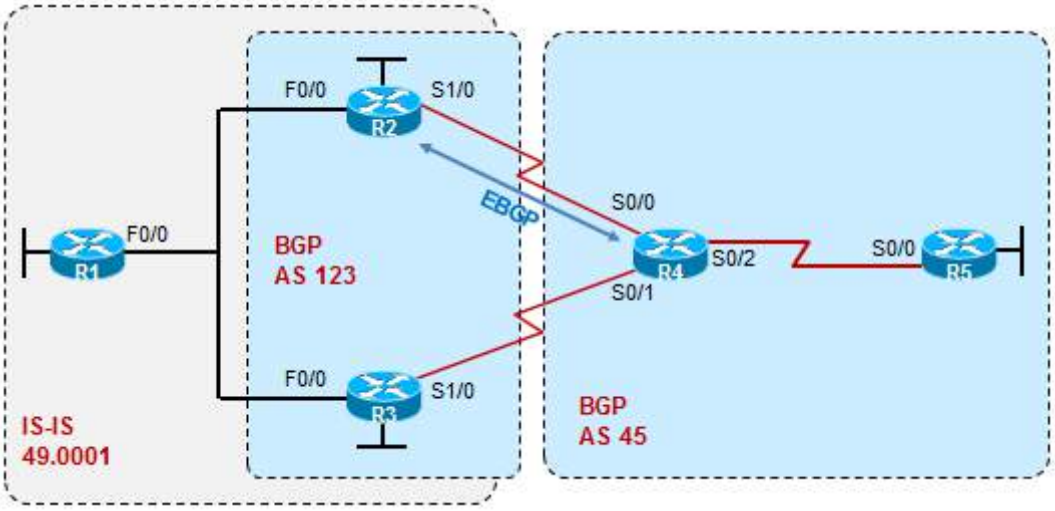
这将导致 R2 产生的 LSP 的 OL 位置位，在 R1 上看到的结果是这样的：

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x00000013   0x9540        724               0/0/0
R2.00-00              0x00000017   0x5C66        1178              0/0/1
R3.00-00              0x00000014   0x2F89        668               1/0/0
R3.02-00              0x00000010   0x4769        722               0/0/0
```

我们发现，R2 产生的 LSP 之前的 attached-bit 没了，但是设置了 Overload-bit。现在我们再去看路由表，R1 再计算最短路径的时候，就不会考虑通过 R2 出区域。于是 R1 上仅剩指向 R3 的默认路由。当然，R2 直连的路由还是能在 R1 上看到的。

2. 测试 2：set-overload-bit on-startup wait-for-bgp

我们要变更一下拓扑：



在 R2 上配置如下：

```
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit on-startup wait-for-bgp
router bgp 123
 no synchronization
 neighbor 10.1.24.4 remote-as 45
 no auto-summary
```

保存配置后重启 R2，在 R2 重启完成后，isis 率先收敛，然而这个时候 bgp 暂未收敛完成，因此 R2 产生的 LSP 中 overload 置位，从 R1 上能看出结果如下，在此期间，R1 不会将 R2 作为去往其他网络的下一跳（R2 直连的网络除外）：

R1 上 show isis database

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x0000002B   0x2995        1112              0/0/0
R2.00-00              0x00000036   0xE7D9        1178              0/0/1
R3.00-00              0x00000029   0x18B7        968               0/0/0
R3.01-00              0x00000026   0x2D6E        1019              0/0/0
```

随后我们等待 R2 上 BGP 的邻居关系建立，可以 debug 一下：

R2 上 debug isis update

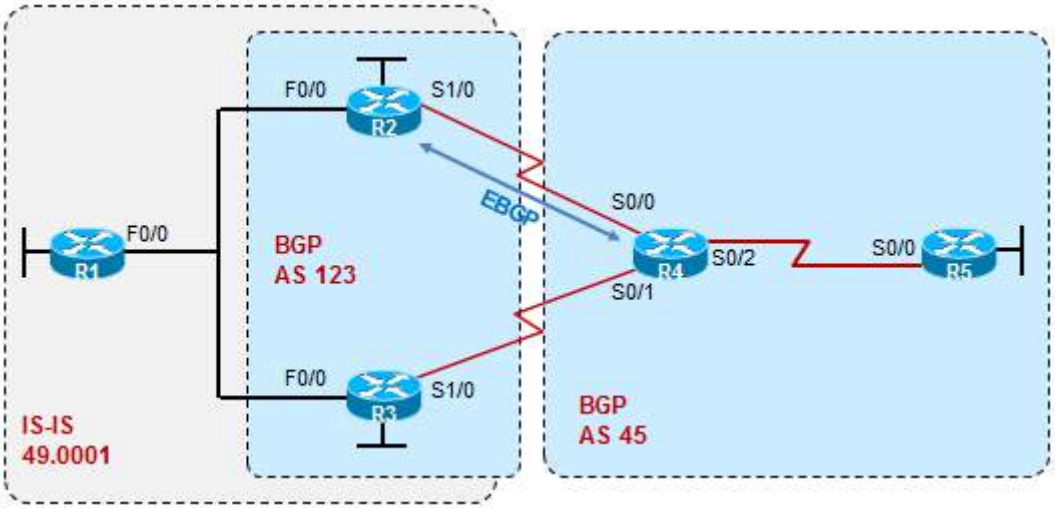
```
*Mar  1 00:00:32.639: %BGP-5-ADJCHANGE: neighbor 10.1.24.4 Up
R2#sh run |
*Mar  1 00:00:32.755: ISIS-Upd: Building L1 LSP
*Mar  1 00:00:32.759: ISIS-Upd: Important fields changed
*Mar  1 00:00:32.759: ISIS-Upd:  full SPF required
*Mar  1 00:00:32.759: ISIS-Upd: Building L2 LSP
*Mar  1 00:00:32.763: ISIS-Upd: Important fields changed
*Mar  1 00:00:32.763: ISIS-Upd:  full SPF required
*Mar  1 00:00:32.783: ISIS-Upd: Sending L1 LSP 0000.0000.0002.00-00, seq 37, ht 1199 on FastEthernet0/0 via high priority queue
```

我们看到，BGP 邻居关系 up 后，R2 马上触发一个 LSP，这个 LSP 清除之前的 overload 置位。

最终 R1 上 show isis database

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x0000002B   0x2995        1103              0/0/0
R2.00-00              0x00000037   0xE1E2        1194              0/0/0
R3.00-00              0x00000029   0x18B7        959               0/0/0
R3.01-00              0x00000026   0x2D6E        1010              0/0/0
```

3. 测试 3：set-overload-bit on-startup 120

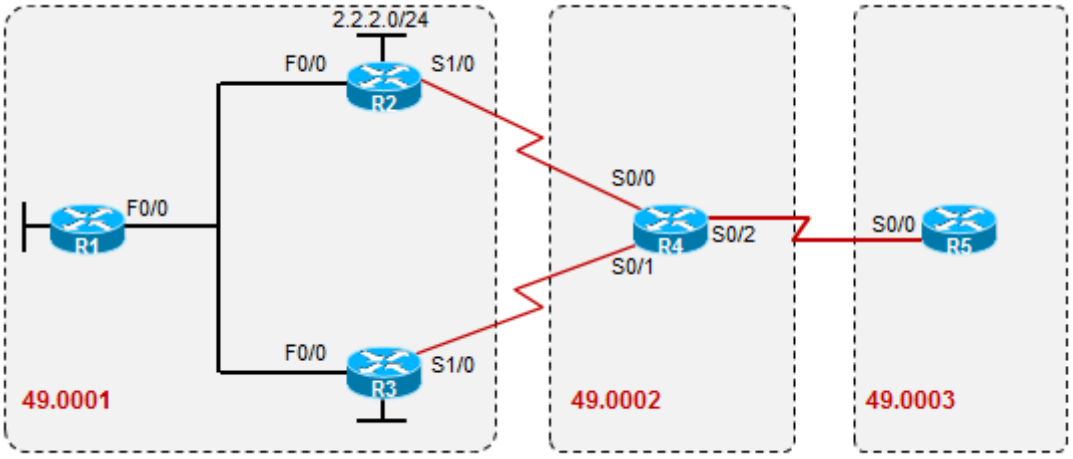


还是这个图，不过 R2 的配置变成：

```
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit on-startup 120            !! 120s
router bgp 123
 no synchronization
 neighbor 10.1.24.4 remote-as 45
 no auto-summary
```

保存配置后重启，在重启后，R2 产生的 LSP 先是设置了 overload-bit 的，这点从 R1 上 isis databse 就能看到，我们在 R2 上同样开启 debug isis update，同时使用 show version | in uptime 查看路由器的启动时间，当启动时间到达 120s 后，从 debug 信息中能看到 R2 触发了一个 LSP，并且清除了 overload-bit。

4. 测试 4：set-overload-bit suppress external



- **Point1：**

R2 的配置如下：

```
route-map test permit 10
router isis
 net 49.0001.0000.0000.0002.00
 redistribute connected level-1                    !!将本地直连的 loopback 2.2.2.0/24 重发布到 level1
 redistribute isis ip level-2 into level-1 route-map test
```

R1 的路由表如下：

```
       1.0.0.0/24 is subnetted, 1 subnets
C         1.1.1.0 is directly connected, Loopback0
       2.0.0.0/24 is subnetted, 1 subnets
i L1      2.2.2.0 [115/10] via 10.1.123.2, FastEthernet0/0
       5.0.0.0/24 is subnetted, 1 subnets
i ia      5.5.5.0 [115/158] via 10.1.123.2, FastEthernet0/0
       10.0.0.0/24 is subnetted, 4 subnets
i ia      10.1.24.0 [115/148] via 10.1.123.2, FastEthernet0/0
i ia      10.1.45.0 [115/158] via 10.1.123.2, FastEthernet0/0
i ia      10.1.34.0 [115/158] via 10.1.123.2, FastEthernet0/0
C         10.1.123.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
                        [115/10] via 10.1.123.2, FastEthernet0/0
```

现在是初始环境，R1 能学习到 inter-area 路由，以及 R2 本地重发布到 level1 的路由 2.2.2.0。当然由于 R2 设置了 attached-bit，因此 R1 也会产生一条指向 R2 的默认路由。

- **Point2**

R2 的配置修改如下：

```
route-map test permit 10
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit                          !! 设置 overload-bit
 redistribute connected level-1                    !!将本地直连的 loopback 2.2.2.0/24 重发布到 level1
 redistribute isis ip level-2 into level-1 route-map test
```

再来看看 R1 的路由表：

```
       1.0.0.0/24 is subnetted, 1 subnets
C         1.1.1.0 is directly connected, Loopback0
```

```
      2.0.0.0/24 is subnetted, 1 subnets
i L1     2.2.2.0 [115/10] via 10.1.123.2, FastEthernet0/0
      5.0.0.0/24 is subnetted, 1 subnets
i ia     5.5.5.0 [115/158] via 10.1.123.2, FastEthernet0/0
      10.0.0.0/24 is subnetted, 4 subnets
i ia     10.1.24.0 [115/148] via 10.1.123.2, FastEthernet0/0
i ia     10.1.45.0 [115/158] via 10.1.123.2, FastEthernet0/0
i ia     10.1.34.0 [115/158] via 10.1.123.2, FastEthernet0/0
C        10.1.123.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
```

我们看到 R1 这些 inter-area 路由都还在，但是 R1 不再指默认路由到 R2。

注意，这个时候 R2 产生的 LSP 已经设置了 overload-bit 了,在 R1 上 show isis database 能看到

- **Point3**

在上述配置的基础上，修改 R2 的配置：

```
route-map test permit 10
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit suppress external          !! 设置 overload-bit，抑制外部路由
 redistribute connected level-1              !!将本地直连的 loopback 2.2.2.0/24 重发布到 level1
 redistribute isis ip level-2 into level-1 route-map test
```

再看看 R1 的路由表：

```
      1.0.0.0/24 is subnetted, 1 subnets
C        1.1.1.0 is directly connected, Loopback0
      5.0.0.0/24 is subnetted, 1 subnets
i ia     5.5.5.0 [115/158] via 10.1.123.2, FastEthernet0/0
      10.0.0.0/24 is subnetted, 4 subnets
i ia     10.1.24.0 [115/148] via 10.1.123.2, FastEthernet0/0
i ia     10.1.45.0 [115/158] via 10.1.123.2, FastEthernet0/0
i ia     10.1.34.0 [115/158] via 10.1.123.2, FastEthernet0/0
C        10.1.123.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
```

R2 重发布的直连路由 2.2.2.0 没了，inter-area 路由都还在并且，R1 不再使用 R2 作为默认路由的下一跳，注意，这时候 R2 发的 LSP 也是 overload-bit 置位了。

注意，此时此刻，R4 上也不会学习到 2.2.2.0/24 了，这条外部路由在 R2 上被彻底抑制住了。

5. 测试 5：set-overload-bit suppress interlevel

修改 R2 的配置：

```
route-map test permit 10
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit suppress interlevel          !! 设置 overload-bit，抑制区域间路由
 redistribute connected level-1                !! 将本地直连的 loopback 2.2.2.0/24 重发布到 level1
 redistribute isis ip level-2 into level-1 route-map test
```

看一下 R1 的路由表：

```
      1.0.0.0/24 is subnetted, 1 subnets
C        1.1.1.0 is directly connected, Loopback0
      2.0.0.0/24 is subnetted, 1 subnets
i L1     2.2.2.0 [115/10] via 10.1.123.2, FastEthernet0/0
      10.0.0.0/24 is subnetted, 2 subnets
i ia     10.1.24.0 [115/148] via 10.1.123.2, FastEthernet0/0
C        10.1.123.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
```

R2 重发布的 2.2.2.0 外部路由还在；inter-area 路由都没了，除了一条 R2 本地直连的 24.0；

另外，R1 仍然不会采用 R2 作为默认路由的下一跳。

还是那句话，只要使用了 set-overload-bit，R2 发出的 LSP 的 overload-bit 就会置位

6. 测试 6：set-overload-bit suppress interlevel external

修改 R2 的配置：

```
route-map test permit 10
router isis
 net 49.0001.0000.0000.0002.00
 set-overload-bit suppress interlevel external
 redistribute connected level-1                !! 将本地直连的 loopback 2.2.2.0/24 重发布到 level1
 redistribute isis ip level-2 into level-1 route-map test
```

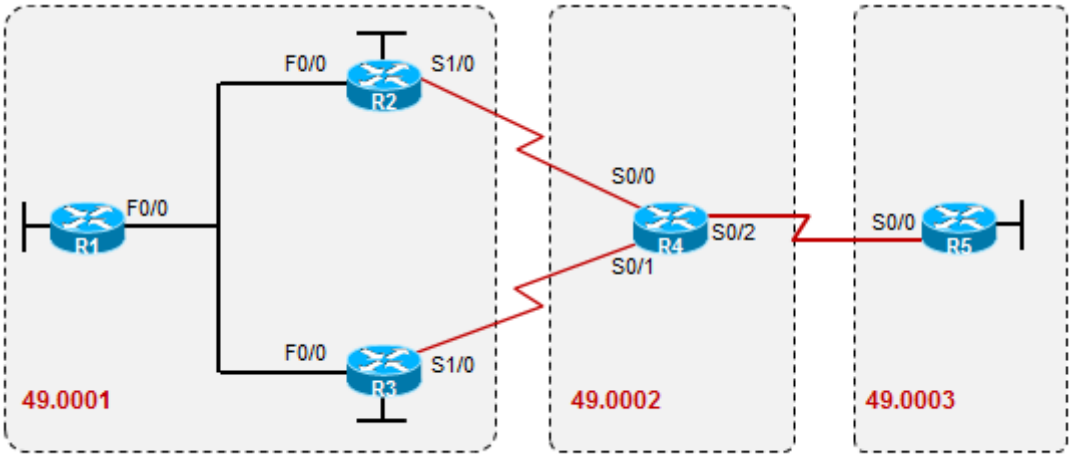
看一下 R1 的路由表：

```
      1.0.0.0/24 is subnetted, 1 subnets
```

```
C        1.1.1.0 is directly connected, Loopback0
      10.0.0.0/24 is subnetted, 2 subnets
i ia     10.1.24.0 [115/148] via 10.1.123.2, FastEthernet0/0
C        10.1.123.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.123.3, FastEthernet0/0
```

我们发现，R2 本地重发布的外部路由没了；R2 从 level2 注入 level1 的 inter-area 路由也没了；同时 R2 也不能作为 R1 默认路由的下一跳。只有一条 R2 本地直连的 24.0 路由。

5.4 路由泄露



我们知道默认的情况下，IS-IS 的 level1 area 很像一个 stub（但是人家允许重发布外部路由进来，所以只是说像），area 内的 L1 router 通过一条默认路由来到达区域外，这就有可能产生次优路径，因为他找的是本区域最近的 L1\L2 router。那么，为了解决这个问题，我们可以采用路由泄露的方法，例如在 R2 上配置：

```
access-list 1 permit 5.5.5.0
route-map test permit 10
  match ip add 1
router isis
redistribute isis ip level-2 into level-1 route-map test
```

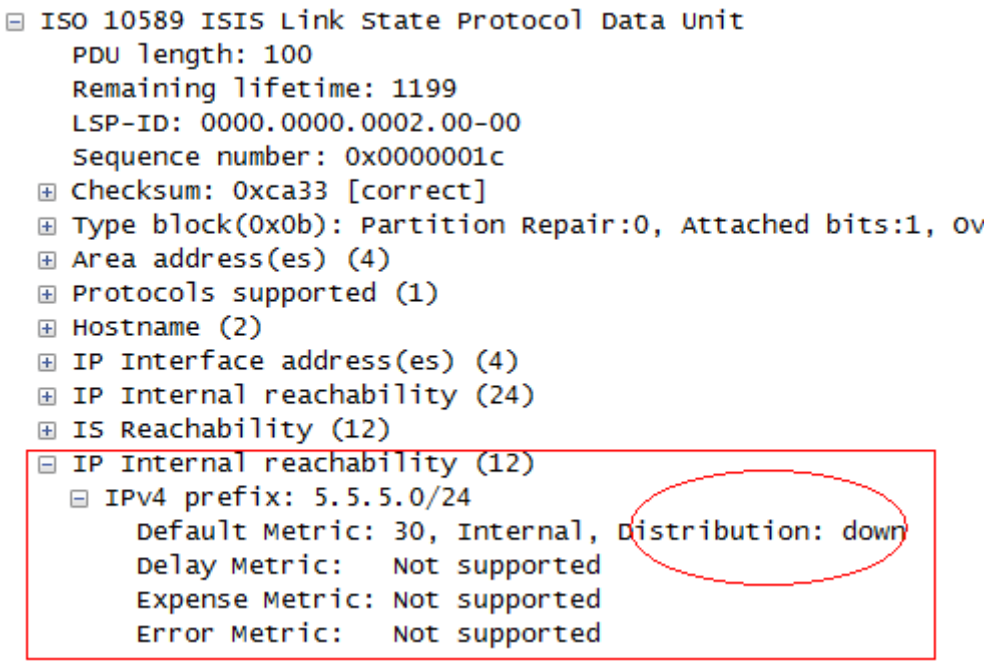
如此一来，L2 的路由中，被 route-map 匹配的路由，就会被注入到 level1 area 中，那么 R1 就能学习到 5.5.5.0。类型为 ia，也就是 isis 的 interarea 路由，即 isis 区域间路由。

R1#show ip route 5.5.5.0

```
Routing entry for 5.5.5.0/24
  Known via "isis", distance 115, metric 168, type inter area
```

```
Redistributing via isis
Last update from 10.1.123.2 on FastEthernet0/0, 00:01:32 ago
Routing Descriptor Blocks:
* 10.1.123.2, from 2.2.2.2, via FastEthernet0/0
    Route metric is 168, traffic share count is 1
```

同时为了避免环路，R2 在向 level1 area 注入这些路由的时候，会打上 downbit，这样一来，R3 收到这条 LSP，针对这些路由，就不会再泛洪到 backbone 里。



5.5 查看及验证

```
log-adjacency-changes
```

强烈建议开启此命令，可以查看到 isis 邻接关系的一些 log，默认是关闭的

- show clns neighbors
- show clns int
- show clns protocol
- show isis database
- show ip route isis
- show isis spf-log
- show isis lsp-log

5.6 多区域集成 IS-IS 的配置

1. 缺省地，Cisco IOS 在 IS-IS router 上同时启动 L1 和 L2 的操作，如果要指定 router 仅仅作为区域 router 或主干 router，则在路由器配置模式下用：

   is-type 命令 如 is-type level-1

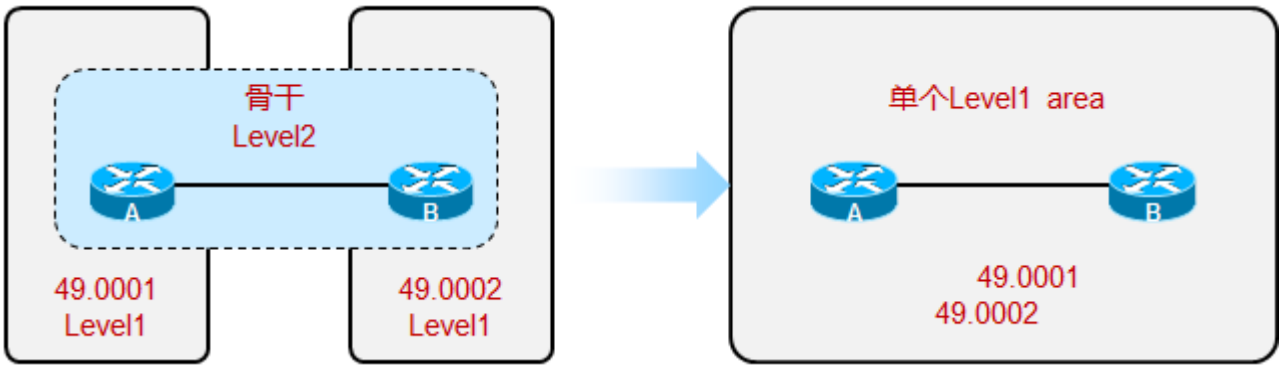
2. 尽管 L1/L2 router 同时具有 L1 和 L2 的功能，但有时在特定的 接口上只需建立第 1 层毗邻关系，而在其他接口上配置第 2 层毗邻关系。

   在接口配置模式下： isis circuit-type 可以让接口工作在 level-1 level-1-2 或 level-2-only

3. 通过接口配置命令： isis metric *value* **level-1 | level-2** 来配置 metric 在同一接口上，也可以分别为第 1 层和第 2 层设定不同的度量值。

5.7 为单个 IS-IS 进程配置多个 NET

一般来讲，一个 IS-IS 进程，仅需要一个 NET，当然你也可以为一个 IS-IS 进程分配多个 areaID 不同的 NET 地址，注意这些 NET 地址的 SystemID 必须相同。这样一来这台路由器可以连接到多个区域，在我们需要把不同区域合并为一个区域时，这个方法很不错。一般来说，一台 Level1 路由器仅仅在其连接到的区域内参与扩散 L1 LSP，如果这台路由器连接了多个区域，那么可以实现 L1 LSP 在多个区域的泛洪，从而完成区域的有效合并。



看上面这个例子：A 在区域 49.0001，B 在 49.0002，由于他们在不同的区域，因此彼此成了 Level2 邻接关系。同时维护本区域内的邻接关系。如果 A 配置上：

```
router isis
  net 49.0001.0000.0000.0001.00
  net 49.0002.0000.0000.0001.00
```

那么 A-B 之间就形成了 Leve1 邻接，因为此刻它两同属一个区域。A 继续维护与区域 49.0001 的 Level1 邻接，B 也继续维护与区域 49.0002 的 Level1 邻接，由于在 AB 之间形成了新的 Level1 邻接，设备开始叫唤最初他们互相隔离的 Level1 数据库，并扩散到各自的 Level1 邻居，这样就实现了区域的合并。

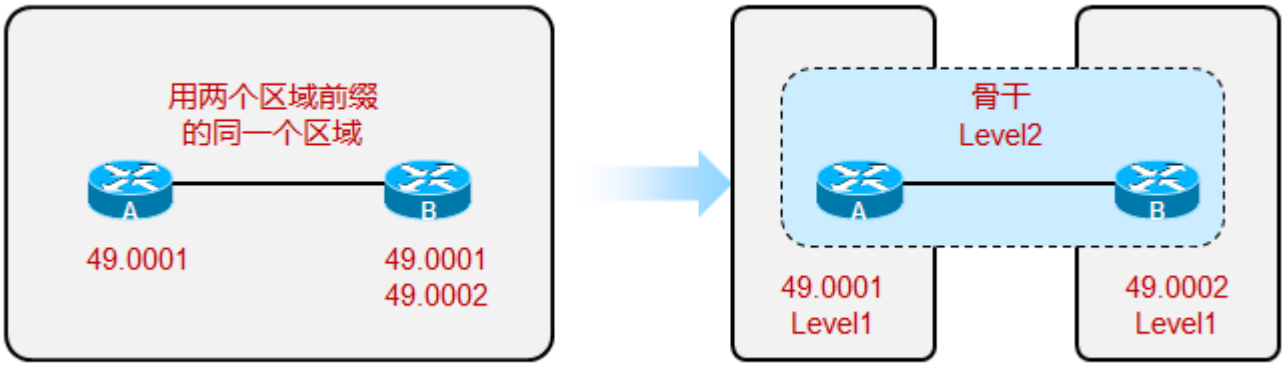
所以利用这个特性（一个 IS-IS 进程配置多个 NET 地址），可以实现：

- 区域合并
- 区域分离
- 重编址

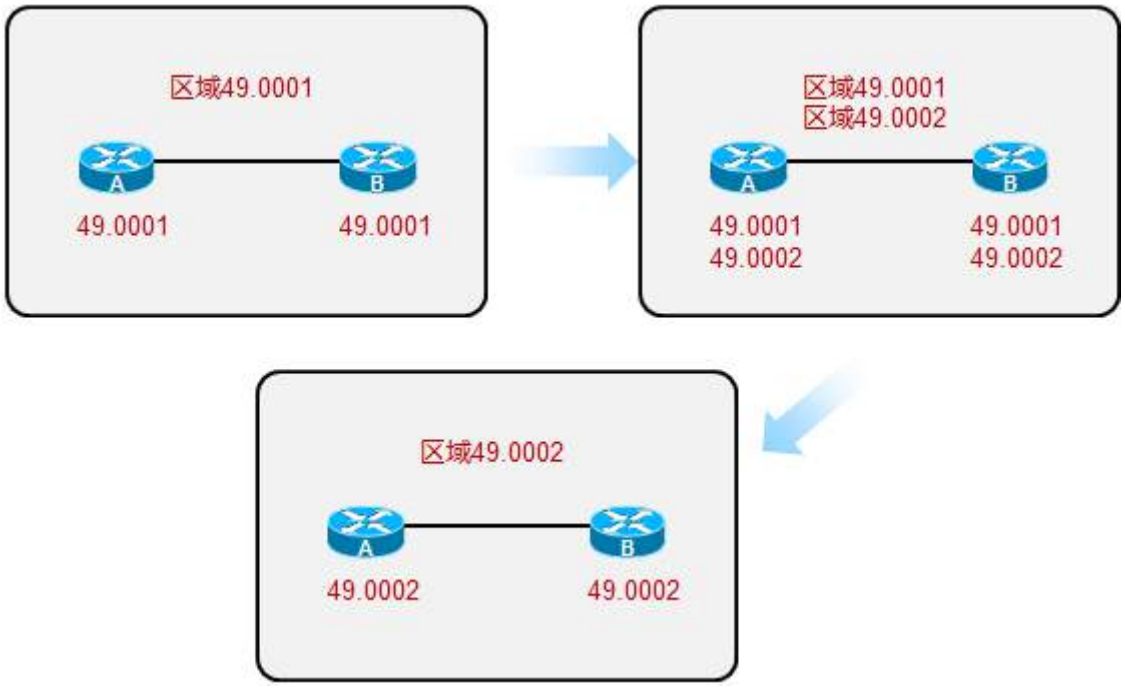
1. 区域合并

上面已经说过了

2. 区域分离



3. NSAP 地址重编址



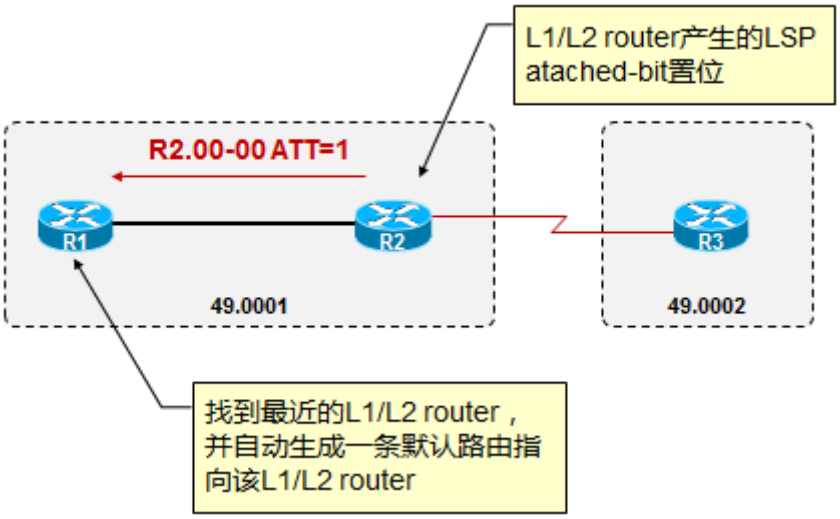
重编址过程和区域合并、分离类似，只是在重编址期间，需要清除一些或者全部路由器的区域前缀，用新的

区域前缀替换。如上图，希望把原先的区域前缀 49.0001 替换成 49.0002，为了实现这个目标，在 A 和 B 上配置一个新区域前缀的 NSAP 地址，注意这两个地址的 SystemID 必须一致。然后依次删除 A 和 B 的区域 ID49.0001 的 NSAP 地址，这样就实现了路由器新的 NSAP 地址的无缝、无冲突和无破坏的重新配置。

5.8 默认路由的注入

在最初的 IS-IS 设计中，Level1 area 是 stub，L1 router 会寻找最近的 L1/L2 router，并生成一条默认路由指向该 L1/L2 router（前提是该 L1/L2 router 进行了 ATT 置位）；然而连接到 Backbone 的 Level2 area 路由器要了解 IS-IS domain 内的所有路由而不会自动产生默认路由。我们可以使用 default-information originate 命令来手工下发一条默认路由进 backbone。该命令产生的路由被装载进 level2 LSP 并泛洪给 backbone 的其他 IS-IS 路由器。

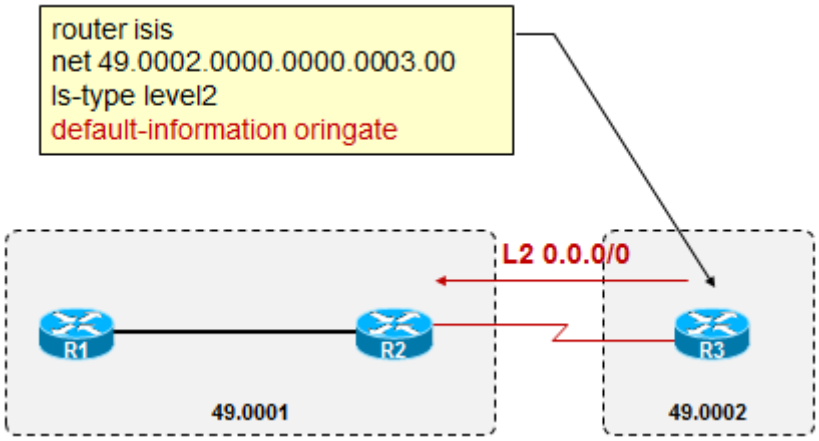
1. **Level1 area 默认路由**



关于 ATT-bit 前面已经讲很多了，这里就不啰嗦了。我在 CISCO IOS Version 12.4(10) -- C3640-JK9O3S-M 上做过测试，在 R1 也就是 L1 router 上，配 default-information originate，无法生成默认路由。

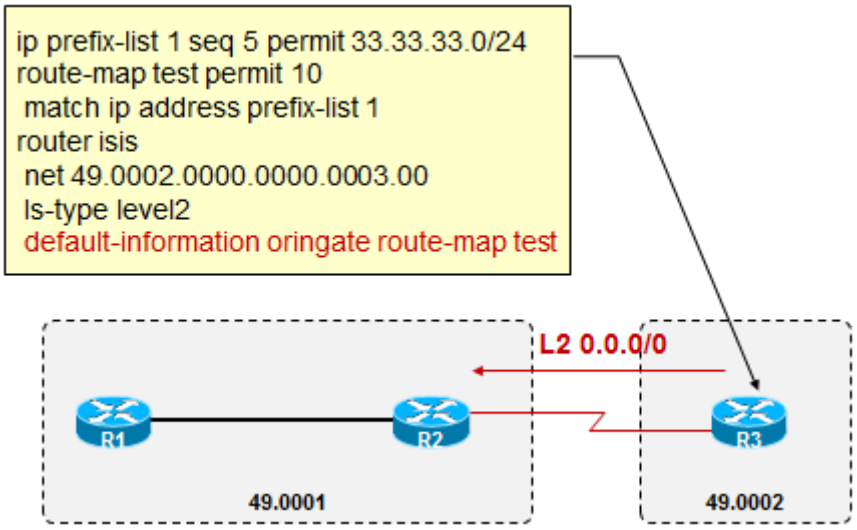
在 L1/L2 rouer 也就是 R2 上配置，会产生默认路由，并泛洪给 R3，但是不会给 R1。

2. **Level2 area 默认路由**



在 R3 上做如图所示的配置，R3 本地路由表无需默认路由，即可产生一条 IS-IS 默认路由并且传递到 backbone，因此 R2 能学习到。这个时候突发奇想，在 R2 上做 level2 路由到 level1 路由的重发布，发现 10.1.23.0 也就是 R2-R3 的直连链路都被导入进了 49.0001，但是 R3 产生的这条默认路由却没有一同进来。

3. **Level2 area 默认路由（条件通告）**



使用 default-information originate 关联一个 route-map，来进行条件通告，只有当 route-map 匹配成立的时候，才会下发默认路由，可以在 route-map 中关联前缀列表匹配特定的路由前缀，只有当 R3 路由表里有这条路由时，才通告默认路由。

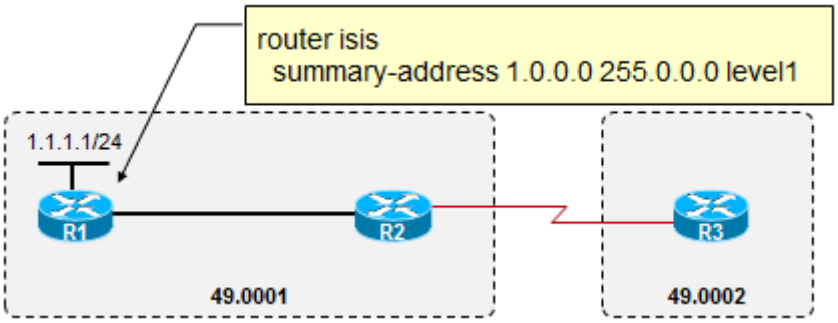
在 CISCO IOS 12.4 上 Acl 用标准和扩展的访问控制列表试过了 ,不成功 ,就算是 acl permit any 都不行。用前缀列表匹配路由就成功了。

5.9 路由汇总

An IS-IS router can be configured to summarize IP routes into Level 1, Level 2, or both, at the same time,

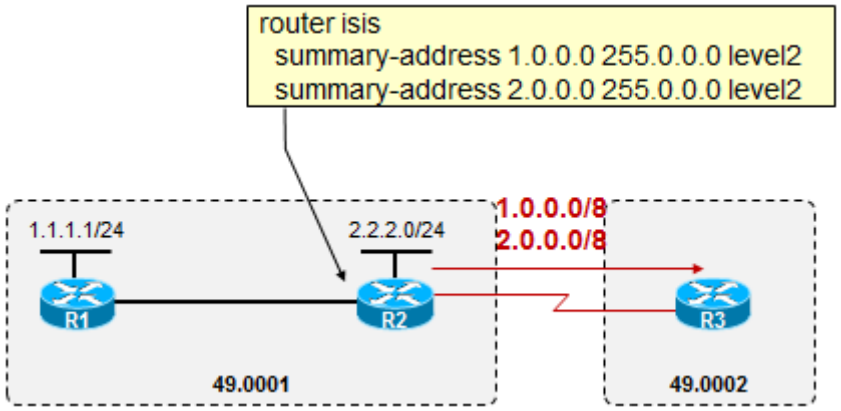
with the following router-level configuration command: **summary-address <prefix> [level-1|level-2|level-1-2].** By default, summaries go into Level 2 if no routing level option is indicated.

1. **在 L1 router 上能否进行路由汇总？**



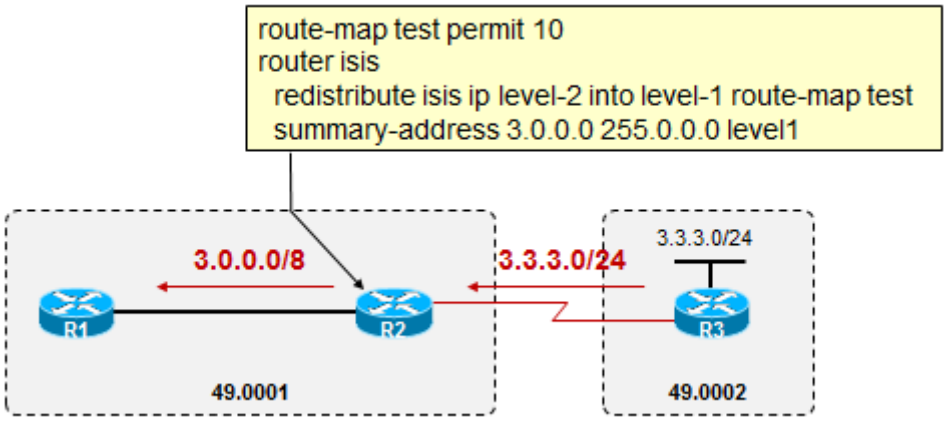
在 R1 这台 L1 router 上企图进行路由汇总，结果是失败的，这点和 OSPF 很像，区域内的路由器是无法做路由汇总的，因为我们要泛洪 LSP，区域内的路由器必须知道区域里拓扑的详细情况，如果允许 L1 路由器进行路由汇总，岂不就失去了链路状态路由协议的天性？

2. **在 L1/L2 router 上进行路由汇总（to level2）**



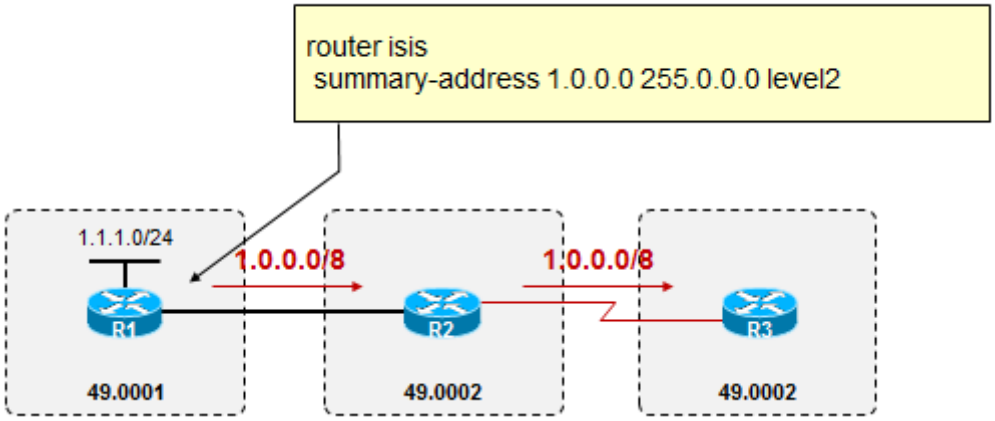
在 L1/L2 router 上，对其下属 Level1 area 的路由进行汇总，使得 backbone 或 level2 area 内学习到汇总路由，配置如上。

3. **在 L1/L2 router 上进行路由汇总（to level1）**



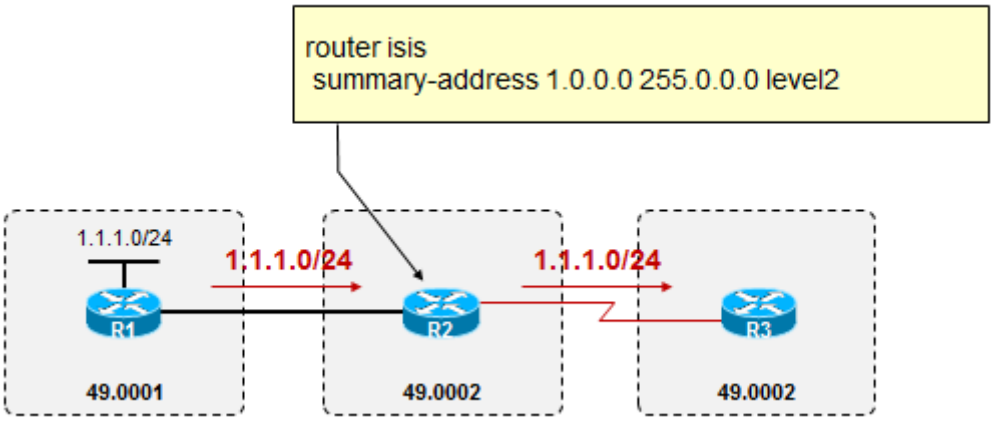
将 level2 路由注入 level1 area，同时在 L1/L2 router 上可以对这些 level2 路由进行汇总，配置如上。

4. **在 L2 router 上进行路由汇总 例 1**



在 L2 router 上进行汇总，对本地始发的路由做汇总，配置如上

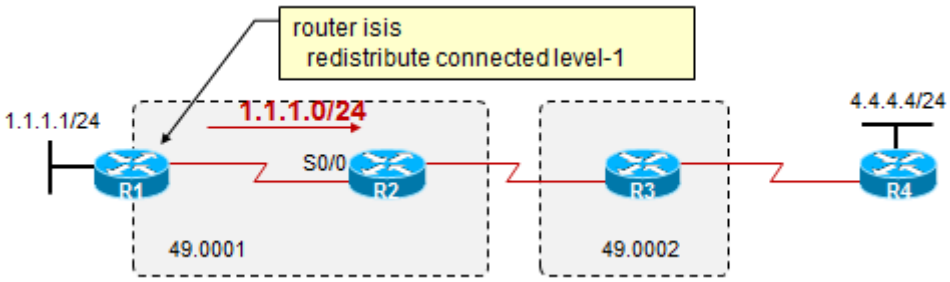
5. **在 L2 router 上进行路由汇总 例 2**



必须在 L2 路由的始发路由器上进行路由汇总，否则无效

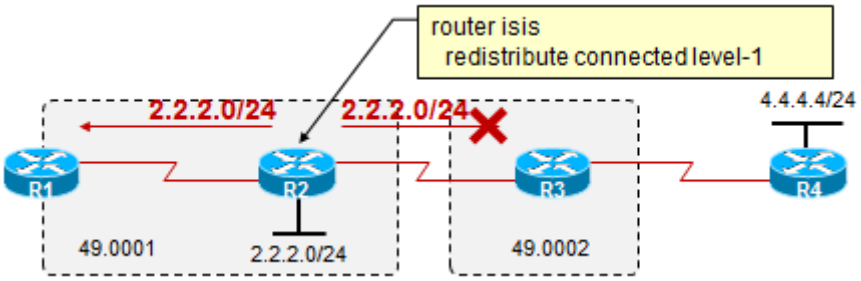
5.10 路由重发布

1. **L1 router 重发布外部路由进 IS-IS**



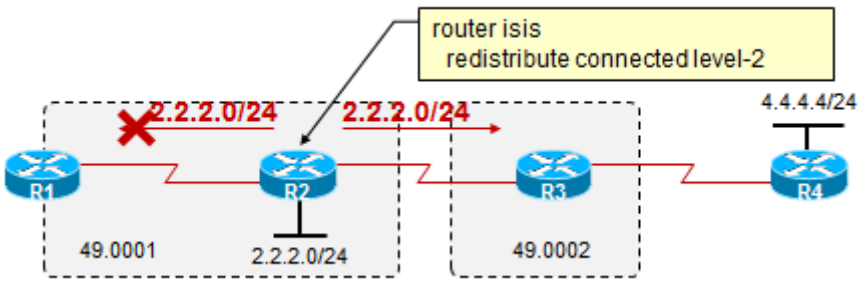
- 在 isis 进程模式中，redistribute 命令直接敲不加 level 参数的话，默认是重发布到 level-2
- 如果不手工设置 metric，重发布进来后默认 metric 为 0，也就是在 R2 上路由的 metric 为 R2 的 S0/0 口的 level1 metric

2. **L1/L2 router 重发布外部路由进 IS-IS （to level1）**



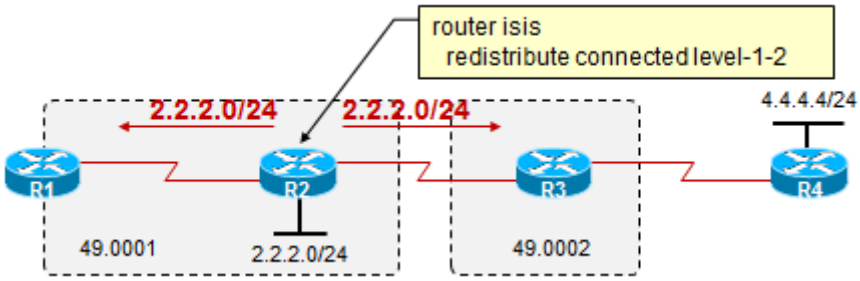
如果在 L1/L2 router 上重发布外部路由到 Level1，则 Level1 area 中能学习到这条路由，但是 R2 不会将路由注入 Level2 area 也就是 Backbone 中

3. **L1/L2 router 重发布外部路由进 IS-IS （to level2）**



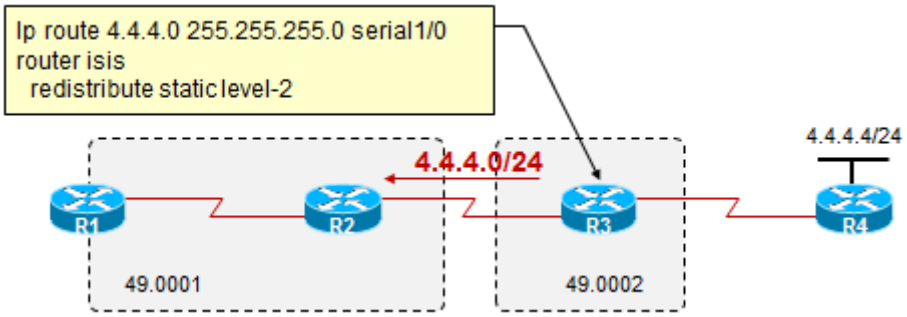
如果重发布命令不加 level 参数，则默认是如图所示的 level-2 参数，也就是将外部路由注入到 level2 中，这样一来 level2 area 都能学习到这条外部路由 ,但是 level1 area 中是无法学习到的 ,因为 R2 在其产生的 level1 LSP 中不会携带该外部路由。

4. L1/L2 router 重发布外部路由进 IS-IS （to level-1-2）

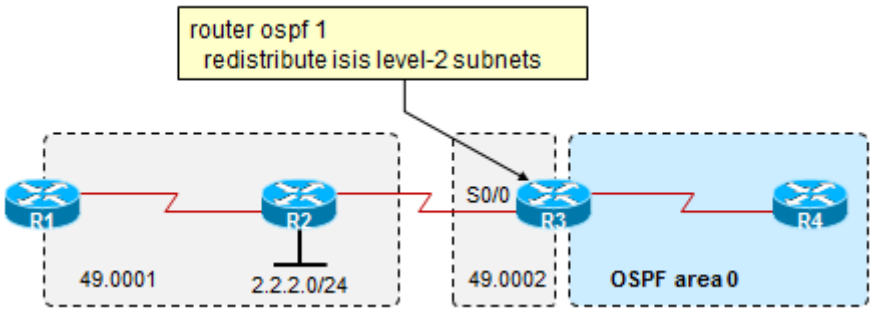


L1\L2 router 在 Level1 area 及 Bacbbone 都会注入这条外部路由

5. L2 rouer 重发布外部路由进 IS-IS



6. IS-IS level2 to OSPF



这条命令只将 R3 上的 L2 路由重发布进 OSPF，但是 R3 激活 IS-IS 的 S0/0 直连口却不会被重发布进 OSPF，这点与其他 IGP 协议不太一样。因此，如果要让路由学全，还需要在 R3 上增加一条重发布直连进 OSPF

5.11 安全及认证

IS-IS 支持明文及密文 MD5 认证。下面做个小小的汇总：

- **明文接口认证**

  我们可以为 IS-IS 的 level1 或 level2 分配一个接口认证密码。如果不指定 level2，则默认的 level1 的认证。如果接口开启了明文认证，那么密码会被插入到所有的 IS-IS 报文中：IIH、LSP、CSNP、以及 PSNP（基于特定的 level）。实际在 CISCO 设备上抓包的时候，发现只有 IIH 中携带了明文的密码信息（放在用于认证的 TLV 中），其他报文没有看到。

  命令如下（接口级别）：

  Isis password <string>

- **明文 area 认证**

  区域验证，其实就是 Level1 area 的验证。在一台 IS-IS 路由器上增加了区域验证后，其将会检查自己收到的 IS-IS 报文，如果报文中没有验证信息或者错误，则忽略。而其自身在产生的 IS-IS 报文则会携带验证信息，如果接收方没有配置区域认证，在接收到这些携带了验证信息的报文后，仍然可以正常的学习。Area 认证的认证信息只会包含在 LSPs 报文中，其他报文不包含。命令如下（路由进程中）：

  area-password <string>

- **明文 domain 认证**

  是在 Level2 LSP、CSNP、PSNP 中插入密码。命令如下（路由进程中）：

  domain-password <string>

  在 CISCO IOS 上，我经过抓包验证，发现使用 domain 认证，IIH 报文、CSNP、PSNP 是不携带认证信息的，LSP 则携带用于认证的 TLV

- **密文接口认证**

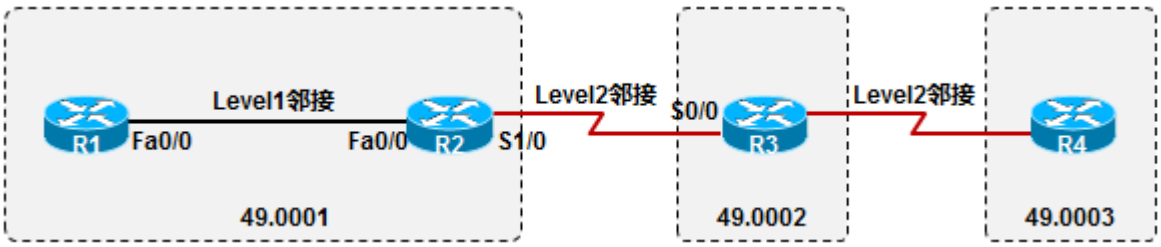
  原理与明文接口认证差不多，只不过可以给予 HMAC-MD5 认证

- **密文 LSPs 认证**

  在 LSP 中携带密文的认证信息，可以指定 level1 或 level2 的密文 lsp 认证

下面一个个的实验一下：

1. **明文接口认证**



我们在 R1 及 R2 之间开启接口认证：

R1 的配置如下（其他基本的配置就直接省略了）：

```
Interface fa0/0
  Isis password cisco
```

在完成这个配置后，来看一下现象：

R1#show isis neighbors

```
System Id        Type Interface IP Address        State Holdtime Circuit Id
```

R1 的 isis 邻居表直接是空的，这是因为它忽略了这个接口上收到的 R2 的 hello 包。

R1#debug isis adj-packets

```
IS-IS Adjacency related packets debugging is on
R1#
*Mar  1 00:09:39.243: ISIS-Adj: Rec L1 IIH from cc01.0a44.0000 (FastEthernet0/0), cir type L1, cir id 0000.0000.0002.02, length 1497
*Mar  1 00:09:39.243: ISIS-Adj: Authentication failed
```

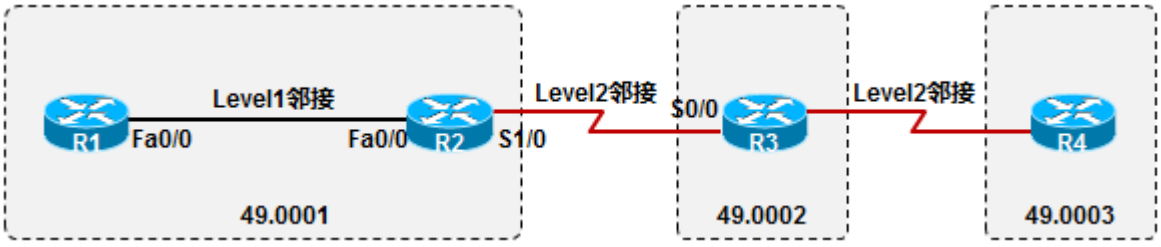
而在 R2 上

R2#sh isis neighbors

System Id	Type	Interface	IP Address	State	Holdtime	Circuit Id
R3	L2	Se1/0	10.1.23.3	UP	28	00
R1	L1	Fa0/0	10.1.12.1	INIT	22	R2.02

R2 已经发送了 IIH，但是 R1 不承认，因为 R2 没有配置认证，这将导致 R1 不会把 R2 放入自己的 IIH 中发回给 R2（包含在 TLV 的 IS neighbors 中）。在 R2fa0/0 口上配上密码，邻接关系就正常了。 通过抓包可以看到在 IIH 中已经包含了用于认证的 TLV，认证密码就在 TLV 中以明文的形式存在。在 LSP、CSNP、PSNP 报文中未看到认证的 TLV。

2. 明文 area 认证



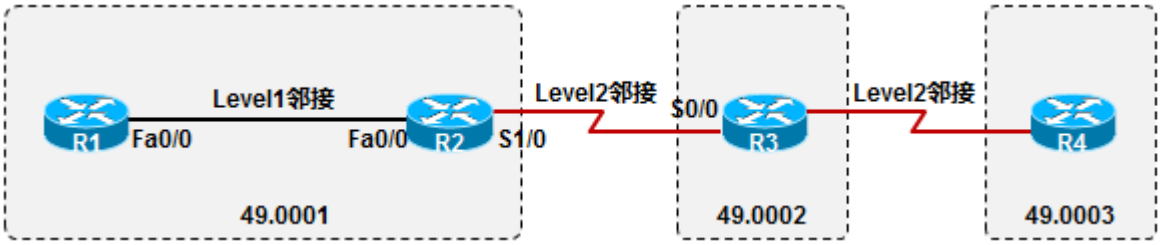
R1 的配置如下（R2 暂时不开启 area password）：

```
router isis
is-type level-1
 area-password cisco
```

配置完成后，我们发现 R1、R2 的邻接关系是 UP 的，也就是说区域认证，IIH 报文中并没有携带认证信息。不仅如此，R1 发出的 PSNP 也没有认证信息，但是发出的 LSP 中，是有明文的认证信息的。实验的现象还是挺有意思的，对于 R1 而言，收到 R2 的所有 IS-IS 消息都是没有认证信息的， R1 主要关注 LSP，发现 R2 发来的 LSP 没有用于认证的 TLV ，因此 R1 忽略 R2 发来的 LSP，这就导致 R1 的路由表是空的；然而虽然 R1 发出来的 LSPs 都是有认证信息的，但是 R2 由于没有开启 area 认证，因此 R2 收到这个包，也可以读取，并且还能用这些 LSP，所以 R2 能学习到 R1 的路由。

这个时候，我们再做一下测试，R1 配置的 area-password 与 R2 配置的不一样。这样一来，R1、R2 虽然邻居关系能建立，但是都无法学习到对方的路由，因为 LSPs 双方都不认啊。

3. 明文 domain 认证



首先配置 R3：

```
Router isis
  Domain-password cisco
```

完成如上配置后，我们发现 R2、R3 以及 R3、R4 的 is-is 邻居关系都是 UP 的。这是因为 IIH 包并没有携带用于验证的 TLV 信息。另外，R3 的路由表是空的，

R2 的路由表如下（省略直连路由）：

```
i L1     1.1.1.0 [115/20] via 10.1.12.1, FastEthernet0/0
i L2     3.3.3.0 [115/20] via 10.1.23.3, Serial1/0
i L2     10.1.34.0 [115/20] via 10.1.23.3, Serial1/0
```

R3 的路由表如下（省略直连路由）：

```
i L2     3.3.3.0 [115/20] via 10.1.34.3, Serial0/0
i L2     10.1.23.0 [115/20] via 10.1.34.3, Serial0/0
```

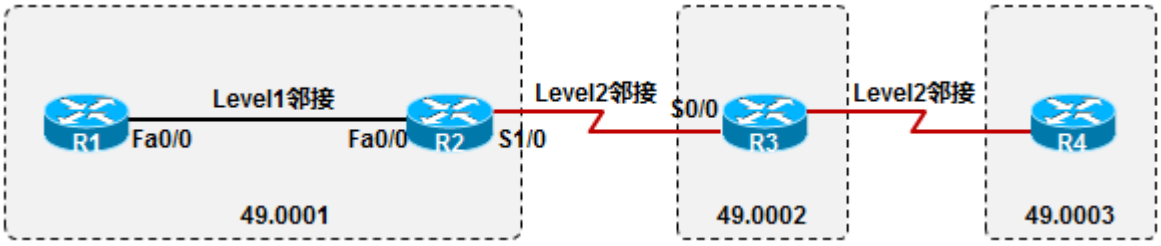
我们发现 R2 能学习到 R1 的 loopback、R3 的 loopback、以及 R3 的直连接口路由 34.0，这是因为 R4 传给 R3 以及 R2 传给 R3 的 LSP 都没有携带认证信息，那么这些 LSPs 到了 R3 之后，R3 是肯定报错的，因此 R3 不会将 R4 的发出的原始 LSP 泛洪给 R2 的，同时 3 也是不会将 R2 发出来的原始 LSP 泛洪给 R4 的。但是，R3 自己发出来的 LSP，里头有 R3 的直连网段 3.3.3.0 及 10.1.23.0、10.1.34.0，这些 LSP 泛洪到了 R2 及 R4，虽然 LSPs 携带了用于认证的 TLV 信息，但是，R2 和 R4 并没有开启 domain 认证啊，因此他们直接无视这些认证信息，然后开始读 LSP，这就是我们实验的结果的解释。当然如果 R2 和 R4 配置了 domain-password 但是密码和其他设备的不一样，那就不用说了，直接丢弃密码不同的 LSP，压根是不会读

取的。

完成实验我们只要在 R2 和 R4 上配置 domain-password 即可，R1 是无需配置的（domain 认证实际上是在 level2 LSP 中嵌入用于认证的 TLV）。

使用 domain 认证，IIH 报文、CSNP、PSNP 是不携带认证信息的，LSP 则携带用于认证的 TLV。

4. 密文接口认证



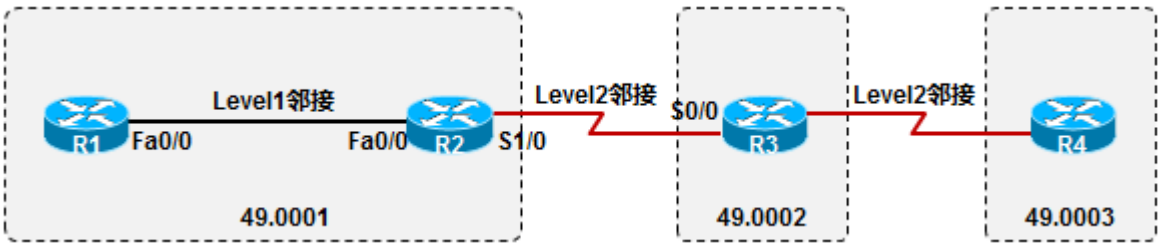
例如要在 R1、R2 之间进行接口密文验证：

R1 的配置如下：

```
R1(config)#key chain test
R1(config-keychain)#key 1
R1(config-keychain-key)#key-string cisco
R1(config-keychain-key)#exit
R1(config-keychain)#exit
!!!
R1(config)#interface fa0/0
R1(config-if)#isis authentication key-chain test level-1
R1(config-if)#isis authentication mode md5 level-1
```

R2 的配置类似，这样一来在 IIH 中用于认证的 TLV 中，装载的就是经过 hash 后的密文，不会暴露密码。

5. 密文 LSPs 认证（Level1）



R1 的配置如下：

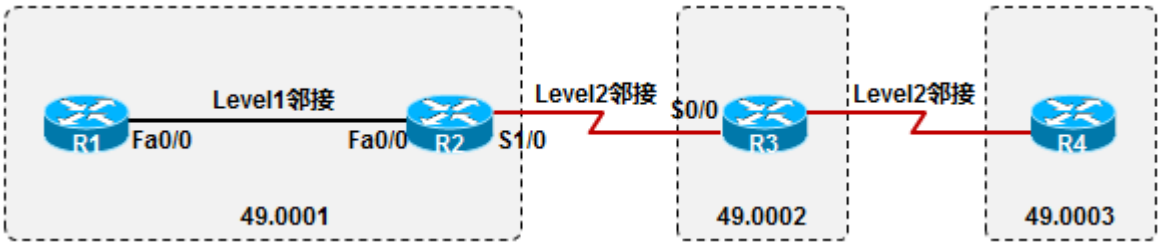
```
R1(config)#key chain test
R1(config-keychain)#key 1
```

```
R1(config-keychain-key)#key-string cisco
R1(config-keychain-key)#exit
R1(config-keychain)#exit
!!!
R1(config)#router isis
R1(config-router)#authentication key-chain test level-1
R1(config-router)# authentication mode md5 level-1
```

这样一来，R1 发送出来的 Level1 LSP 都会携带密文认证的 TLV，然而由于目前为止 R2 并没有开启认证，因此 R1 发送的这些 LSP 到了 R2，R2 直接忽略 LSP 中的认证信息，直接去读取，因此 R2 能学习到 R1 通告的路由。当然 R1 啥路由也没有。

那么只要在 R2 上补全配置就行了。这里不再赘述。

6. 密文 LSPs 认证（Level2）



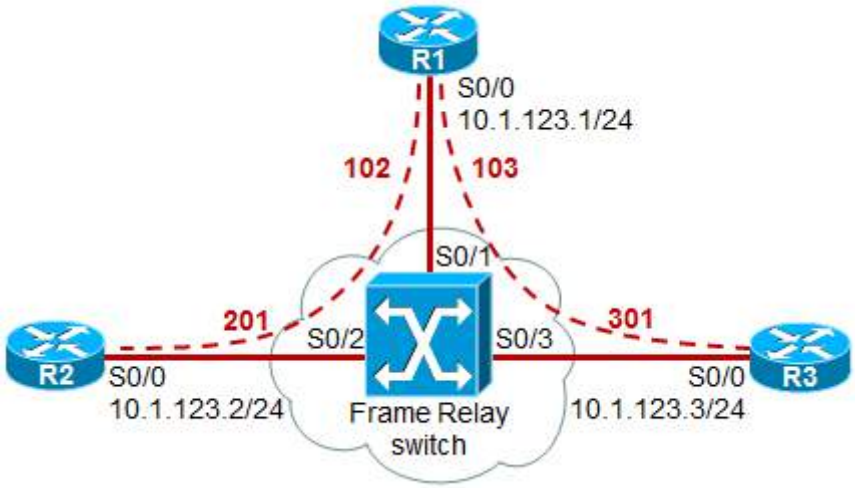
R2 的配置如下：

```
R2(config)#key chain test
R2 (config-keychain)#key 1
R2 (config-keychain-key)#key-string cisco
R2 (config-keychain-key)#exit
R2 (config-keychain)#exit
!!!
R2 (config)#router isis
R2 (config-router)#authentication key-chain test level-2
R2 (config-router)#authentication mode md5 level-2
```

完成如上配置后，R2 发出来的 level2 LSP 就会携带上密文认证的 TLV 信息（level1 LSP 是不会携带的），如果保持这样的配置，我们就会发现 R2 只能学习到 R1 发布出来的路由，这是因为 level1 LSP 不用认证。完成这个实验，R3、R4 补齐配置就成了。

5.12 NBMA 环境中的 ISIS

我们知道 IS-IS 其实真正识别的网络类型就两个，一个是点对点，一个是广播多路访问网络。对于 NBMA，例如帧中继网络来说，IS-IS 的部署要谨慎，来看一下会有什么问题：



上面的网络是一个典型的 Hub&Spoke 网络，R1-R2、R1-R3 之间各维护一条 PVC，R2,R3 之间没有 PVC。

R1 的配置如下：

```
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
 ip router isis
interface Serial0/0
 ip address 10.1.123.1 255.255.255.0
 ip router isis
 encapsulation frame-relay
 isis priority 100                              !! 让 R1 成为 DIS
 frame-relay map ip 10.1.123.2 102 broadcast    !! 这是 IP 的映射
 frame-relay map ip 10.1.123.3 103 broadcast
 frame-relay map clns 102 broadcast             !! 务必不要忘了 CLNS 的映射
 frame-relay map clns 103 broadcast
 no frame-relay inverse-arp
router isis
 net 49.0001.0000.0000.0001.00
```

R2 的配置如下：

```
interface Loopback0
```

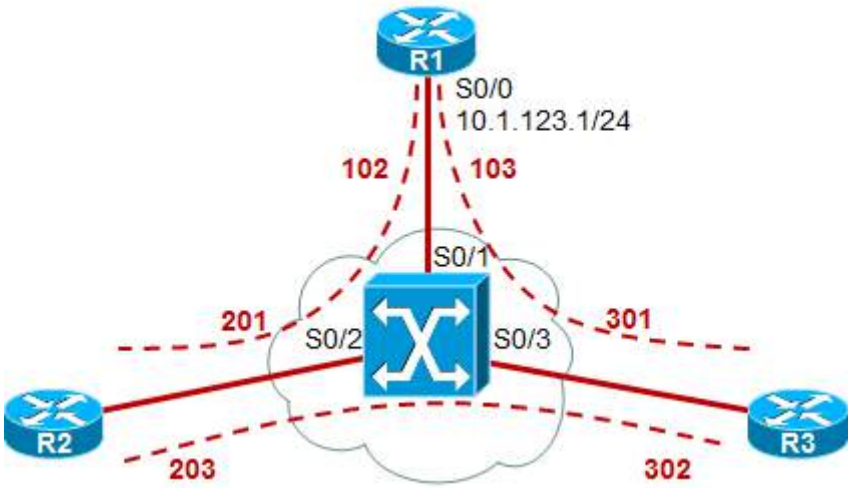
```
 ip address 2.2.2.2 255.255.255.255
 ip router isis
interface Serial0/0
 ip address 10.1.123.2 255.255.255.0
 ip router isis
 encapsulation frame-relay
 frame-relay map ip 10.1.123.1 201 broadcast
frame-relay map ip 10.1.123.2 201 broadcast
frame-relay map ip 10.1.123.3 201 broadcast
 frame-relay map clns 201 broadcast
 no frame-relay inverse-arp
!
router isis
 net 49.0001.0000.0000.0002.00
 is-type level-1
```

R3 的配置如下：

```
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
 ip router isis
interface Serial0/0
 ip address 10.1.123.3 255.255.255.0
 ip router isis
 encapsulation frame-relay
 frame-relay map ip 10.1.123.1 301 broadcast
 frame-relay map ip 10.1.123.2 301 broadcast
frame-relay map ip 10.1.123.3 301 broadcast
 frame-relay map clns 301 broadcast
 no frame-relay inverse-arp
!
router isis
 net 49.0001.0000.0000.0003.00
 is-type level-1
```

在完成上述配置后，我们会发现，R1 能学习到 R2、R3 的路由，但是 R2、R3 之间却无法学习到对端的路由。

接下去，在 R2、R3 之间增加一条 PVC：



路由才算正常，三台路由器都能学习到全网的路由。

所以：

- **ISIS 对于 NBMA 网络的支持还是存在缺陷的，像前面实验演示的那样，如果是在帧中继主接口上跑 ISIS，或者在 P2MP 子接口上跑 ISIS，那么就必须要求 PVC 全互联，否则路由学习不正常，但是即使是 PVC 全互联，也存在隐患，如果某条 PVC 故障了，那么路由还是会出现问题。**
- **所以在 NBMA 网络中，譬如帧中继环境，强烈建议使用 P2P 子接口来跑 ISIS。**

6 IS-IS 对 IPv6 的支持

IETF 的 draft-ietf-isis-ipv6-05.txt 中规定了 IS-IS 为支持 IPv6 所新增的内容。主要是新添加的支持 IPv6 路由信息的两个 TLVs（Type-Length-Values）和一个新的 NLPID（Network Layer Protocol Identifier）。IS-IS 不像 RIP 和 OSPF，这两个协议有单独的版本 RIPng 和 OSPFv3 支持 IPv6。

新增的两个 TLV 分别是：

- IPv6 Reachability：类型值为 236（0xEC），通过定义路由信息前缀、度量值等信息来说明网络的可达性。
- IPv6 Interface Address：类型值为 232（0xE8），它相当于 IPv4 中的“IP Interface Address”TLV，只不过把原来的 32 比特的 IPv4 地址改为 128 比特的 IPv6 地址。

NLPID 是标识网络层协议报文的一个 8 比特字段，IPv6 的 NLPID 值为 142（0x8E）。如果 IS-IS 路由器支持 IPv6，那么它必须以这个 NLPID 值向外发布路由信息。

7 IS-IS 与 OSPF

集成型 IS-IS 和 OSPF 都是 20 世纪 80 年代后期定义的，大约是 1988 年。OSPF 实际上是从 IS-IS 的早期版本进化而来的，不过它采用 IP 作为前提。OSPF 的核心概念，例如链路状态信息的扩散、SPF 算法以及在广播链路中使用指定路由器都是从 IS-IS 早起版本中借用过来的。

OSPF version1 在 1989 年 10 月作为 RFC1131 发布。而 1990 年 12 月具有 IP 路由选择扩展功能的集成 IS-IS 协议作为 RFC1195 被发布。

CISCO 实现的集成 IS-IS 协议最早是在 1991/1992 这个时间段发布的。此后进行了一系列的改进和扩展。在主要的 ISP 网络中大规模开发 IS-IS 协议开始于 1995 年 ,其中另一个重要的原因就是美国政府对 ISO CLNS 的兴趣。虽然集成 IS-IS 具有双重路由选择功能，但是有趣的是当今大多数全球最大的 ISP 网络中使用 IS-IS 协议都只是为了实现 IP 路由选择。

由于 OSPF 天生被设计用来支持 IP 路由选择，而 IS-IS 仍然有部分厂商的设备不能完美支持，因此对于相当一部分的网络来说，OSPF 仍然是首选。然而将集成型 IS-IS 用于 IP 路由选择的兴趣还在增加，欧洲的大多数 ISP 用的都是 IS-IS。许多新的 ISP 也在考虑部署 IS-IS。

当然，在集成 IS-IS 及 OSPF 都在测试期间获得成功并且都已经获得了 ISP 网络内部网 IP 路由选择协议的参选资格。两种协议在过去一段时间都取得了一些新的扩展。例如对 MPLS TE 的支持、对 IPv6 的支持等等。另一个方面，我们能够观察到的一个比较直观的现象是，相比于 IS-IS，OSPF 的技术文档要多得多。

IS-IS 与 OSPF 的术语比较：

IS-IS	OSPF	注释
终端系统 ES	主机	
中间系统 IS	路由器	
电路 circuit	链路 Link	
SNPA	数据链路层地址	
PDU	Packet	
DIS	DR	
-	BDR	
IIH	Hello	
LSP	LSA	
CSNP	DBD	
PSNP	LSR 或 LSAck	
路由 domain	AS	
Level2 area	Backbone area	

Level1 area	常规 area（非骨干区域）	
虚链路	虚链路	两者存在区别
系统 ID	routerID	
LSP ID	链路状态 ID	

共同之处：

- 都是链路状态路由协议，都要求区域内的路由器交换链路状态信息，链路状态信息被收集到链路状态数据库中
- 都使用了一种实现路由选择信息交换的相似机制，即扩散
- 都在广播网络中选择指定路由器来控制扩散并降低这类介质中多对多邻接的系统资源需求
- 都是基于链路状态数据库中的信息，采用几乎相同的算法—SPF 算法来计算最佳路由
- 都支持两个 level 的分层路由选择
- 都支持 IP 前缀的无类路由选择（支持 VLSM）
- 都是公有协议（对比于 CISCO 私有的 EIGRP）

重要区别：

	IS-IS	OSPF
1	IS-IS 支持 ISO CLNP 和 IP 两种环境	仅支持 IP
2	IS-IS 报文封装在数据链路层帧中	OSPF 报文封装在 IP 包中，因此在网络层传输
3	IS-IS 支持 ISO 无连接网络环境，注意数据链路是 ISO 协议（在以太网上数据链路类型为 FEFE），在 ISO 协议栈中 IS-IS 网络层协议 ID 是 0x83	OSPF 封装在 IP 报文中，协议号为 89
4	IS-IS 路由器通告包含直连邻居及路由信息 TLV 的 LSP，使用 LSP 承载所有的路由选择信息	OSPF 使用不同类型的 LSA 承载不同的路由信息，LSA 被封装进 LSU 通告给邻居
5	IS-IS 数据包利用 TLV 字段承载所有易于扩展的信息	OSPF 只有 LSA 可扩展，而 LSA 扩展性太差
6	IS-IS 可以忽略它不支持的 TLV	网络中的路由器为了进行适当的操作必须识别所有 LSA
7	IS-IS 数据包可以承载多个 TLV，只有一个包头，节省带宽	1 类、2 类 LSA 可以承载多个 IP 前缀；3 类、4 类、5 类 LSA 只能承载单个 IP 前缀，如果需要发送多个 IP 前缀信息，需要多个 LSA
8	对于所有实际应用，IS-IS 仅支持广播和点对点链路。不支持 NBMA 链路。在 NBMA 环境下，可配置为 P2P 子接口或者广播链路（如果是全互联的	OSPF 支持如下网络类型：P2P、广播、NBMA、点到多点和按需电路

	连接方式）	
9	仅仅在广播链路实现 3 步邻接关系，IETF 正在努力制定点到点链路的 3 步进程	OSPF 邻接关系的建立涉及到一个更加复杂的过程
10	最初数据库同步在邻接关系建立后进行	最初数据库同步在邻接关系形成之前进行
11	IS-IS 路由器只属于一个特定区域	OSPF 基于接口划分区域，路由器可属于不同区域
12	区域的边界在链路上	区域的边界在路由器上
13	默认情况下 IS-IS 区域是 stub 区域，最新发布的 RFC2966 标准化了从 Level2 到 Level1 的路由泄露	默认情况下，OSPF 区域不是 stub，可以配置为 stub
14	IS-IS 仅支持在点对点链路上可靠扩散，广播链路上的扩散是不可靠的。然而通过 DIS 周期性的广播实现可靠性	OSPF 确保所有链路上扩散的可靠性
15	DIS 无备份 DIS，DIS 可以被抢占，DIS 以 3 倍的频率发送 Hello PDU	有 BDR ,DR 不能被抢占 ,DR 以正常的频率发送 Hello 报文
16	默认情况下 ,IS-IS 的 LSP 最大生存时间为 1200s，刷新间隔为 900s，而且定时器的值可调	OSPF 的 LSA 的老化时间为 3600s，刷新间隔为 1800s，而且是固定值
17	默认情况下，IS-IS 的接口 cost 值为 10	默认情况下，OSPF 的接口 cost 值根据带宽进行计算
18	默认情况下 ,IS-IS 保持时间(holding-time)为 30s，而且在建立邻接关系时不需要双方的保持时间匹配	默认情况下，OSPF 的保持时间（dead-interval）为 40s ,而且为了建立邻接关系 ,必须使双方的保持时间一致
19	IS-IS 通过将 Hello PDU 的大小填充至接口 MTU 大小来检查双方的 MTU 是否匹配	OSPF 通过在 DBD 报文中嵌入接口 MTU 字段来检查双方的 MTU 是否匹配
16	由于 IS-IS 区域中 IP 前缀是 SPF 树的叶子，故部分路由计算（PRC）较多，通常这就意味着在一个大的区域中路由处理器的负载较低	部分 SPF 被限制用于域间和外部路由 ,任何要求较小区域和分层拓扑扩展引起的域间链路动荡导致完全 SPF 计算
17	没有对 Ip 组播路由选择的支持	MOSPF 扩展提供对 IP 组播路由选择的支持

8 参考书目

Cisco Press - IS-IS Network Design Solutions

IS-IS 网络设计解决方案

相关标准：

文档编号	描述
ISO 10589	ISO IS-IS Routing Protocol
ISO 9542	ES-IS Routing Protocol
ISO 8348/Ad2	Network Services Access Points
RFC 1195	Use of OSI IS-IS for Routing in TCP/IP and Dual Environments
RFC 2763	Dynamic Hostname Exchange Mechanism for IS-IS
RFC 2966	Domain-wide Prefix Distribution with Two-Level IS-IS
RFC 2973	Domain-wide Prefix Distribution with Two-Level IS-IS
RFC 3277	IS-IS Transient Blackhole Avoidance
RFC 3358	Optional Checksums in IS-IS
RFC 3373	Three-Way Handshake for IS-IS Point-to-Point Adjacencies
RFC 3567	Intermediate System to Intermediate System (IS-IS) Cryptographic Authentication
RFC 3719	Recommendations for Interoperable Networks using IS-IS
RFC 3786	Extending the Number of IS-IS LSP Fragments Beyond the 256 Limit
RFC 3787	Recommendations for Interoperable IP Networks using IS-IS
RFC 3784	IS-IS extensions for Traffic Engineering
RFC 3847	Restart signaling for IS-IS

MPLS TE

Version 1.5

2013-07-26

文档作者： 红茶三杯

文档备注： 文档将不定期更新，请关注 http://ccietea.com 以便获得最新版本的文档。

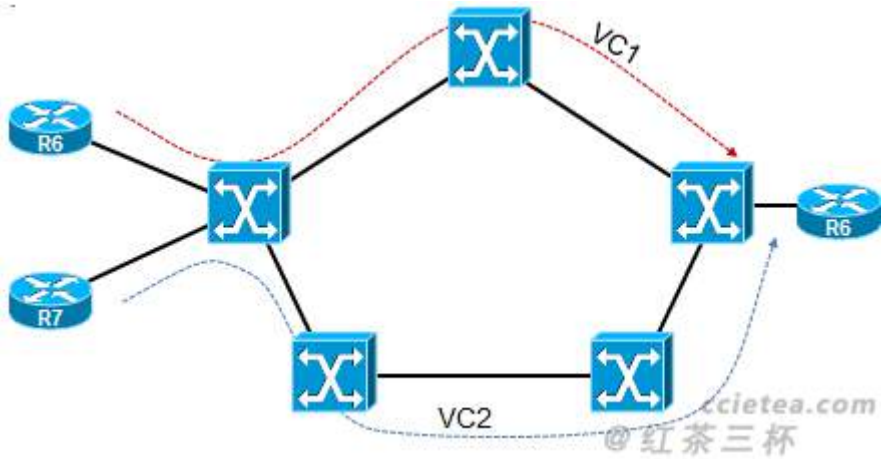
1 MPLS TE 概述

1.1 TE 概述

流量工程（Traffic engineering）我们可以笼统的理解为“驾驭流量穿越网络的能力”，我们能够通过 TE 的部署让流量以最优的方式在网络中从一个节点传输到另一个节点。

传统的 TE 解决方案有如下这么一些示例：

- **TE with Layer3**
  - 通过动态路由协议控制 metric 来影响路由选路
  - 源路由、策略路由、静态路由等等
- **TE with Layer2 overlay**
  - 例如帧中继环境下



1.2 MPLS TE

传统 IP 路由的尴尬：

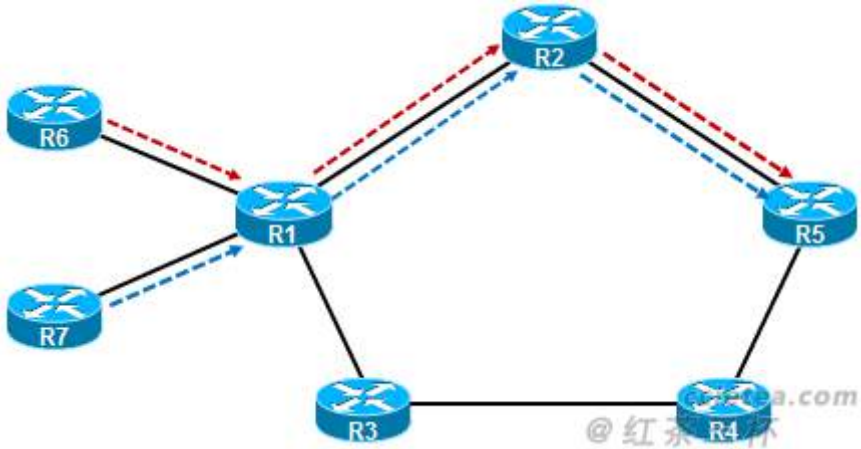
- IP 网络中的路由协议根据最小 metric 原则进行数据包的转发。
- IP 报文在每一跳路由器中的转发都是仅仅根据目的地址进行。
- 每一个被转发到某路由器的 IP 报文在经过该路由器之前及被转发之后都是相同的。

- IP 转发实例并不考虑链路的可用带宽和负载能力，有可能链路所分配的 metric 与实际的度量值并不一样。
- 这种 IP 报文的转发行为可能会导致某些网络中链路过渡使用，而另一些链路使用概率却很小。
- 基于 IP 的流量工程是无连接的，不能实现显式路径（explicit routing）

TE 可以提供一种解决方案，通过对流量，或者部分流量的控制来避免链路过载的情况发生。

下面我们来做个对比：

1. IP 流量工程的状况

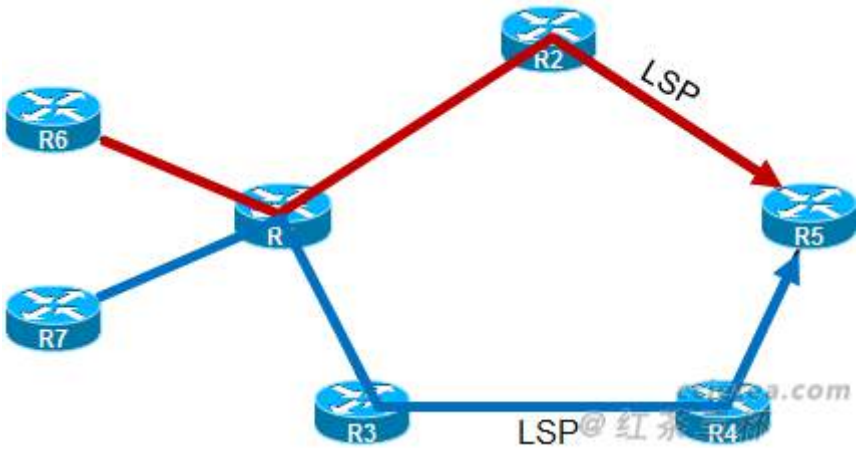


像上图，就非常容易出现流量走单边，R1-R3-R4-R5 链路却没有承载流量的情况，也许可以通过调整诸如 cost 值等手段使得下面的链路也能使用，但是这种方法有可能在调整的同时却给其他的流量带来问题。总的来说，IP 流量工程对流量的引导决策还是比较单一的，无非是像 ospf 协议通过操控 cost 或者利用协议本身的特性来操控路由，或者像 BGP 那样操控路径属性，又或者通过策略路由来操控。

2. MPLS TE 的状况

MPLS TE 提供了如下解决方案：

- MPLS TE 提高了流量在网络中扩散的效率，避免了链路的使用不充分和使用过度
- MPLS TE 考虑了配置（静态）在链路上的带宽
- MPLS TE 考虑了链路的属性参数（如延迟、抖动等）
- MPLS TE 可以通过自适应来改变链路的带宽和属性参数
- 使用流量工程的负载使用的是基于源的路由，而非基于目的 IP 地址的路由



MPLS TE 实现的机制中，一条 LSP 的首端路由器能够了解到网络的拓扑，并可以通过计算得出穿越网络到达该 LSP 尾端路由器的最优路由（不仅仅考虑 metric，还考考虑带宽、链路属性等元素）。

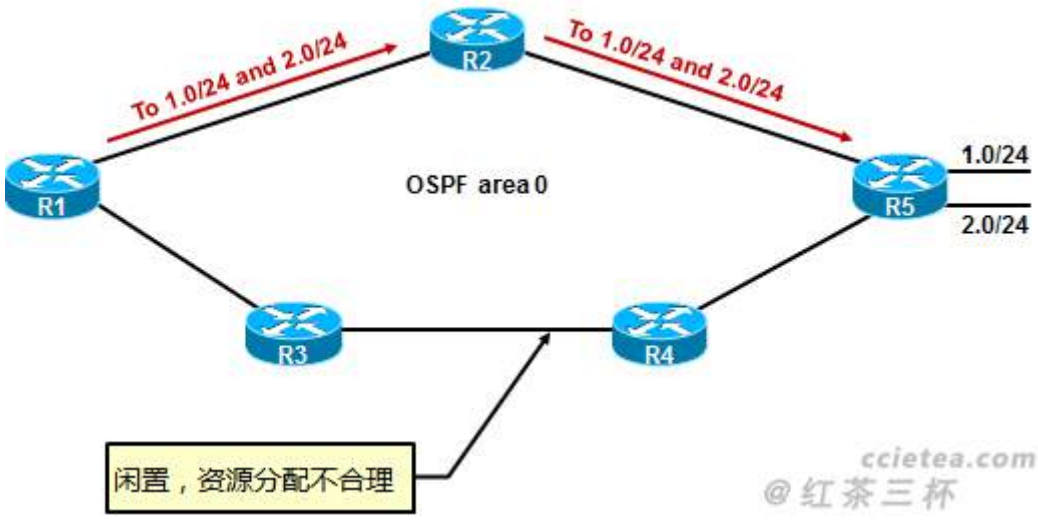
在运行了 MPLS 的网络中，可以将这两条路径设置为不同的 LSP，使用不同的标签。在 R1 上，R6 和 R7 发送给它的标签数据通过不同的入站标签值说明了数据是属于 R6 还是 R7。LSP 中端点 LSR 需要了解链路的带宽以及其他参数。所以 MPLS TE 端点（首、尾端 LSR）的路由协议必须是链路状态路由协议。通过链路状态协议，每个区域内的路由器都对本区域的拓扑有全面的了解。这样一来首端 LSR 就知道如何安排使用基于 MPLS 流量工程的 LSP 了。

这里可以使用基于源的路由。这条 LSP 被称为 MPLS TE 隧道。它是单向的。并且隧道的配置只需要在首端 LSR 上进行就可以了。

1.3 MPLS TE tunnel

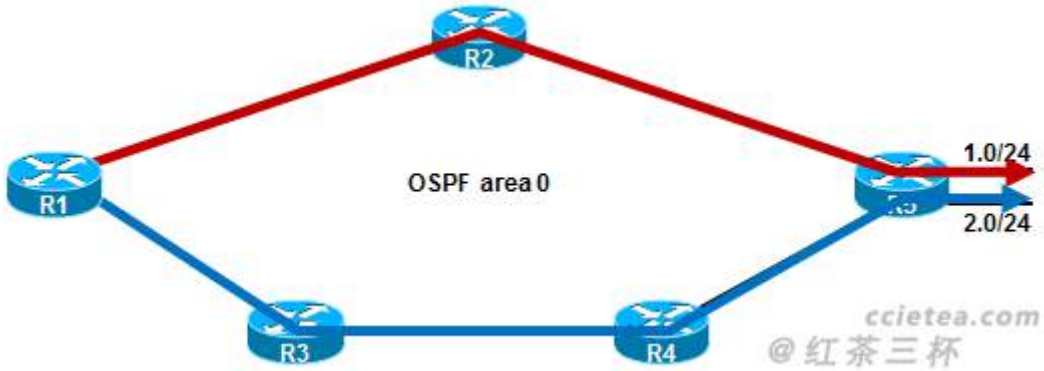
MPLS TE tunnel 也叫做 Traffic trucks，用于克服 hop-by-hop IP 路由的缺陷

- **TE tunnels are an aggregation of traffic flows of the same class (bandwidth , etc.) which are placed inside a common MPLS LSP**
- **TE tunnels are distinct from MPLS LSP through which it traverses:**
  - In operational contexts , a tunnel can be removed from one path and palced onto another.
- **Attributes are explicitly assigned to the TE tunnel through administration action.**
  - Its ingress and exgress LSRs
  - Bandwidth , latency , policy constrains
  - Class of data – The FEC which is mapped onto it



传统的 IP 路由在进行路由选择的时候，只会单纯的考虑 metric，例如如果上图中，我们运行的是 OSPF，那么最终 R1 去往 1.0/24 及 2.0/24 都会优选从 R2 到 R3 这条路径，因为 metric 小。那么这就直接造成了另一条可选路径的闲置以及主走路径的过度拥塞。

那么我们可以利用 TE tunnel 思维，在 R1 上建立两条 TE tunnel，如下图：



这样一来，我的网络资源就得到了更合理的利用，当 R1 去往 1.0/24 及 2.0/24 的时候，会将流量在 R1 上就送入不同的 tunnel。此刻 R1 是我 tunnel 的首端，R5 是尾端。Tunnel 可以设置带宽要求，以便我这条 tunnel 在网络中进行更加科学和合理的选路，而不仅仅是使用 OSPF metric 来进行选路。Tunnel 路径的计算是在 R1 也就是 tunnel 首端路由器上完成了，为了让首端路由器能够更加合理的进行 tunnel 路径的计算（不仅仅使用 metric，还能用可用带宽、链路属性等元素参与路径计算），我们就要求对网络中运行的链路状态路由协议进行扩展，以便泛洪更多路径计算所需的信息，扩展的 OSPF 和 IS-IS 都能完成此重任。

1.4 MPLS TE 组件

链路的限制

每条链路所能支持的最大流量以及链路所能使用的TE隧道

TE信息分发

通过启用了MPLS TE的链路状态路由协议

路径计算CSPF

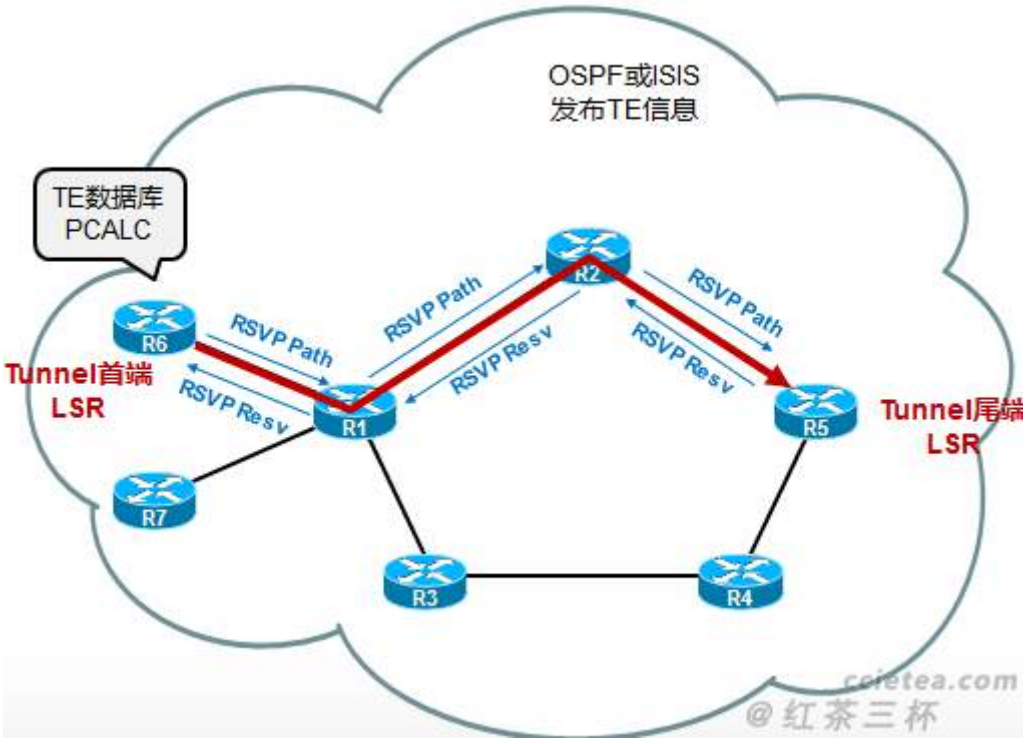
一种用来计算从前端LSR到尾端LSR的最优路径的算法

RSVP

一种用来在穿越网络的TE隧道中发送信号的信令协议

Static、AutoRoute、PBR

一种将流量转发至TE隧道的方法

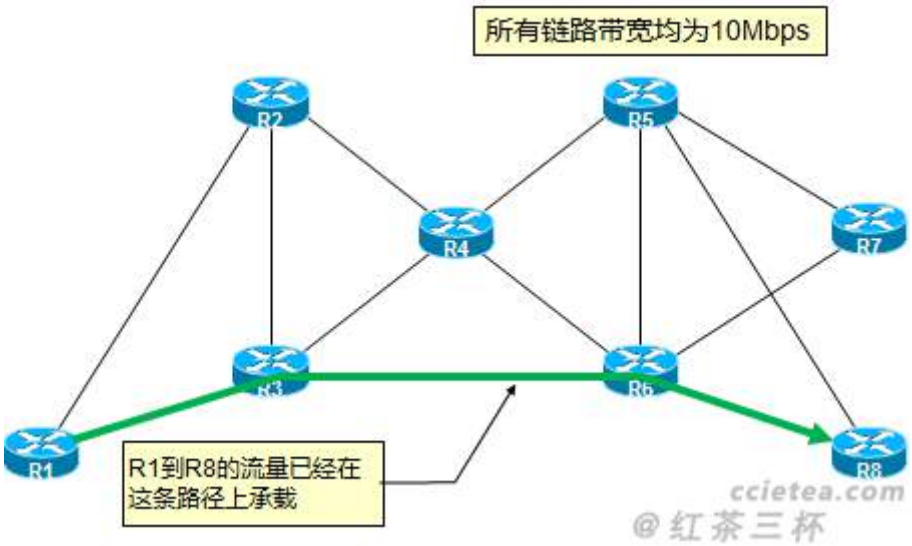


首先我们需要一种能通告各种网络资源（权重、IGP metric、可用带宽、链路属性等）的路由协议，这个路由协议必须是链路状态路由协议，而且必须支持 MPLS TE 扩展，我们有两个可选，一是 OSPF 另一个是 IS-IS。在 CISCO IOS 内部会根据“**针对 MPLS TE 进行扩展的链路状态路由协议**”发送的 TE 信息构建一个 **TE 数据库**，这个数据库包含了所有启用了 MPLS TE 的链路以及这些链路的特征或者参数。可用于 MPLS TE 的带宽以及链路属性等参数在网络中的所有链路都是可配置的。

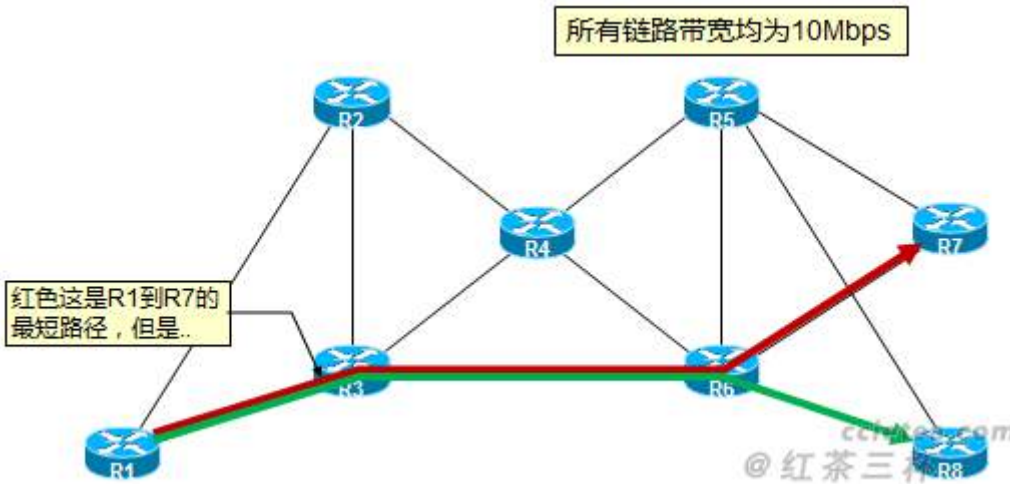
当你在一台 LSR 上配置一条 **TE Tunnel** 的时候，该 LSR 就变成了这条 TE tunnel 的首端 LSR。接下来你就可以指定 TE Tunnel 的目的 LSR，以及它必须遵守的限制例如隧道的保证带宽等等，这些相关参数在首端 LSR 的 Tunnel 口中进行配置。接下去在首端 LSR 上，路径计算 PCALC 或者限制的 SPF（CSPF）算法可以根据上面我们提到的 MPLS TE 数据库结合 Tunnel 的要求计算出最短的、满足所有条件的、从首端 LSR 到尾端 LSR 的路径。完成这些工作后的输出，是一个从 tunnel 头端到尾端的一个 IP 地址序列，指引出**一条路径**。PCALC 和 CSPF 是用于 MPLS TE 的一种 SPF 算法的变体。

前面 CSPF 计算出来的一个结果（一条路径），会输送到另一个模块：**RSVP**。MPLS TE 的流量的靠标签来转发的，这个标签是靠 RSVP 分发的，另外，资源的预留也是通过 RSVP 来完成。LSP 的链路中 LSR 需要了解用于特定 LSP 的 TE 隧道的入站和出站标签。链路中的 LSR 只能通过在首端路由器和链路中的 LSR 之间的信令协议来学习标签。CISCO 使用 TE 扩展的 RSVP 来为 MPLS TE 隧道传递信令。RSVP 的扩展使得 RSVP 可以承载 MPLS 标签信息和其他 TE 特定的参数。本质上说，RSVP 尝试沿着从首端 LSR 到尾端 LSR 的路径传递信令，而这个路径是根据首端 LSR 的 TE 数据库计算得来的。RSVP 的另一个非常重要的功能是建立及维护资源预留（如带宽）。

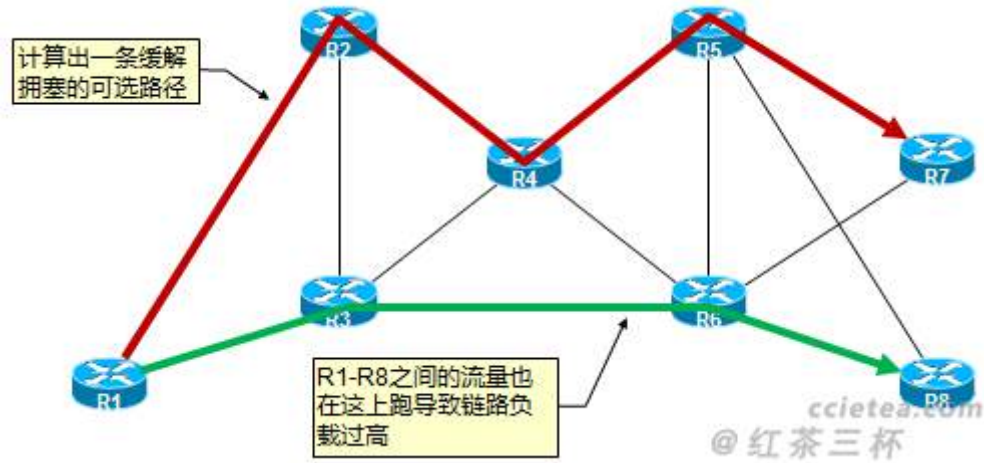
- **工作过程简要介绍**



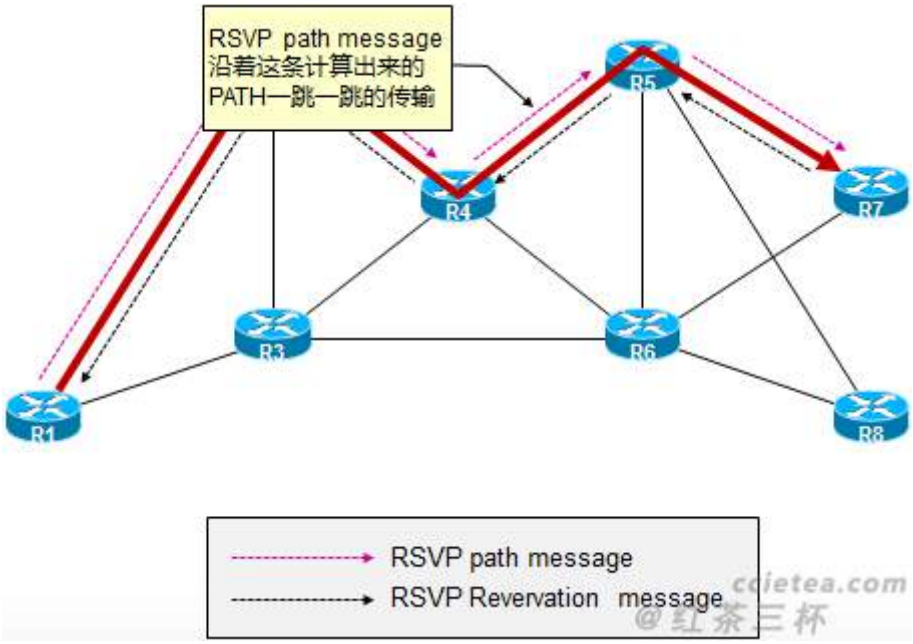
初始环境如上图，R1 到 R8 的流量，已经在图中所示绿色的路径上被承载。



我们假设上述拓扑中，所有链路的带宽均为 10Mbps。那么运行 IGP 协议后，我们知道从 R1 到 R7，最短的路径就是 R1-R3-R6-R7，但是由于网络中 R1-R8 之间的流量已经在这个路径上运行了，并且占用了比较高的带宽，链路的负载过高了，传统的 IP 路由协议是没办法这么智能的考虑到这点，但是利用 MPLS TE，我们可以让 R1 到 R7 走一种更优的路径。。

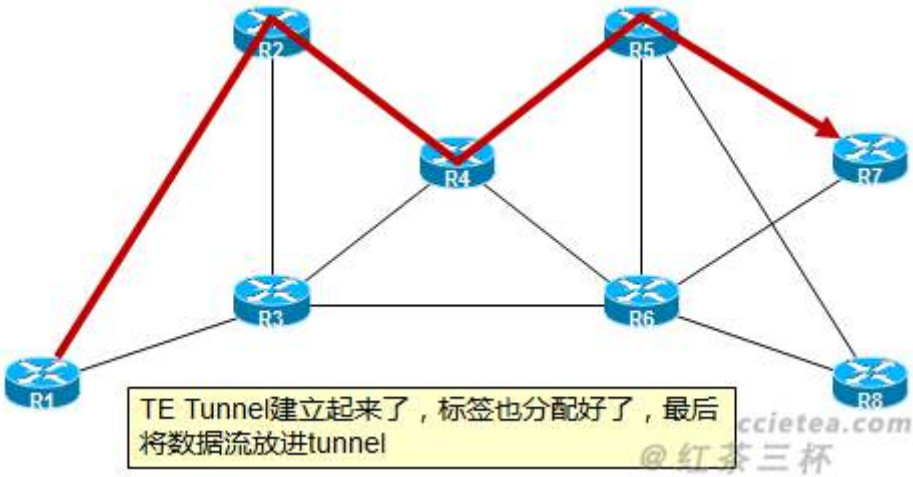


接下去，R1 通过已有的信息，计算出另一条到达 R7 的可选路径：R1-R2-R4-R5-R7，这条路径通过 CSPF 计算得出。上图中，红色的路径就是经过 CSPF 计算得出的。



计算出路径之后，需要通过 RSVP 来预留资源及分发标签，RSVP path message 沿着这条计算出来的 PATH 一跳一跳的传输并且请求资源及标签，然后 R7 会给应答同时将标签带回来，这个标签分发和资源预留的动作也是一跳一跳的进行。这样最终一条 LSP 就建立好了。

那么最后 ,就是将数据引入建立好的 tunnel 上。实际上 ,在 R1 上创建一条 TE Tunnel 后 ,在其本地一个 Tunnel 接口就会被创建。我们只要将流量放入 tunnel 口即可，方法还是有不少，我们在后续的内容中继续介绍。



2 MPLS TE 信息分发

2.1 概述

MPLS TE tunnel 的路径计算需要有相关的信息才能够进行。那么我们就需要使用一种链路状态路由协议来泛洪链路的信息到网络中所有运行了 TE 的路由器中。TE 路由器搜集这些信息后 ,建立自己的 MPLS TE Database。

为了让网络设备进行更加智能化的路径计算，需要发布各种信息：

- 链路状态信息　　IGP 本身就支持
- TE Metric　　相当于权重值的概念，注意这要和 IGP metric 区分开来
- 可用带宽　　接口的可用带宽
- 隧道优先级
- 属性标记及亲和属性

先做个简要的介绍：

- TE metric 是一个用于构建与 IP 拓扑完全不同的 TE 拓扑的参数（虽然默认情况下与 IGP metric 相等）
- 最大带宽就是链路的全部带宽，在 CISCO IOS 中，这个值是匹配物理链路上的，或者是配置的带宽值
- 最大可预留带宽很明显是表示链路中可供 TE 使用的带宽，可以通过 ip rsvp bandwidth 命令配置
- 不可预留带宽是提供给 TE 后所剩下的带宽，也就是链路最大带宽减去目前被 TE 隧道所保留的带宽
- 属性标记是一个 32bits 的字段，这个在后面有详细的介绍

2.2 对 IGP 的要求

2.2.1 针对 TE 的 OSPF 扩展

1. **概述**

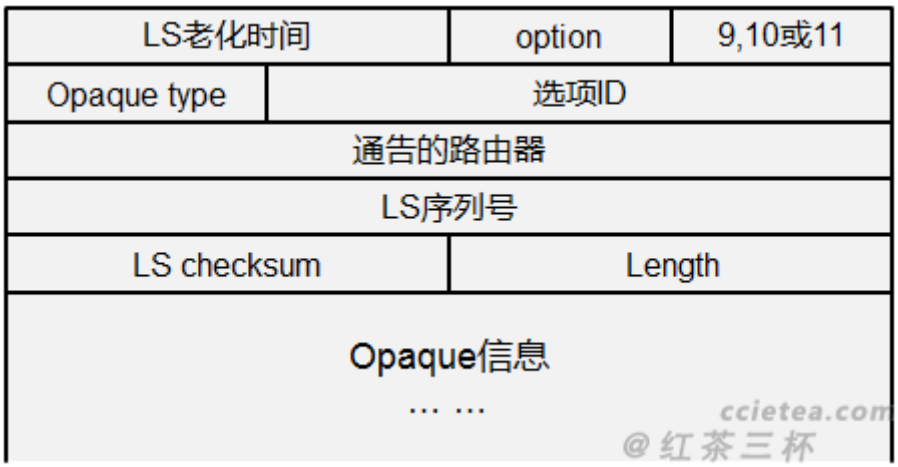
- 在诸如 OSPF 这类 IGP 协议中，使用所配置的带宽或接口 cost 来计算最优路由
- MPLS TE 会将网络资源的实际使用情况纳入考虑，因此它需要更多的信息而不仅仅只是接口带宽或 cost。
- 由 tunnel 的头端进行 CSPF 计算，计算的结果是一串 IP（tunnel 的路径）
- 上述输出的结果供扩展的 RSVP 使用

2. 用于 MPLS TE 的 LSA

RFC2370 描述了 OSPF 的一种扩展 ,定义了 3 种新的 LSA。这些 LSA 称为**迟钝的 LSA(opaque LSA)**，他们的区别仅限于传播范围。这些 LSA 可以准确的将 MPLS TE 所需的信息提供给 OSPF：

- **类型 9　　泛洪范围只在本链路**
- **类型 10　　泛洪范围只在本区域**
- **类型 11　　泛洪范围是整个域（不会进入 stub 区域）**

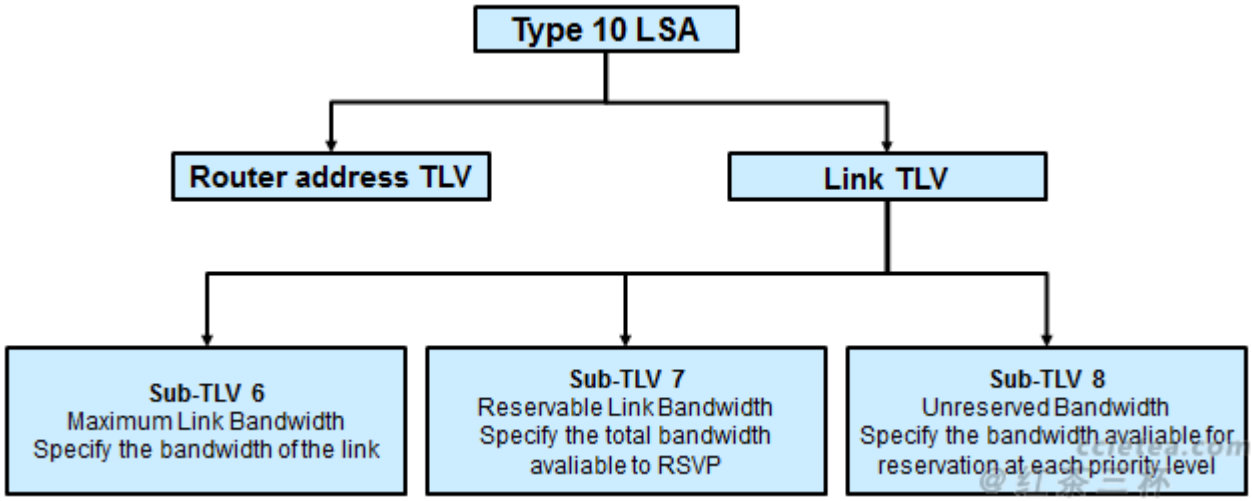
MPLS TE 使用类型 10 的 LSA 来为区域内的 MPLS TE 工作。



```
LS Type: Opaque LSA, Area-local scope
  LS Age: 2 seconds                                  type 10 Opaque LSA
  Do Not Age: False
⊞ Options: 0x20 (DC)
  Link-State Advertisement Type: Opaque LSA, Area-local scope (10)
  Link State ID Opaque Type: Traffic Engineering LSA (1)
  Link State ID TE-LSA Reserved: 0
  Link State ID TE-LSA Instance: 0
  Advertising Router: 1.1.1.1 (1.1.1.1)
  LS Sequence Number: 0x8000000a
  LS Checksum: 0x5df4
  Length: 132
⊟ MPLS Traffic Engineering LSA                       Opaque信息
  ⊟ Router Address: 1.1.1.1
      TLV Type: 1 - Router Address
      TLV Length: 4
      MPLS/TE Router ID: 1.1.1.1 (1.1.1.1)
  ⊟ Link Information
      TLV Type: 2 - Link Information
      TLV Length: 100
    ⊞ Link Type: 2 - Multi-access
    ⊞ Link ID: 10.1.12.2
    ⊞ Local Interface IP Address
    ⊞ Traffic Engineering Metric: 1
    ⊞ Maximum Bandwidth: 12500000 bytes/s (100000000 bits/s)
    ⊞ Maximum Reservable Bandwidth: 9375000 bytes/s (75000000 bits/s)
    ⊞ Unreserved Bandwidth
    ⊞ Resource Class/Color: 0x00000000
    ⊞ Unknown Link sub-TLV: 32770
```

许多TLV，使得OSPF可以 通过一种伸缩性的方式运载数据

可以看到 Opaque 信息这里，有许多 TLV（类型长度值），这些 TLV 使得 OSPF 可以通过一种伸缩性的方式运载数据。这些 TLV 承载了特定的 MPLS TE 数据。一共有两种 TLV 分别是： **路由器地址 TLV** 和 **link TLV**（在上面抓包的结果中的 router address 和 link information 分别就是）。



路由器地址 TLV 承载了用于 TE 的路由器 TE router ID。

而链路 TLV 则承载了一系列描述用于 MPLS TE 的单条链路的子 TLV，关于这些子 TLV 描述如下：

子TLV编号	名称	以8bits为单位的长度
1	链路类型	1
2	链路ID	4
3	本地接口IP地址	4
4	远程接口IP地址	4
5	流量工程度量值	4
6	最大带宽	4
7	最大预留带宽	4
8	未保留的带宽	32
9	Resource class/color	4

- 链路类型说明了链路是点到点还是多路访问链路
- 链路 ID 可以是邻居的 routerID，如果是多路访问链路，则为 DR 的接口地址
- 带宽参数是以 bytes 为单位
- 未保留带宽（不可预留带宽）参数有 32 个 8bits，其他的带宽参数只有 4 个 8bits，因为未保留带宽实际上是 8 个优先级每个优先级别又有 4 个字节。优先级的范围是 0-7。

3. 如何在 CISOC IOS 路由器上查看 opaque LSA-10 呢？

R1#show ip ospf database opaque-area self-originate

```
OSPF Router with ID (1.1.1.1) (Process ID 1)
                Type-10 Opaque Link Area Link States (Area 0)
  LS age: 29
  Options: (No TOS-capability, DC)
  LS Type: Opaque Area Link
  Link State ID: 1.0.0.0
  Opaque Type: 1
  Opaque ID: 0
  Advertising Router: 1.1.1.1
  LS Seq Number: 80000001
```

```
Checksum: 0x6FEB
Length: 132
Fragment number : 0

  MPLS TE router ID : 1.1.1.1

  Link connected to Broadcast network
    Link ID : 10.1.12.2
    Interface Address : 10.1.12.1
    Admin Metric : 1                          !! TE metric，默认和下面的 IGP metric 相等
    Maximum bandwidth : 12500000              !!单位是 Bytes，乘以 8 也就是 100Mbps
    Maximum reservable bandwidth : 9375000    !!最大可预留带宽是 75M，也就是 100M 的 75%
    Number of Priority : 8
    Priority 0 : 9375000      Priority 1 : 9375000
    Priority 2 : 9375000      Priority 3 : 9375000
    Priority 4 : 9375000      Priority 5 : 9375000
    Priority 6 : 9375000      Priority 7 : 9375000
    Affinity Bit : 0x0
    IGP Metric : 1                            !!接口的 OSPF cost

  Number of Links : 1
```

4. O 比特

DN	O	DC	L	N/P	MC	E	●

O 比特被定义使用在 OSPF 的 option 字段中，用来说明路由器是否有能力发送和接收 opaque LSA。

当我们部署 MPLS TE 且使用 OSPF 作为 TE 的时候，OSPF 就需要扩展以便支持 MPLS TE，这时候就可以看到 Obit 位的置位了，而且只能够在这种环境下的 DBD 报文中看到相应的 option 的 obit 置位，而其他报文中的 option 里 obit 仍然为 0。

在 RFC5250 中对这里做了描述：A neighbor is opaque-capable if and only if it sets the O-bit in the Options field of its Database Description packets; the O-bit SHOULD NOT be set and MUST be ignored when

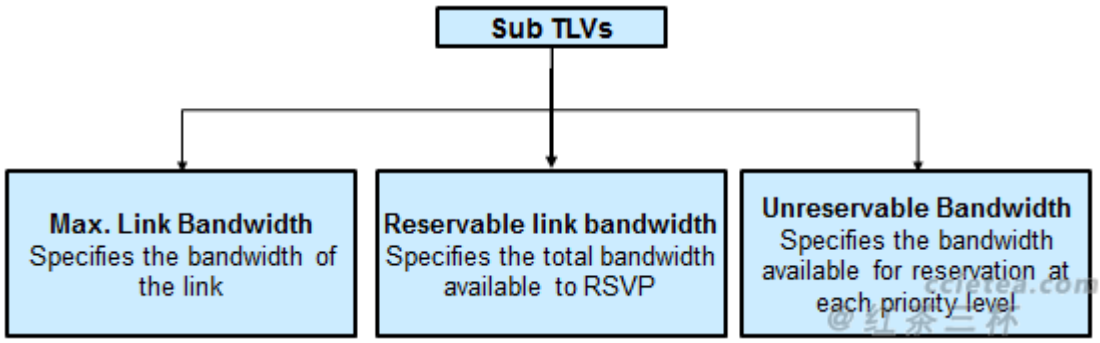
received in packets other than Database Description packets。

2.2.2 针对 TE 的 ISIS 扩展

1. **针对 TS 的扩展 IS-IS 新增了三种类型的 TLS：**
   - **TLV 22：Extended intermediate-System Reachability**

     Ability to send detail link information , including bandwidth . Replaces IS Neighbor TLV

     TLV 22 可以包含 7 个子 TLVs，其中三个包含用于 TE 的带宽信息



   - **TLV134：TE RouterID**

     设备的 TE RouterID
   - **TLV135：Extended IP Rechability**

     Replaces both IP Reachability TLVs（128 and 130） . Uses wide metrics

2. **查看报文**

```
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
  Intra Domain Routing Protocol Discriminator: ISIS (0x83)
  PDU Header Length: 27
  Version (==1): 1
  System ID Length: 0
  PDU Type           : L1 LSP (R:000)
  Version2 (==1): 1
  Reserved (==0): 0
  Max.AREAs: (0==3): 0
⊟ ISO 10589 ISIS Link State Protocol Data Unit
    PDU length: 147
    Remaining lifetime: 1197
    LSP-ID: 0000.0000.0001.00-00
    Sequence number: 0x00000008
  ⊞ Checksum: 0x916f [correct]
  ⊞ Type block(0x01): Partition Repair:0, Attached bits:0, Overload bit:0, IS type:1
  ⊞ Area address(es) (4)
  ⊞ Protocols supported (1)
  ⊞ Hostname (2)
  ⊟ Traffic Engineering Router ID (4) TLV134
      Traffic Engineering Router ID: 1.1.1.1 (1.1.1.1)
  ⊞ IP Interface address(es) (4)
  ⊟ Extended IP Reachability (17)    TLV135
    ⊞ IPv4 prefix: 10.1.12.0/24, Metric: 10, Distribution: up, no sub-TLVs present
    ⊞ IPv4 prefix: 1.1.1.1/32, Metric: 10, Distribution: up, no sub-TLVs present
  ⊟ Extended IS reachability (74) TLV22
    ⊟ IS neighbor: 0000.0000.0002.01
        Metric: 10
        Administrative group(s):
        IPv4 interface address: 10.1.12.1
        Maximum link bandwidth : 100.00 Mbps
        Reservable link bandwidth: 75.00 Mbps
      ⊞ Unreserved bandwidth:
        Traffic engineering default metric: 1
```

2.2.3 IGP 泛洪

IGP 在下列情况发生时泛洪 TE 信息：

- 链路状态发生变化
- 配置变化
- 周期性泛洪
- 预留带宽发生变化（重大变化）
- 在隧道设置失败之后

下面将分别对上述的情况做分析：

1. **和常规的 IP 路由一样，接口的 up down 或者手工修改了针对 IGP 的接口参数的时候， OSPF 将泛洪 LSA 或者 ISIS 会泛洪 LSP。**
2. **OSPF 和 ISIS 也有周期性泛洪的机制**

   OSPF 默认的泛洪周期是 30 分钟，可以通过 timers pacing lsa-group 修改

   ISIS 默认的泛洪周期是 15 分钟，可以通过 lsp-refresh-interval 修改
3. **带宽的变化**

   默认的变化阀值（带宽占用率）:

   Up Thresholds 15 30 45 60 75 80 85 90 95 96 97 98 99 100

   Down Thresholds: 100 99 98 97 96 95 90 85 80 75 60 45 30 15

```
router(config)#interface fast0/0
router(config-if)#mpls traffic-eng flooding thresholds up/down ?
```

   修改带宽 UP/DOWN 阀值

```
router# show mpls traffic-eng link-management bandwidth-allocation
```

   查看带宽的 Thresholds

在通过 IGP 协议（OSPF 或 ISIS）泛洪相关信息后，所有的 TE 路由器都形成区域内统一的 TEDB

TEDB 与 IGP 的链路状态数据库是独立的。

2.3 信息详解

2.3.1 带宽信息

MPLS TE 需要通告的带宽信息有：

- 最大接口可用带宽：

  可使用默认或者用这条命令修改 Router(config-if)# bandwidth ?
- 最大可预留带宽信息：

Router(config-if)# ip rsvp bandwidth ?

这个带宽是给 TE tunnel 预留的，如果命令后面不加带宽值，则缺省为接口带宽的 75%

2.3.2 隧道优先级

1. **我们可以定义 TE tunnel 的优先级**
   - 优先级范围 0-7，越小越优先
   - 隧道优先级有两种类型：
     - 建立优先级 setup priority
     - 保持优先级 hold priority
   - 高优先级的隧道可以抢占低优先级隧道的资源。支持 0-7 共 8 个隧道优先级别
   - 设置命令：

     ```
     router(config-if)#tunnel mpls traffic-eng priority 6 6
     ```

     第一个 6 是 setup priority，第二个 6 是 hold priority。

     如果一个 tunnel1 的建立优先级，比 tunnel2 的保持优先级高，那么 tunnel1 就能抢占 tunnel2

     不能将 tunnel 的建立优先级设置得比保持优先级高。

     当然，我们在部署的时候，强烈建议同一个隧道的建立优先级和保持优先级要相等

2. **每个优先级都对应一个可用带宽**

   支持 0-7 共 8 个隧道优先级别

   接口下根据各自优先级会列出当前的最大可用带宽

   Show mpls traffic-eng topology ？

   Show mpls traffic-eng link-management advertisement

   Show mpls traffic-eng link-management bandwdith-allocation

3. **关于隧道的优先级及当前可用带宽的验证：**

   首先 tunnel 的配置如下：

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel destination 4.4.4.4
 tunnel mode mpls traffic-eng
```

```
tunnel mpls traffic-eng priority 6 6
tunnel mpls traffic-eng path-option 10 dynamic
```

假设本地物理出接口的带宽为 100M，那么 rsvp 默认的可预留带宽是 75%，这里就是 75M

所以我们得到如下的查看信息：

R1#sh mpls traffic-eng topology 1.1.1.1 (1.1.1.1 是本地路由器)

```
IGP Id: 1.1.1.1, MPLS TE Id:1.1.1.1 Router Node    id 1
        link[0 ]:Nbr IGP Id: 10.1.12.2, nbr_node_id:8, gen:47
            frag_id 0, Intf Address:10.1.12.1
            TE metric:1, IGP metric:1, attribute_flags:0x0
            physical_bw: 100000 (kbps), max_reservable_bw_global: 75000 (kbps)
            max_reservable_bw_sub: 0 (kbps)

                                    Global Pool          Sub Pool
                Total Allocated     Reservable           Reservable
                BW (kbps)           BW (kbps)            BW (kbps)
                ---------------     -----------          ----------
        bw[0]:      0               75000                  0
        bw[1]:      0               75000                  0
        bw[2]:      0               75000                  0
        bw[3]:      0               75000                  0
        bw[4]:      0               75000                  0
        bw[5]:      0               75000                  0
        bw[6]:      0               75000                  0
        bw[7]:      0               75000                  0
```

上面我们可以看到所有隧道优先级的可预留带宽。

现在我们来修改优先级为 6 的隧道的可预留带宽。

```
Interface Tunnel0
 ip unnumbered Loopback0
 tunnel destination 4.4.4.4
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 6 6
 tunnel mpls traffic-eng bandwidth    33000        !!改为 33M
```

```
 tunnel mpls traffic-eng path-option 10 dynamic
```

R1#sh ip rsvp interface

```
interface       allocated   i/f max   flow     max sub max
Fa0/0             33M         75M     75M      0              !! 可以看到，这里 f0/0 口已经被分掉了 33M
```

R1#sh mpls traffic-eng topology 1.1.1.1

```
IGP Id: 1.1.1.1, MPLS TE Id:1.1.1.1 Router Node    id 1
        link[0 ]:Nbr IGP Id: 10.1.12.2, nbr_node_id:8, gen:51
            frag_id 0, Intf Address:10.1.12.1
            TE metric:1, IGP metric:1, attribute_flags:0x0
            physical_bw: 100000 (kbps), max_reservable_bw_global: 75000 (kbps)
            max_reservable_bw_sub: 0 (kbps)

                                    Global Pool          Sub Pool
                Total Allocated     Reservable           Reservable
                BW (kbps)           BW (kbps)            BW (kbps)
                ---------------     -----------          ----------
        bw[0]:          0           75000                  0
        bw[1]:          0           75000                  0
        bw[2]:          0           75000                  0
        bw[3]:          0           75000                  0
        bw[4]:          0           75000                  0
        bw[5]:          0           75000                  0
        bw[6]:      33000           42000                  0
        bw[7]:          0           42000                  0
```

可以看到，我们为优先级为 6 的隧道分配了 33M 的带宽，原先接口的总可预留带宽为 75M，分了 33M 出去，所以还有 42M，我们也可以看到，因为我们修改了优先级为 6 的隧道的可预留带宽，因此优先级比其低的隧道，可预留的带宽也变了。

2.3.3 属性标记和亲和属性

1. 概念简介

- **属性标记 attribute-flags**

  用来描述物理链路的属性，由 32bits 组成，每一位都可以单独表示链路的一个属性（如是否加密等）或者管理型的策略。这 32bits 没有特定的语法，每一个 bit 都可以进行设置后者不去动他，网络设计者可以认为的根据需要为特定的 bit 指定特定的意义。

  在物理接口上配置：

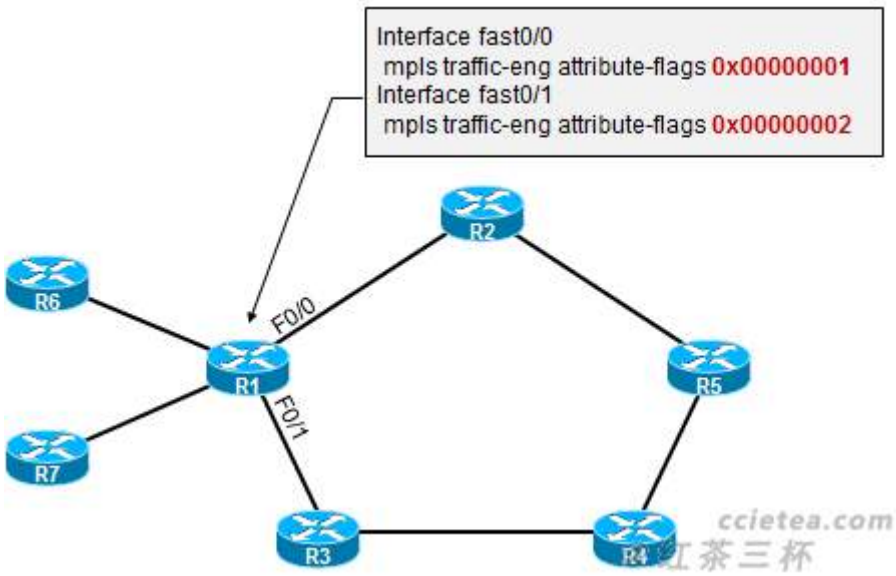
```
Router(config-if)# mpls traffic-eng attribute-flags ?
<0x0-0xFFFFFFFF>    Attribute flags
```

- **亲和属性 Affinity Attributes 及掩码**

  亲和属性是描述 MPLS TE tunnel 的属性，也由 32bits 组成，正好与管理上面所说的 attribute-flags 位数相同。与亲和属性同时被配置的还有一个掩码，也是 32bits，用来给属性标记和亲和属性做匹配动作。

  默认是　0x0 / 0xFFFF（默认掩码是全 1，所以所有的 bits 都无所谓）

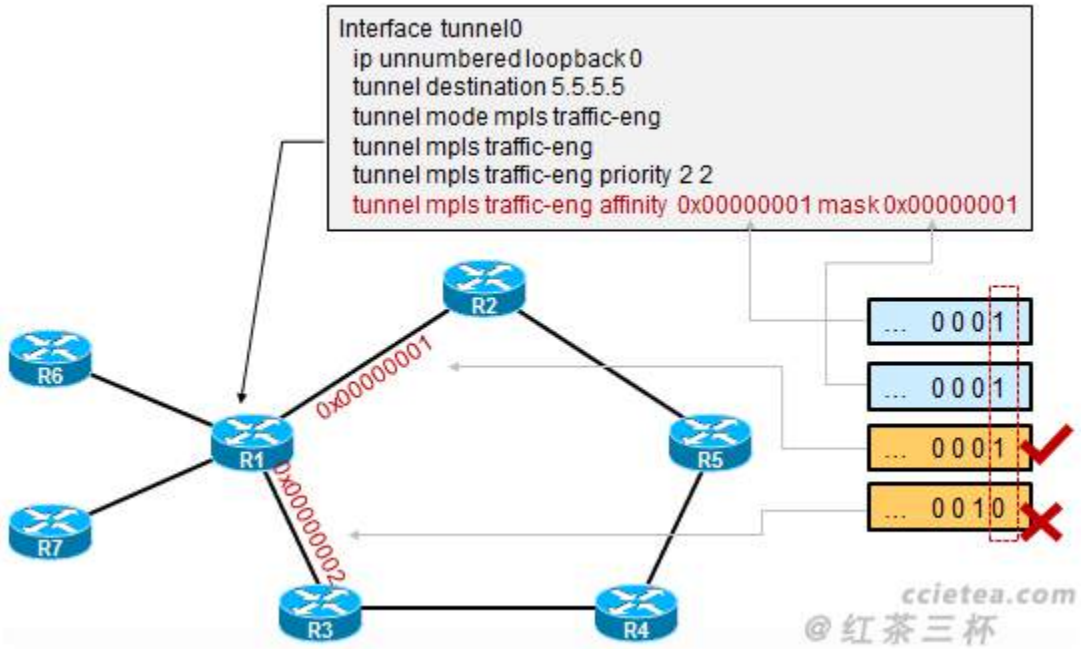
2. 概念详解



上图中，我们给 R1 的两个接口分别定义了属性标记，属性标记 attribute-flags 是一个 32bits 的字段，用 16 进制形式输入和体现。使用的命令如图中所示，命令输入后，将触发 R1 发送 Opaque LSA，并且通告其链路的这个管理属性，LSA 相关内容如下（注意，下面的报文截图与上述网络环境无关，仅仅为解释 attribute-flags 在报文中的体现）:

```
MPLS Traffic Engineering LSA
  Link Information
      TLV Type: 2 - Link Information
      TLV Length: 100
    Link Type: 2 - Multi-access
    Link ID: 10.1.23.2
    Local Interface IP Address
    Traffic Engineering Metric: 1
    Maximum Bandwidth: 12500000 bytes/s (100000000 bits/s)
    Maximum Reservable Bandwidth: 9375000 bytes/s (75000000 bits/s)
    Unreserved Bandwidth
    Resource Class/Color: 0x00000002
      TLV Type: 9: Resource Class/Color
      TLV Length: 4
      MPLS/TE Link Resource Class/Color: 0x00000002
        Group 1
    Unknown Link sub-TLV: 32770
```


接下去，我们在 R1 的 tunnel 中做如下配置：



我们定义了 R1 这个 TE tunnel 的亲和属性及掩码，那么这样一来，R1 在为这条 tunnel 计算路径的时候，R1 将 affinity 值 0x00000001 转换成二进制，与 R1 两个接口上配置好的这两个 attribute-flags 值进行一一比对，由于我们掩码配置的是 0x00000001，将其也写成二进制，掩码为 1 的位，就是必须匹配的，因此仅有 R2-R2 这段链路可用。在实际的网络部署中，我们可以根据自己的需求灵活的定义链路属性每一个 bit 的含义，结合亲和属性就可以进行路径选路的把控和规避等等。

2.3.4 TE Metric：Administrative weight

1. 概念简介

- **缺省情况下用 TE metric 作为 TE tunnel 最短路径的计算依据**

```
router(config-if)#mpls traffic-eng administrative-weight ?
```

  - 可以用上述命令修改 TE metric（在物理接口上配置）
  - 缺省情况下 TE metric 等于 IGP metric（这句话的意思是，在没有使用上面的命令在物理接口中配置的情况下，TE metric 等于 IGP metric）

- **可以修改作为 TE tunnel 最短路径的计算依据**

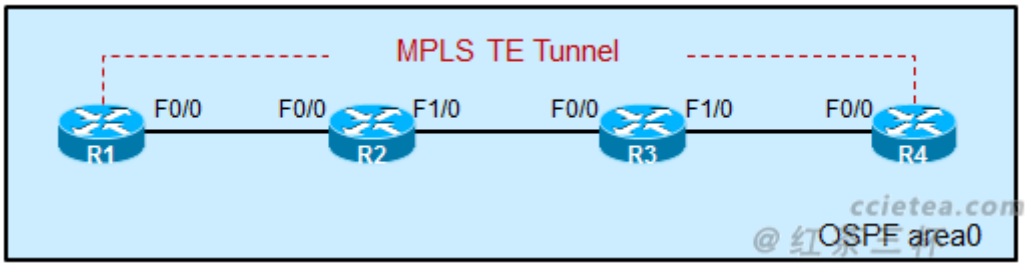
```
router(config)#tunnel mpls traffic-eng path-selection metric ?
    igp   use IGP metric
    te    use TE metric       缺省是这个

上述命令，也可在 tunnel 口中配置
```

如果将 Path-selection metric 改为 igp，那么 tunnel 的最短路径将忽略物理接口中配置的 administrator-weight，而只采用 IGP metric 来计算 tunnel 的最短路径。

2. 验证及查看

```
router#show mpls traffic-eng topology
router#show mpls traffic-eng tunnels
```

使用上述命令查看 TE metric



例如在上述拓扑中，R1 建立了一条到 R4 的 TE tunnel，所有路由器都宣告自己的 loopback 进 OSPF。所有的接口 cost 都为 1。

R1#show mpls traffic-eng tunnels

```
Name: R1_t0                                    (Tunnel0) Destination: 4.4.4.4
  Status:
    Admin: up          Oper: up      Path: valid        Signalling: connected
    path option 10, type dynamic (Basis for Setup, path weight 3)

  Config Parameters:
    Bandwidth: 33000      kbps (Global)  Priority: 6  6    Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:   disabled  LockDown: disabled  Loadshare: 33000      bw-based
    auto-bw: disabled

  InLabel  :  -
  OutLabel : FastEthernet0/0, 200
  RSVP Signalling Info:
       Src 1.1.1.1, Dst 4.4.4.4, Tun_Id 0, Tun_Instance 163
    RSVP Path Info:
      My Address: 10.1.12.1
      Explicit Route: 10.1.12.2   10.1.23.2   10.1.23.3   10.1.34.3
                      10.1.34.   4 4.4.4.4
      Record Route:   NONE
      … … … …
```

可以看到 Path weight=3，这个 3 是怎么来的呢？因为 R1 向 R4 建立 tunnel，那么 R1 到达 R4（注意不是到 R4 的 loopback 口）的 OSPF metric=3，所以默认情况下 TE metric=3

当然，我们也可以修改 TE metric，例如在 R2 的 F1/0 口上：

```
R2(config-if)#mpls traffic-eng administrative-weight 100
```

那么 R1 上再去看 tunnel，他的 TE metric 就成了 102 了。

那么如果在 R1 上的 tunnel 口中，使用如下配置：

```
router(config)#tunnel mpls traffic-eng path-selection metric igp
```

那么 R1 将使用 IGP 的 metric ,而忽略 R2 上物理接口配置的 adminnitration-weight ,因此上面的命令生效后，再去 R1 上查看 tunnel 的 path weight，就又变成了 3。

3 路径计算

3.1 CSPF 算法概述

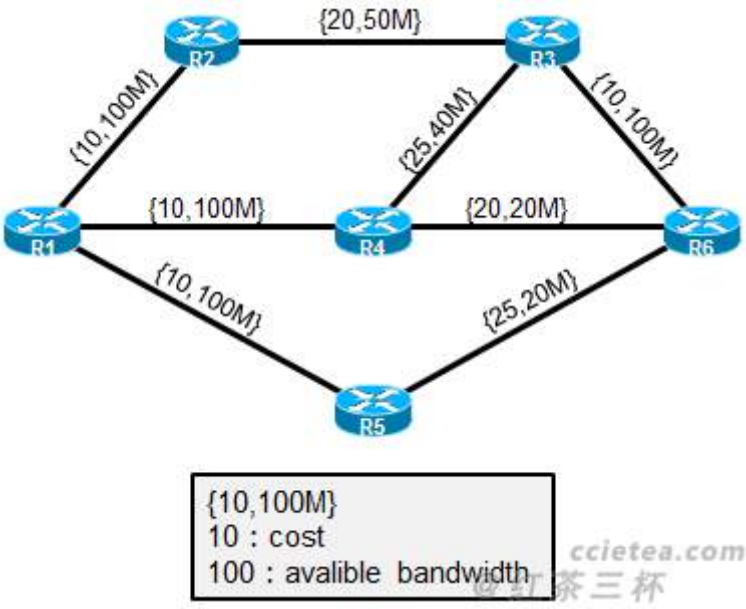
CSPF（Constrained Shortest Path First）有约束条件的 SPF 算法

约束条件：

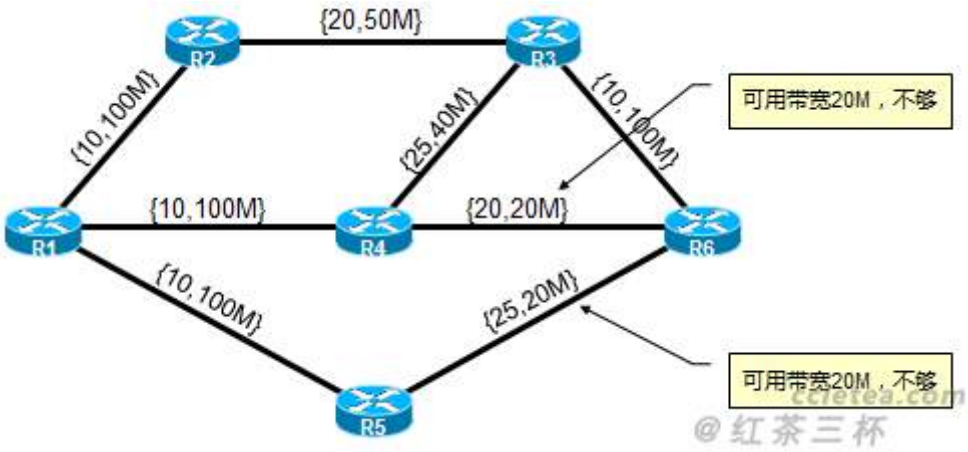
- TE Metric
- 可用带宽
- 链路属性

CSPF 算法的输出是在 tunnel 的两个端点之间的一个 IP 接口地址序列（下一跳路由器地址）。

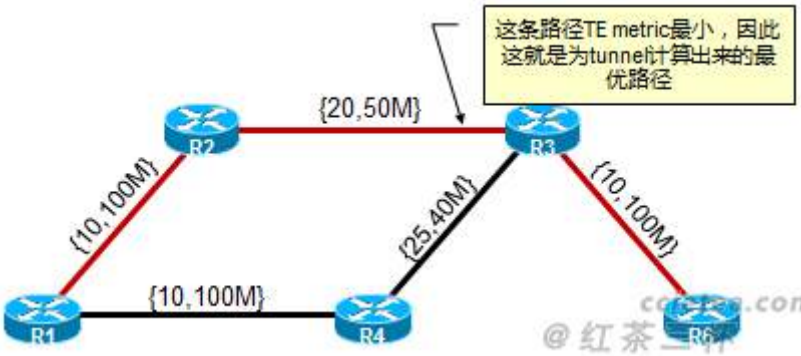
3.2 CSPF 算法工作机制



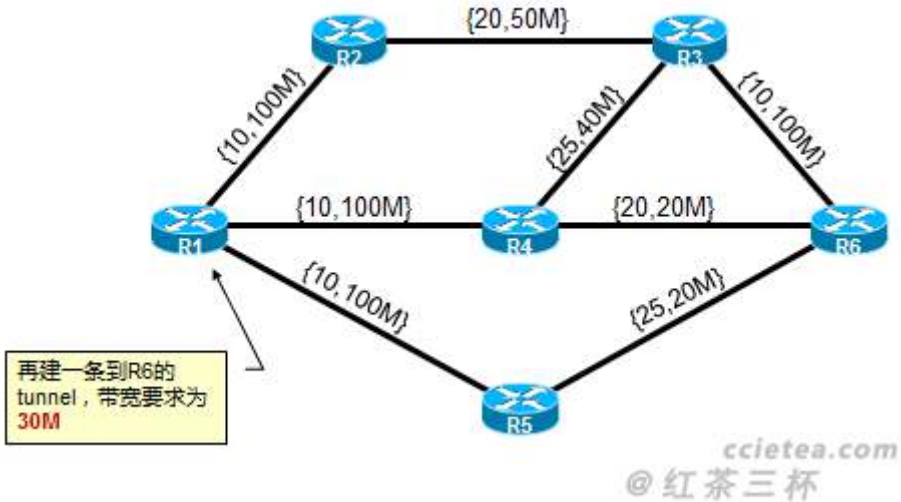
我们看上图，从 R1 到 R6 如果要建立一条 TE tunnel，也就是最优的路径，且带宽要求为 30Mbps，那么计算过程是怎样的呢？首先注意，得益于“针对 MPLS TE 扩展的 OSPF 或 IS-IS”在区域中的链路状态信息的泛洪，R1 此刻已经完成了自己 TEDB 的信息搜集，接下去 tunnel 路径的计算过程在 R1 上完成的。



R4-R6、R5-R6 之间的链路，带宽只有 20M，显然不符合要求。因此 R1 只会考虑如下链路：

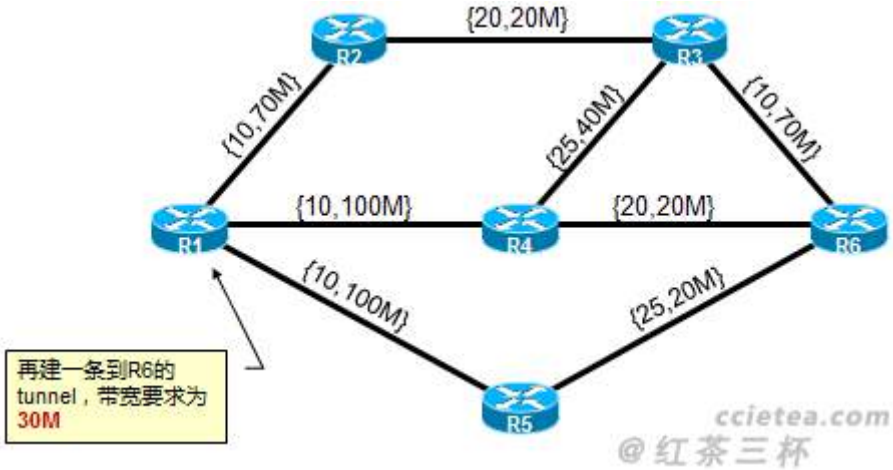


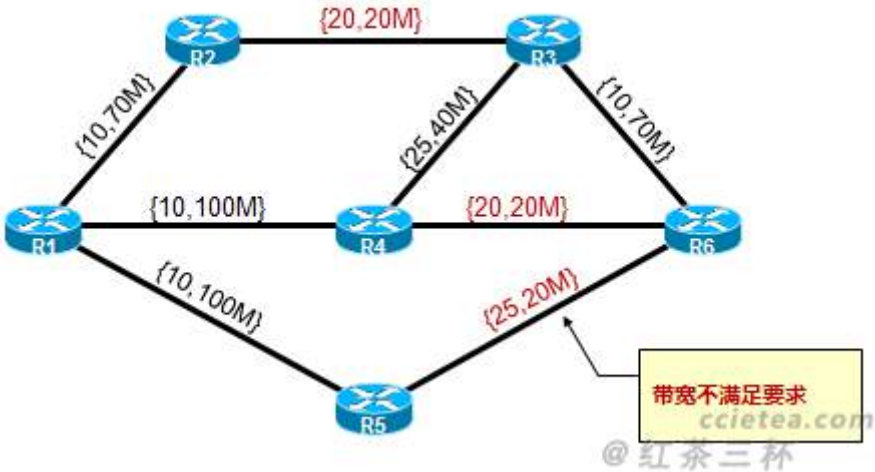
- R1 在剩余的两条路径的基础上，再去比较 TE metric（默认等于 IGP Metric）
- 最终选择上面的这条红色的路径，因为这条路径的 TE metric 更小



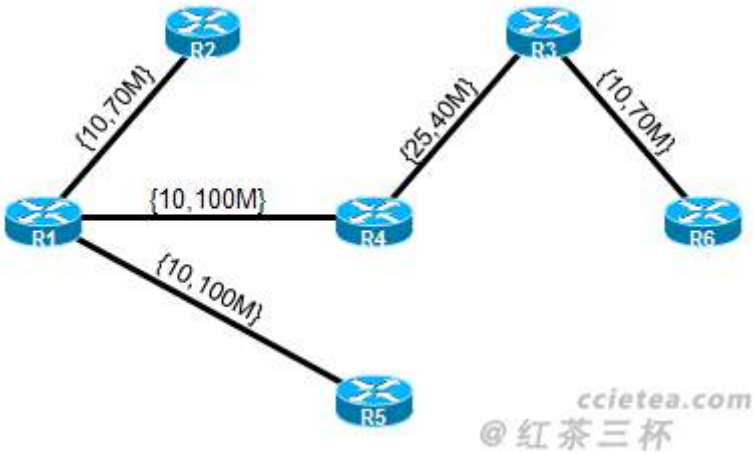
现在，我们要再建立一条 R1 到 R6 的 TE tunnel，带宽要求为 30M，注意，此刻网络中的可用带宽由于上面

那条建立好的 tunnel 的存在已经发生了变化，此时此刻的网络环境如下：

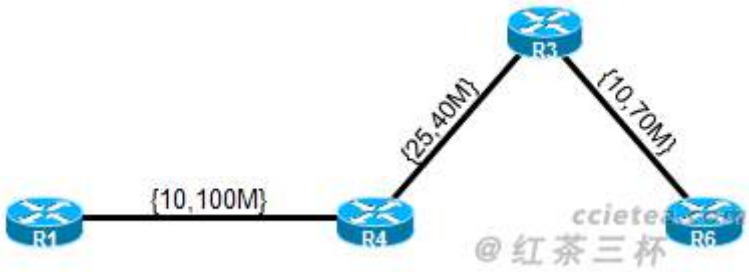




- 带宽不满足要求的链路不考虑。那么经过筛选剩下如下链路：



- 最后，第二条 TE tunnel 的路径如下：



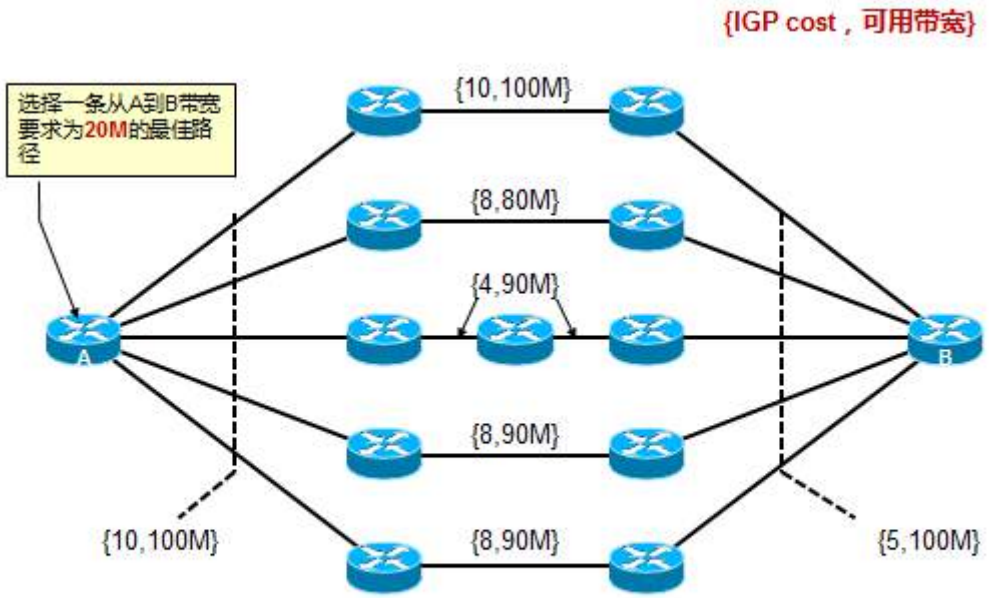
3.3 CSPF 最高仲裁

- 在标准的 SPF 算法中，到达同一个目的地可以有多条等价的路径存在，我们称之为 ECMP（Equal-cost Multipath）
- 但在 CSPF 算法中，对于一个目的地只能有一条路径。当存在多满足基本条件的路径时（这个这个规则只有在 TE metric、带宽及相关属性等都无法决策的情况下才会进行）：

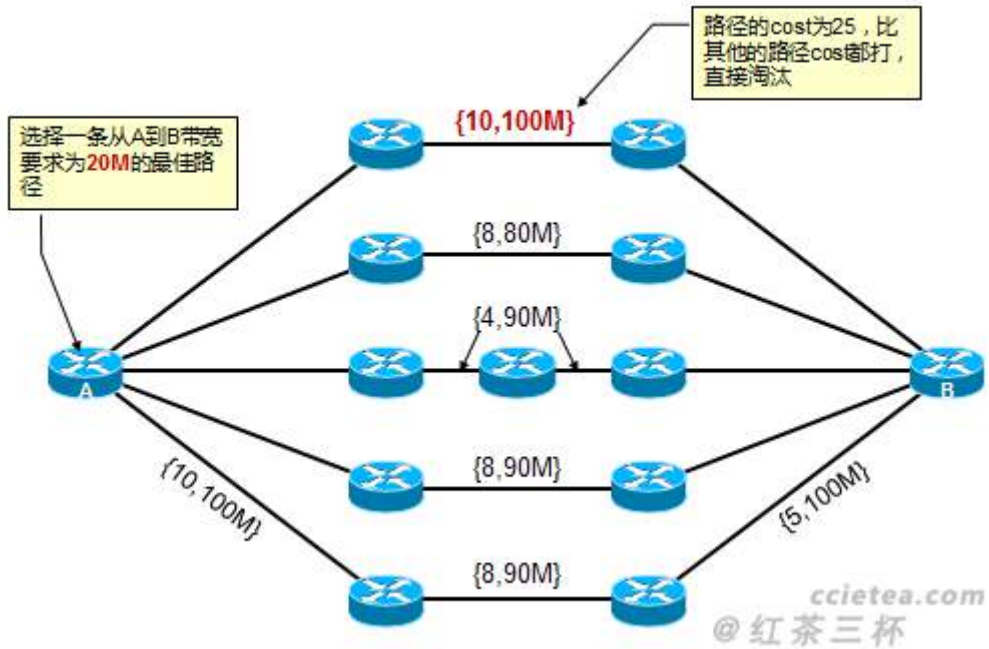
  1. **选择 IGP cost 最小的路径**
  2. **选择有最大的"最小可用带宽"的路径**
  3. **选择最少跳数的路径**
  4. **如果还不能区分，则随机选择一条**

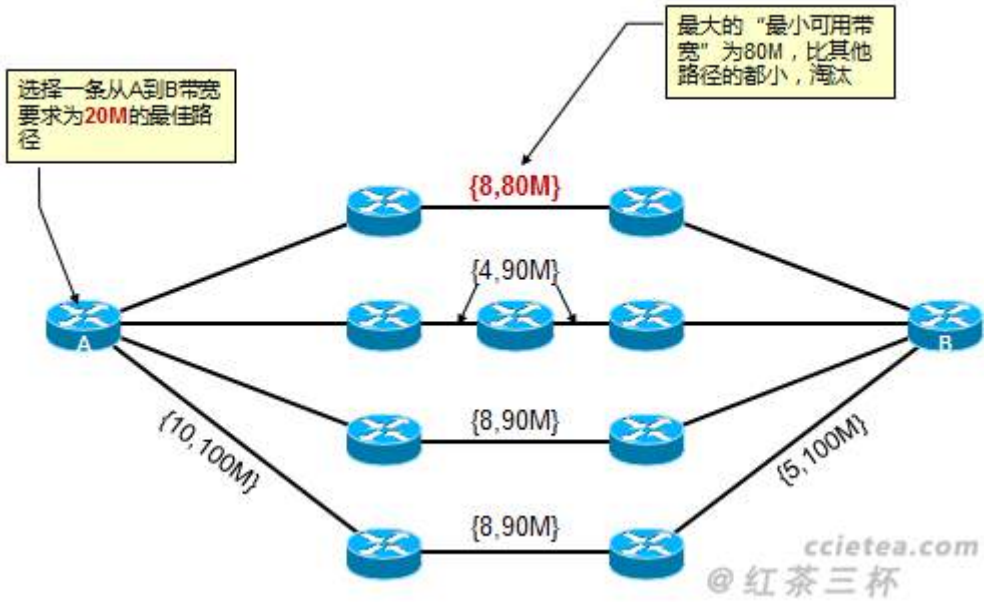
  注意，这个所谓的最高仲裁，只有在多条路径的 TE metric 相等、可用带宽都满足要求，并且没有其他的链路属性的干扰的情况下，才会生效。最高仲裁中的第一条：选择 IGP cost 最小的路径，这个就是 iGP 的 cost，注意和 TE metric 区分，TE metric 在缺省的情况下的确和 IGP cost 一样，当然，我们也可以在物理接口上做修改。

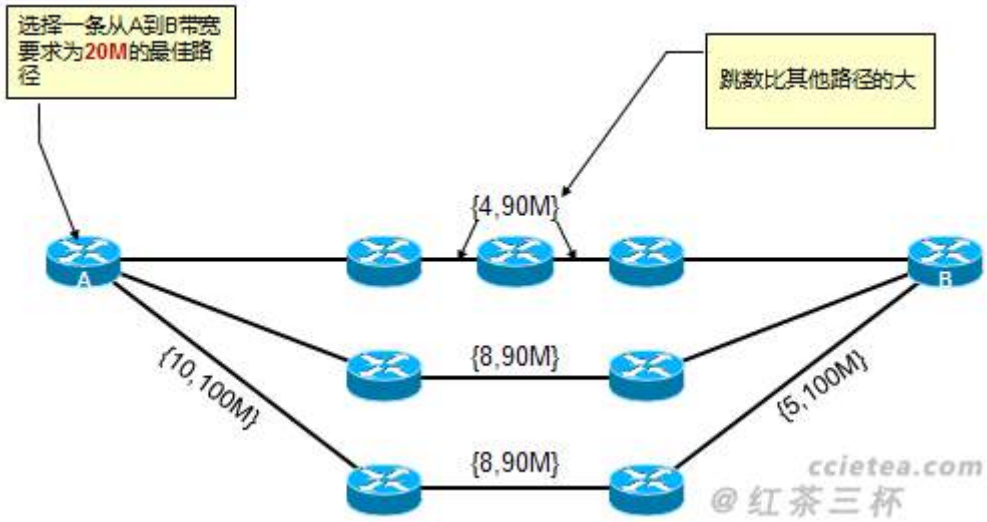
  下面我们来看一个例子：

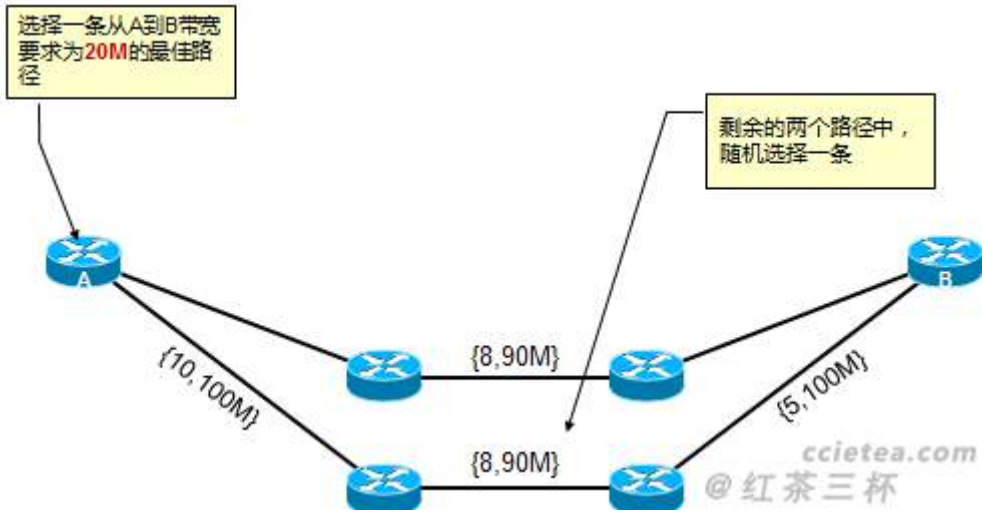


- 注：我们这个环境中所有路径的TE Metric均相等（如果TE metric不相等，就直接淘汰掉部分链路了，而不用进入到最高仲裁）



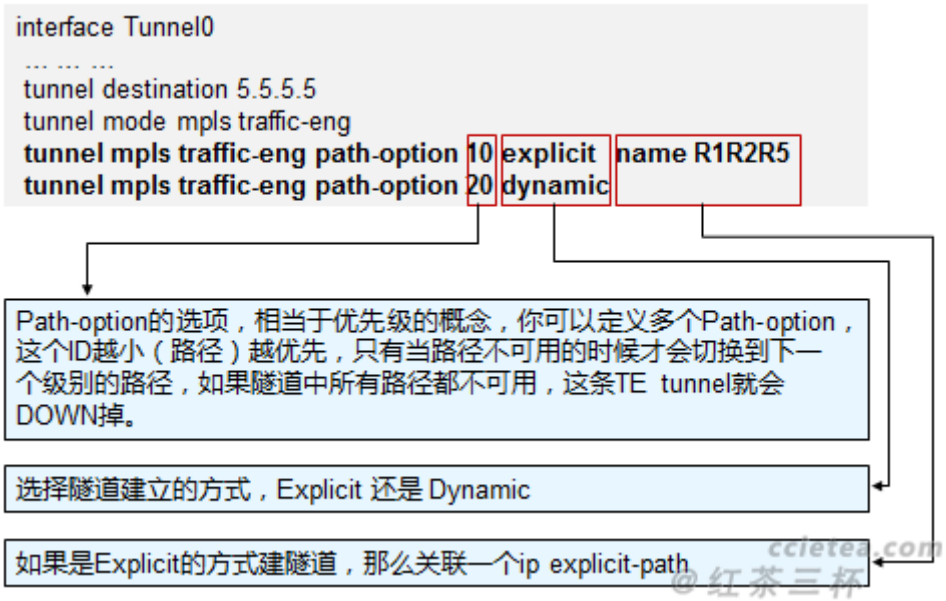

选择一条从A到B带宽要求为20M的最佳路径
最大的“最小可用带宽”为80M，比其他路径的都小，淘汰
{8,80M}
{4,90M}
A
B
{10,100M}
{8,90M}
{5,100M}
{8,90M}
ccietea.com
@红茶三杯


选择一条从A到B带宽要求为20M的最佳路径
跳数比其他路径的大
{4,90M}
A
B
{10,100M}
{8,90M}
{5,100M}
{8,90M}
ccietea.com
@红茶三杯



3.4 路径设置选项 path-option

1. **可以通过两种方式设置 TE tunnel 的路径（Explicit 显式或 Dynamic 动态）**
   - 在动态的设置中，让隧道首端路由器计算出 TE tunnel 能够以最优的方式穿越网络到达尾端的路径。我们只需要配置 tunnel 的 destination，然后首端路由器通过 CSPF 算法自己去算路径。
   - 在显式的设置方式中，我们必须要指定 TE tunnel 需要穿越的路由器，你可以指定链路中路由器的 TE RouterID，或者接口 IP 地址。



2. 关于 Explicit Path

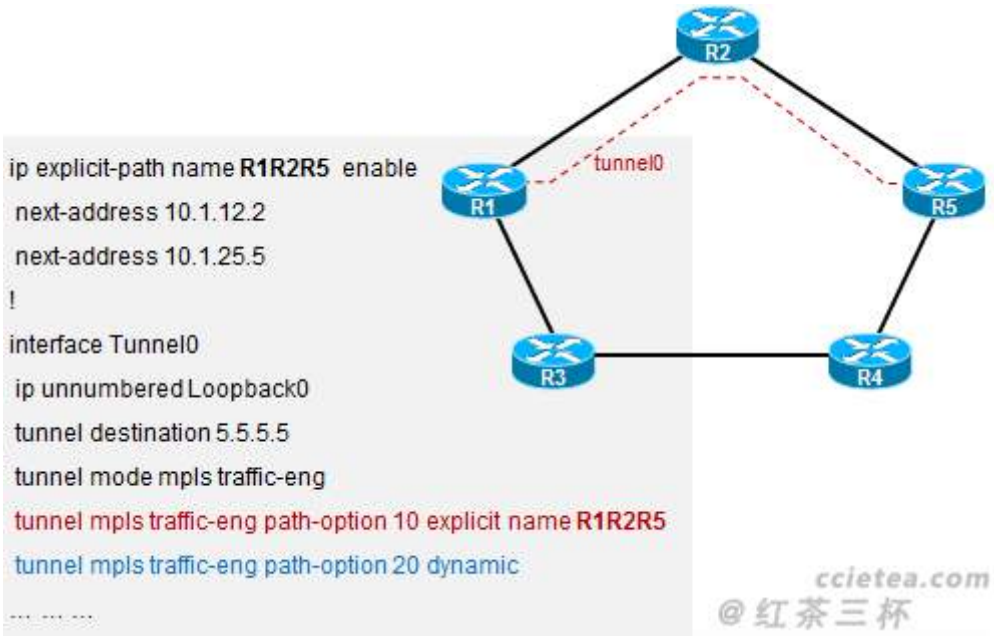
- 当使用 Explicit 方式建立 tunnel 时，需关联一个 explicit path。
- Explicit Path 由一系列 IP 构成，一条 explicit path 上的 IP 可以是接口 IP ,也可以是 MPLS TE 路由器 ID。
- 通过 exclude-address 命令来控制 LSP 不经过某节点（接口或路由器）。可以使用这个关键字搭配某个接口 IP 或者某台 MPLS TE 路由器 ID 从而让 tunnel 路径绕开某段链路或某台路由器。
- 定义一个 Explicit path 的命令如下：

```
Router(config)#ip explicit-path name ccietea
Router(cfg-ip-expl-path)#next-address 10.1.12.2    !!注意输入顺序
Router(cfg-ip-expl-path)#next-address 10.1.25.5
Router(cfg-ip-expl-path)#
!
Router(config)#interface tunnel0
Router(config-if)#tunnel mpls traffic-eng path-option 10 explicit name ccietea    !! 关联到 path-option
```

如上，注意输入的顺序。

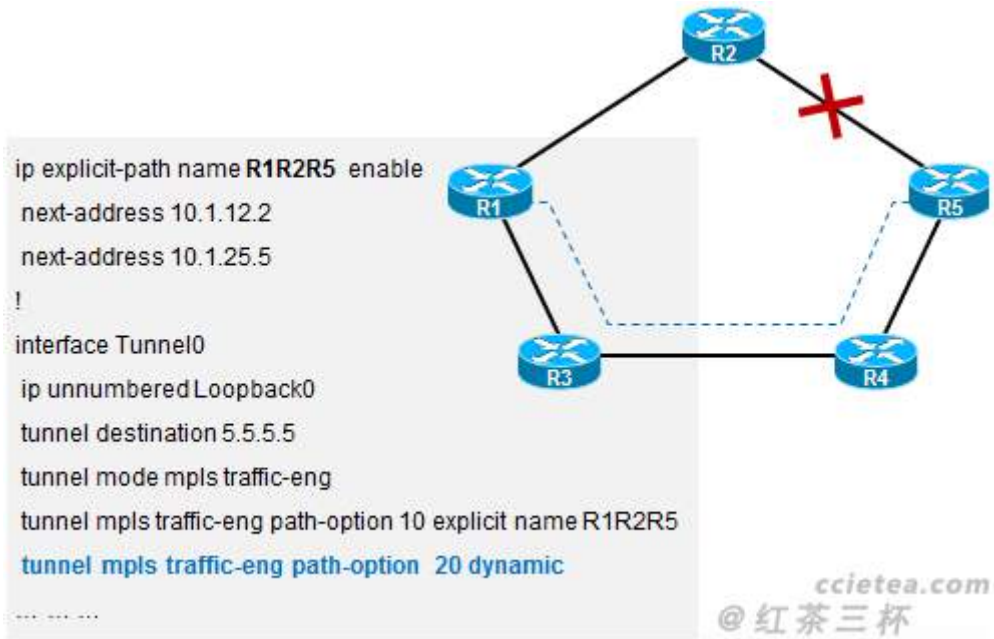
在 cfg-ip-expl-path 模式内，还可以使用 list 关键字查看已经输入好的所有 next-address，使用 index x next-address 指定每一个已经输入的下一跳 IP 的索引编号。

3. 实验验证



看上面这个拓扑，我们要在 R1 及 R5 之间建立一个 TE tunnel，在网络正常情况下，我们希望 tunnel 的路径是 R1-R2-R5，这是一个显式的指定路径。为了让 tunnel 的路径有备份可走，我们再定义一个 path-option，这个路径采用 dynamic 的设置方式，让 CSPF 自己去计算，那么配置如上。

当 R1-R2-R5 这段路径出现问题的时候，例如 R2-R5 之间的链路 DOWN 掉，那么 TE tunnel 会向下一个优先级的 Path-option 切换，也就是 dynamic 设置方式的这条，因此会切换到 R1-R3-R4-R5 上。



3.5 隧道优先级

如果网络中存在多条 Tunnel 为不同的流量服务 ,那么有可能这些 tunnel 的重要程度根据受实际的业务需求影响他们的重要程度也有所不同。那么如果我一跳更重要的 Tunnel 在一条更次要的 tunnel 之后配置的 , 就有可能导致更重要的这条 Tunnel 找不到最优化的、或者足够带宽的路径。我们可以通过**隧道优先级**的方式来规避上面的问题。

- **隧道优先级的范围 0-7，越小越优先**
- **隧道优先级有两种类型：**
  - 设置优先级 setup priority
  - 保持优先级 hold priority

  设置优先级和保持优先级都通过相应的数值来说明一条 TE tunnel 是否可以抢占另一条 TE tunnel。优先级越低重要程度就越高。

  设置优先级说明某条 tunnel 的重要程度以至于可以抢占其他 tunnel；

  保持优先级表明了某条 tunnel 的权重是多少，以便它可以在链路中被保留下来；

  如果 tunnel 0 的设置优先级比 tunnel 1 的保持优先级低，那么 tunnel0 可以抢占 tunnel1。
- **设置命令**

  router(config-if)#tunnel mpls traffic-eng priority *建立优先级 保持优先级*

  注意：不能将 tunnel 的建立优先级设置得比保持优先级高

4 RSVP

4.1 协议概述

1. **相关 RFCs**

   RFC 2205 Resource Reservation Protocol(RSVP)

   RFC 2210 The Use of RSVP with IETF Intergrated Services

   RFC3209 RSVP 对 TE 的扩展

2. **基本概念**

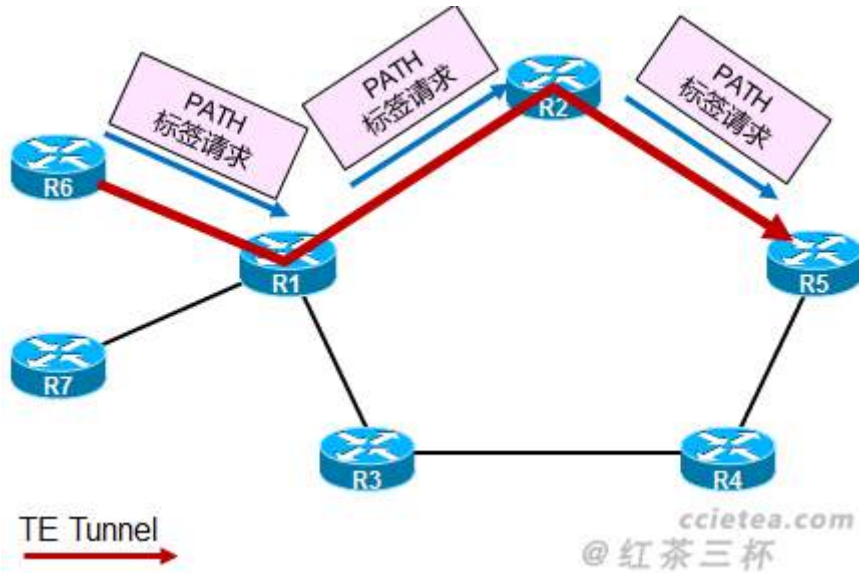
在 TE 隧道中，需要一种信令协议来确保可以使用 TE 隧道所穿越的 LSR 接口的链路，并且可以逐跳地传播 TE 隧道中的流量所需要的标签。

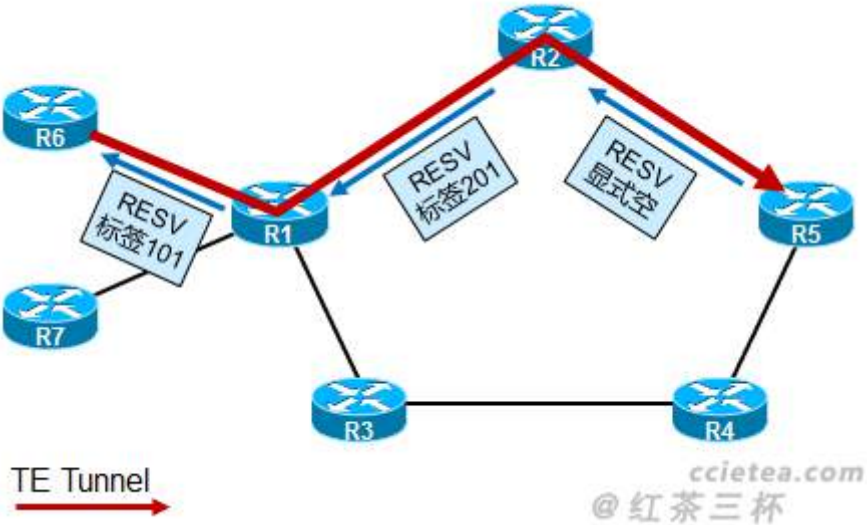
- TE 隧道的首端路由器根据 TE 数据库计算出 TE 隧道的最优路径，参考带宽和限制条件。另外最优路径还可以通过用户配置于隧道接口上的一个显式路径选项来定义。无论哪一种情况，首端路由器都会了解到 TE 隧道所使用的具体路径。
- RSVP 使用 PATH 和 RESV 消息来在一条路径中传递信令。TE 首端路由器发送 PATH 消息到尾端路由器，PATH 消息被一跳一跳的发送到尾端；而 RESV 消息则是通过相反的方向被发送给首端路由器。
- TE 隧道需要穿越的每一跳 LSR 都会被加入一个 ERO。这是一个接口 IP 地址的顺序列表，每一台 LSR 一个 IP 地址。
- PATH 消息是从首端路由器向后面的路由器进行传递的，那么下一跳路由器将从 ERO 中移除自己的 IP 地址，看到下一个 IP 地址是什么之后再将该 PATH 消息发送给下一跳。这个行为向后持续下去，直到该 TE 隧道的尾端路由器收到这个 PATH 消息。
- 在收到这个 PATH 消息后，尾端路由器将沿着 PATH 消息所使用的相同路径返回一个 RESV（保留）消息，只不过方向是逆向。RESV 消息里就包含 RSVP 为 Tunnel 分配的标签。

3. **所以，RSVP 的主要功能有：**
   - 路径的建立和维护
   - 路径的拆除
   - 错误通告

4.2 RSVP 与标签

RSVP 为 TE 隧道在路径中传递信令，但是它还承载 MPLS 标签分发的义务，使得报文可以沿着 TE 隧道的路径进行标签交换。



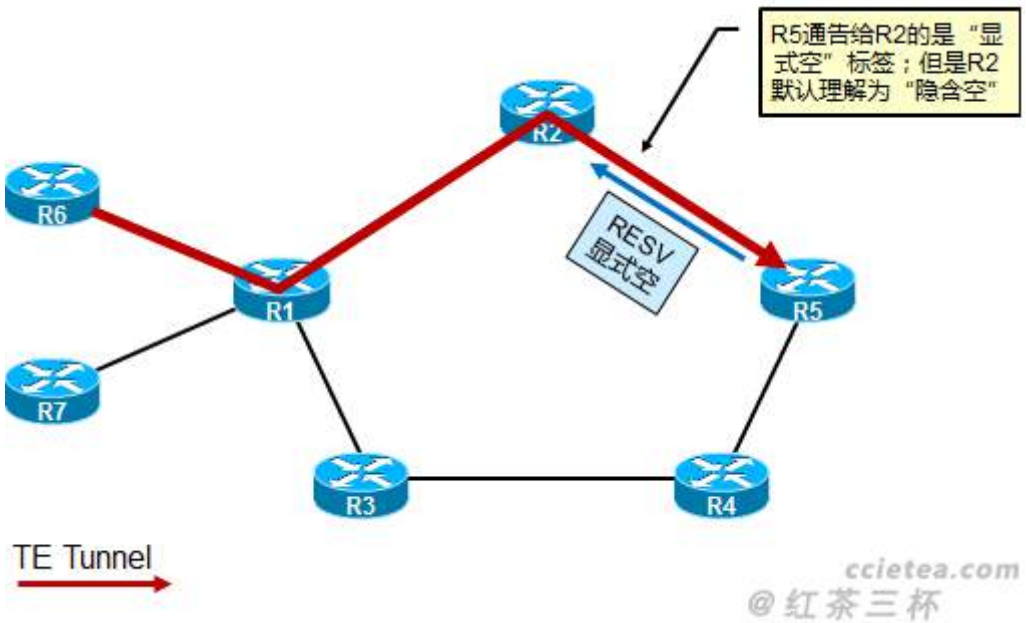


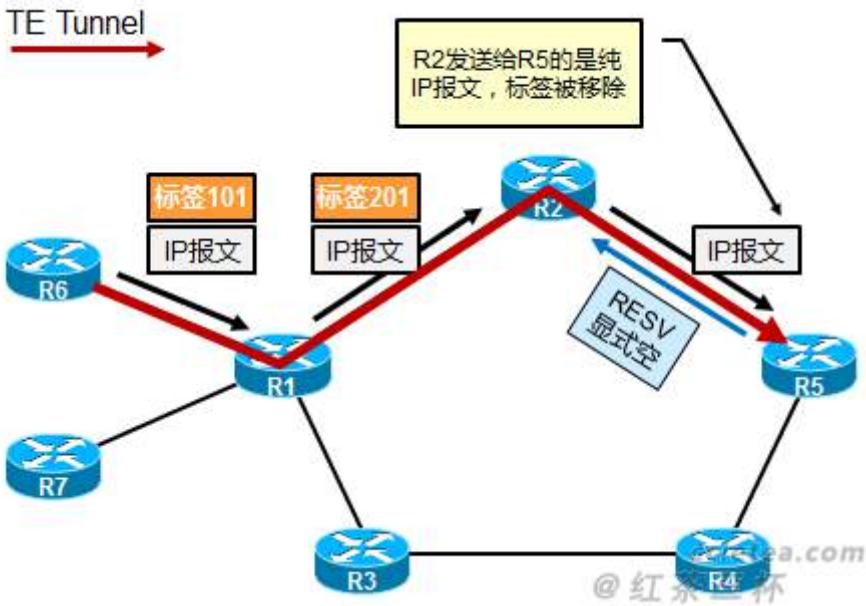
- 扩展的 OSPF 及 IS-IS 将 TE 所需的信息在网络中进行泛洪，那么 Tunnel 首端路由器就可以构建自己的 TE 数据库并且进行 CSPF 算法的计算，计算得出的结果是 Tunnel 的路径—一个 IP 地址的序列。
- 接下去 RSVP 的工作基于这个 IP 地址的序列继续工作。
- PATH 消息携带了一个标签请求对象，从 Tunnel 首端路由器开始一跳一跳的传递到了尾端路由器。
- 当尾端路由器接收到该标签请求对象的时候，它将为这一条 Tunnel LSP 分配一个标签，并且将该标签通过 RESV 消息中的标签对象通告给上游路由器（倒数第二跳）。这个标签是尾端路由器 LFIB 表中的入站标签。
- 倒数第二跳路由器从尾端路由器收到该标签后，将该标签作为这条 TE Tunnel 的出站标签，同时自己再为该隧道分配一个标签并将这个标签放置于 RESV 消息中的标签对象中发送给它自己的上游路由器。
- 这样的行为将持续下去直到 RESV 消息到达该 TE 隧道 LSP 的首端路由器。也就是说，在首端路由器发出请求之后，标签是从尾端路由器向首端路由器逐跳通告的。这说明 TE 隧道使用的是下游被动 DOD 的标签分发

模式。

注意事项：

- 尾端路由器给倒数第二跳路由器通告的标签是“显式空标签”，然而如果倒数第二跳路由器运行的是 CISCO IOS 的话，它将以隐式空标签的形式对此进行理解，也就是说，默认情况下，在 TE 隧道中，倒数第二条路由器发送给尾端路由器的报文是没有标签的（或者说顶部标签已被移除）。



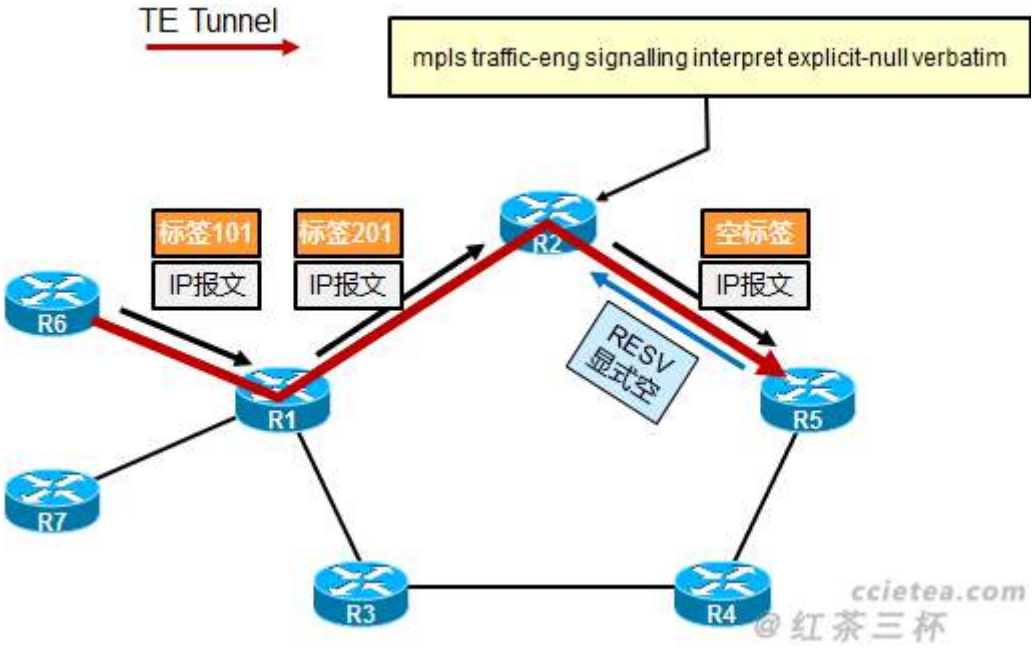


如果你希望倒数第二跳路由器将尾端路由器通告的“显式空标签”就理解为“显式空标签”，那么你可以

在倒数第二跳路由器（也就是上图中的 R2）上配置这条隐藏的命令：

```
mpls traffic-eng signalling interpret explicit-null verbatim
```

这条命令将使得倒数第二跳路由器在发送数据给尾端的时候，仍然带上标签，这个标签是空标签，verbatim 单词的中文翻译是 “完全按照字面的”。这么做的目的，还是为了保留 EXP 字段，以便保留 QoS 信息，如下图：



另外，你可以在尾端路由器上配置如下命令：

```
mpls traffic-eng signalling advertise implicit-null [acl x]
```

这条命令将使得尾端路由器通告隐式空标签。（默认的动作通告显式空标签）

ACL 用来匹配向谁发送显式空标签。

- 因为 RSVP 已经能够完成标签分配的任务了，所以就不需要在接口上再激活标签分发协议（LDP）了。

4.3 消息类型

- **RSVP 消息类型**

- PATH 用来建立和维护、保留
- RESV 响应 PATH 消息，用来建立和维护 LSP 通道
- PATHTear 报文和 PATH 类似，只不过它是在需要拆除首端路由器创建的 TE tunnel LSP 的时候发送，例如 tunnel 接口被手工 shutdown。
- RESVTear 报文和 RESV 类似，只不过它是尾端路由器回应所收到的 PathTear 消息的时候发送
- PATHErr 实际上是发送给首端路由器的，发送该 error 消息的最主要原因是被 TE LSP 使用的链路发生故障，还可能是觉 LSR 收到带有伪造消息的 PATH 消息
- RESVErr 是发送给尾端路由器的

- **RSVP 消息都有一个公共头部，后面跟随一个或多个 Objects**

4.4 RSVP 对 LSP tunnel 的扩展

1. RSVP 对 LSP tunnel 的扩展

RFC3209 RSVP-TE Extensions to RSVP for LSP tunnels 中详细描述了 RSVP 协议对 MPLS TE 功能的扩展

RSVP 对于 LSP tunnel 的扩展：

- 支持 DoD（Downstream-on-Demand）方式的标签分配
- 支持为显式的 LSP tunnel 分配网络资源
- 改变属性（路径、带宽）时，采用 Make Before Break 的方式
- 支持记录 LSP tunnel 经过的每个站点，可以用来防止环路
- 支持对 LSP tunnel 进行诊断

2. RSVP-TE 扩展的新对象

对象名称	存在于报文	描述
Label_request	PATH	标签请求对象
Label	RESV	下游分配过来的标签
Explicit_route	PATH	这个不用说了吧？
Record_route	PATH/RESV	RRO

Session_Attibute	PATH	包含隧道的设置优先级、保持优先级和一些 flags 标志位
Sender_Template	PATH	包含隧道发送者地址、LSP ID 等信息
Filter_Spec	RESV	包含隧道发送者地址、LSP ID 等信息
Session	PATH/RESV	包含隧道终点地址、隧道 ID、扩展隧道 ID（就是隧道起点）等信息
Flowspec	RESV	各种带宽信息
Sender_Tspec	PATH	主要包含 TE tunnel 的带宽请求

上述相同背景色的字段都是有对应关系的。

4.5 Record Route（RRO）

激活 RRO 使用如下配置：

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel destination 5.5.5.5
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth    60000
 tunnel mpls traffic-eng path-option 10 explicit name R1R2R5
 tunnel mpls traffic-eng path-option 20 dynamic
 tunnel mpls traffic-eng record-route
```

激活 record route，可以使得 PATH、RESV 消息携带 record route 对象，记录下 tunnel 沿途的经过的 IP，可以起到一定的防环作用，当然还有其他重要的作用，在后面解释。

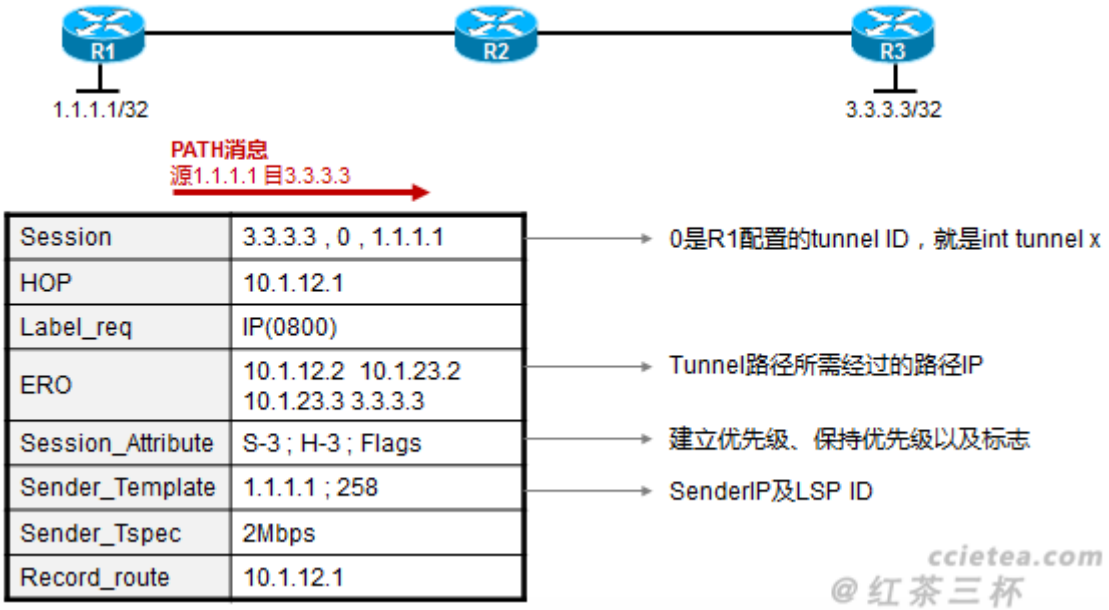
R4#sh mpls tr tun

```
LSP Tunnel R1_t0 is signalled, connection is up
  InLabel   : FastEthernet0/0, 400
  OutLabel : FastEthernet1/0, implicit-null
  RSVP Signalling Info:
        Src 1.1.1.1, Dst 5.5.5.5, Tun_Id 0, Tun_Instance 26
```

```
RSVP Path Info:
    My Address: 10.1.45.4
    Explicit Route: 10.1.45.5 5.5.5.5
    Record Route: 10.1.34.3 10.1.13.1          !!表示从 13.1 到 34.3 再到了本地（34.4）
    Tspec: ave rate=60000 kbits, burst=1000 bytes, peak rate=60000 kbits
RSVP Resv Info:
    Record Route:   NONE
    Fspec: ave rate=60000 kbits, burst=1000 bytes, peak rate=60000 kbits
```

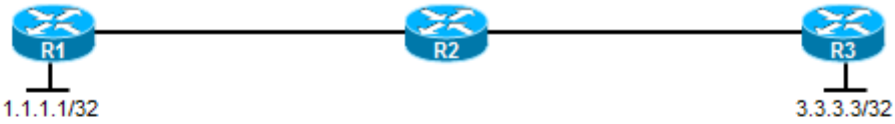
4.6 RSVP 路径建立过程

- TE LSP tunnel 都是由隧道的首端发起建立
- CSPF 计算出来路径
- RSVP-TE 按照上述路径请求建立 LSP tunnel
- LSP tunnel 都是单向的



PATH消息
源1.1.1.1 目3.3.3.3

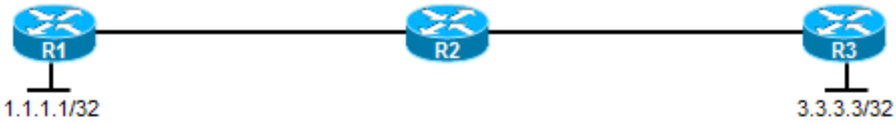
Session	3.3.3.3 , 0 , 1.1.1.1
HOP	10.1.23.2
Label_req	IP(0800)
ERO	10.1.23.3 3.3.3.3
Session_Attribute	S-3 ; H-3 ; Flag
Sender_Template	1.1.1.1 ; 258
Sender_Tspec	2Mbps
Record_route	10.1.23.2 10.1.12.1



PATH消息
源10.1.23.3目10.1.23.2

表示对Make Before Break的支持

Session	3.3.3.3 , 0 , 1.1.1.1
HOP	10.1.23.3
Style	Shared-Explicit
FlowSpec	2Mbps
FilterSpec	1.1.1.1 ; 258
Label	PoP (0)



PATH消息
源10.1.23.3目10.1.23.2

Session	3.3.3.3 , 0 , 1.1.1.1
HOP	10.1.12.2
Style	Shared-Explicit
FlowSpec	2Mbps
FilterSpec	1.1.1.1 ; 258
Label	200

4.7 RSVP 路径维护及拆除

1. **路径维护**
   - 每隔 30S 首端发送 PATH 消息给下游
   - 下游邻居每隔 30S 发送 RESV 消息

2. **路径的拆除**

   如果首端要拆除 tunnel ,则沿 tunnel 路径发送 PATHtear ,收到的节点回应 RESVtear ,这样两点之间的 tunnel 就拆除了，然后继续往下拆除。

想象一下如果 AB 之间的链路承载了大量的 tunnel，那么当这根链路 DOWN 掉的时候，将有可能在短时间内触发大量的 PATHtear 及 RESVtear，那么如何优化呢？

可以使用 router(config)# ip rsvp signalling rate-limit 来限制信令消息的发送速率

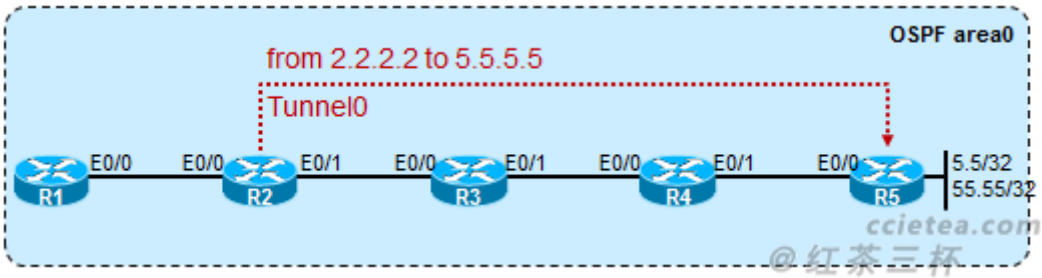
或者使用 router(config-if)# hold-queue x in 来限制接收速率

5 MPLS TE 流量转发

MPLS TE 报文转发是基于标签的，数据包将沿着预先建立好的 LSP 进行报文转发。**那么 tunnel 建立好后，如何把流量引入 MPLS TE Tunnel 呢。流量只有上了 tunnel，才会进入 LSP。有三种方法：**

- 静态路由 Static Route
- 自动路由 Auto Route
- 策略路由 Policy Route

5.1 Static route



假设我们现在已经建立好了一条首端为 R2、尾端为 R5 的 TE Tunnel ,这将在 R2 本地路由表中产生一个 tunnel 接口。那么一种最简单的方法，就是将上 tunnel 的流量直接用静态路由引到 tunnel 接口。例如用下面的配置完成后，R1 及 R2 去往 55.55.55.0/24 的流量就会从 TE tunnel 走。

```
Ip route 55.55.55.55 255.255.255.255 tunnel0
```

R1#traceroute 55.55.55.55

```
1 10.1.12.2 4 msec 0 msec 4 msec
2 10.1.23.3 [MPLS: Label 300 Exp 0] 0 msec 0 msec 0 msec
3 10.1.34.4 [MPLS: Label 400 Exp 0] 0 msec 12 msec 8 msec
4 10.1.45.5 8 msec 8 msec *
```

MPLS TE tunnel 的静态路由转发方式，支持路由递归。

使用静态路由的方式当然是最简单，也是最好把控的，但是扩展性太差，如果存在多条 tunnel 那么配置和维护就比较麻烦。

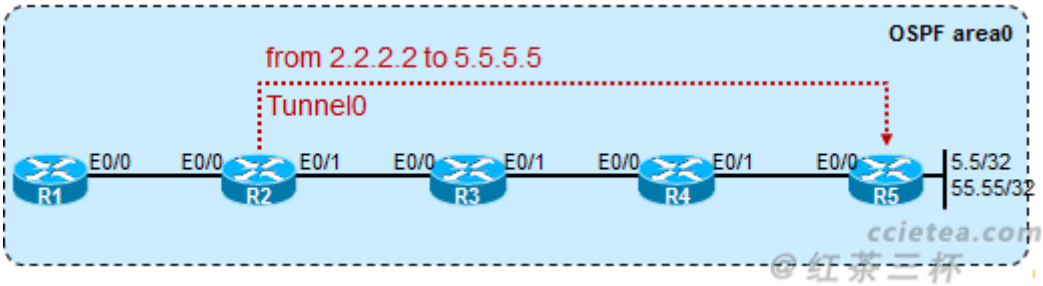
5.2 AutoRoute

5.2.1 AutoRoute 概述

- **MPLS TE tunnel 本身不支持 IGP 路由协议因为：**
  - TE tunnel 是单向的
  - TEDB 已经拥有所有区域内的链路状态信息，所以不需要再使用 IGP 路由协议

- **Auto Route：**
  - MPLS TE 的 AutoRoute 特性把 TE tunnel 作为一个直连链路参与 SPF 计算（只是 R2 自己）
  - 这里的 SPF 计算是普通的 SPF 计算，而不是 CSPF
  - AutoRoute 只会影响 TE 隧道的首端，其他路由器并不知道 TE 隧道的存在，不会影响他们的路由选路

5.2.2 AutoRoute 基本配置



R1 的配置如下：

```
interface eth0/0
   ip address 10.1.12.1 255.255.255.0
interface loopback0
   ip address 1.1.1.1 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router ospf 1
   router-id 1.1.1.1
   network 10.1.12.1 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
```

```
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

R2 的配置如下（暂时没配置 tunnel）：

```
interface eth0/0
  ip address 10.1.12.2 255.255.255.0
interface eth0/1
  ip address 10.1.23.2 255.255.255.0
interface loopback0
  ip address 2.2.2.2 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
  router-id 2.2.2.2
  network 2.2.2.2 0.0.0.0 area 0
  network 10.1.12.2 0.0.0.0 area 0
  network 10.1.23.2 0.0.0.0 area 0
  mpls traffic-eng router-id loopback0
  mpls traffic-eng area 0
!
interface eth 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface eth 0/1
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

R3 的配置如下：

```
interface eth0/0
  ip address 10.1.23.3 255.255.255.0
interface eth0/1
```

```
   ip address 10.1.34.3 255.255.255.0
interface loopback0
   ip address 3.3.3.3 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R4 的配置如下：

```
interface eth0/0
   ip address 10.1.34.4 255.255.255.0
interface eth0/1
   ip address 10.1.45.4 255.255.255.0
interface loopback0
   ip address 4.4.4.4 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 400 499
```

```
!
router ospf 1
   router-id 4.4.4.4
   network 10.1.34.4 0.0.0.0 area 0
   network 10.1.45.4 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R5 的配置如下：

```
interface eth0/0
   ip address 10.1.45.5 255.255.255.0
interface loopback0
   ip address 5.5.5.5 255.255.255.255
interface loopback1
   ip address 55.55.55.55 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 500 599
!
router ospf 1
   router-id 5.5.5.5
   network 10.1.45.5 0.0.0.0 area 0
   network 5.5.5.5 0.0.0.0 area 0
   network 55.55.55.55 area 0
   mpls traffic-eng router-id loopback0
```

```
 mpls traffic-eng area 0
!
interface eth 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

那么到目前为止，基础配置都都完成了。现在我们在 R2 上创建 TE Tunnel：

R2 的配置增补如下：

```
Interface tunnel0
 ip unnumbered loopback 0
 tunnel destination 5.5.5.5
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng path-option 10 dynamic
 tunnel mpls traffic-eng autoroute announce          !!激活 Tunnel 的 autoroute 特性
```

完成上述配置后，相当于 R2 上多了一条到 R5 的单向直连链路。这条链路将直接参与 R2 本身的路由计算。总的结果就是，相当于 R2 上现在有了三个直连接口：eth0/0、eth0/1、tunnel0。

现在我们来验证一下。

R2# show ip route

```
C       2.2.2.2 is directly connected, Loopback0
O       5.5.5.5 [110/31] via 5.5.5.5, 00:00:01, Tunnel0
C       10.1.12.0/24 is directly connected, Ethernet0/0
C       10.1.23.0/24 is directly connected, Ethernet0/1
O       10.1.34.0/24 [110/20] via 10.1.23.3, 00:00:01, Ethernet0/1
O       10.1.45.0/24 [110/30] via 10.1.23.3, 00:00:01, Ethernet0/1
O       55.55.55.55 [110/31] via 5.5.5.5, 00:00:01, Tunnel0
```

我们看到 R2 上，关于 5.5.5.5 及 55.55.55.55 这两条在 R5 也就是 tunnel 尾端的“后方的”网络（的路由），出接口均为 tunnel0。这就意味着，当 R2 收到去往这两个网段的数据，直接放入 TE tunnel。相比于静态路由的放方式，autoroute 的优势就显示出来了。

R1#traceroute 55.55.55.55

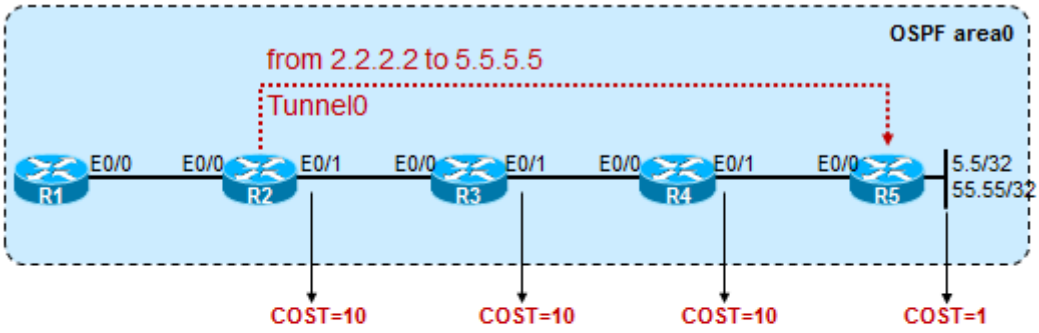
```
Type escape sequence to abort.
```

```
Tracing the route to 55.55.55.55
  1 10.1.12.2 8 msec 4 msec 0 msec
  2 10.1.23.3 [MPLS: Label 300 Exp 0] 4 msec 0 msec 4 msec
  3 10.1.34.4 [MPLS: Label 400 Exp 0] 4 msec 0 msec 4 msec
  4 10.1.45.5 0 msec *    0 msec
```

看到标签交换的过程了吧？证明数据的确是上了 TE tunnel 的。

5.2.3 AutoRoute metric

再进一步分析一下 metric 的问题。我们重点看 55.55.55.55，经过上面的基本配置，目前在 R2 的路由表中的 metric 为 31，计算方法当然很简单，如下：



R2# show ip route

```
O       5.5.5.5 [110/31] via 5.5.5.5, 00:00:01, Tunnel0
O       55.55.55.55 [110/31] via 5.5.5.5, 00:00:01, Tunnel0
```

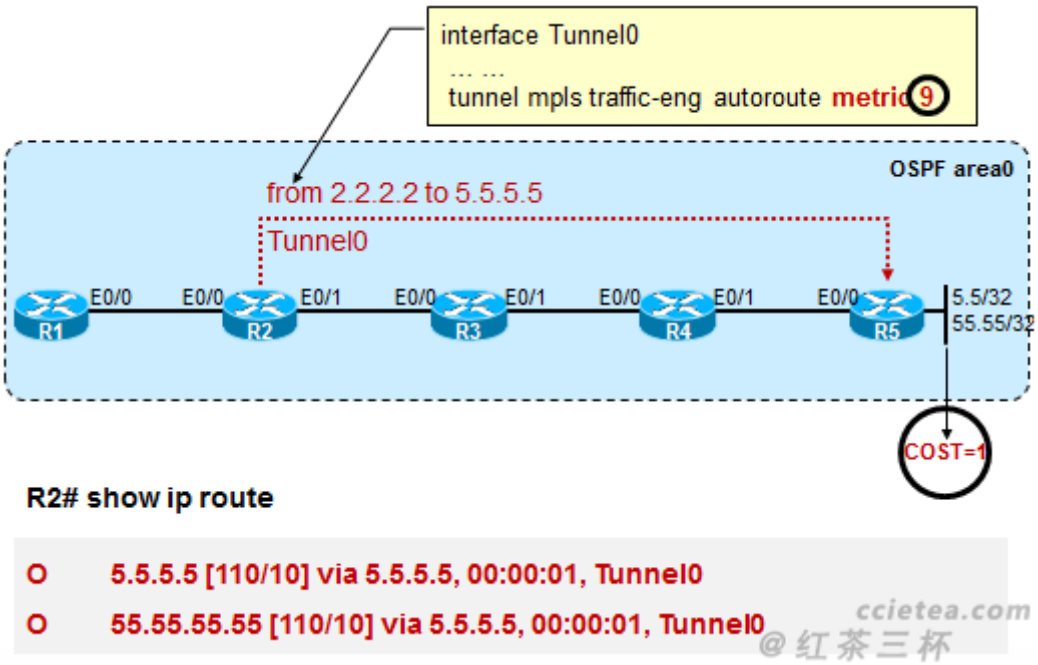

注意，此刻如果希望通过修改 tunnel 接口的 ip ospf cost 从而来调整路由的 cost，这在针对 autoroute 特性计算得到的关联 tunnel 接口的路由是无效的，需使用 tunnel mpls traffic-eng autoroute metric 命令来配置。

- **tunnel mpls traffic-eng autoroute metric 命令解析**

  如果我们在 R2 的 TE tunnel 口中增加如下配置：

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
```

```
tunnel mpls traffic-eng autoroute announce
tunnel mpls traffic-eng autoroute metric 9
tunnel mpls traffic-eng path-option 10 dynamic
```



增加这条命令后，依靠 autoroute 特性计算出来、关联 tunnel0 口的这两条 OSPF 路由 metric 就变成了 9+1：这里的 9 就是我们配置的 autoroute metric 9，这里的 1 就是 R5 的 Loopback 口 cost。注意，**这只会影响 R2 自身的路由选择**。

补充一下：上面这个环境中，loopback 接口 cost=1，其他所有物理接口的 cost=10。那么如果我们配置：

```
tunnel mpls traffic-eng autoroute metric 32
```

这将导致 R2 计算路由的时候，从 tunnel0 口到达 5 及 55 路由 metric 变成 32+1=33，大于从 e0/1 走的路由的 metric=31。于是 R2 的路由表发生了变化，5 及 55 网段的路由变成了：

```
O        5.5.5.5 [110/31] via 10.1.23.3, 00:02:29, Ethernet0/1
O        55.55.55.55 [110/31] via 10.1.23.3, 00:02:29, Ethernet0/1
```

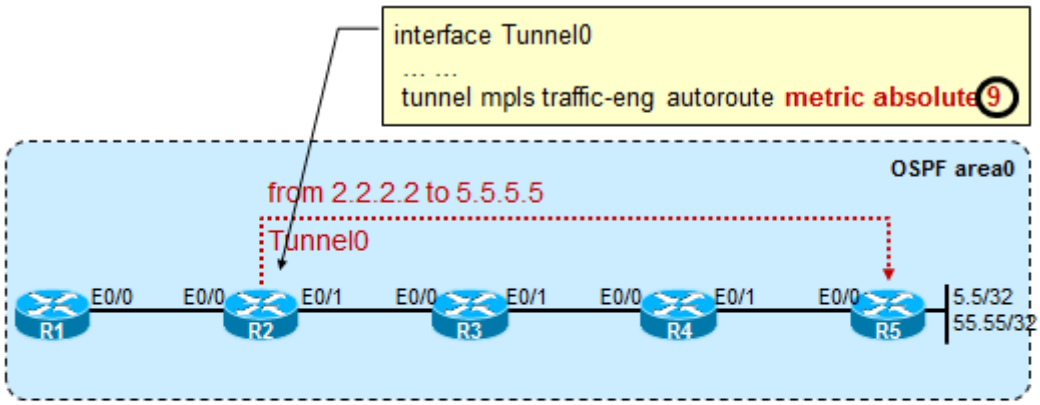
不再走 tunnel 了，此时再从 R2 去 traceroute 55.55.55.55，就不再看到标签了。

- **tunnel mpls traffic-eng autoroute metric absolute 解析**

  如果 R2 的配置变更为：

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
```

```
tunnel destination 5.5.5.5
tunnel mpls traffic-eng autoroute announce
tunnel mpls traffic-eng autoroute metric absolute 9
tunnel mpls traffic-eng path-option 10 dynamic
```



R2# show ip route

```
O    5.5.5.5 [110/9] via 5.5.5.5, 00:00:01, Tunnel0
O    55.55.55.55 [110/9] via 5.5.5.5, 00:00:01, Tunnel0
```


配置这条命令，在 R2 上依赖 autoroute 特性计算出来的、关联 TE tunnel 口 OSPF 路由 metric 都变成了 9。这就是 absolute 关键字的意义，这里不会去做 cost 的累加。

补充一下：上面这个环境中，loopback 接口 cost=1，其他所有物理接口的 cost=10。那么如果我们配置：

tunnel mpls traffic-eng autoroute metric absolute 32

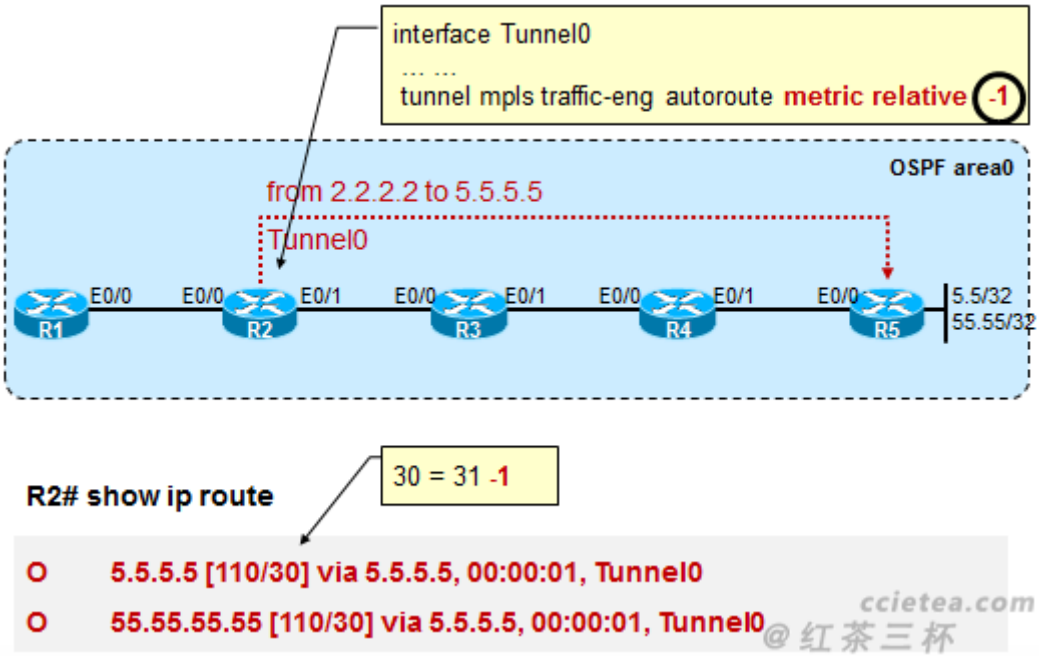
这将导致 R2 的路由发生变化，去往 5.5.5.5 及 55.55.55.55 的路由出接口变成了 e0/1。于是 R2 去往 55.55.55.55 及 5.5.5.5 将不再走 tunnel。

- **tunnel mpls traffic-eng autoroute metric relative 解析**

  如果 R2 的配置变更如下：

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng autoroute metric relative -1
```

tunnel mpls traffic-eng path-option 10 dynamic



这条 relative 的命令，可以配置个-10 到 10 的值。意思是在 IGP 度量值的基础上，加或者减去这个特定值。例如上图中，在不考虑 tunnel 的情况下，R2 上关于 5.5.5.5 和 55.55.55.55 的路由 metric=31。这个值怎么算的上面已经说过了。现在我们配置了这条带有 relative -1 关键字的命令后，依靠 autoroute 计算出来的、指向 tunnel0 口的 OSPF 路由，metric 就变成了 31-1=30。

小结一下，上面说了三条命令：

tunnel mpls traffic-eng autoroute metric x

首先这三条命令，都是对开启了 autoroute 特性的 TE tunnel 计算出来的路由得出的路由起作用的。上面这条命令，修改的是 tunnel 接口本身的 cost。

tunnel mpls traffic-eng autoroute metric absolute y

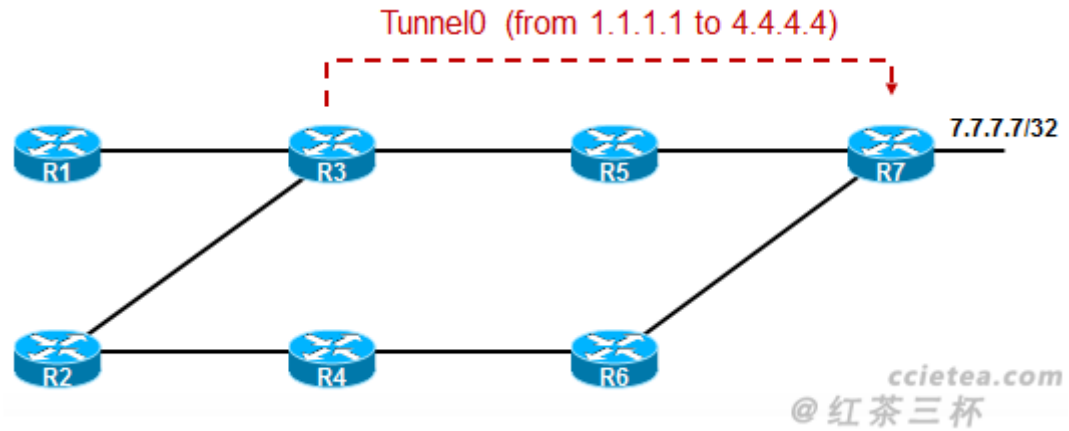
上面这条命令将直接把上述计算出来的路由的 cost 全改成 y

tunnel mpls traffic-eng autoroute metric relative z

上面这条命令将在路由的原始 cost 基础上，加上或者减去 z（z 的区间是-10 到 10）

5.2.4 AutoRoute 的缺陷

- Auto Route 在 MPLS TE tunnel 首端路由器上配置，只影响首端路由器的 OSPF 路由选路。
- 在实际的环境中 ,往往只有在少数的核心结点部署 MPLS TE ,那么边缘结点无法感知 TE tunnel 的存在，只能按照传统的 IGP 选路。



例如上面的例子，tunnel0 是建立在 R3 到 R7 的，对于 R1 和 R2 而言，他们是不知道 tunnel 的存在的，当他们去往 7.7.7.7/32 的时候，只会进行 IGP 的本身的路由计算，那么 R2 就有可能走的是 R4-R6-R7 这条路径，但是我们可能更希望这些流量能够通过 tunnel 来传输，那么对于这个需求，autoroute 就无法实现了，需要借助 Forwarding ajacency。

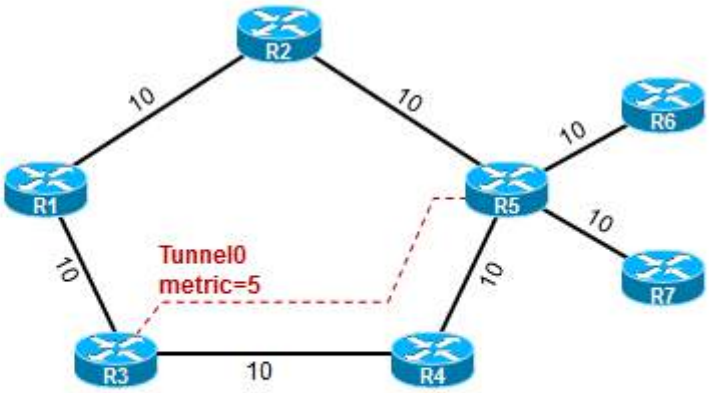
5.3 Forwarding Adjacency 转发邻接

5.3.1 特性概述

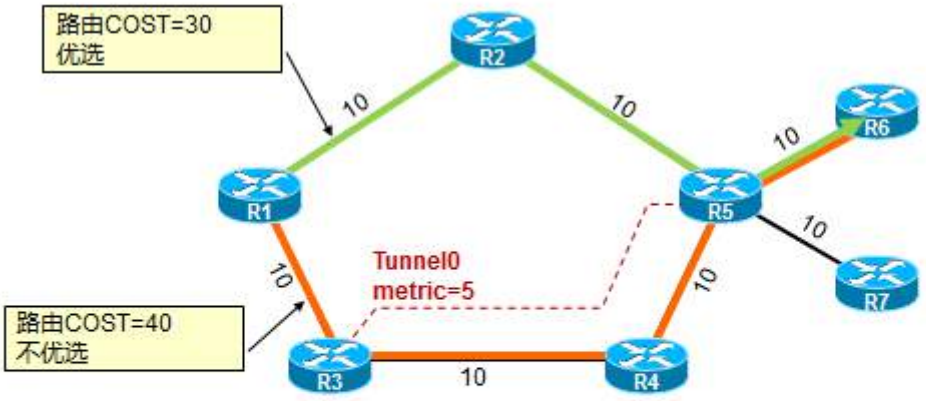
- 转发邻接特性把会使得 TE 路由器将 TE tunnel 作为虚路径（直连链路）在 IGP 路由协议区域内进行通告，这样所有的路由器都知道隧道的存在。这里的 IGP 协议可以是 OSPF 或 IS-IS
- 使用转发邻接特性时，隧道首端和尾端必须在同一个区域中，而且首尾两端的 Tunnel 接口都需要配置，也就是说这必须是一个双向隧道，并且两者的 tunnel 接口都要激活转发邻接

下面我们来看一个例子：

- **首先是无转发邻接的情况：**



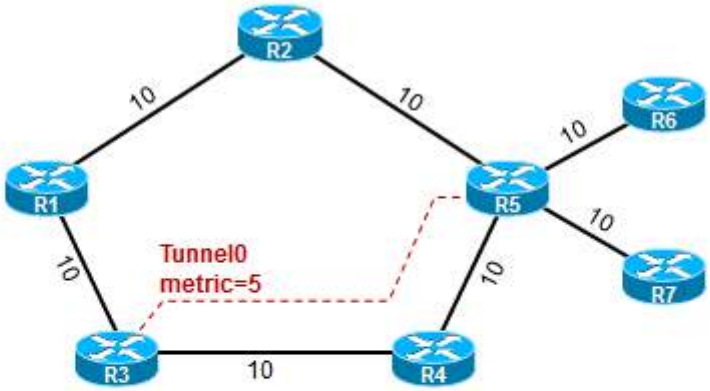
- R3及R5之间建立TE tunnel，注意这个tunnel是双向的



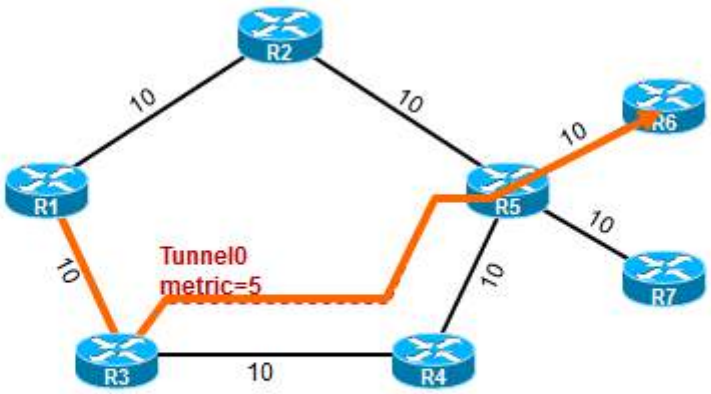
- TE隧道没有激活forwarding-adjacency时，R1到达R6的路径会选择R2作为下一跳，因为R1不知道tunnel0的存在，仍然以IGP metric看全局，从R2走的路由cost更小

我们看到虽然 R3 到 R5 之间建立了 tunnel，但是这条 tunnel 对于 R1 来说是看不见的，那么在 R1 眼里，也就只有 OSPF 的那些链路 OSPF cost，因此 R1 前往 R6，肯定是优选上面那条链路，因为 cost 小。

- **接下去我们在 R3 和 R5 上激活转发邻接，那么 R3 和 R5 就会在区域内通告这条 tunnel（以直连链路的方式）。**



- R3及R5在TE tunnel上激活forwarding-adjacency
- R3和R5会将tunnel的存在以直连链路的方式通过IGP协议（如OSPF或ISIS）告知给区域内其他路由器



- R1将选择R3作为到达R6的下一跳，R1上我们去看这条路径，它的metric=25

这里有几个注意事项：

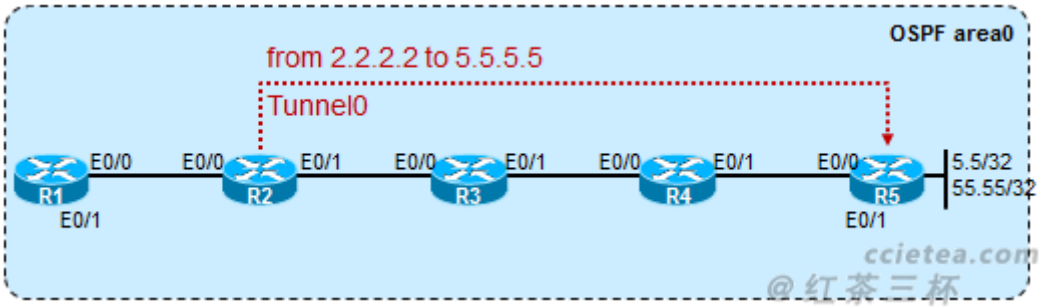
1. R3 及 R5 都要建立 TE tunnel，也就是说，TE tunnel 必须是双向的
2. R3 及 R5 都要激活转发邻接特性
3. 在激活特性后，R3 和 R5 都会在区域内通告这条 tunnel（以直连链路的形式）的存在。相当于咱在拓扑环境中多了一条路径（虚路径）。

5.3.2 配置命令

```
R3(config)#Interface tunnel 0
R3(config-if)#tunnel mpls traffic-eng forwarding-adjacency ?
  holdtime      how long in msecs to wait upon flooding a down forwarding     adjacentcy
```

tunnel 接口可以直接配置 igp metric，例如如果跑 ospf，直接在 tunnel 口中配置 ip ospf cost 即可

5.3.3 配置示例



R1 的配置如下：

```
interface eth0/0
  ip address 10.1.12.1 255.255.255.0
!
interface loopback0
  ip address 1.1.1.1 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router ospf 1
  router-id 1.1.1.1
  network 10.1.12.1 0.0.0.0 area 0
```

```
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R2 的配置如下（暂时没配置 tunnel）：

```
interface eth0/0
   ip address 10.1.12.2 255.255.255.0
interface eth0/1
   ip address 10.1.23.2 255.255.255.0
interface loopback0
   ip address 2.2.2.2 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
   router-id 2.2.2.2
   network 2.2.2.2 0.0.0.0 area 0
   network 10.1.12.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R3 的配置如下：

```
interface eth0/0
   ip address 10.1.23.3 255.255.255.0
interface eth0/1
   ip address 10.1.34.3 255.255.255.0
interface loopback0
   ip address 3.3.3.3 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R4 的配置如下：

```
interface eth0/0
   ip address 10.1.34.4 255.255.255.0
interface eth0/1
   ip address 10.1.45.4 255.255.255.0
interface loopback0
   ip address 4.4.4.4 255.255.255.255
```

```
!
Ip cef
mpls traffic-eng tunnels
mpls label range 400 499
!
router ospf 1
   router-id 4.4.4.4
   network 10.1.34.4 0.0.0.0 area 0
   network 10.1.45.4 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R5 的配置如下：

```
interface eth0/0
   ip address 10.1.45.5 255.255.255.0
interface loopback0
   ip address 5.5.5.5 255.255.255.255
interface loopback1
   ip address 55.55.55.55 255.255.255.255
!
Ip cef
mpls traffic-eng tunnels
mpls label range 500 599
!
router ospf 1
   router-id 5.5.5.5
```

```
  network 10.1.45.5 0.0.0.0 area 0
  network 5.5.5.5 0.0.0.0 area 0
  network 55.55.55.55 area 0
  mpls traffic-eng router-id loopback0
  mpls traffic-eng area 0
!
interface eth 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

那么到目前为止，基础配置都都完成了。现在我们在 R2 和 R5 上创建 TE Tunnel：

R2 的配置增补如下：

```
Interface tunnel0
  ip unnumbered loopback 0
  tunnel destination 5.5.5.5
  tunnel mode mpls traffic-eng
  tunnel mpls traffic-eng path-option 10 dynamic
  tunnel mpls traffic-eng forwarding-adjacency          !!激活 Tunnel 的 forwarding-adjacency
```

R5 的配置增补如下：

```
Interface tunnel0
  ip unnumbered loopback 0
  tunnel destination 2.2.2.2
  tunnel mode mpls traffic-eng
  tunnel mpls traffic-eng path-option 10 dynamic
  tunnel mpls traffic-eng forwarding-adjacency          !!激活 Tunnel 的 forwarding-adjacency
```

完成上述配置后，R2 及 R5 将在自己的 1 类 LSA 中通告这条 tunnel，就像一条直连链路一样。

R2#show ip ospf database router self-originate

```
                OSPF Router with ID (2.2.2.2) (Process ID 1)
                    Router Link States (Area 0)
  LS age: 43
  Options: (No TOS-capability, DC)
  LS Type: Router Links
```

```
Link State ID: 2.2.2.2
Advertising Router: 2.2.2.2
LS Seq Number: 80000011
Checksum: 0xAE93
Length: 72
Number of Links: 4

  Link connected to: another Router (point-to-point)
   (Link ID) Neighboring Router ID: 5.5.5.5
   (Link Data) Router Interface address: 0.0.0.13
    Number of MTID metrics: 0
     TOS 0 Metrics: 1000                    !!默认 tunnel 口通告的 cost=1000

  Link connected to: a Stub Network
   (Link ID) Network/subnet number: 2.2.2.2
   (Link Data) Network Mask: 255.255.255.255
    Number of MTID metrics: 0
     TOS 0 Metrics: 1

  Link connected to: a Transit Network
   (Link ID) Designated Router address: 10.1.23.3
   (Link Data) Router Interface address: 10.1.23.2
    Number of MTID metrics: 0
     TOS 0 Metrics: 10

  Link connected to: a Transit Network
   (Link ID) Designated Router address: 10.1.12.1
   (Link Data) Router Interface address: 10.1.12.2
    Number of MTID metrics: 0
     TOS 0 Metrics: 10
```

R5 也是类似：

此时此刻对于 R2 来说，他有两条路径去往 5.5.5.5 和 55.55.55.55。

一条从 tunnel0 口走，cost=1000+1=1001

一条从 eth0/1 口走，cost=10+10+10+1=31

这时候 R2 去往 55.55.55.55，走的是传统 IP 数据，而不走上 LSP。

R2#show ip route

O 5.5.5.5 [110/31] via 10.1.23.3, 00:03:44, Ethernet0/1

O 55.55.55.55 [110/31] via 10.1.23.3, 00:03:44, Ethernet0/1

但是如果我们在 R1 上修改配置：

```
Interface tunnel0
  Ip ospf cost 1
```

那么从 tunnel0 口，cost=1+1=2，更优，于是 R2 的路由表就变了：

R2#show ip route

O 5.5.5.5 **[110/2]** via 0.0.0.0, 00:00:01, **Tunnel0**

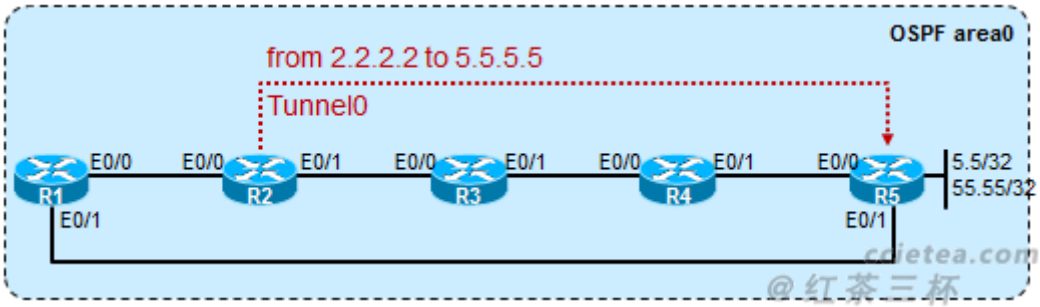
O 55.55.55.55 **[110/2]** via 0.0.0.0, 00:00:01, **Tunnel0**

R2 优选从 tunnel 口走的路由，这时候 R2 去往 55.55.55.55 的流量就上了 tunnel，走 LSP 进行标签交换。

所以我们在 CISCO IOS 路由器上，激活转发邻接特性，除了让 TE tunnel 端点路由器将 tunnel 当成直连链路进行通告，同时在自身进行 SPF 算法计算的时候，也会一并考虑这条 tunnel。而且如果我们修改 tunnel 口的 IGP metric，这个 tunnel 的 metric（对于 OSPF 来说这个 tunnel 口默认 cost=1000）也会随着链路状态信息一并通告给其他的邻居，并且也影响其他路由器的选路。

这里要注意和 autoroute 区分开来。我们上面的配置中没有激活 autoroute 特性。autoroute 特性以及 autoroute metric 的配置都只影响 tunnel 端点路由器自身，而不会通告给其他邻居。但是转发邻接特性就不同了。而且两者修改 metric 的方式也不用，autoroute 需要搭配 autoroute 关键字来修改 metric 并且只影响配置者自身。而在转发邻接中，是直接在 tunnel 接口中修改 metric，例如使用 ISIS，那么就是 isis metric 命令，如果用 ospf，那就是 ip ospf cost。

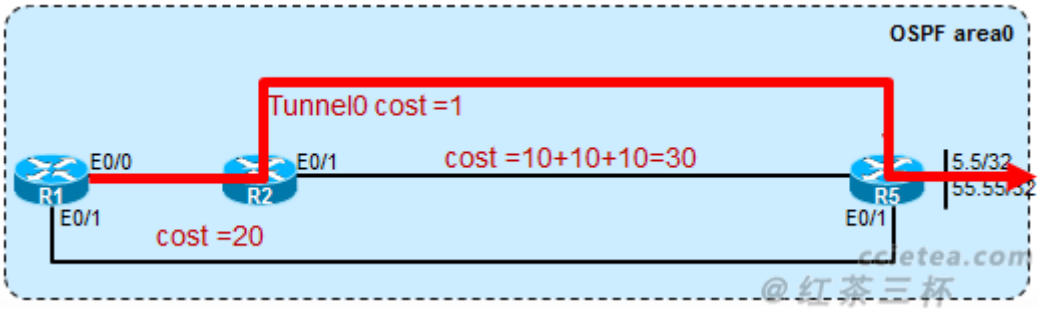
现在我们来考虑另一个场景，假设 R1 和 R5 之间增加一条物理连接，为了方便实验我直接在 R1-R5 之间拉了条线，大家权当这条新增的路径中有多台路由器，发挥下想象吧亲：



那么默认情况下，R1 去往.5.5.5.5 及 55.55.55.55 肯定是优选从 E0/1 直接到 R5。

但是，如果我们希望 R1 去往.5.5.5.5 及 55.55.55.55 流量上 TE tunnel 呢？

其实这就是个 metric 的把戏，只不过我们在考虑的时候，记得将 tunnel 这条虚路径一并考虑进去。



在本环境中，所有物理接口的 ospf cost=10，loopback 口的 cost=1

默认情况下 tunnel0 口的 cost=1000，我们可以将 tunnel 接口的 cost 修改为 1；再把 R1 的 e0/1 口的 cost 改为 20.

这样一来首先 R2 自己会优选从 tunnel 口到 R5，metric=1+1=2。而对于 R1 去 R5 有两个选择：

R1 从 R2 走然后上 tunnel，这条路径的 cost=10+1+1=12

R1 从 eth0/1 到 R5，这条路径的 cost=20+1=21

5.4 策略路由

- TE 的策略路由与传统的策略路由没什么区别，也不会改变路由表流量的转发基于配置的策略
- 只需把满足条件的流量的下一跳扔到 tunnel 接口即可

6 MPLS TE 配置及实验

6.1 配置命令

1. **基本配置**

router(config)#Ip cef

这个不用说了吧？激活 CEF

router(config)#mpls traffic-eng tunnels

全局开启 MPLS TE tunnel 特性，相当于一个全局的开关

router(config-if)#mpls traffic-eng tunnels

激活接口上的 MPLS TE tunnel 特性

router(config-if)#ip rsvp bandwidth *[total-flow-kbps] [single-flow-kbps]*

激活接口上的 RSVP，并配置预留带宽（可选）

- 如果该命令后不接带宽参数，那么该接口的“总可预留带宽”默认为该接口物理带宽的 75%
- ip rsvp bandwidth 2000　　这条命令将接口的总可预留带宽设置为 2M
- ip rsvp bandwidth 2000 500　　后面的 500 是针对单个流的，限制在 500K

router(config-router)#mpls traffic-eng area x

在 OSPF 进程模式中配置，这条命令将激活特定区域的 MPLS TE

router(config-router)#mpls traffic-eng router-id {interface}

在 IGP（如 OSPF 或 ISIS）的路由进程下配置。这条命令设置 TE router 的 routerID

router(config)#interface tunnel x

创建一个 tunnel

router(config-if)#tunnel destination *ip-addr*

配置 tunnel 的目的地

router(config-if)#tunnel mode mpls traffic-eng

配置 tunnel 的封装模式为 MPLS TE

router(config-if)#tunnel mpls traffic-eng bandwidth *BW*

配置 tunnel 的 requested bandwidth

router(config-if)#tunnel mpls traffic-eng path-option *x* **{dynamic | explicit {name** *path-name* **|** *path-number* **} } [lockdown]**

配置 tunnel 使用静态指定的路径或通过 TE 拓扑数据库计算出来的动态路径

2. **3 种可能导致 TE tunnel 重新最优化（reoptimize）的触发因素。**

- **周期性 reoptimize**

  在 CISCO IOS 中，一条 TE tunnel 的 reoptimize 默认每 1 小时进行一次。

  router(config)#mpls traffic-eng reoptimize timers frequency ?

  如果该时间指定为 0，那么周期性 reoptimize 将会在路由器上的所有 TE tunnel 中被关闭。当然，你可以为单条 TE tunnel 关闭 reoptimize，命令如下：

  Router(config-if)# mpls traffic-eng path-option x {dynamic | explicit name y } [**lockdown**]

  使用 lockdown 关键字（注意上述配置是在 TE tunnel 口中配置）。

- **事件导致的 reoptimize**

  缺省情况下，CISCO IOS 不会因为网络中的一条链路重新可以被一条 TE tunnel 所使用的时候而触发重新最优化，但是可以激活这个操作。要在一条链路在 MPLS TE 中变为可操作的时候启用重新最优化，使用下面的命令：

  mpls traffic-eng reoptimize events link-up

  经实验验证有效。可用 debug mpls traffic-eng tunnels events 及 debug mpls traffic-eng tunnels reoptimize 查看。

- **手工重新 reoptimize**

  Router# mpls traffic-eng reoptimize

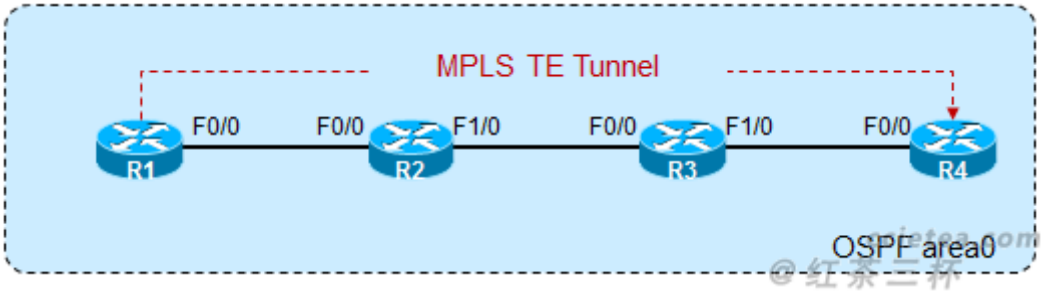
  手工重新最优化

6.2 验证查看

debug ip rsvp dump-messages

可以看到出站 PATH 及入站 RESV 消息

6.3 基础实验 1（OSPF）



1. **实验环境**
   - 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
   - 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
   - 全网运行 OSPF，宣告直连和 loopback 接口
   - 在 R1 上建立一个 TE Tunnel，源为自身的 loopback0，目的为 R4 的 4.4.4.4
2. **实验步骤**
   - 完成基本的 IP 配置（配置省略）
   - 所有路由器运行 OSPF
   - 全局激活 MPLS TE tunnel，并设定 MPLS 标签空间
   - 为每台路由器的 OSPF 激活 MPLS TE 的扩展，并且手工设置用于 MPLS TE 的 RouterID
   - 每台路由器的接口都激活 RSVP 及 MPLS TE tunnel 的支持
   - 在 R1 上完成 MPLS TE 的配置
   - 测试数据流的传输过程，观察现象

3. **配置命令**

R2 的配置如下（所有配置都省去了接口 IP 的配置）：

```
ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
  router-id 2.2.2.2
  network 2.2.2.2 0.0.0.0 area 0
```

```
 network 10.1.12.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
 mpls traffic-eng router-id loopback0
 mpls traffic-eng area 0
!
interface fast 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface fast 1/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R3 的配置如下：

```
Ip cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
 router-id 3.3.3.3
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 mpls traffic-eng router-id loopback0
 mpls traffic-eng area 0
!
interface fast 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface fast 1/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R4 的配置如下：

```
Ip cef
```

```
mpls traffic-eng tunnels
mpls label range 400 499
!
router ospf 1
   router-id 4.4.4.4
   network 4.4.4.4 0.0.0.0 area 0
   network 10.1.34.4 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface fast 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R1 的配置非常关键，我们重点来看一下：

R1 的配置如下：

```
Ip cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router ospf 1
   router-id 1.1.1.1
   network 1.1.1.1 0.0.0.0 area 0
   network 10.1.12.1 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface fast 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
!
Interface tunnel0                                    !! TE tunnel
```

```
ip unnumbered loopback 0
tunnel destination 4.4.4.4
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng priority 7 7                    !!tunnel 的建立和保持优先级
tunnel mpls traffic-eng bandwidth 20000                 !!带宽要求为 20M
tunnel mpls traffic-eng path-option 10 dynamic          !!使用动态计算的方式建立 tunnel PATH
```

4. 验证及查看

首先我们在 R1 上观察一下：

R1#show ip ospf database

```
    Type-10 Opaque Link Area Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum Opaque ID
1.0.0.0         1.1.1.1         366         0x80000005 0x0067EF 0
1.0.0.0         2.2.2.2         697         0x80000001 0x0090AA 0
1.0.0.0         3.3.3.3         1230        0x8000001A 0x002ED2 0
1.0.0.0         4.4.4.4         947         0x80000018 0x008F6C 0
1.0.0.1         3.3.3.3         1230        0x80000018 0x001C01 1
1.0.0.2         3.3.3.3         1230        0x80000011 0x00F741 2
```

上面输出的就是在本环境中，各路由器在 area0 内泛洪的 type10 的 opaque-area LSA。

可以进一步详细的看一下，例如看 R1 自己产生 LSA10：

R1#show ip ospf database opaque-area self-originate

```
OSPF Router with ID (1.1.1.1) (Process ID 1)
                    Type-10 Opaque Link Area Link States (Area 0)
  LS age: 29
  Options: (No TOS-capability, DC)
  LS Type: Opaque Area Link
  Link State ID: 1.0.0.0
  Opaque Type: 1
  Opaque ID: 0
  Advertising Router: 1.1.1.1
  LS Seq Number: 80000001
```

```
Checksum: 0x6FEB
Length: 132
Fragment number : 0

  MPLS TE router ID : 1.1.1.1

  Link connected to Broadcast network
    Link ID : 10.1.12.2                          !!只有一个接口激活 MPLS TE
    Interface Address : 10.1.12.1
    Admin Metric : 1                             !! TE metric，默认和下面的 IGP metric 相等
    Maximum bandwidth : 12500000                 单位是 Bytes，乘以 8 也就是 100Mbps
    Maximum reservable bandwidth : 9375000       !!最大可预留带宽 是 75M
    Number of Priority : 8
    Priority 0 : 9375000      Priority 1 : 9375000
    Priority 2 : 9375000      Priority 3 : 9375000
    Priority 4 : 9375000      Priority 5 : 9375000
    Priority 6 : 9375000      Priority 7 : 9375000
    Affinity Bit : 0x0
    IGP Metric : 1                               !!接口的 OSPF cost

  Number of Links : 1
```

接下去在 R1 上看一下 Tunnel：

R1#show mpls traffic-eng tunnels

```
Name: R1_t0                                (Tunnel0) Destination: 4.4.4.4
  Status:
    Admin: up          Oper: up     Path: valid       Signalling: connected

    path option 10, type dynamic (Basis for Setup, path weight 3)

  Config Parameters:
    Bandwidth: 20000   kbps (Global)   Priority: 7   7    Affinity: 0x0/0xFFFF
```

```
    Metric Type: TE (default)              !!带宽要求为 20M，隧道抢占优先级及保持优先级为 7
    AutoRoute:   disabled   LockDown: disabled   Loadshare: 20000      bw-based
    auto-bw: disabled

  InLabel  :  -
  OutLabel : FastEthernet0/0, 201                    !! 出站标签 201，这是 R2 给的
  RSVP Signalling Info:
        Src 1.1.1.1, Dst 4.4.4.4, Tun_Id 0, Tun_Instance 2
    RSVP Path Info:
      My Address: 10.1.12.1
      Explicit Route: 10.1.12.2   10.1.23.2   10.1.23.3   10.1.34.3
                      10.1.34.4 4.4.4.4                  !!LSP 的路径
      Record Route:   NONE
      Tspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
    RSVP Resv Info:
      Record Route:   NONE
      Fspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
  History:
    Tunnel:
      Time since created: 10 minutes, 5 seconds
      Time since path change: 9 minutes, 49 seconds
    Current LSP:
      Uptime: 9 minutes, 33 seconds
      Selection: reoptimation
    Prior LSP:
      ID: path option 10 [1]
      Removal Trigger: configuration changed
```

接下去在 R2 上看一下 Tunnel：

R2#show mpls traffic-eng tunnels

```
LSP Tunnel R1_t0 is signalled, connection is up
  InLabel  : FastEthernet0/0, 201              !!入站标签是 201
```

```
OutLabel : FastEthernet1/0, 301              !!出站标签是 301
RSVP Signalling Info:
      Src 1.1.1.1, Dst 4.4.4.4, Tun_Id 0, Tun_Instance 2
  RSVP Path Info:
    My Address: 10.1.23.2
    Explicit Route: 10.1.23.3 10.1.34.3 10.1.34.4 4.4.4.4
    Record Route:   NONE
    Tspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
  RSVP Resv Info:
    Record Route:   NONE
    Fspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
```

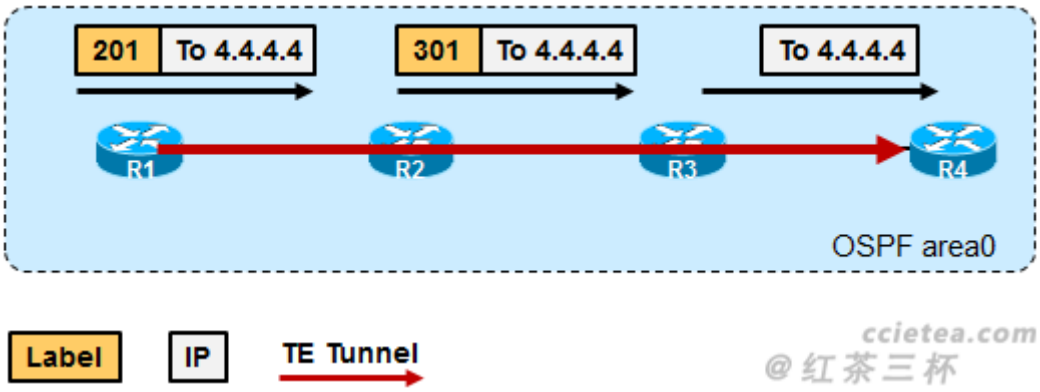
现在我们在 R1 上将其去往 4.4.4.4 的流量引到 tunnel，用一个最简单的方法：静态路由。

```
R1(config)#ip route 4.4.4.4 255.255.255.255    tunnel 0
R1#show ip route
S          4.4.4.4 is directly connected, Tunnel0
```

接下去就可以在 R1 上去测试流量的转发了。

R1#traceroute 4.4.4.4

```
Type escape sequence to abort.
Tracing the route to 4.4.4.4

  1 10.1.12.2 [MPLS: Label 201 Exp 0] 132 msec 184 msec 88 msec
  2 10.1.23.3 [MPLS: Label 301 Exp 0] 124 msec 96 msec 104 msec
  3 10.1.34.4 160 msec *    140 msec
```



继续来 show 一下：

R2#show mpls traffic-eng link-management advertisements

```
Flooding Status:          ready
Configured Areas:         1
IGP Area[1] ID::    ospf area 0
  System Information::
    Flooding Protocol:      OSPF
  Header Information::
    IGP System ID:          2.2.2.2
    MPLS TE Router ID:      2.2.2.2
    Flooded Links:          2
  Link ID::   0                    !! R2 有两条直连链路参与 MPLS TE，这是第一条，也就是 F0/0 口
    Link IP Address:        10.1.12.2
    IGP Neighbor:           ID 10.1.12.2
    TE metric:              1
    IGP metric:             1
    Physical Bandwidth:     100000 kbits/sec        !!接口的物理带宽
    Res. Global BW:         75000 kbits/sec         !!接口的最大可预留带宽
    Res. Sub BW:            0 kbits/sec
    Downstream::
                                    Global Pool     Sub Pool    !!接口为各个优先级 tunnel 的带宽池
                                    -----------     ----------
      Reservable Bandwidth[0]:          75000           0 kbits/sec
      Reservable Bandwidth[1]:          75000           0 kbits/sec
```

```
    Reservable Bandwidth[2]:        75000          0 kbits/sec
    Reservable Bandwidth[3]:        75000          0 kbits/sec
    Reservable Bandwidth[4]:        75000          0 kbits/sec
    Reservable Bandwidth[5]:        75000          0 kbits/sec
    Reservable Bandwidth[6]:        75000          0 kbits/sec
    Reservable Bandwidth[7]:        75000          0 kbits/sec
  Attribute Flags:          0x00000000
Link ID::   1                          !! R2 有两条直连链路参与 MPLS TE，这是第二条
  Link IP Address:          10.1.23.2
  IGP Neighbor:              ID 10.1.23.2
  TE metric:                 1
  IGP metric:                1
  Physical Bandwidth:       100000 kbits/sec
  Res. Global BW:            75000 kbits/sec
  Res. Sub BW:               0 kbits/sec
  Downstream::
                                    Global Pool     Sub Pool
                                    -----------     ----------
    Reservable Bandwidth[0]:        75000          0 kbits/sec
    Reservable Bandwidth[1]:        75000          0 kbits/sec
    Reservable Bandwidth[2]:        75000          0 kbits/sec
    Reservable Bandwidth[3]:        75000          0 kbits/sec
    Reservable Bandwidth[4]:        75000          0 kbits/sec
    Reservable Bandwidth[5]:        75000          0 kbits/sec
    Reservable Bandwidth[6]:        75000          0 kbits/sec
    Reservable Bandwidth[7]:        55000          0 kbits/sec
Attribute Flags:          0x00000000
```

现在我们可以试着修改 R1 的 TE tunnel 配置：

```
Interface tunnel0
 tunnel mpls traffic-eng priority 5 5          !!将 TE tunnel 的建立优先级和保持优先级改为 5
```

然后再去 R2 上看一下：

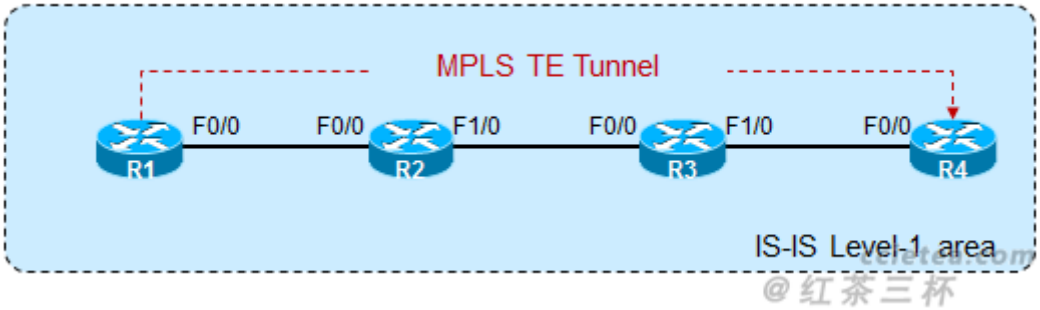
R2#sh mpls tr link-management advertisements

```
Flooding Status:          ready
Configured Areas:         1
IGP Area[1] ID::    ospf area 0
  System Information::
    Flooding Protocol:       OSPF
  Header Information::
    IGP System ID:             2.2.2.2
    MPLS TE Router ID:       2.2.2.2
    Flooded Links:             2
  Link ID::    0
    Link IP Address:          10.1.12.2
    IGP Neighbor:              ID 10.1.12.2
    TE metric:                 1
    IGP metric:                1
    Physical Bandwidth:      100000 kbits/sec
    Res. Global BW:           75000 kbits/sec
    Res. Sub BW:               0 kbits/sec
    Downstream::
                                      Global Pool     Sub Pool
                                      -----------     ----------
      Reservable Bandwidth[0]:          75000               0 kbits/sec
      Reservable Bandwidth[1]:          75000               0 kbits/sec
      Reservable Bandwidth[2]:          75000               0 kbits/sec
      Reservable Bandwidth[3]:          75000               0 kbits/sec
      Reservable Bandwidth[4]:          75000               0 kbits/sec
      Reservable Bandwidth[5]:          75000               0 kbits/sec
      Reservable Bandwidth[6]:          75000               0 kbits/sec
      Reservable Bandwidth[7]:          75000               0 kbits/sec
    Attribute Flags:          0x00000000
  Link ID::    1
    Link IP Address:          10.1.23.2
```

```
    IGP Neighbor:            ID 10.1.23.2
    TE metric:               1
    IGP metric:              1
    Physical Bandwidth:      100000 kbits/sec
    Res. Global BW:          75000 kbits/sec
    Res. Sub BW:             0 kbits/sec
    Downstream::
                                    Global Pool     Sub Pool
                                    -----------     ----------
      Reservable Bandwidth[0]:          75000           0 kbits/sec
      Reservable Bandwidth[1]:          75000           0 kbits/sec
      Reservable Bandwidth[2]:          75000           0 kbits/sec
      Reservable Bandwidth[3]:          75000           0 kbits/sec
      Reservable Bandwidth[4]:          75000           0 kbits/sec
      Reservable Bandwidth[5]:          55000           0 kbits/sec
      Reservable Bandwidth[6]:          55000           0 kbits/sec
      Reservable Bandwidth[7]:          55000           0 kbits/sec
Attribute Flags:         0x00000000
```

我们看到，低于优先级 5 的 tunnel，可用带宽跟着变成了 75-20=55M。

6.4 基础实验 2（IS-IS）



1. **实验环境**
   - 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
   - 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
   - 全网运行 IS-IS，宣告直连和 loopback 接口
   - 在 R1 上建立一个 TE Tunnel，源为自身的 loopback0，目的为 R4 的 4.4.4.4
2. **实验步骤**
   - 完成基本的 IP 配置（配置省略）
   - 所有路由器运行 IS-IS
   - 全局激活 MPLS TE tunnel，并设定 MPLS 标签空间
   - 为每台路由器的 OSPF 激活 MPLS TE 的扩展，并且手工设置用于 MPLS TE 的 RouterID
   - 每台路由器的接口都激活 RSVP 及 MPLS TE tunnel 的支持
   - 在 R1 上完成 MPLS TE tunnel 的配置
   - 测试数据流的传输过程，观察现象

3. **配置命令**

R2 的配置如下（所有配置都省去了接口 IP 的配置）：

```
ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router isis
  net 49.0001.0000.0000.0002.00
  is-type level-1                          !!IS 的类型为 level1
  metric-style wide                        !!metric 必须改为 wide 的
  mpls traffic-eng router-id loopback 0    !!设置 MPLS TE routerID
  mpls traffic-eng level-1                 !!激活 level1 的 MPLS TE
!
interface fast0/0
  ip router isis
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface fast1/0
```

```
 ip router isis
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R3 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router isis
 net 49.0001.0000.0000.0003.00
 is-type level-1
 metric-style wide
 mpls traffic-eng router-id loopback 0
 mpls traffic-eng level-1
!
interface fast0/0
 ip router isis
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface fast1/0
 ip router isis
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R4 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 400 499
!
router isis
 net 49.0001.0000.0000.0004.00
 is-type level-1
 metric-style wide
```

```
 mpls traffic-eng router-id loopback 0
 mpls traffic-eng level-1
!
interface fast0/0
 ip router isis
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface loopback0
 ip router isis
```

最后我们来完成 R1 的配置：

R1 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router isis
 net 49.0001.0000.0000.0001.00
 is-type level-1
 metric-style wide
 mpls traffic-eng router-id loopback 0
 mpls traffic-eng level-1
!
interface fast0/0
 ip router isis
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface loopback0
 ip router isis
!
Interface tunnel0                    !! TE tunnel
 ip unnumbered loopback 0
```

```
tunnel destination 4.4.4.4
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng priority 7 7                    !!tunnel 的建立和保持优先级
tunnel mpls traffic-eng bandwidth 20000                 !!带宽要求为 20M
tunnel mpls traffic-eng path-option 10 dynamic          !!使用动态计算的方式建立 tunnel PATH
```

完成配置后，Tunnel 就起来了。

R1#show isis database verbose R1.00-00

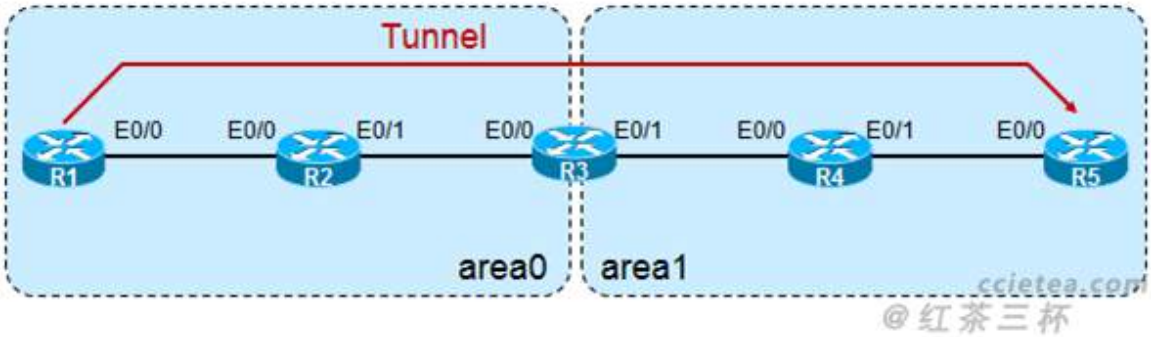
```
IS-IS Level-1 LSP R1.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x0000000C   0x8973        772               0/0/0
  Area Address: 49.0001
  NLPID:         0xCC
  Hostname: R1
  Router ID:     1.1.1.1
  IP Address:    1.1.1.1
  Metric: 10         IP 10.1.12.0/24
  Metric: 10         IP 1.1.1.1/32
  Metric: 10         IS-Extended R2.01
    Affinity: 0x00000000
    Interface IP Address: 10.1.12.1
    Physical BW: 100000 kbits/sec
    Reservable Global Pool BW: 75000 kbits/sec
    Global Pool BW Unreserved:
      [0]:  75000 kbits/sec,  [1]:  75000 kbits/sec
      [2]:  75000 kbits/sec,  [3]:  75000 kbits/sec
      [4]:  75000 kbits/sec,  [5]:  75000 kbits/sec
      [6]:  75000 kbits/sec,  [7]:  55000 kbits/sec
    Admin. Weight: 1
```

R1#show mpls traffic-eng tunnels

```
Name: R1_t0                                (Tunnel0) Destination: 4.4.4.4
  Status:
    Admin: up          Oper: up     Path: valid       Signalling: connected

    path option 10, type dynamic (Basis for Setup, path weight 3)

  Config Parameters:
    Bandwidth: 20000    kbps (Global)  Priority: 7  7   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:  disabled  LockDown: disabled  Loadshare: 20000     bw-based
    auto-bw: disabled

  InLabel  :  -
  OutLabel : FastEthernet0/0, 200
  RSVP Signalling Info:
       Src 1.1.1.1, Dst 4.4.4.4, Tun_Id 0, Tun_Instance 39
    RSVP Path Info:
      My Address: 10.1.12.1
      Explicit Route: 10.1.12.2 10.1.23.2 10.1.23.3 10.1.34.3
                      10.1.34.4 4.4.4.4
      Record Route:   NONE
      Tspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
    RSVP Resv Info:
      Record Route:   NONE
      Fspec: ave rate=20000 kbits, burst=1000 bytes, peak rate=20000 kbits
  History:
    Tunnel:
      Time since created: 5 hours, 7 minutes
      Time since path change: 12 seconds
    Current LSP:
      Uptime: 12 seconds
```

```
    Selection: reoptimation
  Prior LSP:
    ID: path option 10 [36]
    Removal Trigger: configuration changed
```

6.5 跨区域 TE Tunnel（OSPF）



1. **实验环境**
   - 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
   - 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
   - 全网运行 OSPF，宣告直连和 loopback 接口
   - R3 为 ABR，连接 area0 及 area1
   - 在 R1 上建立一个 TE Tunnel，源为自身的 loopback0，目的为 R4 的 4.4.4.4
2. **实验步骤**
   - 完成基本的 IP 配置（配置省略）
   - 所有路由器运行 OSPF
   - 全局激活 MPLS TE tunnel，并设定 MPLS 标签空间
   - 为每台路由器的 OSPF 激活 MPLS TE 的扩展，并且手工设置用于 MPLS TE 的 RouterID
   - 每台路由器的接口都激活 RSVP 及 MPLS TE tunnel 的支持
   - 在 R1 上完成 MPLS TE 的配置
   - 测试数据流的传输过程，观察现象

3. **配置命令**

R2 的配置如下（所有配置都省去了接口 IP 的配置）：

```
ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
   router-id 2.2.2.2
   network 10.1.12.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth0 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth0 1/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R3 的配置如下（注意，area0 和 area1 都要激活 MPLS TE）：

```
ip cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.34.3 0.0.0.0 area 1
   network 10.1.23.2 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
```

```
  mpls traffic-eng area 1                    !!注意 area0 和 area1 都要激活 MPLS TE
!
interface eth0 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface eth0 1/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

R4 的配置如下

```
Ip cef
mpls traffic-eng tunnels
mpls label range 400 499
!
router ospf 1
  router-id 4.4.4.4
  network 10.1.34.4 0.0.0.0 area 1
  network 10.1.45.4 0.0.0.0 area 1
  mpls traffic-eng router-id loopback0
  mpls traffic-eng area 1
!
interface eth0 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface eth0 1/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

R5 的配置如下

```
Ip cef
mpls traffic-eng tunnels
mpls label range 500 599
!
```

```
router ospf 1
   router-id 5.5.5.5
   network 5.5.5.5 0.0.0.0 area 1
   network 10.1.45.5 0.0.0.0 area 1
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 1
!
interface eth0 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

接下去在 R1 上配置：

```
Ip explicit R1toR5
   next-address 10.1.12.2                    !!本区域内直接用 strict 下一跳
   next-address 10.1.23.3
   next-address loose 10.1.34.4              !!跨区域的话必须用 loose 下一跳
   next-address loose 10.1.45.5              !!这条就可要可不要了，你懂的
!
interface Tunnel0
   ip unnumbered Loopback0
   tunnel destination 5.5.5.5
   tunnel mode mpls traffic-eng
   tunnel mpls traffic-eng autoroute announce
   tunnel mpls traffic-eng priority 7 7
   tunnel mpls traffic-eng bandwidth    50000
   tunnel mpls traffic-eng autoroute announce
   tunnel mpls traffic-eng path-option 10 explicit name R1toR5
```

R1#show mpls traffic-eng tunnels

```
Name: R1_t0                                (Tunnel0) Destination: 5.5.5.5
   Status:
     Admin: up          Oper: up       Path: valid          Signalling: connected
```

```
  path option 10, type explicit test (Basis for Setup, path weight 20)

Config Parameters:
  Bandwidth: 0          kbps (Global)  Priority: 7  7   Affinity: 0x0/0xFFFF
  Metric Type: TE (default)
  AutoRoute:  disabled  LockDown: disabled  Loadshare: 0          bw-based
  auto-bw: disabled
Active Path Option Parameters:
  State: explicit path option 10 is active
  BandwidthOverride: disabled  LockDown: disabled  Verbatim: disabled

InLabel  :  -
OutLabel : Ethernet0/0, 201
RSVP Signalling Info:
       Src 1.1.1.1, Dst 5.5.5.5, Tun_Id 0, Tun_Instance 9
  RSVP Path Info:
    My Address: 10.1.12.1
    Explicit Route: 10.1.12.2 10.1.23.2 10.1.23.3 10.1.34.4*
                    10.1.45.5*
    Record   Route:
    Tspec: ave rate=0 kbits, burst=1000 bytes, peak rate=0 kbits
  RSVP Resv Info:
    Record   Route:  10.1.23.2 10.1.34.3 10.1.45.4 10.1.45.5
    Fspec: ave rate=0 kbits, burst=1000 bytes, peak rate=0 kbits
History:
  Tunnel:
    Time since created: 11 minutes, 42 seconds
    Time since path change: 10 seconds
    Number of LSP IDs (Tun_Instances) used: 9
  Current LSP:
    Uptime: 10 seconds
    Selection: reoptimization
```

```
Prior LSP:
  ID: path option 10 [8]
  Removal Trigger: configuration changed
```

7 MPLS TE 路径优化

7.1 TE tunnel reoptimize

- **周期性 reoptimize**

  在 CISCO IOS 中，一条 TE tunnel 的 reoptimize 默认每 1 小时进行一次。

  ```
  router(config)#mpls traffic-eng reoptimize timers frequency ?
  ```

  如果该时间指定为 0，那么周期性 reoptimize 将会在路由器上的所有 TE tunnel 中被关闭。当然，你可以为单条 TE tunnel 关闭 reoptimize，命令如下：

  ```
  Router(config-if)# mpls traffic-eng path-option x {dynamic | explicit name y } [lockdown]
  ```

  使用 lockdown 关键字（注意上述配置是在 TE tunnel 口中配置）。

- **事件导致的 reoptimize**

  缺省情况下，CISCO IOS 不会因为网络中的一条链路重新可以被一条 TE tunnel 所使用的时候而触发重新最优化（例如可用带宽发生变化，这个变化被 IGP 协议泛洪出来了），但是可以激活这个操作。要在一条链路在 MPLS TE 中变为可操作的时候启用重新最优化，使用下面的命令：

  ```
  mpls traffic-eng reoptimize events link-up
  ```

  经实验验证有效。

  可用 debug mpls traffic-eng tunnels events 及 debug mpls traffic-eng tunnels reoptimize 查看。

- **手工重新 reoptimize**

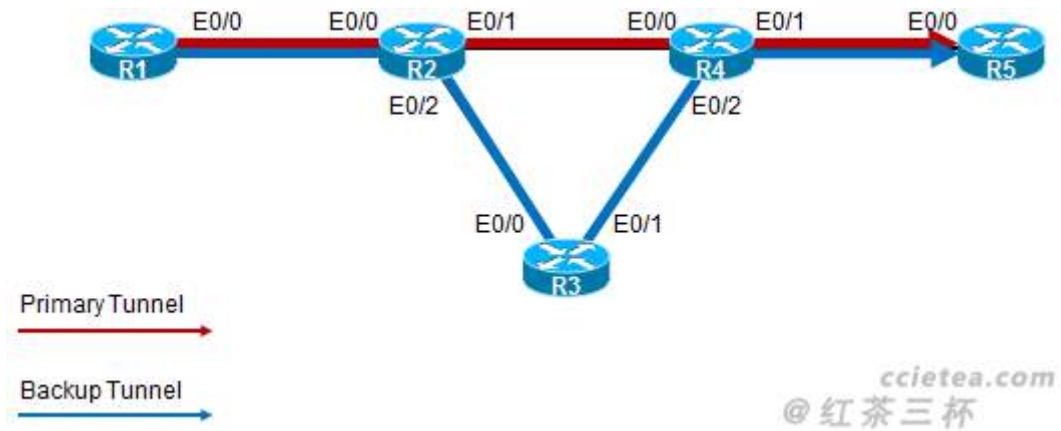
  ```
  Router# mpls traffic-eng reoptimize
  ```

  手工重新最优化

7.2 Path Protection

7.2.1 基本概念

7.2.2 基础实验



1. **实验环境**
   - 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
   - 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
   - 全网运行 OSPF，宣告直连和 loopback 接口
   - 在 R1 上建立一个 TE Tunnel ,路径为 R1-R2-R4-R5 ,再创建另一个备份 Tunnel 路径为 R1-R2-R3-R4-R5，这条路径作为前者的备份 Tunnel。
2. **设备配置**

   R1 的配置如下（暂时不配置 tunnel）：

```
ip cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router ospf 1
```

```
 router-id 1.1.1.1
 network 1.1.1.1 0.0.0.0 area 0
 network 10.1.12.1 0.0.0.0 area 0
 mpls traffic-eng router-id loopback0
 mpls traffic-eng area 0
!
interface eth 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R2 的配置如下

```
lp cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
 router-id 2.2.2.2
 network 10.1.12.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
 network 10.1.24.2 0.0.0.0 area 0
 mpls traffic-eng router-id loopback0
 mpls traffic-eng area 0
!
interface eth 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface eth 0/1
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface eth 0/2
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R3 的配置如下

```
Ip cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R4 的配置如下

```
Ip cef
mpls traffic-eng tunnels
mpls label range 400 499
!
router ospf 1
   router-id 4.4.4.4
   network 10.1.45.4 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

```
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/2
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

R5 的配置如下

```
Ip cef
mpls traffic-eng tunnels
mpls label range 500 599
!
router ospf 1
   router-id 5.5.5.5
   network 10.1.45.5 0.0.0.0 area 0
   network 5.5.5.5 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
```

现在在 R1 上建立 tunnel：

```
ip explicit-path name R2R4 enable                    !! Primary tunnel
 next-address 10.1.12.2
 next-address 10.1.24.4
!
ip explicit-path name R2R3R4 enable                  !! Backup tunnel
 next-address 10.1.12.2
 next-address 10.1.23.3
 next-address 10.1.34.4
!
```

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng path-option 10 explicit name R2R4
 tunnel mpls traffic-eng path-option protect 10 explicit name R2R3R4
!
ip route 5.5.5.5 255.255.255.255 Tunnel0
```

注意，这里主 tunnel 强烈建议是 explicit 的，如果这里主 LSP 是 dynamic 的，那么当主 LSP DOWN 掉后（例如 shutdown 掉 R2 的 E0/1 口），将切换到备份 LSP 上来。但是如果这时候 R2 的 E0/1 口恢复了，经过在 CISCO IOS 路由器上测试，发现流量无法切回主 LSP。用 explici path 则没这个问题。

R1#show mpls traffic-eng tunnels protection

```
R1_t0
  LSP Head, Tunnel0, Admin: up, Oper: up
  Src 1.1.1.1, Dest 5.5.5.5, Instance 30
  Fast Reroute Protection: None
  Path Protection: 2 Common Link(s), 2 Common Node(s)
    Link Sharing Detail:
      P2P Links:                0
      Multiacces Links:         2
          Both interfaces:          2
          1 interface:              0
          0 interfaces:             0 (only media is shared)
    Primary lsp path:10.1.12.1 10.1.12.2
                     10.1.24.2 10.1.24.4
                     10.1.45.4 10.1.45.5
                     5.5.5.5
    Protect lsp path:10.1.12.1 10.1.12.2
                     10.1.23.2 10.1.23.3
                     10.1.34.3 10.1.34.4
                     10.1.45.4 10.1.45.5
```

```
                    5.5.5.5
  Path Protect Parameters:
    Bandwidth: 0          kbps (Global)   Priority: 7   7    Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
  InLabel  :  -
  OutLabel : Ethernet0/0, 204
  RSVP Signalling Info:
       Src 1.1.1.1, Dst 5.5.5.5, Tun_Id 0, Tun_Instance 31
    RSVP Path Info:
      My Address: 10.1.12.1
      Explicit Route: 10.1.12.2 10.1.23.2 10.1.23.3 10.1.34.3
                      10.1.34.4 10.1.45.4 10.1.45.5 5.5.5.5
      Record   Route:   NONE
      Tspec: ave rate=0 kbits, burst=1000 bytes, peak rate=0 kbits
    RSVP Resv Info:
      Record   Route:   NONE
      Fspec: ave rate=0 kbits, burst=1000 bytes, peak rate=0 kbits
```

7.3 Fast Reroute

7.3.1 FRR 基本概念

- 如果 link 或 node 出现故障，流量会因为故障而被重新进行路由，如果重新路由需要花上几秒钟的话，由于链路的高性能会导致有大量指向该故障点的流量都会被丢弃，对于某些业务来说可能会导致业务中断。
- **Fast Reroute（FRR）**是一种 link 或 node 保护机制。允许在 link 或 node 出现问题的时候，头端路由器 rerouting，并且使用一条临时的、预先建立好的路径绕过故障点。

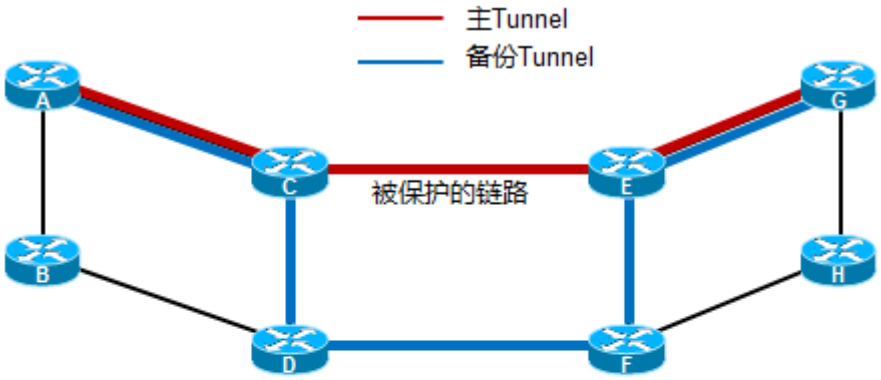
按保护对象可分为：

- Link Protection（链路保护）

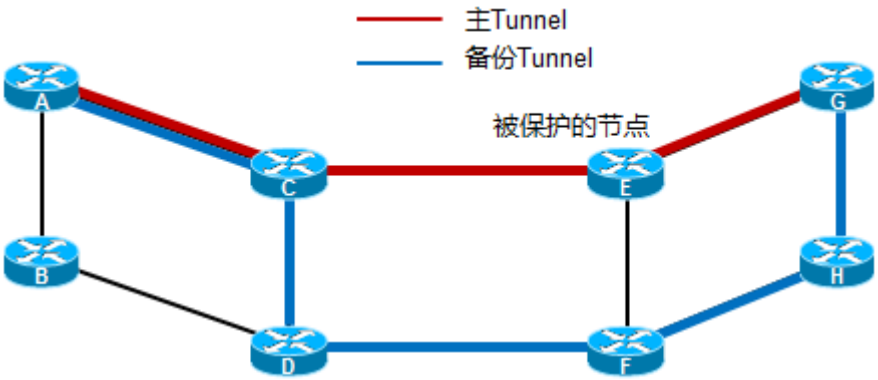
- Node Protction（节点保护）

7.3.2 术语

- PLR（Point of local repair）: 本地修理点，这是备份隧道的首端
- MP（Merge Point）: 汇合点，这是备份隧道的尾端，也可以理解为主隧道和备份隧道交接点
- NHop（Next Hop Router）: 下一跳路由器，PLR 的下一跳路由器
- NNHop（Next-Next Hop Router）: 下下一跳路由器，PLR 下一跳路由器的下一跳

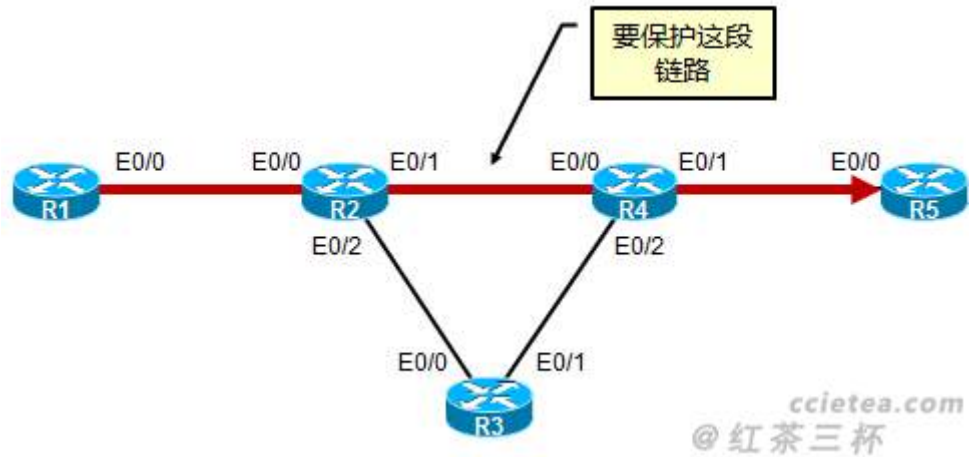


- C是备份Tunnel的PLR（Point of local repair）
- E是备份Tunnel的MP（Merge Point）
- E是C的NHop（Next-hop）
- G是C的NNHop（Next-Next-Hop）



- C是备份Tunnel的PLR（Point of local repair）
- G是备份Tunnel的MP（Merge Point）
- E是C的NHop（Next-hop）
- G是C的NNHop（Next-Next-Hop）

7.3.3 基础实验（LinkProtection）



1. **环境描述**
   - 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
   - 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
   - 全网运行 OSPF，宣告直连和 loopback 接口
   - 在 R1 上建立一个 TE Tunnel，源为自身的 loopback0，目的为 R5 的 5.5.5.5

2. **实验步骤**

- 完成基本的 IP 配置（配置省略）
- 所有路由器运行 OSPF
- 全局激活 MPLS TE tunnel，并设定 MPLS 标签空间
- 为每台路由器的 OSPF 激活 MPLS TE 的扩展，并且手工设置用于 MPLS TE 的 RouterID
- 每台路由器的接口都激活 RSVP 及 MPLS TE tunnel 的支持
- 在 R1 上完成 MPLS TE 的配置
- Shutdown 掉 R2 的 e0/1 口，在 R1 上抓取 debug 信息
- 在 R1 上激活 fast-reroute 特性，再观察现象
- 在 R2 上配置路径保护，再观察现象

3. **设备配置**

R1 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 100 199
!
router ospf 1
   router-id 1.1.1.1
   network 1.1.1.1 0.0.0.0 area 0
   network 10.1.12.1 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
!
ip explicit-path name R2R4R5 enable
 next-address 10.1.12.2
 next-address 10.1.24.4
 next-address 10.1.45.5
!
```

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 2000
 tunnel mpls traffic-eng path-option 10 explicit name R2R4R5
 no routing dynamic
```

R2 的配置如下：

```
Ip cef
mpls traffic-eng tunnels
mpls label range 200 299
!
router ospf 1
   router-id 2.2.2.2
   network 2.2.2.2 0.0.0.0 area0
   network 10.1.12.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   network 10.1.24.2 0.0.0.0 area 0
   mpls traffic-eng router-id loopback0
   mpls traffic-eng area 0
!
interface eth 0/0
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/1
   mpls traffic-eng tunnels
   ip rsvp bandwidth
interface eth 0/2
   mpls traffic-eng tunnels
```

```
 ip rsvp bandwidth
```

R3 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 300 399
!
router ospf 1
 router-id 3.3.3.3
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
 mpls traffic-eng router-id loopback0
 mpls traffic-eng area 0
!
interface eth 0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth
interface eth 0/1
 mpls traffic-eng tunnels
 ip rsvp bandwidth
```

R4 的配置如下：

```
lp cef
mpls traffic-eng tunnels
mpls label range 400 499
!
router ospf 1
 router-id 4.4.4.4
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
 network 10.1.24.4 0.0.0.0 area 0
 network 10.1.45.4 0.0.0.0 area 0
```

```
  mpls traffic-eng router-id loopback0
  mpls traffic-eng area 0
!
interface eth 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface eth 0/1
  mpls traffic-eng tunnels
  ip rsvp bandwidth
interface eth 0/2
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

R5 的配置如下：

```
Ip cef
mpls traffic-eng tunnels
mpls label range 500 599
!
router ospf 1
  router-id 5.5.5.5
  network 5.5.5.5 0.0.0.0 area 0
  network 10.1.45.5 0.0.0.0 area 0
  mpls traffic-eng router-id loopback0
  mpls traffic-eng area 0
!
interface eth 0/0
  mpls traffic-eng tunnels
  ip rsvp bandwidth
```

4. **现象观察**

初始化配置如上，在现在的环境中，R1 并没有激活 fastreroute，而且 R2 上也没有去部署保护性的 tunnel。

首先在 R1 上 debug ip rsvp dump-messages

然后 shutdown R2 的 e0/1 口：

R1#deb ip rsvp dump-messages

***Aug 18 04:37:06.239: Incoming PathError: !!R2 发给 R1 的 patherror 消息**
*Aug 18 04:37:06.239: version:1 flags:0000 cksum:7B25 ttl:255 reserved:0 length:132
*Aug 18 04:37:06.239: SESSION type 7 length 16:
*Aug 18 04:37:06.239: Tun Dest: 5.5.5.5 Tun ID: 0 Ext Tun ID: 1.1.1.1
*Aug 18 04:37:06.239: ERROR_SPEC type 1 length 12:
*Aug 18 04:37:06.239: **Error Node: 10.1.12.2**
*Aug 18 04:37:06.239: Error Code: 24 (Routing Problem)
*Aug 18 04:37:06.239: Error Value: 0x5 **(No route available toward destination)**
*Aug 18 04:37:06.239: Flags: 0x0
*Aug 18 04:37:06.239: SENDER_TEMPLATE type 7 length 12:
*Aug 18 04:37:06.239: Tun Sender: 1.1.1.1 LSP ID: 20
*Aug 18 04:37:06.239: SENDER_TSPEC type 2 length 36:
*Aug 18 04:37:06.239: version=0, length in words=7
*Aug 18 04:37:06.239: Token bucket fragment (service_id=1, length=6 words
*Aug 18 04:37:06.239: parameter id=127, flags=0, parameter length=5
*Aug 18 04:37:06.239: average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 04:37:06.239: peak rate =250000 bytes/sec
*Aug 18 04:37:06.239: min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 04:37:06.239: ADSPEC type 2 length 48:
*Aug 18 04:37:06.239: version=0 length in words=10
*Aug 18 04:37:06.239: General Parameters break bit=0 service length=8
*Aug 18 04:37:06.239: IS Hops:1
*Aug 18 04:37:06.239: Minimum Path Bandwidth (bytes/sec):1250000
*Aug 18 04:37:06.239: Path Latency (microseconds):0
*Aug 18 04:37:06.239: Path MTU:1500
*Aug 18 04:37:06.239: Controlled Load Service break bit=0 service length=0
*Aug 18 04:37:06.239:
***Aug 18 04:37:06.239: Outgoing PathTear: !!拆除 tunnel**
*Aug 18 04:37:06.239: version:1 flags:0000 cksum:DEB3 ttl:255 reserved:0 length:132
*Aug 18 04:37:06.239: SESSION type 7 length 16:

```
*Aug 18 04:37:06.239:     Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 04:37:06.239:    HOP                          type 1 length 12:
*Aug 18 04:37:06.239:     Hop Addr: 10.1.12.1 LIH: 0x02000403
*Aug 18 04:37:06.239:    SENDER_TEMPLATE          type 7 length 12:
*Aug 18 04:37:06.239:     Tun Sender: 1.1.1.1    LSP ID: 20
......  .............  ........................
*Aug 18 04:37:06.239:
*Aug 18 04:37:06.243: Incoming ResvTear:
*Aug 18 04:37:06.243:     version:1 flags:0000 cksum:FFAA ttl:255 reserved:0 length:92
*Aug 18 04:37:06.243:    SESSION                      type 7 length 16:
*Aug 18 04:37:06.243:     Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 04:37:06.243:    HOP                          type 1 length 12:
*Aug 18 04:37:06.243:     Hop Addr: 10.1.12.2 LIH: 0x02000403
*Aug 18 04:37:06.243:    STYLE                        type 1 length 8 :
*Aug 18 04:37:06.243:     Shared-Explicit (SE)
*Aug 18 04:37:06.243:    FLOWSPEC                     type 2 length 36:
*Aug 18 04:37:06.243:        version = 0 length in words = 7
*Aug 18 04:37:06.243:        service id = 5, service length = 6
*Aug 18 04:37:06.243:        tspec parameter id = 127, flags = 0,length = 5
*Aug 18 04:37:06.243:        average rate = 250000 bytes/sec, burst depth = 1000 bytes
*Aug 18 04:37:06.243:        peak rate       = 250000 bytes/sec
*Aug 18 04:37:06.243:        min unit = 0 bytes,max pkt size = 1500 bytes
*Aug 18 04:37:06.243:    FILTER_SPEC                  type 7 length 12:
*Aug 18 04:37:06.243:     Tun Sender: 1.1.1.1, LSP ID: 20
*Aug 18 04:37:06.243:
R1#
*Aug 18 04:37:24.735: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
down
```

现在激活 R1 的 fast-reroute 特性，修改 R1 的 Tunnel 口配置如下：

```
interface Tunnel0
 ip unnumbered Loopback0
```

```
tunnel mode mpls traffic-eng
tunnel destination 5.5.5.5
tunnel mpls traffic-eng autoroute announce
tunnel mpls traffic-eng priority 7 7
tunnel mpls traffic-eng bandwidth 2000
tunnel mpls traffic-eng path-option 10 explicit name R2R4R5
tunnel mpls traffic-eng fast-reroute
```

R1 上 debug 信息如下：

```
*Aug 18 09:06:02.699: Outgoing Path:      !! R1 发送出去的 path 消息发生了变化
*Aug 18 09:06:02.699:    version:1 flags:0000 cksum:5878 ttl:255 reserved:0 length:216
*Aug 18 09:06:02.699:   SESSION                  type 7 length 16:
*Aug 18 09:06:02.699:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 09:06:02.699:   HOP                      type 1 length 12:
*Aug 18 09:06:02.699:    Hop Addr: 10.1.12.1 LIH: 0x20000403
*Aug 18 09:06:02.699:   TIME_VALUES              type 1 length 8 :
*Aug 18 09:06:02.699:    Refresh Period (msec): 30000
*Aug 18 09:06:02.699:   EXPLICIT_ROUTE           type 1 length 52:
*Aug 18 09:06:02.699:    10.1.12.2 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:    10.1.24.2 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:    10.1.24.4 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:    10.1.45.4 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:    10.1.45.5 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:    5.5.5.5 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:02.699:   LABEL_REQUEST            type 1 length 8 :
*Aug 18 09:06:02.699:    Layer 3 protocol ID: 2048
*Aug 18 09:06:02.699:   SESSION_ATTRIBUTE        type 7 length 16:
*Aug 18 09:06:02.699:    Setup Prio: 7, Holding Prio: 7
*Aug 18 09:06:02.699:    Flags: (0x7) Local Prot desired, Label Recording, SE Style      !! Flag 位里有体现了
*Aug 18 09:06:02.699:    Session Name: R1_t0
*Aug 18 09:06:02.699:   SENDER_TEMPLATE          type 7 length 12:
```

```
*Aug 18 09:06:02.699:    Tun Sender: 1.1.1.1    LSP ID: 247
*Aug 18 09:06:02.699:   SENDER_TSPEC              type 2 length 36:
*Aug 18 09:06:02.699:    version=0, length in words=7
*Aug 18 09:06:02.699:    Token bucket fragment (service_id=1, length=6 words
*Aug 18 09:06:02.699:      parameter id=127, flags=0, parameter length=5
*Aug 18 09:06:02.699:      average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 09:06:02.699:      peak rate     =250000 bytes/sec
*Aug 18 09:06:02.699:      min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 09:06:02.699:   ADSPEC                    type 2 length 48:
*Aug 18 09:06:02.699:   version=0    length in words=10
*Aug 18 09:06:02.699:   General Parameters    break bit=0    service length=8
*Aug 18 09:06:02.699:                                              IS Hops:1
*Aug 18 09:06:02.699:                    Minimum Path Bandwidth (bytes/sec):1250000
*Aug 18 09:06:02.699:                              Path Latency (microseconds):0
*Aug 18 09:06:02.699:                                              Path MTU:1500
*Aug 18 09:06:02.699:   Controlled Load Service    break bit=0    service length=0
*Aug 18 09:06:02.699:
*Aug 18 09:06:07.919: Incoming PathError:
*Aug 18 09:06:07.919:    version:1 flags:0000 cksum:7A42 ttl:255 reserved:0 length:132
*Aug 18 09:06:07.919:   SESSION                   type 7 length 16:
*Aug 18 09:06:07.919:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 09:06:07.919:   ERROR_SPEC                type 1 length 12:
*Aug 18 09:06:07.919:     Error Node: 10.1.12.2
*Aug 18 09:06:07.919:     Error Code: 24 (Routing Problem)
*Aug 18 09:06:07.919:     Error Value: 0x5    (No route available toward destination)
*Aug 18 09:06:07.919:    Flags: 0x0
*Aug 18 09:06:07.919:   SENDER_TEMPLATE          type 7 length 12:
*Aug 18 09:06:07.919:    Tun Sender: 1.1.1.1    LSP ID: 247
*Aug 18 09:06:07.919:   SENDER_TSPEC              type 2 length 36:
*Aug 18 09:06:07.919:    version=0, length in words=7
*Aug 18 09:06:07.919:    Token bucket fragment (service_id=1, length=6 words
*Aug 18 09:06:07.919:      parameter id=127, flags=0, parameter length=5
```

```
*Aug 18 09:06:07.919:       average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 09:06:07.919:       peak rate     =250000 bytes/sec
*Aug 18 09:06:07.919:       min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 09:06:07.919:   ADSPEC                      type 2 length 48:
*Aug 18 09:06:07.919:   version=0   length in words=10
*Aug 18 09:06:07.919:   General Parameters   break bit=0   service length=8
*Aug 18 09:06:07.919:                                                IS Hops:1
*Aug 18 09:06:07.919:                   Minimum Path Bandwidth (bytes/sec):1250000
*Aug 18 09:06:07.919:                            Path Latency (microseconds):0
*Aug 18 09:06:07.919:                                                Path MTU:1500
*Aug 18 09:06:07.919:   Controlled Load Service   break bit=0   service length=0
*Aug 18 09:06:07.919:
*Aug 18 09:06:07.919: Incoming PROXY_PATH:
*Aug 18 09:06:07.919:    version:1 flags:0000 cksum:0000 ttl:255 reserved:0 length:232
*Aug 18 09:06:07.919:   SESSION                     type 7 length 16:
*Aug 18 09:06:07.919:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 09:06:07.919:   HOP                         type 1 length 12:
*Aug 18 09:06:07.919:    Hop Addr: 127.0.0.1 LIH: 0x00000000
*Aug 18 09:06:07.919:   TIME_VALUES                 type 1 length 8 :
*Aug 18 09:06:07.919:    Refresh Period (msec): 30000
*Aug 18 09:06:07.919:   EXPLICIT_ROUTE              type 1 length 68:
*Aug 18 09:06:07.919:    1.1.1.1 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.12.1 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.12.2 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.24.2 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.24.4 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.45.4 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    10.1.45.5 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:    5.5.5.5 (Strict IPv4 Prefix, 8 bytes, /32)
*Aug 18 09:06:07.919:   SESSION_ATTRIBUTE       type 7 length 16:
*Aug 18 09:06:07.919:    Setup Prio: 7, Holding Prio: 7
*Aug 18 09:06:07.919:    Flags: (0x7) Local Prot desired, Label Recording, SE Style
```

```
*Aug 18 09:06:07.919:    Session Name: R1_t0
*Aug 18 09:06:07.919:  SENDER_TEMPLATE          type 7 length 12:
*Aug 18 09:06:07.919:    Tun Sender: 1.1.1.1    LSP ID: 248
*Aug 18 09:06:07.919:  SENDER_TSPEC             type 2 length 36:
*Aug 18 09:06:07.919:    version=0, length in words=7
*Aug 18 09:06:07.919:    Token bucket fragment (service_id=1, length=6 words
*Aug 18 09:06:07.919:      parameter id=127, flags=0, parameter length=5
*Aug 18 09:06:07.919:      average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 09:06:07.919:      peak rate     =250000 bytes/sec
*Aug 18 09:06:07.919:      min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 09:06:07.919:  ADSPEC                   type 2 length 48:
*Aug 18 09:06:07.919:  version=0    length in words=10
*Aug 18 09:06:07.919:  General Parameters    break bit=0    service length=8
*Aug 18 09:06:07.919:                                          IS Hops:0
*Aug 18 09:06:07.919:                  Minimum Path Bandwidth (bytes/sec):2147483647
*Aug 18 09:06:07.919:                           Path Latency (microseconds):0
*Aug 18 09:06:07.919:                                        Path MTU:4294967295
*Aug 18 09:06:07.919:  Controlled Load Service    break bit=0    service length=0
*Aug 18 09:06:07.919:  LABEL_REQUEST            type 1 length 8 :
*Aug 18 09:06:07.919:    Layer 3 protocol ID: 2048
*Aug 18 09:06:07.919:
*Aug 18 09:06:07.919: Outgoing PathTear:
*Aug 18 09:06:07.919:    version:1 flags:0000 cksum:BFD0 ttl:255 reserved:0 length:132
*Aug 18 09:06:07.919:  SESSION                  type 7 length 16:
*Aug 18 09:06:07.919:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 09:06:07.919:  HOP                      type 1 length 12:
*Aug 18 09:06:07.919:    Hop Addr: 10.1.12.1 LIH: 0x20000403
*Aug 18 09:06:07.919:  SENDER_TEMPLATE          type 7 length 12:
*Aug 18 09:06:07.919:    Tun Sender: 1.1.1.1    LSP ID: 247
*Aug 18 09:06:07.919:  SENDER_TSPEC             type 2 length 36:
*Aug 18 09:06:07.919:    version=0, length in words=7
*Aug 18 09:06:07.919:    Token bucket fragment (service_id=1, length=6 words
```

```
*Aug 18 09:06:07.919:        parameter id=127, flags=0, parameter length=5
*Aug 18 09:06:07.919:        average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 09:06:07.919:        peak rate      =250000 bytes/sec
*Aug 18 09:06:07.919:        min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 09:06:07.919:    ADSPEC                          type 2 length 48:
*Aug 18 09:06:07.919:    version=0    length in words=10
*Aug 18 09:06:07.919:    General Parameters    break bit=0    service length=8
*Aug 18 09:06:07.919:                                                  IS Hops:0
*Aug 18 09:06:07.919:                    Minimum Path Bandwidth (bytes/sec):2147483647
*Aug 18 09:06:07.919:                              Path Latency (microseconds):0
*Aug 18 09:06:07.919:                                                  Path MTU:4294967295
*Aug 18 09:06:07.919:    Controlled Load Service    break bit=0    service length=0
*Aug 18 09:06:07.919:
```

在 R2 配置备份链路，R2 上增加配置如下：

```
ip explicit-path name R3R4 enable
 next-address 10.1.23.3
 next-address 10.1.34.4
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 4.4.4.4
 tunnel mpls traffic-eng path-option 10 explicit name R3R4
!
Interface eth0/1
  mpls traffic-eng backup-path tunnel0
```

从 R1 的 debug 信息可以看出：

```
*Aug 18 11:26:02.546: Incoming Resv:      !! R2 发给 R1 的 resv 消息
*Aug 18 11:26:02.546:     version:1 flags:0000 cksum:CA2B ttl:255 reserved:0 length:160
*Aug 18 11:26:02.546:    SESSION                         type 7 length 16:
```

```
*Aug 18 11:26:02.546:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 11:26:02.546:   HOP                          type 1 length 12:
*Aug 18 11:26:02.546:    Hop Addr: 10.1.12.2 LIH: 0x0D000408
*Aug 18 11:26:02.546:   TIME_VALUES                  type 1 length 8 :
*Aug 18 11:26:02.546:    Refresh Period (msec): 30000
*Aug 18 11:26:02.546:   STYLE                        type 1 length 8 :
*Aug 18 11:26:02.546:    Shared-Explicit (SE)
*Aug 18 11:26:02.546:   FLOWSPEC                     type 2 length 36:
*Aug 18 11:26:02.546:       version = 0 length in words = 7
*Aug 18 11:26:02.546:       service id = 5, service length = 6
*Aug 18 11:26:02.546:       tspec parameter id = 127, flags = 0,length = 5
*Aug 18 11:26:02.546:       average rate = 250000 bytes/sec, burst depth = 1000 bytes
*Aug 18 11:26:02.546:       peak rate       = 250000 bytes/sec
*Aug 18 11:26:02.546:       min unit = 0 bytes,max pkt size = 1500 bytes
*Aug 18 11:26:02.546:   FILTER_SPEC                  type 7 length 12:
*Aug 18 11:26:02.546:    Tun Sender: 1.1.1.1, LSP ID: 521
*Aug 18 11:26:02.546:   LABEL                        type 1 length 8 :
*Aug 18 11:26:02.546:    Labels: 200
*Aug 18 11:26:02.546:   RECORD_ROUTE                 type 1 length 52:
*Aug 18 11:26:02.546:     2.2.2.2/32, Flags:0x21 (Local Prot Avail/to NHOP, Node-id)
!!  这里也发生了变化
*Aug 18 11:26:02.546:       Label subobject: Flags 0x1, C-Type 1, Label 200
*Aug 18 11:26:02.546:     4.4.4.4/32, Flags:0x20 (No Local Protection, Node-id)
*Aug 18 11:26:02.546:       Label subobject: Flags 0x1, C-Type 1, Label 403
*Aug 18 11:26:02.546:     5.5.5.5/32, Flags:0x20 (No Local Protection, Node-id)
*Aug 18 11:26:02.546:       Label subobject: Flags 0x1, C-Type 1, Label 0
*Aug 18 11:26:02.546:
```

上面是正常情况下的 debug 信息。

R1#show mpls traffic-eng tunnels

```
Name: R1_t0                                (Tunnel0) Destination: 5.5.5.5
  Status:
```

```
    Admin: up              Oper: up        Path: valid          Signalling: connected
    path option 10, type explicit R2R4R5 (Basis for Setup, path weight 30)

  Config Parameters:
    Bandwidth: 2000        kbps (Global)   Priority: 7   7     Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:   enabled     LockDown: disabled   Loadshare: 2000        bw-based
    auto-bw: disabled
  Active Path Option Parameters:
    State: explicit path option 10 is active
    BandwidthOverride: disabled   LockDown: disabled   Verbatim: disabled

  InLabel   :   -
  OutLabel : Ethernet0/0, 200
  RSVP Signalling Info:
         Src 1.1.1.1, Dst 5.5.5.5, Tun_Id 0, Tun_Instance 521
    RSVP Path Info:
      My Address: 10.1.12.1
      Explicit Route: 10.1.12.2 10.1.24.2 10.1.24.4 10.1.45.4
                      10.1.45.5 5.5.5.5
      Record     Route:     NONE
      Tspec: ave rate=2000 kbits, burst=1000 bytes, peak rate=2000 kbits
    RSVP Resv Info:
      Record     Route:   2.2.2.2(200) 4.4.4.4(403)
                          5.5.5.5(0)
!! 发生变化了，在 head 路由器这，会记录沿路的标签。
      Fspec: ave rate=2000 kbits, burst=1000 bytes, peak rate=2000 kbits
  History:
    Tunnel:
      Time since created: 5 hours, 38 minutes
      Time since path change: 10 minutes, 34 seconds
      Number of LSP IDs (Tun_Instances) used: 521
```

```
    Current LSP:
      Uptime: 5 minutes, 53 seconds
    Prior LSP:
      ID: path option 10 [248]
      Removal Trigger: re-route path verification failed
R1#
```

现在，将 R2 的 e0/1 口 shutdown 掉。

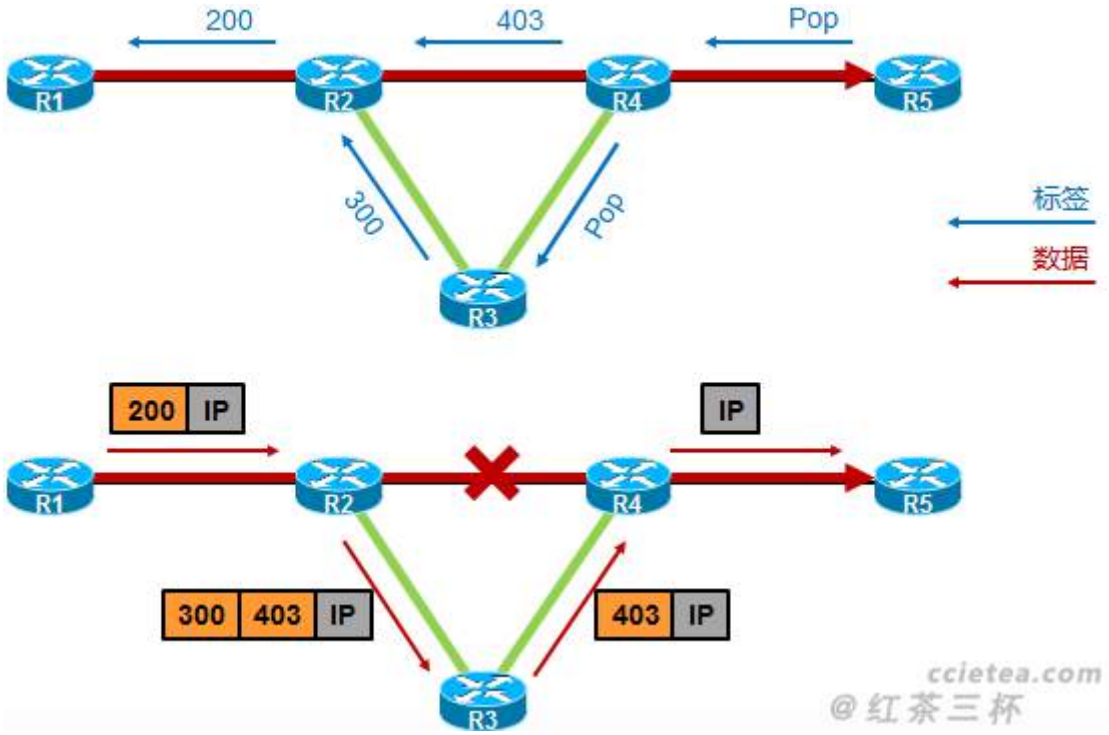
R1 上的 debug 信息

```
*Aug 18 11:31:44.598: Incoming PathError:          !! R2 发给 R1 的 Patherror 消息
*Aug 18 11:31:44.598:    version:1 flags:0000 cksum:7931 ttl:255 reserved:0 length:132
*Aug 18 11:31:44.598:   SESSION                    type 7 length 16:
*Aug 18 11:31:44.598:    Tun Dest:     5.5.5.5    Tun ID: 0    Ext Tun ID: 1.1.1.1
*Aug 18 11:31:44.598:   ERROR_SPEC                 type 1 length 12:
*Aug 18 11:31:44.598:    Error Node: 10.1.12.2
*Aug 18 11:31:44.598:    Error Code: 25 (Notify)
*Aug 18 11:31:44.598:     Error Value: 0x3    (Tunnel locally repaired)
                        !!密切注意这里
*Aug 18 11:31:44.598:    Flags: 0x0
*Aug 18 11:31:44.598:   SENDER_TEMPLATE          type 7 length 12:
*Aug 18 11:31:44.598:    Tun Sender: 1.1.1.1    LSP ID: 521
*Aug 18 11:31:44.598:   SENDER_TSPEC             type 2 length 36:
*Aug 18 11:31:44.598:    version=0, length in words=7
*Aug 18 11:31:44.598:    Token bucket fragment (service_id=1, length=6 words
*Aug 18 11:31:44.598:      parameter id=127, flags=0, parameter length=5
*Aug 18 11:31:44.598:      average rate=250000 bytes/sec, burst depth=1000 bytes
*Aug 18 11:31:44.598:      peak rate     =250000 bytes/sec
*Aug 18 11:31:44.598:      min unit=0 bytes, max pkt size=2147483647 bytes
*Aug 18 11:31:44.598:   ADSPEC                     type 2 length 48:
*Aug 18 11:31:44.598:   version=0    length in words=10
*Aug 18 11:31:44.598:   General Parameters    break bit=0    service length=8
```

```
*Aug 18 11:31:44.598:                                    IS Hops:1
*Aug 18 11:31:44.598:          Minimum Path Bandwidth (bytes/sec):1250000
*Aug 18 11:31:44.598:                  Path Latency (microseconds):0
*Aug 18 11:31:44.598:                                    Path MTU:1500
*Aug 18 11:31:44.598:  Controlled Load Service    break bit=0    service length=0
*Aug 18 11:31:44.598:
```

R1# traceroute 5.5.5.5

```
Type escape sequence to abort.
Tracing the route to 5.5.5.5
  1 10.1.12.2 [MPLS: Label 200 Exp 0] 4 msec 0 msec 0 msec
  2 10.1.23.3 [MPLS: Labels 300/403 Exp 0] 0 msec 4 msec 0 msec
  3 10.1.34.4 [MPLS: Label 403 Exp 0] 0 msec 0 msec 4 msec
  4 10.1.45.5 0 msec
```



由于我们在 R2 上部署了 link 保护，保护的是 R2-R4 之间的链路，因此当 shutdown 掉 R2 的 E0/1 口后，R2 发送给 R1 的 Patherror 消息中，提示 R1 并不需要去拆掉 tunnel。所以 R1 的 tunnel 不会拆除。

并且在 R2 这里，标签包被压入了新的一层标签，外层标签是 R3 分配的，内层标签是 R4 分配的。

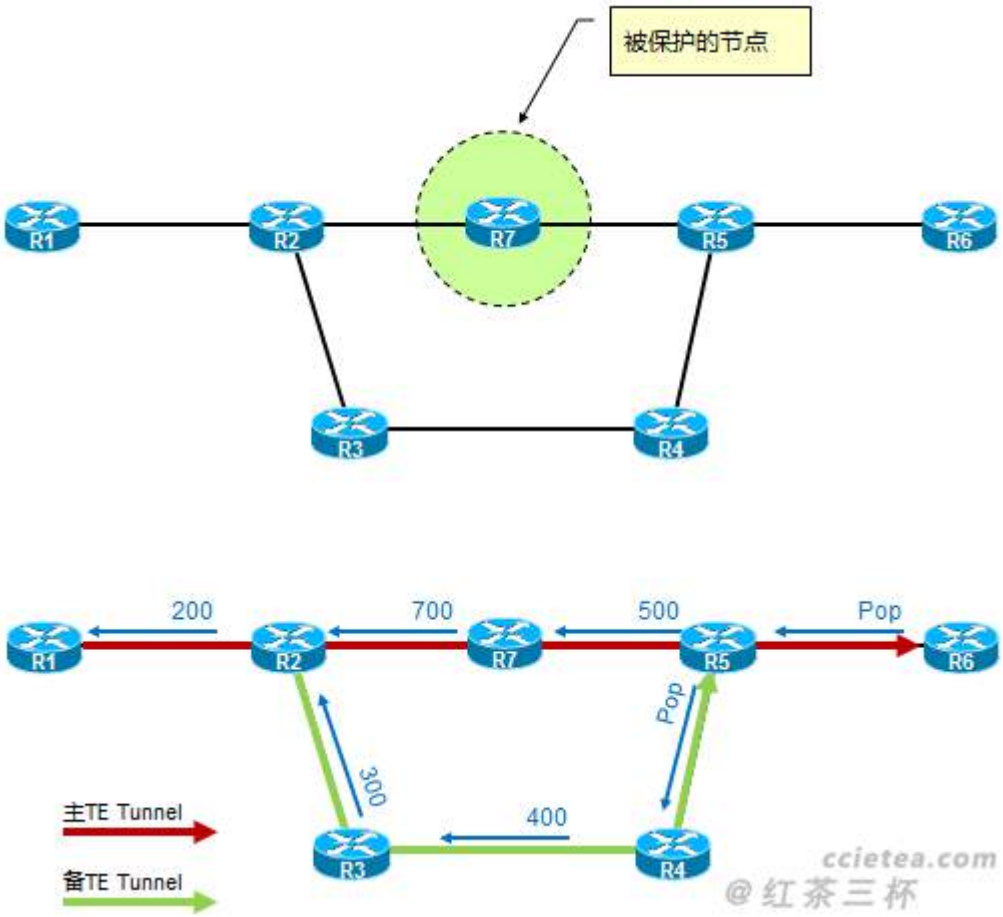
这时候再在 R2 上看一下：

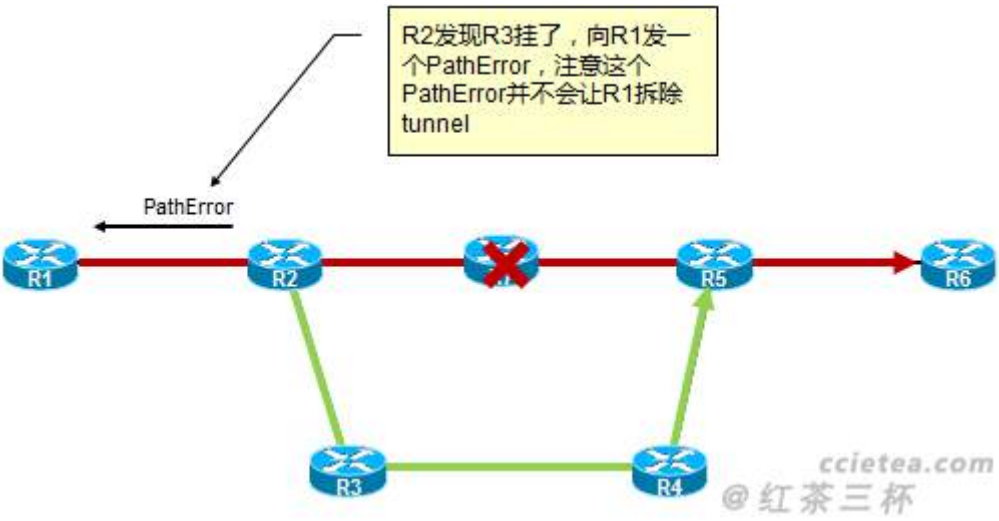
R2#show mpls traffic-eng fast-reroute database

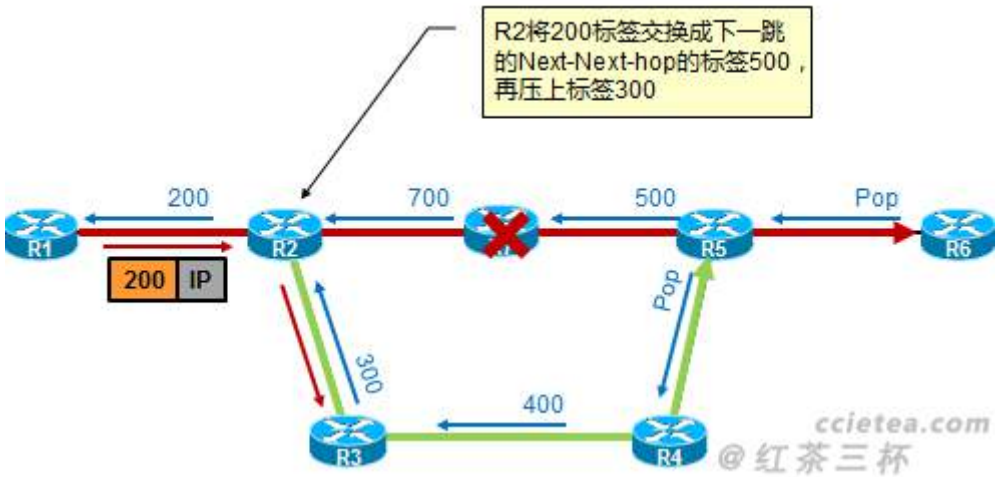
```
Headend frr information:
Protected tunnel                  In-label Out intf/label     FRR intf/label     Status

LSP midpoint frr information:
LSP identifier                    In-label Out intf/label     FRR intf/label          Status
1.1.1.1 0 [521]                   200      Et0/1:403          Tu0:403                 active
```

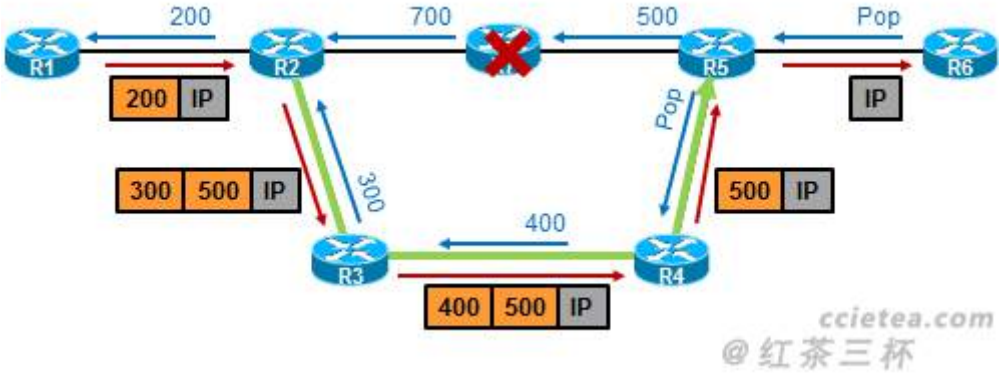
这里状态为 active。当主链路没有失效时，status 显示为 ready

7.3.4 基础实验（NodeProtection）




R2发现R3挂了，向R1发一个PathError，注意这个PathError并不会让R1拆除tunnel
PathError
R1
R2
R5
R6
R3
R4
ccietea.com
@红茶三杯


R2将200标签交换成下一跳的Next-Next-hop的标签500，再压上标签300
200
700
500
Pop
R1
R2
R5
R6
200 IP
Pop
300
400
R3
R4
ccietea.com
@红茶三杯


200
700
500
Pop
R1
R2
R5
R6
200 IP
IP
Pop
300 500 IP
300
500 IP
400
R3
R4
400 500 IP
ccietea.com
@红茶三杯

7.4 AutoBandwidth

Traffic Engineering (TE) automatic bandwidth feature adjusts the bandwidth allocation for TE tunnels based on their measured traffic load:

- It periodically changes tunnel bandwidth(BW) reservation based on traffic out tunnel.
- 基于每个 tunnel 测量平均 output 速率
- The allocated bandwidth is periodically adjusted to be the largest sample for the tunnel since the last adjustment

```
mpls traffic-eng auto-bw timers frequency 300
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng path-option 10 dynamic
 tunnel mpls traffic-eng bandwidth 2500
 tunnel mpls traffic-eng auto-bw frequency 3600 max-bw 3000 min-bw 1000
```

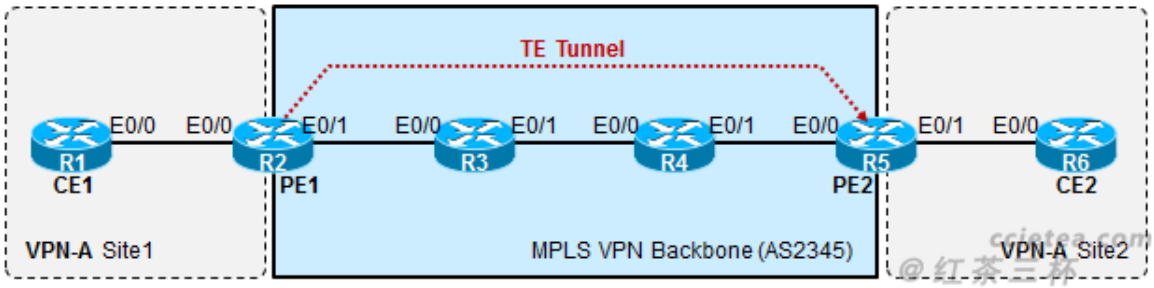
初始情况下tunnel的保证带宽为2.5M

每300S采样一次，取3600S内采样的带宽最大值。

无论采样的结果如何，带宽最小值保证为1M，最大为3M

8 MPLS TE and MPLS VPN

8.1 实验 1



1. 实验环境

- 设备互联网段为 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大
- 所有设备的 loopback0 地址空间为 x.x.x.x/32，x 为设备编号
- R2、R3、R4、R5 为 MPLS VPN Backbone 路由器，运行的 Backbone IGP 协议是 OSPF，进程号使用 100。R2 及 R5 之间建立 MP-iBGP 连接。
- PE-CE 间运行的 VRF IGP 为 OSPF，使用进程号 1
- MPLS VPN Backbone 内进行针对 MPLS TE 的 OSPF 扩展，并在 R1 上建立一个 TE Tunnel，源为自身的 loopback0，目的为 R5 的 5.5.5.5

2. 实验配置

R1 的配置如下：

```
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Ethernet0/0
 ip address 10.1.12.1 255.255.255.0
!
router ospf 1
 router-id 1.1.1.1
 network 1.1.1.1 0.0.0.0 area 0
```

```
 network 10.1.12.1 0.0.0.0 area 0
```

R6 的配置如下：

```
interface Loopback0
 ip address 6.6.6.6 255.255.255.255
!
interface Ethernet0/0
 ip address 10.1.56.6 255.255.255.0
!
router ospf 1
 router-id 6.6.6.6
 network 6.6.6.6 0.0.0.0 area 0
 network 10.1.56.6 0.0.0.0 area 0
```

R2-PE1 的配置如下：

```
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls traffic-eng tunnels
mpls label range 200 299
mpls ldp router-id Loopback0
!
ip vrf VPN-A
 rd 2345:1
 route-target export 2345:2
 route-target import 2345:5
!
interface Ethernet0/0
 ip vrf forwarding VPN-A
 ip address 10.1.12.2 255.255.255.0
!
interface Ethernet0/1
 ip address 10.1.23.2 255.255.255.0
```

```
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
router ospf 1 vrf VPN-A
 redistribute bgp 2345 subnets
 network 10.1.12.2 0.0.0.0 area 0
!
router ospf 100
router-id 2.2.2.2
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
router bgp 2345
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 neighbor 5.5.5.5 remote-as 2345
 neighbor 5.5.5.5 update-source Loopback0
 !
address-family vpnv4
  neighbor 5.5.5.5 activate
  neighbor 5.5.5.5 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  no synchronization
  redistribute ospf 1 vrf VPN-A match internal external 1 external 2
 exit-address-family
!
interface Tunnel0
```

```
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 2000
 tunnel mpls traffic-eng path-option 10 dynamic
```

R3 的配置如下：

```
mpls traffic-eng tunnels
mpls label range 300 399
mpls ldp router-id Loopback0
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
!
interface Ethernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
interface Ethernet0/1
 ip address 10.1.34.3 255.255.255.0
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
router ospf 100
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 3.3.3.3
```

```
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
```

R4 的配置如下：

```
mpls traffic-eng tunnels
mpls label range 400 499
mpls ldp router-id Loopback0
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.255
!
interface Ethernet0/0
 ip address 10.1.34.4 255.255.255.0
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
interface Ethernet0/1
 ip address 10.1.45.4 255.255.255.0
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
router ospf 100
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 4.4.4.4
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
 network 10.1.45.4 0.0.0.0 area 0
```

R5-PE2 的配置如下：

```
interface Loopback0
 ip address 5.5.5.5 255.255.255.255
!
mpls traffic-eng tunnels
mpls label range 500 599
mpls ldp router-id Loopback0
!
ip vrf VPN-A
 rd 2345:6
 route-target export 2345:5
 route-target import 2345:2
!
interface Ethernet0/0
 ip address 10.1.45.5 255.255.255.0
 mpls traffic-eng tunnels
 mpls ip
 ip rsvp bandwidth
!
interface Ethernet0/1
 ip vrf forwarding VPN-A
 ip address 10.1.56.5 255.255.255.0
!
!
router ospf 1 vrf VPN-A
 redistribute bgp 2345 subnets
 network 10.1.56.5 0.0.0.0 area 0
!
router ospf 100
router-id 5.5.5.5
 network 5.5.5.5 0.0.0.0 area 0
 network 10.1.45.5 0.0.0.0 area 0
```

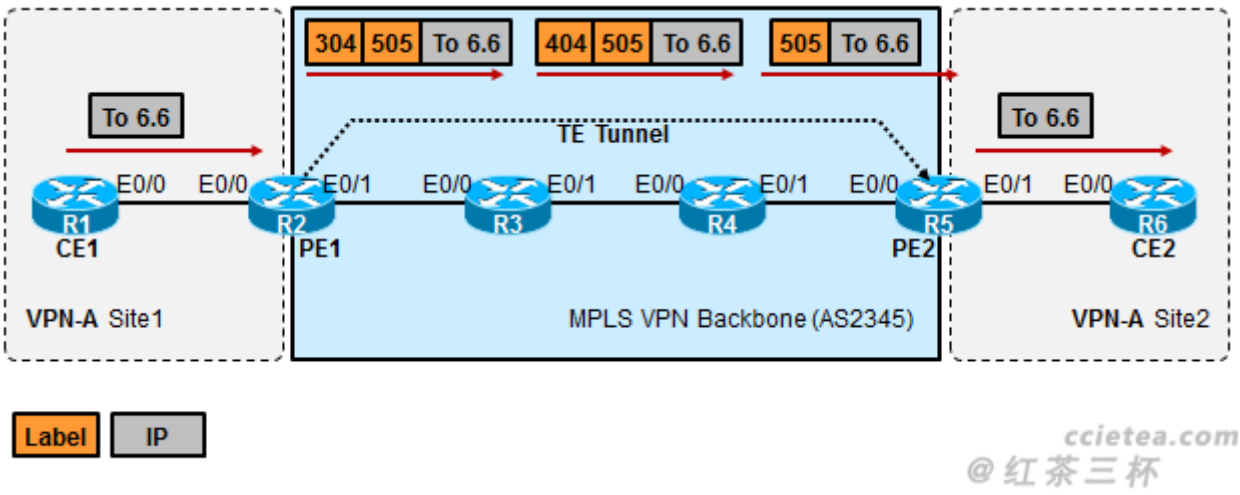
```
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
router bgp 2345
 bgp router-id 5.5.5.5
 no bgp default ipv4-unicast
 neighbor 2.2.2.2 remote-as 2345
 neighbor 2.2.2.2 update-source Loopback0
 !
address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  no synchronization
  redistribute ospf 1 vrf VPN-A match internal external 1 external 2
 exit-address-family
```

3. 现象解析

我们在 R1 上去 traceroute 6.6.6.6

R1#traceroute 6.6.6.6

```
Type escape sequence to abort.
Tracing the route to 6.6.6.6
  1 10.1.12.2 4 msec 4 msec 0 msec
  2 10.1.23.3 [MPLS: Labels 304/505 Exp 0] 4 msec 4 msec 4 msec
  3 10.1.34.4 [MPLS: Labels 404/505 Exp 0] 0 msec 0 msec 0 msec
  4 10.1.56.5 [MPLS: Label 505 Exp 0] 8 msec 0 msec 0 msec
  5 10.1.56.6 4 msec *    4 msec
```



首先 R2 的路由表：

R2-PE1#show ip route

```
O        5.5.5.5 [110/31] via 5.5.5.5, 00:06:39, Tunnel0
```

我们看到 Tunnel 已经起来了，而且 R2 上去往 5.5.5.5 的路由，出接口为 tunnel，上了 TE tunnel 了。

R2-PE1#show ip bgp vpnv4 all labels

```
   Network          Next Hop      In label/Out label
...... ......
Route Distinguisher: 2345:6
   6.6.6.6/32       5.5.5.5       nolabel/505
   10.1.56.0/24     5.5.5.5       nolabel/506
```

我们看到 R5-PE2 为 VPN 客户路由 6.6.6.6 分配了 505 的标签。

所以 R2 收到去往 6.6.6.6 的数据包，先压入 505 的内层 VPN 标签。

接着要需要顶层的标签，这就看下一跳，下一跳是 5.5.5.5，那么肯定要找 5.5.5.5 的标签。由于从路由表我们已经看到，此刻 R2 去往 5.5.5.5 实际上是进了 TE tunnel，那么：

R2-PE1#show mpls forwarding-table detail

```
Local      Outgoing    Prefix          Bytes Label   Outgoing    Next Hop
Label      Label       or Tunnel Id    Switched      interface
...... ......
204        Pop Label   5.5.5.5/32      0             Tu0         point2point
           MAC/Encaps=14/18, MRU=1500, Label Stack{304}, via Et0/1
           0E00003017000E00003016108847 00130000
```

```
        No output feature configured
```

所以这里 R2 将标签包送入 tunnel 时，给压入了外层的标签为 304，而这个 304 很明显是 R3 分配的，而且是 RSVP 为这条 TE Tunnel 分配的标签。

上面的标签，还有另一个方法可以看到，就是直接在 R2 上看 TE Tunnel：

R2-PE1#show mpls traffic-eng tunnels

```
Name: R2-PE1_t0                                (Tunnel0) Destination: 5.5.5.5
  Status:
    Admin: up          Oper: up     Path: valid       Signalling: connected
    path option 10, type dynamic (Basis for Setup, path weight 30)
...... ......
  InLabel  :  -
  OutLabel : Ethernet0/1, 304
  RSVP Signalling Info:
       Src 2.2.2.2, Dst 5.5.5.5, Tun_Id 0, Tun_Instance 5
    RSVP Path Info:
      My Address: 10.1.23.2
      Explicit Route: 10.1.23.3 10.1.34.3 10.1.34.4 10.1.45.4
```

接下去看 R3：

R3#show mpls forwarding-table

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
300	Pop Label	2.2.2.2/32	13223	Et0/0	10.1.23.2
301	Pop Label	4.4.4.4/32	0	Et0/1	10.1.34.4
302	Pop Label	10.1.45.0/24	0	Et0/1	10.1.34.4
303	403	5.5.5.5/32	6621	Et0/1	10.1.34.4
304	**404**	**2.2.2.2 0 [5]**	**5135**	**Et0/1**	**10.1.34.4**

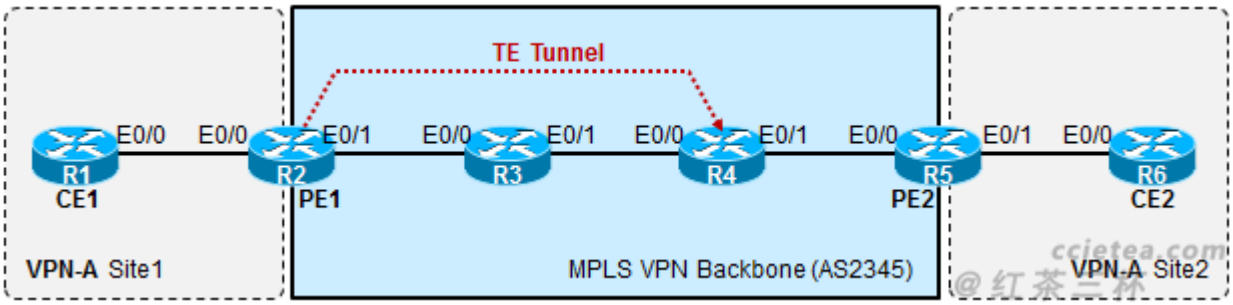
R3 将 304 替换成 404。注意，这里 304 和 404 都是 RSVP 分配的标签，因此我们在查看标签的时候还是要关注 tunnel。 上面的表项 ,2.2.2.2 0[5] ,注意不要搞错 ,不是 2.2.2.2 这个前缀对应的标签 ,而是为这条 tunnel 对应的标签。

再接下去就不用解释了吧？到了 R4，R4 将顶层标签弹出，然后将包转发给 R5.

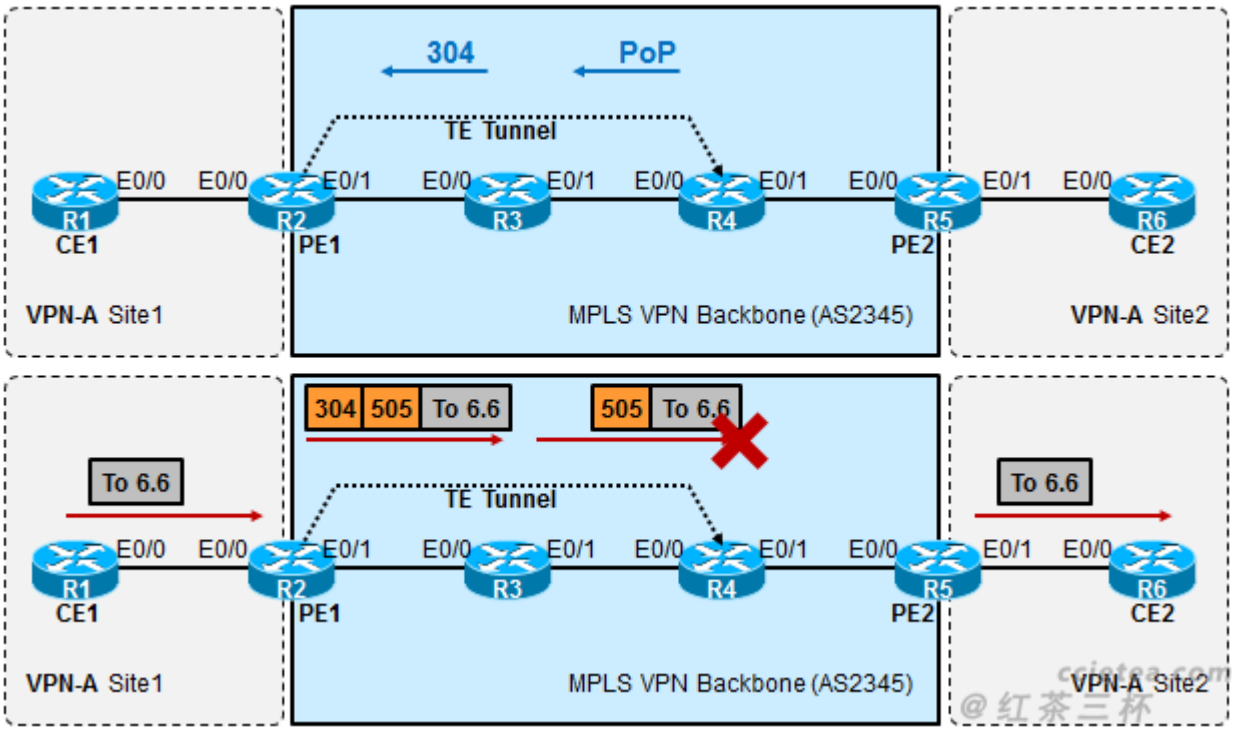
因此，在 MPLS VPN 环境中，如果我们在 MPLS VPN Backbone 内在 R1 到 R5 建立一条 TE Tunnel，并不

会影响 VPN 的流量。

8.2 实验 2



这一次我们做的测试，环境就跟实验 1 不同了，TE Tunnel 的尾端现在是到了 R4 的 4.4.4.4。



在这个环境中，由于 Tunnel 尾端在 R4，那么 R4 通过 RSVP 为这条 tunnel 分配的标签就是 POP。

这样一来当 R2 转发标签数据时，数据压入两层标签，外层是 TE Tunnel 的标签 304，内层是 VPNv4 标签 505，这个标签包到了 R3 后，R3 将顶层的标签弹出，然后将数据转发给 R4，但是到了 R4 它就傻逼了，因为 505 标签是由 R5 产生的，R4 压根不认识这 505 标签，于是就丢包了。

解决方案：在 R2 及 R4 之间建立 targeted LDP 邻居关系

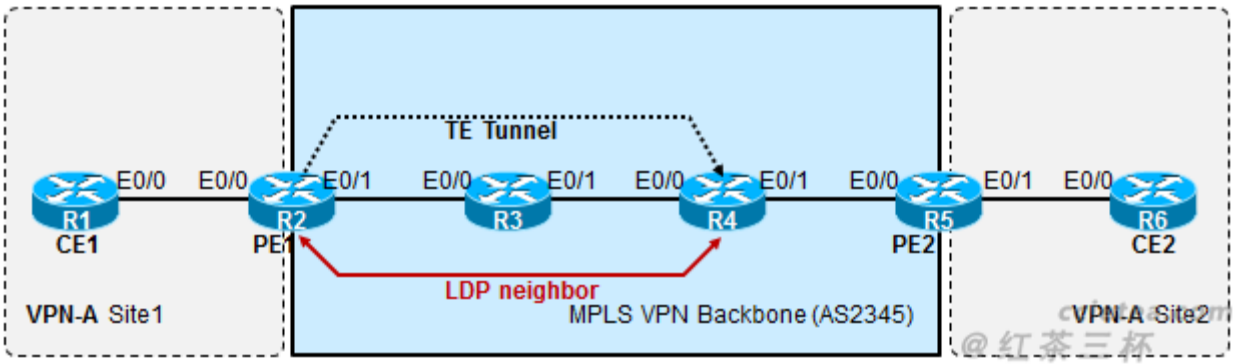
R2 的 tunnel 配置修改如下：

```
interface Tunnel0
 ip unnumbered Loopback0
 mpls ip
 tunnel mode mpls traffic-eng
 tunnel destination 4.4.4.4
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 2000
 tunnel mpls traffic-eng path-option 10 dynamic
```

上述配置完成后，R2 就会通过已经建立好的、到达 R4 的 tunnel 去发 LDP。注意 Tunnel 一定要激活 MPLS IP

R4 的配置修改如下：

```
mpls ldp neighbor 2.2.2.2 targeted ldp
```



在 R2 上看一下 LDP 邻居：

R2-PE1#show mpls ldp neighbor

```
    Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 2.2.2.2:0
        TCP connection: 3.3.3.3.62244 - 2.2.2.2.646
        State: Oper; Msgs sent/rcvd: 112/112; Downstream
        Up time: 01:29:44
        LDP discovery sources:
          Ethernet0/1, Src IP addr: 10.1.23.3
        Addresses bound to peer LDP Ident:
```

```
        10.1.23.3          10.1.34.3          3.3.3.3
    Peer LDP Ident: 4.4.4.4:0; Local LDP Ident 2.2.2.2:0
        TCP connection: 4.4.4.4.24940 - 2.2.2.2.646
        State: Oper; Msgs sent/rcvd: 19/19; Downstream
        Up time: 00:07:56
        LDP discovery sources:
          Targeted Hello 2.2.2.2 -> 4.4.4.4, active
        Addresses bound to peer LDP Ident:
          10.1.34.4          10.1.45.4          4.4.4.4
```

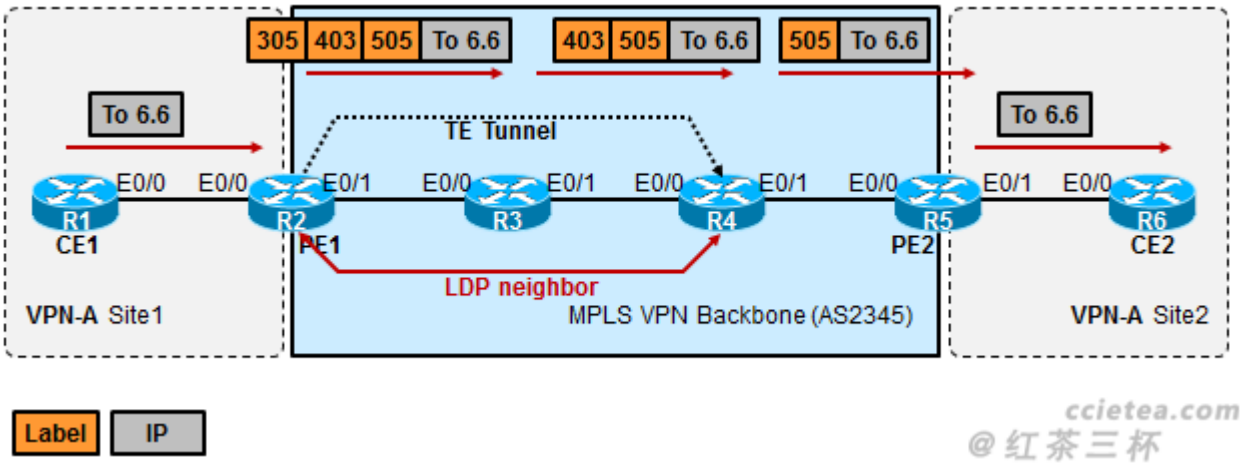
已经和 R4 建立起来了 targeted LDP 邻居关系。

接着我们再去 R1 上测试一下：

R1#traceroute 6.6.6.6

```
Type escape sequence to abort.
Tracing the route to 6.6.6.6
  1 10.1.12.2 12 msec 4 msec 4 msec
  2 10.1.23.3 [MPLS: Labels 305/403/505 Exp 0] 8 msec 4 msec 4 msec
  3 10.1.34.4 [MPLS: Labels 403/505 Exp 0] 0 msec 4 msec 0 msec
  4 10.1.56.5 [MPLS: Label 505 Exp 0] 4 msec 0 msec 0 msec
  5 10.1.56.6 8 msec *    0 msec
```

我们看到数据的转发层面是这样的：



那么来详细分析一下：

首先是 R2 上

由于去往 6.6.6.6 的 IP 数据包到了 R2，IP 数据包内层下压入一个 VPN 的标签 505，这个没问题。

接着我们继续，R2 上关于 6.6.6.6 的 VPNv4 路由是由 5.5.5.5 也就是 R5 传递过来的。

而 R2 的路由表此刻是这样的：

R2-PE1#show ip route

```
O        5.5.5.5 [110/31] via 4.4.4.4, 00:23:47, Tunnel0
```

走的是 tunnel。

因此：

R2-PE1#show mpls forwarding-table detail

```
Local      Outgoing    Prefix           Bytes Label   Outgoing    Next Hop
Label      Label       or Tunnel Id     Switched      interface
204        403         5.5.5.5/32       0             Tu0         point2point
        MAC/Encaps=14/22, MRU=1496, Label Stack{305 403}, via Et0/1
        0E00003017000E00003016108847 0013100000193000
        No output feature configured
```

我们看到，这里是有两层标签。所以 R2 将原始的 IP 包先压入一层 VPN 标签 505，在压入 305 403 这两层标签。

这里指的注意的是 ,这里的是标签 403 是由 R4 分配为前缀 5.5.5.5/32 所分配的 LDP 标签 ,这是通过 targeted LDP 连接传递给 R2 的。而 305 这是 R3 通过 RSVP 为 TE Tunnel 所分配的标签。

下面可以验证一下：

R2-PE1# show mpls forwarding-table

```
Local      Outgoing    Prefix           Bytes Label   Outgoing    Next Hop
Label      Label       or Tunnel Id     Switched      interface
204    [T] 403         5.5.5.5/32       0             Tu0         point2point
```

我们在 R2 的 LFIB 表里看到了 R4 分配给前缀 5.5.5.5/32 的 LDP 标签 403

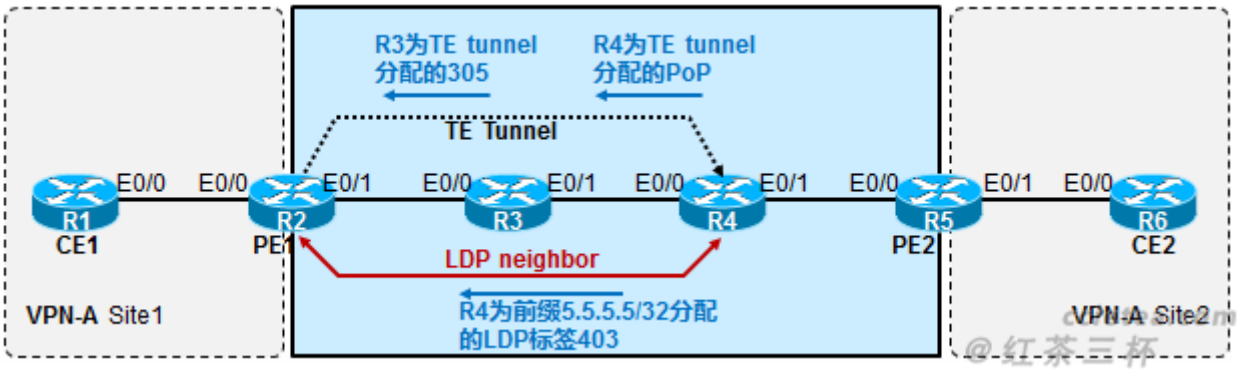
最终这个压入了 403 和 505 的标签包要被送入 TE Tunnel，因此：

R2-PE1#show mpls traffic-eng tunnels

```
Name: R2-PE1_t0                           (Tunnel0) Destination: 4.4.4.4
  Status:
    Admin: up          Oper: up     Path: valid       Signalling: connected
```

```
    path option 10, type dynamic (Basis for Setup, path weight 20)

Config Parameters:
    Bandwidth: 2000        kbps (Global)    Priority: 7    7     Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:    enabled     LockDown: disabled    Loadshare: 2000        bw-based
    auto-bw: disabled
Active Path Option Parameters:
    State: dynamic path option 10 is active
    BandwidthOverride: disabled    LockDown: disabled    Verbatim: disabled

InLabel    :    -
OutLabel : Ethernet0/1, 305
RSVP Signalling Info:
         Src 2.2.2.2, Dst 4.4.4.4, Tun_Id 0, Tun_Instance 6
    RSVP Path Info:
        My Address: 10.1.23.2
        Explicit Route: 10.1.23.3 10.1.34.3 10.1.34.4 4.4.4.4
        Record      Route:      NONE
        Tspec: ave rate=2000 kbits, burst=1000 bytes, peak rate=2000 kbits
......
```

标签的分发是这样的：



9 参考书籍

《MPLS 技术架构》

MPLS 技术笔记

红茶三杯 CCIE 学习文档

文档版本：	1.5	更新时间：	2013-04-02
文档作者：	红茶三杯	文档地址：	http://ccietea.com
文档备注：	请关注文档版本及更新时间。		

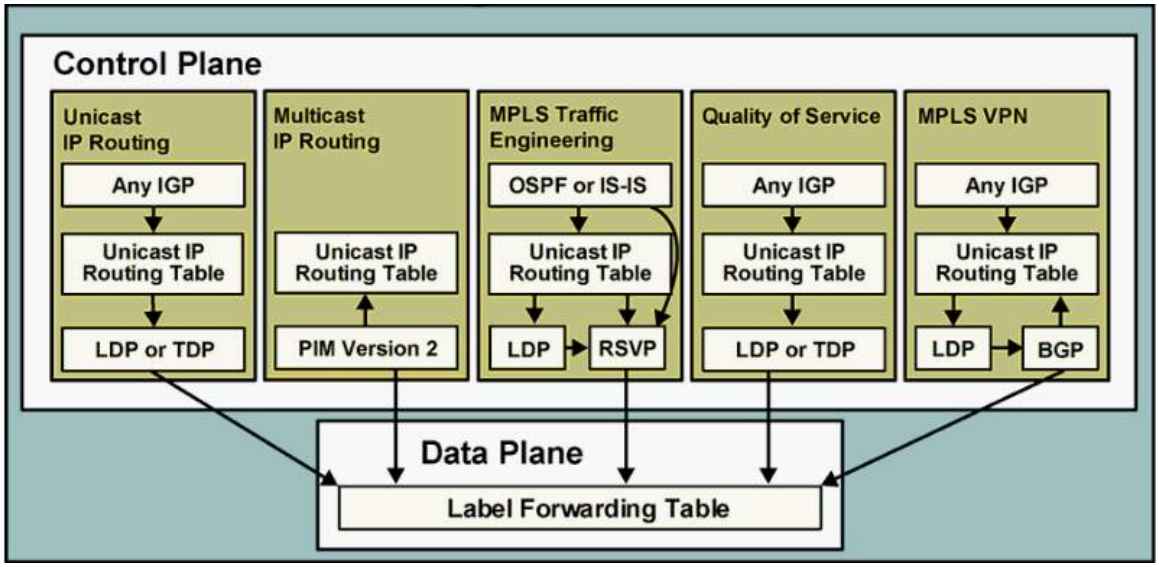
红茶三杯
学习 沉淀 成长 分享
微博：http://weibo.com/vinsoney
博客：http://blog.sina.com.cn/vinsoney
站点：http://ccietea.com

1 Basic

1.1 基础知识点

1. **MPLS 的特性**
   - MPLS 依赖 IP 路由和 CEF 交换
   - MPLS 是一种依据数据包标签的转发技术
   - MPLS 支持多种三层协议
   - MPLS 涉及的目标网段与传统的 IP 转发是一样的

2. **MPLS 的应用**



3. **当一个数据包进入 PE 后，是查路由表（FIB）还是查标签表（LFIB）？**

   具体查哪个表，主要取决于所收到的数据的二层封装，例如如果二层封装是以太网（数据帧），则看帧头的“类型/长度”字段的值：

   Type：0x8847（单播）上层承载的是 MPLS，查找 LFIB

   Type：0x8848（组播）上层承载的是 MPLS，查找 LFIB

   Type：0x0800 承载的是 IPv4 报文，查找 FIB

4. **在初始路由器上，是查找 FIB 还是 LFIB 呢？**

   由于在初始路由器上，产生的是 IP 报文，查找的是 CEF 表，用“show ip cef <ip address> detail”命令，可以从输出中看到对于此路由是否压入标签，分配标签是根据 CEF 表中的前缀来分配的。

   R1#show ip cef 4.4.4.4

```
4.4.4.4/32, version 12, epoch 0, cached adjacency 10.1.12.2
0 packets, 0 bytes
  tag information set
    local tag: 104
    fast tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
  via 10.1.12.2, FastEthernet0/0, 0 dependencies
    next hop 10.1.12.2, FastEthernet0/0
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
```

例如这条 CEF 显示，去往 4.4.4.4 的数据，要压上 203 的标签，然后发送给 10.1.12.2。

1.2 基本名词

1. **A label switch router (LSR)**

   一台支持并激活了 MPLS 的设备。It is capable of understanding MPLS labels and of receiving and transmitting a labeled packet on a data link. Three kinds of LSRs exist in an MPLS network:

   - **Ingress LSRs**—Ingress LSRs receive a packet that is not labeled yet, insert a label (stack) in front of the packet, and send it on a data link.
   - **Egress LSRs**—Egress LSRs receive labeled packets, remove the label(s), and send them on a data link. Ingress and egress LSRs are edge LSRs.
   - **Intermediate LSRs**—Intermediate LSRs receive an incoming labeled packet, perform an operation on it, switch the packet, and send the packet on the correct data link.

2. **A label switched path (LSP)**

   is a sequence of LSRs that switch a labeled packet through an MPLS network or part of an MPLS network. Basically, the LSP is the path through the MPLS network or a part of it that packets take. //单向路径。

   实际上，LSP 是报文在穿越 MPLS 网络或者部分 MPLS 网络时的路径。

3. **A Forwarding Equivalence Class (FEC)**

   is a group or flow of packets that are forwarded along the same path and are treated the same with regard to the forwarding treatment.

   在转发过程中，具有相同处理方式的一组数据，可通过地址、隧道、COS 等方式来标识，通常在一台设备上，对于一个 FEC 分配相同的标签。属于一个 FEC 的流量具有相同的转发方式、转发路径和转发待遇。

但是并不是所有拥有相同标签的报文都属于一个 FEC，因为这些报文的 EXP 值可能不相同，执行方式可能不同，因此他们可能属于不同的 FEC。

决定报文属于哪一个 FEC 的路由器是入站 LSR，因为是它对报文进行分类和压入标签。

下面是一些 FEC 的范例：

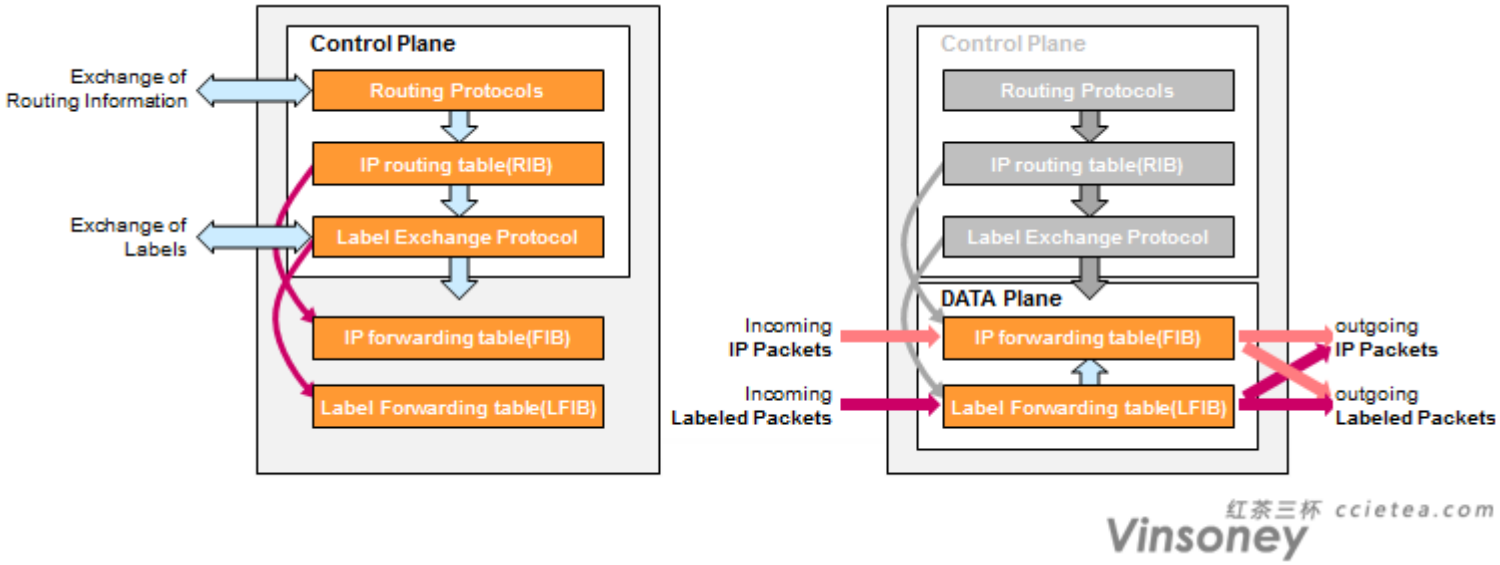
- 第三层目的 IP 匹配同一个特定前缀的报文
- 属于某个特定组播组的组播报文
- 根据进程或者 IP　DSCP 字段有相同处理方式的报文

注：一条 FEC 可以包含多个流，但不是一个流一个 FEC，比如一台主机在看新浪的网页，这是一个流，又在看新浪的视频，这又是一个流，这两个流在新浪发给远程主机时，走的路径应该是相同的，所以一个 FEC 有多个流，但是每个流并没有属于单独的 FEC

1.3 基本架构

Control plane： 交换三层**路由信息**（如 OSPF、ISIS、BGP 等）及**标签**（如 TDP、LDP、BGP 及 RSVP 等）；

Data plane： 基于标签进行数据转发

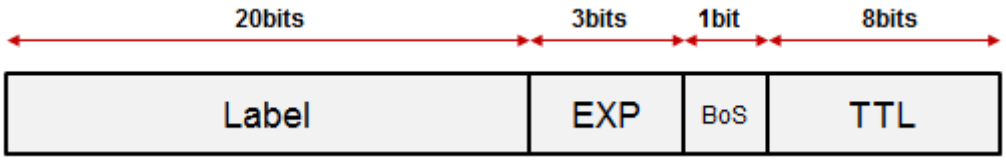


注意：LDP 或 TDP 只会对 IGP 协议的路由条目产生标签，不会对 BGP 的路由条目产生标签

1.4 MPLS 标签

1. 标签描述

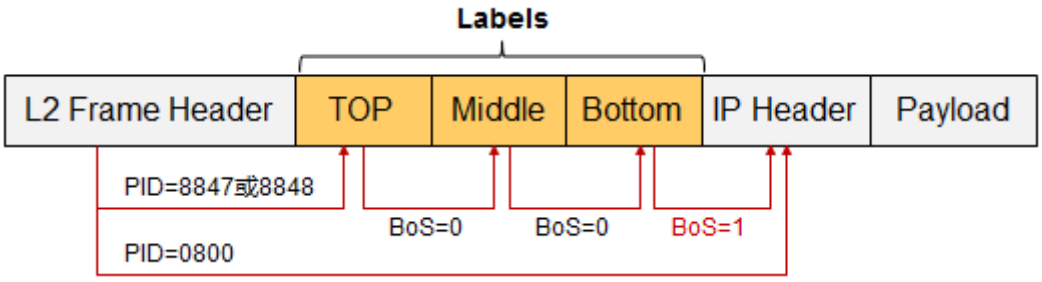
两种模式：frame mode ；ATM mode（cell mode），我们主要研究的是 frame mode



标签头一共 32 位，包含的字段如下：

LABEL	20bits
BoS	1bit 为栈底位（这一位 置 1，则为最后一个标签。 可以给一个数据包压多层标签，最后一个标签的 Bos 位则置 1，当处理到这层，上层数据包则为普通数据包。）
TTL	8bit 最大 255，通常在加标签的时候，是把普通 ip 报文的 TTL 直接 copy 进来
EXP	3bit 实验位用于 QoS

2. 标签的位置



路由器如何知道一个报文是普通报文还是一个标签包呢？ MPLS 报文会在二层头和三层包间插入一个 MPLS 标签头。同时二层数据链路层帧头会做相应的指示，例如以太网数据帧，MAC 层的 TYPE 字段会指示上层数据是否为 mpls 标签帧（如果是 IPv4 报文的话值为 0x0800，如果是标签包则为 8847-单播或 8848-组播）

上图是封装了三层标签包，对于一个 LSR 来说，只处理第一个标签。每一层标签都有个 BoS 栈底位，用来表示是否已经是标签栈的栈底，最后一个标签的 BoS=1。

3. MPLS 的编码

标签栈放置在第三层包头之前，也就是说，被传输的协议头部之前，同时在第二层包头之后。

对于第 2 层链路层封装可以是 CISCO IOS 所支持的所有封装类型，PPP、HDLC、Ethernet 等等。那么在这些第 2 层的帧头，就需要有相应的协议字段来指示上层是一个标签头。

第2层协议类型	第2层协议标识名称	值（16进制）
PPP	PPP协议字段	0281
以太网/802.3 LLC/SNAP封装	TYPE	8847（单播） 8848（组播）
HDLC	协议	8847
帧中继	NLPID（网络级别协议ID）	80

4. **标签的处理方式**
   - Insert（impose or push）
   - Swap
   - Remove（PoP）

   关于标签处理方式的更详细内容，见下文。

2 转发带标签的报文

2.1 转发带标签的报文

The first label is imposed on the ingress LSR and the label belongs to one LSP. The path of the packet through the MPLS network is bound to that one LSP. All that changes is that the top label in the label stack is swapped at each hop. The ingress LSR imposes one or more labels on the packet. The intermediate LSRs swap the top label (the incoming label) of the received labeled packet with another label (the outgoing label) and transmit the packet on the outgoing link. The egress LSR of the LSP strips off the labels of this LSP and forwards the packet.

For this to work, adjacent LSRs must agree on which label to use for each IGP prefix. Therefore, each intermediate LSR must be able to figure out with which outgoing label the incoming label should be swapped

MPLS 与 Tag switching

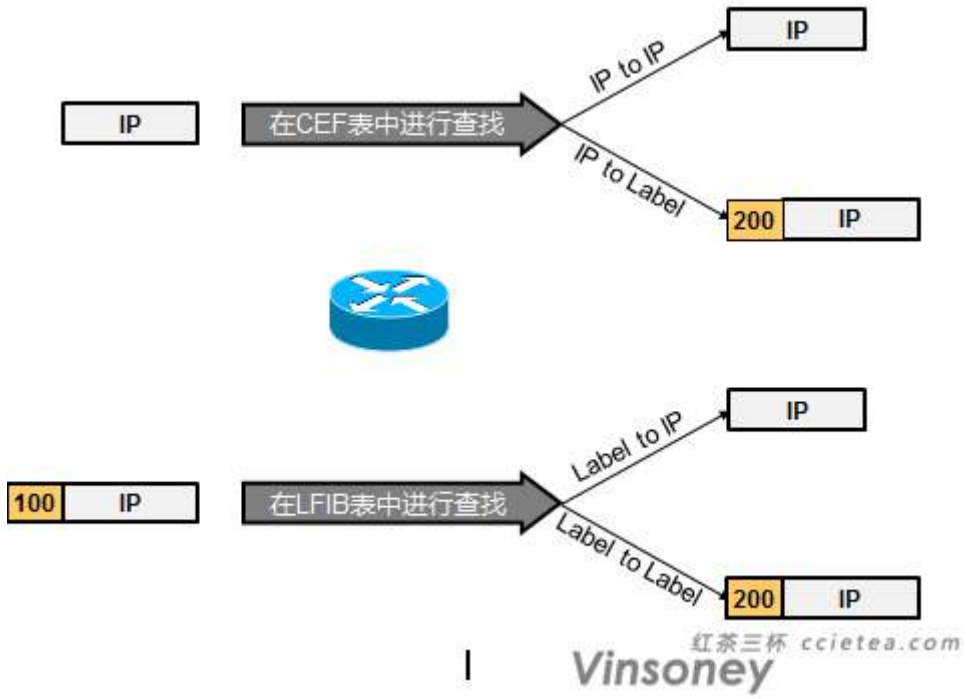
MPLS 是 CISCO 私有协议，后来被 IETF 借用，在 IETF 叫做 tag switching，可以理解为名字不同而已

两者在启用时，用的协议不一样

LDP　　IETF 标准　　（TCP/UDP 端口号 646）

TDP　　CISCO 私有　（TCP/UDP 端口号 711）

2.2 IP 查找和标签查找



如果 LSR 收到一个 IP 报文，则查看 FIB 表，假设 FIB（CEF 表）相关表项输出如下：

```
R1#show ip cef 4.4.4.0 detail
4.4.4.0/24, epoch 0
  local label info: global/16
  nexthop 10.1.12.2 FastEthernet0/0 label 204
```

那么 LSR 将这个 IP 包压入一个标签头，打上标签 204，再从 F0/0 接口转发出去。在 CISCO IOS 中，CEF 交换是唯一的一种可用于标记报文的 IP 转发模式，因此在启用 MPLS 的时候必须在路由器上开启 CEF。

而当一台路由器收到一个带标签的报文的时候，则在 LFIB 表中进行查找，LFIB 表中相关的匹配条目会有针对该入站标签的出站动作或标签，以及下一跳信息。

【注意】如果路由器收到一个带标签的数据包，并且顶部标签不能在本地 LFIB 中找到，那么 CISCO IOS 将丢弃

这个数据包。

2.3 三张表

1. **FIB**

   也就是 CEF 表

2. **LIB 标签信息库**

   路由器为每一个 IGP 前缀在本地生成一个标签并分发给 LDP 邻居，同时也从其他 LDP 邻居收到为特定前缀分发的标签。路由器将本地标签和远程标签（LDP 邻居发给我的）存储在 LIB 中。

3. **LFIB 标签转发信息库**

   LSR 路由可能收到某个特定前缀的多个标签（多个 LDP 邻居分发的），但他只需要使用其中一个，IP 路由表用来确定这个 Ipv4 前缀的下一跳（LFIB 的构造需要一来 IGP 路由表，因为 LFIB 的下一跳就是 IGP 表算来的）。LSR 用这样的信息来创建它的 LFIB。在 LFIB 中本地捆绑的标签作为入站标签，LDP 邻居通告的标签作为出站标签。

   构成 LFIB 的标签可以不是 LDP 分发的 ,在 MPLS 流量工程中使用 RSVP 分配标签 ,在 MPLS VPN 中，VPN 标签由 BGP 分发。在任何情况下，LFIB 总是用来转发入站的带标签的报文。

- **查看 MPLS 转发表（LFIB）**

R1#show mpls forwarding-table

R1#show mpls forwarding-table

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
16	204	4.4.4.0/24	0	Fa0/0	10.1.12.2
17	205	3.3.3.0/24	0	Fa0/0	10.1.12.2
100	Pop Label	2.2.2.0/24	0	Fa0/0	10.1.12.2
103	202	10.1.34.0/24	0	Fa0/0	10.1.12.2
104	Pop Label	10.1.23.0/24	0	Fa0/0	10.1.12.2

R1#show mpls forwarding-table 4.4.4.0 detail

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
16	204	4.4.4.0/24	0	Fa0/0	10.1.12.2
	MAC/Encaps=14/18, MRU=1500, Label Stack{204}				
	CA014FEC0008CA004FEC00088847 000CC000				

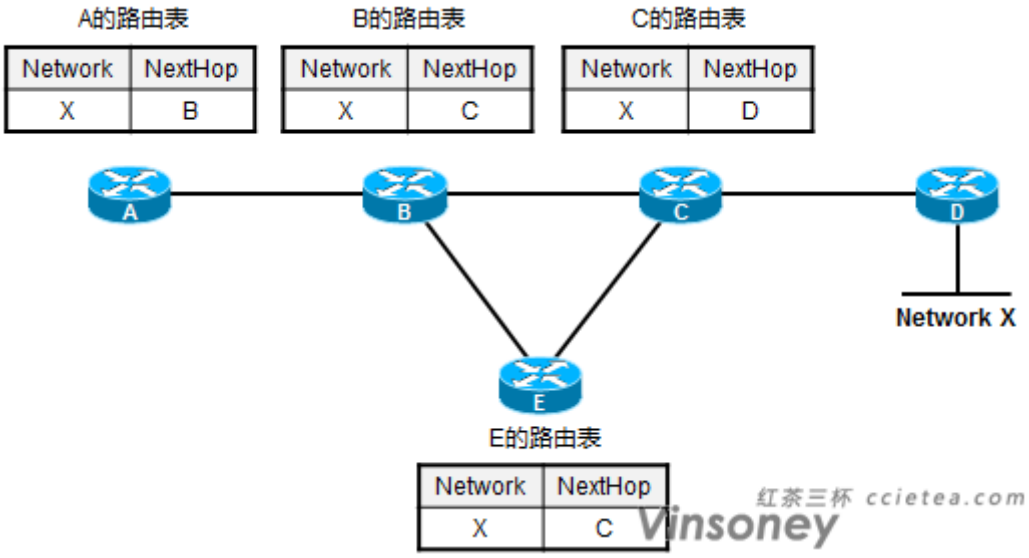
```
No output feature configured
```

使用 detail 关键字能查看到 LFIB 转发条目的详细信息，包括二层的信息，以及所有的标签内容，如果不加则只能看到顶层标签。

2.4 标签分配过程概述

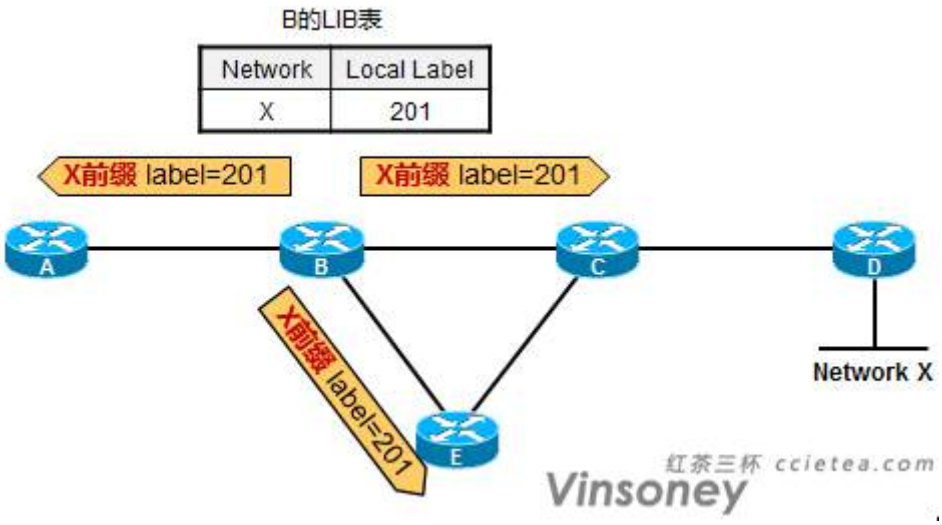
注意，我们研究的是 frame-mode 的标签分配。

1. 构建 IP 路由表

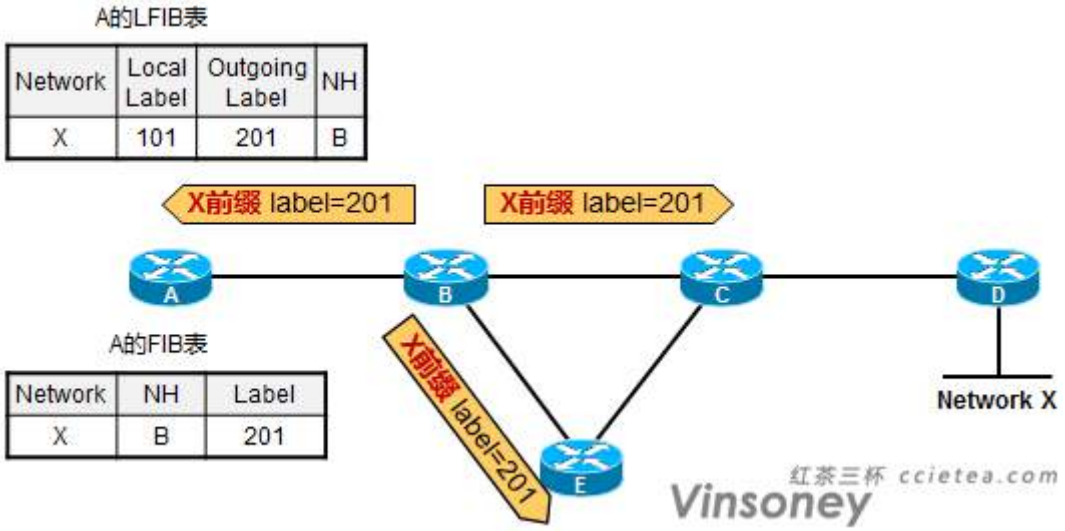


首先呢，所有的路由器都运行路由协议，全网路由互通。接下去在路由器的接口上激活 LDP，那么路由器两两之间就会形成 LDP 邻接关系。

2. 分配并分发标签



激活 LDP 后，每台路由器会为自己本地路由表中的路由前缀分配并捆绑标签，例如本图中的 X 路由，所有路由器都会自己为 X 路由分配一个标签 ,然后将为 X 路由捆绑的标签保存在本地的 LIB 表里。随后 ,所有的 LDP 路由器，将自己为各个路由前缀捆绑的标签发给它的 LDP 邻居，如图所示，B 将捆绑的标签发给了 A、E 和 C。这里有个小细节，注意，标签的分发没有水平分割的概念，也就是说虽然 B 可能是从 C 学习到路由 X，但是 B 仍然会将为前缀 X 捆绑的标签传递给 C。C 也会将传递自 B 的标签放在 LIB 中，不过不用担心会有环路，因为 LDP 可以借助 IGP 路由协议来防止环路。



当 LDP 路由器从他的 LDP 邻居收到邻居的标签捆绑，它会将捆绑的结果存储在自己的 LIB 表中，LIB 表中有 local label ,这是自己为特定路由前缀分发的标签 ,remote label 这是邻居为该路由前缀分发的标签。LDP 路由器会将邻居分配的 remote label 都放在 LIB，但是只会从 LIB 中所有可能的标签中选择出一个可能的出站标签放入 LFIB 中。LDP 路由器根据 LIB 表，结合路由表，构成 LFIB 表，如上图中 A 的 LFIB 表。

这里有一点要注意，我们看 E 路由器，实际上，它是会收到自己 LDP 邻居 B 和 C 的标签捆绑，其中都有关于路由 X 捆绑的标签，那么 E 在自己的 LFIB 表里，存的是哪位邻居的 remote label 呢？答案是 C，因为 C 是 E 路由器去往 X 的下一跳，E 会借助路由表，来判断谁的标签捆绑“更优”。

3. LIB and LFIB 表的建立

A的LFIB表

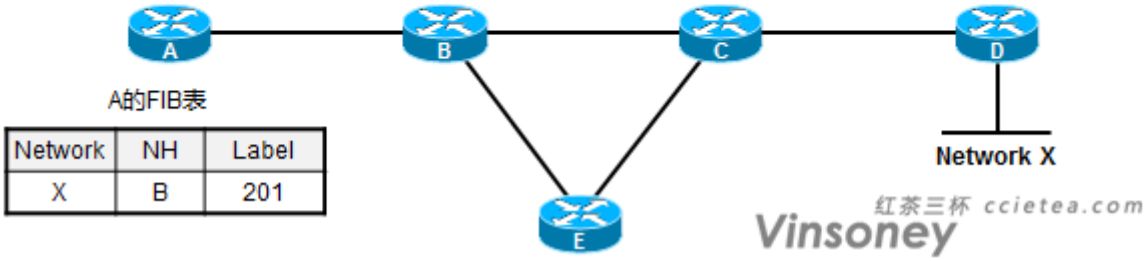
Network	Local Label	Outgoing Label	NH
X	101	201	B

B的LFIB表

Network	Local Label	Outgoing Label	NH
X	201	301	C

C的LFIB表

Network	Local Label	Outgoing Label	NH
X	301	PoP	D



在所有的 LDP 路由器都互相发送标签捆绑后，大家逐步构建自己的 LFIB。

A的LFIB表

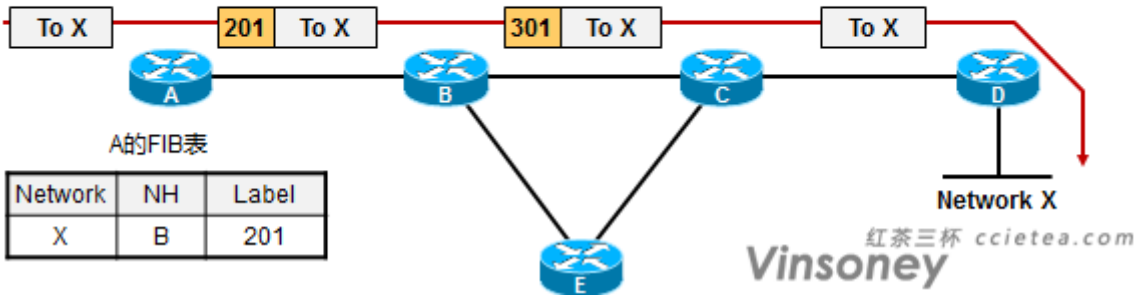
Network	Local Label	Outgoing Label	NH
X	101	201	B

B的LFIB表

Network	Local Label	Outgoing Label	NH
X	201	301	C

C的LFIB表

Network	Local Label	Outgoing Label	NH
X	301	PoP	D

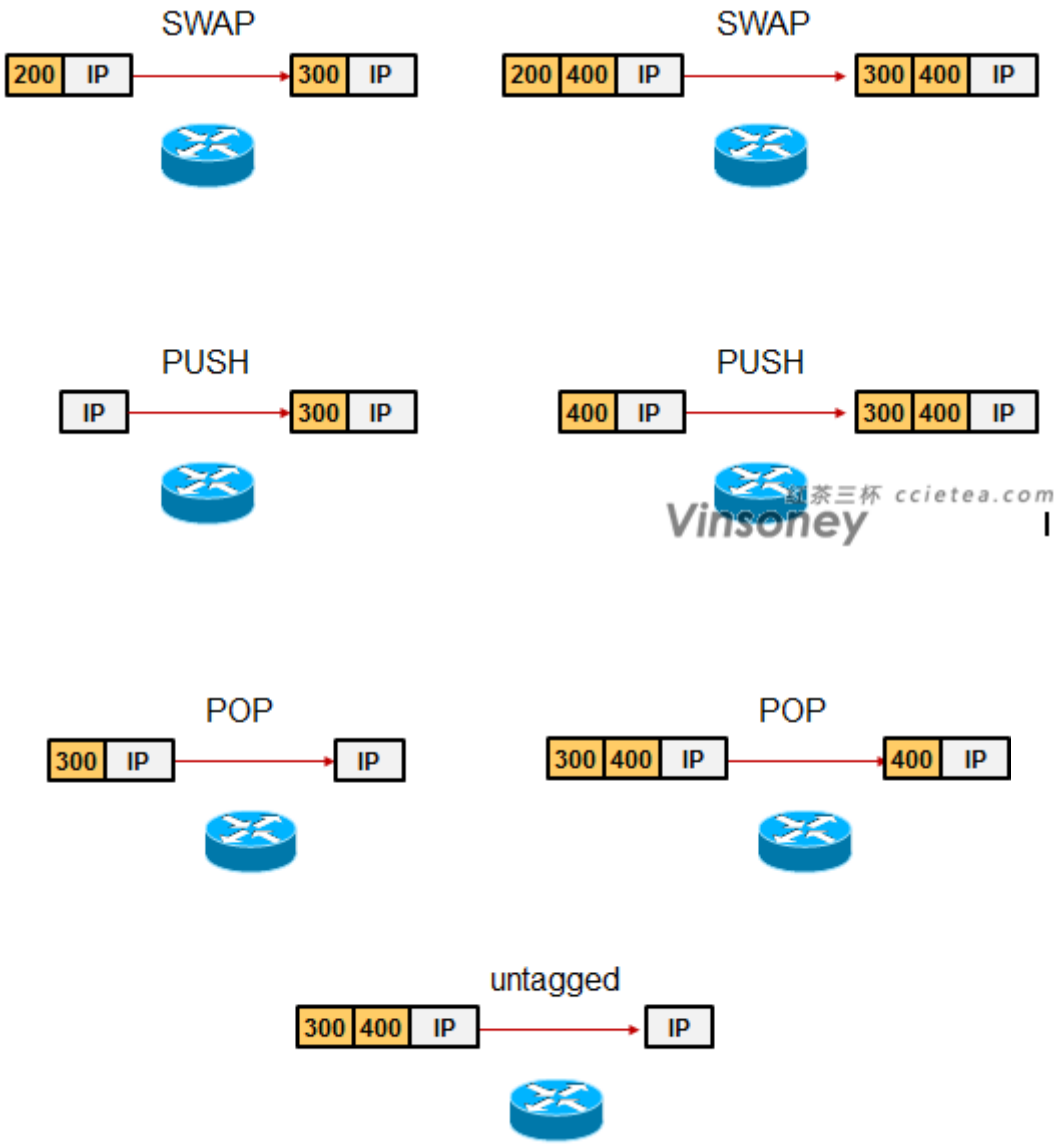


大家的 LFIB 构建完成后，我们就能看到，当 A 收到一个 IP 数据包，要访问 X，那么 A 查看自己的 CEF 表（为什么是查看 CEF 还记得么？），发现要给数据包压标签，于是它压上标签头，标签值为 201，然后下一跳是 B ,也就是发送给 B。那么 B 收到这个标签包 ,发现标签头里的 label 字段为 201 ,于是查看自己的 LFIB，发现 201 标签的 outgoing label 是 301 并且下一跳是 C，于是它将 201 标签替换成 301，然后交给 C。到了 C 它也去查 FIB 表，发现 301 标签对应的 outgoing label 是 PoP 并且下一跳是 D，于是它将标签头弹出，这就露出了原始的 IPv4 数据，C 将这个数据包交给 D。

2.5 标签处理动作的小结

- **移除：** 顶部标签被移除。报文的转发依靠标签栈中余下的标签，或者将其作为无标签的报文进行转发。
- **交换：** 顶部标签被移除，同时使用一个新的标签代替被移除的标签。
- **添加：** 顶部标签被一个新的标签所替代（交换），并且在代替的标签上层会再增加一个或者多个标签。

- **未标记/无标签：** 标签栈整个被移除，并且报文按照无标签的方式转发。
- **聚合：** 标签栈被移除，并且对该 IP 报文进行 IP 查找



1. **Outgoing tag(Label)中的 POP 与 Untag 有什么区别?**
   - POP　仅会将顶层标签头部弹出，经过该动作转发后的报文可以是 IP 报文也可以是 MPLS 标签报文
   - Untag　会将所有标签头部移除，变成一个纯 IP 报文转发出去

2. **产生 PoP 和 untag 的情况**

   PoP：　收到了下游发来的分配给某特定前缀的空标签，这个标签的值是 3，那么该 LSR 向这个下游 LSR 发送去往该前缀的数据的时候，他会弹出顶层标签（POP）进行转发，注意，这时候对于本 LSR 来说只需要查

找一次 LFIB，LFIB 中有下一跳的信息，因此弹出顶层标签，然后交给下一跳即可，而不用再次查找 FIB 表(如果弹出标签后是 IP 包)。

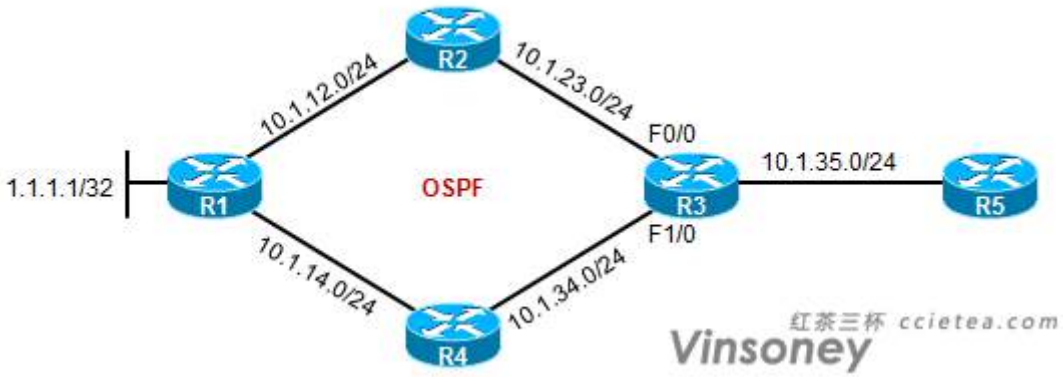
Untag： 弹出所有标签，随后根据下一跳（查找 FIB 表）转发。出现 untag 有以下 3 种原因：

- 下游不能分配标签，没有启用 MPLS
- 下游分了标签但传不出来，是因为 LDP Neighbor 没有建立
- 分配标签错误 ,这种情况仅出现在 IGP 是 OSPF 的情况下 ,因为如果用环回口作为 ldp 的 router-id，并且不是 32 位的，OSPF 会自动以 32 位的环回地址发布，这样会导致标签就分配错误。

3. 关于 untagged 、聚合等的详细介绍和区别、产生原因，请见《红茶三杯 MPLS VPN 技术文档》

2.6 标签报文的负载均衡

1. CEF 中的等价负载均衡



1) 环境描述

- 所有路由器运行 OSPF，全网互通，IP 规划如图所示，所有设备的 loopback0 为 x.x.x.x/32 编址，x 为设备编号。
- R1 通告 Loopback 路由.1.1.1.1/32
- R5 接口上配置两个地址：10.1.35.5，以及 10.1.35.55，用于模拟两个不同的源主机

2) 实验现象

在 R3 上看看路由表：

```
O        1.1.1.1 [110/3] via 10.1.34.4, 00:53:36, FastEthernet1/0
                         [110/3] via 10.1.23.2, 00:53:36, FastEthernet0/0
```

路由表中，R3 去往 1.1.1.1，使用 10.1.34.4 及 10.1.23.2 等价负载均衡。

我们在 R3 上开启了 CEF，实际上在 CEF 中，默认的负载均衡方式是基于目的地的负载均衡。CEF 的

基于目的地负载均衡实际上是通过对目的和源IP地址进行HASH后实现的。也就是说 ,实际上是基于源、目地址对进行负载均衡的。

R3#show ip cef exact-route 10.1.35.5 1.1.1.1

10.1.35.5 -> 1.1.1.1 : FastEthernet1/0 (next hop 10.1.34.4)

通过上述命令，可以查看 R3 上，源为 10.1.35.5，目的为 1.1.1.1 的报文的实际转发出接口及下一跳，我们看到，其实这个源目地址对的流量，使用的实际下一跳是 10.1.34.4，出接口是 F1/0 口，也就是说，只要是来自 10.1.35.5，去往 1.1.1.1 的流量，恒定走 Fa1/0 口出去。

R3#show ip cef exact-route 10.1.35.55 1.1.1.1

10.1.35.55 -> 1.1.1.1 : FastEthernet0/0 (next hop 10.1.23.2)

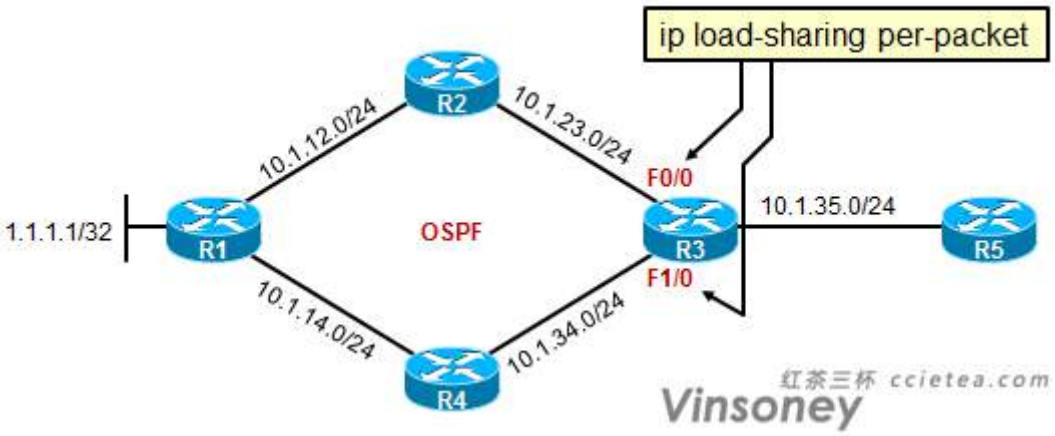
再看，现在看的是源为 10.1.35.55 目的为 1.1.1.1 的流量，我们发现 R3 使用了另一条等价路径，也即是从 F0/0 口出去。这就是基于源目地址对的含义。

在 R5 上可以使用：

traceroute 1.1.1.1 source 10.1.35.55

traceroute 1.1.1.1 source 10.1.35.5

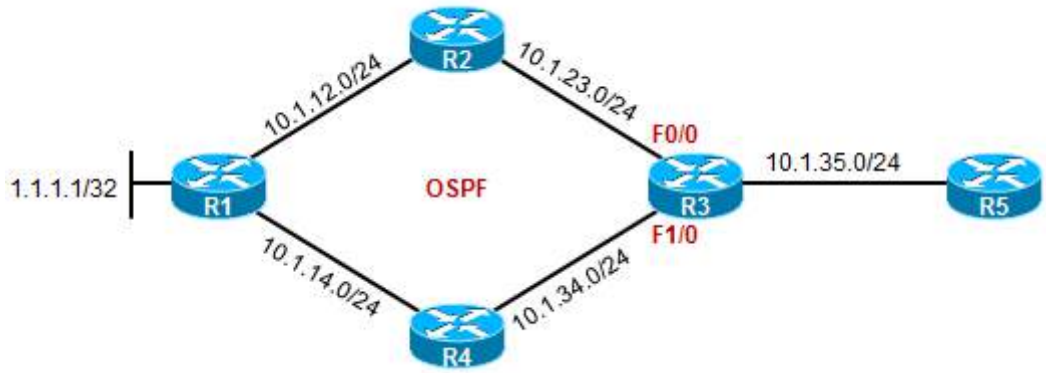
进行测试，看看数据的转发路径。那么如果希望**基于报文做负载均衡**呢？



在 R3 的 F0/0 及 Fa1/0 口上做如图配置即可。如此一来 ,即使是源目的地址不变 ,R3 也会在 F0/0 及 F1/0 接口上平均的分配流量。

2. 带标签报文的负载均衡

如果 MPLS 的有效负载是一个 IPv4 或者 IPv6 的报文的话，在基于目的的负载均衡环境中，CISCO IOS 使用 CEF 的哈希算法来选择出战接口。注意，如果有一条 IP（无标签）路径及一条带标签的路径拥有相同的度量值的话，只有带标签的路径才用于报文转发。这是因为在某些情况下，经过不带标签路径的流量无法到达目的地。例如 MPLS VPN 环境中的 P 路由器上。



现在，我们激活所有接口的 LDP。

然后在 R3 上：

R3#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
300	400	1.1.1.1/32	0	Fa1/0	10.1.34.4
	200	1.1.1.1/32	0	Fa0/0	10.1.23.2

我们看到，在 R3 的 LFIB 中，去往 1.1.1.1 这个目的地，有两条标签路径负载均衡。

那么实际上 ,这里默认的仍然是基于源目地址对的负载均衡。从测试的结果来看 ,10.1.35.55 访问 1.1.1.1 走的是 f.0/0，用的出站标签是 200，而 10.1.35.5 访问 1.1.1.1 走的是 F1/0，用的出站标签是 400。

在没有 IPv6 能力的 CISCO IOS 路由器中，对带标签的报文实施负载均衡的通用规则如下：

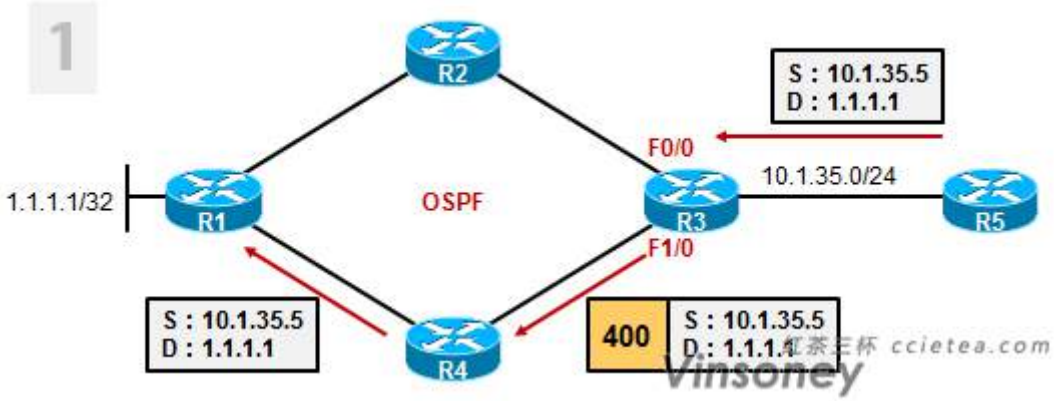
- 如果 MPLS 的有效负载是一个 IPv4 报文的话 ,负载均衡将通过对 IPv4 报文头部的源和目的 IP 地址进行哈希来实施；
- 如果 MPLS 的有效负载不是一个 IPv4 报文的话 ,负载均衡将通过查看 bottom 底部标签的值来进行。

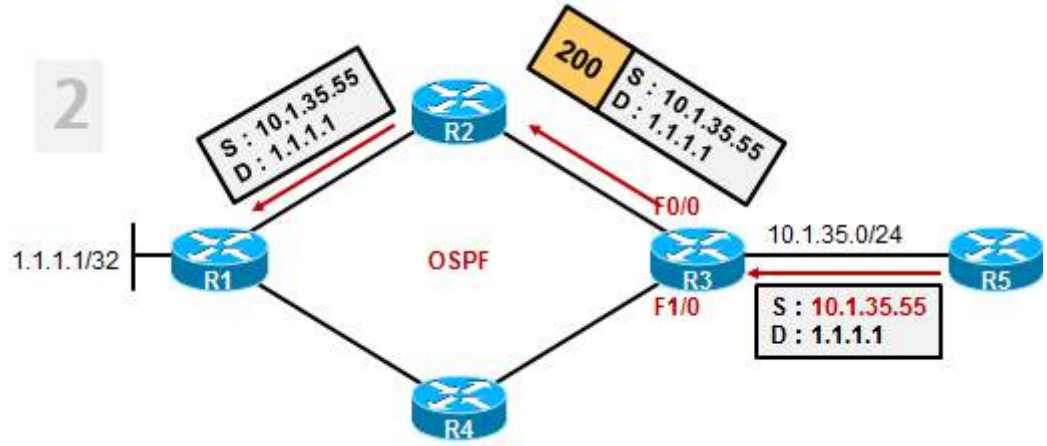
而对于支持 IPv6 的路由器，对于 MPLS 报文执行负载均衡的算法如下：

- 如果 MPLS 的有效负载是一个 IPv4 报文的话 ,负载均衡将通过对 IPv4 报文头部的源和目的 IP 地址进行哈希来实施；
- 如果 MPLS 的有效负载是一个 IPv6 报文的话，那么负载均衡就根据 IPv6 头部中的源和目的地址来进行
- 如果 MPLS 的有效负载不是一个 IPv4 报文也不是一个 IPv6 报文的话，负载均衡将通过查看底部标签的值来进行。

如果执行负载均衡的路由器是 IP 报文的入站 LSR，那么他很容易就能知道，收到的这个 IP 报文是 IPv4 还是 IPv6 的，因为它收到的是个 IP 报文，而不是标签包。

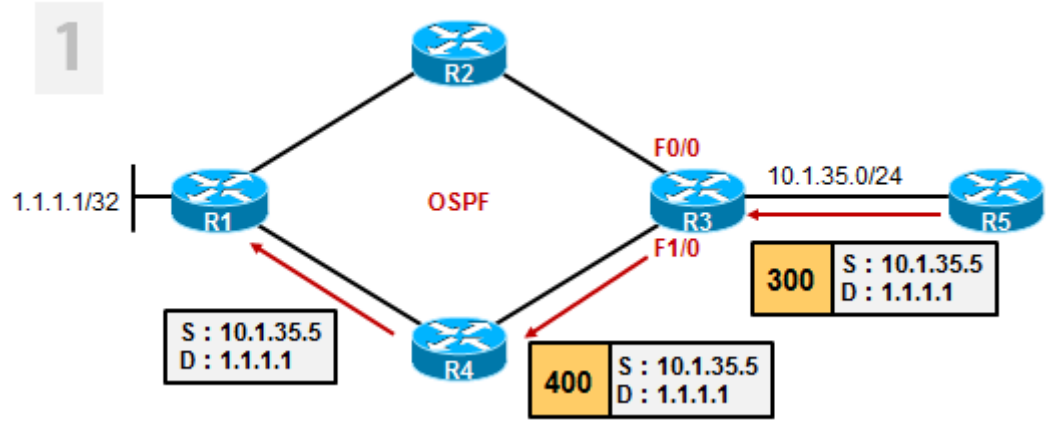
但是如果执行负载均衡的路由器是 P 路由器，它收到的是一个有多个标签的标签包，咋办？我们知道不管你有多少个标签，标签栈的后面，跟着就是 IP 的头部，而 IP 报文的第一个字段就是 version，如果为 4 就是 IPv4，如果为 6 就是 IPv6，因此 LSR 通过这个值来判断报文是 V4 还是 V6，再根据不同的 IP 版本来选择复杂均衡的算法。

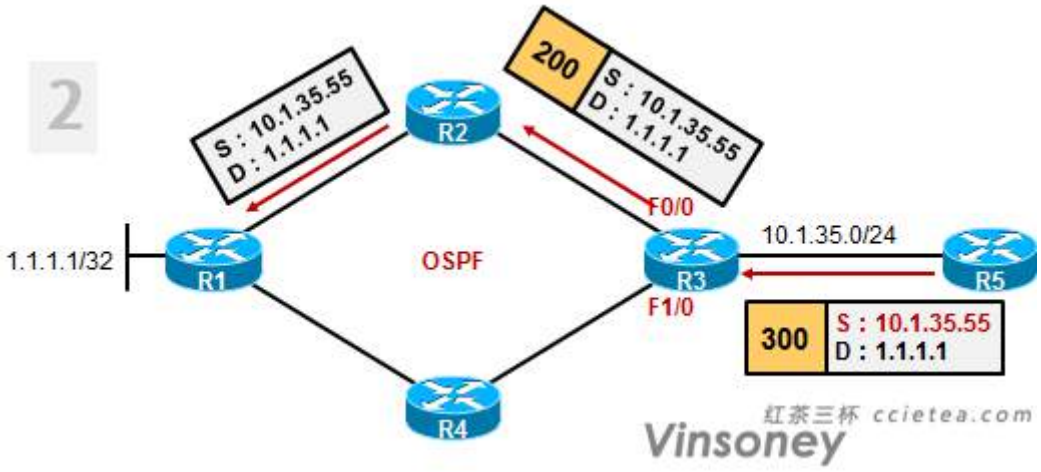




所以上图中，如果 R5 发给 R3 的是 IPv4 的数据包，那么 R3 能直接识别，并对 Ipv4 包头的源和目的地址进行哈希来实施负载均衡。

而如果 R3-R5 之间，跑起 LDP，让 R5 发送标签包，R3 收到这个标签包，查看标签栈后的字节，发现值为 4，因此确定是 IPv4 报文，于是将 IPv4 包头的源和目的地址拿出来做哈希执行负载均衡。





3 LDP

3.1 协议概述

MPLS 的基本特点就是所有的报文都是有标签的，那么如果让 OSPF、EIGRP、RIP、ISIS 这样的协议来分发标签是不可能的了。那么就需要一个新的协议，独立于所有的路由协议并且能够结合所有 IGP 一起使用，LDP 就是一个这样的协议。当然 BGP 因为运载的是外部路由，所以认为用 BGP 本身来分发标签更为有效，甚至非常完美，所以 BGP 可以实现多协议，用它来分发标签信息所产生的影响是非常小的。而且 BGP 是唯一一种在 AS 自治系统之间分发前缀的协议。

对 IP 路由表中的每一条 IGP 的 IP 前缀来说，每一台运行 LDP 协议的 LSR 都会进行本地捆绑，也就是说，为 IPv4 前缀分配标签，然后 LSR 再将该分配的标签分发给所有的 LSR 邻居。这些接收到的标签转换为远程标签 remote label，之后邻居将该远程标签和本地标签存储于一张特殊的表中，这个表就是**标签信息库 LIB**。通常一台 LDP 路由器会有多个 LDP 邻居，那么这些邻居都会给路由分配标签然后将这些标签传给自己。

在所有捆绑某一特定前缀的 remote label 中，LSR 只使用其中一个标签来确定该前缀的出站标签。RIB，也就是路由表来决定 IPv4 前缀的下一跳是什么。而 LSR 从下游 LSR 收到的远程标签中选择其路由表中到达该前缀的下一跳的标签。LSR 用这样的信息来创建它自己的**标签转发信息库 LFIB**。

注意在 CISCO IOS 中，LDP 不会为 BGP 的 IPv4 前缀捆绑标签。

3.2 LDP 邻接建立过程

1. **两个过程：邻居发现过程、会话建立过程**

2. 除了 HELLO 报文基于 UDP646 外，其他报文基于 TCP 端口号 646

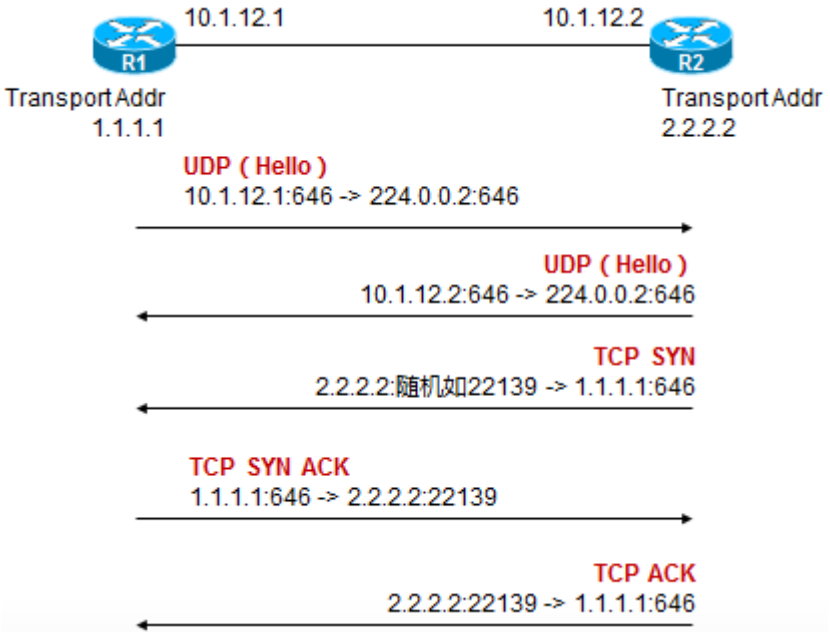
HELLO 包发向组播地址 224.0.0.2，源地址为**接口 IP**

3. 查看是否收到对方的 HELLO

验证是否收到对方的 LDP Hello

```
r1#sh mpls ldp discovery
    Local LDP Identifier:
    1.1.1.1:0
        Discovery Sources:
        Interfaces:
        FastEthernet0/0.12 (ldp): xmit/recv
        LDP Id: 2.2.2.2:0        // 2.2.2.2 为 router-id；0 为标签间隙
```

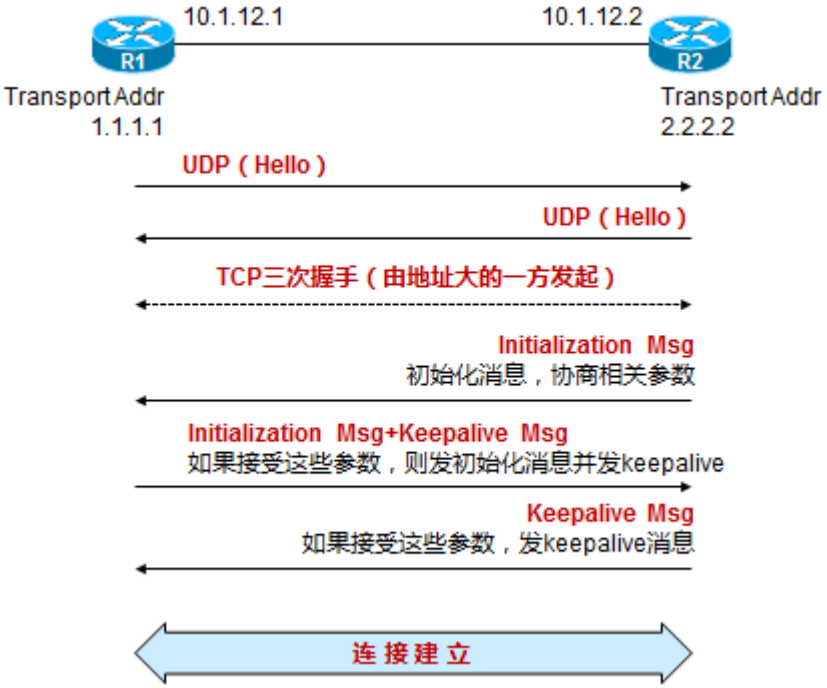
4. LDP 邻居发现



- 邻居发现是借助 UDP 的 Hello 包来进行的，这个 Hello 包源和目的端口都是 UDP646。
- 发现到邻居后，传输地址大的一方为主动发起方；注意，这里 TCP 握手报文，源地址是本地的 Transport Address，目的地址是对端的 Transport Address。传输地址为默认情况下为路由器的 LDP Router-ID。所以必须保证两个 Transport Address 之间是路由可达的。

  否则可考虑使用接口下的 mpls ldp discovery transport-address interface，将 Transport Address 配置为直联接口的 IP 地址。

5. LDP 会话建立



连接建立成功后，开始交互初始化消息，初始化消息中包含各种参数。对方也发送自己的初始化消息，并且如果接收前者的初始化消息中的各项参数，则发送一个 keep alive 消息表示接受。到此 LDP 的邻居关系就建立起来了，接下去就可以互相传递标签映射消息了。

6. LDP 非直连邻居

LDP 允许非直连邻居，这样一来邻居发现无需借助组播的 HELLO 包，而是采用单播包。

3.3 LDP ID

LDP ID 是一个 6B 的字段，是 LSR 的 LDP 标示符。包含 4B 的 LSR 标示符和 2B 的标签空间。

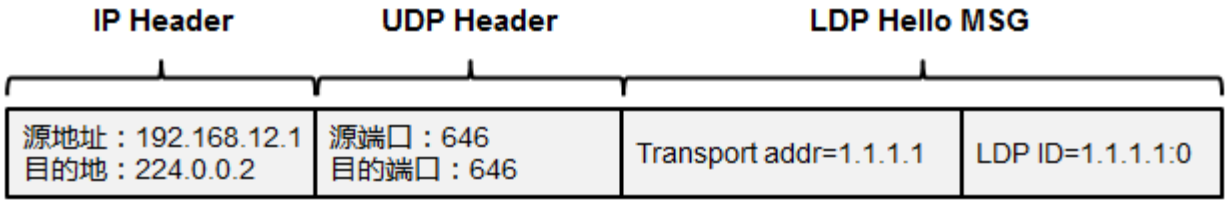
- LDP IP 的前 4B 是当存在 loopback 接口时，loopback 接口的最大 IP，如果没有 lp 接口，则是活动物理接口的最大 IP，使用如下命令可以更改：

  mpls ldp router-id *interface* [force]

  加 force 关键字则为马上生效，否则只有当前面选作 routeriD 的接口 DOWN 掉的时候，才会进行重新选择。注意：LDP 邻居的 routerID，必须在本地有可达的路由。
- 如果 2B 的标签空间都是 0 ,那么就是基于设备的标签空间 ,如果是非 0 ,就说明用的是基于接口的标签空间。那么在这种情况下，可以使用多个 LDP ID，这些 LDP ID 的前 4 字节都是相同的，但是后 2 个字节标识不同的标签空间，基于每个接口的标签空间用于 LC-ATM。

3.4 消息格式

1. HELLO 消息



LDP 邻居建立首先发送 HELLO 包（HELLO 包使用 UDP，源目端口都是 646）

LDP ID 为 6 个字节（4 字节的 IP 加 2 字节的 LABEL SpaceID）

两个路由器建立 LDP 邻居，要保证双方到对方的 LDP ID 三层可达

Transport addr 除非手工指定（mpls ldp discovery transport-address），否则一般等于 LDP ID。

注意这里（上面所描述的）其实是有三种地址：接口 IP、Transport addr、LDP ID

LDP ID 的选举和 OSPF routerID 一样

```
Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 224.0.0.2 (224.0.0.2)
User Datagram Protocol, Src Port: ldp (646), Dst Port: ldp (646)
Label Distribution Protocol
    Version: 1
    PDU Length: 30
    LSR ID: 1.1.1.1 (1.1.1.1)
    Label Space ID: 0
  Hello Message
      0... .... = U bit: Unknown bit not set
      Message Type: Hello Message (0x100)
      Message Length: 20
      Message ID: 0x00000000
    Common Hello Parameters TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: Common Hello Parameters TLV (0x400)
        TLV Length: 4
        Hold Time: 15
        0... .... .... .... = Targeted Hello: Link Hello
        .0.. .... .... .... = Hello Requested: Source does not request periodic hellos
        ..00 0000 0000 0000 = Reserved: 0x0000
    IPv4 Transport Address TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: IPv4 Transport Address TLV (0x401)
        TLV Length: 4
        IPv4 Transport Address: 1.1.1.1 (1.1.1.1)
```

使用如下命令可以看到相关信息：

R1#show mpls ldp discovery detail

```
Local LDP Identifier:
```

1.1.1.1:0

Discovery Sources:

Interfaces:

FastEthernet0/0 (ldp): xmit/recv

Enabled: Interface config

Hello interval: 5000 ms; Transport IP addr: 1.1.1.1

LDP Id: 2.2.2.2:0; no host route to transport addr

Src IP addr: 10.1.12.2; Transport IP addr: 2.2.2.2

Hold time: 15 sec; Proposed local/peer: 15/15 sec

Reachable via 2.2.2.0/24

Password: not required, none, in use

2. Initialization 消息

```
Transmission Control Protocol, Src Port: 22139 (22139), Dst Port: ldp (646), Seq: 1, Ack: 1
Label Distribution Protocol
  Version: 1
  PDU Length: 32
  LSR ID: 2.2.2.2 (2.2.2.2)
  Label Space ID: 0
⊟ Initialization Message
    0... .... = U bit: Unknown bit not set
    Message Type: Initialization Message (0x200)
    Message Length: 22
    Message ID: 0x00000001
  ⊟ Common Session Parameters TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Common Session Parameters TLV (0x500)
      TLV Length: 14
    ⊟ Parameters
        Session Protocol Version: 1
        Session KeepAlive Time: 180
        0... .... = Session Label Advertisement Discipline: Downstream Unsolicited proposed
        .0.. .... = Session Loop Detection: Loop Detection Disabled
        Session Path Vector Limit: 0
        Session Max PDU Length: 0
        Session Receiver LSR Identifier: 1.1.1.1 (1.1.1.1)
        Session Receiver Label Space Identifier: 0
```

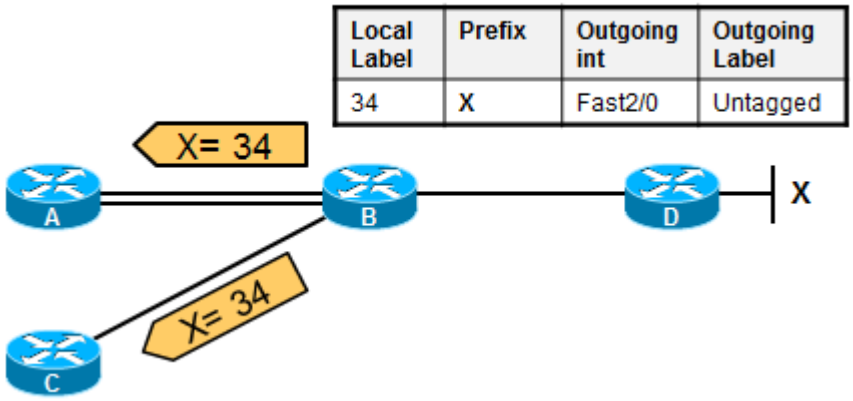
3. Label Mapping 消息

一个 LDP 报文中会承载多个标签映射消息

每个标签映射消息包含两要素:FEC TLV 和 Label TLV

```
Label Mapping Message
    0... .... = U bit: Unknown bit not set
    Message Type: Label Mapping Message (0x400)
    Message Length: 24
    Message ID: 0x00000029
  Forwarding Equivalence Classes TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Forwarding Equivalence Classes TLV (0x100)
      TLV Length: 8
    FEC Elements
      FEC Element 1
          FEC Element Type: Prefix FEC (2)
          FEC Element Address Type: IPv4 (1)
          FEC Element Length: 32
          Prefix: 9.9.0.1
  Generic Label TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Generic Label TLV (0x200)
      TLV Length: 4
      Generic Label: 3
```

3.5 标签空间



默认的 LDP 标签空间是基于平台的 per-platform 或者说基于设备的。什么意思呢，我们看 B 路由器，它为前缀 X 捆绑了标签 34，并且将这个标签捆绑信息发布给所有的 LDP 邻居，给大家的都一样，而且人人有份，都是标签 34。这就是基于平台。那么除此之外还有基于接口的标签空间。

Label space:Per-Platform

- Label Space ID 一般为 0，表示我们的标签是基于平台（Per-platform）的标签空间
- LFIB 表不包含入接口信息
- 为前缀分配的标签在本地任意 MPLS 接口可用并且会分发给所有 LSR 邻居
- 本地分配的标签会分发给邻居，如果与单个邻居有多条连接，则该标签在所有连接上均有效。那么不管本地

从哪个接口上收到一个标签包，只要有这个标签，都会对其进行交换。

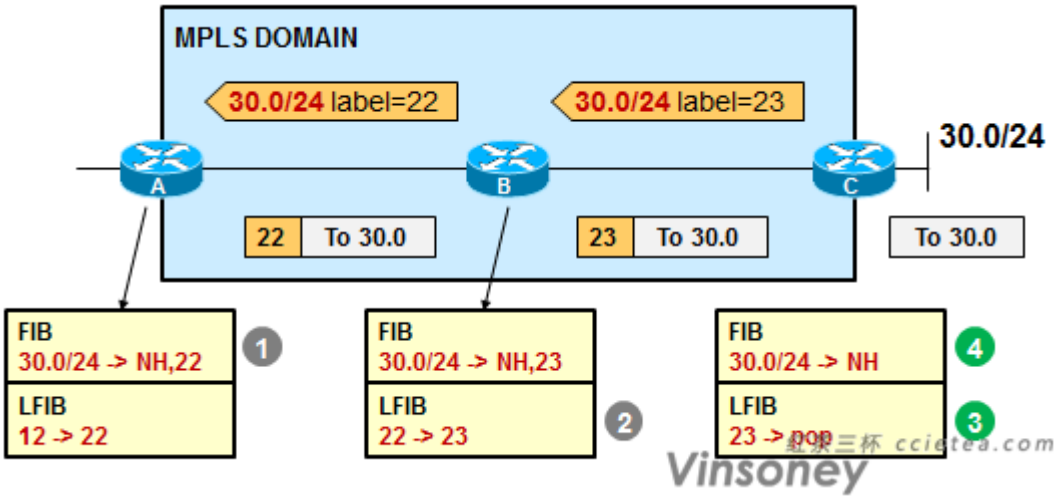
- 基于平台（per-platform）的标签空间相比于基于接口（per-interface）的标签空间安全性要更低

Negotiating Label space

- **LSRs 为每个标签空间只维护一个 LDPsession**
  基于平台（Per-platform）的标签空间只需要一条 LDPsession，即使在 LDP 邻居间存在多条冗余链路
- **基于平台（Per-platform）的标签空间 label space ID=0**
  如 LDP ID = 1.1.1.1:0

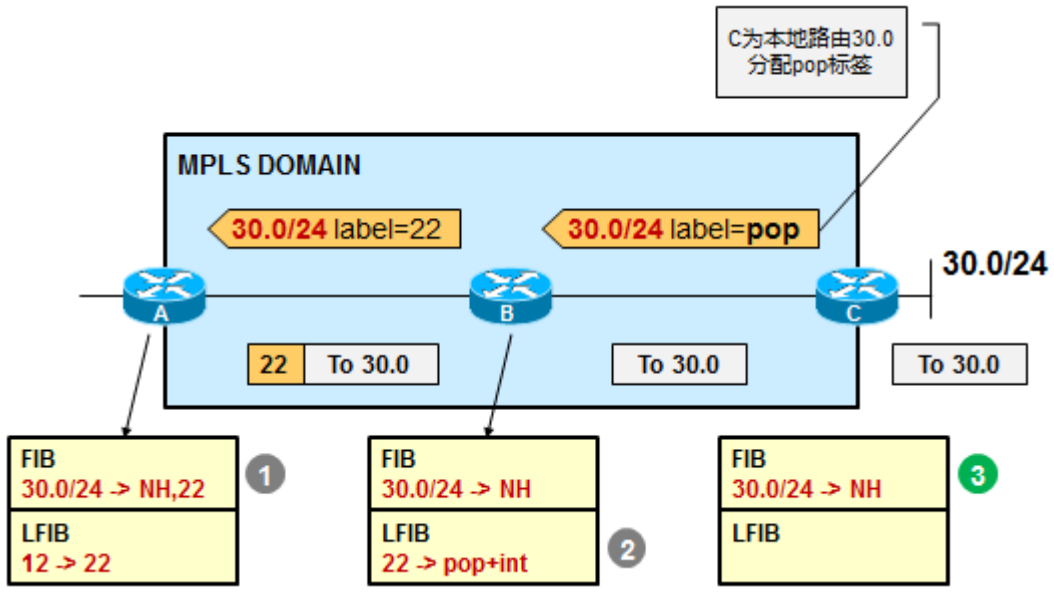
3.6 PHP 倒数第二跳弹出机制

如果没有 PHP 机制：



- 关于 30.0/24 这个前缀，C 分配的标签是 23，这个映射传递给了 B；B 本地给 30.0/24 分配的标签是 22，这个映射传递给了 A。
- 现在 A 下面有用户发送数据到 30.0 网络，A 将数据压上标签头，标签值为 22。标签包到了 B，B 将标签替换成 23，然后传递给 C。
- C 上，先查找 LFIB 表，发现要将标签弹出，于是它将标签弹出，弹出后发现是个 IP 报文，于是又去查 FIB 表，最终将这个 IP 数据包转发出去。C 进行了两次查找。这降低了转发效率。
- 标签可以在（倒数第二跳）上弹出，C 只需查找 FIB 表将收到的 IP 报文进行转发

有了 PHP 机制：



- 有了 PHP 倒数第二跳弹出机制的话，C 为本地的直连的前缀分配 POP 标签并通告给其他 LDP 邻居
- 如此一来，B 收到一个 A 发送过来的标签值为 22 的标签包，会将标签弹出得到 IP 包，再转发给 C，则 C 仅需对 IP 包进行 FIB 表的查找和转发。

LDP 在帧模式（Frame Mode）下，LSR 会为每一条路由分配一个标签；而为本地的直连路由分配的是 POP 标签。

倒数第二跳弹出机制有两种标签，一是 POP 或 implicit null，在 LDP 中标签值为 3；另一个是 explicit null，在 LDP 中标签值为 0。如果收到邻居发送来的关于某条路由分配的标签值为 3，则我发送数据给该邻居时，我会将该标签弹出，再将内层数据转给邻居。而如果邻居关于某条路由分配的标签值为 0，那么本地在转数据给邻居时，会带上标签（为 0 的），一并发给邻居。

这里要留意的是，如果收到一个标签包，标签为 0，则直接弹出标签，并将数据交给 FIB 进行查找，不会有两次查找的损耗。标签为 0 的标签包，为什么不干脆将标签去掉（分配个 13 值给路由下一跳让下一跳将标签去掉啊），为什么还要保留这个为 0 的标签头呢？这是为了在某种情况下保持网络规划的统一性，例如部署了 MPLS 的 QoS，则需使用标签包中的 EXP 字段，那么就需要有标签。在实施 QoS 时，最后一跳必须携带 exp 位，因此标签不能被弹出，需配置 mpls ldp explicit-null,此时分配给特定路由的标签值为 0 并传递给 LDP 邻居（如倒数第二跳）。

3.7 保留的标签

标签 0-15 都是被保留的标签。以下是一些有特定作用的保留标签：

- 标签 0　　显式空标签
- 标签 3　　隐式空标签

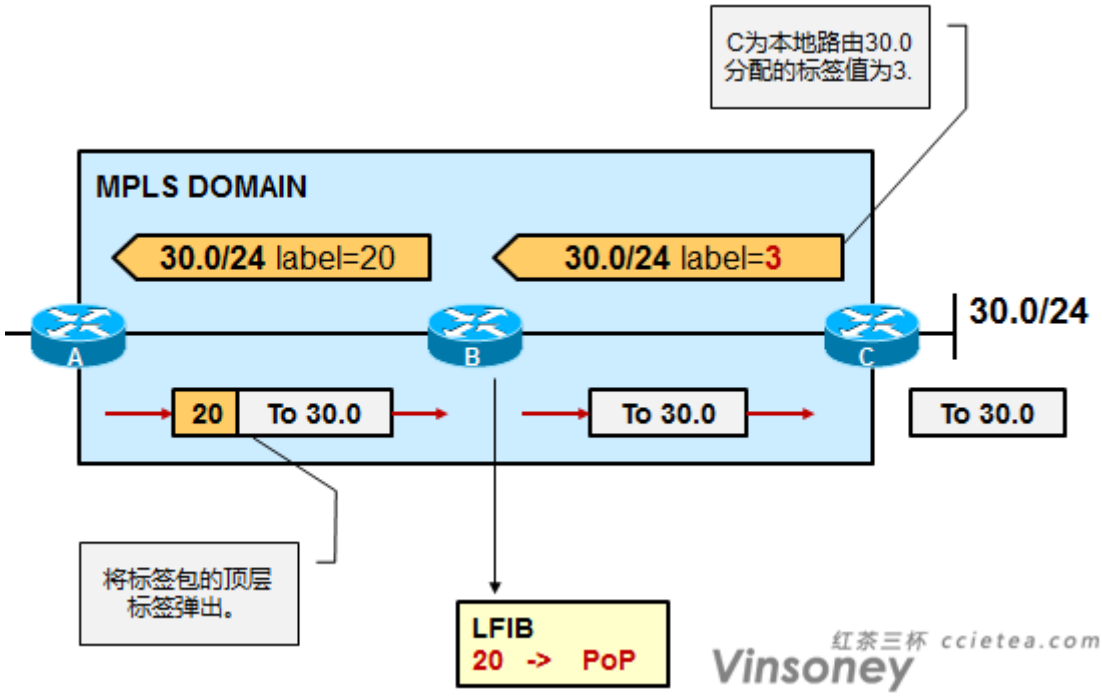
- 标签 1　　路由器报警标签
- 标签 14　　OAM 报警标签

其他 0-15 之间的被保留标签的功能目前暂时没有定义。因此我们的可用标签是 16 到 1048575（2^{20}-1）

1. 隐式空标签

在 PHP 中，我们已经了解了隐式空标签的作用，当然，隐式空标签不局限在 PHP 中。它还可以运用在标签栈中有 2、3 个或者更多的标签的报文中。在出站 LSR 上使用隐式空标签（在 LDP 中，值为 3）将会通知倒数第二跳路由器移除顶层标签，而向出站 LSR 传递的带标签报文其标签数量就会少一个，这样的话，出站 LSR 就不需要执行两个标签的查找了。注意，使用隐式空标签并不是必须将标签栈中的所有标签都弹出，而是弹出顶层标签。

尽管隐式空标签也使用了一个标签值为 3 的标签，但是标签 3 永远不会出现在 MPLS 报文的标签栈中，这也正是其叫隐式空标签的原因。

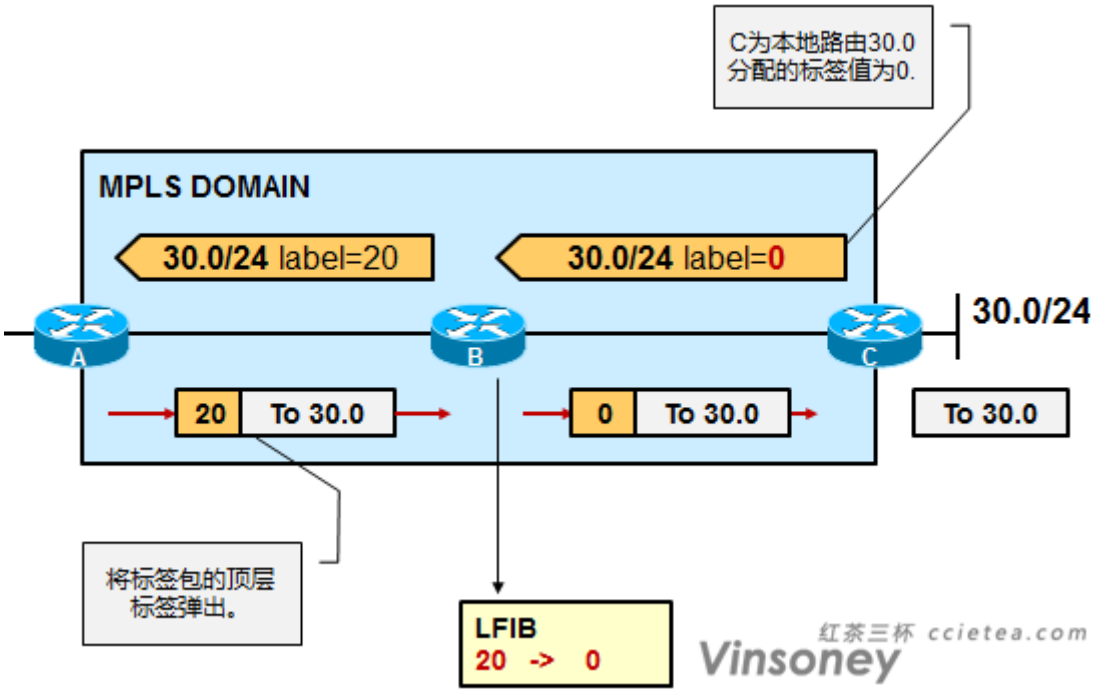


2. 显式空标签

在 IPv4 中，显示空标签为 0，ipv6 中为 2。

上面的隐式空标签已经介绍过了，它确实可以增加效率，但是也有一个问题，因为如果我收到一个下游邻居发送过来的关于某个特定前缀捆绑的隐式空标签，那么我在转发标签数据给该邻居之前，我会先将顶层标签弹出，那么这个弹出的动作，实际上是将整个顶层标签头都弹出了，也就是连带着标签字段、EXP 等字段都弹出了，而 EXP 我们知道，用于做 QoS 的，它也被弹出了，意味着这里就丢失了用于 QoS 的部分信息。

因此我们又定义了显式空标签，用于应对上面描述的场景。



我们看上图，C 针对 30.0/24 的前缀捆绑了标签 0，也就是显式空标签，然后将标签映射发给 B，B 也产生自己的标签映射然后发给 A。那么这时候，如果 B 收到来自 A 的一个标签包，顶层标签的值为 20，那么 B 查找自己的 LFIB，发现要标签要转换成 0。于是，B 将顶层标签替换成 0，然后转发给 C，那么这个时候对于 C 来说，它就收到了一个标签值为 0 的标签包，C 不能通过在 LFIB 中查找标签值 0 来转发这样的报文，因为这个标签值可以分配给多个 FEC，C 只是仅仅弹出 0 标签也就是显式空标签，之后不得不进行另外一种查找 ,虽然这里不得不进行两次查找 ,但是 C 就可以通过查看标签头的 EXP 位来获得该报文的 QoS 信息了。

3. 路由器报警标签

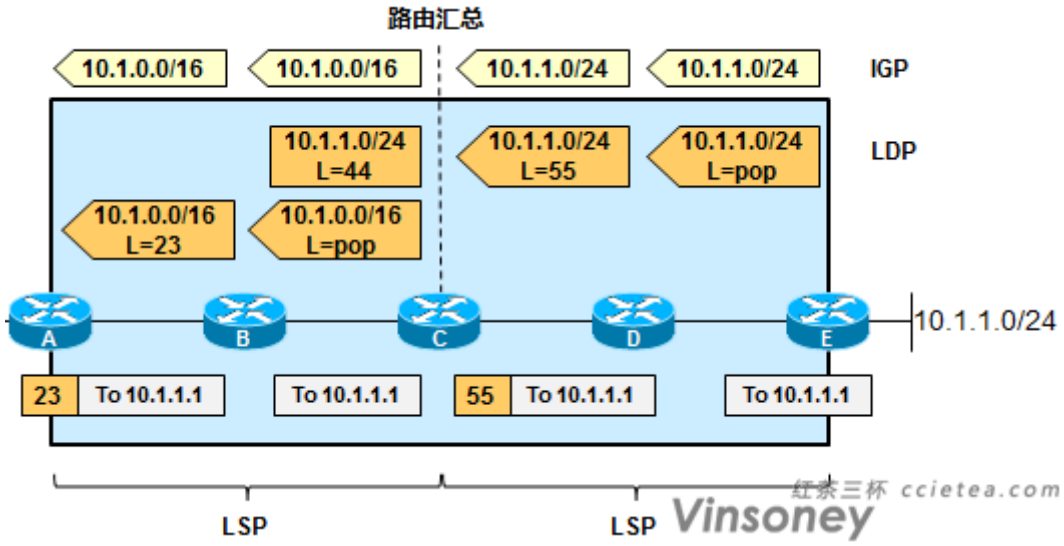
标签值为 1，这个标签可以出现在标签栈的任何位置，除了栈底位外。

当路由器报警标签位于栈顶时，它向 LSR 发出警告说该报文需特别注意。这样一来该报文就不会通过硬件传输，而是通过软件进程传输。一旦这个报文开始被转发，标签 1 首先被移除，接下来 LSR 在 LFIB 中对标签栈中的下一个标签进行查找然后执行相应的标签操作(添加、移除、交换) ,标签 1 又会被添加到标签栈的顶部，最后才被转发出去。

4. OAM 报警标签

OAM 基本上用于错误检测、定位和监控实施。该标签将普通报文和 OAM 报文区分开来。CISCO IOS 不使用标签 14，它会执行 MPLS OAM 但不是通过标签 14 来实现。

3.8 路由汇总对 MPLS 的影响



E 上有条路由 10.1.1.0/24 被通告了出来，现在在 C 上做了汇总，汇总路由是 10.1.0.0/16 并向 B 通告。

- 路由汇总将原先的一段 LSP 分割成了两段
- A 将数据压上标签 23，到了 B，将标签头弹出交给 C（PHP 机制）；C 收到这个 IP 包去查找 FIB 表，又将包压上标签 55 交给 D，D 最后查找 LFIB 并将标签弹出将 IP 包丢给 E，E 将该 IP 包转发到目的地。
- 在这个环境中看似没有什么问题，但是路由汇总在点到点的 LSP 环境下就有问题了，例如 MPLS VPN、TE 等。

3.9 MPLS 模式

1. **分配模式:Label Allocation**

// 本地为一条路由前缀绑定一个标签的前提条件

独立控制模式:Independent Control

只要本地通过 IGP 学习到路由前缀，就会为这条路由前缀分配标签(本地也会为直连路由分配 POP 标签)

有序控制模式:Odered Control

本地通过 IGP 学习到路由前缀，但必须该路由前缀的下一跳路由器将该前缀所对应的标签映射消息通告给本地，本地才会为该前缀分配标签

2. **分发模式:Label Distribution**

// 本地将一个标签映射消息通告给邻居的前提条件

下游主动模式:Downstream Unsolicited

本地会主动将所生成的标签映射消息通告给所有 LDP 邻居

下游按需模式:Downstream On Demand

只有邻居向本地请求某条前缀的标签映射消息时,本地才会通告标签映射消息给邻居

3. **保留模式:Label Retention**

// 本地是否会在数据库中保留从邻居接收到的所有标签映射消息

自由模式: Liberal Retention

本地将从邻居接收的所有标签映射消息都保存在数据库中

保守模式: Conservative Retention

本地仅保存最优路由下一跳邻居所通告的该路由前缀的标签映射消息

- **标签空间:Label Space**

// 本地所通告出去的标签是对局部(接口)有意义还是对全局有意义

基于平台: Per-Platform

本地通告出去的标签映射消息对全局有意义,从不同的接口通告出去的同一 FEC 所对应的标签相同

基于接口: Per-Interface

本地通告出去的标签映射消息对局部有意义,从不同的接口通告出去的同一 FEC 所对应的标签不同

【注意】帧模式(Frame Mode)的标签行为

分配模式(Label Allocation)： 独立控制模式(Independent Control)

分发模式(Label Distribution)： 下游主动模式(Downstream Unsolicited)

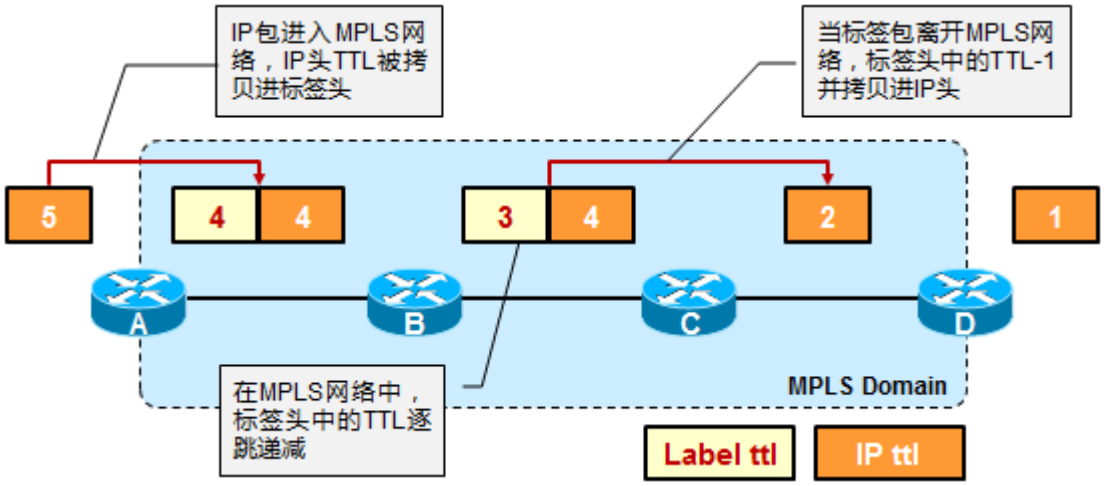
保留模式(Label Retention)： 自由模式(Liberal Retention)

标签空间(Label Space)： 基于平台(Per-Platform)

3.10 LOOP Detection

- LDP 的环路检测机制依赖于 IGP 协议
- 如果出现的环路（一般是 IGP 出了问题，如静态路由的配置错误），标签头中的 TTL 将防止标签包无止尽的被转发
- 标签头中的 TTL 与 IP 头中的 TTL 是一样的 ,通常拷贝自 IP 头中的 TTL 值(当一个 IP 包进入 MPLS 网络时), 这是 TTL propatation

1. 传统的 TTL 操作



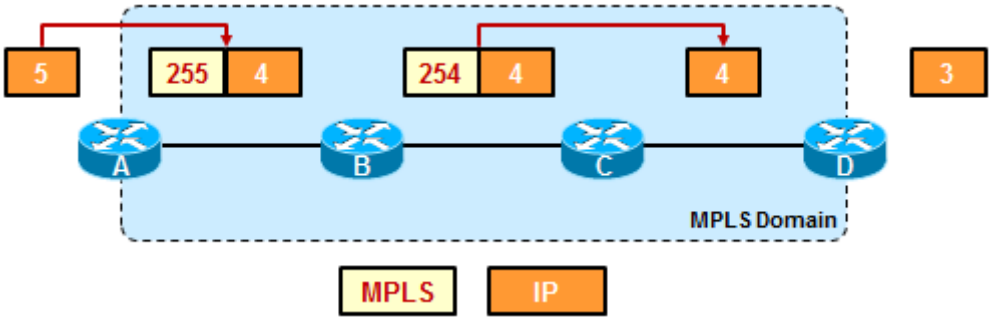
数据包被 A 压上标签 ,标签的头的 TTL 值拷贝 IP 头中的 TTL(当然要先减 1),并且随着数据帧在 MPLS 网络中传输，MPLS 头中的 TTL 值在递减，到 0 则丢弃，直到数据帧出了 MPLS 域，那么 IP 头中的 TTL 才开始工作。在出站时，标签头中的 TTL 减 1 后拷贝到 IP 头中的 TTL。

这里实际上存在一定的隐患，例如使用 traceroute，可能会暴露网络内部结构（TTL=0 后，路由器会返回差错消息，就可能会暴露网络信息）。

我们来看一下，为什么会暴露网络信息？假设 A 下面有个路由器 E，E 发了一个数据包要到 D 下的 F，数据包被 A 打上标签转给 B ,假设在 BC 之间出现了环路随后标签 TTL 递减至 0 ,由于 TTL 值设置的非常小，因此很快，TTL 就递减到了 0，这时候 B（报文丢弃者），就会发送 ICMP 出错消息回 E，那么这时候如果 B 有 E 的路由（或标签），B 就会直接将 ICMP 出错消息发回 E，如果没有，B 会继续将出错信息打上标签然后逐渐传递到 D，D 作为 PE，有关于 E 的路由，于是再装上标签送回给 E。那么无论什么情况，最终 E 都能收到由 B 产生的 ICMP 差错消息。

这时可以关闭 TTL propagation（CISCO 路由器默认开启）

2. 关闭 TTL propagation



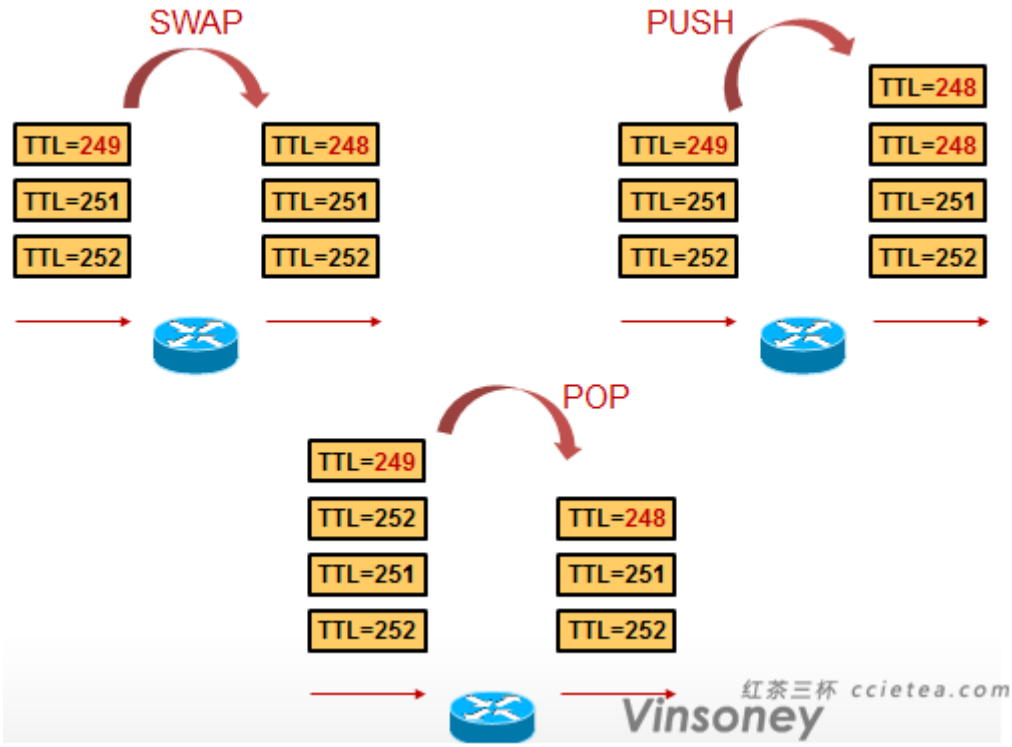
在 A 设备（一般在边界设备）上配置该特性 no mpls ip propagate-ttl

这样一来在边界设备上将 IP 包加标签头的时候，就不去拷贝原 IP 头里的 TTL 值了，而是用一个如 255 的 TTL 放入标签头。关闭 TTL propagation 可以避免 MPLS 网络被暴露（通过 traceroute 方式）。

```
no mpls ip propagate-ttl [ forwarded | local ]
```

forwarded 关键字表示这条命令针对穿越本路由器的流量生效。Local 关键字表示针对本地产生的流量生效

3. 补充知识点：在 SWAP、PUSH、POP 操作中标签到标签的 TTL 扩散行为



前面描述的是 IP-标签包的 TTL 扩散过程，现在来看一下标签到标签的 TTL 扩散过程。

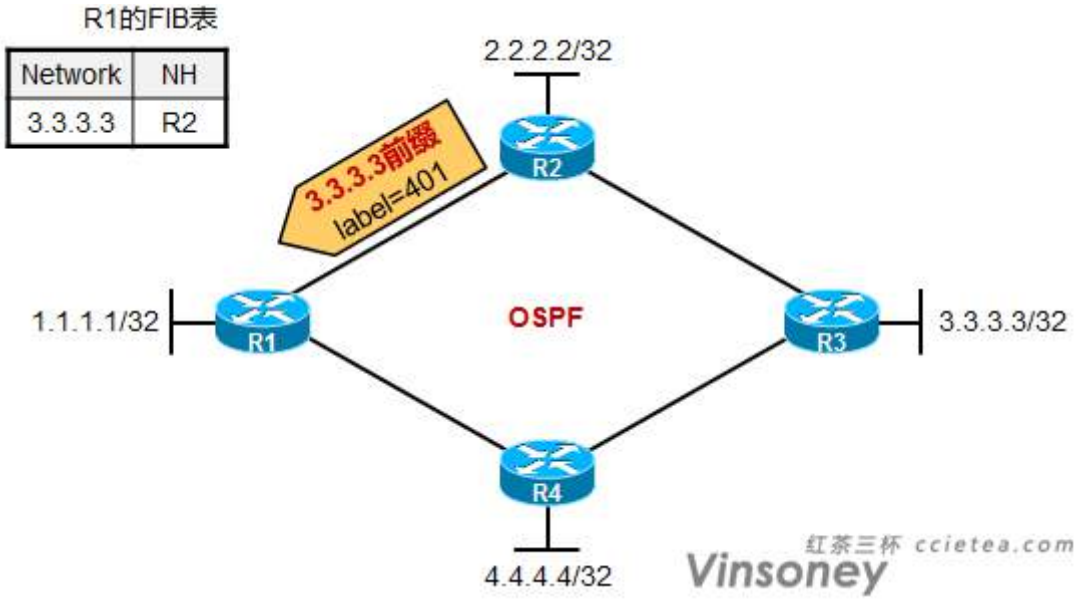
SWAP 这个过程很好理解，注意，LSR 在处理标签栈中有多个标签的标签数据时，只会处理顶层标签。因此 SWAP 这个过程，标签在交换后，入站标签 TTL -1，然后拷贝到出站标签 TTL。

PUSH 也是类似的理解，只对顶层标签操作，首先入站顶层标签的 TTL249 先减 1，然后新压入的标签头 TTL 拷贝这个值。

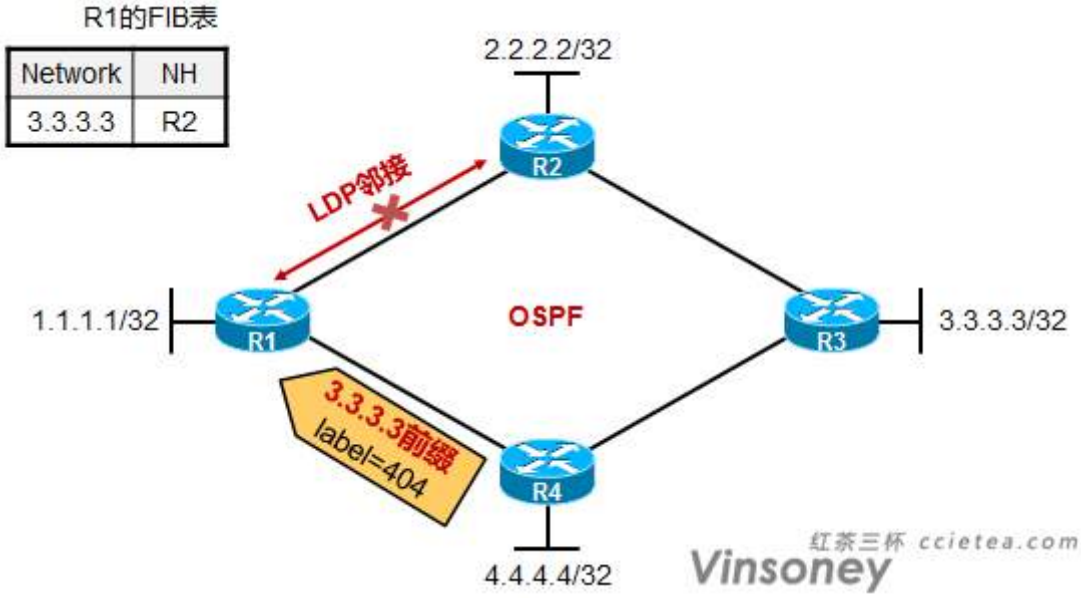
POP 则是顶层标签的 TTL 先减去 1，然后弹出，新的 TTL 值被写入到出站数据的顶层标签上。

3.11 LDP 与 IGP 的同步

1. 关于 LDP 与 IGP 的同步



MPLS 网络中的一个很重要的问题就是 LDP 和 IGP 的同步，所谓的**同步的是意思是，IGP 和 LDP 都认可某条链路为待转发报文的出站链路的**。例如上图中，四台路由器都运行 OSPF，同时激活 LDP。那么在 R1 上，如果其去往 3.3.3.3/32 的路由，在路由表中下一跳为 R2（假设我们调了 cost），同时又收到了 R2 发过来的针对 3.3.3.3/32 前缀捆绑的标签 401，那么这时候，IGP 和 LDP 就同步了，R1 可以正常的使用 R2 作为下一跳来转发 MPLS 标签报文。



然而可能出现这样一个问题，R1、R2 之间的 LDP 连接，由于某种原因断掉了，但是 R1、R2 之间的 OSPF 邻接没 DOWN，那么 R1 的路由表中，关于 3.3.3.3/32 依然选择 R2 作为下一跳，这就出问题了。R1 上的 LFIB 里，关于 3.3.3.3/32 的条目可能是这样的：

R1#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop

104 **Untagged** 3.3.3.3/32 0 Fa0/0 10.1.12.2

这样，R1 发送数据去往 3.3.3.3 就有直接以 IPv4 报文的形式发送出去了，这在本拓扑中看似没什么问题，确实能够通，但是，在 MPLS VPN 环境中就可能出问题了。因为报文在 MPLS VPN Backbone 中传输往往是需要带标签的。

另一个造成 IGP 和 LDP 不同步的例子是，LSR 重启的时候，IGP 可以很快就建立起邻接关系，而 LDP 的会话建立起来可能就慢点，也就是说，在 LFIB 装载必须的信息来进行正确的标签交换之前，IGP 的转发就已经开始了。于是报文可能就错误的转发，或者在进入 MPLS 网络后在某处被丢弃。

可以使用 MPLS LDP-IGP 同步来解决这个问题。注意，MPLS LDP-IGP 的解决方案不能用于 BGP 的标签分发。

2. MPLS LDP-IGP 同步如何工作

当在一个接口上激活了“MPLS LDP-IGP 同步“之后，IGP 将会通告该链路的度量值为最大，一直到同步完成或者 LDP 会话在该接口上成功建立。在 OSPF 中，metric 最大为 65535。这样一来，可以保证在 LDP 处于断开状态的时候不会有路径会经过该接口。在 LDP 会话成功建立并且标签捆绑已经开始进行交换以后，IGP 才会用正常的 IGP 度量值来通告该链路。这个时候穿越这个接口的流量就是进行标签交换的流量了。基本上说，如果 LDP 会话没有建立，OSPF 是不会在这个链路上建立邻接关系的，压根不发 HELLO（当然，这里有个基本上说的字眼，也就说，还有二般情况）。

一直到 LDP 会话成功建立，或者“同步保持时间”超时，否则 OSPF 邻接关系是不会建立的。这里的同步表示本地标签捆绑已经通过 LDP 会话发送给了 LDP 对等体。但是如果路由器上 A 激活了“MPLS LDP-IGP 同步 “，并且 A 和 B 之间只有一条链路，而没有其他可用路径到 B 的话，OSPF 邻接关系将永远不会建立。因为 OSPF 会等待 LDP 会话的建立，但是 LDP 会话根本无法建立，因为 A 没办法在它的路由表中学习到 B 的 LDP routerID 路由，这样就进入了一个死循环，OSPF 和 LDP 的邻接关系将永远无法建立。对于这种情况，LDP-IGP 同步会无条件激活 OSPF 的邻接关系，这样一来链路将通告最大的 metric 值，直到同步完成。

然而在某些环境中，LDP 会话的这个问题可能永远无法解决，因此可能并不希望永远等待 IGP 邻接关系的建立。那么可以通过配置“同步保持时间”。

3. MPLS LDP-IGP 同步的配置

MPLS LDP-IGP 同步是在 IGP 进程中启用的，它会应用到所有运行了该 IGP 的接口上

```
router ospf 1
  mpls ldp sync
```

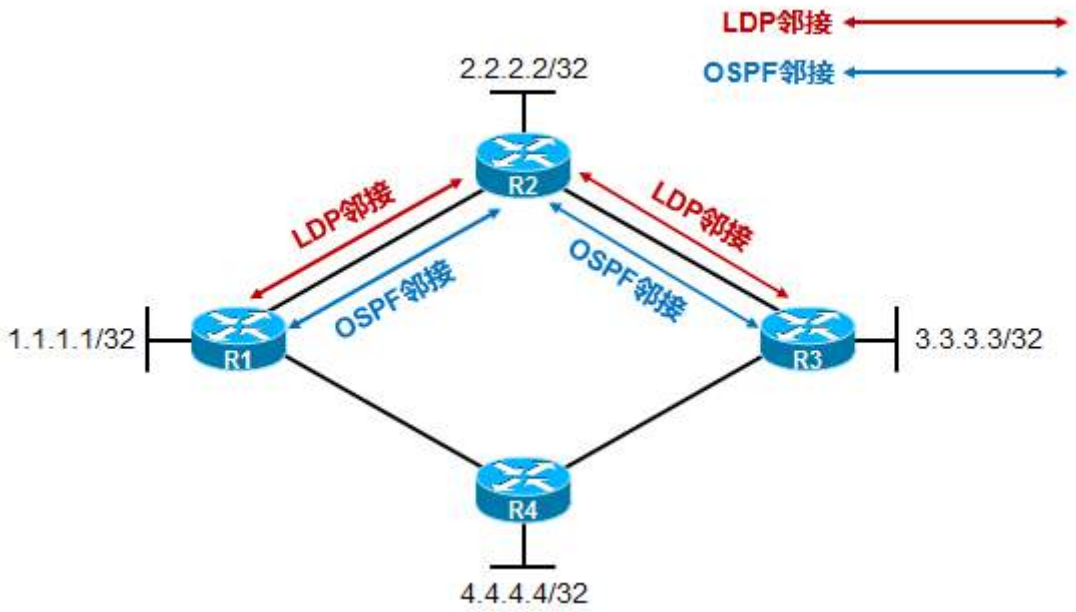
可以在某个特定的接口上使用 no mpls ldp igp sync 命令来关闭接口的同步功能。

默认情况下，如果同步没有完成，IGP 并没有明确在建立邻接关系之前所需等待的时间，可以通过：

mpls ldp igp sync holddown *msecs*

这条全局命令来进行修改。在该计时器到期之后，IGP 将在该链路上建立邻接关系，一旦 IGP 的邻接关系建立成功同时 LDP 会话还未同步的话，IGP 将通告该链路的度量值为最大。

4. MPLS LDP-IGP 同步示例



为了先保证实验环境的简洁 ,我先忽略 R4 以及 R4 直连链路的存在 ,大家当他不存在就好 ,等会儿丫才上场。

R1、R2、R3 运行 OSPF，通告直连及自己的 Loopback 口。并且在直连接口上都激活 LDP。

现在是一个正常的情况，

R1#show mpls forwarding-table （这是 R1 的 LFIB 表）

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
101	204	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.12.2
104	**201**	3.3.3.3/32	0	Fa0/0	10.1.12.2
105	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2

R1 的路由表如下（直连路由忽略了）:

```
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/2] via 10.1.12.2, 00:00:37, FastEthernet0/0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/3] via 10.1.12.2, 00:00:37, FastEthernet0/0
O       10.1.23.0 [110/2] via 10.1.12.2, 00:00:37, FastEthernet0/0
O       10.1.34.0 [110/3] via 10.1.12.2, 00:00:39, FastEthernet0/0
```

现在我们将 R2 上，完成如下配置：

```
access-list 100 deny    udp any any eq 646
access-list 100 permit ip any any
```

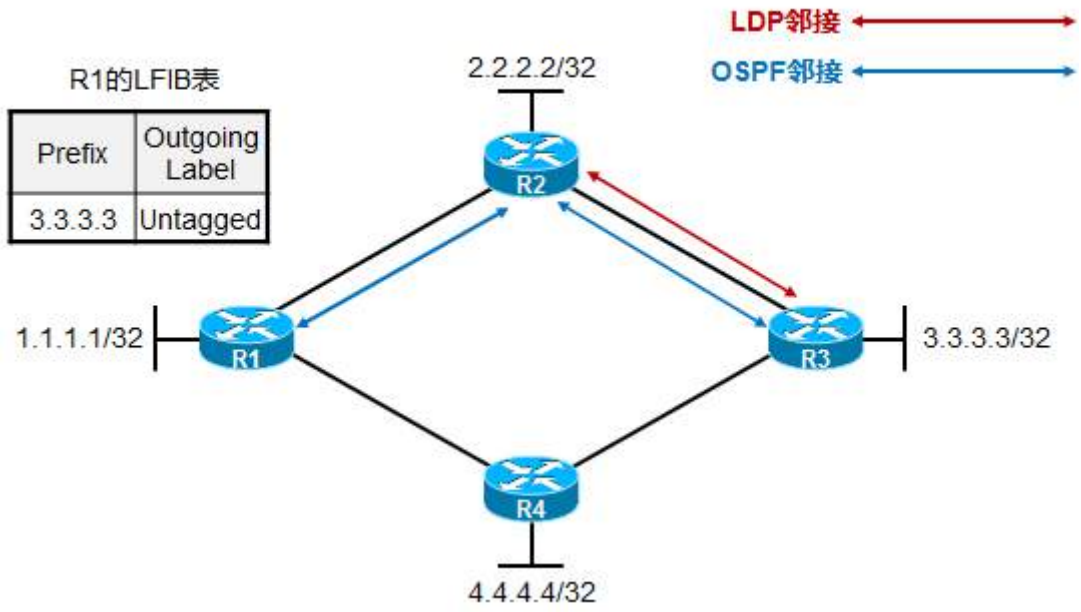
然后将 ACL 应用在 R2 上，连接 R1 的接口上，in 方向

这样，R2 将忽略掉接口上收到的，来自 R1 的 LDP hello 包，然后 R1-R2 之间的 LDP 邻接过一会就 DOWN 了。这时候：

R1#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
101	**Untagged**	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	**Untagged**	2.2.2.2/32	0	Fa0/0	10.1.12.2
104	**Untagged**	3.3.3.3/32	0	Fa0/0	10.1.12.2
105	**Untagged**	10.1.23.0/24	0	Fa0/0	10.1.12.2

我们看到 R1 上，相关路由前缀的 outgoing label 都是 untagged。而 R1 的路由表暂时没有任何变化。在现在这个环境中，虽然 R1 仍然能够 ping 通 3.3.3.3，但是实际上已经出问题了，因为这是直接走的 IPv4 包，而不是标签包。



现在我们在 R1、R2 上配置 MPLS LDP-IGP 同步：

```
router ospf 1
  mpls ldp sync
```

注意，这时候，虽然 R1、R2 之间的 LDP 邻接已经断了，原则上说，LDP 邻接不起来，在开启同步的情况下，

OSPF 邻接关系是无法建立的，但是这里 R1 只有一条可用路径到达 R2，因此 OSPF 邻接无条件建立。与此同时 R1 及 R2 对外通告 10.1.12.0/24 这段直连链路，将以最大的 metric 65535 来通告。这样做的目的是，如果网络环境是冗余链路的环境，那么可以让从 IGP 的角度让这条链路的 metric 最差，从而使得 LDP 路径绕开这段链路。我们在 R3 上来瞄一眼：

R3#show ip route

```
      1.0.0.0/32 is subnetted, 1 subnets
O        1.1.1.1 [110/65537] via 10.1.23.2, 00:09:07, FastEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
O        2.2.2.2 [110/2] via 10.1.23.2, 00:09:07, FastEthernet0/0
      10.0.0.0/24 is subnetted, 3 subnets
O        10.1.12.0 [110/65536] via 10.1.23.2, 00:09:07, FastEthernet0/0
```

我们看到，由于 R2 更新出来的 LSA 中，关于 10.1.12.0/24 这个直连网段，metric 设置成了 65535，因而，在 R3 的路由表里，我们看到 1.1.1.1/32 及 10.1.12.0/24 的路由 metric 都离奇的大。再看一下 R2 产生的 1 类 LSA：

R3#show ip ospf database router 2.2.2.2

```
            OSPF Router with ID (3.3.3.3) (Process ID 1)
                Router Link States (Area 0)
  LS age: 705
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 2.2.2.2
  Advertising Router: 2.2.2.2
  LS Seq Number: 8000001F
  Checksum: 0x580A
  Length: 60
  Number of Links: 3

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 2.2.2.2
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1                          !! 2.2.2.2 链路的 metric 不变

    Link connected to: a Transit Network
```

```
    (Link ID) Designated Router address: 10.1.23.2
    (Link Data) Router Interface address: 10.1.23.2
     Number of TOS metrics: 0
      TOS 0 Metrics: 1                          !! 10.1.23.2 的也不变

   Link connected to: a Transit Network
    (Link ID) Designated Router address: 10.1.12.2
    (Link Data) Router Interface address: 10.1.12.2
     Number of TOS metrics: 0
      TOS 0 Metrics: 65535                      !!出问题的链路,metric 被设置成了 65535
```

R1 的路由表如下：

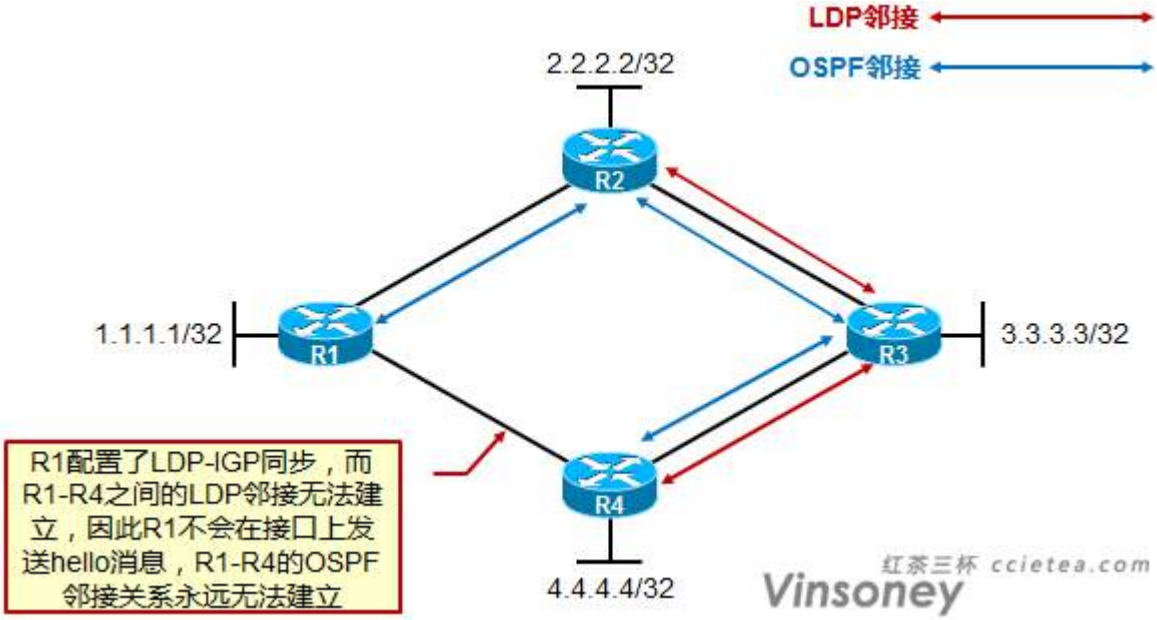
```
O        2.2.2.2 [110/65536] via 10.1.12.2, 00:00:47, FastEthernet0/0
O        3.3.3.3 [110/65537] via 10.1.12.2, 00:00:47, FastEthernet0/0
O        10.1.23.0 [110/65536] via 10.1.12.2, 00:00:48, FastEthernet0/0
```

看到没 R1 将直连链路 10.1.12.0 的 cost 调成了 65545。

R1#sh ip os mpls ldp interface fa0/0

```
FastEthernet0/0
  Process ID 1, Area 0
  LDP is not configured through LDP autoconfig
  LDP-IGP Synchronization : Required
  Holddown timer is not configured
  Interface is up and sending maximum metric
```

现在我们将 R4 加进来，注意，R4 在这个时候只配 OSPF，先不在连接 R1 的接口上配置 mpls ip（或者使用 ACL 过滤掉 LDP 报文），这样使得 R1、R4 之间的 LDP 邻接无法建立，我们来观察一下现象。

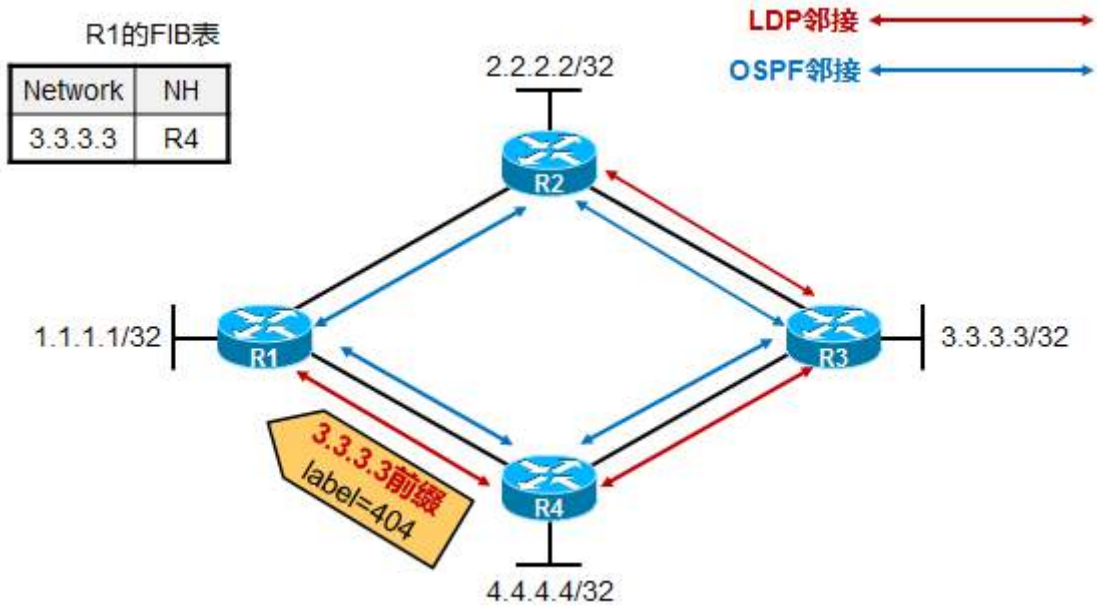


注意，由于 R1 激活了 MPLS LDP-IGP 同步，因此，在 R1-R4 之间的 LDP 邻接关系建立起来之前，R1 上连接 R4 的接口是不会发送 OSPF HELLO 包的，也就是说 R1-R4 的 OSPF 邻接关系是永远无法建立的。当然，我们也不希望看着 R1-R4 这么拧巴下去，对谁都不好，是吧？ 所以在 R1 上来配置个：

```
mpls ldp igp sync holddown 5000
```

将同步 holddown 计时器设置为 5S，这样一来 5S 超时后，R1-R4 就建立起来了 OSPF 邻接关系。

又或者，我让 R1-R4 之间的 LDP 邻接建立起来，那么 R1-R4 之间的 OSPF 自然也就起来了。好了，我们现在让 R1-R4 之间的 LDP 邻接关系起来，那么随之 OSPF 邻接关系马上也会自动建立起来。



现在一来，由于 R1 上有 MPLS LDP-IGP 同步，R1 将 10.1.12.0/24 这段链路 metric 调到 65535（其实这时候 R1-R2 之间的 OSPF 邻接关系应该 DOWN 掉的，保存配置将 R1 重启，会发现 R1-R2 之间的 OSPF 邻接关系起不来了），使得 R1 优选 R4 作为去往 3.3.3.3 网络的下一跳，同时 R1-R4 之间又维持着 LDP 邻接关系，

因此，R1 将只采用 R4 关于 3.3.3.3 的标签映射，也就是使用标签 404 来发送标签包到 3.3.3.3。

3.12 MTU 问题

```
nterface fast0/0
 mpls mtu mtu-size
```

接口级命令，修改该接口的 MPLS mtu，要注意 mpls mtu 不能大于接口 mtu

4 MPLS 配置（frame mode）

4.1 基础命令

1. **基础配置**

```
lp cef
!
mpls ldp router-id loopback0                    !! ldp 的 routerid 使用 loopback 口 IP
mpls label protocol ldp                         !! 标签协议使用 LDP（默认就是 LDP）
mpls label range 100 199                        !! 指定本地标签的范围，这个在实验中可以极大的方便观
察现象及排错
!
Interface loopback
   Ip address 10.1.255.1 255.255.255.255
Interface fast 0/0
   Ip address 10.1.12.1 255.255.255.0
   mpls ip                                      !! 接口上激活 mpls，实际上是激活 ldp
```

2. 邻居关系建立

Router(config)# mpls ldp discovery hello interval *seconds*

修改 ldp hello 消息发送间隔，默认 5S

Router(config)# mpls ldp discovery holdtime *seconds*

修改 ldp holdtime，默认 15S

如果两个 LDP 对等体的 LDP 保持时间配置的不同，那么其中较小的那个值将被用作 LDP 发现的保持时间。注意 LDP HELLO 消息中没有携带 hello interval 时间，只携带了 holdtime，因此如果两端配置了不同的 holdtime，那么小的那么值将被用作 LDP 发现的保持时间。Cisco IOS 可能会对已配置的 LDP Hello 间隔进行重写。

Router(config)# mpls ldp backoff *initial-backoff maximum-backoff*

initial-backoff 默认 15s、maximum-backoff 默认 120s。

当两台 LDP 邻居对等体在交换参数时发现又不匹配的话，这条命令可以减缓两台 LDP LSR 之间的尝试性 LDP 会话建立时间，如果会话建立失败，那么下一次再尝试的时间间隔会成倍数增长，直到 maximum-backoff 计时器超时。

4.2 Show 及 debug

```
Show ip cef detail
Show tcp brief
Show mpls ldp discovery detail
Show mpls ldp parameters                    //查看协议的基本参数
Show mpls ldp neighbor
Show mpls ldp discovery
show mpls forwarding-table
show mpls ldp bindings
```

```
show mpls ldp parameters
Protocol version: 1
Session hold time: 180 sec; keep alive interval: 60 sec
Discovery hello: holdtime: 15 sec; interval: 5 sec
Discovery targeted hello: holdtime: 90 sec; interval: 10 sec
```

Downstream on Demand max hop count: 255

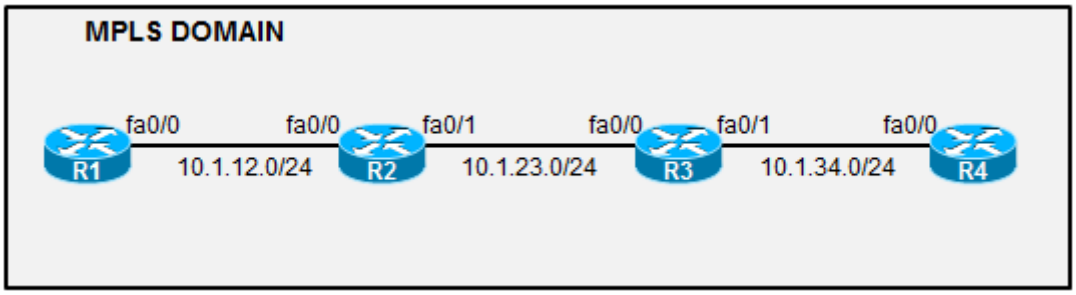
Downstream on Demand Path Vector Limit: 255

LDP for targeted sessions

LDP initial/maximum backoff: 15/120 sec

LDP loop detection: off

4.3 基础实验 1



1. **实验环境**
   - R1、R2、R3、R4 运行 OSPF，宣告直连接口，以及 Loopback 接口，Loopback 口 IP 为 **x.x.x.x/32**，x 为设备编号，该 IP 同时为 LDP routerID。
   - 设备互联网段如图所示 ,例如 10.1.23.0/24 ,这是 R2-R3 互联地址段 ,那么 R2 的接口 IP 就是 10.1.23.2，R3 的接口 IP 就是 10.1.23.3、
   - 在所有设备上激活 LDP，为了方便观察现象，为每台设备指定 label range，如 R1 的 label range 为 100 199，R2 的为 200 299，其他设备依此类推。

2. **实验需求**
   - 认识 FIB、LIB、LFIB 表
   - 了解 LDP 邻居关系建立过程
   - 了解数据在 MPLS 域中的转发过程

3. **实验配置**

R1 的配置如下：

```
Ip cef                                    !! 注意，运行 MPLS，IP cef 必须打开
Interface fas0/0
  Ip address 10.1.12.1 255.255.255.0
Interface loopback0
```

```
   Ip address 1.1.1.1 255.255.255.255
!
router ospf 1
   router-id 1.1.1.1
   network 10.1.12.1 0.0.0.0 area 0
   network 1.1.1.1 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 100 199
interface fast0/0
   mpls ip
```

R2 的配置如下：

```
Ip cef
Interface fas0/0
   Ip address 10.1.12.2 255.255.255.0
Interface fas1/0
   Ip address 10.1.23.2 255.255.255.0
Interface loopback0
   Ip address 2.2.2.2 255.255.255.255
!
router ospf 1
   router-id    2.2.2.2
   network 10.1.12.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   network 2.2.2.2 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 200 299
interface fast0/0
   mpls ip
interface fast1/0
   mpls ip
```

R3 的配置如下：

```
ip cef
Interface fas0/0
   Ip address 10.1.23.3 255.255.255.0
Interface fas1/0
   Ip address 10.1.34.3 255.255.255.0
Interface loopback0
   Ip address 3.3.3.3 255.255.255.255
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   network 3.3.3.3 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 300 399
interface fast0/0
   mpls ip
interface fast1/0
   mpls ip
```

R4 的配置如下：

```
ip cef
Interface fas0/0
   Ip address 10.1.34.4 255.255.255.0
Interface loopback0
   Ip address 4.4.4.4 255.255.255.255
!
router ospf 1
   router-id 4.4.4.4
   network 10.1.34.4 0.0.0.0 area 0
   network 4.4.4.4 0.0.0.0 area 0
!
```

```
mpls ldp router-id loopback0
mpls label range 400 499
interface fast0/0
  mpls ip
```

4. **实验现象**

完成上述配置后，我们可以检验一下，现在是路由全网是互通的。

R1#show mpls ldp neighbor

```
    Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 2.2.2.2.31044 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 16/16; Downstream
        Up time: 00:05:38
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 10.1.12.2
        Addresses bound to peer LDP Ident:
          2.2.2.2         10.1.12.2       10.1.23.2
```

上面是 R1 上显示的 LDP 邻居，有一个 LDP 邻居，它的 LDP routerID 是 2.2.2.2，label spaceID=0，说明是基于平台的标签空间。

TCP connection: 2.2.2.2.31044 - 1.1.1.1.646，表示这个 LDP 连接是建立在 TCP 的 1.1.1.1 源端口 646，到目的地 2.2.2.2 的 31044 端口。因为 2.2.2.2 地址大，所以它是发起方。

R1#show mpls ldp bindings （查看 R1 的 LIB 表）

```
tib entry: 1.1.1.1/32, rev 2
        local binding:   tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
tib entry: 2.2.2.2/32, rev 6
        local binding:   tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 3.3.3.3/32, rev 13
        local binding:   tag: 103          !! 本地为前缀 3.3.3.3/32 分配的标签
        remote binding: tsr: 2.2.2.2:0, tag: 202   !! 邻居 R2 为前缀 3.3.3.3/32 分配的标签
tib entry: 4.4.4.4/32, rev 14
        local binding:   tag: 104
        remote binding: tsr: 2.2.2.2:0, tag: 203
```

tib entry: 10.1.12.0/24, rev 4

local binding: tag: imp-null

remote binding: tsr: 2.2.2.2:0, tag: imp-null

tib entry: 10.1.23.0/24, rev 8

local binding: tag: 101

remote binding: tsr: 2.2.2.2:0, tag: imp-null

tib entry: 10.1.34.0/24, rev 12

local binding: tag: 102

remote binding: tsr: 2.2.2.2:0, tag: 201

一旦 LDP 激活后，LSR 会为路由表中的前缀在本地产生一个标签，然后和前缀捆绑在一起，将这个标签映射消息发送给所有的 LDP 邻居。当我收到 LDP 邻居发来的（remote binding），针对某些前缀的标签捆绑后，我会将这些标签，以及我本地为特定前缀捆绑的标签（local binding），放置于 LIB 中。

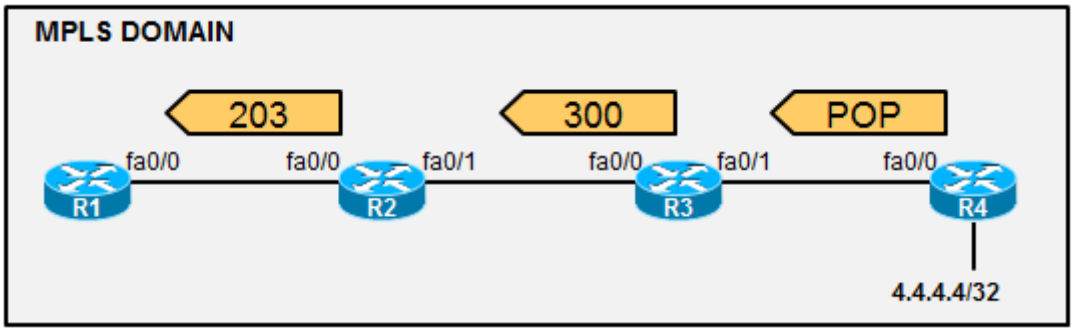
当然，并不是 LIB 中的 remote 标签都会被用上，我们还需结合 FIB 表，来获得有关前缀的下一跳信息。最后形成 LFIB 表：

R1#show mpls forwarding-table　　R1 的 LFIB 表

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.12.2
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2
104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

好，现在我们来分析一下，当 R1 要发送数据去往 R4 的 Loopback 4.4.4.4，数据是如何传送的。

首先分析一下控制层面：



由于大家都通过 OSPF 学习到了 4.4.4.4/32，那么所有的 LSR 都会为 4.4.4.4/32 在本地产生一个标签，然后将这个标签捆绑在前缀上传递给其他 LDP 邻居，如图所示。

好，那么现在当 R1 要去 ping 4.4.4.4 时，R1 得查自己的 FIB，也就是 CEF 表，注意，这是一个 IP 查找：

R1#show ip cef 4.4.4.4

```
4.4.4.4/32, version 12, epoch 0, cached adjacency 10.1.12.2
0 packets, 0 bytes
  tag information set
    local tag: 104
    fast tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
  via 10.1.12.2, FastEthernet0/0, 0 dependencies
    next hop 10.1.12.2, FastEthernet0/0
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
```

CEF 的条目指示，要去往 4.4.4.4，需要给 IP 报文压上一层标签，值为 203，然后将数据包丢给下一跳 10.1.12.2，从 Fa0/0 口扔出去。

接下来 R2 收到这个标签包，R2 从这个数据包的二层以太网帧头的类型字段，知道了这是一个标签包，因此它去查找自己的 LFIB 表：

R2#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	1.1.1.1/32	0	Fa0/0	10.1.12.1
201	Pop tag	10.1.34.0/24	0	Fa1/0	10.1.23.3
202	Pop tag	3.3.3.3/32	0	Fa1/0	10.1.23.3
203	**300**	**4.4.4.4/32**	**0**	**Fa1/0**	**10.1.23.3**

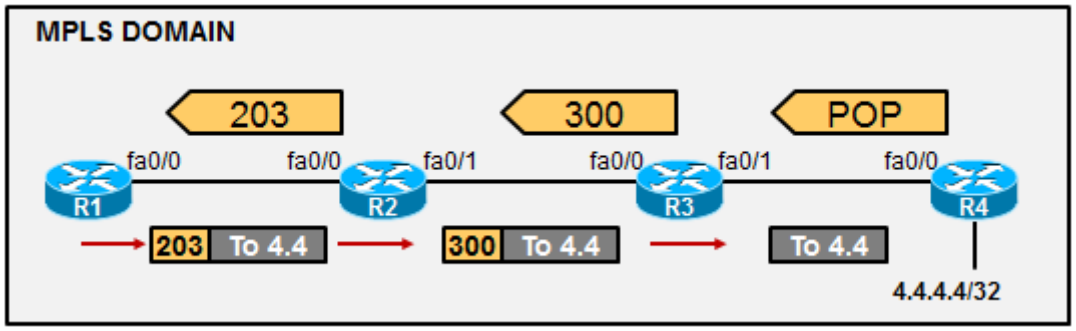
这个入站的标签包，标签值为 203，那么在 R2 的 LFIB 表中指示，203 需要交换成 300，然后丢给下一跳 10.1.23.3 从 Fa1/0 口送出去。于是，R2 将标签替换成 300，然后丢给了 R3。

接下来 R3 收到了这个标签包，同样，查看自己的 LFIB：

R3#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
300	**Pop tag**	**4.4.4.4/32**	**0**	**Fa1/0**	**10.1.34.4**
301	Pop tag	10.1.12.0/24	0	Fa0/0	10.1.23.2
302	200	1.1.1.1/32	0	Fa0/0	10.1.23.2
303	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.23.2

R3 发现，入站标签 300 的标签包，出站标签是个 POP，于是他将顶层标签弹出（实际上就一层），然后直接将弹出后的数据丢给 10.1.34.4 ,注意 ,这时候它无需再次查找 FIB 表 ,因为 LFIB 表中已经有下一跳信息了。最终这个数据被传到了 R4。



我们可以验证一下：

R1#traceroute 4.4.4.4

Type escape sequence to abort.

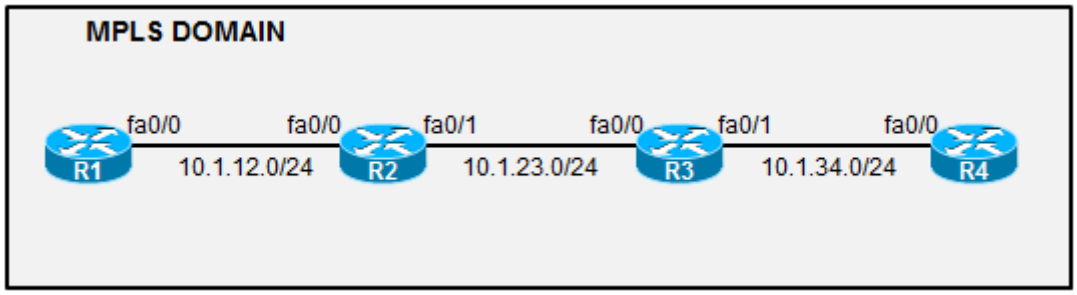
Tracing the route to 4.4.4.4

1 10.1.12.2 **[MPLS: Label 203 Exp 0]** 200 msec 84 msec 136 msec

2 10.1.23.3 **[MPLS: Label 300 Exp 0]** 108 msec 116 msec 64 msec

3 10.1.34.4 52 msec * 120 msec

5. 关于 MPLS 环境下 OSPF 的一点小话题



前面的配置，中所有设备的 Loopback 都是/32 的，因此没出啥问题。现在我们再看看 R1.

R1#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	**Pop tag**	**2.2.2.2/32**	**0**	**Fa0/0**	**10.1.12.2**
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2

104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

重点关注一下 2.2.2.2/32 这个条目，现在的 outgoing label 是 pop，也就是空标签。

那么现在这样的，我们将 R2 的 loopback 口地址改为 2.2.2.2/24，改成/24 的，来看看会有什么现象。

R1#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	**Untagged**	**2.2.2.2/32**	**0**	**Fa0/0**	**10.1.12.2**
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2
104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

变成 untagged 了，之前是 POP，为什么现在是 untagged 呢？分析一下，我们修改了 R2 的 loopback 口，改成 2.2.2.2/24，那么对于 R2 自己，这个直连路由就是 2.2.2.0/24 对吧？ R2 会为这个 2.2.2.0/24 分配标签，由于这是直连，所以 R2 给这条前缀分了个空标签 3。然后将标签映射消息发给其他 LDP 邻居包括 R1 和 R3：

```
⊟ Forwarding Equivalence Classes TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Forwarding Equivalence Classes TLV (0x100)
    TLV Length: 7
  ⊟ FEC Elements
    ⊟ FEC Element 1
        FEC Element Type: Prefix FEC (2)
        FEC Element Address Type: IPv4 (1)
        FEC Element Length: 24
        Prefix: 2.2.2.0
⊟ Generic Label TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Generic Label TLV (0x200)
    TLV Length: 4
    Generic Label: 3
```

那么 R1 和 R3 收到了个标签映射，前缀是 2.2.2.0/24，标签是 3。与此同时，他们也收到了 R2 更新过来的路由，由于 loopback 接口路由被 OSPF 更新默认是采用/32 的方式更新的，因此 R1、R3 上学习到的关于 2.2.2.2 的路由，是/32 位的。那么这就出问题了，我这的路由条目是 2.2.2.2/32，但是你发给我的标签消息里却是 2.2.2.0/24，不匹配啊，于是 R1 就认为，走标签，是到不了 2.2.2.2 了，干脆就给了个 untagged，就像我们看到的 R1 的 LFIB 表。这样一来，当 R1 收到标签包去往 2.2.2.2，R1 会将该标签包的标签栈整个弹出，然后去查找自己的 FIB 表，将报文转发出去。这在本拓扑中，貌似没什么问题，但是，在许多环境下，却会出问题，譬如 MPLS VPN 等等。那么怎么解决呢？很好办，R2 loopback 口来个 ip ospf network point-2-point，或者改掩码都成。

4.4 高级特性

1. **TTL-propagate**

no mpls ip propagate-ttl [forwarded | local]

可以防止由于 TTL=0 返回差错消息而暴露核心传输网的结构

2. **targeted 邻居关系建立**

通常 LDP 会话是建立在直连的 LSR 之间的，但是在某些情况下，可能需要一个远程的、或者基于目的的 LDP 会话，例如 MPLS VPN 中的 TE 隧道。当存在链路翻动的时候，一个基于目的的 LDP 邻居相对于直连的 LDP 对等体来说，可以提高标签收敛的时间。因为如果是直连 LDP 邻居，两台 LSR 之间的链路如果 DOWN 掉了，那么 LDP 会话也就 DOWN 了，然而基于目的的 LDP 会话，另一条可选路径就可以用来承载 LDP TCP 报文，这样的话，就算就算链路 DOWN 掉了，LDP 会话仍然有效，所有的标签都会被保留，在链路重新恢复的时候可以很快将 LIB 中的标签更新到 LFIB 中去。

Router(config)# mpls ldp neighbor [vrf vpn-name] **ip-addr targeted [ldp | tdp]**

建立 targeted 邻居

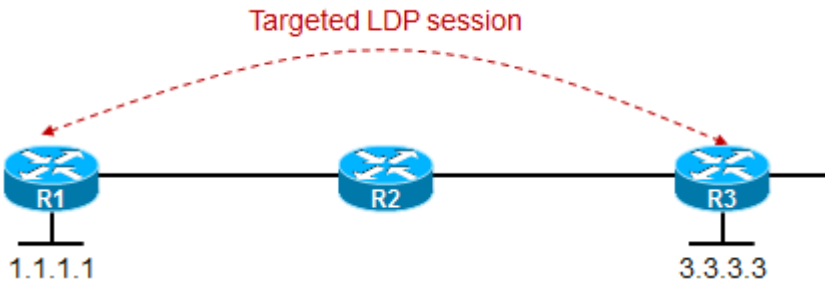
Router(config)# mpls ldp targeted-hello accept from *acl*

配置 targeted-hello accept acl

Router(config)# mpls ldp targeted-hello holdtime x interval y

修改 targeted ldp 会话相关参数

- **配置范例 1：**



R1 的配置如下：

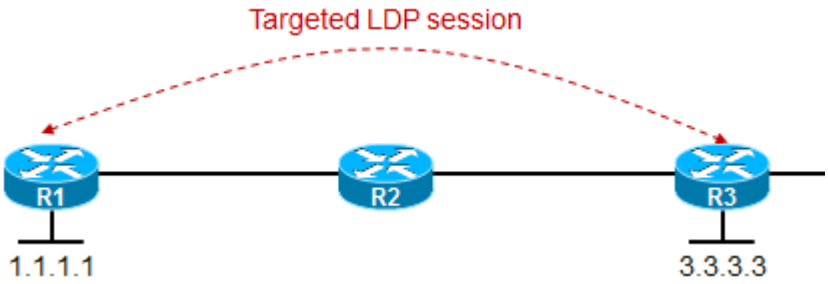
mpls ldp neighbor 3.3.3.3 targeted ldp

R3 的配置如下：

mpls ldp neighbor 1.1.1.1 targeted ldp

这种配置适合有明确的会话端点的情况。

- **配置范例 2：**



R1 的配置如下：

```
mpls ldp neighbor 3.3.3.3 targeted ldp
```

R3 的配置如下：

```
ip access-list standard accept-ldp
 permit 1.1.1.1
mpls ldp discovery targeted-hello accept from accept-ldp
```

R3 是一个被动的会话接受者。

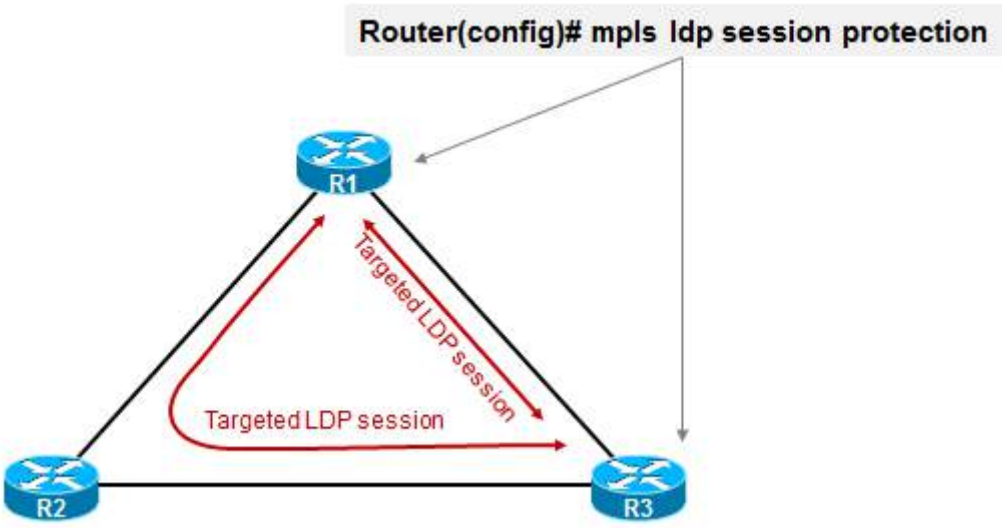
3. LDP 会话保护

链路出现翻动，将导致 LDP 会话重建，并且必须再次交换标签捆绑，要避免重建 LDP 会话，这是非常低效的，我们可以对会话进行保护，在两台直连 LSR 之间的 LDP 会话实施保护之后，将会在这两台 LSR 之间建立基于目的的 LDP 会话，也就是 targeted LDP session，当这两台 LSR 之间的直连链路 DOWN 掉后，只要在这两台 LSR 之间存在可替代的路径，基于目的的 LDP 会话将会得到维持，因此当链路再次恢复后，LSR 就不需要重新建立 LDP 会话了，收敛的效率也就提高了。

Router(config)# mpls ldp session protection for *acl-peer* **duration** *sencondes*

其中 acl-peer 是关联 ACL 用于匹配需要保护的 LDP peer，注意这里 ACL 中匹配的需是邻居的 LDP routerID

Duration 指的是在 LDP 链路的邻接关系 DOWN 掉后，需要得到持续保护（基于 target LDP session）的时间，默认为永久。



为了保护特性正常工作，你需要在两端的 LSR 上都启用该特性。如果有一端不支持该特性，那么可以在一段配置该特性，另一端配置 mpls ldp discovery targeted-hello accept 命令来接收 target-hello。

上图中，R1、R2、R3 运行 OSPF，宣告直连及 Loopback，Loopback 编址为 x.x.x.x/32，x 为设备编号；同时所有的直连接口激活 LDP。

R1#show mpls ldp neighbor

Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
TCP connection: 2.2.2.2.61914 - 1.1.1.1.646
State: Oper; Msgs sent/rcvd: 13/13; Downstream
Up time: 00:04:05
LDP discovery sources:
FastEthernet0/0, Src IP addr: 10.1.12.2
Addresses bound to peer LDP Ident:
10.1.12.2 10.1.23.2 2.2.2.2
Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 1.1.1.1:0
TCP connection: 3.3.3.3.33664 - 1.1.1.1.646
State: Oper; Msgs sent/rcvd: 13/13; Downstream
Up time: 00:03:47
LDP discovery sources:
FastEthernet1/0, Src IP addr: 10.1.13.3
Addresses bound to peer LDP Ident:
10.1.23.3 10.1.13.3 3.3.3.3

接下去我们在 R1 上 shutdown Fa1/0 口，然后在 R1 上看一看：

R1#sh mpls ldp bindings

```
tib entry: 1.1.1.1/32, rev 2
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
tib entry: 2.2.2.2/32, rev 4
        local binding:    tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 3.3.3.3/32, rev 6
        local binding:    tag: 101
        remote binding: tsr: 2.2.2.2:0, tag: 201
tib entry: 10.1.12.0/24, rev 10
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 10.1.13.0/24, rev 8(no route)
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 202
tib entry: 10.1.23.0/24, rev 12
        local binding:    tag: 102
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
```

很明显由于丢失了与 R3 的直连链路，R1-R3 之间的 LDP 邻接挂掉了，自然而言 LIB 表里，之前 R3 传递过来的标签捆绑也就丢失了，现在，如果 R1-R3 之间的链路回复，他们又要重新建 LDP 邻接，重新发送标签捆绑。

现在我们先 no shutdown R1 的 Fa1/0 口，然后在 R1 及 R3 上开启 LDP 会话保护。

R1#show mpls ldp neighbor

```
    Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 2.2.2.2.61914 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 26/22; Downstream
        Up time: 00:09:33
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 10.1.12.2
        Addresses bound to peer LDP Ident:
          10.1.12.2       10.1.23.2       2.2.2.2
    Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 3.3.3.3.41954 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 9/9; Downstream
```

```
        Up time: 00:00:33
        LDP discovery sources:
          FastEthernet1/0, Src IP addr: 10.1.13.3
          Targeted Hello 1.1.1.1 -> 3.3.3.3, active, passive
        Addresses bound to peer LDP Ident:
          10.1.23.3       10.1.13.3       3.3.3.3
```

发现 R1-R3 之间建立起了 targeted session，现在，我们再来 shutdown R1 的 Fa1/0 口

R1#sh mpls ldp bindings 我们仍然在 R1 的 LIB 表里能看到之前 R3 传递过来的标签捆绑：

```
  tib entry: 1.1.1.1/32, rev 2
        local binding:   tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
        remote binding: tsr: 3.3.3.3:0, tag: 300
  tib entry: 2.2.2.2/32, rev 4
        local binding:   tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
        remote binding: tsr: 3.3.3.3:0, tag: 301
(.....省略.....)
```

而且，由于默认的 duration 是无限期，因此这些条目会一直保留着。当然，如果在 R1 上修改配置：

```
access-list 1 permit 3.3.3.3
mpls ldp session protection for 1 duration 30
```

那么 R1 将限定建立 targeted session 的 peer，为 3.3.3.3，这是 R3 的 LDP routerID，另外，LDP 会话保护的时间为 30S，也就是说，30S 后如果 LDP 邻居还没起来，targeted session 将失效。自然的，LIB 里之前存储的被保护的标签信息也就没了。

4. LDP 认证

LDP 会话是一种 TCP 会话，TCP 会话可能遭受 TCP 碎片欺骗的攻击。可以使用 LDP 认证来进行保护。MD5 将添加一个签名—称为 MD5 摘要，到 TCP 分段中。

mlps ldp neigbor *[vrf vpn-name] ip-addr* **password** *[0-7] pswd*

5. 通过 LDP 控制标签通告（条件通告）

router(config)# mpls ldp advertise-labels [for *prefix-access-list* [to *peer-access-list*]]

Parameters:

- **For prefix-access-list** – the IP access list that selects the destinations for which the labels will be

generated

- **To peer-access-list** - the IP access-list that selects the MPLS neighbors that will receiver the labels

注意：要使用上述命令完成条件通告，需先使用“no mpls ldp advertise-labels”，否则默认情况下，LSR 通告所有的标签捆绑。

6. LDP 入站标签捆绑过滤

过滤 LDP 邻居通告过来的入站标签捆绑

router(config)# mpls ldp neighbor *x.x.x.x* labels accept *acl*

neighbor 是邻居的 LDP routerID

accept acl 用来匹配前缀

7. LDP autoconfig

通常的做法是在每个运行 IGP 协议的接口上使用 mpls ip 来激活 LDP 协议，有一种更加简便的方法：

```
router ospf 1
  mpls ldp autoconfig area 0
```

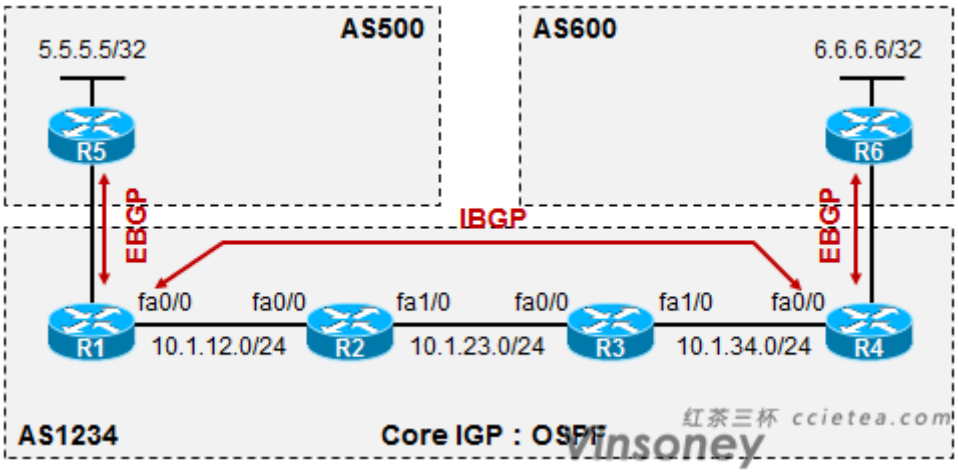
则所有 ospf area0 的接口都将自动激活 LDP

如果某个特定接口不希望激活 LDP，可使用 no mpls ldp igp 来关闭 LDP。

```
Router# show mpls interface detail
Interface FastEthernet0/0:
   ip labeling enable(LDP):
   interface config          <-- 在接口上激活LDP
   LSP tunnel labeling not enabled
   BGP labeling not enabled
   ... ...
Interface FastEthernet0/1:
   IP labeling enabled(LDP):
   IGP config                <-- 使用autoconfig方式激活LDP
   LSP tunnel labeling not enabled
```

5 MPLS 环境下的 BGP 路由传递

1. MPLS 不会为 BGP 路由分配标签，但为 BGP 路由的下一跳分配标签



拓扑环境描述：

- R1、R2、R3、R4 处于 Transit AS 1234。在 AS 内运行的 IGP 协议是 OSPF
- 所有的互联 IP 如图所示
- 所有设备的 Loopback0 口地址为 x.x.x.x/32，x 为设备编号
- R1 与 R4 之间建立 IBGP 邻接关系，IBGP 邻接关系建立在物理接口上。R1 与 R5、R4 与 R6 之间建立 EBGP 邻接关系，也是建立在物理接口上。
- 在这个实验测试中，我们在 OSPF 中宣告 R1-R5 和 R4-R6 的直连网段。
- R5 及 R6 各自在 BGP 进程中宣告自己的 Loopback 路由

实验结果：

由于 R2、R3 没有运行 BGP 协议，并且 Core OSPF 内也没有 5.5.5.0 及 6.6.6.0 的路由，因此最终的结果是 R5 及 R6 虽然能够学习到彼此的路由，但是却无法互访，因为在 R2 及 R3 上出现了路由黑洞。

解决的办法就是用 MPLS，我们将 Core 变成 MPLS 域：

R1 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 100 199
interface fa0/0
  mpls ip
```

R2 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 200 299
```

```
interface fa0/0
  mpls ip
interface fa1/0
  mpls ip
```

R3 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 300 399
interface fa0/0
  mpls ip
interface fa1/0
  mpls ip
```

R4 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 400 499
interface fa0/0
  mpls ip
```

R1、R2、R3、R4 运行 LDP 协议。

要注意，LDP 默认是不会为 BGP 路由分配标签的，但是我们也知道，BGP 路由都有 next-hop，而这个下一跳是 IGP 路由可达的，LDP 则会为这条（下一跳地址所在的）路由分配标签，这一点非常之重要。

这样一来我们 5.5.5.5 和 6.6.6.6 之间就能够互访了，在 R6 上

R6#traceroute 5.5.5.5 soure 6.6.6.6

```
Type escape sequence to abort.
Tracing the route to 5.5.5.5
  1 10.1.46.4 128 msec 152 msec 120 msec
  2 10.1.34.3 [MPLS: Label 303 Exp 0] 856 msec 1012 msec 1072 msec
  3 10.1.23.2 [MPLS: Label 203 Exp 0] 964 msec 1124 msec 1008 msec
  4 10.1.12.1 1060 msec 984 msec 1080 msec
  5 10.1.15.5 1564 msec 1124 msec 1484 msec
```

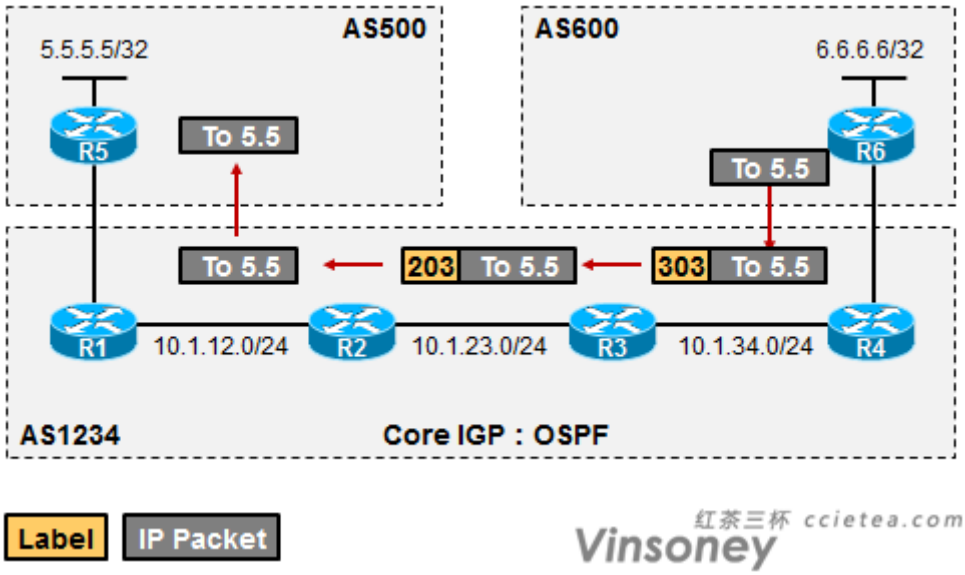
我们发现 ,R6 始发的报文是 IP 的 ,到了 R4 ,R4 查 CEF 表 ,发现目的地 :5.5.5.5 的条目 ,关联了一个 Label：303，于是 R4 将 IP 包压上标签 303，然后丢给下一跳 10.1.34.3 也就是 R3。下面就是 R4 的 CEF 表项：

```
5.5.5.5/32, version 22, epoch 0, cached adjacency 10.1.34.3
0 packets, 0 bytes
  tag information from 10.1.15.0/24, shared
    local tag: 403
```

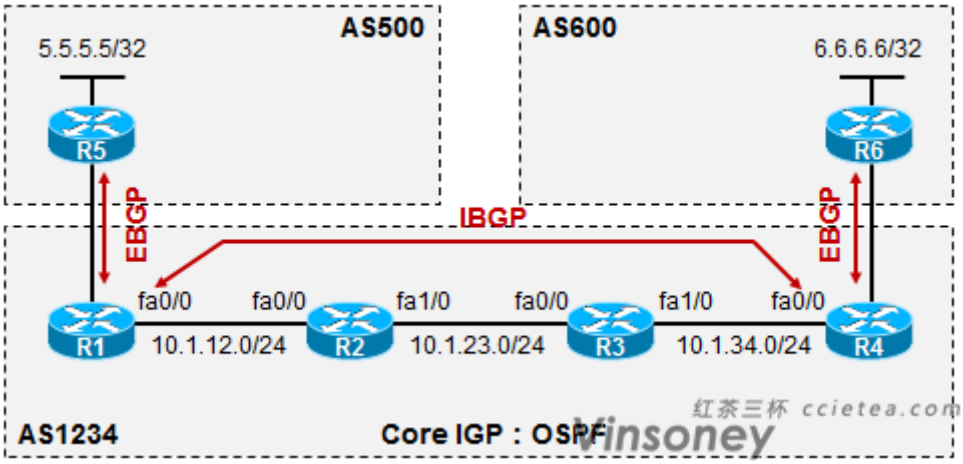
```
    fast tag rewrite with Fa0/0, 10.1.34.3, tags imposed: {303}
via 10.1.15.5, 0 dependencies, recursive
    next hop 10.1.34.3, FastEthernet0/0 via 10.1.15.0/24
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.34.3, tags imposed: {303}
```

注意这里这个 303 很明显，是 R3 分配的，然后将这个分配结果给到了 R4，R4 是在用 R3 分配的标签去压到 IP 包前面。那么这个 303，实际上是为 BGP 路由 5.5.5.5 的下一跳 10.1.15.0 这条路由分配的标签。还是那句话，LDP 不会为 BGP 路由分配标签，但是会为 BGP 路由的下一跳（路由）分配标签。

接下去数据到了 R3，R3 查自己的 LFIB 表后，将 303 标签置换成 203，然后丢给下一跳 R2。R2 收到这个标签包后，查看自己的 LFIB 表，发现 outgoing 动作是一个 PoP，于是将标签弹出，变成最原始的那个 IP 包，然后丢给 R1，最后 R1 将这个 IP 包转发到了 R5。那么这里为什么 R2 这里会 PoP 呢？答案是这条 BGP 路由，前面我们讲过了，其实是用的是它的下一跳 10.1.15.0 的标签，10.1.15.0 是 R1 的直连网段，因此 R1 在为这条路由分配标签时，给了个 PoP，然后将这个结果分发给 R2，这就是原因。

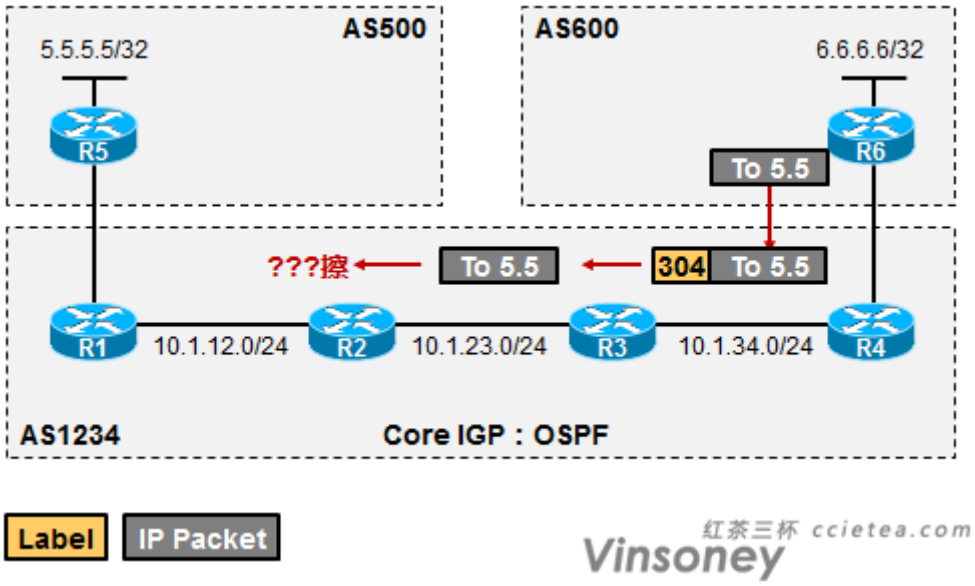


2. （承上）不宣告 R1-R5 及 R4-R6 之间的直连网段进 OSPF



在上面的实验中，我们在 Core OSPF 中宣告了 R1-R5 及 R4-R6 的直连网段，然而我们知道，在实际的场合中，往往不会在 Core IGP 中宣告这条 AS 外的链路，因此接下去我们来看看，如果不在 OSPF 中宣告这两个直连链路，会有什么现象：由于 OSPF 没有了这两个直连网段，那么 5.5.5.5 及 6.6.6.6 学习进来就不优了，需在 R1 上对 R4 去使用 next-hop-self，R4 对 R1 也同理。

那么这样一来，R4 上去往 5.5.5.5 下一跳就是 10.1.12.1 了，那么当 6.6.6.6 访问 5.5.5.5 的时候，IP 包到达 R4，R4 查 CEF 并且为 IP 包压上标签，关键在这个标签上，R4 会使用 10.1.12.0 这条路由的标签（12.0 是去往 5.5.5.5 的下一跳），而这个标签是 R3 分配的（我给出口数据打上的标签永远是 LDP 邻居给我的标签）。那么这个标签包被传递到了 R3，R3 会怎么处理呢？实验现象表明，R3 会将标签 PoP，变成 IP 包然后丢给 R2。可是为什么？原因是 R2 为路由 10.1.12.0 分配的标签就是 PoP，因为 12.0 是 R2 的直连网段，这里有次末跳弹出机制。那么这个 IP 包到了 R2 就歇菜了，因为 R2 根本没有 6.6.6.6 和 5.5.5.5 的路由啊。怎么办？



解决的办法：R1 及 R4 使用 loopback 接口建立 IBGP 邻居关系

前面的实验之所以在 R1 和 R4 上使用物理接口建立 IBGP 邻接 ,是为了帮助大家理解一下 MPLS 环境下 BGP 的路由问题，好了，现在我们在 R1 和 R4 上，使用 loopback 接口来建立 IBGP 邻接关系，问题就解决了。 这时候 R4 上，去往 5.5.5.5 的下一跳就是 R1 的 Loopback 口地址，所以 R4 在给去往 5.5.5.5 的 IP 包压标签的时候，用的就是 1.1.1.1/32 路由的标签，R3、R2 也一样，那么标签包就能通过 1.1.1.1/32 打通的 LSP 一路传到 R2，并在 R2 这弹出标签变成 IP 包再转给 R1，R1 再将 IP 包转发给 R5，搞定。

因此，路由器不会对 BGP 路由直接分配标签，而是为 BGP 路由通过递归后的下一跳路由分配标签，这样做是很有好处的。BGP 中的路由条目是相当多的，如此一来我们通过 MPLS，可大大简化路由器的性能损耗，BGP 的 Transit AS 的路由黑洞问题也得到了很好的解决。

6 参考书籍

- MPLS 技术架构
- MPLS VPN 体系结构 CCSP 版
- MPLS VPN 体系结构 卷二
- MPLS 设计与实现
- MPLS VPN configuration

MPLS VPN 技术笔记

文档版本： **2.0** 更新时间： 2013-07-23

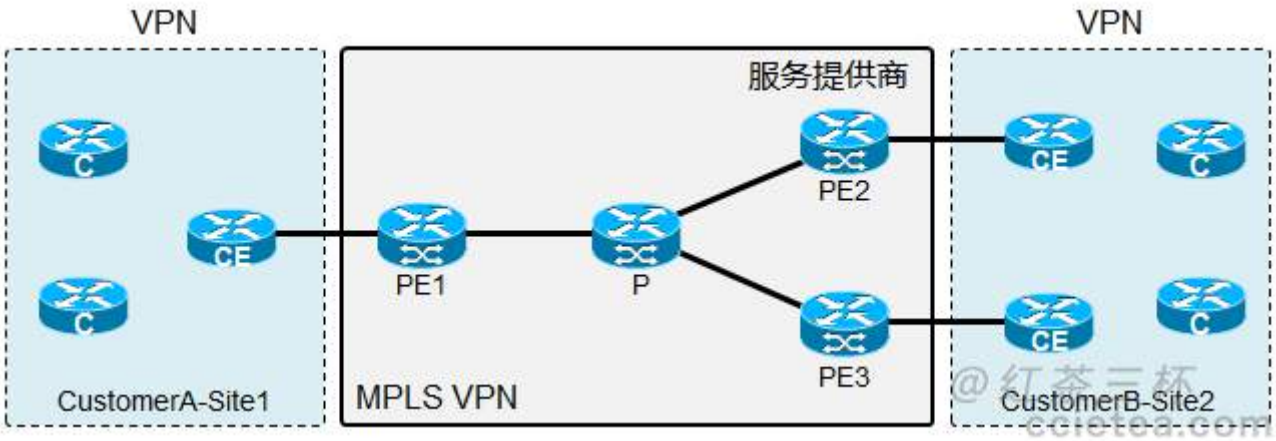
文档作者： 红茶三杯（http://weibo.com/vinsoney）

文档备注： 文档将不定期更新，请关注 http://ccietea.com 以便获取最新版本的文档

1 目录

2 MPLS VPN 基础

2.1 MPLS VPN 架构



1. **PE（Provider edge）运营商边界设备**

   运营商边界设备，与客户边界设备相连接。支持 MPLS。在 PE 上，使用 VRF 对 VPN 客户进行隔离。PE 通过与客户设备 CE 运行路由协议以便获取客户路由，并将路由生成 VPNv4 前缀放入 MPLS VPN Backbone 传递到对端 PE。

2. **P（Provider）运营商设备**

   运营商设备 ,不直接与客户设备相连接。支持 MPLS ,P 设备往往并不知道 VPN 客户网络、以及客户的路由。它只负责在 Backbone 内运载标签数据。

3. **CE（customer edge ）客户边界设备**

   客户网络中，与 PE 直连的设备，主要的功能是将 VPN 客户的路由通告给 PE，以及从 PE 学习同一个 VPN 下其他站点的路由。

2.2 VRF

MPLS VPN 一个非常吸引人的地方 ,就是可以让不同客户的路由及数据穿越运营商的 MPLS VPN Backbone，而且这些路由和数据又是相互隔离和独立的，即使不同的客户拥有相同的 IPv4 地址空间也不要紧。

那么作为这些客户路由进入 MPLS Backbone 入口的设备 -- PE，就显得非常重要了。在 PE 上有个非常重要

的概念—VRF（严格的说，VRF 的作用现在已经扩展了，我们这里重点讨论在 MPLS VPN PE 中的运用）。

VRF：Virtual Routing and Forwarding，翻译成虚拟路由及转发，它是一种 VPN 路由和转发实例。一台 PE 路由器，由于可能同时连接了多个 VPN 用户，这些用户（的路由）彼此之间需要相互隔离，那么这时候就用到了 VRF，PE 路由器上每一个 VPN 都有一个 VRF。PE 路由器除了维护全局 IP 路由表之外，还为每个 VRF 维护一张独立的 IP 路由表，这张路由表称为 **VRF 路由表**。要注意的是全局 IP 路由表，以及每个 VRF 的路由表都是相互独立或者说相互隔离的。

因为每一个 VPN 都有一张独立的 VRF 路由表，所以 PE 路由器上每一个 VPN 也会有一张独立的 CEF 表来转发这些报文，这就是 **VRF CEF 表**。

一旦在 PE 路由器上创建了一个 VRF，我们就可以将特定的接口（物理或逻辑的）放入这个 VRF，那么这个接口将不再属于全局 IP 路由表或其他任何 VRF，只为该 VRF 服务。关于 VRF 的配置在下文给大家介绍。

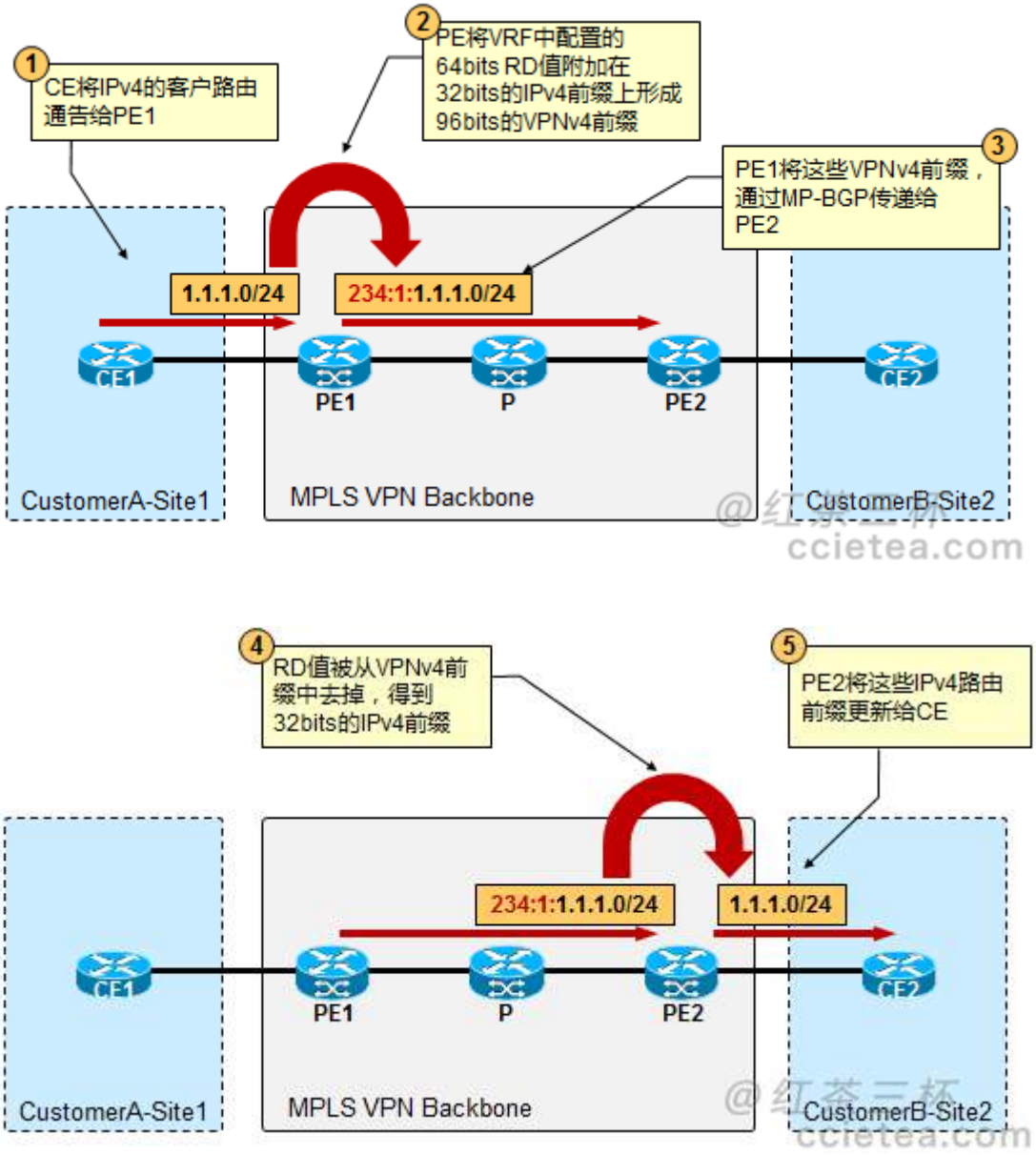
2.3 RD

由于 VPN 前缀是通过 MP-BGP 在 MPLS VPN 网络中扩散，那么可能，同一时间，运营商的 MPLS VPN 网络承载着多个客户的 VPN 前缀，甚至有可能是使用相同的 IPv4 地址空间，那么怎么做到客户之间地址空间的唯一性呢？就需要用到 RD 值了。RD 值在 VRF 中进行配置。

RD（route distinguisher）64bits，用于在 MP-BGP 运载 VRF 前缀时，确保这些前缀的唯一性。但是 RD 并不会说明该前缀属于哪一个 VRF（需要搭配 RT），RD 的功能并不是 VPN 标示符，因为在一些复杂的 VPN 环境中，可能一个 VPN 存在多个 RD。RD 的最重要的两个功能：

1. **与 32bits 的 Ipv4 前缀一起构成 96bits 的 VPNv4 前缀；**
2. **如果不同的 VPN 客户，存在相同的 IPv4 地址空间，那么可以通过设置不同的 RD 值从而保证前缀的唯一性。**

这个 64 比特的值可以有两种表现形式：AS:nn 或者 IP-address:nn。其中 nn 代表编号。最常用的格式是 AS:nn，其中 AS 代表 AS 号。通常 AS 是 IANA 分配给服务提供商的 AS 号，nn 是服务提供商分配给 VRF 的唯一号码。产生的 VPNv4 前缀通过 MP-BGP 在 PE 路由器之间被传递，见下图。



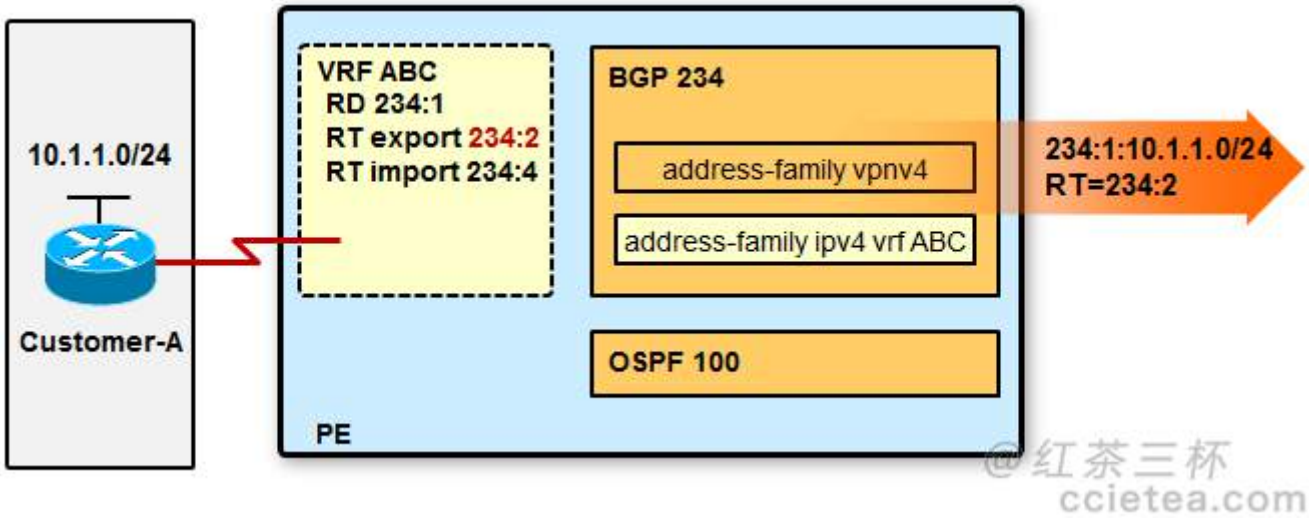
2.4 RT

Route Targets 这玩意就是用来区分 VPN customer 的。是 BGP community 的扩展属性，在 VRF 中进行配置。它跟在 VPNv4 前缀后面被一起传递。一条路由可以附加多个 RT 值。

- **Export RTs**

  通过在 vrf 中定义 export RT 值，将使得输出的 VPNv4 路由携带上该 RT 值一起传递 – 以 BGP 扩展

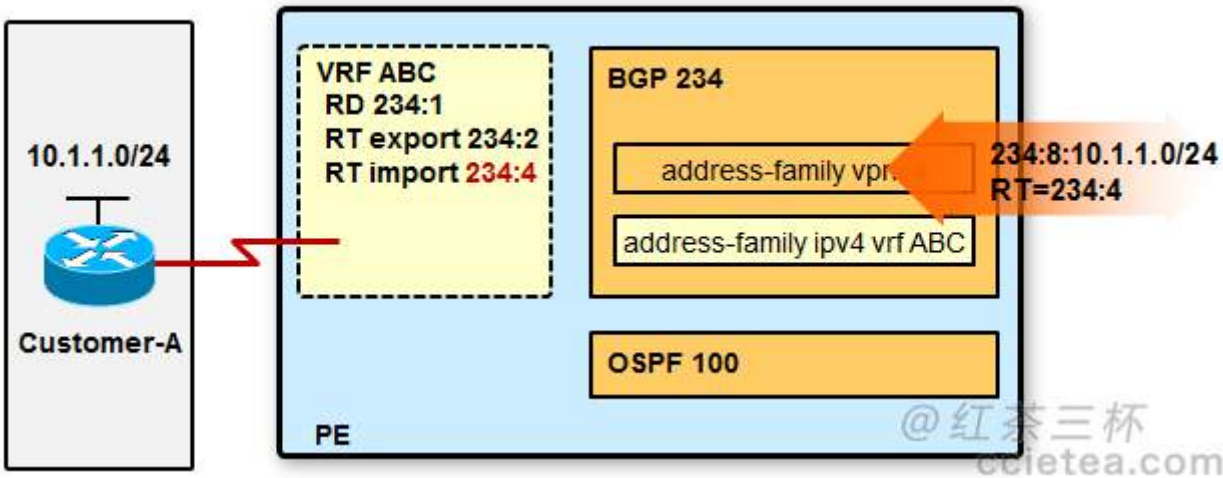
community 的方式。注意这些 VPNv4 路由，是由 VPN 客户的 IPv4 路由导入 VRF 后，加上 VRF 中配置的 RD 值所形成的。



- **Import RTs**

  PE 会从其他 MP-BGP 对等体的 PE 那收到 VPNv4 的前缀，这些前缀都是携带 RT 值的。默认情况下，PE 是不会将这些 VPNv4 路由以 IPv4 的形式装载到 VRF 路由表里，除非在本地的 VRF 中，配置 import RTs，那么如果 import RTs 与收到的 VPNv4 前缀中的 RT 匹配的话，这些 VPNv4 前缀才会被以 IPv4 的形式装载到相应的 VRF 路由表里，相当于在这里 RT 起到一个前缀过滤或者识别的作用，这个功能在许多场景中非常有用。

  VPNv4 路由可能携带不止一个 RT 值，只要有一个匹配 import RT 即可导入到 VRF 路由表。



2.5 VRF 配置

Router(config)# ip vrf *name*

- 创建 VRF，VRF 名字本地有效，且大小写敏感
- VRF 在创建后，需设定 RD 值，否则不可用

Router(config-vrf)# rd *route-distinguisher*

- 进入 VRF 模式后，分配 rd 值给该 VRF
- 可使用 ASN:xx 或 A.B.C.D:xx 格式输入 rd，建议采用 ASN:xx 的形式
- PE 路由器上每个 VRF 的 RD 必须是唯一的

Router(config-vrf)# route-target export *RT*

- 设置当路由被从 VRF 导出到 MP-BGP 时携带的 RT 值
- 可以输入多个 RT，也就是说，一条 VPNv4 路由运行携带多个 RT 值
- RT 值实际上是通过 BGP 的扩展 community 属性来携带

Router(config-vrf)# route-target import *RT*

- 通过设置 import RT 从而将匹配的路由放进 VRF，注意，只有被匹配的路由，才会被导入到 VRF 中
- 可以输入多个 RT

Router(config)# interface e 0/0
Router(config-if)# ip vrf forwarding *vrf-name*
Router(config-if)# ip address 10.1.12.2 255.255.255.0

- 将接口分配到特定的 VRF，一旦配置 ip vrf forwarding xx 命令后，该接口的 IP 地址将被清除，因此需再手工配置 IP 地址。

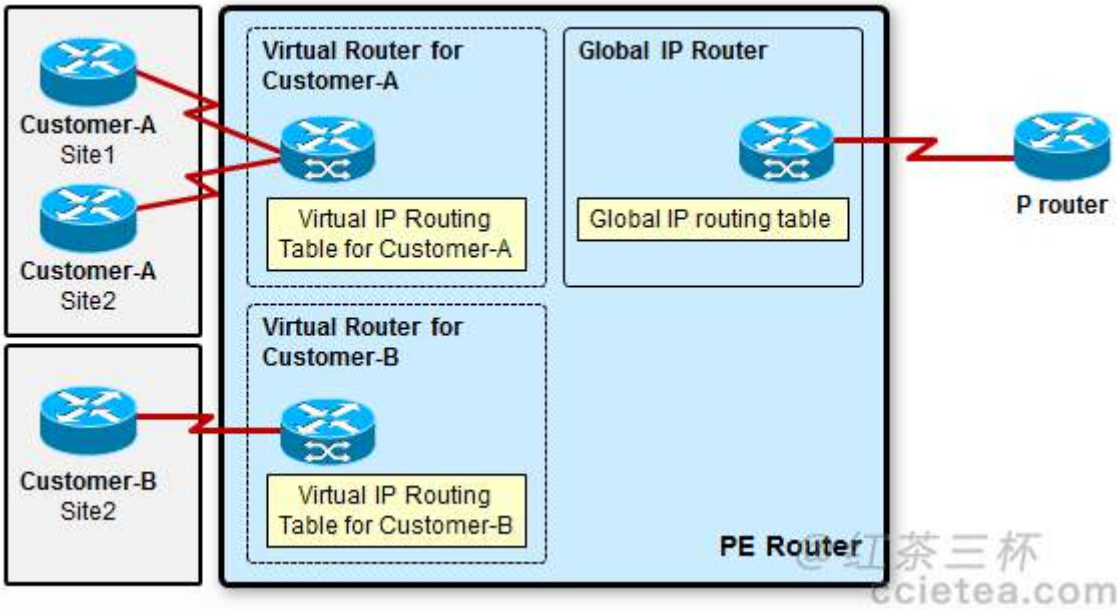
Show ip vrf
Show ip vrf detail
Show ip vrf detail interface
Show ip vrf interface

Show ip protocol vrf

```
Show ip route vrf
```

```
Show ip cef vrf x
```

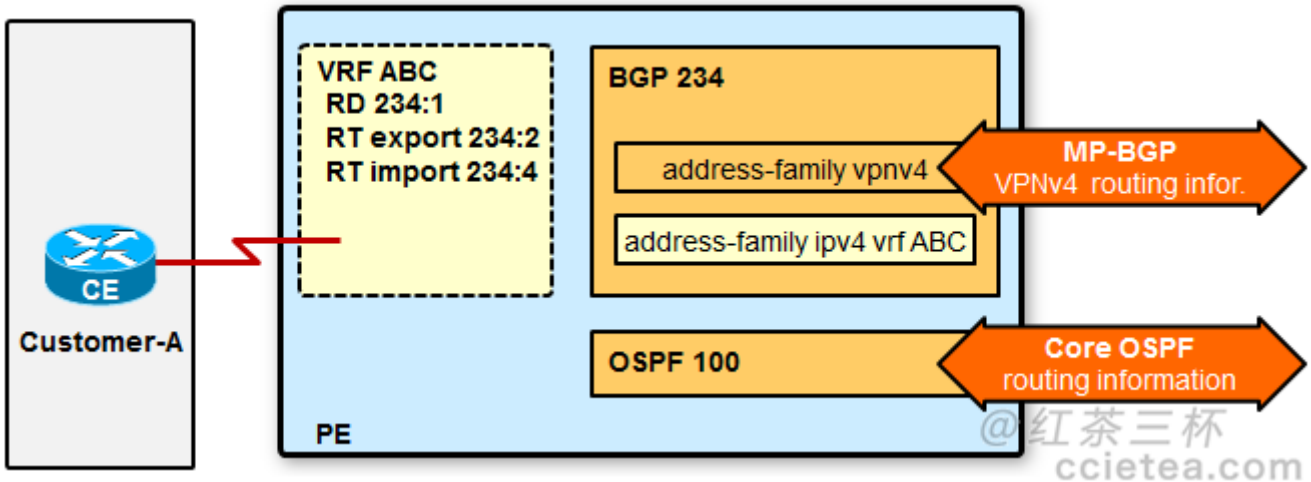
2.6 PE 设备逻辑



PE 设备是 MPLS VPN 部署非常关键的一个环节。上面这张图就是一个典型的 PE 路由器的逻辑分解图。蓝色的这个框框代表的就是一台 PE 路由器 ,内部的两个虚线框我们可以理解为为 VPN 所创建的两个“虚拟路由器”。我们看到这个 PE 设备创建了两个 VRF，VRF-A 对应的是客户 A，VRF-B 对应的是客户 B。

一旦创建了两个 VRF，我们就可以将特定的接口放入特定的 VRF，那么这些接口将只为所属的 VRF 服务。上图中的 PE 路由器实际上就有了三张路由表，分别是两张 VRF 路由表，以及一张全局 IP 路由表。

一个 PE 路由器可以连接不同的 VPN 客户，使用类似虚拟路由器（VRF 实例）的概念，来进行逻辑上的进行区分，看到上图中的 virtual router for A 和 virtual router for B 了么。这些客户甚至有可能使用相同的地址空间，我们在一台 PE 上，使用 VRF 路由表，将客户及客户的路由进行逻辑上的隔离。这里 virtual 路由表是相对于我们的全局 IP 路由表的概念。从 global 接口上学习到的路由，放入全局 IP 路由表，从 VRF 接口上学习到的路由，放入相应的 VRF 路由表。不同的 virtual 路由表完全隔离。

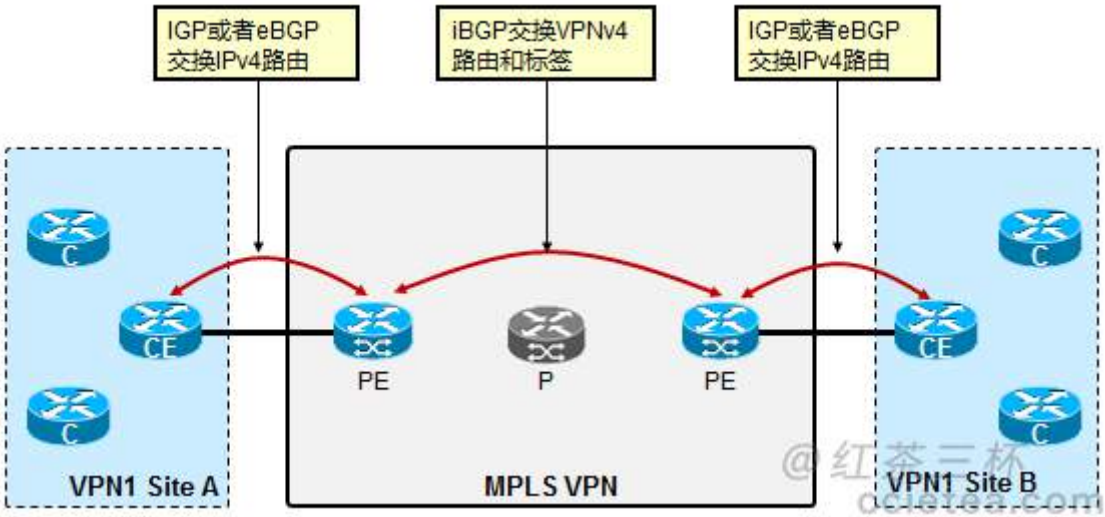


我们，现在我们要从宏观进入到微观层面来理解了，这就是本人常说的："从宏观到微观，从抽象到具象"。再来看仔细一点，上面这个 PE，我们创建了 VRF ABC，那么同时一并出现的还有 VRF ABC 的路由表以及 VRF ABC 的 CEF 表这个是大家在头脑里都要一并出现的。我们将 PE 上与 CE 直连的接口放入 VRF ABC。接下去来分解一下，看看 PE 这里头有啥玩意儿：

- PE 上，会运行一个 **Core 的 IGP 协议**，这里用的是 OSPF 100，这个 OSPF 进程是为全局 IP 路由表贡献路由的。OSPF 100 与运营商骨干网内的其他设备形成 OSPF 邻居关系并且交互骨干网 Core 内的路由，交互这些路由的目的是，可以为后面的 MP-BGP 的建立而服务的，因为 MP-BGP 往往是通过 PE 之间的 Loopback 来建，MP-BGP 邻居关系建立需要这些 IGP 路由。另外后续的 LDP 也依赖这个 Core 的 IGP 协议。通过 OSPF 进程 100 学习到的路由，放入该 PE 路由器的全局 IP 路由表。
- 然后 PE 创建了 **VRF**，名字是 ABC，这就相当于出现了一台虚拟路由器。创建 VRF 的时候需要同时定义 RD 和 RT import 和 RT export 值。RD 值用于和 IPv4 前缀组装形成 VPNv4 路由以便通过 MP-BGP 传播。RT 值用于识别 VRF 或者说客户。
- 接着 PE 上跑一个 **PE-CE 的路由协议**，例如静态、RIP、EIGRP、BGP 等等，目的是为了从 VPN 客户那，也就是 CE 设备那学习到客户的路由。注意由于连接 CE 的接口被放入了 VRF ABC，因此通过这个接口学习到的路由，被放入了 VRF ABC 的路由表。放入了 VRF ABC 路由表之后呢？接下来就要将路由引入到 MP-BGP（的 address-family ipv4 vrf ABC）中，如果 PE-CE 之间运行的是路由协议是非 BGP，那么就需要做路由重发布，将客户路由重发布进 address-family ipv4 vrf ABC 这个地址族下面，而如果 PE-CE 之间运行的已经是 BGP 了，那么路由当然就直接进入 MP-BGP 了。
- PE 上运行一个 **MP-BGP**，MP-BGP 至少有两个 address-family 也就是地址族，一个是 address-family vpnv4，用于和对端的 PE 交互 VPNv4 前缀。另一个地址族是 address-family ipv4 vrf ABC，注意这个地址族是和 vrf ABC 关联的，用于获取 VPN 客户（这个图中就是 customer-A）的路由。
- 现在 MP-BGP 已经有了 VPN 客户的路由，现在要将这些 IPv4 的路由前缀，变成 **VPNv4 的路由前缀**，通过已经建立起来的 VPNv4 的邻接关系传递给对端 PE。由于这些路由是属于 VRF ABC 的，而 VRF 是定义了 RD、RT 值的，那么这些值在这里就派上用场了。32bits 的 IPv4 路由前缀，搭配上 64bits 的 RD 值，就形成

了 96bits 的 VPNv4 的前缀。另外，RT export 值跟随着这个 VPNV4 前缀，被 MP-BGP 更新给了对端 PE。

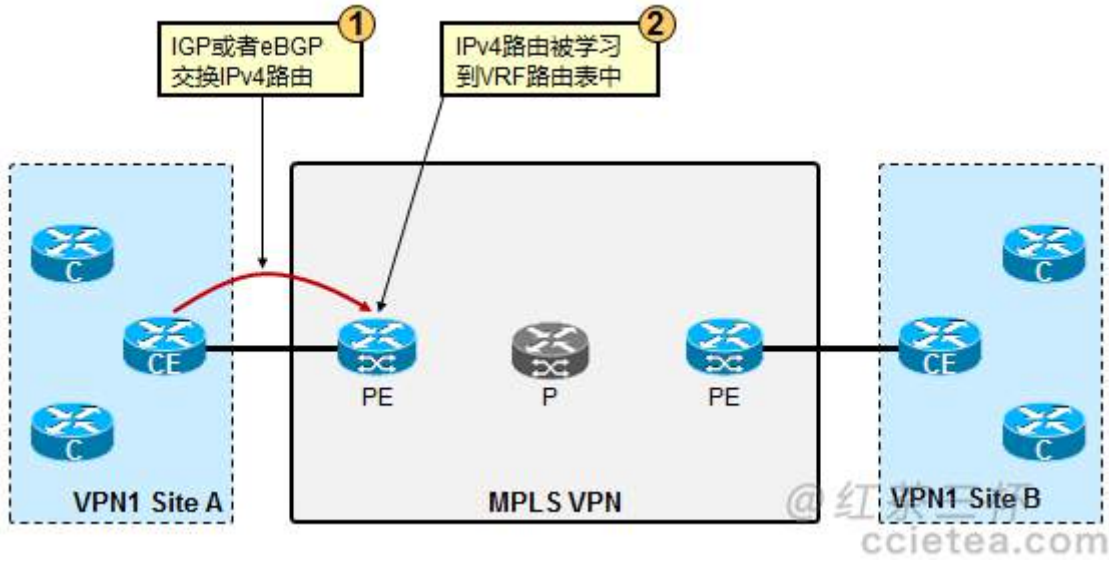
2.7 MPLS VPN 路由模型



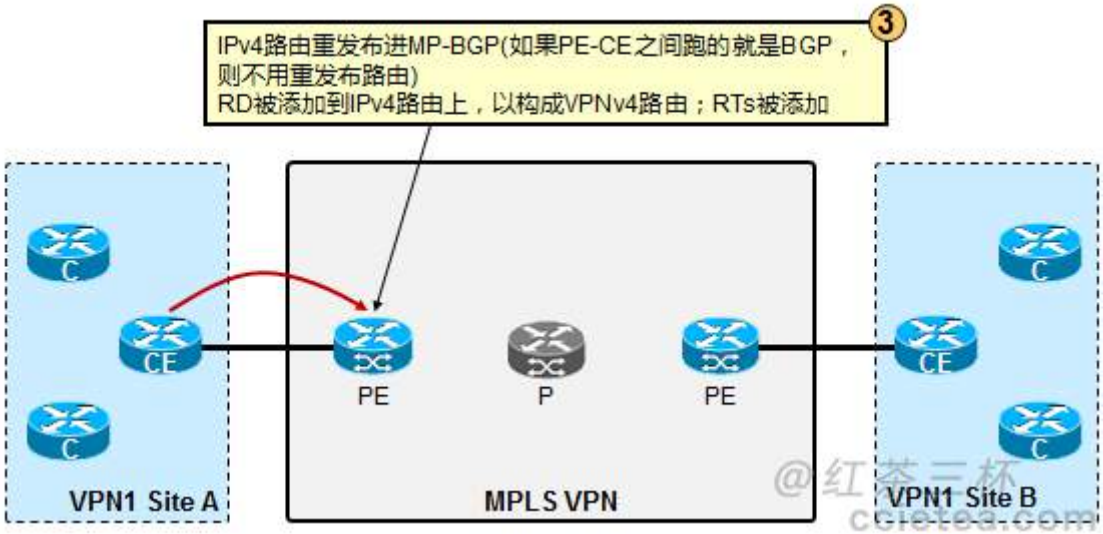
VRF 可以在 PE 路由器上隔离用户路由 ,PE 路由器通过 PE-CE 的路由协议(可以是 IGP ,也可以是 eBGP) , 从 VRF 接口学习到客户的路由后，路由被放入 VRF 路由表。随后被注入到 MP-BGP，这些路由将以 VPNv4 路由前缀的形式通过建立好的 MP-BGP 连接最终被传递到其他 PE 设备上。

但是这些成千上万的客户路由如何在运营商的网络中传递呢？BGP 无疑是最理想的候选人。客户 VPN 路由从 IPv4 前缀，加上我们在 VRF 中配置的 RD 后成为 96bits 的 VPNv4 前缀，从而使得每一个客户的 VPNv4 路由具有唯一性，可以很安全的在运营商网络中传输。而到目前为止我们接触的 BGP，都只能够承载 32bitsIPv4 的路由前缀，这里要承载 96bits 的 VPNv4 前缀，就需要多 BGP 做一些协议上的扩展，我们把这种扩展后的、能够支持多协议的 BGP 成为 Multi-prototol BGP，简称 MP-BGP。通过构建起来的 MP-BGP 连接，VPNv4 的前缀能够被实现传播。

接下去我们来看看详细的路由传播过程：



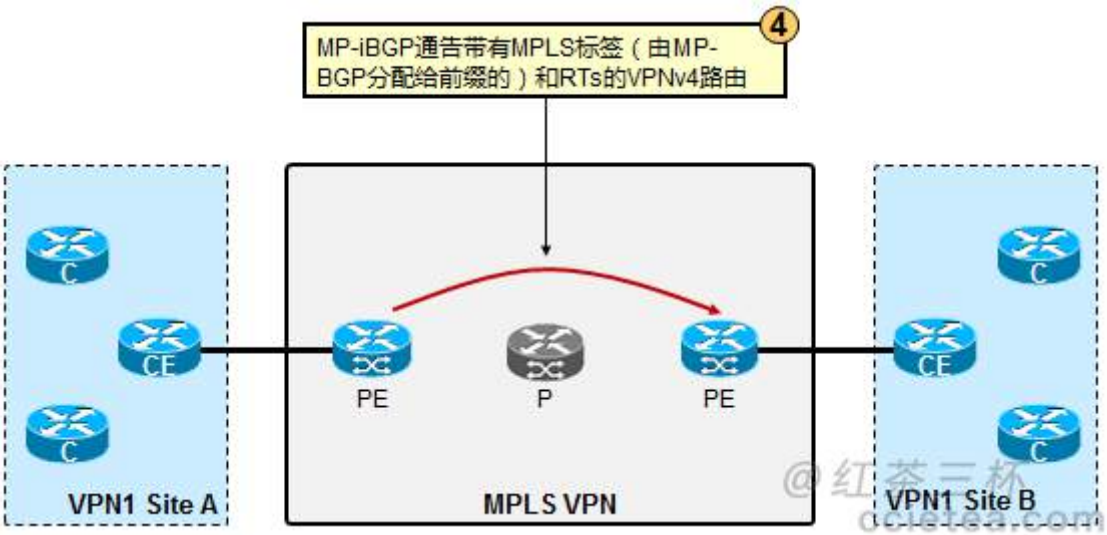
- PE-CE 的动态路由协议有多种选择，如 Static、RIP、EIGRP、OSPF、IS-IS、BGP 等等。PE-CE 之间跑路由协议的目的是为了让 PE 学习到客户的 VPN 路由。由于 PE 上连接特定 CE 的接口是属于特定 VRF 的，因此从这个接口上学习到的客户路由都会进入该 VRF 的路由表。



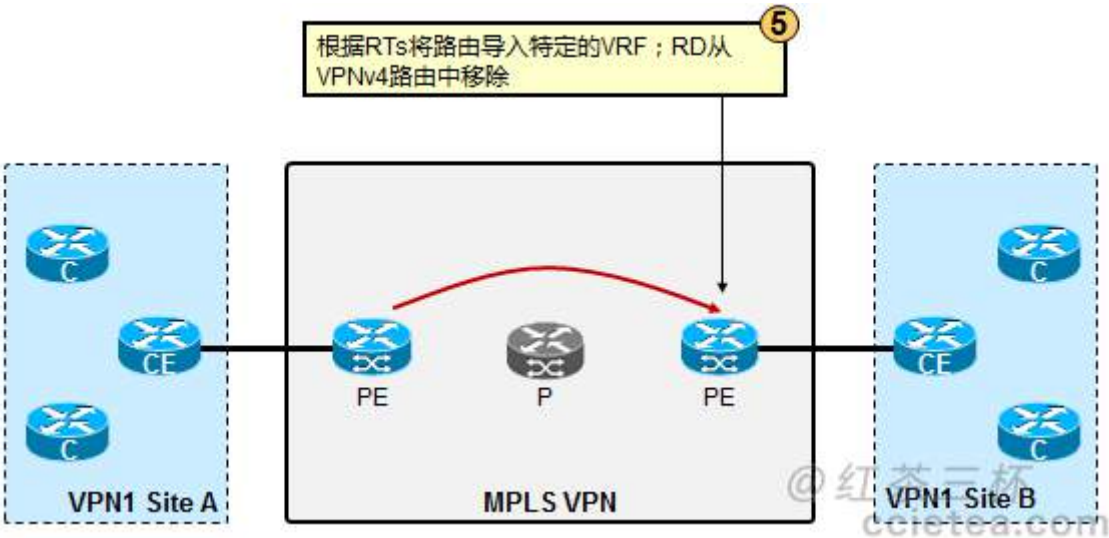
- PE 之间运行的是 MP-BGP 多协议的 BGP，经过扩展的 BGP 协议能够承载 VPNv4 路由。

  在经过前面的步骤，PE 的 VRF 路由表里已经学习到了客户的路由，接下去要让 MP-BGP 知晓这些路由，如果 PE-CE 之间运行的是非 BGP 协议，那么当然，需要将 VRF 路由表中的客户路由重发布到 BGP 中，如果 PE-CE 之间运行的已经是 BGP 了，自然不用再做重发布了。

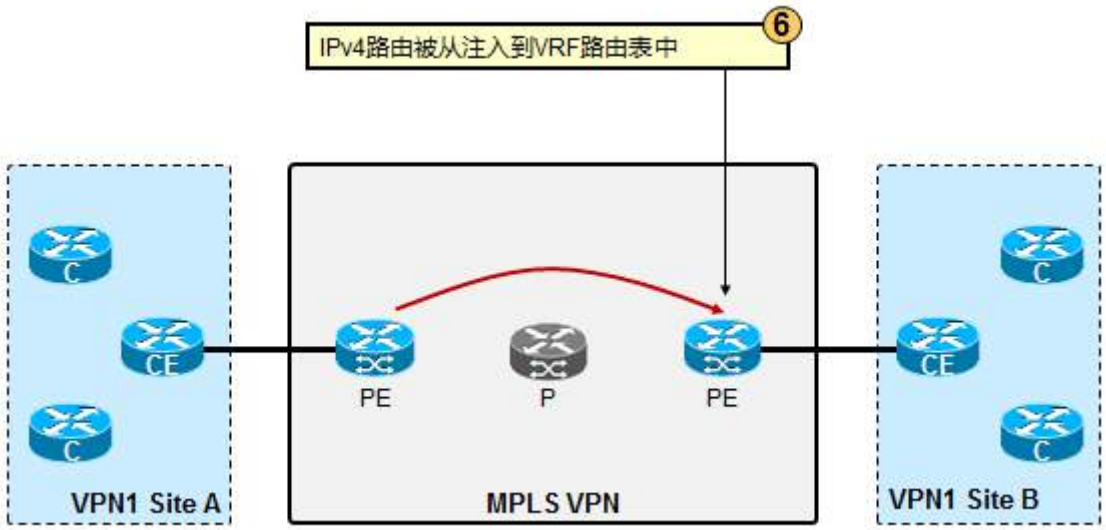
  客户的 IPv4 路由被注入 MP-BGP 后，RD 被添加到了这些 IPv4 路由前缀前面，就构成了 VPNv4 路由前缀，同时 RTs 也被附加到 VPNv4 路由前缀。



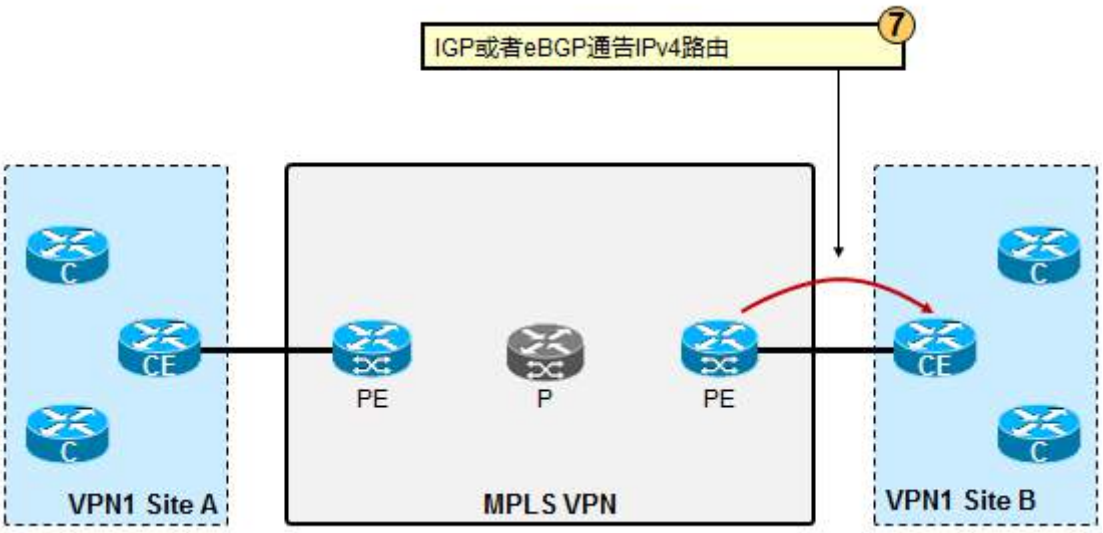
- 由于 PE 之间已经建立起了 MP-BGP 的邻接，因此 PE 将 VPNv4 路由前缀（连同这个前缀关联的 MPLS 标签、RTs 等属性）一并通告给另一端的 PE。



- 对端的PE也配置了VRF ,VRF中定义了import RTs。它根据RTs将收到的VPNv4路由前缀导入特定的VRF，RD 被从 VPNv4 路由中移除。



- 经过上面这部，IPv4 路由就被注入到了 VRF 路由表中。当然，这些路由现在是 BGP 的路由条目。



- 同样的，PE-CE 之间跑的路由协议帮助我们将 PE 上的 IPv4 路由最终更新给 CE。

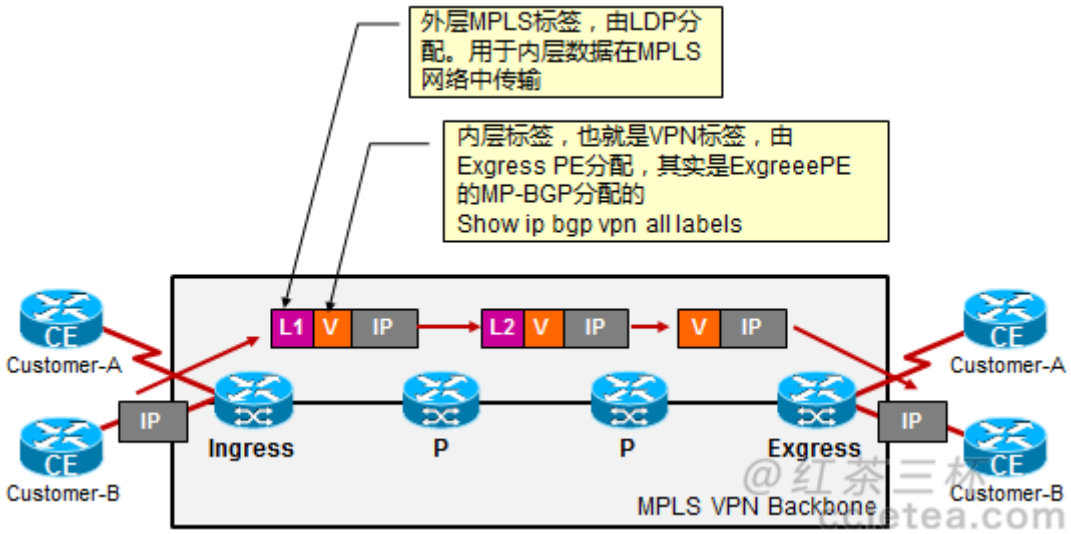
2.8 MPLS VPN 中的报文转发

报文无法以纯 IP 的形式在场点之间进行转发，P 路由器不能转发这些报文，因为 P 压根没有 VRF 路由。MPLS 可以通过对报文粘贴标签解决这个问题，让所有的 P 和 PE 运行 LDP，有了 LDP，这些客户的数据在穿越 MPLS

VPN 网络的时候，就“包裹在”一层标签里头，P 路由器只需要对这层标签进行交换和转发即可，P 路由器永远不会执行目的 IP 的查找工作，这是在入站 PE 路由器和出站 PE 路由器之间报文的交换方式。这层标签称为 **IGP 标签**（我们也常称为 LDP 标签，相对于后面的 VPN 标签），因为该标签是粘连在 P 和 PE 路由器的全局路由表的 IPv4 前缀之上的（我们在 P、PE 上运行的基于全局路由表的 IGP 协议，如 OSPF，就是为了让 MPLS VPN Backbone 内的 IGP 路由都可达，这样 Backbone 内 LDP 的标签分发才能正常工作）。

IGP 标签解决了客户数据在 MPLS Backbone 内的传输问题，现在假设数据传到了出站 PE 路由器，它如何知道报文属于哪一个 VRF 呢（或者说，报文在出站 PE 上应该转发给哪一个下一跳的 CE 路由器）？这个信息不在 IP 头里，也不能从 IGP 标签中获得。那么我们就需要加入另一层标签，来说明报文所属的 VRF。

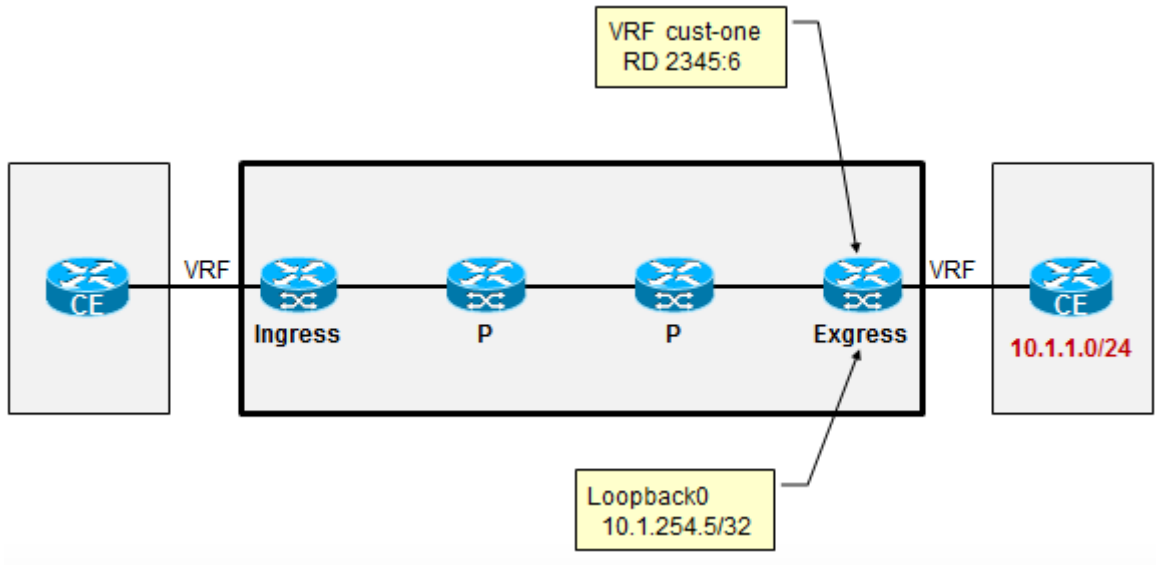
因此报文会有两层标签，一个 IGP 标签在顶部，一个 VPN 标签在底部。IGP 标签是通过 LDP 或者用于 TE 的 RSVP 在所有 P 和 PE 路由器上一跳一跳分发的。VPN 标签是在 PE 之间用 MP-BGP 进行通告的。



再详细一点看看这个问题：

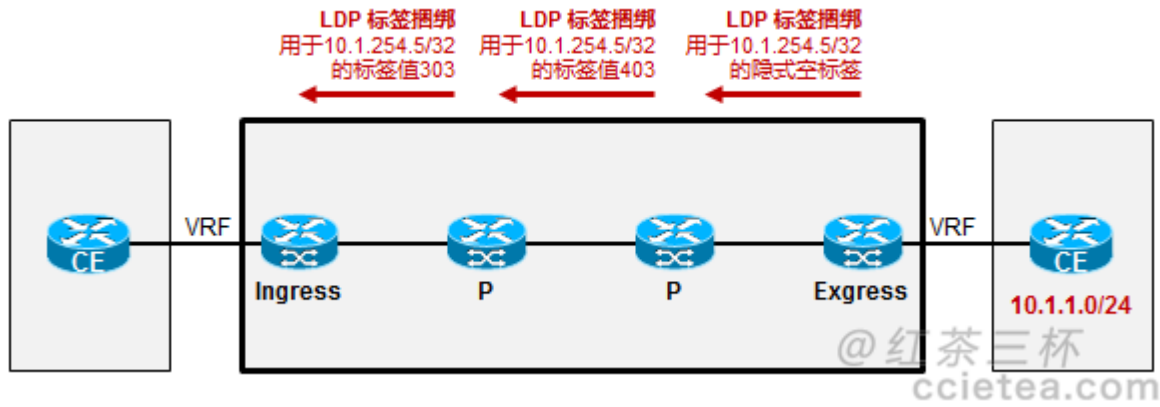
- 客户的数据要想在 MPLS VPN Backbone 里传输，是需要借助标签的，有了标签，Backbone 内的设备在转发数据的时候只要去针对标签进行查找和操作即可，而不用去理会里头的 IP 目的地址。这层标签是由 LDP 分配并分发的。如上图中，紫色的标签 L1 及 L2。这个标签最终会将数据引导到 EpressPE。但是数据到了 Exgress PE，它怎么知道这个数据应该丢给哪一个下一跳的 CE 设备呢？
- 因此，Ingress PE 会给原始 IP 包先压一层 VPN 标签（上图中橙色的标签），再压一层 IGP 标签作为外层，VPN 标签由 egressPE 的 MP-BGP 分配并通过 MP-BGP 告知给 Ingress，所以其实在 Ingress PE 上，压入的 VPN 标签用的是 Exgress PE 所告知的、为目的地路由前缀所分配的 VPN 标签。
- 上图中 V 的标签就是 VPN 标签，由 egressPE 分配（因为我去往对端的路由，是由 ExgressPE 传递过来的），
- 带 V 的标签的数据到了 egress PE 的动作应该是 untag 了（因为没人再传给我标签啊，到我这就结束了），然后从某个接口扔出去（V 标签是由其自己分发的，所有他自己肯定知道里头的数据应该交给谁）

下面是一个例子：



- Ingress PE 及 Exgress PE 之间建立基于 loopback 的 MP-iBGP 邻居关系，Exgress PE 的 Loopback 地址为 10.1.254.5/32。
- 右侧的 Site，CE 有一条路由 10.1.1.0/32，这条路由被 Exgress PE 传递给了对端 Site

1. LDP 分配外层标签



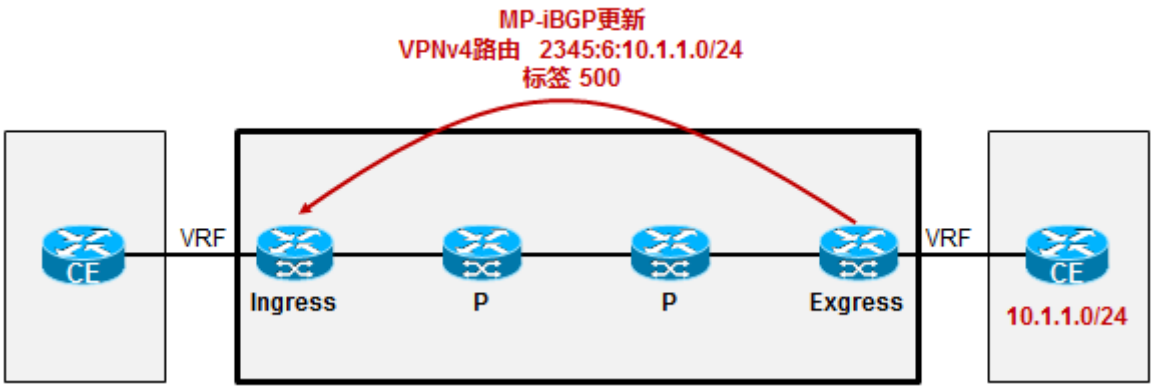
我们先考虑路由层面发生的事情，我们思考的过程是从右到左。首先看 LDP 分配的外层标签。Backbone 的设备都跑了 IGP 协议，例如 OSPF，通过 OSPF，Backbone 内的路由器都学习到了彼此的直连以及 Loopback 路由。然后 Bacboboe 内的路由器激活 LDP 邻接，大家都会为 backbone 内的路由捆绑并分发标签。

Exgress PE 为自己的 Loopback 10.1.254.5/32 分配了一个空标签，因为这个网段是直连。这个 loopback 路由很关键，因为 ExgressPE 和 IngressPE 之间是使用 loopback 建立的 MP-BGP 邻居关系。所以当 Exgress PE 通过 MP-iBGP 更新 VPNv4 前缀给 IngressPE 的时候，这些路由的 Next_Hop 就是 ExgressPE 的 Loopback 地址 10.1.254.5。

Exgress PE 将给 10.1.254.5/32 分配的标签告知给 P 路由器。P 路由器自己也为 10.1.254.5/32 分配自己

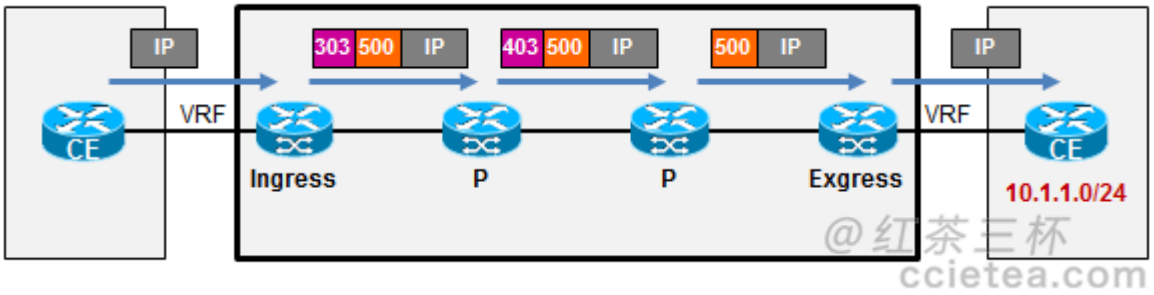
的标签 ,然后也将标签捆绑发给下游 LDP 邻接。接下去的过程就不用多说了吧？总之到最后大家的 LFIB 里，关于 10.1.254.5/32 这个前缀，都有了 local label 和 outgoing label。

2. MP-BGP 分配内层标签



现在是 VPN 标签，VPN 标签由 MP-BGP 分配并告诉给其他的 peer。ExgressPE 为 VPNv4 路由 2345:6:10.1.1.0/24 分配了标签 500，并且将该 VPNv4 路由以及所分配的标签（和其他属性等）通过已经建立好的 MP-iBGP 连接传递给 IngressPE。

3. 数据包在 MPLS VPN 网络中的传输：



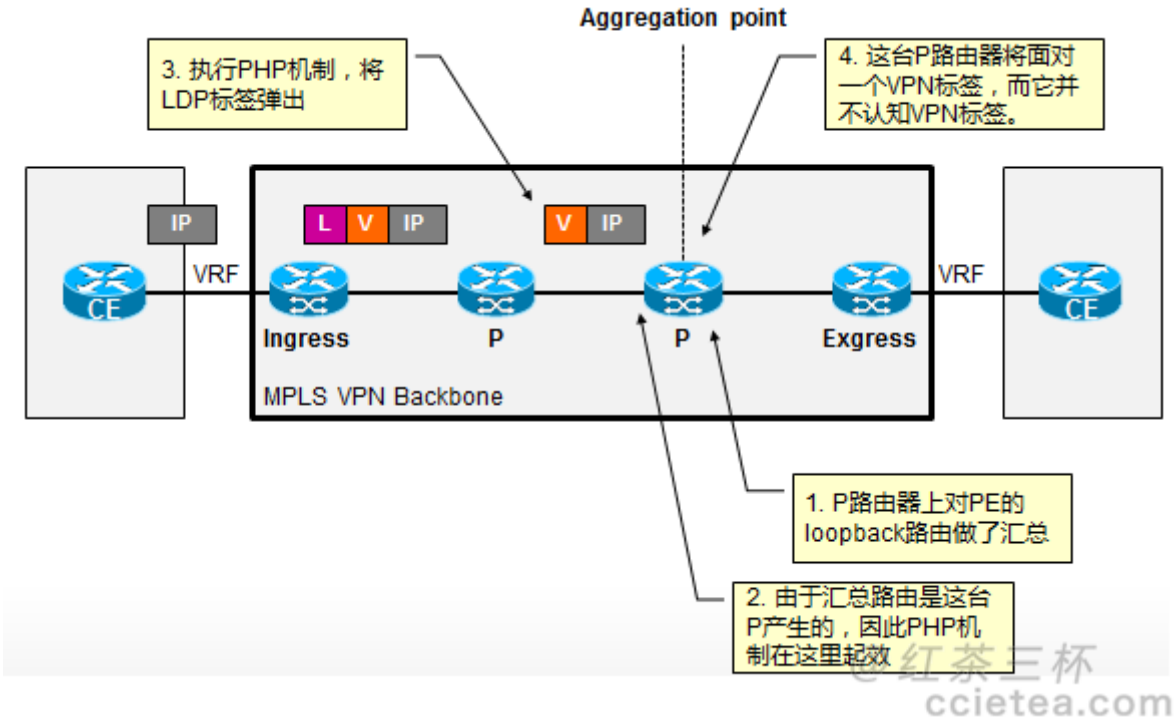
好了，现在左边的 CE，要去访问右边 CE 的 10.1.1.0/24 网络。IPv4 数据包到了 IngressPE，PE 将数据包压上两层标签，外层是 LDP 标签为 303，这是它的上游 P 给他的（为 10.1.254.5 这条前缀分配的），内层标签是 500，这是 Exgress PE 给它的。现在数据包传递到了与 Ingress PE 直连的 P 上，这个 P 将顶层的标签进行交换，交换成 403 并且继续往右传。到了与 ExgressPE 直连的 P，它查看自己的 LFIB 表，发现入站标签 403 对应的出站动作是 POP，于是他将 403 标签弹出，然后把数据丢给 Exgress PE。这里其实是一个 PHP 机制。携带着 VPN 标签 500 的包到了 Exgress PE ,它将 VPN 标签剥去 ,然后将 IP 包转给特定的 CE。

通常来说，PHP 机制会在最后一台 P 和 PE 路由器上出现，因此，LDP 标签会在最后一台 P 中被移除，而进入出站 PE 的报文标签栈中会只剩下 VPN 标签。出站 PE 在 LFIB 表中查找 VPN 标签 ,以进行转发决策。

因为此时本地的标签为 no label，所以剩下的标签栈将被全部移除。报文将以一个 IP 包转发给 CE。如果 IP 报文头部的出站标签为 no label 的话，PE 不会对目的 IP 地址执行查找，正确的下一跳信息将会在 LFIB 中进行标签查找之后获得。只有当出站标签为 Aggregate 的时候，出站 PE 才会在 LFIB 中进行标签查找之后再通过 VRF CEF 表执行 IP 查找（关于 untagged 及 Aggregate 的区别，在本文档有一个小节专门详细解释）。

```
PE1#sh ip bgp vpnv4 rd 234:1 1.1.1.0
BGP routing table entry for 234:1:1.1.1.0/24, version 4
Paths: (1 available, best #1, table ABC)
  Not advertised to any peer
  Local
    10.1.12.1 from 0.0.0.0 (2.2.2.2)
      Origin incomplete, metric 2, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:234:1 OSPF DOMAIN ID:0x0005:0x000000010200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.1.12.2:0
      mpls labels in/out  21/nolabel
```

路由汇总对 VPN 数据传输的影响：



3 MP-BGP

3.1 BGP 多协议扩展

- BGP-4 RFC1771 描述 BGP 用于承载 IPv4 前缀
- BGP 支持多协议 RFC2858“BGP-4 的多协议扩展”
- Open 消息中包含了关于协议性能的参数(ability)
  - 通过 debug ip bgp 可以在邻接关系建立过程中查看到协商过程
  - 使用 show ip bgp neighbors 可以查看到 capabilities 参数

BGP4 的多协议扩展为 BGP 定义了两个新的属性：多协议可达 NLRI、多协议不可达 NLRI，
这两个属性分别用来通告回退回路由。
这两个属性都维护着两个字段：地址族标识符 AFI、后续地址族标示符 SAFI，
这两个字段用来准确的描述 BGP 所承载的是什么类型的路由。

一些地址族 AFI 号码：

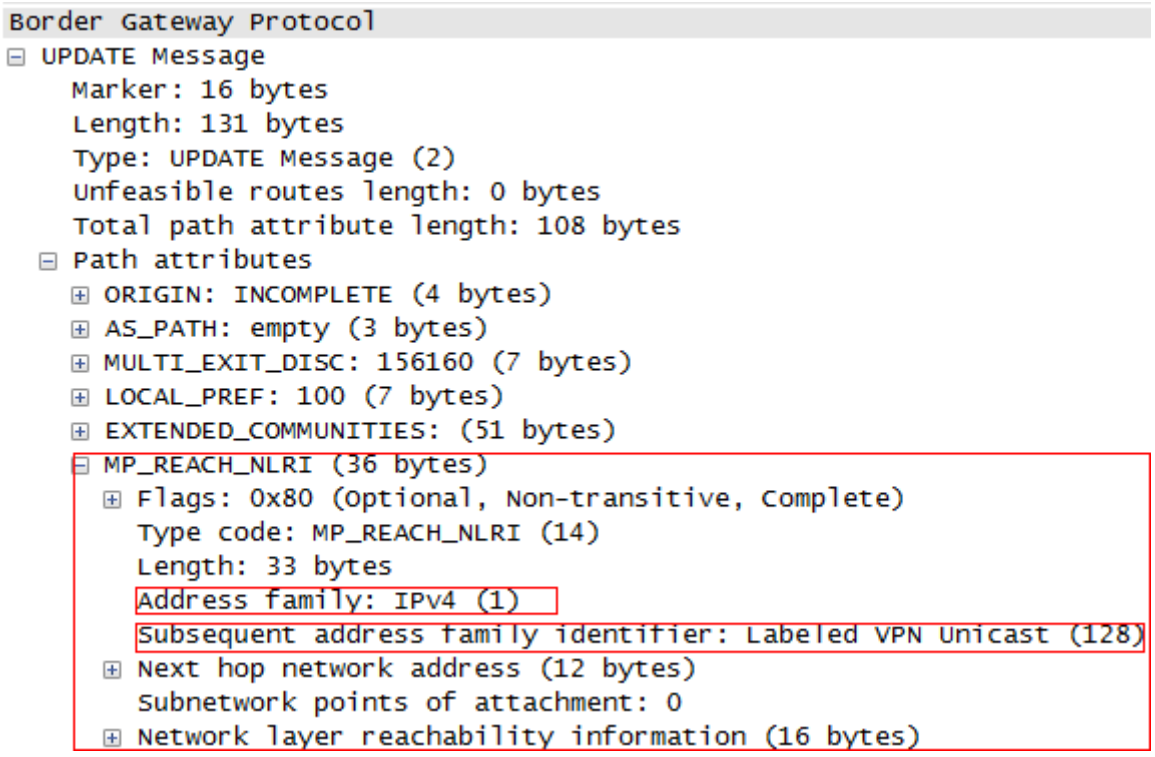
号码	描述
0	保留
1	IPv4
2	IPv6
11	IPX
12	AppleTalk

一些后续地址族标 SAFI 示符：

号码	描述
1	用于转发单播的 NLRI
2	用于转发组播的 NLRI
3	用于转发单播和组播的 NLRI
4	用于转发 ipv4 和标签的 NLRI

128	用于转发带标签 VPN 的 NLRI

例如：



3.2 MP-BGP 的 Update

一个 MP-BGP 的更新包含如下内容

- **VPNv4 前缀**
- **扩展 Community 值**

  RTs、SOO ...
- **Label used for VPN packets forwarding**
- **其他常规 BGP 路径属性**

  MED、LP、AS_PATH、Origin、standard community...

```
Border Gateway Protocol
⊟ UPDATE Message
    Marker: 16 bytes
    Length: 131 bytes
    Type: UPDATE Message (2)
    Unfeasible routes length: 0 bytes
    Total path attribute length: 108 bytes
  ⊟ Path attributes
    ⊟ ORIGIN: INCOMPLETE (4 bytes)
      ⊞ Flags: 0x40 (Well-known, Transitive, Complete)
        Type code: ORIGIN (1)
        Length: 1 byte
        Origin: INCOMPLETE (2)
    ⊟ AS_PATH: empty (3 bytes)
      ⊞ Flags: 0x40 (Well-known, Transitive, Complete)
        Type code: AS_PATH (2)
        Length: 0 bytes
        AS path: empty
    ⊟ MULTI_EXIT_DISC: 156160 (7 bytes)
      ⊞ Flags: 0x80 (Optional, Non-transitive, Complete)
        Type code: MULTI_EXIT_DISC (4)
        Length: 4 bytes
        Multiple exit discriminator: 156160
    ⊟ LOCAL_PREF: 100 (7 bytes)
      ⊞ Flags: 0x40 (Well-known, Transitive, Complete)
        Type code: LOCAL_PREF (5)
        Length: 4 bytes
        Local preference: 100
```

```
⊟ EXTENDED_COMMUNITIES: (51 bytes)
  ⊞ Flags: 0xc0 (Optional, Transitive, Complete)
    Type code: EXTENDED_COMMUNITIES (16)
    Length: 48 bytes
  ⊟ Carried Extended communities
      UnknownRoute Target: 234:2
      Unknown
      Unknown
      Unknown
      Unknown
      Unknown
⊟ MP_REACH_NLRI (36 bytes)
  ⊞ Flags: 0x80 (Optional, Non-transitive, Complete)
    Type code: MP_REACH_NLRI (14)
    Length: 33 bytes
    Address family: IPv4 (1)
    Subsequent address family identifier: Labeled VPN Unicast (128)
  ⊞ Next hop network address (12 bytes)
    Subnetwork points of attachment: 0
  ⊟ Network layer reachability information (16 bytes)
    ⊟ Label Stack=205 (bottom) RD=1:1, IPv4=1.1.1.1/32
        MP Reach NLRI Prefix length: 120
        MP Reach NLRI Label Stack: 205 (bottom)
        MP Reach NLRI Route Distinguisher: 1:1
        MP Reach NLRI IPv4 prefix: 1.1.1.1 (1.1.1.1)
```

Show ip bgp vpnv4 ?

查看 vpnv4 路由

Show ip bgp vpnv4　rd x:y labels

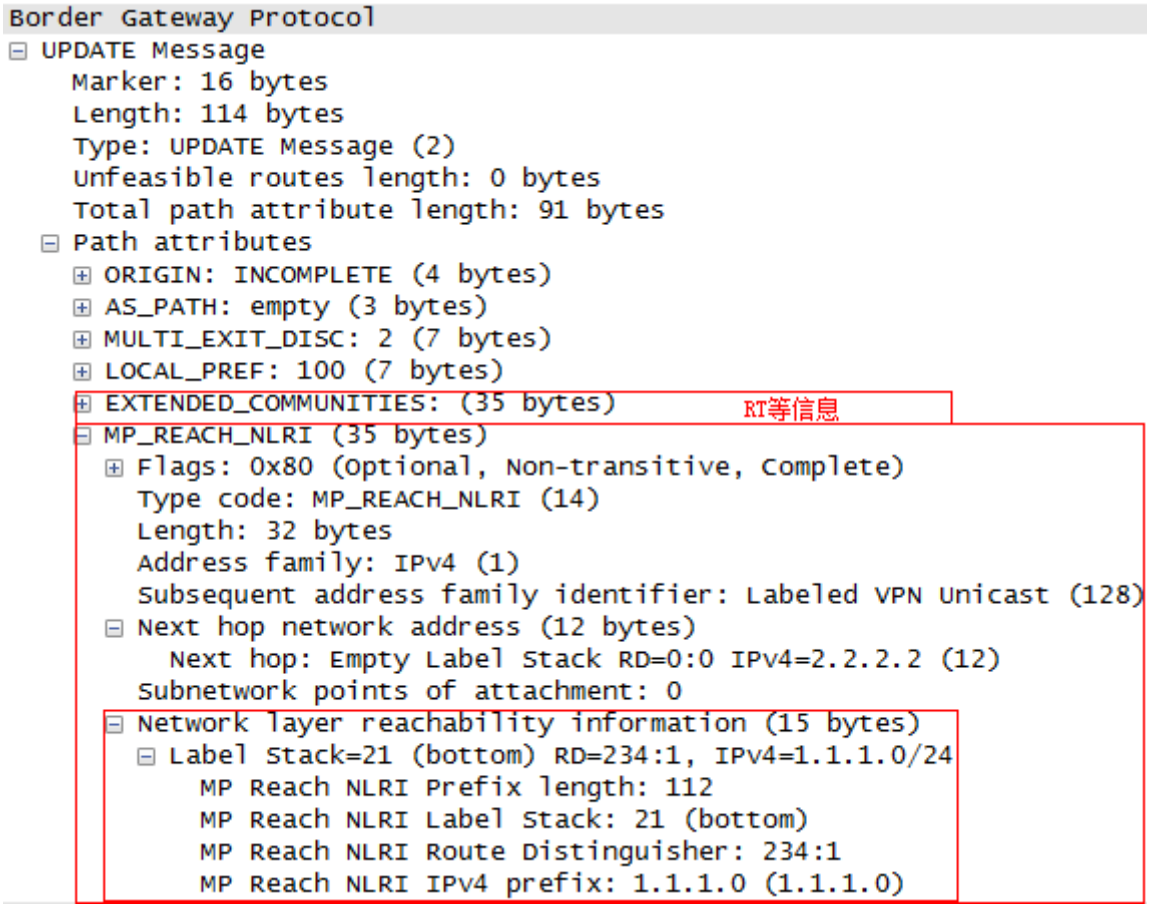
查看 vpnv4 路由分发的标签

3.3 BGP 运载标签

MPLS 在 MPLS VPN 网络中通告 VPNv4 路由，但是这对于正确转发 VPN 流量来说并不够，想想看，一个 IP 数据包，由于要在 MPLS 网络中传输，首先在出站 PE 上被压上一层 LDP 的标签，那么标签包到了目的 PE，也就是入站 PE，PE 怎么知道该将这个 IP 数据包转给哪个 CE？那么我们就再压上一层标签，这层标签叫 VPN 标签，由路由的出站 PE 端分发，并传递给数据的出站 PE，那么路由的出站 PE 会在本地做个标签与 VPNv4 路由

前缀的映射，如此一来，当该路由器收到一个数据包，携带了特定的 VPN 标签，那么它就知道该讲数据包扔给谁。

路由器传递给其 MP-BGP peer 的 BGP 更新包中，可能包含类似如下的内容：



Effects of MPLS VPN Label Propagation：

- **VPN 标签需由 BGP next-hop 路由器分配**
- **路由的 Next-hop 属性在 MP-IBGP 中传递时，不能被更改，例如使用 next-hop-self 命令**
- **PE 路由器必须是 BGP next-hop**
- **当 next-hop 发生变化时，标签会重新分配。这个现象在 inter-AS MPLS VPN 的时候很常见**

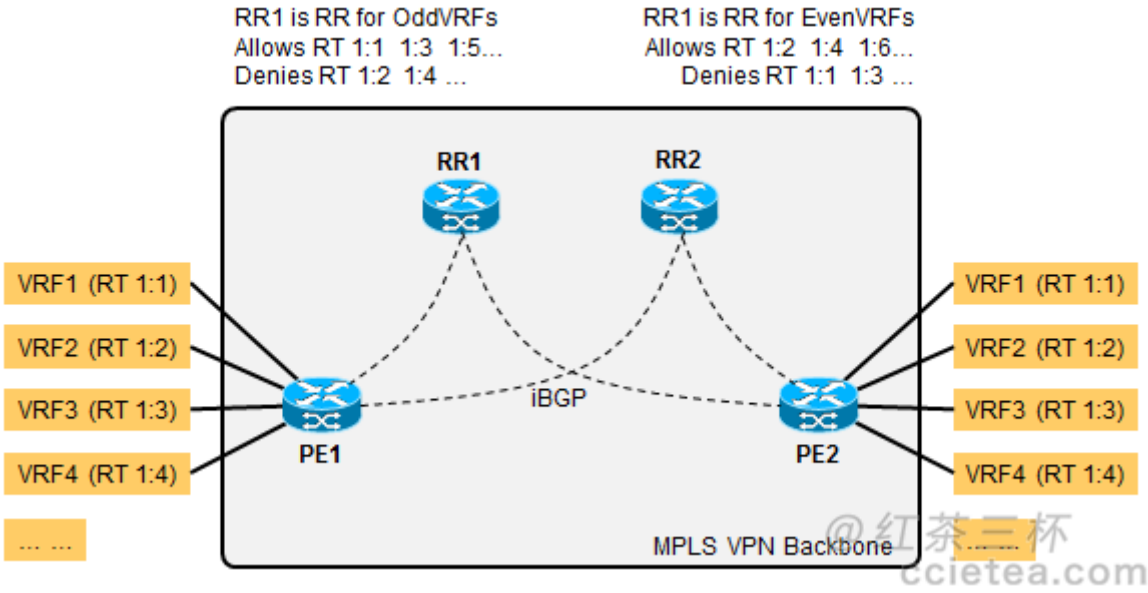
3.4 RR

在 MPLS VPN 网络中，RR 和其他 BGP 设备（PE）的工作方式有所不同，在 RT 没有在 RR 上配置的时候，RR 并不会拒绝 VPNv4 路由，而 PE 路由器如果收到一条**没有在任何 RT import 到 VRF 的 VPNv4** 路由的话，该路由就会被拒绝，PE 采用这种方式来节省内存。而 RR 会接收并且保存所有的 BGP 路由，要想减轻这个负担，

可以通过实施 RR 组来让多个 RR 或者多组 RR 分摊对反射的 VPNv4 路由的负载。

没有必要让一个 RR 或一组 RR 拥有或反射 BGP 表里所有的 VPNv4 路由。可以将这些 VPNv4 路由分成几组，然后让多个 RR 或多个 RR 组来分别承载这几组路由。这样操作可以增强网络的可扩展性。

操作的方法是，在 VPNv4 地址族模式中通过命令 **bgp rr-group** *extcommunity-list* 来实现，这个扩展的 comm-list 用来指定希望通过这个 RR 允许或拒绝的 RT。




```
Router bgp 1
  address-family vpnv4
    neighbor 10.1.254.3 activate
    neighbor 10.1.254.3 route-reflector-client
    neighbor 10.1.254.3 send-community extended
    neighbor 10.1.254.4 activate
    neighbor 10.1.254.4 route-reflector-client
    neighbor 10.1.254.4 send-community extended
    bgp rr-group 1
    exit-address-family
!
ip extcommunity-list 1 permit rt 1:1
ip extcommunity-list 1 permit rt 1:3
ip extcommunity-list 1 permit rt 1:5
```

3.5 BGP 路由选择

1. BGP 多路径

BGP 在做路径选择的时候，只会选择一条最优的也就是 best 的路由，当然，BGP 允许同时将多条 BGP 路由放进路由表来使用，但是，best（也就是在 BGP 表里，用“>”标记的路由）只会有一条。

BGP 多路径有三种情况：

- iBGP　同时注入一条或多条内部 BGP 路径到路由表
- eBGP　同时注入一条或多条外部 BGP 路径到路由表
- eiBGP　同时注入一条或多条内部和外部 BGP 路径到路由表

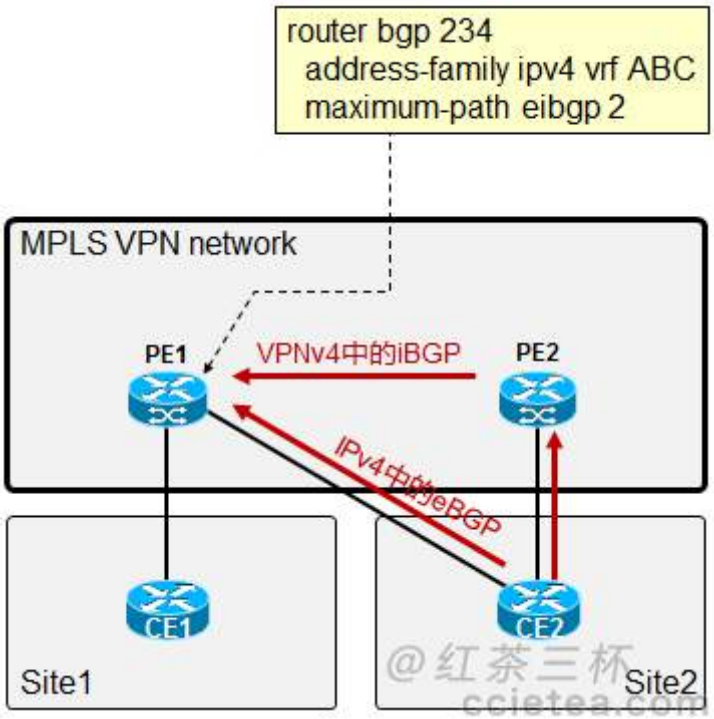
BGP 多路径命令：

BGP 多路径	命令
eBGP	maximum-path n
iBGP	maximum-path ibgp n
eiBGP	maximum-path eibgp n

注意，这些命令是配置在 BGP 地址族模式下的，如 address-family ipv4 vrf ABC。

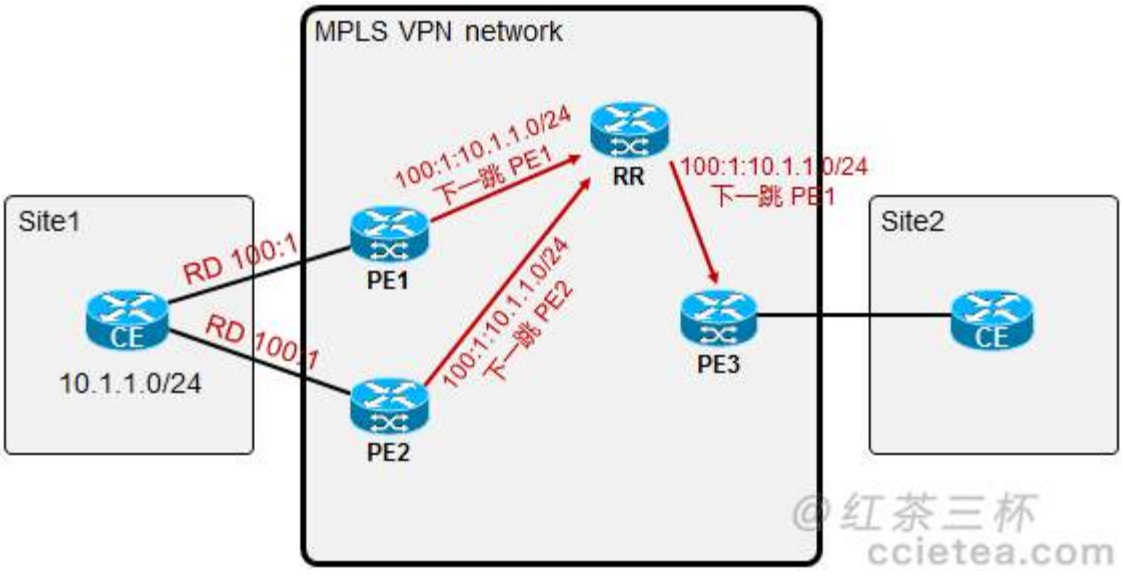
n 默认=1，也就是禁止多路径特性。

BGP 多路径示例：eibgp 多路径



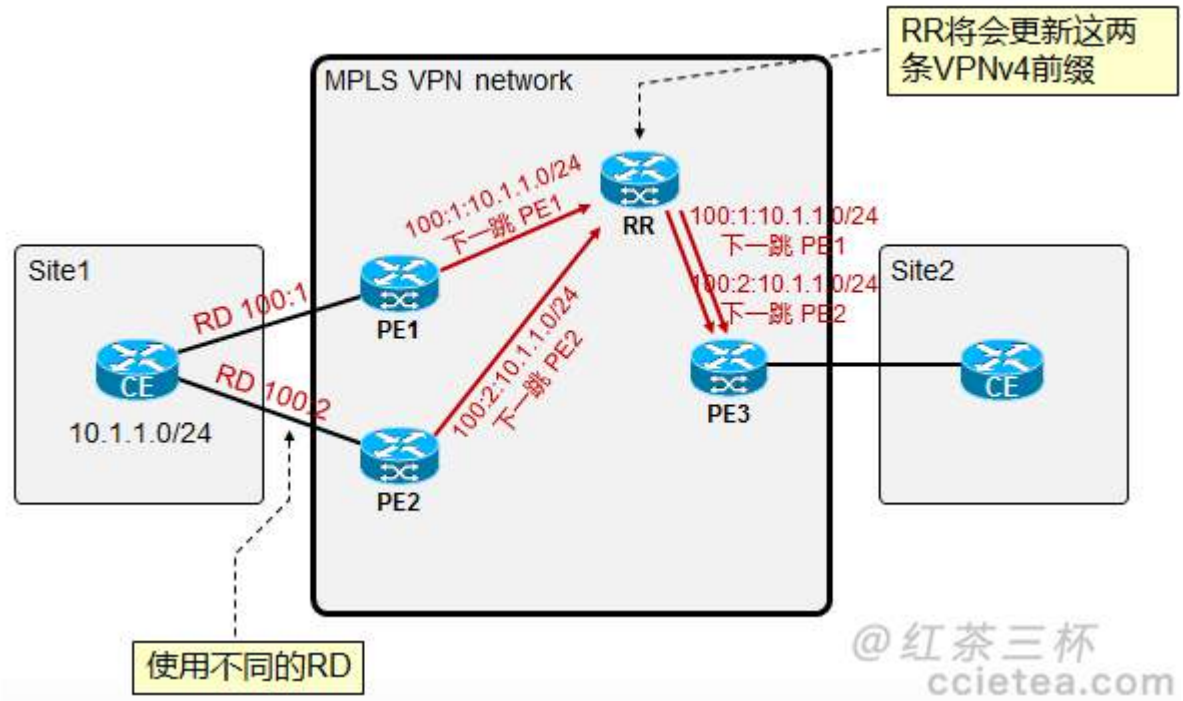
PE 从 Site2 收到一条路由，这条路由通过 PE2 及 CE2 都可达。一条是来自穿越 VRF 接口指向 CE2 的直连 eBGP 对等体，另一条是来自 PE2 对等体的 iBGP，如果 eiBGP 多路径在 PE1 的 address-family ipv4 vrf 中被启用的话，那么 PE1 就可以将这两条 BGP 路径注入到 IP VRF 路由表中负载均衡。

2. 使用多 RD：



我们看上图，CE 双宿主到 PE1 及 PE2，由于 PE1 及 PE2 的 VRF 上我们使用的 RD 都是 100:1，那么 Site1 内 10.1.1.0/24 这条路由，就变成 VPNv4 路由 100:1:10.1.1.0/24，通过 PE1 及 PE2 传递给 RR，而由于这两

条是一模一样的 VPNv4 前缀，因此 RR 只会优选 best 的那条并且更新给 PE3，这样一来就可能存在问题，解决这个问题的办法是，使用多 RD，例如在 PE2 上的 VRF，我们使用 100:2，那么 100:1 及 100:2 两个 RD 附加到 10.1.1.0/24，就变成了两条 VPNv4 前缀，就不存在上面说的问题了。



3.6 MP-BGP 配置

1. **基本配置**

Router(config)# router bgp *as-num*

Router(config-router)# address-family vpnv4

激活 BGP 的 VPNv4 地址族，并进入相关地址族的配置

Router(config-router-af)# neighbor *A.B.C.D* **activate**

在 VPNv4 地址族中，激活特定的 MP-BGP 邻居

Router(config-router-af)# neighbor *A.B.C.D* **route-reflector-client**

如果存在 VPNv4 路由反射环境，那么 RR 的配置需在 VPNv4 的地址族中进行配置。

Router(config-router-af)# neighbor *A.B.C.D* next-hop-self

与 IPv4 地址族中的 next-hop-self 同理，只不过这条命令是针对 VPNv4 路由的 Next-hop 的修改的，谨慎使用，因为我们知道，NH 地址变了，VPN 标签是要重新分发的。该命令主要在域间 VPN 中使用

Router(config-router)# address-family ipv4 vrf *vrf-name*

进入已创建的 VRF 的 BGP ipv4 vrf address-family

每创建一个 VRF，将在 BGP 配置下自动创建 address-family ipv4 vrf vrf-name

2. 其他配置

mpls label mode vrf x protocol bgp-vpnv4 per-prefix !!默认是这个

mpls label mode all-vrfs protocol bgp-vpnv4 per-vrf !!可以改成 per-vrf 的

默认是 per-prefix 分配标签的。也就是一个 VPNv4 的前缀分配一个标签。

可以改成基于 VRF 的标签分配，就是一个 VRF 分配一个标签，不建议修改。

Router(config)# no bgp default-target filter

默认情况下，PE 都忽略 MP-BGP 邻居传递过来的 VPNv4 路由，除非在本地的 VRF 中设置 import RT，这样一来传递过来的那些个 VPNv4 前缀，只有匹配了我配置的这个 import RT 才会被导入到 VRF 路由表中。这条命令关闭基于 RT 的过滤行为，这条命令一旦配置了，那么所有的 VPNv4 路由都被被本地接收。

3. 查看及验证

Show ip bgp all summary

查看所有地址族的邻居等

Show ip bgp vpnv4 all

查看所有的 VPNv4 路由

Show ip bgp vpnv4 all label

查看 vpnv4 路由的标签

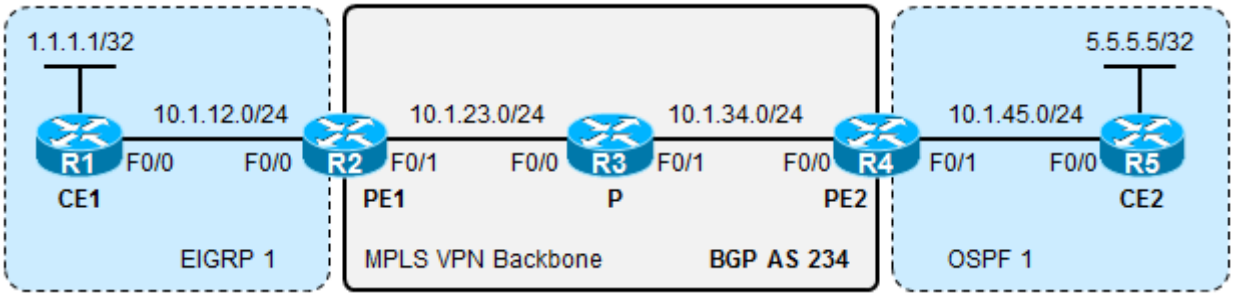
Clear ip bgp vpnv4 unicast ?

清一下 bgp vpnv4 单播路由

Show ip bgp vpnv4 all [vrf x]

查看 vpnv4 路由

3.7 MPLS VPN 基础实验



3.7.1 拓扑及环境

这是一个典型的 MPLS VPN 基础实验环境，R2、R3、R4 为运营商的设备，其中 R2 为 PE1，R4 为 PE2。

R1 及 R5 分别是 CE1 及 CE2，代表同一个 VPN 客户的两个站点。

- IP 编址如图所示，设备互联地址采用 10.1.xy.0/24，其中 xy 为设备编号，x 小 y 大。另外所有的设备均配置 Loopback0，地址采用 x.x.x.x/32，x 为设备编号。
- CE1 与 PE1 之间运行的 PE-CE 路由协议是 EIGRP，EIGRP 进程号是 1；
  CE2 与 PE2 之间运行的 PE-CE 路由协议是 OSPF，使用的 OSPF 进程号是 1。
- MPLS VPN Backbone 内运行的全局 IGP 是 OSPF，使用进程号 100。
- PE1 与 PE2 之间维护 MP-iBGP 邻接关系，交互 VPNv4 路由，BGP 的 AS 号是 234。
- R2、R3、R4 之间维护 LDP 邻接关系，交互 IGP 标签。

3.7.2 需求及步骤

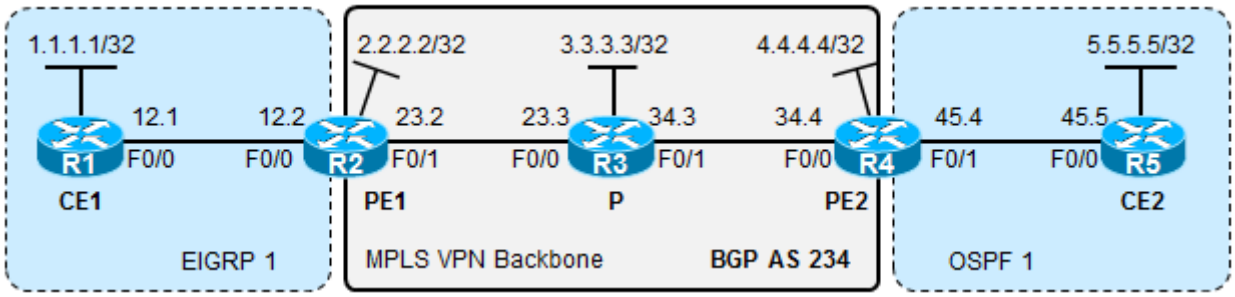
1. 完成基本 IP 配置
2. Core 内运行 OSPF（进程号 100），激活 LDP

3. 在 PE 上创建 VRF，将 PE-CE 间的接口放入 VRF；在 PE 和 CE 之间运行 IGP 协议
4. PE 配置 MP-BGP，建立 MP-iBGP 邻接关系
5. 完成 PE-CE 之间路由的重发布

3.7.3 配置及实现

1. 完成基本 IP 配置

这个就不多说了吧？



2. Core 内运行 OSPF（进程号 100），激活 LDP

R2（PE1）的配置如下：

```
router ospf 100                          !! Core 内运行的 OSPF，用于交互 Core 内的路由
   router-id 2.2.2.2
   network 10.1.23.2 0.0.0.0 area 0
   networl 2.2.2.2 0.0.0.0 area 0
!
Ip cef                                   !!注意，必须开启 ip cef
mpls ldp router-id loopback0             !!设定 LDP router-ID
mpls label range 200 299                 !!为了方便观察实验现象及排错，设置标签范围
interface fast0/1
   mpls ip
```

R3（P）的配置如下：

```
router ospf 100                          !! Core 内运行的 OSPF，用于交互 Core 内的路由
   router-id 3.3.3.3
```

```
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   networl 3.3.3.3 0.0.0.0 area 0
!
Ip cef                                  !!注意，必须开启 ip cef
mpls ldp router-id loopback0            !!设定 LDP router-ID
mpls label range 300 399                !!为了方便观察实验现象及排错，设置标签范围
interface fast0/0
   mpls ip
interface fast0/1
   mpls ip
```

R4（PE2）的配置如下：

```
router ospf 100                         !! Core 内运行的 OSPF，用于交互 Core 内的路由
   network 10.1.34.4 0.0.0.0 area 0
   networl 4.4.4.4 0.0.0.0 area 0
!
Ip cef                                  !!注意，必须开启 ip cef
mpls ldp router-id loopback0            !!设定 LDP router-ID
mpls label range 400 499                !!为了方便观察实验现象及排错，设置标签范围
interface fast0/0
   mpls ip
```

完成配置后，R2、R3、R4 之间就能建立起 LDP 的邻居了。例如：

R2(PE1)#sh mpls ldp neighbor

```
    Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 2.2.2.2:0
        TCP connection: 3.3.3.3.44250 - 2.2.2.2.646
        State: Oper; Msgs sent/rcvd: 11/11; Downstream
        Up time: 00:02:43
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 10.1.23.3
        Addresses bound to peer LDP Ident:
          10.1.23.3       10.1.34.3       3.3.3.3
```

3. **在 PE 上创建 VRF，将 PE-CE 间的接口放入 VRF；在 PE 和 CE 之间运行 IGP 协议**

R2（PE1）的配置如下：

```
ip vrf cisco
   rd 1:1
   route-target export 234:2          !!本地的 RT export
   route-target import 234:4          !!匹配 PE2 所配置的 RT export ,用于将 PE2 传递过来的路由导入 VRF
!
Interface fast0/0
   ip vrf forwarding cisco            !!将该接口放入 VRF cisco
   ip address 10.1.12.1 255.255.255.0
!
router eigrp 1                              !! PE-CE 之间的 IGP 协议，用于从 CE1 学习 VPN 客户路由
   address-family ipv4 vrf cisco
      network 10.1.12.0 0.0.0.255
      autonomous-system 1             !! 指定 EIGRP AS 号，必须指定，否则邻居关系起不来
      exit-address-family
```

R4（PE2）的配置如下：

```
ip vrf cisco
   rd 1:1
   route-target export 234:4          !!本地的 RT export
   route-target import 234:2          !!匹配 PE1 所配置的 RT export
!
Interface fast0/1
   ip vrf forwarding cisco
   ip address 10.1.45.4 255.255.255.0
!
router ospf 1 vrf cisco                !! PE-CE 之间的 IGP 协议，用于从 CE2 学习 VPN 客户路由
   network 10.1.45.4 0.0.0.0 area 0
```

注意，OSPF 的 VRF 是直接体现在 OSPF 进程上的，配置方式如 EIGRP、BGP、RIP 等有所不同。

R1（CE1）的配置如下：

```
router eigrp 1
  network 10.0.0.0
  network 1.1.1.1
```

R5（CE2）的配置如下：

```
router ospf 1
  router-id 5.5.5.5
  network 10.1.45.5 0.0.0.0 area 0
  network 5.5.5.5 0.0.0.0 area 0
```

这样一来 PE-CE 间的 IGP 就配置好了。

R2-PE1#show ip eigrp vrf cisco neighbors

```
IP-EIGRP neighbors for process 1
H   Address      Interface   Hold Uptime    SRTT   RTO   Q     Seq
                             (sec)          (ms)         Cnt   Num
0   10.1.12.1    Fa0/0       14 00:07:59    1256   5000  0     3
```

R2-PE1#show ip route vrf cisco

```
D        1.1.1.1 [90/156160] via 10.1.12.1, 00:09:55, FastEthernet0/0
      10.0.0.0/24 is subnetted, 1 subnets
```

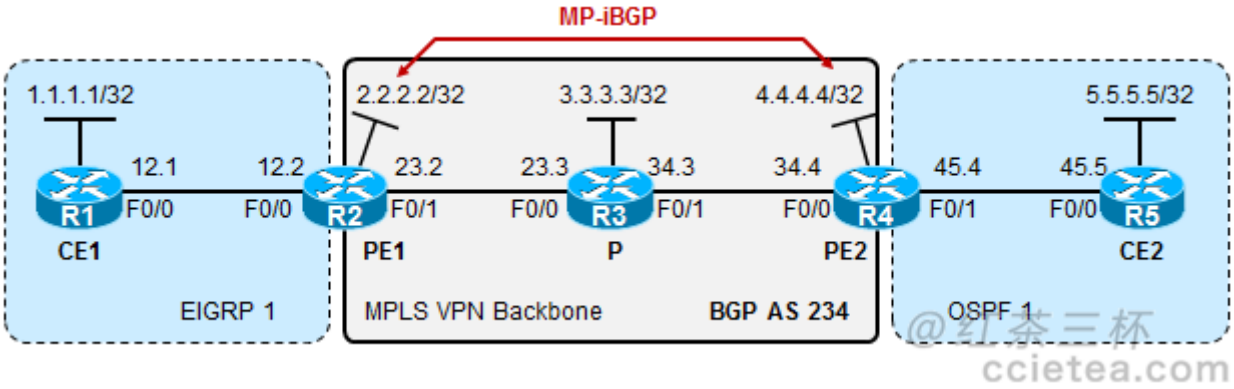
PE1 已经学习到了 CE1 的客户路由。

R4-PE2#sh ip route vrf cisco

```
O        5.5.5.5 [110/2] via 10.1.45.5, 00:03:13, FastEthernet0/1
      10.0.0.0/24 is subnetted, 1 subnets
```

PE2 也学习到了 CE2 的路由。

4. PE 配置 MP-BGP，建立 MP-iBGP 邻接关系



R2（PE1）的配置如下：

```
router bgp 234
   bgp router-id 2.2.2.2
   no bgp default ipv4-unicast                    !! 由于这里我们不需要 PE1-PE2 交互 IPv4 前缀，因此将默认就会自动建立 IPv4 的 BGP 邻接的开关关掉。
   neighbor 4.4.4.4 remote 234
   neighbor 4.4.4.4 update-source loopback 0      !! 配置 VPNv4 的邻接，需要在进程中先指 neighbor，然后再去 VPNv4 地址族内激活，注意这里 PE1-PE2 的邻接关系是建立在 Loopback 接口上的
!
   address-family vpnv4
      neighbor 4.4.4.4 activate                   !!激活与 R4 也就是 PE2 的 VPNv4 连接
      neighbor 4.4.4.4 send-community extended    !! CISCO IOS 会在激活 VPNv4 邻接后自动添加这条命令，但是还是要养成好习惯。
```

R4（PE2）的配置如下：

```
router bgp 234
   bgp router-id 4.4.4.4
   no bgp default ipv4-unicast
   neighbor 2.2.2.2 remote 234
   neighbor 2.2.2.2 update-source loopback 0
!
   address-family vpnv4
      neighbor 2.2.2.2 activate
      neighbor 2.2.2.2 send-community extended
```

验证一下：

R2-PE1#show ip bgp vpnv4 all summary

```
BGP router identifier 2.2.2.2, local AS number 234
BGP table version is 1, main routing table version 1
Neighbor        V    AS    MsgRcvd  MsgSent  TblVer  InQ OutQ  Up/Down  State/PfxRcd
4.4.4.4         4    234   6        8        1       0   0      00:00:29 0
```

R2-PE1#show ip bgp vpnv4 all neighbors 4.4.4.4 | in VPNv4

```
      Address family VPNv4 Unicast: advertised and received
 For address family: VPNv4 Unicast
```

5. 完成 PE-CE 之间路由的重发布

现在，我们要 PE1 已经学习到了 CE1 的路由，接下去，要将这些路由重发布到 BGP 里，以便形成 VPNv4 的前缀来通过 MP-iBGP 传递给 PE2；PE2 上也是同理。再者，为了让 CE1 和 CE2 之间学习到对端站点的路由，还要在 PE1 及 PE2 上，将 BGP 路由重发布进 PE-CE 间的 IGP。

R2（PE1）的配置如下：

```
router eigrp 1
  address-family ipv4 vrf cisco                    !! 注意，是在 IPv4 vrf 地址族下进行重发布
    redistribute bgp 234 metric 100000 100 255 1 1500
!
router bgp 234
  address-family ipv4 vrf cisco                !! 注意，是在 IPv4 vrf 地址族下进行重发布
    redistribute eigrp 1
    no synchronization
```

R4（PE2）的配置如下：

```
router ospf 1 vrf cisco
  redistribute bgp 234 subnets
!
router bgp 234
  address-family ipv4 vrf cisco
    redistribute ospf 1 match internal external
    no synchronization
```

到目前位置，实验涉及的所有配置就完成了。

R1-CE1#sh ip route

D EX 5.5.5.5 [170/53760] via 10.1.12.2, 00:01:08, FastEthernet0/0

D EX 10.1.45.0 [170/53760] via 10.1.12.2, 00:01:08, FastEthernet0/0

CE1 已经学习到了 CE2 所在站点的路由

R5-CE2#sh ip route

O E2 1.1.1.1 [110/1] via 10.1.45.4, 00:02:56, FastEthernet0/0

O E2 10.1.12.0 [110/1] via 10.1.45.4, 00:02:56, FastEthernet0/0

CE2 已经学习到了 CE1 所在站点的路由。

我们来测试一下：

R1-CE1#ping 5.5.5.5 source 1.1.1.1，通了。

然后：

R1-CE1#traceroute 5.5.5.5 source 1.1.1.1

Type escape sequence to abort.

Tracing the route to 5.5.5.5

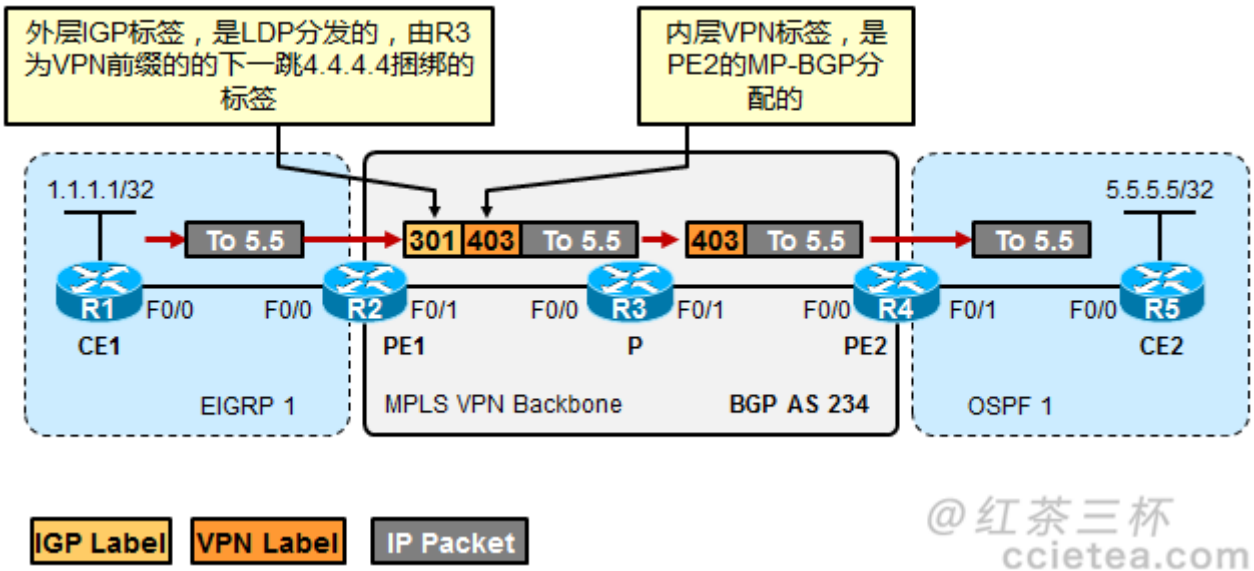
1 10.1.12.2 96 msec 112 msec 20 msec

2 10.1.23.3 **[MPLS: Labels 301/403 Exp 0]** 140 msec 128 msec 140 msec

3 10.1.45.4 **[MPLS: Label 403 Exp 0]** 60 msec 180 msec 116 msec

4 10.1.45.5 124 msec * 124 msec

那么在数据层面上，从 1.1.1.1 访问 5.5.5.5，报文是如何传输的呢：



我们从 traceroute 的结果可以得出上面的数据转发过程。首先 R1 发出的是 IPv4 的报文，源是 1.1.1.1，目的是 5.5.5.5，数据包到达了 R2，由于这是个 IP 数据包，又是从自己的 VRF 接口收到的，因此 R2 查看自己的 vrf CEF 表：

R2-PE1#show ip cef vrf cisco 5.5.5.5

```
5.5.5.5/32
  nexthop 10.1.23.3 FastEthernet0/1 label 301 403
```

CEF 中关于目的 5.5.5.5 ,指示需要压入两层标签 ,然后交给 10.1.23.3 ,于是 R2 将 IPv4 数据包压入标签栈，这个标签栈包含两层标签，然后标签包被交给 R3。

R3 收到这个包，发现是个标签包，于是查看自己的 LFIB 表：

R3#show mpls forwarding-table

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
300	Pop Label	2.2.2.2/32	18716	Fa0/0	10.1.23.2
301	**Pop Label**	**4.4.4.4/32**	**19362**	**Fa0/1**	**10.1.34.4**

LFIB 表显示，当我收到一个顶层标签为 301 的标签包，我要将该顶层标签 POP 弹出，然后交给下一跳 10.1.34.4。于是 R3 将收到的标签包的 301 标签弹出，然后直接交给 10.1.34.4。注意，这里其实是个 PHP 机制。

那么 R4 也就是 PE2 收到了 R3 发过来的包，发现是个标签包，于是它也查看 LFIB 表：

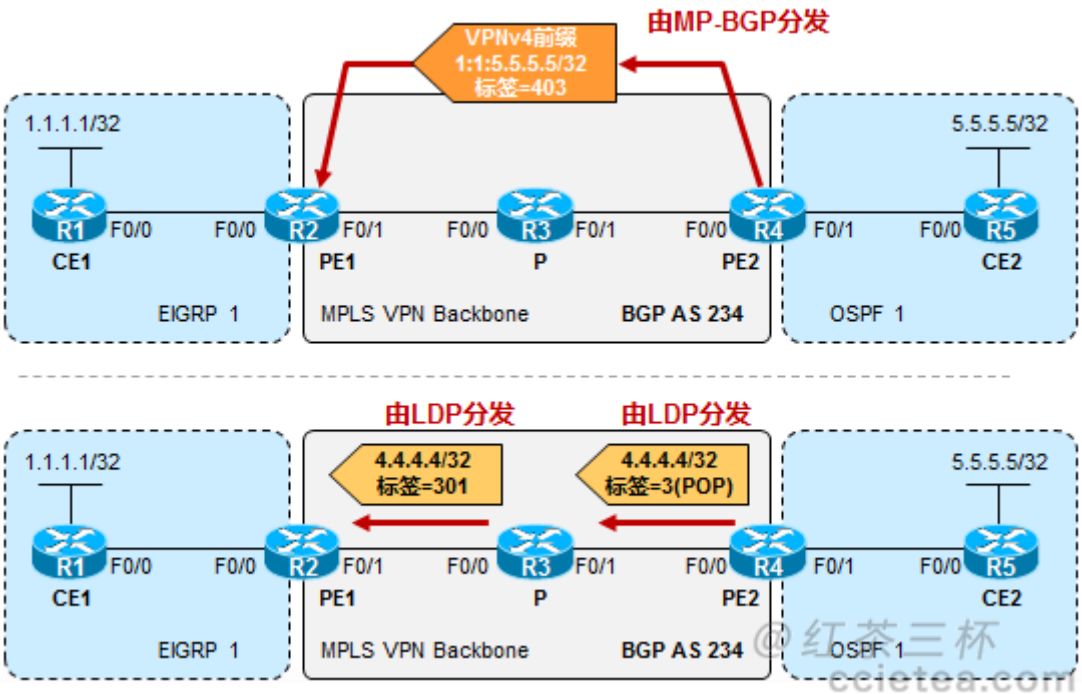
R4-PE2#show mpls forwarding-table

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
400	Pop Label	3.3.3.3/32	0	Fa0/0	10.1.34.3

401	300	2.2.2.2/32	0	Fa0/0	10.1.34.3
402	Pop Label	10.1.23.0/24	0	Fa0/0	10.1.34.3
403	**No Label**	**5.5.5.5/32[V]**	**4038**	**Fa0/1**	**10.1.45.5**
404	No Label	10.1.45.0/24[V]	0	aggregate/cisco	

它发现标签 403，对应的 outgoing 是 no label，于是它将整个标签栈都弹出（实际上只剩下一层标签了），然后将原始数据，也就是 IPv4 数据直接丢给 10.1.45.5。数据包就这么到了 R5 也就是 CE2。

了解了数据层面的工作，我们再看看控制层面的工作，也就是标签是怎么传递的：

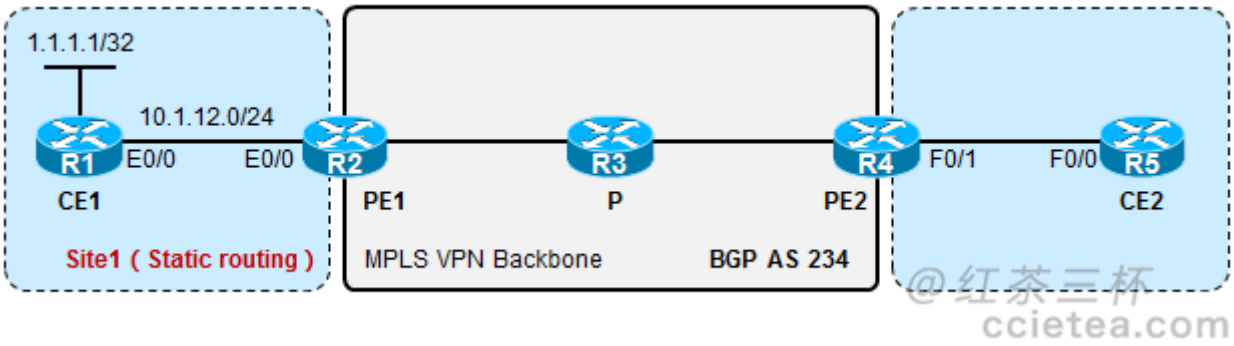


这个图大家就非常清楚了吧？首先看内层 VPN 标签，拿 5.5.5.5 举例，R4(PE2)在通过 MP-iBGP 通告 VPNv4 前缀 1:1:5.5.5.5/32 时，会为这个前缀同时捆绑一个 VPN 标签 403，然后发给 PE1。这层标签，用于帮助 PE2 识别所收到的标签数据归属于哪一个 VRF 哪一个下一跳 CE。那么当 PE1 要向 5.5.5.5 发送数据时，报文进 MPLS Backbone 前就要先压入 PE2 分配给 5.5.5.5 的 VPN 标签也就是 403。

那么压入了 403 的标签，是不是就能在 MPLS VPN Backbone 里传输了呢？运营商里 P 路由器是不可能知道客户路由的，往往也并不知道 VPNv4 前缀。那么为了让这些 VPN 流量能够在 P 网络中传输，势必要增加一层标签，这就是 IGP 标签，它由 LDP 捆绑并分发。注意，当 PE1 要将已经压了一层 VPN 标签的报文，放进 MPLS Backbone 时，外层标签压入的是什么标签呢？其实是压的是 R3 给 4.4.4.4 这个前缀分配的 LDP 标签。而 4.4.4.4 又是 PE1 去往 VPN 路由 5.5.5.5 的下一跳。因此实际上 PE1 这里使用的是下一跳的标签。

4 PE-CE 路由协议

4.1 使用静态路由



1. **实验环境**
   - Site1 内 CE1 的 E0/0 接口 IP 为 10.1.12.1/24；PE2 的 E0/0 为 10.1.12.2/24
   - Site1 内 PE-CE 使用静态路由
   - 我们重点看 CE1 及 PE1 的配置，其他设备的配置在这里不是重点，但是大家还是要有个全局观

2. **实验配置**

CE1 的配置如下：

```
ip route 0.0.0.0 0.0.0.0 10.1.12.2
```

这个实验中，我们 CE1 采用最简单的方法，直接指一条默认路由出去。

当然最好的方式还是指 Site2 的 VPN 客户路由。因为有可能默认路由需要用于本地访问 Internet。

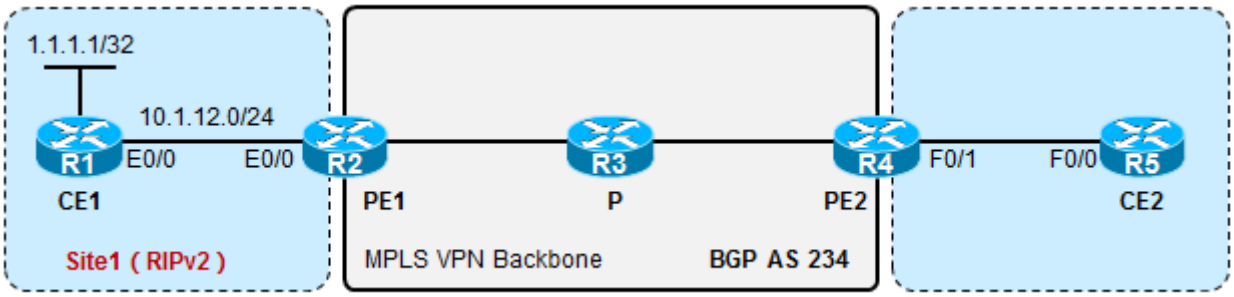
PE1 的配置如下：

```
ip vrf ABC
   rd 1:1
   route-target 234:2
!
ip route vrf ABC 1.1.1.1 255.255.255.255 10.1.12.1 e0/0
                                !! 为 VRF ABC 路由表创建一条指向 CE 客户的 VPN 路由
router bgp 234
   address-family ipv4 vrf ABC
```

```
  redistribute static
  !! 将静态的 VPN 路由重发布进 BGP，以便形成 VPNv4 的前缀更新给 VPN 对端站点。
exit-address-family
```

注意，在配置 VRF 静态路由的时候，由于 VRF 跟接口有关联，因此强烈建议采用关联出接口及下一跳 IP 的方式来配置这条 VRF 静态路由。

4.2 使用 RIPv2



1. **实验环境**
   - Site1 内 CE1 的 E0/0 接口 IP 为 10.1.12.1/24；PE2 的 E0/0 为 10.1.12.2/24
   - Site1 内 PE-CE 使用 RIPv2
   - 我们重点看 CE1 及 PE1 的配置，其他设备的配置在这里不是重点，但是大家还是要有个全局观

2. **设备配置**

CE1 的配置如下：

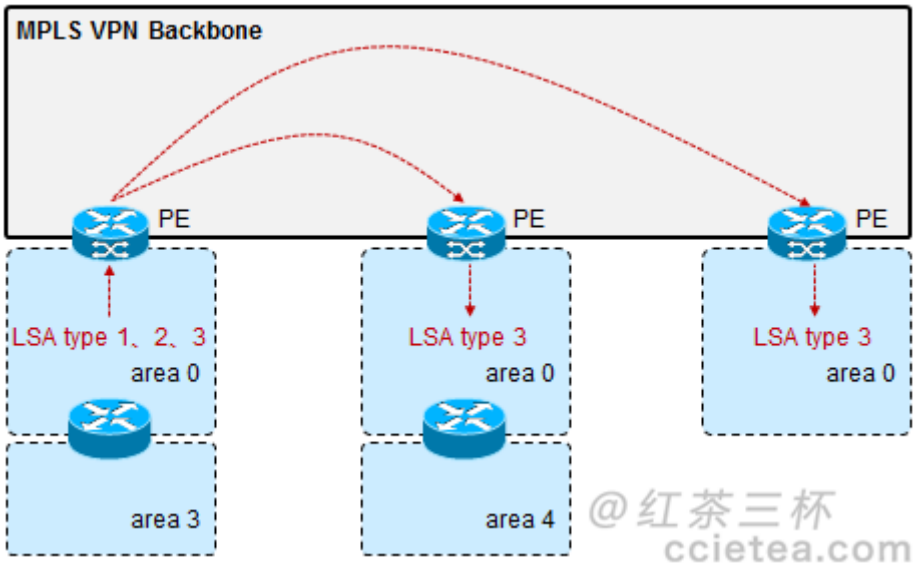
```
router rip
  version 2
  no auto-summary
  network 10.0.0.0
  network 1.0.0.0
```

PE1 的配置如下：

```
router rip
  address-family ipv4 vrf ABC          !! 注意是在 RIP 进程的 IPv4 vrf 地址族下进行配置
    version 2
```

```
    no auto-summary
    network 10.0.0.0
    redistribute bgp 234 metric [transparent]
!! 将 BGP 路由重发布进 RIP，以便让 CE1 能够学习到对端 Site 的客户路由。
!! 动态路由协议重发布到 RIP 默认的种子度量是无穷大，因此在重发布的时候需指定 metric，如果使用 transparent 关键字，则这些 RIP 路由将继承 BGP 的 MED 值。
!
router bgp 2345
 address-family ipv4 vrf ABC
    redistribute rip
  exit-address-family
```

4.3 使用 OSPF

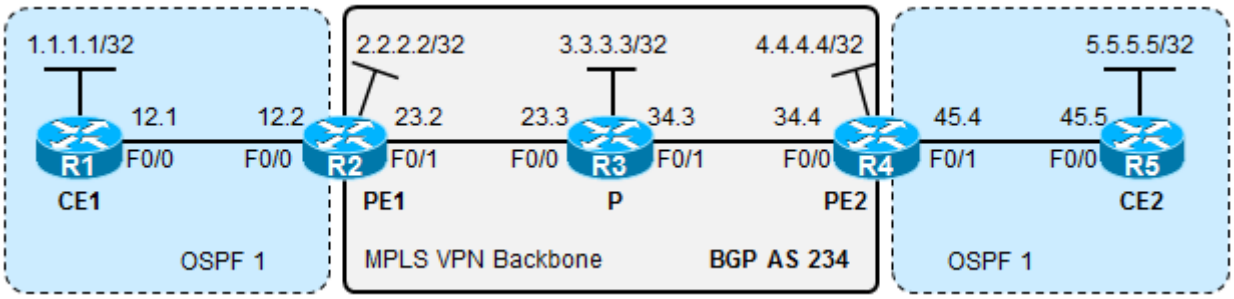


如果在 PE-CE 链路中使用 OSPF，那么需要在 PE 路由器上将 OSPF 重发布进 MP-BGP，以及将 MP-BGP 重发布进 OSPF。在远端 PE 如果它接收本地 PE 传过去的 VPNv4 路由再重发布进本地 VPN 的 OSPF 域后，变成外部路由就导致路由的优先级变低，另一个问题是，MPLS VPN Backbone 将用户的 OSPF 网络从设计上“切断”了。针对这些问题，MPLS VPN 有非常人性化的设计。

MPLS VPN 在对 OSPF 网络的承载上有非常非常独特的设计。从宏观层面上说 ,对于 OSPF 而言,MPLS VPN Backbone 相当于一个 OSPF 超级骨干区域。 实际上，来自 CE 的 OSPF 内部路由(LSA1、2、3)在被重发布进

MP-iBGP、变成 vpnv4 路由再被远端 PE 重发布进 OSPF 后，其实是以 3 类 LSA 的形式注入到其本地 OSPF 域的（当然还存在许多复杂的情况，这里说的是一般，一般情况）。这么说来，这些 PE，在这种情况下，又有那么点 ABR 的问题，但其实他们的身份是 ASBR，很有意思，对吧？更详细的内容，在本章的 OSPF 网络设计一节展开阐述。

4.3.1 OSPF VRF 配置



R2 的配置如下：

```
ip vrf ABC
   rd 1:1
   route-target export 234:2
   route-target import 234:4
!
interface Loopback0
   ip address 2.2.2.2 255.255.255.255
!
mpls label range 200 299
!! 为了方便观察实验现象，设置一下标签空间
mpls ldp router-id Loopback0
!
interface FastEthernet0/0
   ip vrf forwarding ABC
   !!接口 F0/0 放入 VRF ABC
   ip address 10.1.12.2 255.255.255.0
!
interface FastEthernet0/1
   ip address 10.1.23.2 255.255.255.0
   mpls ip
!
router ospf 1 vrf ABC
   redistribute bgp 234 subnets
```

```
router ospf 100
   network 2.2.2.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   !! 全局的 OSPF ,用于和 MPLS Backbone 内交互路由，以便建立 MP-BGP 邻接关系，以及为后续的 LDP 服务
!
!
router bgp 234
   no bgp default ipv4-unicast
   neighbor 4.4.4.4 remote-as 234
   neighbor 4.4.4.4 update-source Loopback0
!
   address-family vpnv4
      neighbor 4.4.4.4 activate
      !! 激活与 4.4.4.4 也就是 PE2 的 VPNv4 连接
      neighbor 4.4.4.4 send-community extended
      exit-address-family
!
    address-family ipv4 vrf ABC
       redistribute ospf 1 vrf ABC match internal external
       !!将 OSPF 1 VRF ABC 中的路由注入到 BGP，从而形成 VPNv4 的前缀
       exit-address-family
```

network 10.1.12.2 0.0.0.0 area 0 **!!VRF OSPF 进程，用于从 CE 学习 VPN 客户的路由** （未完，向右看）	

R2#show ip ospf 1

Routing Process "ospf 1" with ID 10.1.12.2

Domain ID type 0x0005, value 0.0.0.1

!! Domain ID 等会会介绍

Start time: 00:03:15.920, Time elapsed: 00:29:15.052

Supports only single TOS(TOS0) routes

Supports opaque LSA

Supports Link-local Signaling (LLS)

Supports area transit capability

Connected to MPLS VPN Superbackbone, VRF ABC

It is an area border and autonomous system boundary router

!! 这句话非常耐人寻味

Redistributing External Routes from,

bgp 234, includes subnets in redistribution

Router is not originating router-LSAs with maximum metric

Initial SPF schedule delay 5000 msecs

Minimum hold time between two consecutive SPFs 10000 msecs

Maximum wait time between two consecutive SPFs 10000 msecs

4.3.2 用于 OSPF 的 BGP 扩展 community

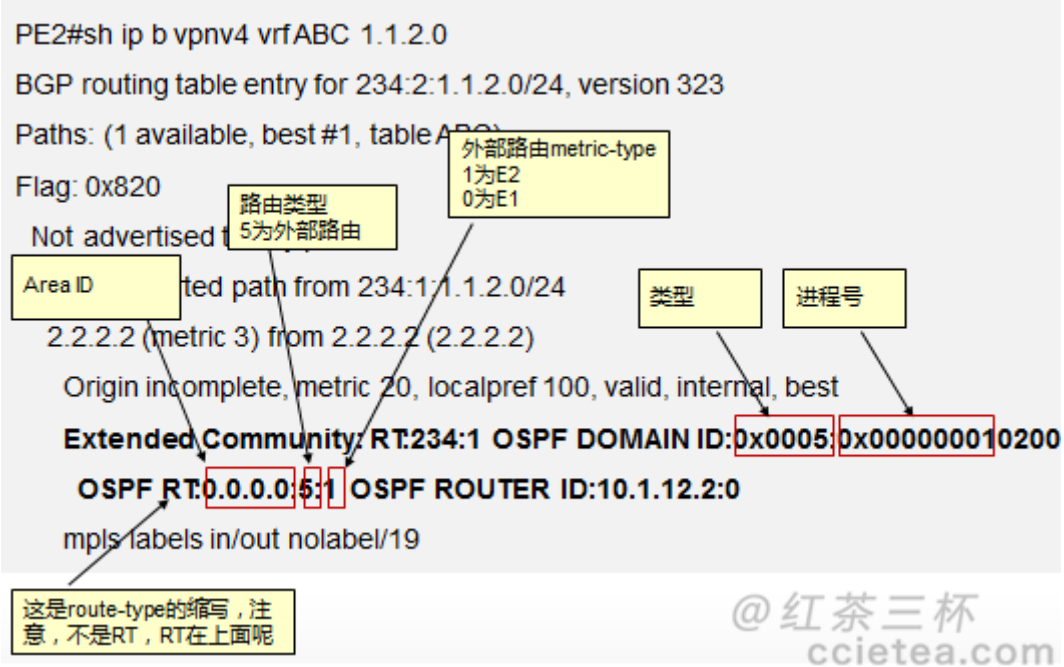
要想让 OSPF 路由的特征能够穿越 MPLS VPN 骨干网络，需要额外定义一些 BGP 扩展 community。这些扩展 community 设计使得 OSPF 路由可以非常华丽丽的地在远端 PE 上重建，从而保持我们 OSPF 网络设计的一致性，非常牛逼的设计和想法。

可以通过 MP-BGP 传递的 OSPF 特性包括：

- 路由类型
- 区域号
- OSPF 路由器 ID

- 域 ID
- OSPF 外部路由的度量值类型 1 或 2

如下图：



- **DomainID 是一个域 ID，默认情况下等于 OSPF 的进程号。**

  DomainID 告诉远端 PE 路由器，通告的是否为一条域外的 OSPF 路由。

  如果 OSPF 路由类型为 LSA1、2、3（且本地 PE 及远端 PE 的 OSPF VRF 进程号相同），则路由重发布到远端 PE 的 OSPF 域后是以 LSA3 的形式注入。

  如果 PE 路由器所收到的路由的 domainID 与本地 OSPF VRF 进程的 DomainID 不一致的话（其实说白点，就是两个 PE 上，OSPF VRF 进程的进程号不一致），这条路由将会以一条 OSPF 外部路由也就是 LSA5 类型的形式通告，以提供对网络中不同 OSPF 进程之间重发布 IP 路由的支持。

  如果 domainID 能够匹配 OSPF 进程 ID，该路由将以 LSA3 的形式通告。

  当然，如果两端 PE 的 OSPF 进程号相同，传递过来的路由又是内部路由，但是你又希望路由重发布到本地 OSPF 后以外部路由的形式注入，那么可以在 PE 路由器上修改 DomainID。命令如下：

```
router ospf 1 vrf ABC
  domain-id ?
```

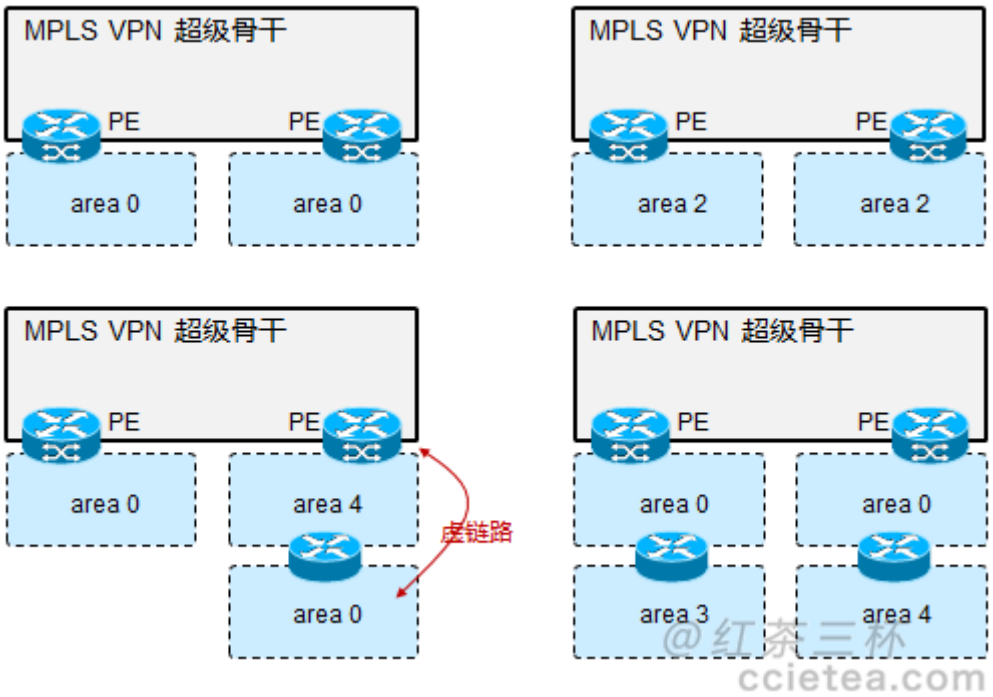
  修改之后，可使用 show ip ospf 1 来查看。

- **关于上图中 OSPF RT，也就是 OSPF Route-type 后面的路由类型的含义：**
  - :1:0or:2:0 indicating intra-area LSA-1 routes

- :3:0 indicating inter-area LSA-3 routes
- :5:0or:5:1 indicating external LSA-5 routes (type1 and type 2 respectively)
- :7:0or:7:1 indicating NSSA LSA-7 routes (type 1 and type2 respectively)

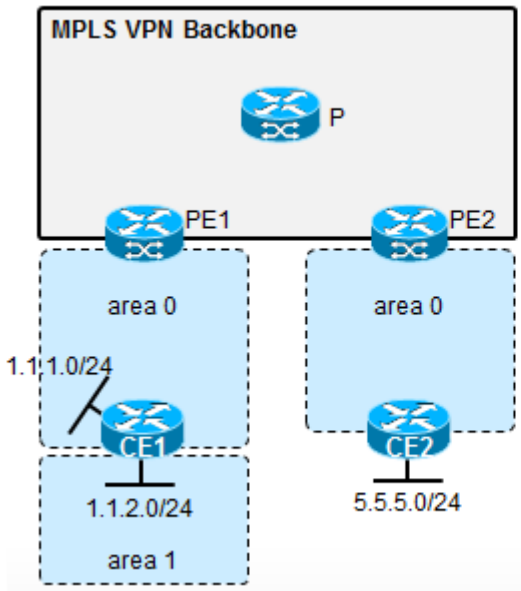
4.3.3 OSPF 网络设计

我们来考虑一下所有可能的情况：



我们分别来讨论这几个 CASE：

- **OSPF 网络设计 case1**

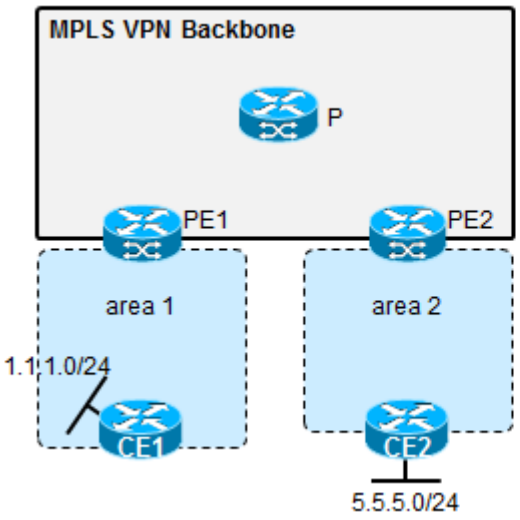


- PE1、PE2在VRF上跑ospf，与各自的CE建立邻接关系
- PE1、PE2上vrf ospf进程号相同时
- CE1上1.1.1.0属area0，1.1.2.0属area1
- CE2上5.5.5.0为重发布
- CE1上学习到的5.5.5.0路由为OE
- CE2上学习到的1.1.1.0路由为O IA
- CE2上学习到的1.1.2.0路由为O IA

- 在PE2上修改OSPF进程号（与PE1）不同，或者修改DomainID，则上述路由均变成OE类型

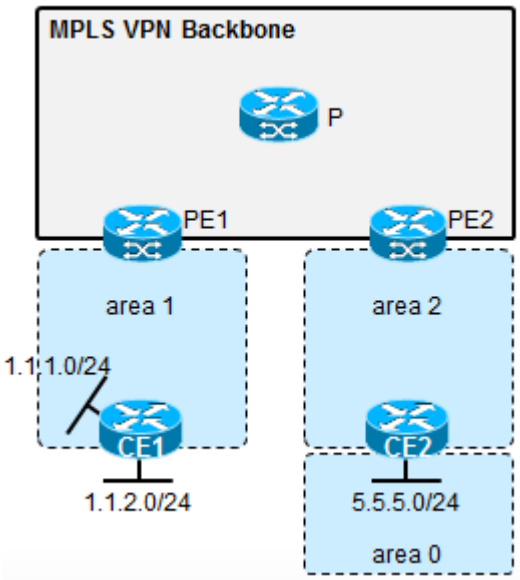
这个实验中，PE1 及 PE2 的 VRF OSPF 都采用相同的进程 ID，得出的现象如上。

- **OSPF 网络设计 case2**



- PE1、PE2在VRF上跑ospf，与各自的CE建立邻接关系
- PE1、PE2上vrf ospf进程号相同时
- CE1上1.1.1.0属area1，PE1与CE1在area1建立邻接
- CE2上5.5.5.0为重发布
- CE1上学习到的5.5.5.0路由为OE
- CE2上学习到的1.1.1.0路由为O IA

- **OSPF 网络设计 case3**



- PE1、PE2在VRF上跑ospf，与各自的CE建立邻接关系
- PE1、PE2上vrf ospf进程号相同时
- CE1上1.1.1.0属area1，1.1.2.0为重发布
- PE1与CE1在area1建立邻接
- CE2上5.5.5.0为属area0
- CE2上学习到的1.1.1.0路由为O IA
- CE2上学习到1.1.2.0的OE路由
- **CE2的area0里无法学习到1.1.1.0、2.0路由（实则CE2没有将这些三类LSA转进area0）**
- **PE2上能收到CE2发送的5.5.5.0的LSA3，但是不会装载进路由表，自然CE1也就无法学习到5.5.5.0**

这个实验就需要格外注意了。实验的结果是：

CE1 这里过来的两条路由 ,1.1.1.0 及 1.1.2.0 分别是内部及外部路由 ,这两条路由经过超级骨干到了 PE2，PE2 将 OSPF 路由重建，注入 area2，那么 CE2 上能学习到 1.1.1.0 的 OIA 路由及 1.1.2.0 的 OE 路由。实际上 CE2 的 OSPF DATABASE 里 area2 确实存在上述相关的 LSA ,但是 CE2 却不将这些 LSA 转进 area0，虽然它是一台 ABR。CE2 此时的 OSPF Database 如下：

```
R5#sh ip ospf da

            OSPF Router with ID (5.5.5.5) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
5.5.5.5         5.5.5.5         268         0x80000001 0x00F801 1

                Summary Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
10.1.45.0       5.5.5.5         263         0x80000001 0x007D72
// 没有为 area0 注入关于 1.1.1.0 的三类 LSA
                Summary ASB Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
44.44.44.44     5.5.5.5         263         0x80000001 0x00FE77

                Router Link States (Area 2)
Link ID         ADV Router      Age         Seq#       Checksum Link count
```

```
5.5.5.5          5.5.5.5          262         0x80000007 0x007E0C 1
44.44.44.44      44.44.44.44      281         0x80000005 0x00DA77 1

                Net Link States (Area 2)
Link ID          ADV Router       Age         Seq#       Checksum
10.1.45.4        44.44.44.44      1427        0x80000002 0x00A1E5

                Summary Net Link States (Area 2)
Link ID          ADV Router       Age         Seq#       Checksum
1.1.1.0          44.44.44.44      58          0x80000001 0x00C442
5.5.5.0          5.5.5.5          265         0x80000001 0x0048D0
10.1.12.0        44.44.44.44      58          0x80000001 0x00CB28

                Type-5 AS External Link States
Link ID          ADV Router       Age         Seq#       Checksum Tag
1.1.2.0          44.44.44.44      58          0x80000001 0x004CE3 3489661162
```

另一方面，CE2 的路由 5.5.5.0，由于 CE2 是 ABR，因此它将 5.5.5.0 的 3 类 LSA 注入 area2，那么 PE2 就能学习到这个 LSA ,但是 ,却不装载进路由表 ,因为 PE2 认为自己是台 ABR ,OSPF 要求 3 类 LSA 必须经过 area0 骨干区域来中转，然而这里 PE2 并没有与 area0 直连，因此 5.5.5.0 没有被装入路由表，自然 CE1 也就无法学习到。通过观察 PE2 的 OSPF database 可以一目了然：

```
PE2#sh ip os 1 da

            OSPF Router with ID (44.44.44.44) (Process ID 1)
                Router Link States (Area 0)
Link ID          ADV Router       Age         Seq#       Checksum Link count
44.44.44.44      44.44.44.44      142         0x80000001 0x008955 0

                Summary Net Link States (Area 0)
Link ID          ADV Router       Age         Seq#       Checksum
1.1.1.0          44.44.44.44      142         0x80000001 0x00C442
10.1.12.0        44.44.44.44      142         0x80000001 0x00CB28
10.1.45.0        44.44.44.44      138         0x80000001 0x00E66C
```

// 这里很关键，PE2 并没有任何接口属于 vrf OSPF 进程，但是，database 里却有 area0，这是因为 MPLS

VPN 中，PE 充当 ABR 的角色，我们可以看到 area0 中有 CE1 发过来的路由的相关 LSA，以及 PE2 与 CE2 直连网段 LSA

Router Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
5.5.5.5	5.5.5.5	143	0x80000008	0x008602	1
44.44.44.44	44.44.44.44	134	0x80000006	0x00E26D	1

Net Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum
10.1.45.5	5.5.5.5	143	0x80000001	0x00A281

Summary Net Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum	
1.1.1.0	44.44.44.44	137	0x80000002	0x00C243	
5.5.5.0	**5.5.5.5**	**655**	**0x80000001**	**0x0048D0**	**//收到了 3 类 LSA**
10.1.12.0	44.44.44.44	137	0x80000002	0x00C929	

Type-5 AS External Link States

Link ID	ADV Router	Age	Seq#	Checksum	Tag
1.1.2.0	44.44.44.44	137	0x80000002	0x004AE4	3489661162

找到了问题的原因，那么我们在 PE2 与 CE2 之间，通过 area2 建立个 virtual-link，
这样，路由就都能学习到了。

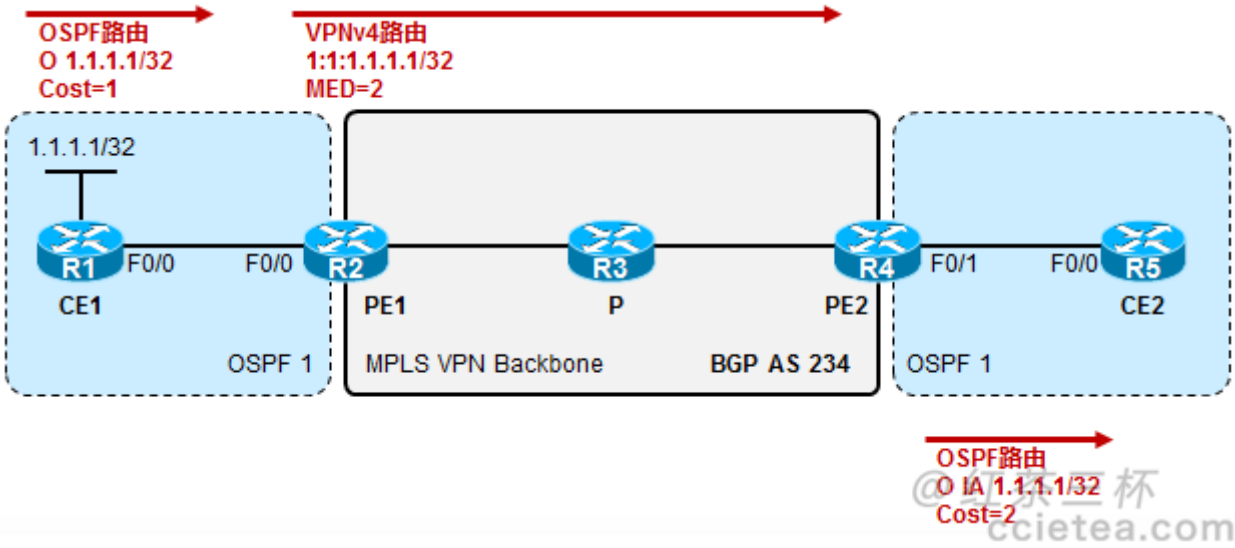
- **总结：**
  - MPLS VPN 在 OSPF 场点之间存在一个超级骨干区域 superbackbone，这当然不是一个 OSPF 区域，不过，它扮演了一个骨干区域（area0）的角色。同时 PE 也就扮演了一个 ABR 的角色（通过查看 PE 产生的 1 类 LSA 也能证明这点）。
  - 从客户的角度，可以将 superbackbone 看做一个 cost=0 的 area0
  - 如果一个 VRF 的多个场点有一台 PE 在区域 0 中，那么这个 area0 被分割成了多块，通常来说被分割的骨干区域需要使用 virtual-link 来连接，但是在 MPLS VPN 中由于有 iBGP 来运载 OSPF 路由，因此不需要虚链路。OSPF 路由会在 PE 路由器上重建（超级骨干不会直接泛洪 LSA）。
  - PE 路由器扮演了 ABR 的角色。它将 type3 LSA 通告给 CE 路由器，CE 路由器可以在 area0 中，也可以在其他区域，但是如果一个场点拥有多个区域，PE 路由器就必须在 area0 中，因为他们是 ABR，如

果他们不在 area0 中，就需要在 PE 上创建 virtual-link 来确保 PE 和 area0 的连接。

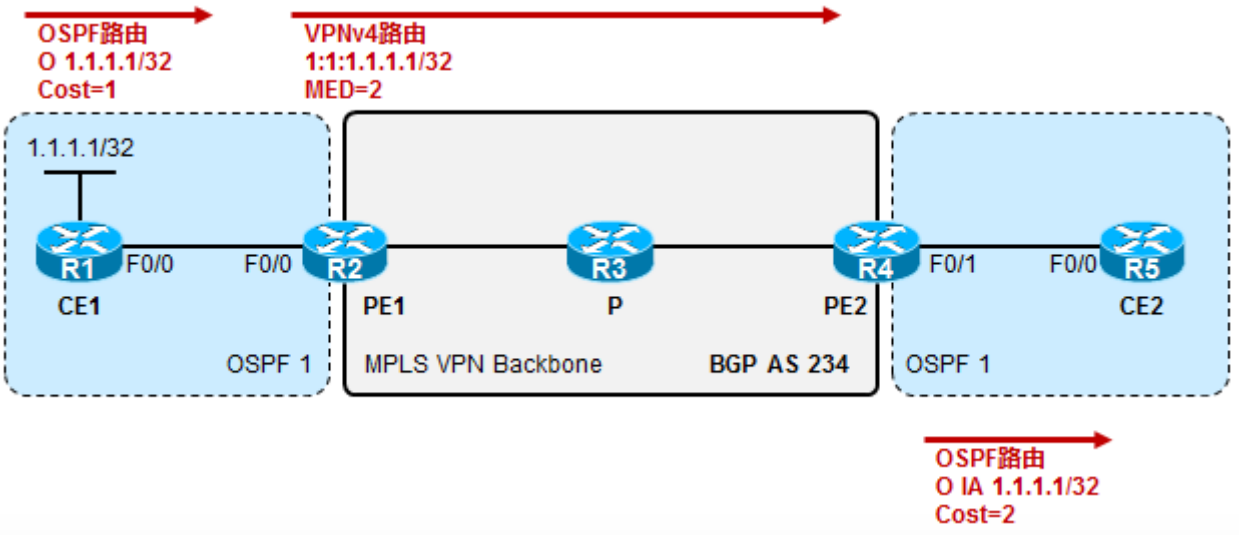
– Routes from Area 0 at one site appear as interarea routes in Area 0 at another site

4.3.4 OSPF metric 传递

1. PE1 及 PE2 的 VRF OSPF 的进程号相同的情况下，OSPF 内部路由的 metric 传递：

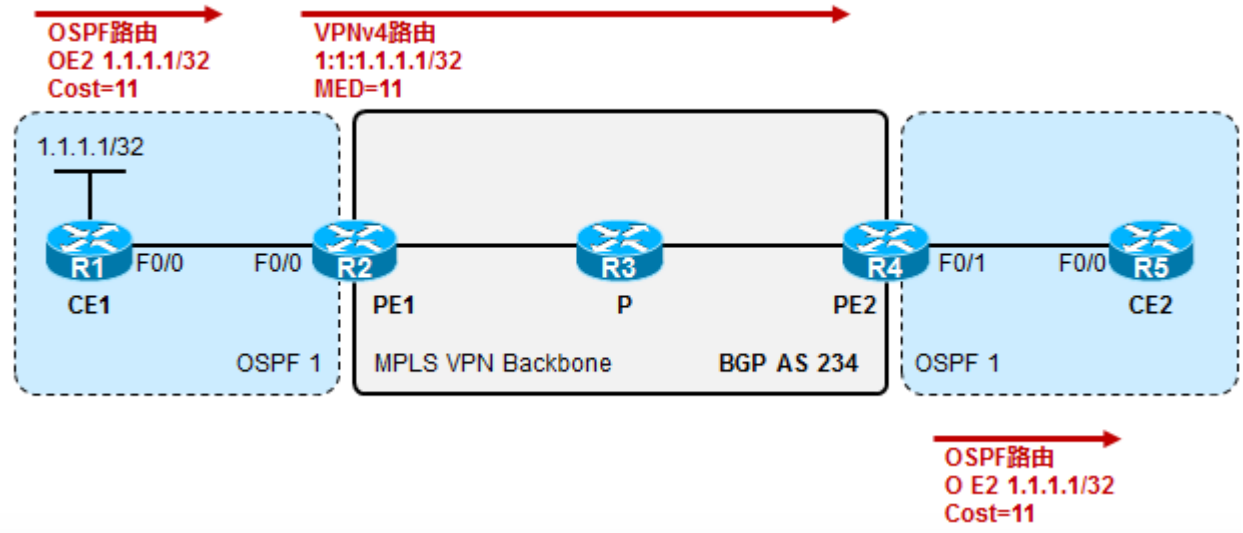


在 PE 路由器上将 OSPF 内部和外部路由重发布进 BGP 的时候，PE 路由将使用 OSPFmetric 来设置 BGP MED。例如上图，CE1 传了一条 OSPF 路由过来（实际上是 LSA），携带的 cost=1。那么 R2 通过 VRF 的 OSPF 进程学习到之后，在累加上本地接口的 cost（为 1）后，最终在 VRF 路由表中的 cost 为 2。现在 R2 也就是 PE1 将 VRF 路由表中 OSPF 的路由注入到 BGP 后，路由的 cost 将会拷贝到 BGP 路由的 MED 上。然后传递出去。



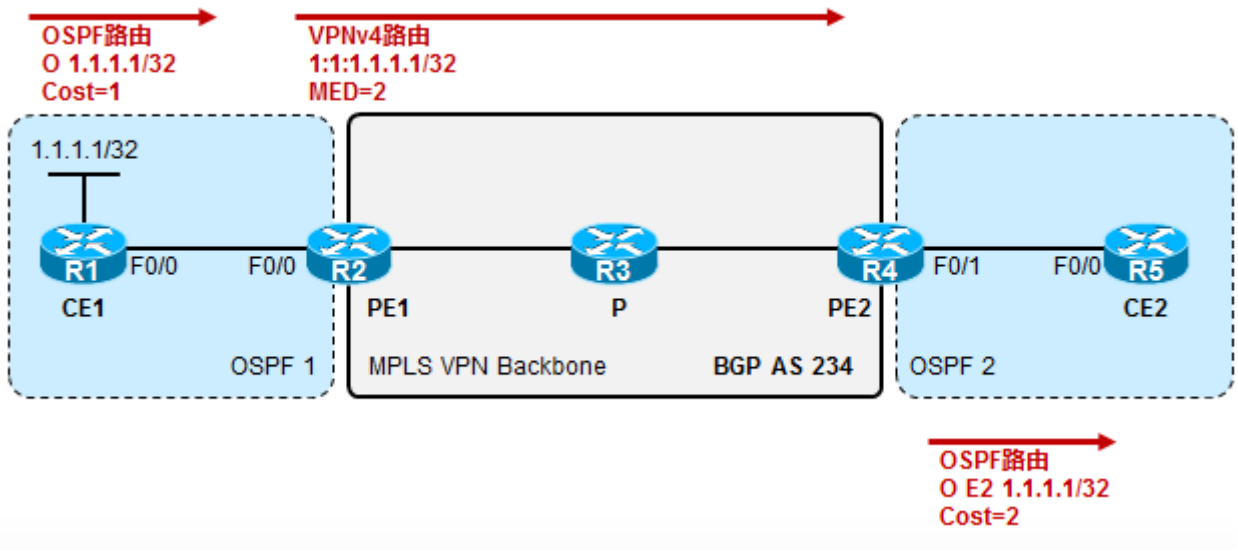
而对端 PE2 那里，要将 BGP 路由注入回 OSPF，那么产生的 OSPF 路由为 3LSA，metric 也是从 BGP 的 MED 中拷贝。注意，如果这里过来的 BGP 路由没有携带 MED 值，那么 OSPF 将使用默认的种子 metric。

2. PE1 及 PE2 的 VRF OSPF 的进程号相同的情况下，OSPF 外部路由的 metric 传递：



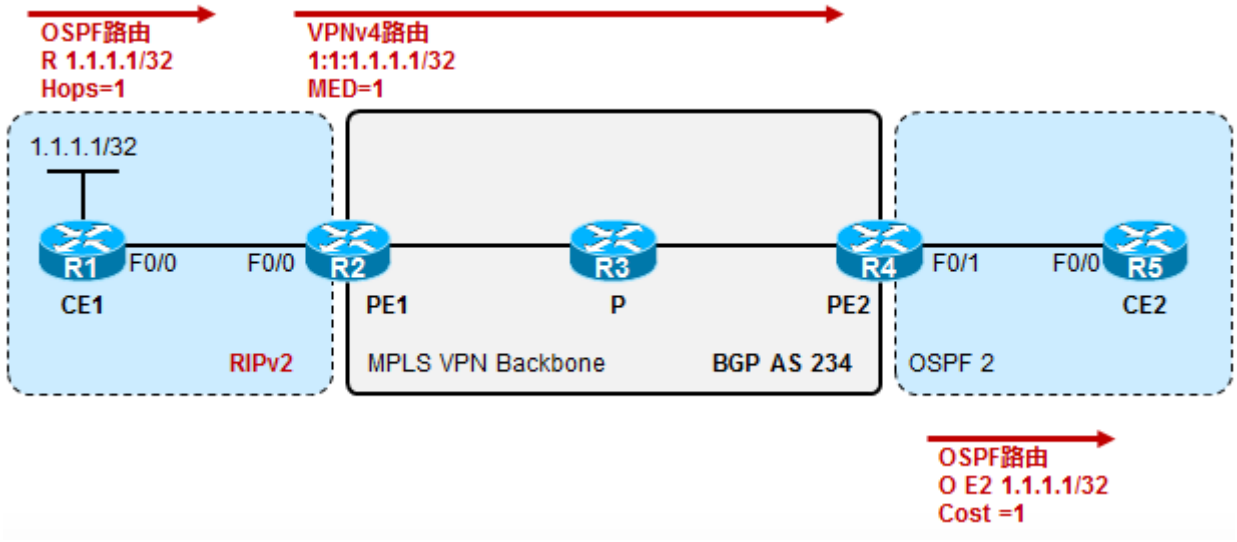
R1 始发的 OSPF 路由 1.1.1.1/32 是一条 OE2 的外部路由，那么这条路由在 PE1 学习到后在 VRF 路由表里的 metric 还是 11。现在 PE1 将 OSPF 路由重发布到 BGP，11 这个 cost 将会被拷贝到 VPNv4 前缀 1:1:1.1.1.1/32 的 MED 属性中。路由被传递到了 PE2，那么 PE2 将 BGP 路由重发布到 OSPF 形成了 OE2 的外部路由 1.1.1.1/32，这条路由的 metric 仍然拷贝 BGP 的 MED。

3. PE1 及 PE2 的 VRF OSPF 的进程号不同的情况下，OSPF 内部路由的 metric 传递：



注意，这里 PE1 及 PE2 的 VRF OSPF 进程 ID 不一致。那么 CE1 上始发的 OSPF 内部路由，携带 cost=1，到了 PE1 被注入到 BGP 后，MED 属性拷贝 cost 值 2，然后传递给 PE2。到了 PE2，BGP 重发布到 OSPF 后，由于 DOMAIN-ID 不匹配因此形成外部路由 1.1.1.1/32，metric 同样拷贝 BGP 路由的 MED 属性值。

4. 非 OSPF 始发路由的 metric 传递问题



现在 CE1 和 PE1 之间运行的是 RIPv2，那么这条路由 PE1 上学习到之后为 1 跳，重发布 BGP 后，VPNv4 前缀携带的 MED 属性值为 1，这是直接拷贝的 RIP 路由的 metric。 那么 VPNv4 前缀传递到 PE2 后，重发布进 OSPF ,形成 OE2 的外部路由 1.1.1.1/32 ,同样的 ,也是拷贝了 BGP 路由的 MED 属性值 ,所以 cost=1。

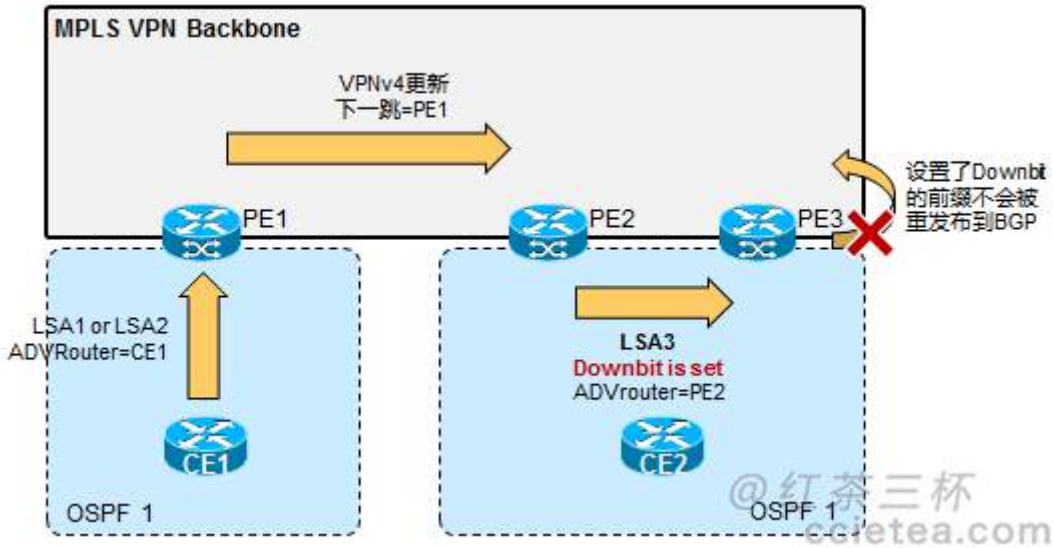
注明：上述所有测试均在(C7200-ADVENTERPRISEK9-M), Version 12.4(24)T 的 IOS 上测试过。

4.3.5 Down Bit

1. Down Bit 概述

- An additional bit – down bit has been introduced in the Options field of the OSPF LSA header
- PE routers set the down bit when redistributing routes from MP-BGP into OSPF
- PE routers never redistribute OSPF routes with the down bit set into MP-BGP

2. Down Bit 技术背景

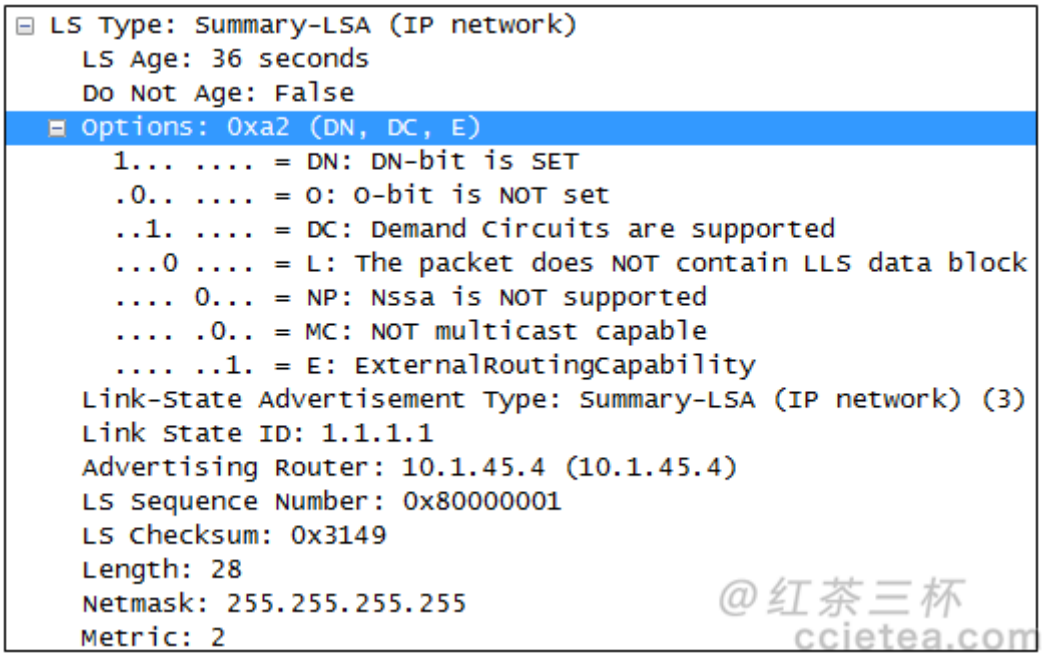


在上图中，右侧的 VPN 站点有两个 PE：PE2 及 PE3。

当 PE2 通过已经建立好的 MP-iBGP 连接从 PE1 学习到 CE1 客户路由 ,它将路由重发布到本地的 OSPF 进程中，这样 CE2 就能学习到这些客户路由，但是，CE2 有可能会将路由更新给 PE3，由于 OSPF 路由协议的 AD 比 iBGP 的 AD 要小，因此 OSPF 路由就会出现在 PE3 的路由表里（而不是从 PE1 的 MP-iBGP 学习到的 BGP 路由）,而恰恰 PE3 又部署了 BGP 到 OSPF 的双向重发布 ,因此 OSPF 路由又被重发布回 BGP 最终被更新回 PE1。这当然是我们不希望看到的现象。

因此我们有了 DownBit 的设计，在 PE2 将从 PE1 学习过来的那些 BGP 路由注入到本地 VRF 的 OSPF 进程中时，如果是以 3LSA 的形式注入，那么这些 3LSA 会做特殊的置位，**也就是 DOWNbit 置位**，那么当 PE3 收到 PE2 从 BGP 重发布到 OSPF 的 3 类 LSA，发现这些路由都设置了 Down Bit，PE3 在计算路由的时候，将直接忽略这些 LSA，自然这些路由也就不会被 PE3 从 OSPF 再重发布回 BGP。这样就可以起到防环的目的。□

Down Bit 只出现在 3 类 LSA 中 ,在 RFC 2547 – BGP/MPLS VPNs 中定义。Down Bit 在报文中的位置：



怎么 show 出它来呢？show ip ospf database summary 1.1.1.1 举例：

Routing Bit Set on this LSA

LS age: 125

Options: (No TOS-capability, DC, **Downward**)

LS Type: **Summary Links(Network)** **!! LSA3 才携带**

Link State ID: 1.1.1.1 (summary Network Number)

Advertising Router: 10.1.45.4

LS Seq Number: 80000001

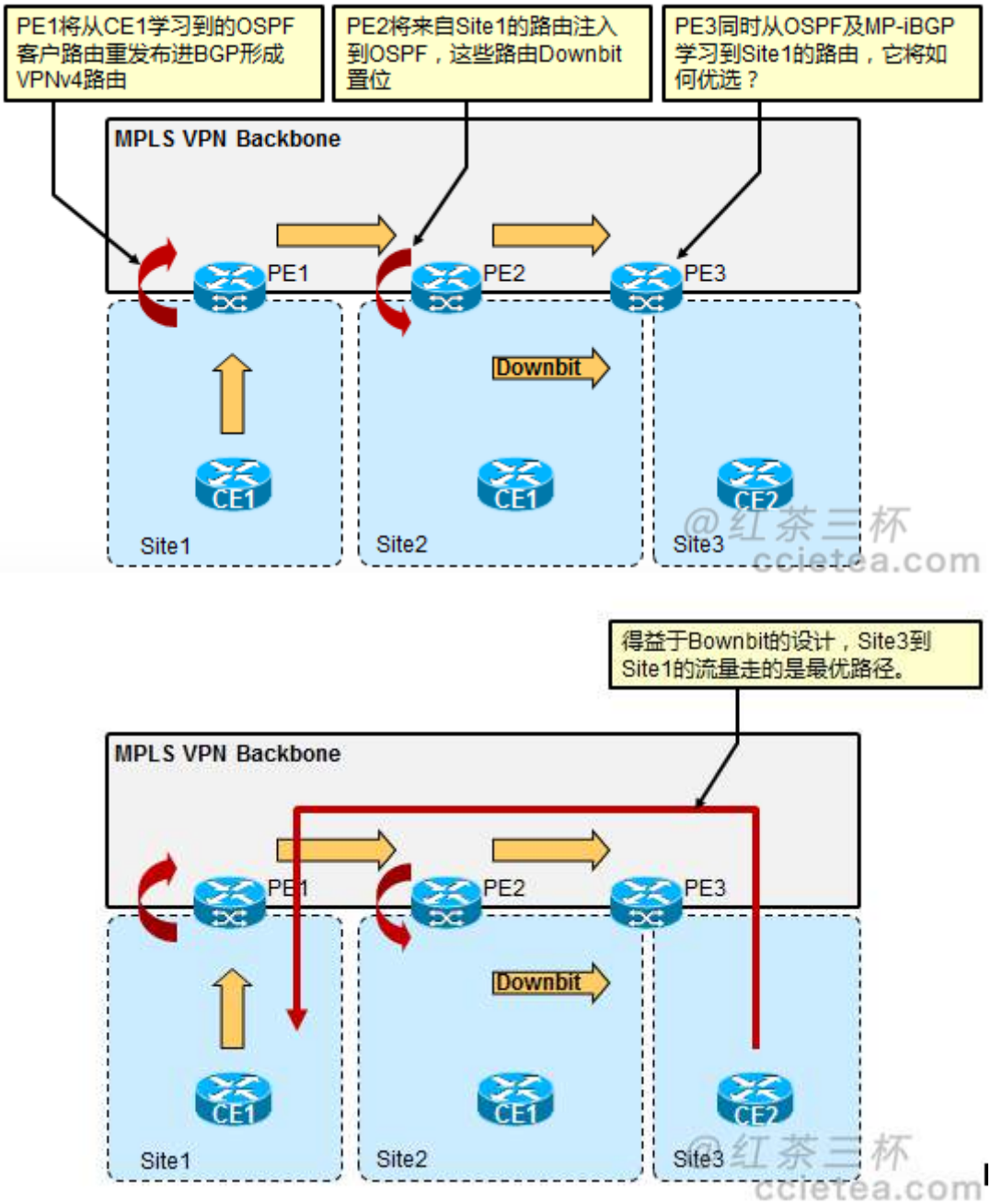
Checksum: 0x3149

Length: 28

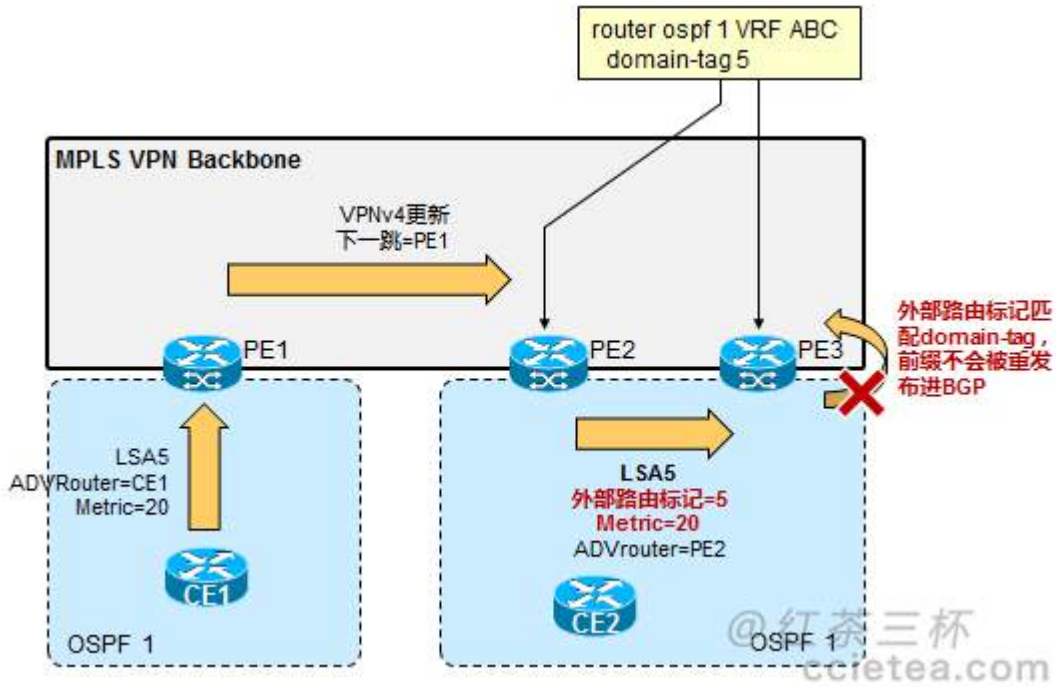
Network Mask: /32

TOS: 0 Metric: 2

3. Down Bit 对网络起到的优化作用：



4.3.6 Domain-tag



Domain-tag 和 Down Bit 功能类似，只不过它是用于 OSPF 外部路由。

如上图，在 PE2 上，将 BGP 路由重发布进 OSPF 后，如果是以外部路由的形式进入 OSPF，则这些路由会被打上 tag（这个 tag 有默认的设置方式，参照 RFC1745，也可以通过 domain-tag 命令在 PE 路由器上手工配置）。

这些路由在 Site 内，如果传递到本 Site 的另一台 PE 路由器 PE3，那么一旦它自己本地设置的 Domain-tag 匹配到了这个路由携带的 tag，这条路由就不会被重发布进 BGP（这些 OSPF 路由不会被装载进 VRF 路由表，自然无法被重发布进 BGP）。

默认情况下，Domain-tag 的设置是根据 RFC1745 来进行的。BGP 的 AS 号码将会在无意义的 16bits 中被编码为 OSPF 外部路由标记。在上图中，PE2 上 BGP 重发布到 OSPF 后，这些路由被打上 tag，tag 中包含 BGP 的 AS 号，路由传递到 PE3 后，PE3 发现这些 OSPF 路由的 tag 与本地的匹配，那么它就只将 LSA 放进 OSPF DATABASE，而不将路由装载进路由表，因此 PE3 上，对于这些远端站点的路由仍然是优选通过 BGP 学习到的（而忽略从 CE 的 OSPF 学到的）。但是如果在 PE2 上，做 BGP 向 OSPF 路由重发布的时候手工去改个 tag，使得与 PE3 的 tag 不匹配，那么 PE3 同时从 OSPF 及 BGP 学习到这些外部路由，并且 TAG 不匹配，那么 PE3 会优选 OSPF 路由（AD 值使然）。

例如：

```
R5#sh ip ro 1.1.2.0
Routing entry for 1.1.2.0/24
  Known via "ospf 1", distance 110, metric 20
  Tag Complete, Path Length == 1, AS 234, , type extern 2, forward metric 1
```

```
Last update from 10.1.45.4 on FastEthernet0/0, 00:00:00 ago
Routing Descriptor Blocks:
* 10.1.45.4, from 44.44.44.44, 00:00:00 ago, via FastEthernet0/0
    Route metric is 20, traffic share count is 1
    Route tag 3489661162
```

这个 tag**3489661162**，将它转换成 2 进制：11010000000000000**0000000011101010，最后这 16 个 bits，再转成 10 进制，就是 BGP 的 AS 号，234.**

TAG 值在 OSPF 域之间传递。也就是在不同的 OSPF 域间传递，不会丢失，这个特性非常有用。

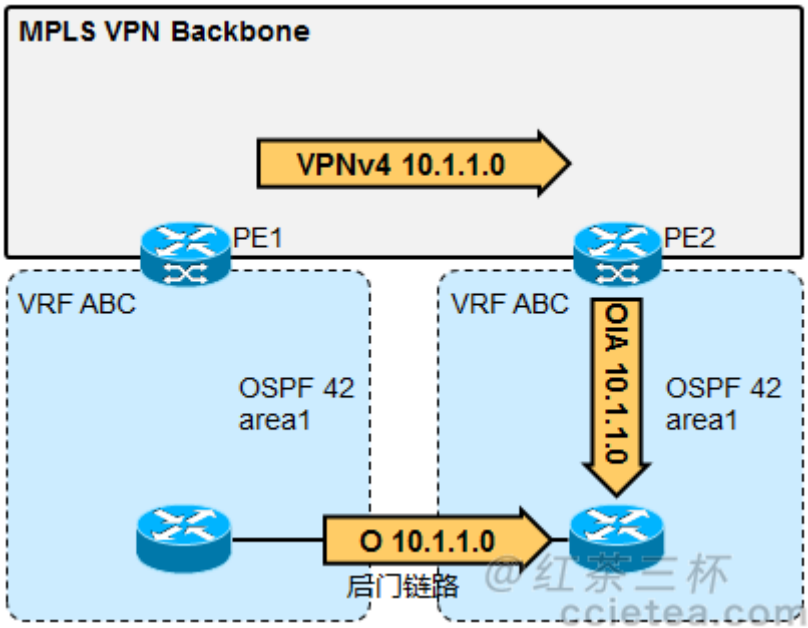
Tag 值也可以手工设置：redistribute … tag ?这样路由被重发布进 OSPF 后，tag 值就是这个手工设置的值。

如果要修改本地 OSPF 进程的 Domain-tag，则：

```
R4(config)#router os 1 vrf ABC
R4(config-router)#domain-tag ?
  <1-4294967295>  OSPF domain tag - 32-bit value
```

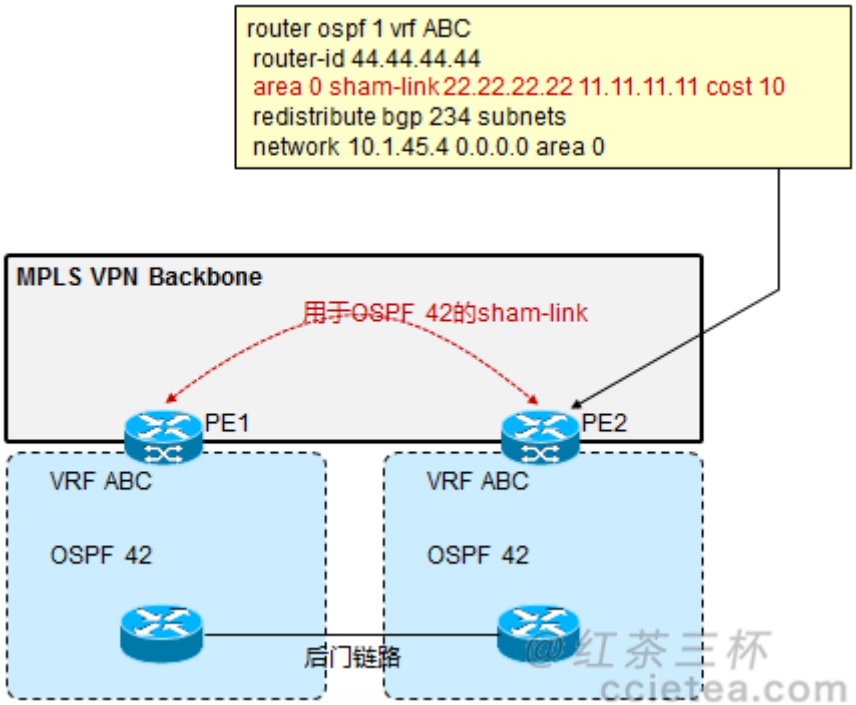
4.3.7 Sham-link

1. 技术概述

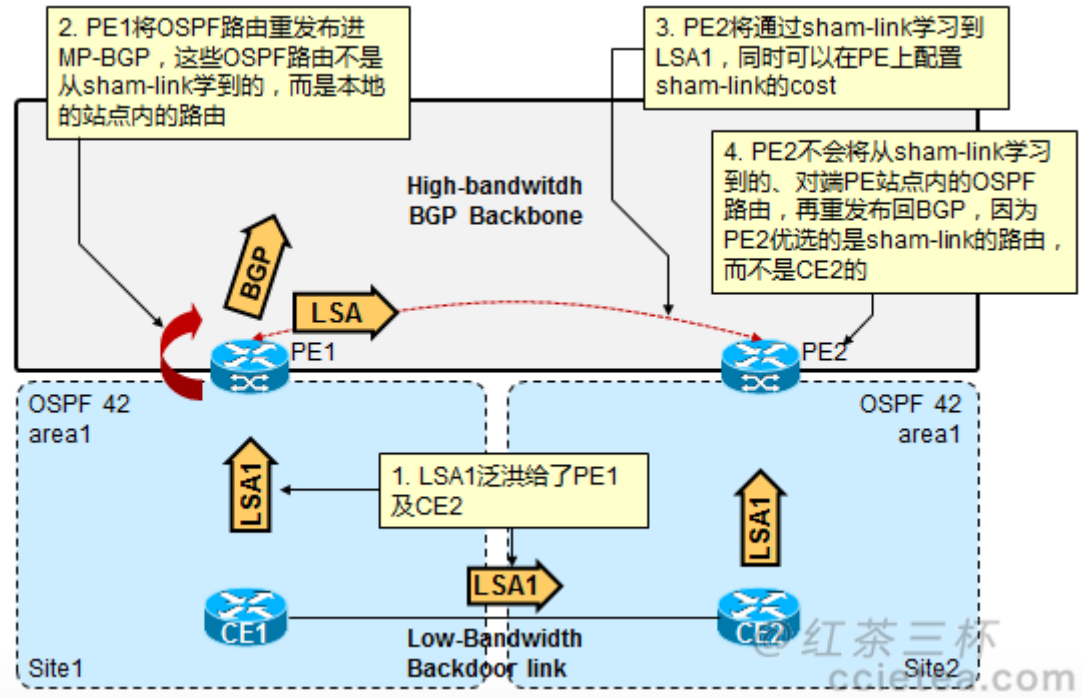


如果在两个站点的 CE 之间增加一条后门链路，并且直接运行 OSPF 交互路由，那么通过这条后门链路学习到的内部路由为 O，通过 PE 过来的路由为 OIA，无疑优选 O 路由，那么如果我们希望在这个环境中，

流量能从 MPLS VPN Backbone 走怎么办？



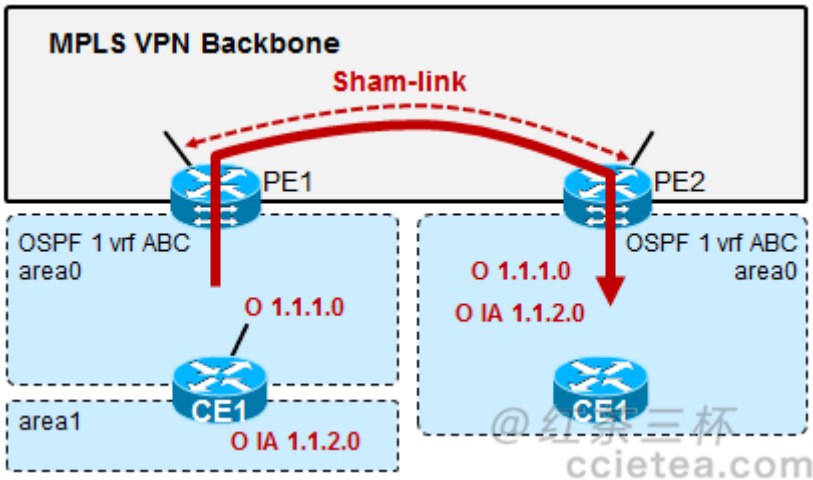
可以考虑在 PE1 及 PE2 之间建立 sham-link，它在两台 PE 之间创建一条区域内链路。在每一台 PE 上的链路端点都是特定 VRF 中的一个接口（或地址），iBGP 必须以 VPNv4 前缀的形式将这个接口的路由传递给另一台 PE（也就是说我们必须在 MP-BGP 的 ipv4 vrf 地址族里宣告这个地址，以便形成 VPNv4 前缀并传递给对端 PE）。Sham-link 和其他 OSPF 中的链路一样，会进行最短路径优先的计算，当 LSA 在伪装链路中泛洪，所有的 OSPF 路由类型都不会改变，不会转换成 LSA3 或者 LSA5 的类型。如果 sham-link 断掉，那么又恢复成前面所讲的机制。



2. 注意事项：

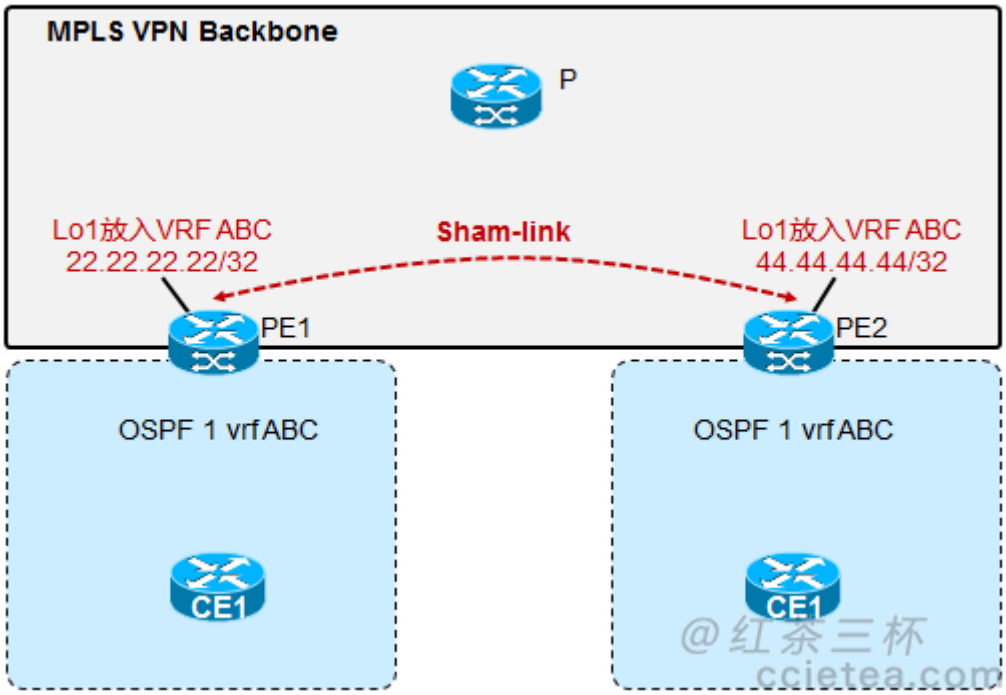
- 一定要在 iBGP 中通告用于 sham-link 的端点 IP ,而不能用 core 的 OSPF ,否则 sham-link 会发生抖动。因为首先 iBGP 应该能够学习到 sham-link 端点的路由，然后才能创建 sham-link。如果这时候，在 core 的 OSPF 中又通告了端点的路由，由于 OSPF 的 AD 小于 IBGP，因此它将覆盖掉 iBGP 路由，一旦端点路由不能通过路由表中的 iBGP 学到，那么伪装链路就会断开。然后再次开始重新建立，这样就导致 sham-link 不断抖动。
- 用于 sham-link 的 IP（所属的接口）必须分配给特定的 VRF

3. 一些细节：



在 PE1 及 PE2 之间建立 sham-link，则 PE1 这边，原本是 O 的路由，经过 sham-link 传递到 PE2 这边，还是 O，而原本是 OIA 的，经 sham-link 传递到右边还是 OIA。

4. 配置如下：



R2（PE1）的配置如下：

```
ip vrf ABC
 rd 1:1
 route-target export 234:2
 route-target import 234:4
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
interface Loopback1
 ip vrf forwarding ABC                          !!这个接口用于建立 sham-link，需要放入 VRF
 ip address 22.22.22.22 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface FastEthernet0/0
 ip vrf forwarding ABC
 ip address 10.1.12.2 255.255.255.0
!
```

```
interface FastEthernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
!
router bgp 234
 no bgp default ipv4-unicast
 neighbor 4.4.4.4 remote-as 234
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family vpnv4
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf ABC
  redistribute ospf 1 vrf ABC match internal external 1 external 2
  network 22.22.22.22 mask 255.255.255.255          !!要在 BGP 的 ipv4 vrf ABC 地址族中宣告 LO1
 exit-address-family
!
router ospf 1 vrf ABC
 area 0 sham-link 22.22.22.22 44.44.44.44 cost 10      !!配置 sham-link
 redistribute bgp 234 subnets
 network 10.1.12.2 0.0.0.0 area 0
!
```

PE2 的配置大同小异，这里不再赘述。

在 R2 上看一下：

R2#sh ip os sham-links

```
Sham Link OSPF_SL0 to address 44.44.44.44 is up
Area 0 source address 22.22.22.22
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 10 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:01
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
Last retransmission scan time is 0 msec, maximum is 0 msec
```

4.4 使用 EIGRP

4.4.1 概述

- PE 路由器在做 EIGRP 到 BGP 的重发布的时候，可以通过诸如 BGP 扩展 community 属性等的支持来保存 EIGRP 路由的部分内容，包括 metric、AS、TAG、以及对外部路由而言的远程 AS 号、远程 ID、远程协议和远程度量值。这些都是可以在 EIGRP 的拓扑表中找到的 EIGRP 特征。
- 如果 EIGRP 所通告的路由是内部路由，并且 BGP 扩展 community 属性中所携带的源 AS 号码能够匹配到目的 AS 号的话，这条路由将会以内部路由的形式通告给远端站点。如果 AS 号不匹配，则这条路由会被重建为一条 EIGRP 外部路由。
- 用于 EIGRP 的 BGP 扩展 community 属性：

类型	用法	值
0x8800	通用路由信息	标记+TAG
0x8801	路由度量值信息和自治系统	自治系统+延迟
0x8802	路由度量值信息	可靠性+跳数+带宽
0x8803	路由度量值信息	保留字段+负载+MTU
0x8804	外部路由信息	远程自治系统+远程 ID
0x8805	外部路由信息	远程协议+远程度量值

来看一条 EIGRP 内部路由在 PE 上重发布进 BGP 后

PE2#sh ip b vpnv4 all 1.1.1.0 （内部路由）

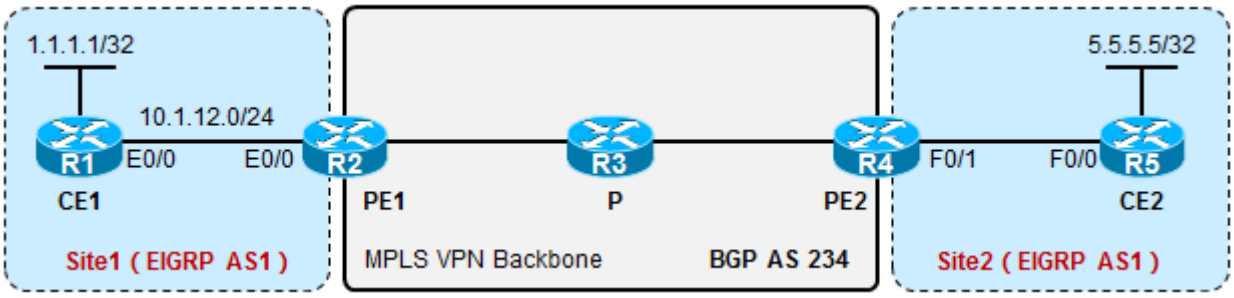
```
BGP routing table entry for 234:2:1.1.1.0/24, version 1489
Paths: (1 available, best #1, table ABC)
Flag: 0x820
  Not advertised to any peer
  Local, imported path from 234:1:1.1.1.0/24
    2.2.2.2 (metric 3) from 2.2.2.2 (2.2.2.2)
      Origin incomplete, metric 156160, localpref 100, valid, internal, best
      Extended Community: RT:234:1 Cost:pre-bestpath:128:156160
        0x8800:32768:0 0x8801:1:130560 0x8802:65281:25600 0x8803:65281:1500
      mpls labels in/out nolabel/21
```

PE2#sh ip b vpnv4 all 1.1.2.0 (外部路由)

```
BGP routing table entry for 234:1:1.1.2.0/24, version 1486
Paths: (1 available, best #1, table ABC)
Flag: 0x820
  Not advertised to any peer
  Local
    2.2.2.2 (metric 3) from 2.2.2.2 (2.2.2.2)
      Origin incomplete, metric 261120, localpref 100, valid, internal, best
      Extended Community: RT:234:1 Cost:pre-bestpath:129:261120 0x8800:0:0
        0x8801:1:5120 0x8802:65281:256000 0x8803:65281:1500 0x8804:0:16843009
        0x8805:11:0
      mpls labels in/out nolabel/19
```

这些扩展的 community 值能够很好的保存 EIGRP 路由的特性，使得 EIGRP 路由在从一个站点经过 MPLS VPN Backbone 传输到另一个站点后这些路由能够在远端 PE 上被很好的还原，这句话哥讲的太特么有深度了，具体怎么个有深度法大家看本文档的“MPLS VPN 双 PE 防环”这一章。

4.4.2 配置



R1（CE）的配置

```
router eigrp 1
  no auto-summary
  network 10.0.0.0
  network 1.0.0.0
```

R2（PE1）的配置

```
router eigrp 1
  no auto-summary
  address-family ipv4 vrf ABC          !!在地址族下配置
    network 10.1.12.0 0.0.0.255
    no auto-summary
    autonomous-system 1                !! eigrp 的 as 号，必须和 CE 上的匹配
    redistribute bgp 234
  exit-address-family
```

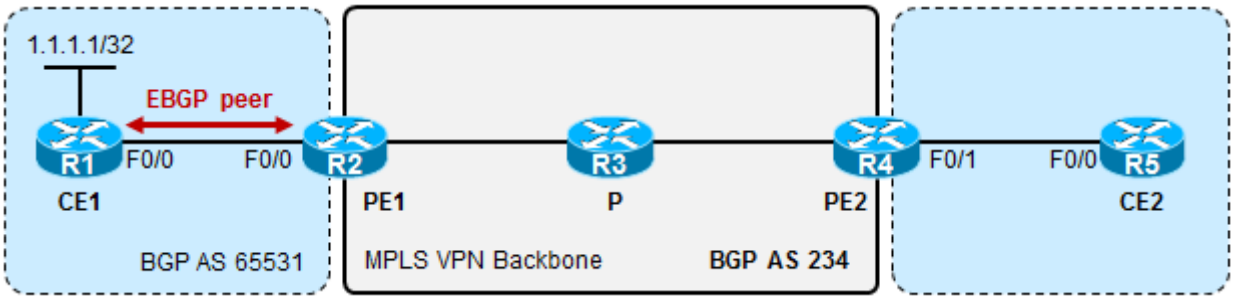
注意，EIGRP VRF 的配置也是在 ipv4 vrf 地址族中进行的。

4.4.3 SOO

请见“MPLS VPN 的高级部分”这一章节

4.5 使用 EBGP

4.5.1 配置



1. **实验环境**
   - AS65531 内 CE1 的 F0/0 接口 IP 为 10.1.12.1/24；PE1 的 F0/0 为 10.1.12.2/24
   - 我们重点看 CE1 及 PE1 的配置，其他设备的配置在这里不是重点，但是大家还是要有个全局观

2. **设备配置**

R1(CE)的配置：

```
router bgp 65531
  neighbor 10.1.12.2 remote 234
  network 1.1.1.1 mask 255.255.255.255          !! 通告客户路由
```

R2(PE)的配置：

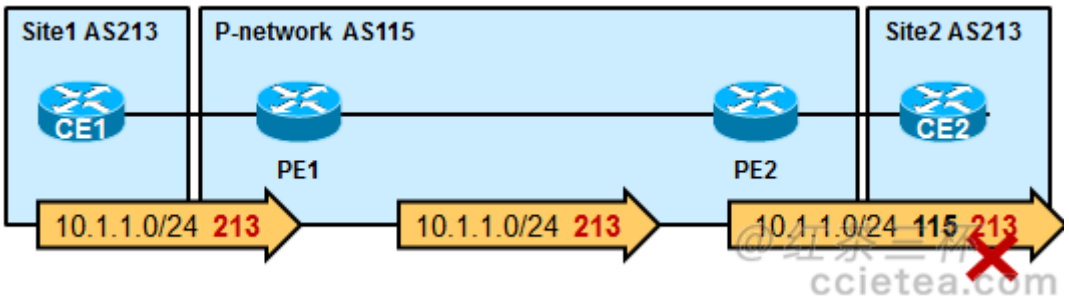
```
ip vrf ABC
  rd 1:1
  route-target export 234:2
  route-target import 234:4
.....
router bgp 234
  address-family ipv4 vrf ABC
    neighbor 10.1.12.1 remote-as 65531
    neighbor 10.1.12.1 activate
```

注意指向 CE 的 neighbor 指令不能在全局 BGP 进程里去指，因为 F0/0 这个接口，是在 vrf ABC 里的，全局路由表里找不到该直连网段。

但 vpnv4 的邻居，需要在 BGP 主进程中先指 neighbor，再去 vpnv4 地址族中激活。

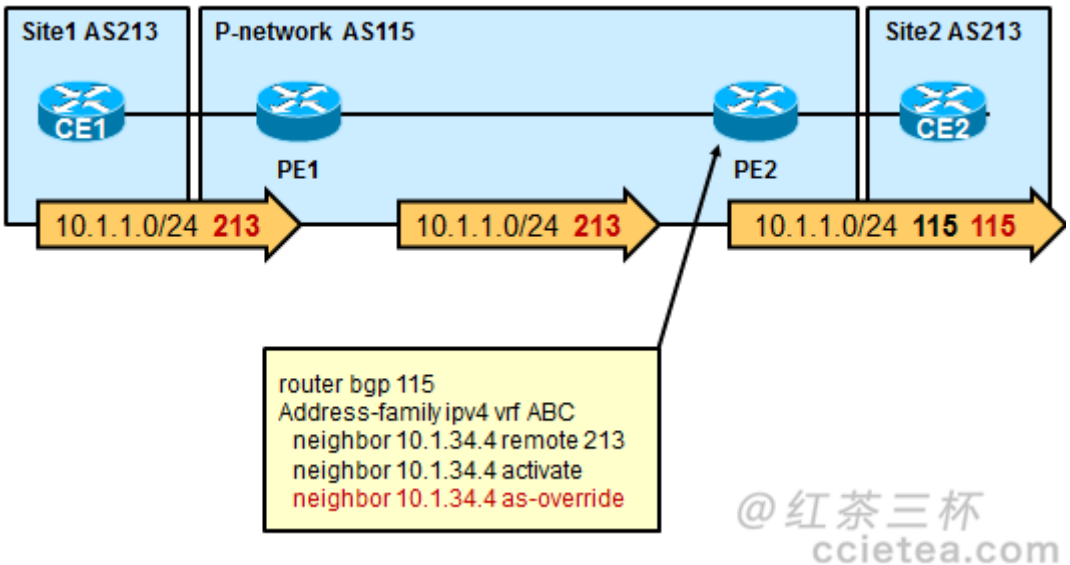
使用 show ip bgp vpnv4 vrf ABC summary 查看 vrf BGP 邻居是否建立。

4.5.2 AS-Override

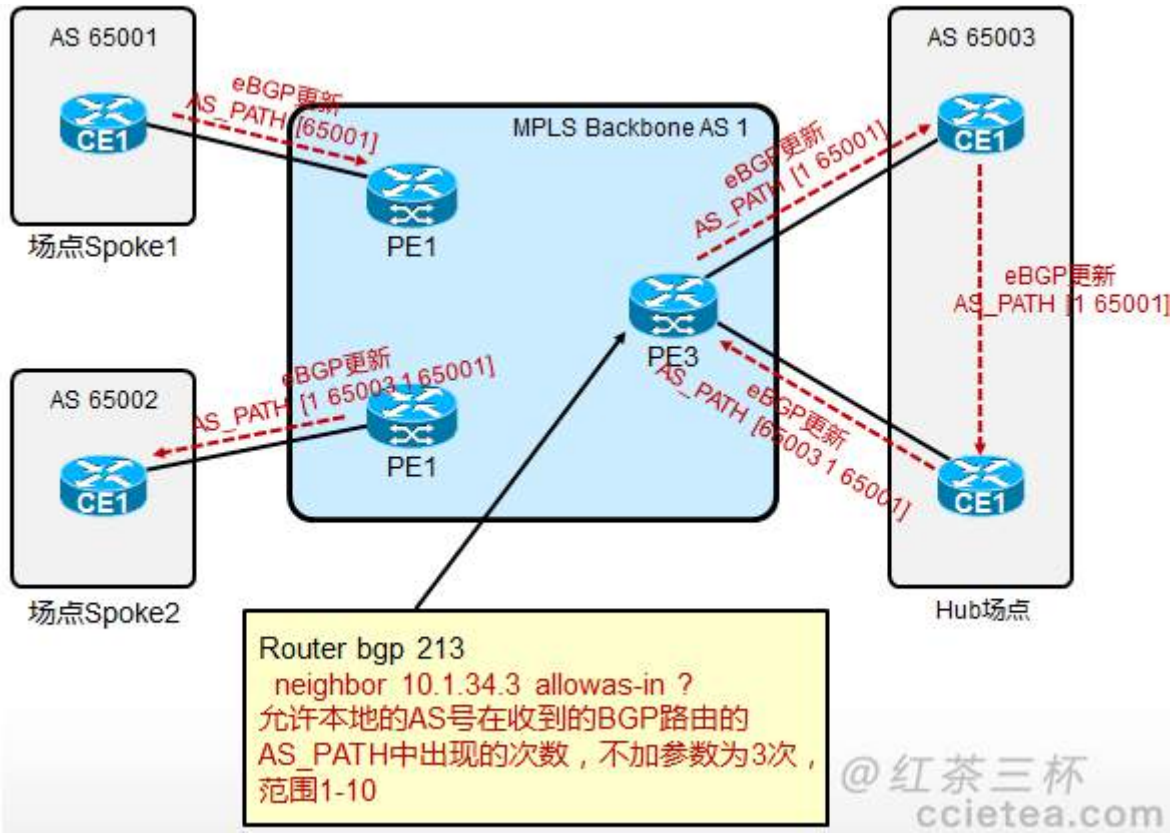


CE 与 PE 之间运行 EBGP，CE1 及 CE2 使用相同的 AS 号，那么由于 AS_PATH 中出现了自己的 AS 号，因此双方都会忽略源自对方的路由更新。

- **当实施了 AS-Override**
  - 如果 AS_PATH 中的 AS 号与该 PE 的 CE（EBGP 对等体）的 AS 号相同，则将该 AS 号替换成 Provider 的 AS 号
  - 如果这个 AS 号在 AS_PATH 中重复出现了（可能上游用了 as-path prepend），那么所有的重复项都会被一一替换
  - 在上述替换动作完成后，Provider 的 AS 号再插入到 AS_PATH 中



4.5.3 Allowas-in

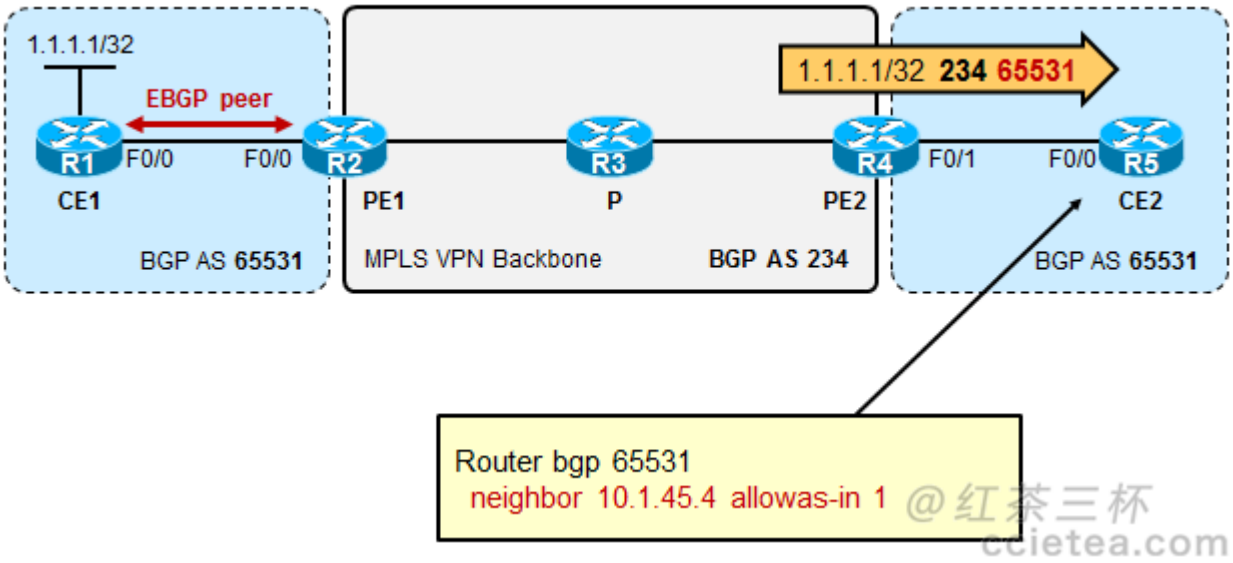


注意这是个 hub&spoke 环境，客户要求 spoke 之间的通信，需通过 hub 场点。那么这样一来 Spoke 的场点路由就需要先经过 MPLS Backbone AS 1 然后传递到 Hub，然后再从 Hub 传回 Backbone AS1 再到另一个 Spoke 场点。

那么这样一来，对于 PE3 来说，就有问题了，有可能在路由的 AS-PATH 里看到自己的 AS。

那么就可以用到 allows-in 特性在 PE3 上部署。

配置示例：



5 MPLS VPN 高级部分

5.1 限制 VRF 中的路由条目

服务提供商运营 MPLS VPN 业务，如果 PE-CE 之间采用 BGP 连接，由于 BGP 往往用于承载大量的路由前缀信息，因此实际上 PE 设备承担着一定的风险，例如 CE 网络中的攻击行为，或者路由条目过多导致的 PE 设备过载等等。因此，需要针对特定客户所分配的资源进行限制

CISCO IOS 提供了两种办法：

- 限制从 BGP 邻居收到的路由前缀条目数量
- 限制 VRF 中的路由条目数量

1. 限制从 BGP 邻居收到的路由前缀条目数量

```
router bgp 2345
    address-family ipv4 vrf ABC
    neighbor 5.5.5.5 maximum-prefix   ?
```

neighbor maximum-prefix xx

限制从邻居接受的前缀最大数 ,如果超出了这个数 ,路由器就会关闭与该邻居的 BGP 连接 ,在应用 clear ip bgp xxx 之前都不会再次建立 BGP 会话

neighbor maximum-prefix xx restart y

超过 xx 这个数，断开与邻居的 BGP 连接，y 分钟后才能重新连接

Neighbor maximum-prefix 300 90% warning-only

当从邻居接受的前缀数量超过了最大数 300 的 90%时（默认 75%），生成一条日志消息

2. 限制 VRF 中的路由条目数量

```
Ip vrf ABC
    maximum ?
```

注意，VRF 路由表的路由学习来源有两个，分别是从 CE（PE-CE 之间的 IGP），和 VPNv4 也就是 MP-BGP，那么这个特性的限制将对这两个途径学习到的路由都有效。

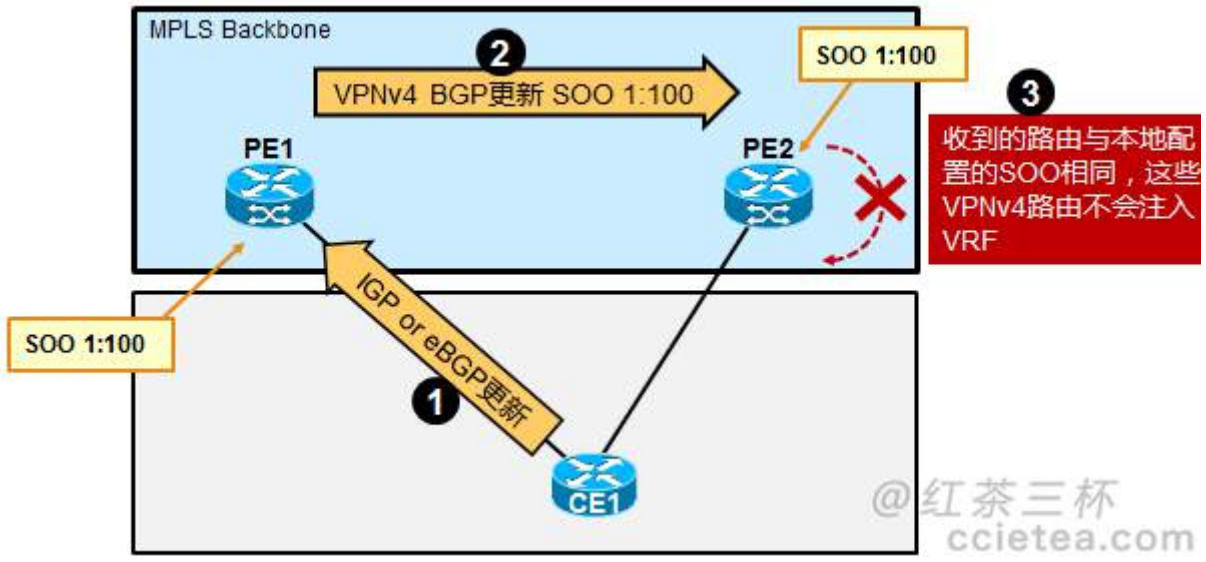
5.2 SOO

1. 关于 SOO

- **Site of Origin 唯一标识路由起源的场点**
- **as-override 和 allowas-in 这两特性实际上是放宽或者说突破了 BGP 基于 AS_PATH 的防环机制**
- **SOO（BGP 扩展 community 属性）可以用来在上述情况下避免环路**

  SOO 在多宿环境下是需要的，例如下图所示的一台 CE 路由器，连接到两台 PE，这叫双宿。

  在单宿的环境中，例如 CE 只连接到一台 PE 上，这种情况下 SOO 就不大需要了。



- When EBGP is run between PE and CE routers,SOO is configured through a route map command.
- For other routing protocols, SOO can be applied to routes learned through a particular VRF interface during the redistribution into BGP.

2. **SOO 的配置**

- **SOO 的配置（PE-CE 运行 EBGP）**

  如果 CE-PE 间运行的是 EBGP，那么可以直接使用 route-map，然后在 neighbor 的时候进行关联

```
route-map test permit 10
  set extcommunity soo 1:100
router bgp 1
  address-family ipv4 vrf ABC
    neighbor 10.10.2.1 remote-as 1
    neighbor 10.10.2.1 route-map test in
```

- **SOO 的配置（PE-CE 运行非 BGP）**

  这个 route-map 就应该配置在恰当的 VRF interface 的 ip vrf sitemap 命令中

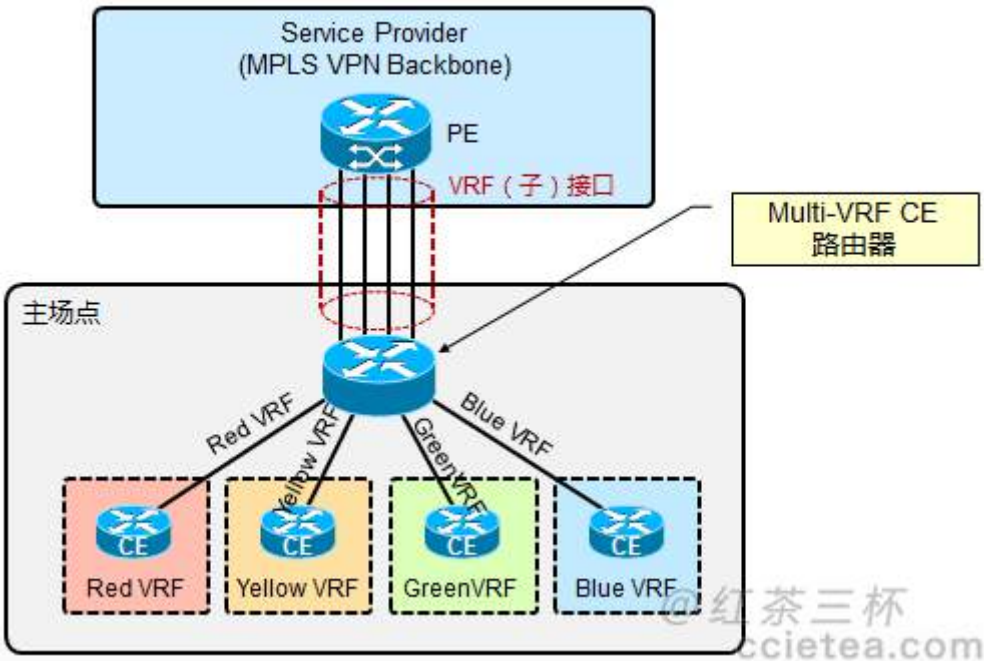
```
Interface fast0/0
  Ip vrf sitemap test
```

  当然，还可以在直连和静态路由重发布到 BGP 的时候关联设置 SOO 的 route-map。

5.3 Multi-VRF 的 CE

Multi-VRF CE 特性也被称为 VRF-lite，这种特性是将 VPN 功能扩展到 CE 路由器上的一种廉价的方法。

- Multi-VRF 使得我们即使在没有 MPLS 的情况下也能实现类似 PE 的功能
- 在单台物理路由器上可以使用 VRF 来实现路由的隔离（形成 vrf 路由表）
- 这样一来一台路由器可以衍生出多台虚拟路由器
- 一个物理接口可以通过子接口的方式从而属于 milti-VRF



上图中，主场点 CE 路由器无需支持 MPLS VPN，通过 VRF 将各部门进行隔离。Multi-VRF CE 路由器上有一个 VRF 接口指向每一个与该路由器相连的 CE 路由器，同时每一个指向 CE 路由器的 VRF 还必须有一个 VRF 接口指向 PE 路由器，可以考虑使用子接口的方式来从而节省成本。

另一个需要注意的问题是，如果 Multi-VRF CE 路由器运行 OSPF 的话，前面我们介绍过对于 3 类 LSA 需检查 downbit，对于外部路由需检查 tag，由于 Multi-VRF CE 拥有类似 PE 的功能，因此它也会做上述两个检查，那么这个时候 ,如果 PE 有路由传递给它 ,由于这些路由来自 PE ,因此可能设置了 downbit 或者 tag ,那么 Multi-VRF CE 就不会将这些路由传递给下面的 CE。因此需在 Multi-VRF CE 上配置 capability vrf-lite，以跳过这两项检查。

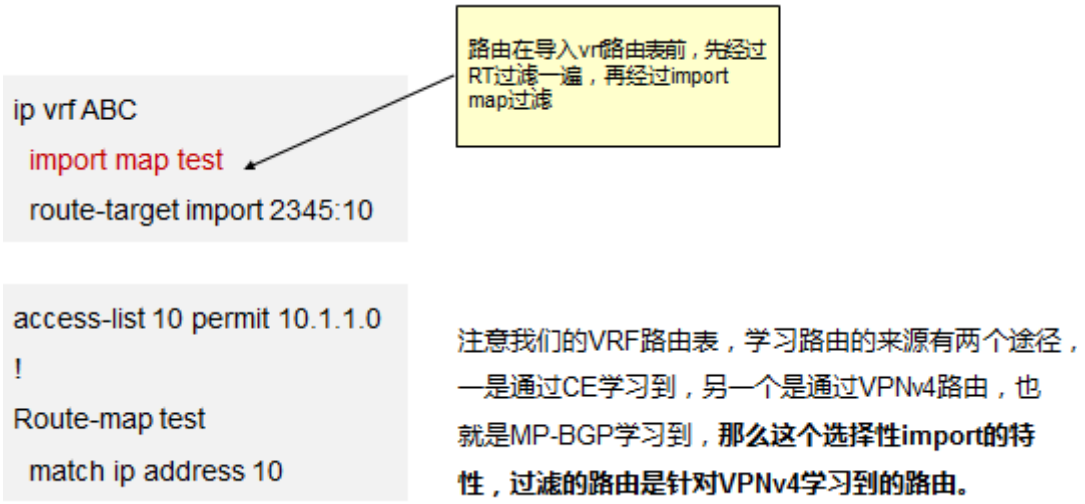
```
Router ospf 1 vrf ABC
  capability vrf-lite
```

5.4 Advanced VRF import/Export Features

- 选择性 import
- 选择性 export
- VRF route limit

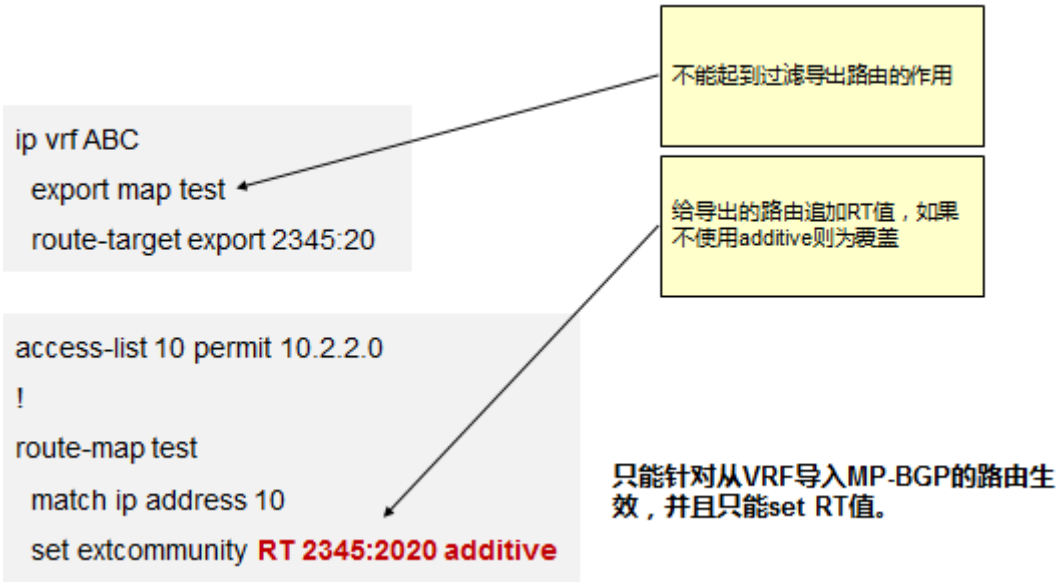
  见 PE-CE 间运行 BGP 协议部分

1. 选择性 import



注意我们的 VRF 路由表，学习路由的来源有两个途径，1 是通过 CE 学习到，另一个是通过 VPNv4 路由，也就是 MP-BGP 学习到，**那么这个选择性 import 的特性，过滤的路由是针对 VPNv4 学习到的路由。**再补充一点 ,VRF 的 route limit 则是针对从 CE 学习到的 VRF 路由以及从 MP-BGP 学习到的 VPNv4 路由共同生效。

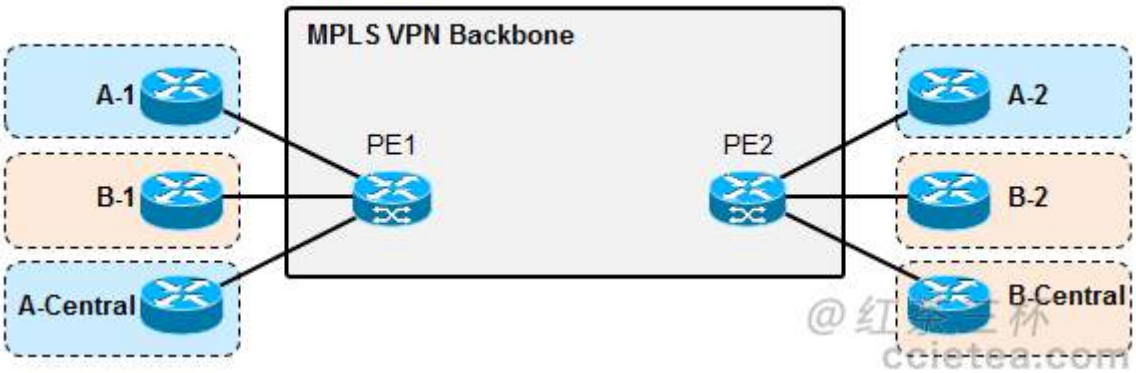
2. 选择性 export



6 Complex VPNs

6.1 Overlapping VPNs

1. **模型概述**

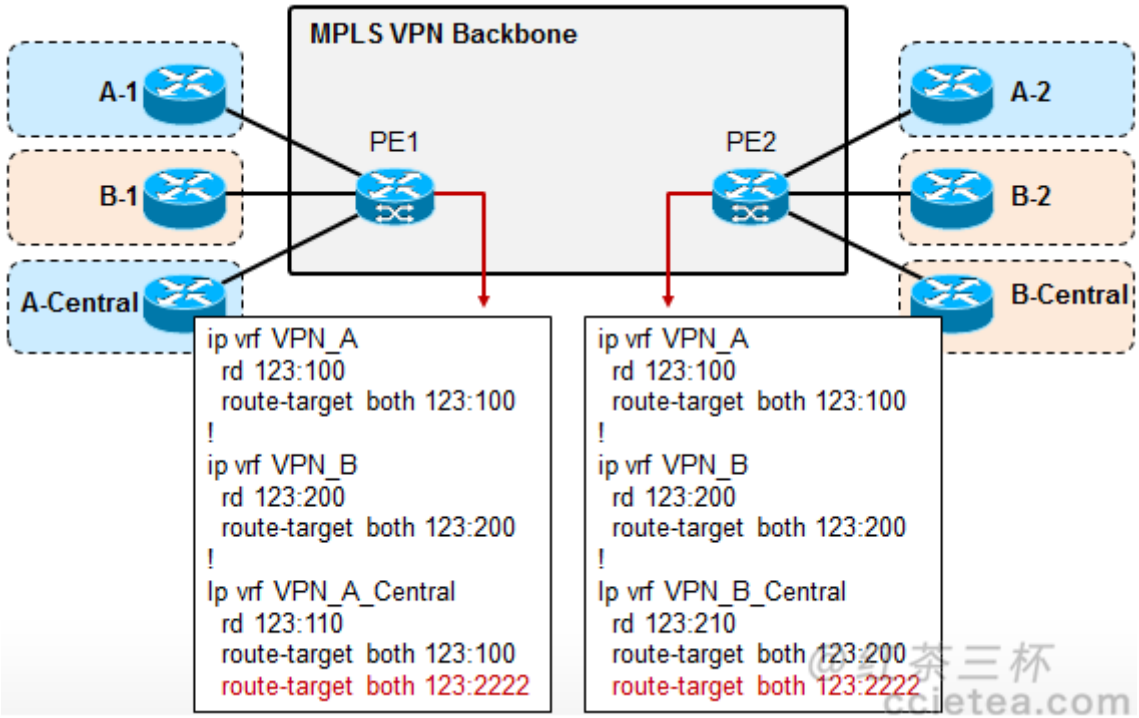


客户 A 有两个站点，1 和 2；客户 B 也有两个站点 1 和 2。

客户 A 及客户 B 各有一个中心站点，访问需求如下：

- VPN-A 的 CE1 需与 CE2 及 Central 通信（VPN 客户 A 内所有站点要同信）
- VPN-B 的 CE1 需与 CE2 及 Central 通信
- VPN-A 的 Central 需与 VPN-B 的 Central 通信

2. **解决办法：**

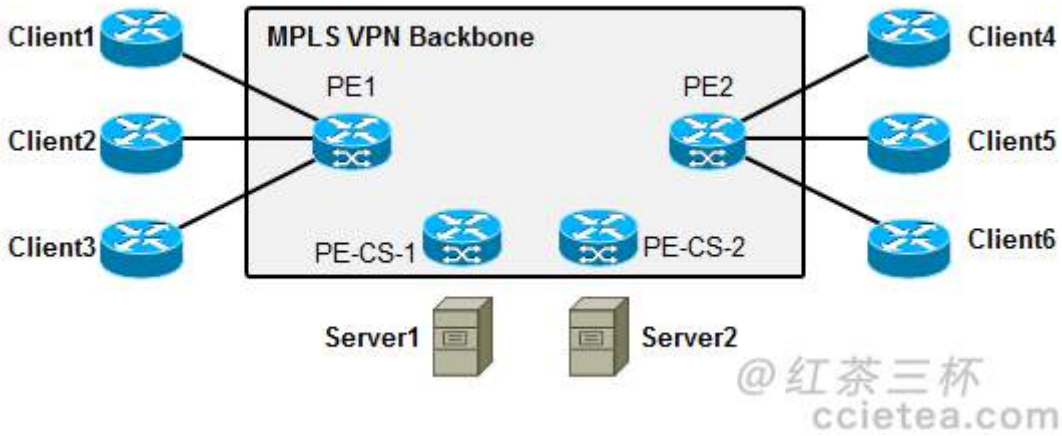


我们看到，首先两个 PE 上，VPN_A 内的用户站点之间要能够互访、VPN_B 内的站点之间要能够互访，那么对于 VPN_A 使用 RT 123:100、对于 VPN_B 使用 123:200。就可以起到 VPN 隔离的作用。

在者，A-Central 要和 B-Central 互访，我们使用另一个 RT，123:2222。并且只在 A-Central 及 B-Central 的 VRF 里 export 和 import。

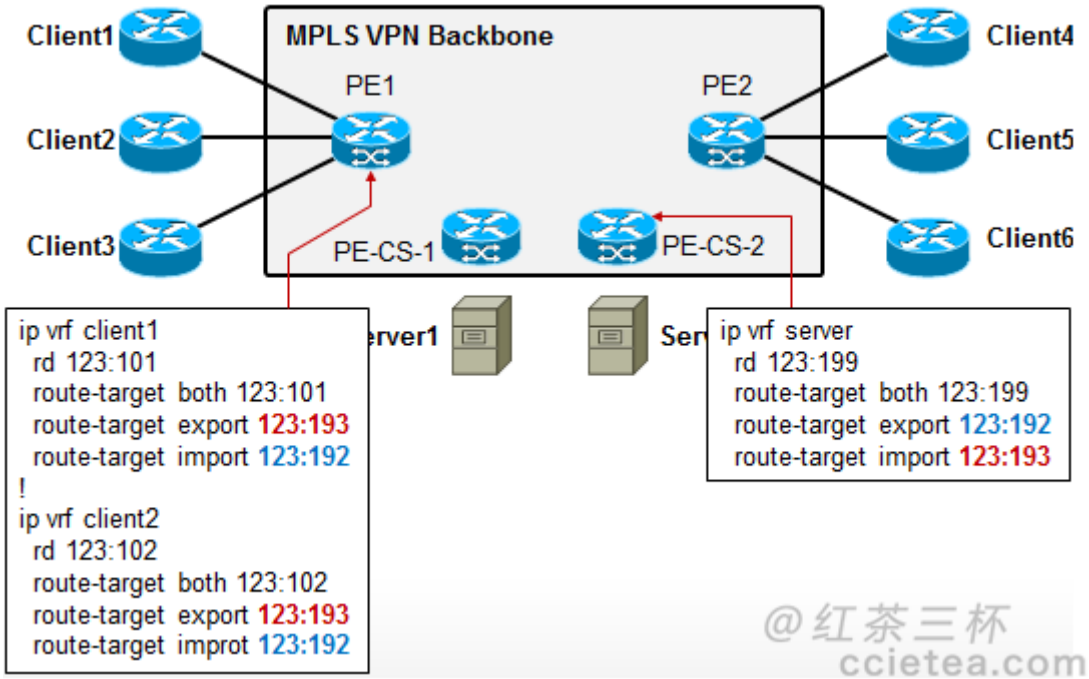
6.2 中心服务 VPN (Central Services VPN)

1. **模型概述**



- Clients 需与 Central 中心的 Server 通信
- Servers 之间也有访问需求
- Clients 之间无法相互访问

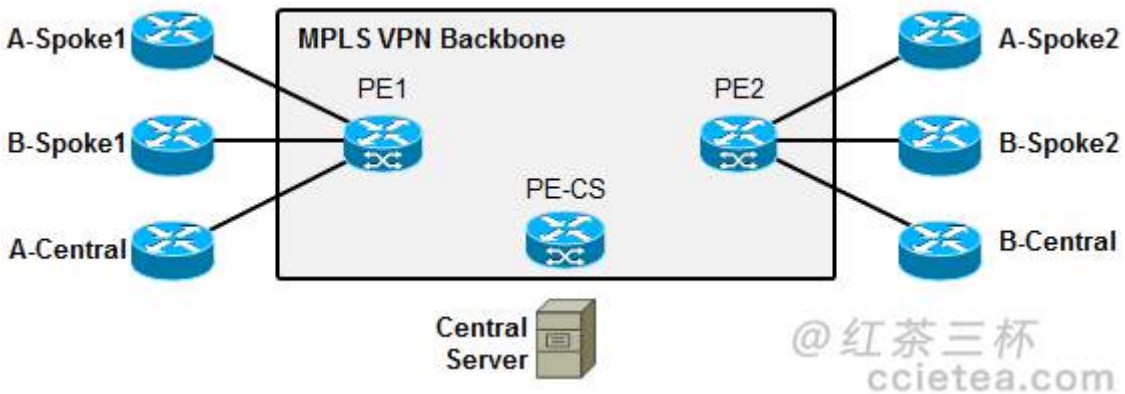
2. 解决办法：



上图中，你看那些个 Client 内，都有一个和 RD 值相同的 RT 值，并且都 import 和 export 了，这个在本环境中没有明显的效果，可能的一个情况是，在 PE1 和 PE2 上，均有如 client1 的用户，那么这个 RT 可以保证分布在 PE1 和 PE2 上的 Client1 的站点之间能够通信。

6.3 Managed CE Routers

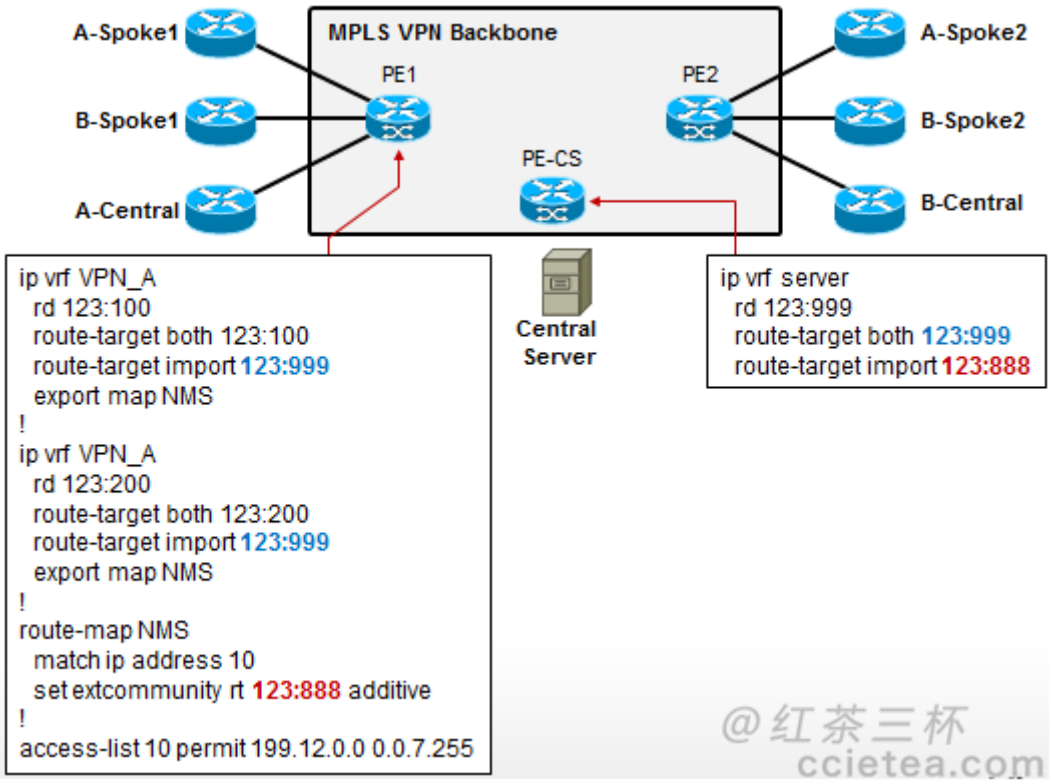
1. 模型概述



中心站点内有一台具备管理权限的服务器，现在，需要让这台管理性服务器能够管理所有 VPN 的 CE 路由器，注意，只是管理 CE 路由器而已，无需访问 CE 路由器所在站点内的其他资源。

因此可以在这些 CE 路由器上配置 Loopback 专门用于被管理使用。

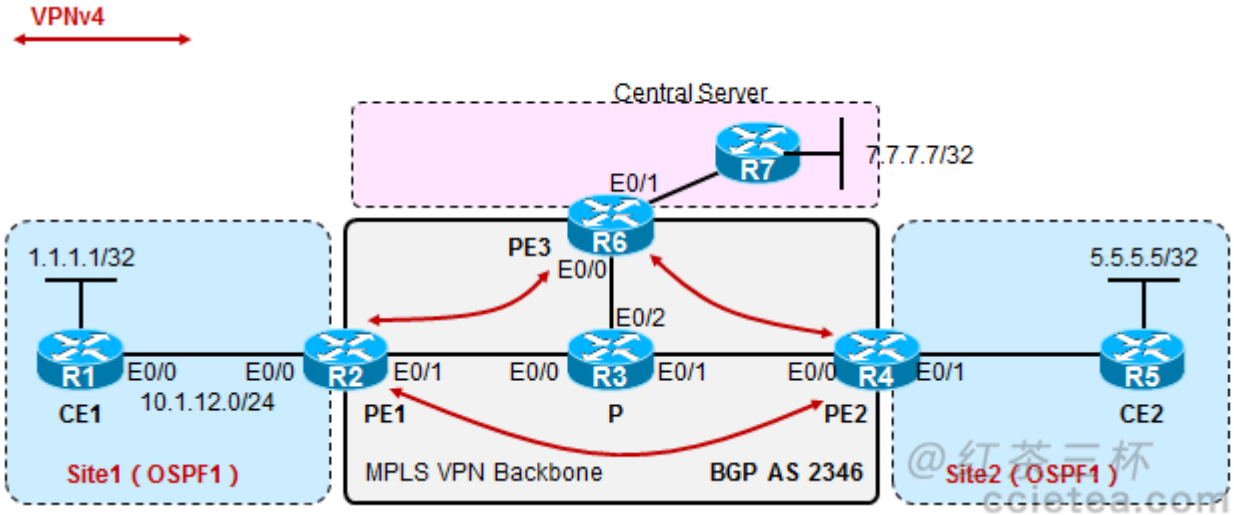
2. 解决办法



上面的配置中，ACL10 用于匹配分配给 CE 路由器的管理性 Loopback 的网段，现在通过 route-map NMS，只将这路由在更新的时候附加上 123:888 的 RT。这样一来，就可以在 PE-CS 上 import。

另一方面，所有的 VRF 都 import RT 值 123:999。

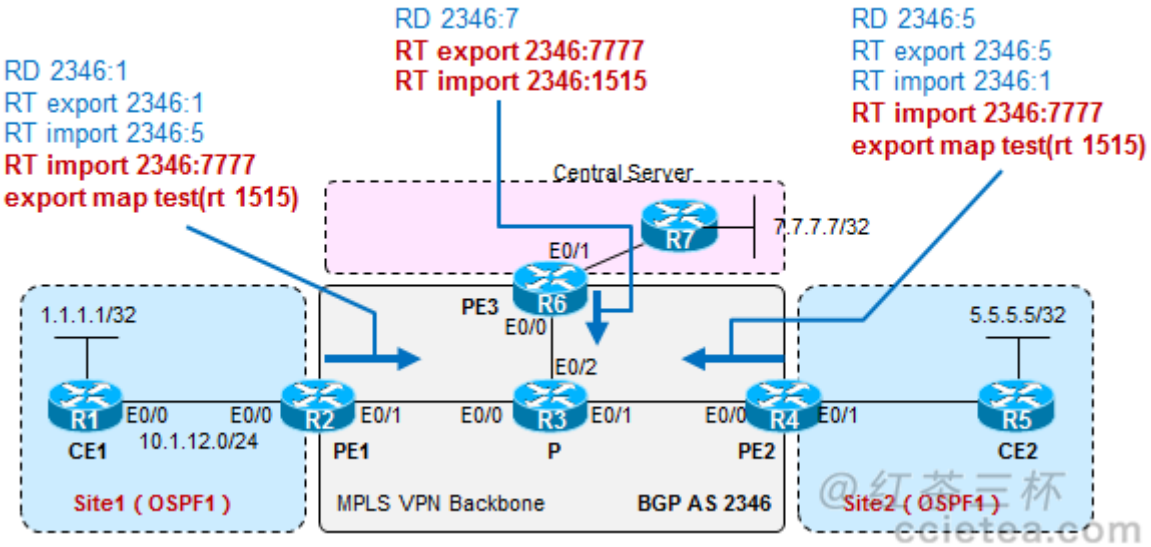
3. 实验演示



环境描述

- **本实验共有两个站点，Site1 及 Site2，另外有一个 Central Site**
- **Central Site 的功能是管理 CE1、CE2 两台设备，对于 R7-CentralServer 来说，只有 CE1 和 CE2 的 Loopback0 的可达性。只能通过这个 IP 管理两台 CE**
- **Site1 和 Site2 之间的可达性就不说了这个比较简单**

实验简述



其实就是个 RT 的游戏对吧？

上图中已经标记出了我们为这个游戏设计的 RT，蓝色字体部分是为了让 Site1 及 Site2 的 VPN 路由互通，

这个就不管了，重点看红色字体部分。

我们在 PE1 及 PE2 上，使用 route-map test，将匹配 CE1 及 CE2 的 Loopback 路由，增加一个 export RT 2346:1515，然后在 PE3 上 import；另外，PE3 上，将 Central-CE 的路由 RT export 设置为 2346:7777，然后在 PE1 及 PE2 上 import。完事儿。

看一下配置吧：

PE1 的配置如下：

```
ip vrf cisco
 rd 2346:1
 export map test
 route-target export 2346:1
route-target import 2346:5
 route-target import 2346:7777
 !
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface Ethernet0/0
 ip vrf forwarding cisco
 ip address 10.1.12.2 255.255.255.0
interface Ethernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 1 vrf cisco
redistribute bgp 2346 subnets
 network 10.1.12.2 0.0.0.0 area 0
!
router ospf 100
```

```
network 1.1.1.1 0.0.0.0 area 0
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
!
router bgp 2346
 no bgp default ipv4-unicast
neighbor 4.4.4.4 remote-as 2346
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 6.6.6.6 remote-as 2346
 neighbor 6.6.6.6 update-source Loopback0
 !
 address-family vpnv4
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
  neighbor 6.6.6.6 activate
  neighbor 6.6.6.6 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf cisco
  no synchronization
  redistribute ospf 1 vrf cisco match internal external 1 external 2
 exit-address-family
!
access-list 1 permit 1.1.1.1
route-map test permit 10
 match ip address 1
 set extcommunity rt   2346:1515 additive                    !! 追加 RT 2346:1515
```

其他的没啥了。

7 因特网接入

7.1 概述

首先强调一下，本章是基于《MPLS 技术架构》一书的相关内容做的扩展讲解，实际环境中 MPLS VPN 网络的因特网接入往往更加复杂和多样，大家权当本章是纯技术宅的角度做的解析，朕也是费了很大劲儿编写的，各位看官笑纳。

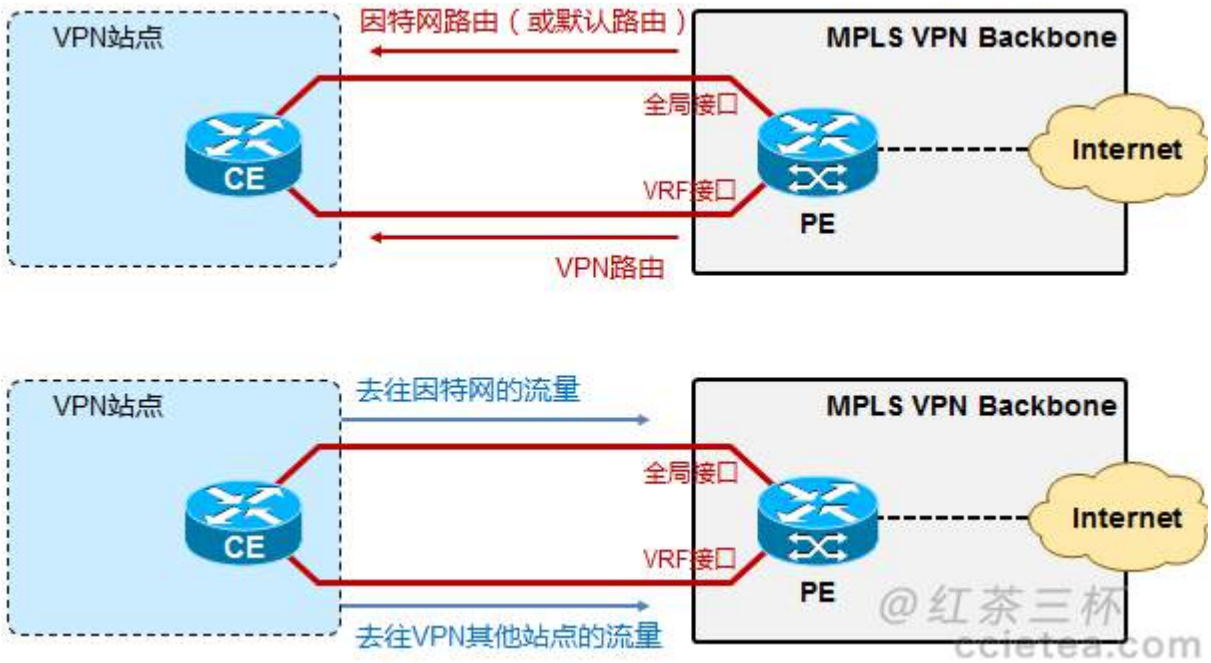
首先有个基本的认知，我们现在已经清楚的知道 MPLS VPN 技术框架下的 VPN 的概念，一般来说一个 VPN 客户，可能有多个 Site 站点，站点可能在地理上处于不同的位置，通过最近的 PE 接入 MPLS VPN 网络。通过在 PE 上部署 VRF 来进行不同 VPN 客户的隔离。而同一个 VPN 客户的不同站点之间，可以灵活的交互客户的路由信息从而实现互访，并且不同的 VPN 之间不会互相干扰和冲突。

那么如果在 MPLS VPN 环境下 ,客户希望通过 MPLS VPN 网络访问因特网该如何解决？前面已经说了 ,VPN 客户的站点 ,相当于是出于 PE 路由器的 VRF 保护之下的 ,作为 CE 来讲 ,它是不可能看到 MPLS VPN Backbone 内的路由的，同样，Backbone 内也是看不到 VPN 客户的路由的。那么因特网的入口，又往往是挂在 MPLS VPN Backbone 的全局路由空间之中 ,这个路由空间又是完全独立于 VRF 路由空间的 ,换句话说 ,对于 VPN 客户来说，他们无法从 VRF 路由空间“跳”到 Backbone 的全局路由空间啊。这就是本章要解决的内容。

7.2 注入因特网路由到 VRF

这是一种看似简单，但是也是纯 2B 的解决方案。服务提供商将完整的因特网路由表注入到 VRF 中。这个就不用多说了，太坑爹了。因特网路由条目那么多，全给 VRF 不得直接把丫撑爆了啊。接下去来详细看看几种稍微比较靠谱点的解决方案：

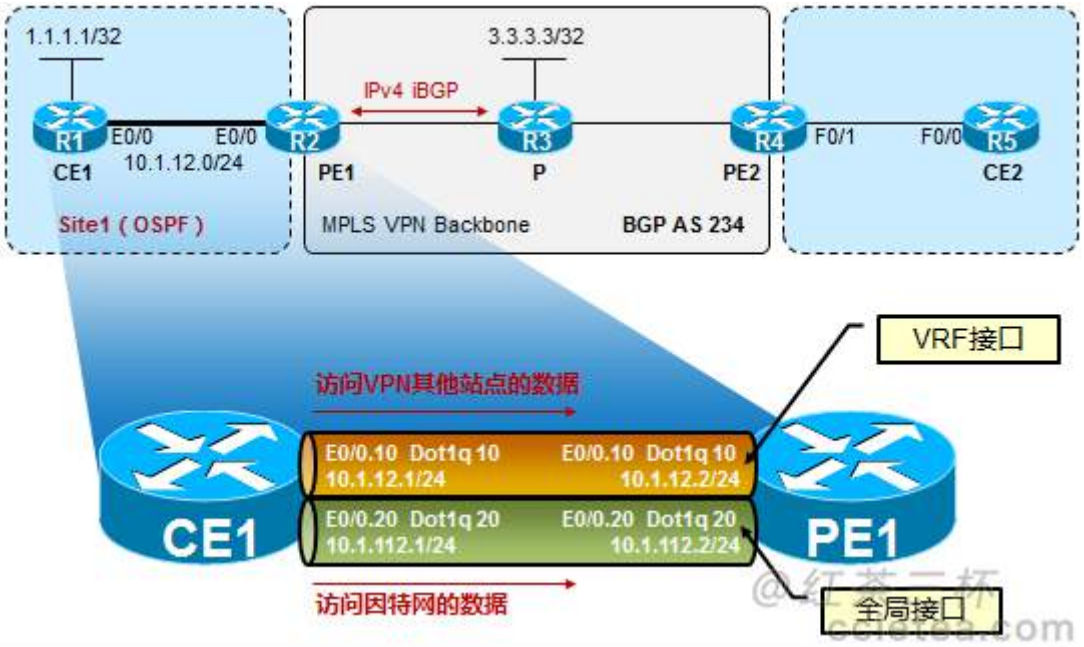
7.3 PE-CE 间双链路或子接口互联



这种解决方案，在 PE 上用两个接口连接 CE，一个是 PE 全局路由表的接口，另一个是 VRF 接口。PE 通过全局路由表接口，与 CE 交互因特网路由；用 VRF 接口与 CE 交互 VPN 路由。这样一来，当用户访问 VPN 其他站点的时候，就将流量转发到 PE 的 VRF 接口上，如果访问因特网就将流量转发到另一个接口上。这种解决方案的一个明显的缺点是，需要占用 PE 两个物理接口。

另一个方案是，用一个物理接口来互联 PE-CE，但是划分子接口。实现的思路和上面描述的是一样的。

实验演示：



1. 实验环境

- 我们重点看 Site1，Site1 的 CE-PE 运行 OSPF。（Site1 和 Site2 之间的 VPN 流量互通我们就不再赘述了，到这个阶段应该非常熟了）
- 现在假设 R3 也就是 P 路由器有个出口直接通往 Internet，这个接口处于路由器的全局路由空间中，为了简化实验，我们就在 R3 上开一个 Loopback 100.100.100.100/32 来模拟 Internet。这个 Internet 节点在 Backbone 内的全局路由表里是可达的。
- 那么现在，如何让 CE1（重点看 1.1.1.1）访问到这个 Internet 节点（100.100.100.100）呢？

首先为了让 PE1（的全局路由表）学习到 Internet 的路由，我们在 PE1 及 P 之间激活 BGP IPv4 地址族来交互因特网路由。当然实际的情况可能未必是这样来操作，这只是个模拟环境，不过我们只要记住一点，现在这些 Internet 路由的可达性，仅仅存在于 Backbone 内的全局路由空间，PE1 下的 VRF 是无法访问到的。再强调一下，这只是模拟环境，别较真啊。

2. 设备配置

CE1 的配置如下：

```
interface Ethernet0/0
    no shutdown
interface Ethernet0/0.10
  encapsulation dot1Q 10
  ip address 10.1.12.1 255.255.255.0
```

```
interface Ethernet0/0.20
   encapsulation dot1Q 20
   ip address 10.1.112.1 255.255.255.0
interface Loopback0
   ip address 1.1.1.1 255.255.255.255
!
router ospf 1
   network 1.1.1.1 0.0.0.0 area 0
   network 10.1.12.1 0.0.0.0 area 0
!
ip route 0.0.0.0 0.0.0.0 10.1.112.2 eth0/0.20          !!用于匹配访问 Internet 的流量
```

3. **PE1 的配置如下：**

```
interface Ethernet0/0
   no shutdown
interface Ethernet0/0.10
   encapsulation dot1Q 10
   ip vrf forwarding cisco                    !!这个子接口是 VRF 接口，用于和 CE1 交互 VPN 路由及流量
   ip address 10.1.12.2 255.255.255.0
interface Ethernet0/0.20
   encapsulation dot1Q 20
   ip address 10.1.112.2 255.255.255.0        !!这个子接口是全局路由接口，用于交互 Internet 路由及流量
!
router ospf 1 vrf cisco
   redistribute bgp 234 subnets
   network 10.1.12.2 0.0.0.0 area 0
!
router ospf 100                               !!Core 内的 IGP 协议，这里不用理会
network 2.2.2.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
!
router bgp 234
   no bgp default ipv4-unicast
```

```
 neighbor 3.3.3.3 remote-as 234
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 4.4.4.4 remote-as 234
 neighbor 4.4.4.4 update-source Loopback0
!
 address-family ipv4
    no synchronization
    neighbor 3.3.3.3 activate                    !!激活 3.3.3.3 的 IPv4 地址族（全局路由空间），以便交互 Internet 路由及指向 CE1 网络的回程路由。注意，这不是 VRF 的路由空间。
    network 1.1.1.1 mask 255.255.255.255        !!将静态配置的指向 CE1 网络的回程路由通告给 Backbone 内的运营商路由器知晓。这样 CE 访问 Internet 的流量才能有去有回
    no auto-summary
 exit-address-family
!
 address-family vpnv4
 neighbor 4.4.4.4 activate
 neighbor 4.4.4.4 send-community extended
exit-address-family
!
 address-family ipv4 vrf cisco
 no synchronization
 redistribute ospf 1 vrf cisco match internal external 1 external 2
exit-address-family
!
ip route 1.1.1.1 255.255.255.255 Ethernet0/0.20          !!指向 CE1 网络的回程路由
```

P 路由器的配置比较简单。

```
interface Loopback1
 ip address 100.100.100.100 255.255.255.255
!
router bgp 234
 neighbor 2.2.2.2 remote-as 234
 neighbor 2.2.2.2 update-source Loopback0
```

```
!
 address-family ipv4
  no synchronization
  neighbor 2.2.2.2 activate
  network 100.100.100.100 mask 255.255.255.255
  no auto-summary
 exit-address-family
!
```

CE1 和 PE1 通过一个物理接口、两个子接口互联。其中一个子接口（子接口 E0/0.10）用于 PE-CE 之间交互 VPN 路由，CE1 可以通过这个子接口去访问 VPN 的其他站点，数据包进入 PE1 的这个子接口（子接口 E0/0.10）将会被压上标签送入 MPLS VPN Backbone。当然，为啥 CE1 访问其他 VPN 站点，流量就一定会从子接口 E0/0.10 出去呢？因为它是从这个子接口学习到其他站点的 VPN 路由的，这些路由是精确路由（相比于指向另一个子接口的默认路由而言）。

CE1 上的另一个子接口 E0/0.20 连接着 PE1 的 E0/0.20，CE1 上指一条默认路由从 E0/0.20 接口出去；PE1 上将这个子接口放入全局路由空间，并且指一条到 CE1 的回程路由。然后 PE1 将这条指向 CE1 网络的路由放进 BGP，让 Backbone 内的路由器（这里是 P 路由器）去学习。这么做，似乎有点不合理，客户的 VPN 路由岂不是暴露在 Backbone 里了？

我们可以在 PE1 上使用 NAT 来解决这个问题，让 CE1 访问 Internet 的流量，在 PE1 送出去之前将源地址转换掉。

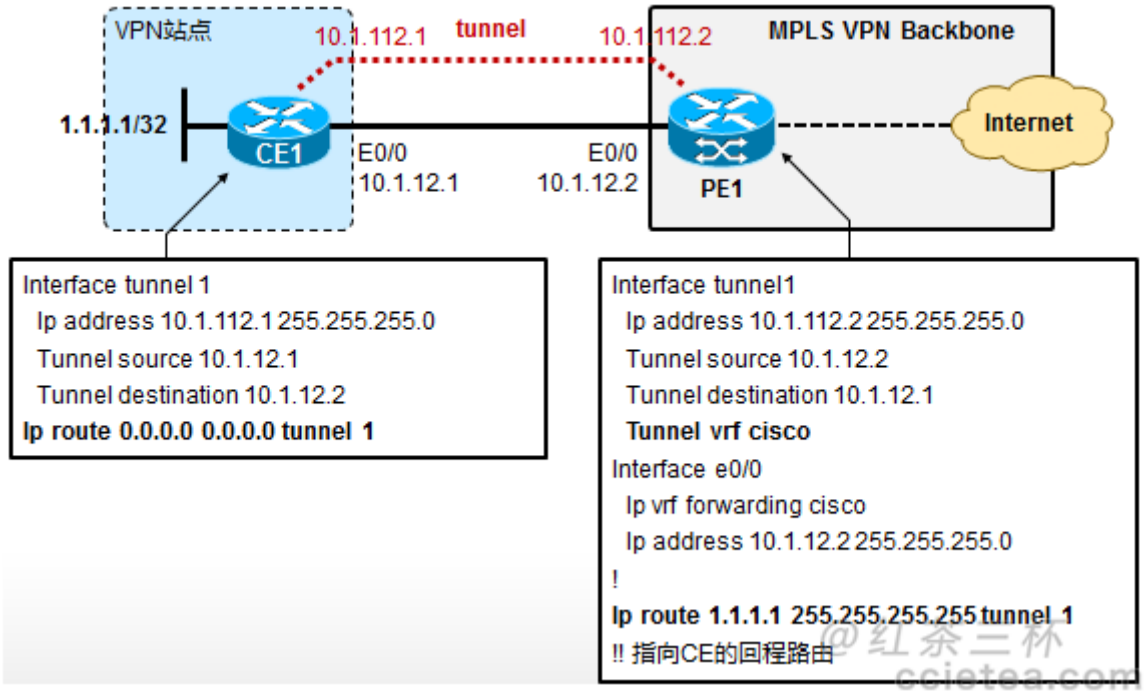
那么 PE1 的配置改成下面这样：

```
..... 省略不相干的配置 .....
router bgp 234
  no bgp default ipv4-unicast
neighbor 3.3.3.3 remote-as 234
  neighbor 3.3.3.3 update-source Loopback0
  neighbor 4.4.4.4 remote-as 234
  neighbor 4.4.4.4 update-source Loopback0
 !
  address-family ipv4
    no synchronization
    neighbor 3.3.3.3 activate                    !!激活 3.3.3.3 的 IPv4 地址族，以便交互 Internet 路由
   -network 1.1.1.1 mask 255.255.255.255-        !!CE1 的路由不再被通告到 Backbone
```

```
    no auto-summary
  exit-address-family
 !
 address-family vpnv4
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf cisco
  no synchronization
  redistribute ospf 1 vrf cisco match internal external 1 external 2
 exit-address-family
!
ip route 1.1.1.1 255.255.255.255 Ethernet0/0.20          !!指向 CE1 网络的回程路由
!
interface Ethernet0/0.20
  ip nat inside
interface Ethernet0/1
  ip nat outside
ip nat inside source list 1 interface Ethernet0/1 overload
!
access-list 1 permit 1.1.1.1
```

7.4 使用 tunnel

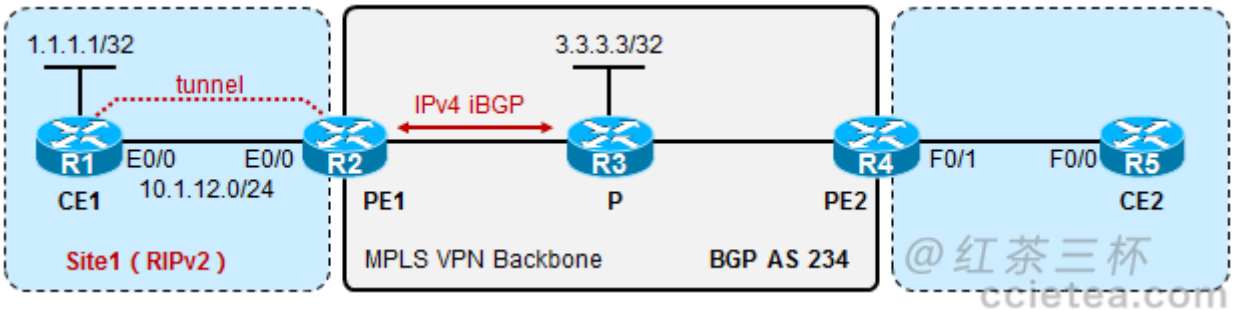
这个解决方案，是在 PE 和 CE 上，跑一个 tunnel，这个 tunnel，用来承载访问因特网的流量。



如此一来，CE1 上去往其他 VPN 的数据，将会走物理接口到 PE1 上的 VRF 物理接口，从而转入 MPLS VPN 网络；而去往 Internet 的数据，将直接走 tunnel 到 PE1，从而进入 PE1 的全局路由空间。

注意 PE1 的配置，在 PE 路由器上的隧道接口需要使用 tunnel vrf 命令，这是因为隧道的端点没有存在于全局路由表中，而是赋给了某个 VRF，同时，因为隧道口没有使用 ip vrf forwarding x 命令，因此它是存在全局路由空间的，也就是在全局路由表中能看到 PE 本地配置的隧道口直连路由。所以实际上 PE1 上的这个 tunnel 口，是帮助打通了 VRF 路由空间与全局路由空间。

实验示例：



1. **实验环境**

这个实验大环境跟上面是一样的。

CE1 及 PE1 之间跑一个 Tunnel，这个 tunnel 专门用来承载因特网流量。

2. **设备配置**

CE1 的配置如下：

```
router ospf 1
```

```
 network 1.1.1.1 0.0.0.0 area 0
 network 10.1.12.1 0.0.0.0 area 0
!
Interface tunnel 1
   Ip address 10.1.112.1 255.255.255.0
   Tunnel source 10.1.12.1
   Tunnel destination 10.1.12.2
Ip route 0.0.0.0 0.0.0.0 tunnel 1
```

PE1 的配置如下：

```
...... 省略不相干的配置 ......
Interface tunnel1
   Ip address 10.1.112.2 255.255.255.0
   Tunnel source 10.1.12.2
   Tunnel destination 10.1.12.1
   Tunnel vrf cisco
Interface e0/0
   Ip vrf forwarding cisco
   Ip address 10.1.12.2 255.255.255.0
!
Ip route 1.1.1.1 255.255.255.255 tunnel 1          !! 指向 CE 的回程路由
!
router bgp 234
 ……
   address-family ipv4
      neighbor 3.3.3.3 activate
      network 1.1.1.1 mask 255.255.255.255      !!将指向 CE1 网络的回程路由宣告进 BGP，以便 Backbone 内的路由器能够学习到，往返数据才没有问题。!
```

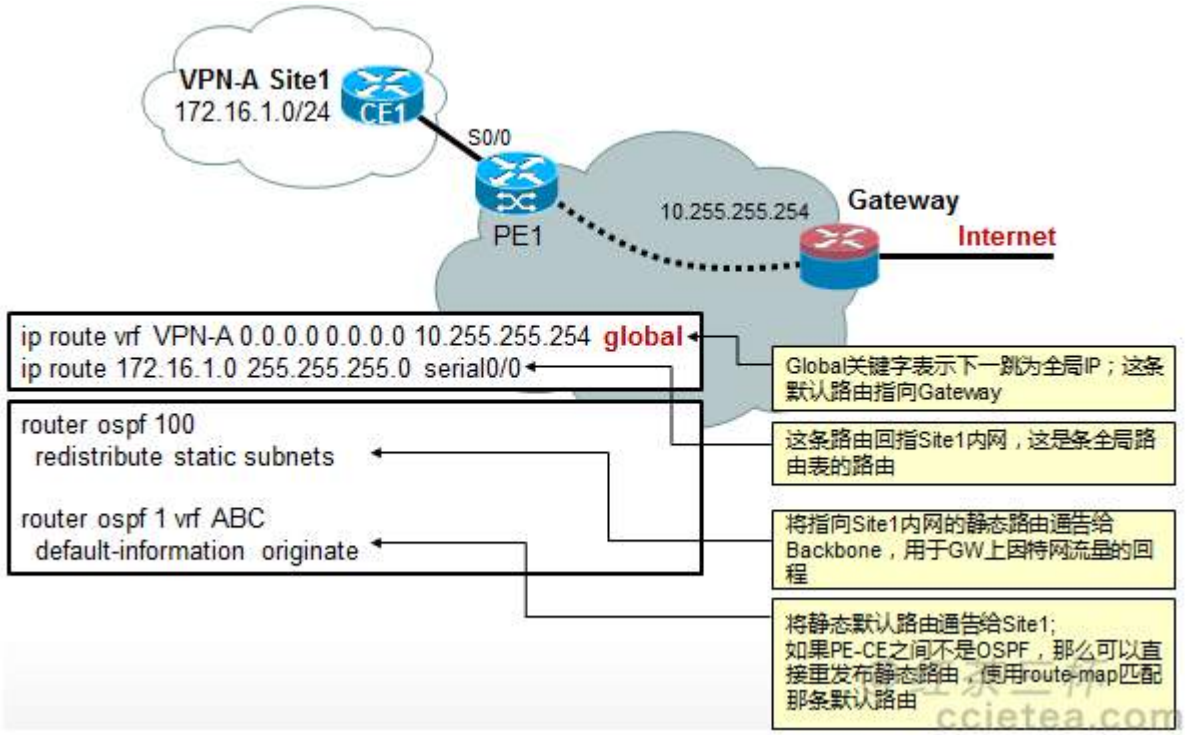
还是那个问题，将 VPN 客户的路由导入 Backbone 的全局路由空间是没有必要的。还是老办法，用 NAT。

那么 PE1 的配置改成下面这样：

```
...... 省略不相干的配置 ......
Interface tunnel1
```

```
 Ip address 10.1.112.2 255.255.255.0
 Tunnel source 10.1.12.2
 Tunnel destination 10.1.12.1
 Tunnel vrf cisco
 Ip nat inside
Interface e0/0
 Ip vrf forwarding cisco
 Ip address 10.1.12.2 255.255.255.0
!
Interface eth0/1
 Ip nat outside
!
Ip route 1.1.1.1 255.255.255.255 tunnel 1          !! 指向 CE 的回程路由
!
router bgp 234
  ……
 address-family ipv4
    neighbor 3.3.3.3 activate
    network 1.1.1.1 mask 255.255.255.255          !!不再将指向 CE 网络的路由注入 Backbone。
!
access-list 1 permit 1.1.1.1
ip nat inside source list 1 interface Ethernet0/1 overload
```

7.5 VRF Specify Default route



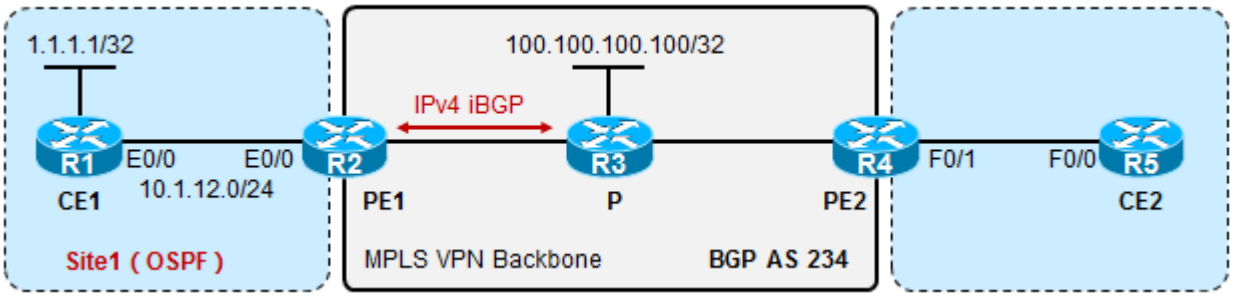
Ip route vrf *vrf-name 目的地 掩码 下一跳* global

关键点就是这条命令，该条静态路由本身最终是装载在 VRF 路由表中的，本来呢 VRF 路由表和全局路由表是隔离的，但是使用 global 关键字的话，我们可以让 VRF 表中的路由条目使用全局路由表的 IP 作为下一跳。

然后呢再将这条 VRF 静态路由重发布进 PE-CE 路由协议，让 VPN 客户网络学习到；这样，大家就都知道怎么访问因特网了。

同时，PE 上还需回指向 Site 内网的路由，这条路由是被添加在 PE 的全局路由表上，并将路由通告进 Backbone 网络，让 Internet gateway 也能学习到。那么当有数据要访问因特网的时候，数据包到达 PE 上连接 CE 的 VRF 接口，PE 查看 VRF 路由表，下一跳关联到了全局路由表，于是按照下一跳进行转发，将数据包送到了 Gateway。

实验演示：



1. 实验环境

大环境和前面几个小节的差不多，不再赘述。

2. 设备配置

CE1 的配置如下：

```
router ospf 1
  network 10.1.12.1 0.0.0.0 area 0
  network 1.1.1.1 0.0.0.0 area 0
```

PE1 的配置如下：

```
ip route vrf cisco 0.0.0.0 0.0.0.0 3.3.3.3 global        !! 为 VRF 路由表创建一条默认路由 ,注意这条路由的下一跳，是在全局 IP 路由表里的，因此需使用 global 关键字。
ip route 1.1.1.1 255.255.255.255 Ethernet0/0             !! 指向 CE1 的回程路由
!
router ospf 100                                          !! Core 内的 IGP，这里不用理会
 network 2.2.2.2 0.0.0.0 area 0
network 10.1.23.2 0.0.0.0 area 0
!
router ospf 1 vrf cisco
  redistribute bgp 234 subnets
  network 10.1.12.2 0.0.0.0 area 0
  default-information originate                          !! 将默认路由下发给 CE1，以便它能学习到
!
router bgp 234
 no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 234
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 4.4.4.4 remote-as 234
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family ipv4
   no synchronization
   neighbor 3.3.3.3 activate     !! 激活与 P 也就是 R3 的 IPv4 BGP 地址族为的是交互 Internet 路由，以及
```

```
客户回程路由
    network 1.1.1.1 mask 255.255.255.255        !! 将指向 CE1 的回程路由，通告给因特网出口也就是 R3 知道，不然，回程流量就会有问题
  exit-address-family
!
  address-family vpnv4
    neighbor 4.4.4.4 activate
    neighbor 4.4.4.4 send-community extended
  exit-address-family
!
address-family ipv4 vrf cisco
  no synchronization
  redistribute ospf 1 vrf cisco match internal external 1 external 2
exit-address-family
```

经过上面的配置，当 1.1.1.1 要访问 100.100.100.100 的时候，IP 数据包发给了 PE1，PE1 是从 VRF 接口上收到这个数据包的，因此查找 VRF 路由表，结果被那条默认路由匹配住了：

ip route vrf cisco 0.0.0.0 0.0.0.0 3.3.3.3 global

而默认路由的下一跳地址 3.3.3.3 在全局路由表里，因此又对下一跳地址在全局路由表里递归，得到下一跳 10.1.23.3，于是将 IP 数据包直接转给 10.1.23.3 也就到了 R3 上。

那么接下去就是 R3 要回包了，由于我们在 PE1 上 BGP 的 IPv4 全局地址空间里通告了 1.1.1.1，因此 R3 有回程路由，将报文回给 PE1，然后再由 PE1 转给 CE1。

在这个解决方案中，一个非常明显的缺陷是，我们竟然要向 MPLS VPN Backbone 的全局路由空间里通告客户的路由。因为如果你不通告，那么回程的流量肯定要受影响，怎么办呢？我们可以在 PE1 上做 NAT，让 CE1 访问 Internet 的流量在从 PE1 进入到 Backbone 之前先被 NAT 掉，这样，Backbone 内就不需要客户内网的路由了。

现在 PE1 的配置修改如下：

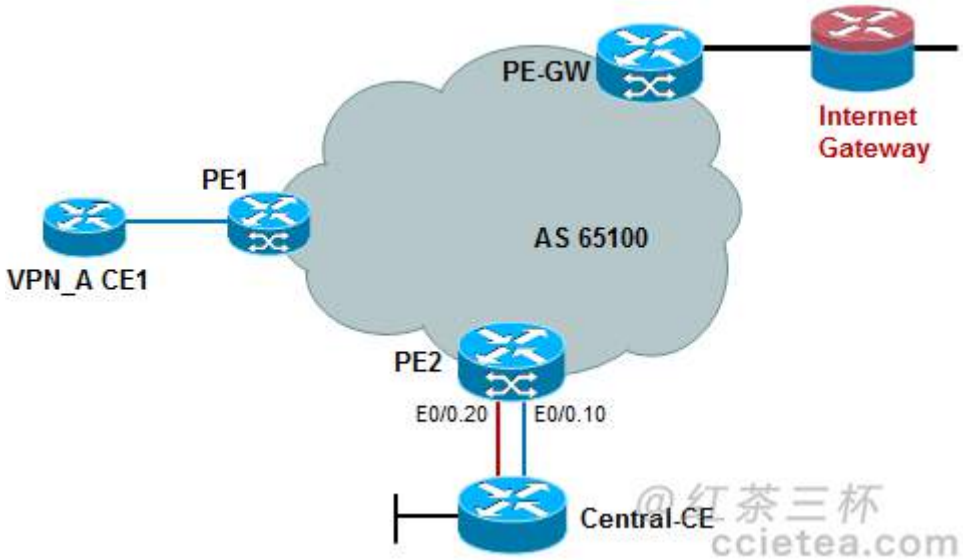
```
ip route vrf cisco 0.0.0.0 0.0.0.0 3.3.3.3 global       !! 这条不变
ip route 1.1.1.1 255.255.255.255 Ethernet0/0            !! 指向 CE1 的回程路由，PE1 自己是需要的
!
router ospf 100                                         !! Core 内的 IGP，这里不用理会
```

```
 network 2.2.2.2 0.0.0.0 area 0
network 10.1.23.2 0.0.0.0 area 0
!
router ospf 1 vrf cisco
   redistribute bgp 234 subnets
   network 10.1.12.2 0.0.0.0 area 0
   default-information originate                !! 将默认路由下发给 CE1，以便它能学习到
!
router bgp 234
no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 234
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 4.4.4.4 remote-as 234
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family ipv4
      no synchronization
      neighbor 3.3.3.3 activate    !! 激活与 P 也就是 R3 的 IPv4 BGP 地址族为的是交互 Internet 路由
      ~~network 1.1.1.1 mask 255.255.255.255~~      !! 这条命令去掉，现在不用向 Backbone 全局路由空间通
告 CE 的路由了，因为数据从 PE 出来就已经被 NAT 了。
   exit-address-family
 !
   address-family vpnv4
      neighbor 4.4.4.4 activate
      neighbor 4.4.4.4 send-community extended
   exit-address-family
 !
 address-family ipv4 vrf cisco
   no synchronization
   redistribute ospf 1 vrf cisco match internal external 1 external 2
 exit-address-family
!
access-list 1 permit 1.1.1.1
```

```
interface e0/0
  ip nat inside
interface e0/1
  ip nat outside
ip nat inside source list 1 interface Ethernet0/1 overload
```

测试的时候，直接在 CE1 上 ping 100.100.100.100 source 1.1.1.1 即可。

7.6 使用一个专门的 VPN 来访问 Internet

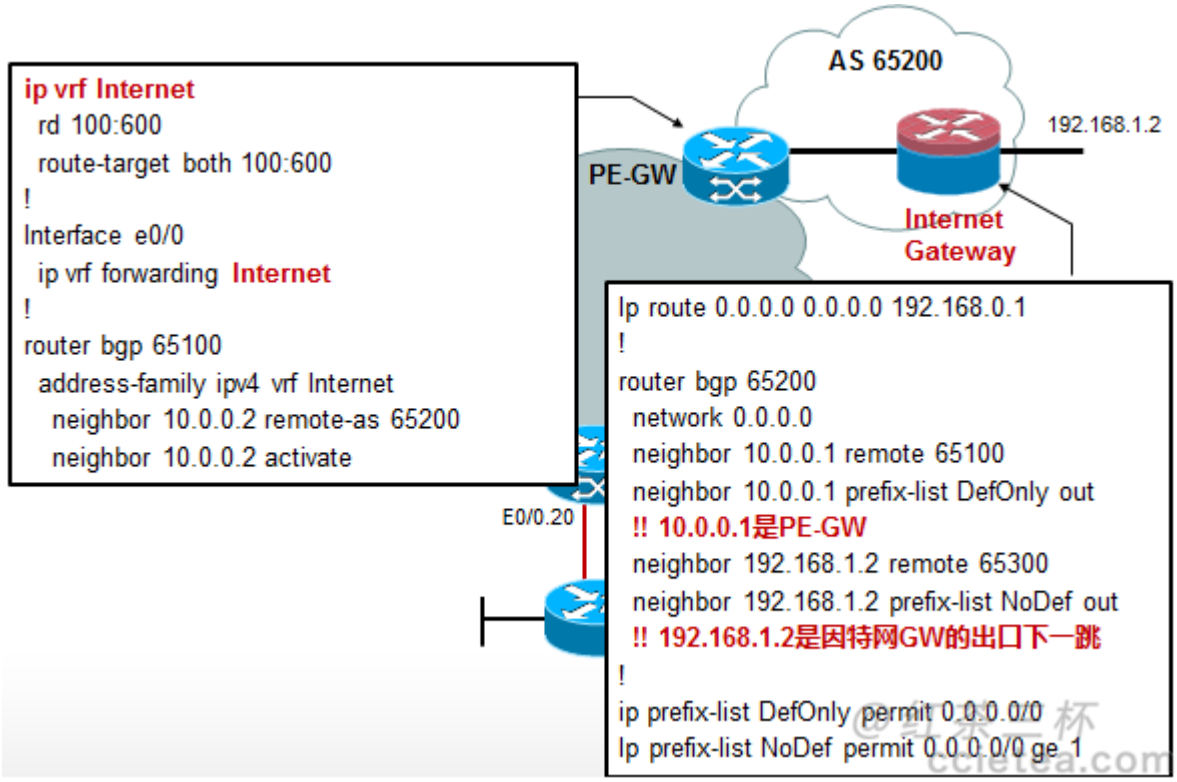


前面我们介绍的几种解决方案，普遍存在一个问题，那就是，我们不得不将 Internet 路由注入到 MPLS VPN Backbone 中，也就是说，P 路由器都会从 Core 里的 IGP 或者 BGP 学习到这些路由。这里其实是不大现实的。那么我们可以这样来操作，在 MPLS VPN 环境中，单独拿出一个 VPN 来，专门用于传输 Internet 路由。例如上面这个图，我们这样这来，将 Internet Gateway（此刻是一台 CE），挂在一个 PE 下面就是图中的 PE-GW，然后在这个 PE-GW 上，创建一个 VRF Internet，PE-GW 将连接 Internet GW 的接口放进这个 VRF，于是 PE-GW 从 Internet GW 上学习到的 Internet 路由就放进了 VRF Internet 路由表。那么接下去，Internet 路由就能够通过 MP-BGP 以 VPNv4 前缀的方式传播了。

我们现在 ,设置一个中心站点 ,图中的 PE2 就是中心站点的 PE。这个中心站点我们的设计是 ,PE2 与 PE-GW 建立 MP-BGP 连接，获取 VPNv4 的前缀。PE2 与中心站点内的 Central-CE 采用两个子接口对接，其中一个子接口划入 VPN-A ,用于和 VPN-A 的其他站点交互 VPN 客户路由。另一个子接口 ,划入 VRF Internet ,用于从 PE-GW

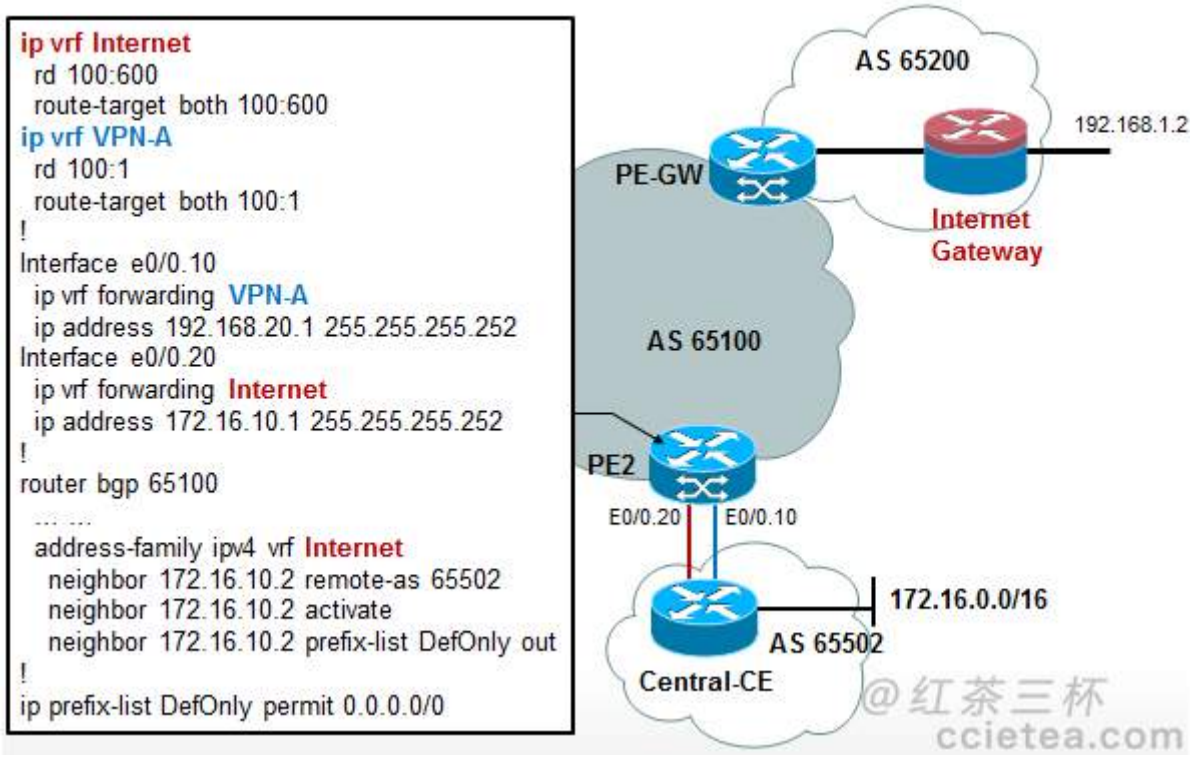
学习获取到因特网路由。

这样一来 Central-CE 等于说是即有了 VPN-A 各站点的 VPN 路由，又有因特网路由。那么它可以将 VPN-A 的站点路由传递给 PE-GW，又可以将因特网路由传递给 PE1。如此，VPN-A 的其他站点用户，就能通过 Central 站点来访问因特网。这就是一个典型的 Hub and Spoke 网络。这样做的一个典型的好处就是，所有的访问因特网的流量都经过中心站点，那么在中心站点，就可以进行统一的策略部署和安全防护，流量管控等等。

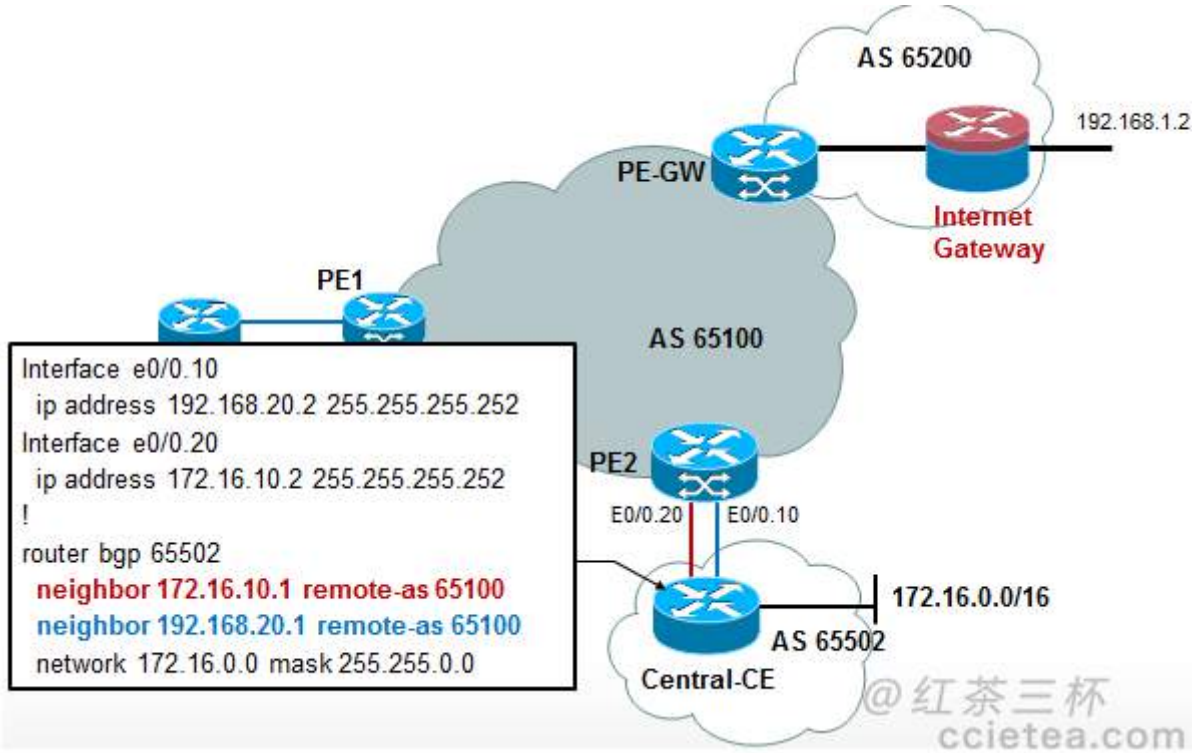


上面是 PE-GW 及 Internet GW 的配置。我们看到在 PE-GW 上创建了一个 VRF Internet，RT 为 100:600。然后将接口 E0/0 也就是连接 InternetGW 的接口放入 VRF Internet。然后 PE-GW 在 BGP 的 vrf internet 地址族中指 BGP 邻居 InternetGW。

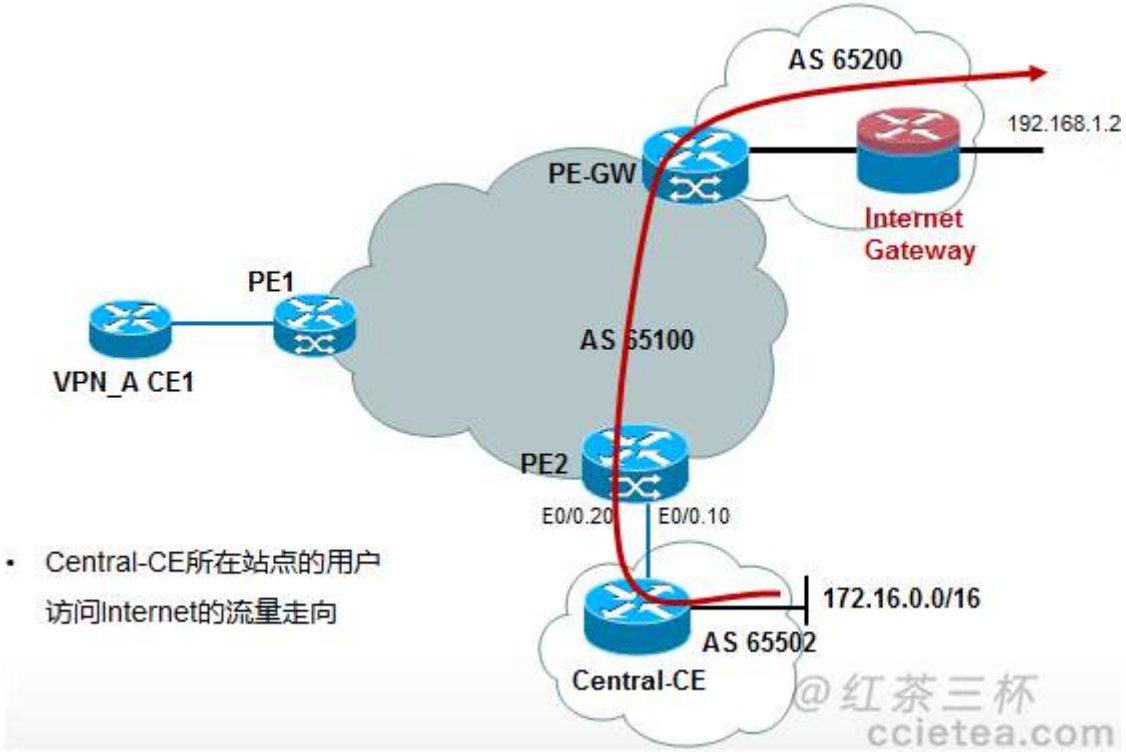
接着是 InternetGW（这里的角色是 CE）的配置，InternetGW 配置了一条默认路由，并将默认路由通告进了 BGP，从而 PE-GW 学习到这条默认路由，放进了 VRF Internet。



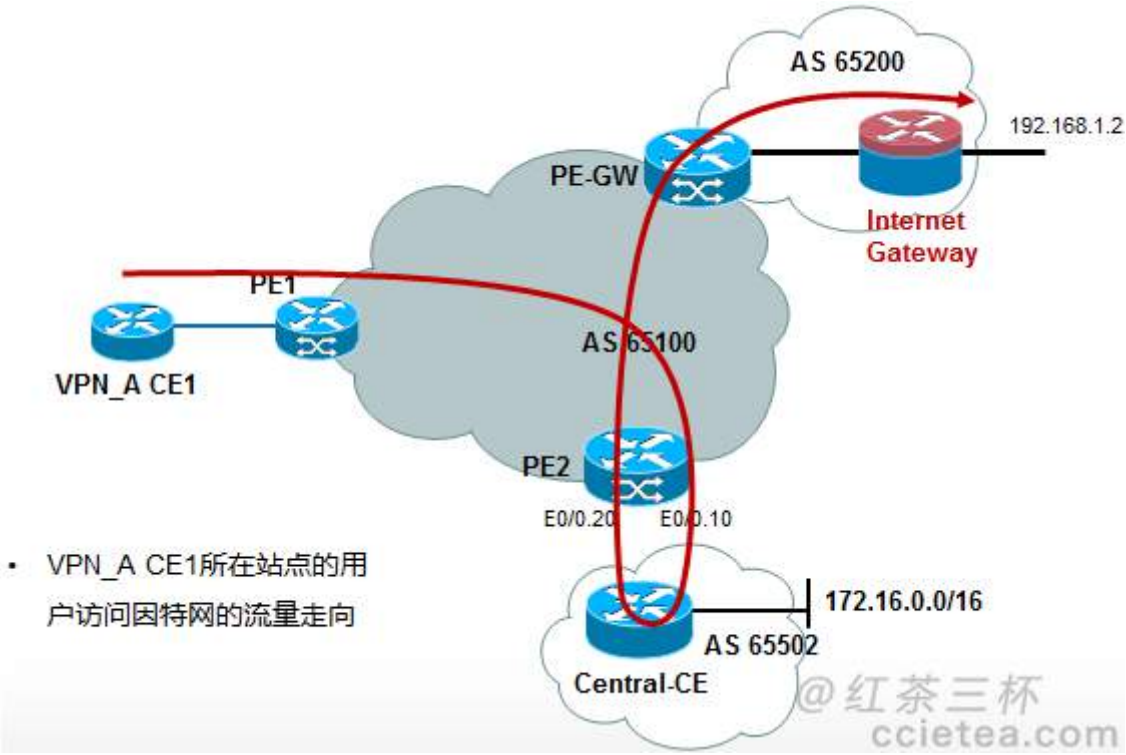
接着是是 PE2 的配置。PE2 上创建两个 VRF，分别是 VRF Internet 及 VRF VPN-A。PE2 创建两个子接口，分别与 Central-CE 对接。子接口 E0/0.10 放入 VRF VPN-A，另一个子接口 E0/0.20 放入 VRF Internet。然后 PE2 需与 PE-GW 及 PE1 建立 VPNv4 连接。同时在 BGP 的 ipv4 vrf Internet 地址族中，指 BGP 邻居 Central-CE 172.16.10.2。



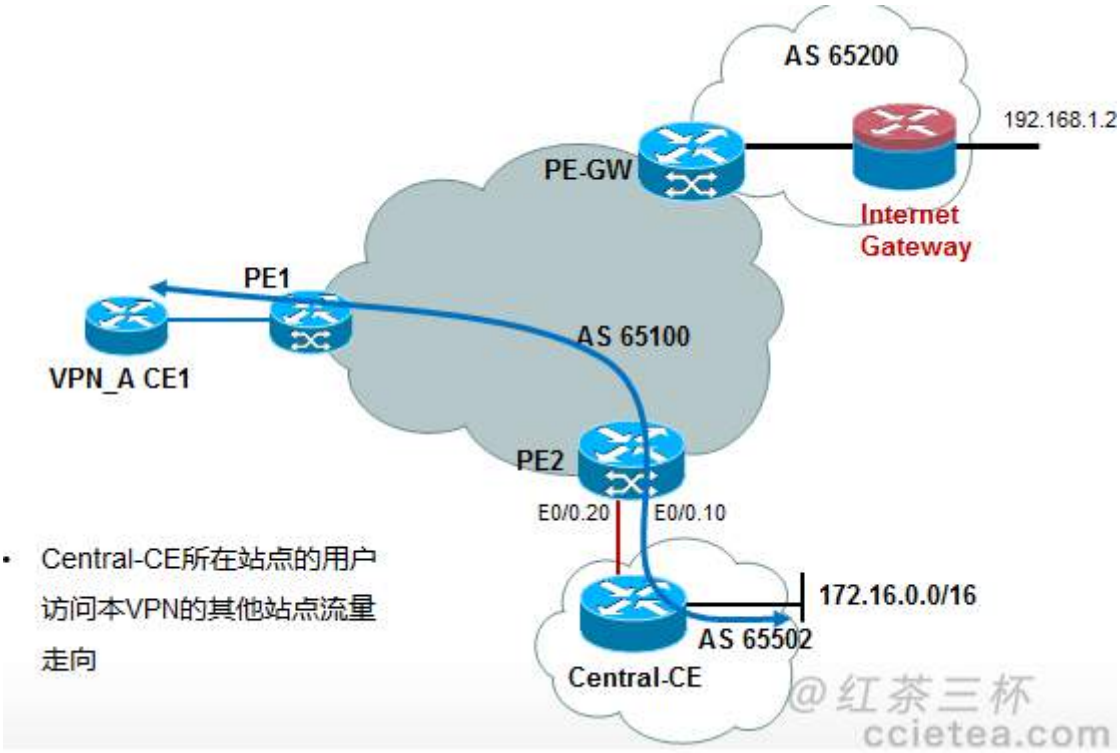
现在，我们再来看看流量：



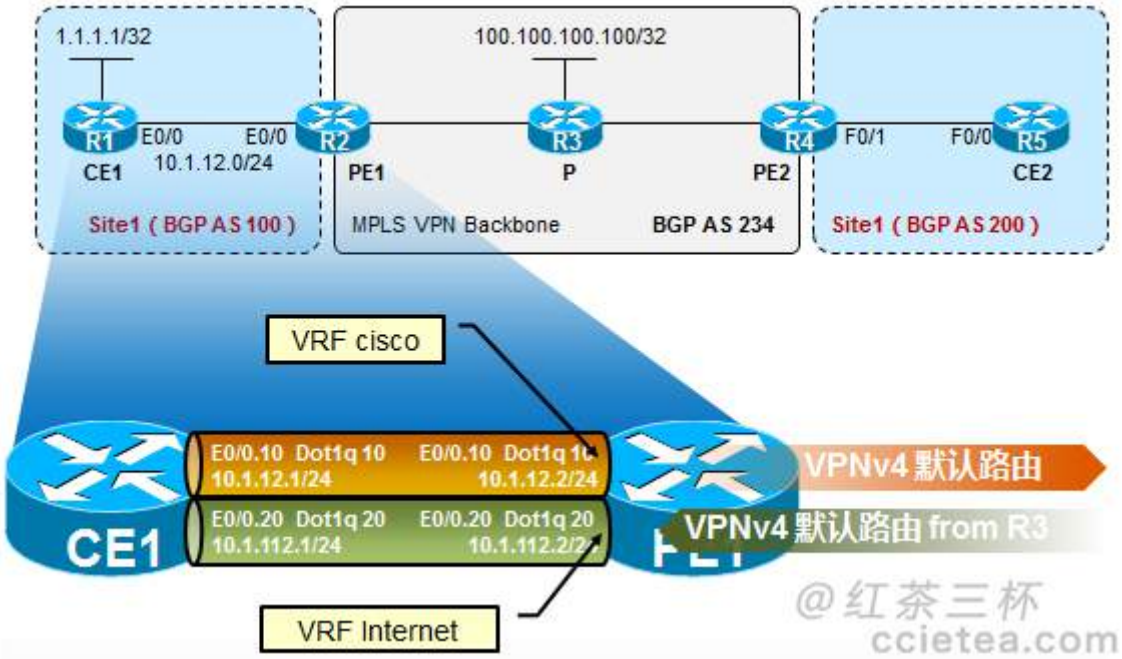
- Central-CE所在站点的用户访问Internet的流量走向



- VPN_A CE1所在站点的用户访问因特网的流量走向



实验演示：



1. **实验环境**

   CE1 及 PE2 之间跑两个子接口。

   在这个环境中，我们来模拟一个 Hub and Spoke 的网络，Site1(CE1 所在的站点)是 HUB，Site2 是 Spoke。Site2 访问因特网的流量，需流经 CE1，再从 CE1 出来到 P 路由器的因特网出口。

PE1 上创建两个 VRF，分别是 CISCO 和 Internet，其中 VRF Cisco 用于和 Site2 交互路由；VRF Internet 用来与 P 路由器交互路由。

PE1 与 P 激活 VPNv4 地址族；PE1 与 PE2 激活 VPNv3 地址族。

2. 实验需求

Site1 的 1.1.1.1 及 Site2 的 5.5.5.5 都要能够访问因特网 100.100.100.100

Site2 访问因特网的流量需先到达 Site1，再出 Site1 到因特网

3. 设备配置

CE1 的配置如下：

```
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
interface Ethernet0/0.10
 encapsulation dot1Q 10
 ip address 10.1.12.1 255.255.255.0
interface Ethernet0/0.20
 encapsulation dot1Q 20
 ip address 10.1.112.1 255.255.255.0
!
router bgp 100
 no synchronization
 network 1.1.1.1 mask 255.255.255.255          !! 通告 1.1.1.1 路由
 neighbor 10.1.12.2 remote-as 234
 neighbor 10.1.112.2 remote-as 234
```

PE1 的配置如下：

```
ip vrf Internet                 !! VRF Internet
 rd 234:999
 route-target export 234:999
 route-target import 234:999
!
ip vrf cisco
 rd 1:1
```

```
 route-target export 234:2
 route-target import 234:4
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface Ethernet0/0.10
 encapsulation dot1Q 10
 ip vrf forwarding cisco
 ip address 10.1.12.2 255.255.255.0
interface Ethernet0/0.20
 encapsulation dot1Q 20
 ip vrf forwarding Internet
 ip address 10.1.112.2 255.255.255.0
!
interface Ethernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
!
router bgp 234
 no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 234
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 4.4.4.4 remote-as 234
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 10.1.12.1 remote-as 100
```

```
!
address-family vpnv4
 neighbor 3.3.3.3 activate
 neighbor 3.3.3.3 send-community extended
 neighbor 4.4.4.4 activate
 neighbor 4.4.4.4 send-community extended
exit-address-family
!
address-family ipv4 vrf Internet
 no synchronization
 neighbor 10.1.112.1 remote-as 100
 neighbor 10.1.112.1 activate
 neighbor 10.1.112.1 allowas-in 1          !! 这是个难点，因为 Site2 的路由，在传递到 Site1 的时候，AS-PATH 里已经有 234 了，那么这些 VPN 路由，会经由 CE1 的另一个子接口传回给 PE1，PE1 由于看到传过来的 BGP 路由的 AS_PATH 里，出现了自己的 BGP AS 234，它将忽略这些路由，这样一来 P 就学习不到 Site2 的 VPN 路由了 ,Site2 也就访问不了因特网了。所以用上这条命令 ,让 PE1 把条件放宽一点，允许自己的 AS 号在 AS_PATH 中出现一次。
exit-address-family
!
address-family ipv4 vrf cisco
 no synchronization
 neighbor 10.1.12.1 remote-as 100
 neighbor 10.1.12.1 activate
 neighbor 10.1.12.1 allowas-in 1
exit-address-family
```

P 路由器的配置如下：

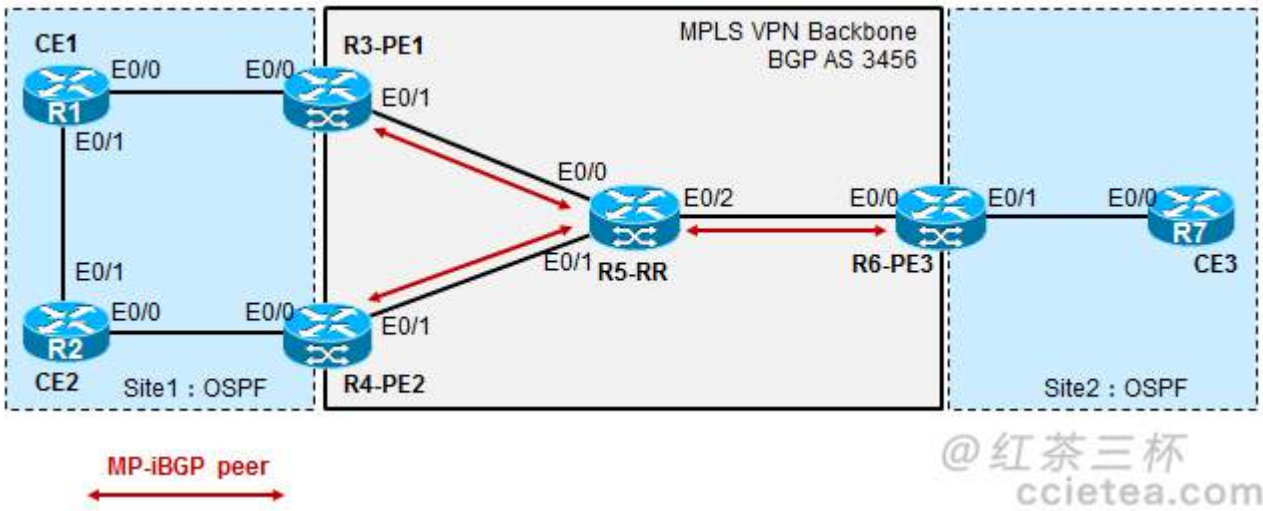
```
ip vrf Internet
 rd 234:999
 route-target export 234:999
 route-target import 234:999
!
interface Loopback0
```

```
 ip address 3.3.3.3 255.255.255.255
interface loopback1
ip vrf forwarding Internet
 ip address 100.100.100.100 255.255.255.255
!
mpls label range 300 399
mpls ldp router-id Loopback1
interface Ethernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
!
interface Ethernet0/1
 ip address 10.1.34.3 255.255.255.0
 mpls ip
!
router ospf 100
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
!
router bgp 234
 neighbor 2.2.2.2 remote-as 234
 neighbor 2.2.2.2 update-source Loopback1
 !
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf Internet
  no synchronization
  network 0.0.0.0
 exit-address-family
```

```
!
ip route vrf Internet 0.0.0.0 0.0.0.0 Null0          !!创建一条默认路由并通告进 BGP 让 PE 学到默认路由
```

8 MPLS VPN 双 PE 防环

8.1.1 实验 1：两边站点 PE-CE 间都运行 OSPF



8.1.1.1 实验描述

左右两个 Site，两边的 PE-CE 路由协议都是 OSPF。R5 为 RR，PE1、PE2、PE3 都是它的 client。

左右两个 Site 都将客户路由放给 PE，最终要实现 Site 之间能够互访。

我们重点关注两个 Site 在各种 OSPF 网络设计的情况下，网络的特征和产生的变化，以及 Site1 内双 PE 在各种环境下产生的问题和解决办法。

8.1.1.2 设备配置

CE1、CE2、CE3 的配置就不说了。

PE1 的配置如下：

```
ip vrf cisco
 rd 1:1
 route-target export 3456:12
 route-target import 3456:3
 route-target import 3456:12
!
ip cef
mpls label range 300 399
mpls ldp router-id Loopback0
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
interface Ethernet0/0
 ip vrf forwarding cisco
 ip address 10.1.13.3 255.255.255.0
interface Ethernet0/1
 ip address 10.1.35.3 255.255.255.0
 mpls ip
!
router ospf 1 vrf cisco
 redistribute bgp 3456 subnets
 network 10.1.13.3 0.0.0.0 area 0
!
router ospf 100
 router-id 3.3.3.3
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.35.3 0.0.0.0 area 0
!
router bgp 3456
 no bgp default ipv4-unicast
 neighbor 5.5.5.5 remote-as 3456
 neighbor 5.5.5.5 update-source Loopback0
```

```
address-family vpnv4
 neighbor 5.5.5.5 activate
 neighbor 5.5.5.5 send-community extended
exit-address-family
address-family ipv4 vrf cisco
 no synchronization
 redistribute ospf 1 vrf cisco match internal external 1 external 2
exit-address-family
```

PE2、PE3 的配置大同小异，这里不在赘述

PE2 的 VRF 配置：

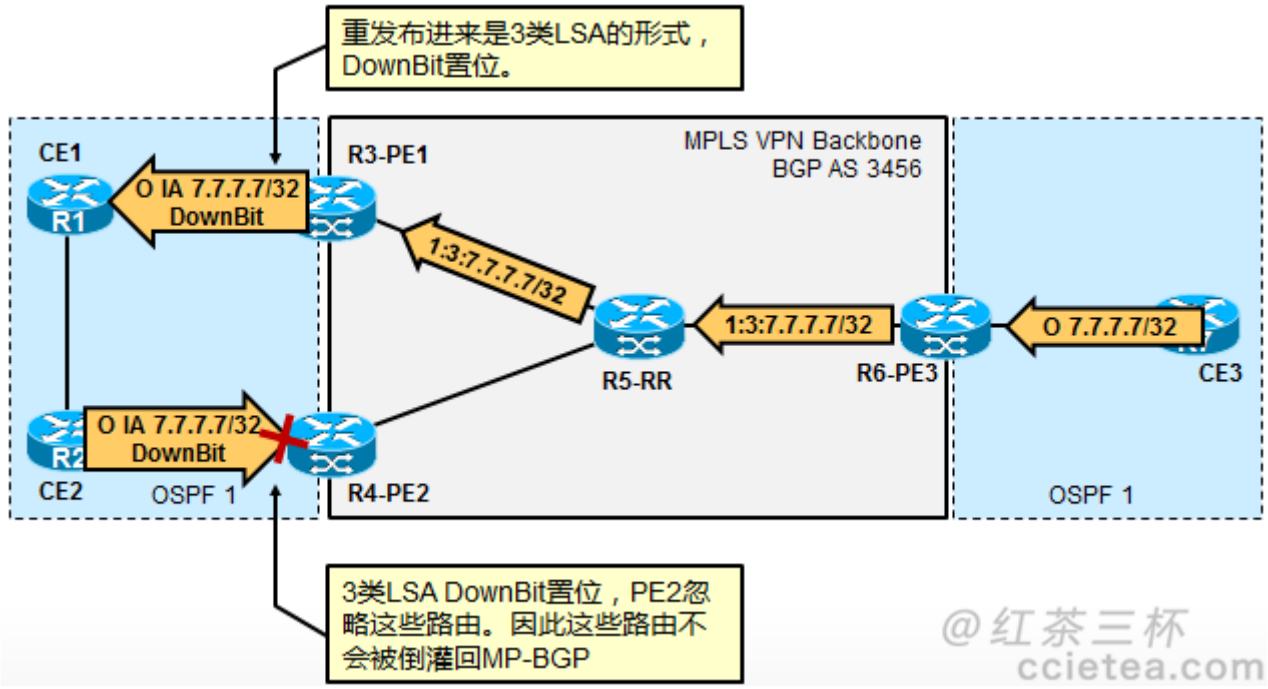
```
ip vrf cisco
 rd 1:2
 route-target export 3456:12
 route-target import 3456:3
 route-target import 3456:12
```

PE3 的 VRF 配置：

```
ip vrf cisco
 rd 1:3
 route-target export 3456:3
 route-target import 3456:12
 ip vrf forwarding cisco
```

8.1.1.3 实验现象

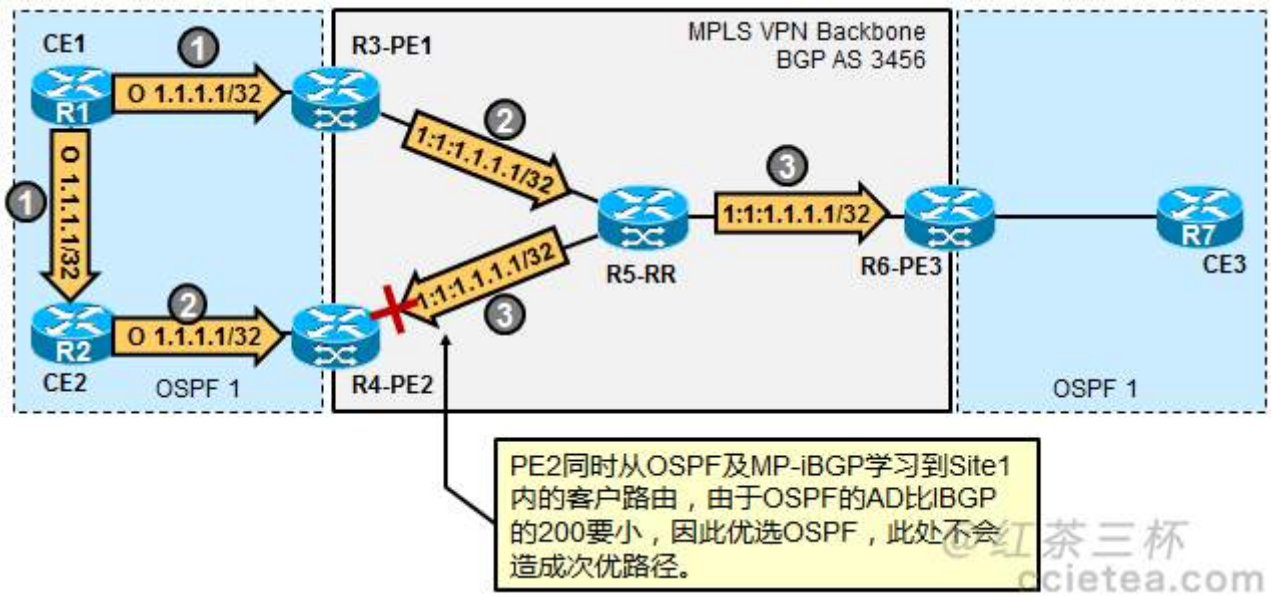
1. **测试 1：两边站点 PE 上的 VRF OSPF 进程号相同，且 Site1 内是 OSPF 单区域**
   - **首先考虑下 Site2 到 Site1 的路由传递过程**



我们假设先配置的是 PE1，后配置 PE2，来看看路由的传递过程。关键看 VPNv4 路由从 R5 传递给 PE1 后，PE1 将路由注入到 OSPF，形成 3 类 LSA，这些 3 类 LSA 的 DownBit 都置位了，因此，即使再经过 Site1 内的 OSPF 网络被传到 PE2，PE2 也不会用这些 LSA3 参与路由的计算，因为他们 DownBit 都置位了。既然忽略了这些 3 类 LSA，自然就不会加载进路由表，更不会在 PE2 上经过 OSPF 到 BGP 的重发布倒灌回 BGP。注意，此刻 PE2 是恒定忽略这些 Downbit 置位的 3 类 LSA 的，即使 shutdown 掉 PE2 的 e0/1 口，也是一样。

因此在这个环境下，得益于 OSPF Downbit 的设计，网络显得很可靠。

- **再考虑下 Site1 到 Site2 的路由传递过程**

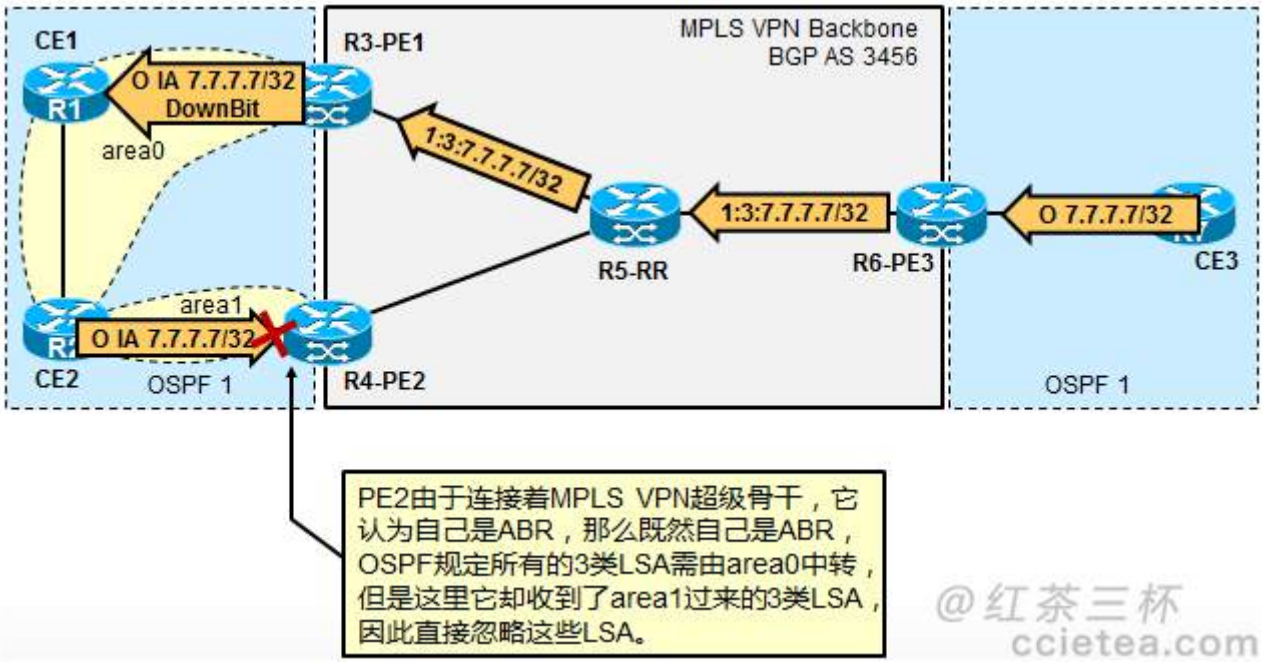


假设 PE1 先配置的 OSPF 到 BGP 的重发布，那么 site1 内的客户路由，被 PE1 注入到 MP-BGP 形成了 VPNv4 的前缀，经由 RR 反射给了 PE2，这时候，对于 PE2 来说，一边是从 VRF 的 OSPF 路由进程学习到 Site1 内的路由，另一边，从 MP-iBGP 也学习到这些路由，由于 OSPF 的 AD110 小于 IBGP 的 200，因此 PE2 优选 OSPF 路由，从而不会造成次优路径的问题。

2. 测试 2：两边站点 PE 上的 VRF OSPF 进程号相同，但 Site1 是多区域

注意，在这个测试中，我们的环境变了，变化的地方在 site1，从 OSPF 单区域，变成了多区域，R1 属于 area0，R2 是 ABR，跟 PE2 直连的网段，以及自身的 loopback 2.2.2.2 属于 area1。我们来网络会有什么影响。

- **首先考虑下 Site2 到 Site1 的路由传递过程**

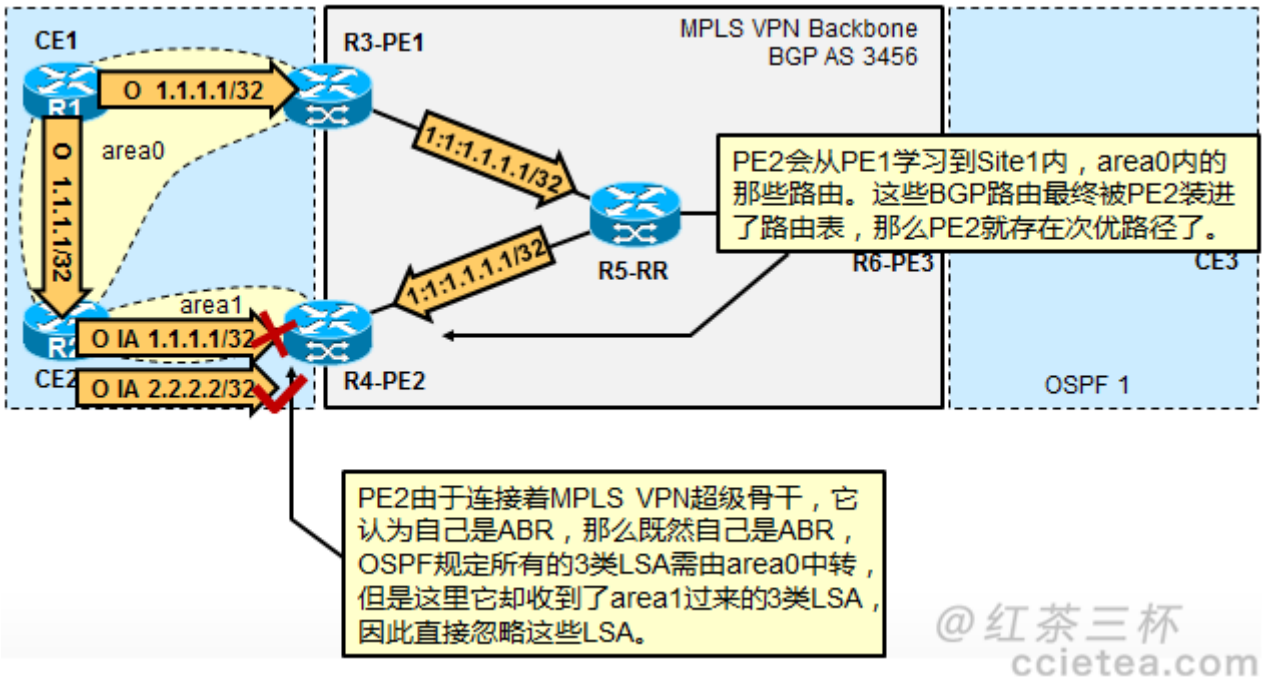


首先看 PE1 上发生什么事情，PE1 上首先是通过 BGP 学习到了 7.7.7.7 也就是 Site2 内的路由，然后将路由重发布到 OSPF，形成 LSA3，这些 LSA3 Downbit 置位。那么 CE1 就能学习到关于 7.7.7.7 的 OIA 路由，并且继续将 LSA3 传递给 CE2，CE2 也收下了，由于它是 ABR，从 area0 收到 LSA3，于是将 LSA3 又注入常规区域 area1，但是，这条 LSA3 就发生变化了，DOWNbit 位被清除。当然，这还不是最糟糕的，更糟糕的是，3 类 LSA 被传递给 PE2 后，我们前面分析过了，对于 OSPF 来说 MPLS VPN Backbone 就是一个 super backbone area 超级骨干区域，那么此刻 PE2 即是一台 ASBR 又是一台 ABR，我是一台 ABR，但是我却从 area1 收到 3 类 LSA，这是违反 OSPF 关于“3 类 LSA 必须经过 area0 中转”这一原则的，因此 PE2 忽略在 area1 上手收到的来自 CE2 的任何 3 类 LSA，当然，产生自 CE2 的 area1 内的区域内部 1 类 LSA 还是会收的。

正是由于 PE2 的 ABR 属性，造成的另一个问题是，来自 Site1 内 area0 的路由，经由 CE2 以 LSA3 的形式通告给 PE2，PE2 在路由计算的时候也照样忽略它们。这将直接导致接下去发生的问题：次优路径。怎么造成的呢，我们知道 PE1 上已经学习到了 Site1 内的 OSPF 路由，它将这些 OSPF 重发布到 BGP，并经

RR 反射给了 PE2，于是，PE2 从 BGP 学习到了这些 Site1 内的路由，另一方面，它又忽略了 CE2 更新过来的关于 Site1 内的 3 类 LSA，因此 PE2 的路由表里装载的 Site1 里 area0 的路由全是 BGP 的，这就是次优路径了，**此时此刻 PE2 的路由表是这样的：**

```
B        1.1.1.1 [200/11] via 3.3.3.3, 01:27:08
O        2.2.2.2 [110/11] via 10.1.24.2, 00:00:04, Ethernet0/0
B        7.7.7.7 [200/11] via 6.6.6.6, 01:27:08
B        10.1.12.0/24 [200/20] via 3.3.3.3, 01:27:08
B        10.1.13.0/24 [200/0] via 3.3.3.3, 01:27:08
B        10.1.67.0/24 [200/0] via 6.6.6.6, 01:27:08
```

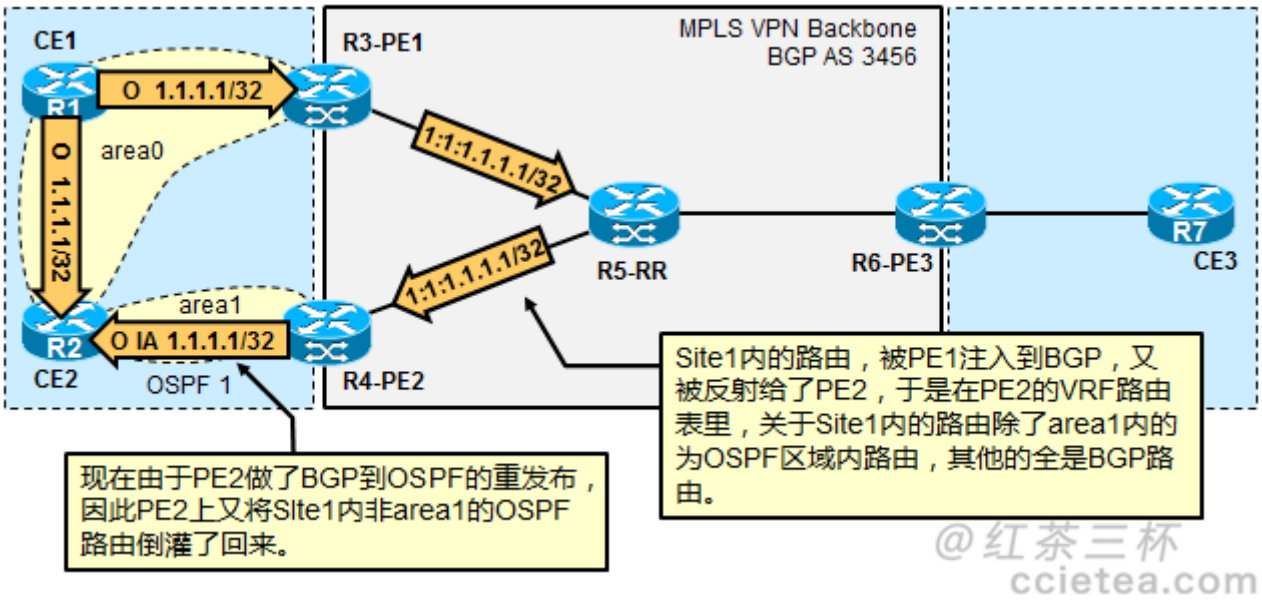


刚才描述的过程 ,可以用上面的图来概述。哥们实验的时候 ,做了个测试 ,将 PE2 上连接 RR 的接口 shutdown 了。于是 PE2 的 VRF 路由表变成了：

```
O        2.2.2.2 [110/11] via 10.1.24.2, 00:10:48, Ethernet0/0
```

只有一条 area1 内的路由，但是查看 PE2 的 OSPF database，发现还是有许多 CE2 发过来的 LSA3 的，很明显这就验证了我们前面的说法，PE2 这哥们认为爷爷我是个 ABR，直接忽略你 LSA3。

这个事情还没完，由于 PE2 路由表里的这个问题，加上 PE2 又做了 OSPF 和 BGP 的双向重发布，那么，PE2 上 ,这些从 PE1 倒灌过来的关于 Site1 内的路由 ,由于此刻在 PE2 的路由表里是 BGP 路由 ,那么在 BGP 到 OSPF 的重发布动作后，这些路由又被坑爹的倒灌回了 Site1：



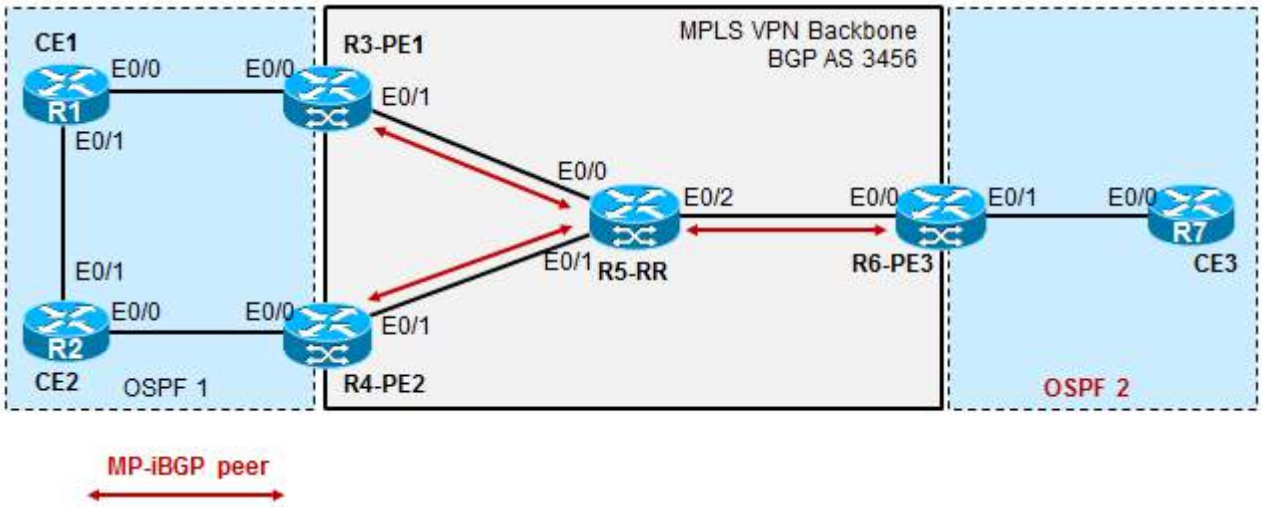
这样一来，就得看 CE2 的了。

- 对于 Site1 内 area0 里的路由，从 area0 学到的是 LSA1 及 LSA2，而从 PE2 倒灌回来的是 LSA3，CE2 当然优选 LSA1 及 LSA2 计算得出的区域内部路由，所以这里天然防环。
- 对于 Site2 的路由，CE2 同时学到 PE1 及 PE2 更新过来的 LSA3，因此比 metric，最后优选 PE2 作为下一跳。
- 关键点，注意，这里有个细节，R2,是一台 ABR 啊各位亲戚朋友。既然是一台 ABR，那么当它收到 PE2 重发布进来的 OSPF LSA3 ,即使放进了 LSDB 本该是忽略的 ,因为它是从常规区域 area1 收到这些 LSA3 的，违反 OSPF 的法则吧？但是实际上，在这个环境中，CE2 不但将 LSA 装载进了 LSDB，而且，这些 LSA 还参与路由计算。但是这些 LSA3CE2 自己是收下了也参与了路由的计算，却不会将他们泛洪到 area0 中，也就是说，只影响了 CE2 自己。

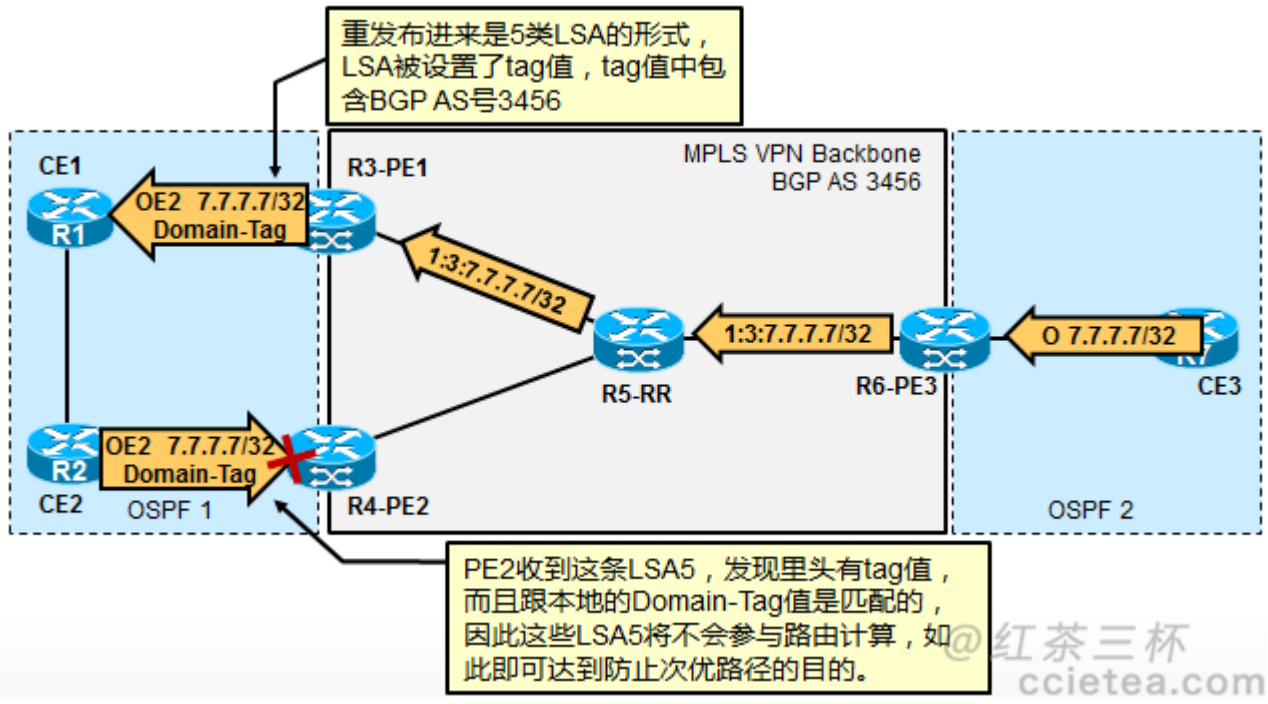
综上所述，当 Site1 内规划成多区域时，有可能引发一系列的潜在问题。所以：

- MPLS VPN 环境中，OSPF 网络的设计要非常谨慎
- 上面的问题，可以考虑在 CE2 和 PE2 之间，建立一个 virtual-link
- 再有就是可以利用一些策略工作过滤掉路由

3. 两边站点 PE 上的 VRF OSPF 进程号不相同，且 Site1 内是 OSPF 单区域



两边的 VRF OSPF 进程号不一样，大家都知道会发生什么了，Site2 的路由被 PE1 注入进 OSPF 后，会以 LSA5 的形式注入，那么接下去：

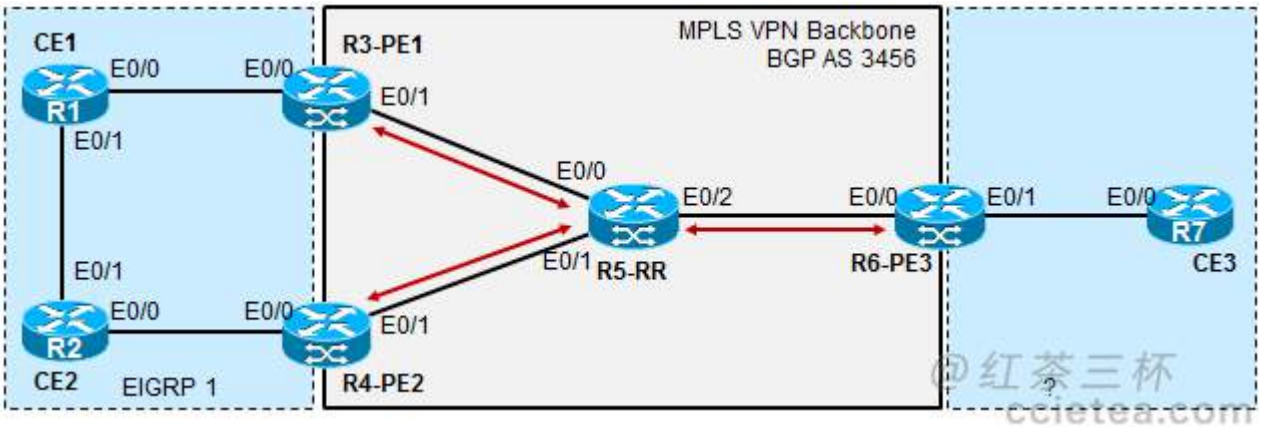


得益于 Domain-Tag 的设计，我们的网络还是比较健壮的。

同样，也不同担心 Site1 内的路由被 PE1 注入到 BGP 然后反射回 PE2 的问题，这里直接比 AD 值然后 BGP 光荣落败，没什么难度。

8.1.2 实验 2：双 PE 站点 PE-CE 运行 EIGRP 的情况

8.1.2.1 实验描述

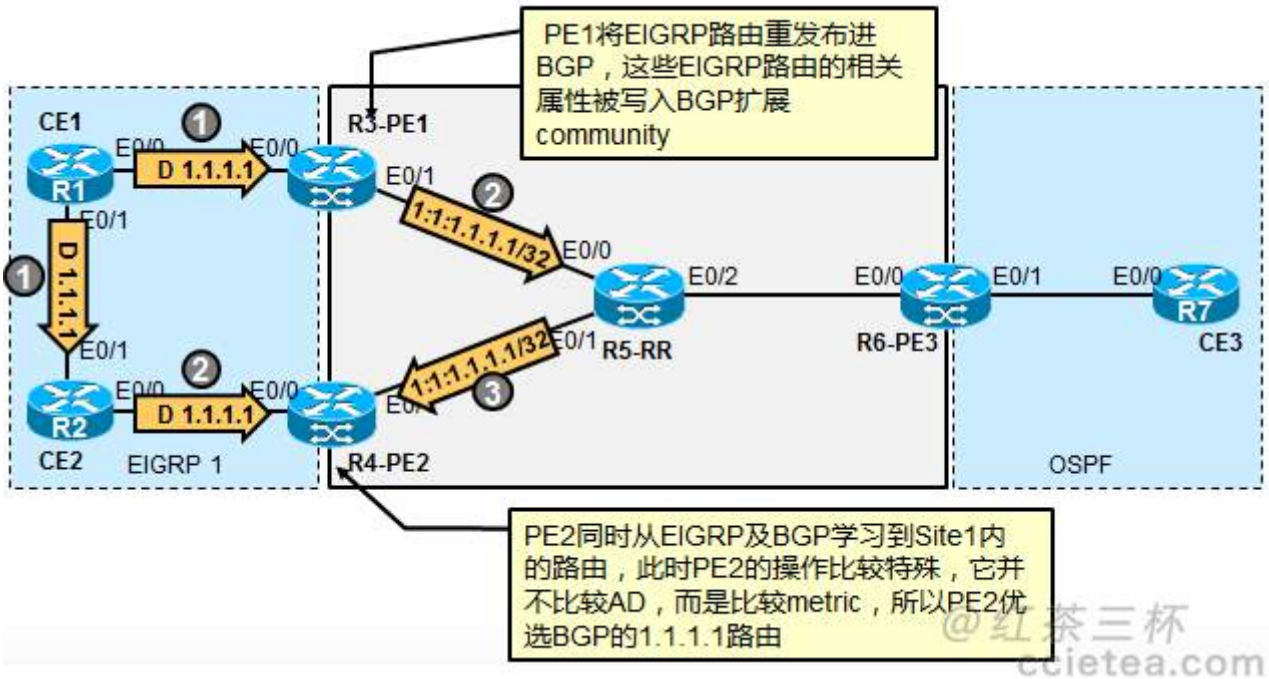


这个实验，配置基本上与实验 1 一样，只不过，Site1 内的 PE-CE 协议换成了 EIGRP，AS 号使用 1。

8.1.2.2 实验现象

1. **测试 1：Site1 使用 EIGRP，Site2 使用非 EIGRP**

   这个环境中 Site2 的 PE-CE 路由协议使用的是 OSPF。



- **首先我们看看 Site1 内的路由传递问题。**

  拿 1.1.1.1 举例 ,CE2 及 PE1 都能学习到这条 EIGRP 内部路由。那么在 CE2 及 PE1 上 ,1.1.1.1 为 EIGRP

内部路由 ,且 Metric 都是 537600。现在 R2 将这条路由更新给了 PE2 ,那么 PE2 上 路由的 metric 是 563200。

另一方面，PE1 上，由于路由表里已经有了 EIGRP 路由 1.1.1.1 且 metric 为 537600，因此这条路由在 EIGRP 到 BGP 的重发布过程中，被注入到了 BGP，而这条路由的一些个 EIGRP 的特性，注如带宽、负载、延迟、AS 号等等都被放进 BGP 路由的扩展 community 中与前缀一并传递给 RR。RR 将路由反射给了 PE2。

那么现在，关键问题就在 PE2 上，PE2 将同时从 EIGRP 以及 BGP 学习到这条路由。PE2 会怎办？回想一下如果这里 PE-CE 运行的是 OSPF 那么它会去比较 OSPF 及 BGP 的 AD ,最后 OSPF 胜出 ,对吧？ 但是这里情况不一样了，这里 PE2 不会去比较协议的 AD，而是 Metric，很好玩吧？

为什么会这样？这是因为，Site1 内的 EIGRP 路由，被 PE1 将路由注入到 BGP 后，这些 EIGRP 路由的许多特性，都很好的被保存在 BGP 的扩展 community 中，跟随着路由一起被运载到了 PE2 上，那么 PE2 就可以根据这些扩展 community 去还原 EIGRP 路由。

R3-PE1#sh ip b vpnv4 all 1.1.1.1

```
BGP routing table entry for 1:1:1.1.1.1/32, version 1072
Paths: (1 available, best #1, table cisco)
  Advertised to update-groups:
     4
  Local
    10.1.13.1 from 0.0.0.0 (3.3.3.3)
      Origin incomplete, metric 409600, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:3456:12 Cost:pre-bestpath:128:409600
        0x8800:32768:0 0x8801:1:153600 0x8802:65281:256000 0x8803:65281:1500
        !! 丰富的扩展 community 值很好的保护了 EIGRP 路由的原始生态特征
      mpls labels in/out 313/nolabel
```

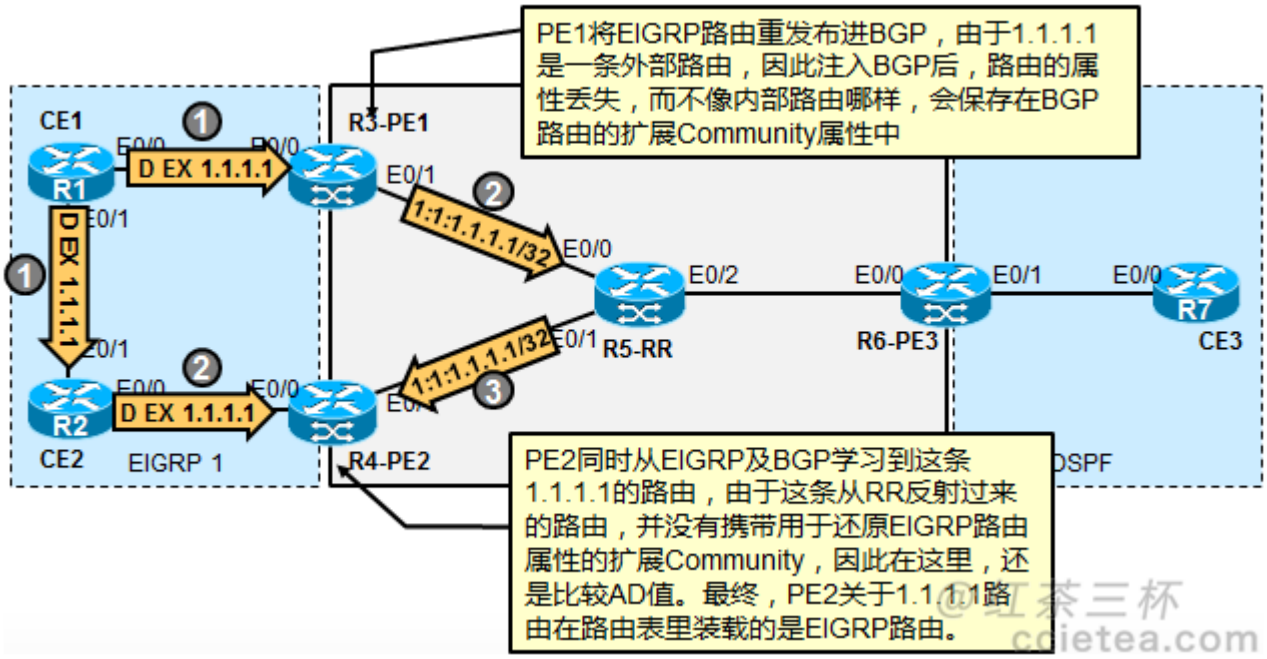
因此，对于 EIGRP 来说，此刻 MPLS VPN Backbone 就相当于一个透明的玩意儿。所以，PE2 虽然同时从 EIGRP 及 MP-iBGP 收到 1.1.1.1/32，它不会去比 AD，而是比较 Metric，好了，CE2 将 EIGRP 路由更新给 PE2，PE2 走 CE2 去到 1.1.1.1 的 metric 是 563200，而从 MPLS VPN 这一侧通过 PE1 走，Metric 是 537600，因此最终，PE2 优选 BGP 的 1.1.1.1 路由放进了路由表。

这样一来就出问题了 ,对于 PE2 而言 ,去往 1.1.1.1 走的是 MPLS VPN Backbone 这条次优路径 ,而且，更郁闷的是由于这条路由在 PE2 的 VRF 路由表里是 BGP 的，而 PE2 又部署了 BGP 与 EIGRP 的双向重发布，因此路由又被倒灌回了 EIGRP，这也是我们非常不想见到的。

解决的办法，可以考虑在 PE1 及 PE2 上，BGP 进程里过滤掉 Site1 内的路由而防止自己从 MP-iBGP 再学习到这些路由，这种方法有个潜在的问题是，如果 CE1-CE2 之间的链路断掉的话，那么 CE1-CE2 间的连通性就丢失了。另一个用于解决该问题的方法是采用 SOO (具体请见本文档 PE-CE 间运行 EIGRP 协议的相

关章节），同样的，也存在刚才说的这个缺陷。

好，接下去换一个角度看看：



如果 R1 始发出来的 1.1.1.1 路由不是 EIGRP 内部路由，而是直接重发布直连产生的路由。

那么情况就大不一样了。

这条路由首先是分别被 PE1 和 PE2 学习到，那么假设我们先在 PE1 上做的 EIGRP 到 BGP 的重发布，那么这条路由又通过 MP-iBGP 传递给了 PE2，所以，症结又在 PE2 这了，PE2 会如何优选？注意，此时 PE2 不再比较 metric 了，而是比较 AD 值，所以 PE2 路由表里装的 1.1.1.1 路由，还是 EIGRP 的。

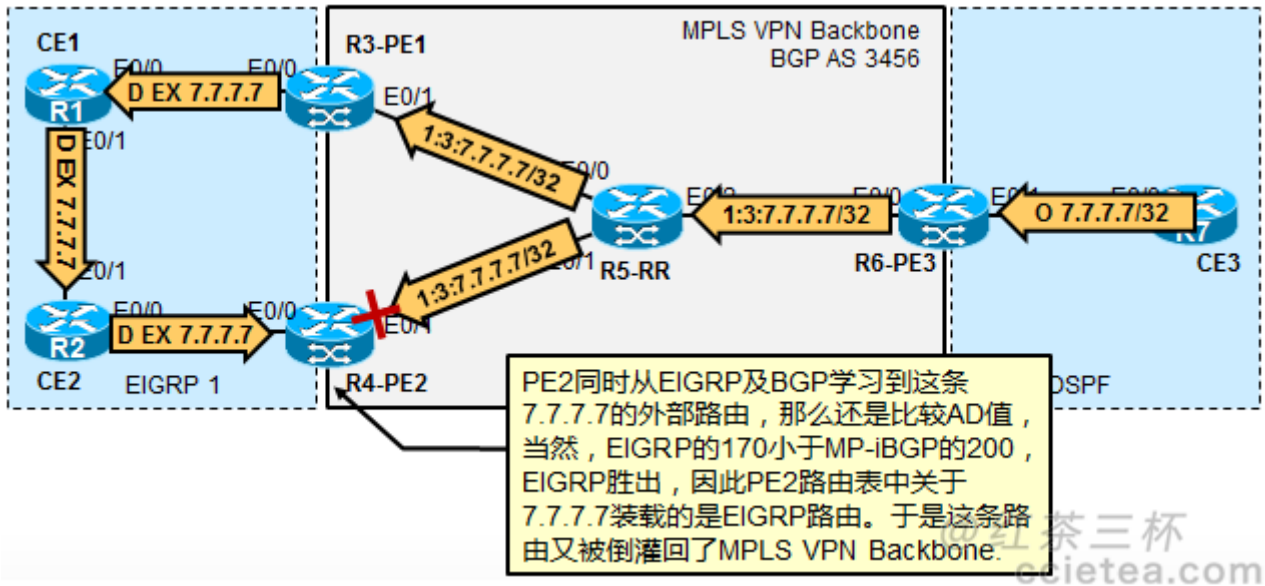
为啥？卧槽一会比 AD 一会又不比玩儿我呢么？其实都是有道理的，我们看看 PE1 发出来的 VPNv4 前缀：

R3-PE1#sh ip b vpnv4 all 1.1.1.1

```
BGP routing table entry for 1:1:1.1.1.1/32, version 1049
Paths: (2 available, best #2, table cisco)
    10.1.13.1 from 0.0.0.0 (3.3.3.3)
      Origin incomplete, metric 537600, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:3456:12          !! 扩展 Community 里只有 RT
      mpls labels in/out 300/nolabel
```

我们发现，PE1 学习到的这条 EIGRP 外部路由，在注入 BGP 后，并没有像内部路由那么，将各个参数保存到 BGP 的扩展 Community 中，这样的一个直接的后果是，在路由被 RR 反射到 PE2 那，PE2 完全没有足够的信息将路由还原成 EIGRP 路由。因此，既然这样，PE2 就当它是条纯的，从 MP-iBGP 学习到的路由，自然而然的，就与另一侧的 EIGRP 去 PK 路由协议的 AD 值，最终 EIGRP 外部 170 的 AD 还是胜出了。

- **接着我们再看看 Site2 的路由传播问题**



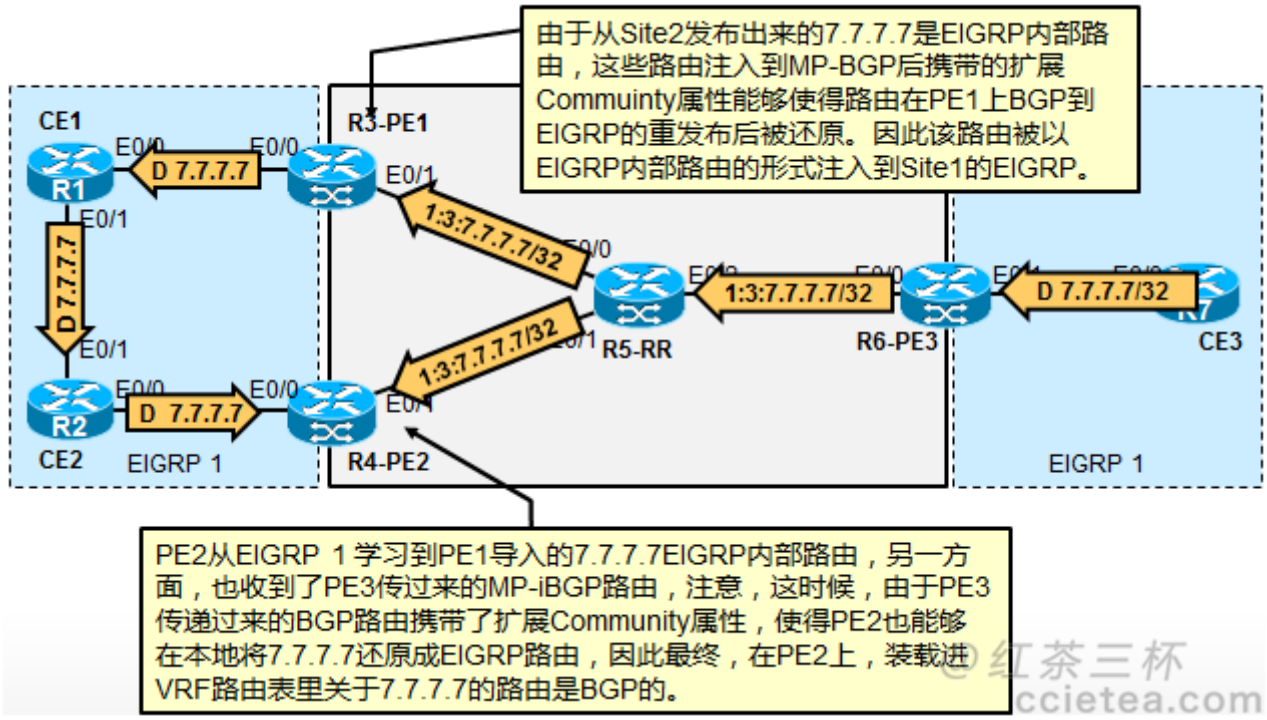
现在，我们再来考虑 Site2 的路由到 Site1 的传播过程。

假设 PE1 先做的重发布 那么 7.7.7.7 的路由被注入到了 EIGRP 中 ,由于这条路由的来源是 OSPF 协议，因此路由是以外部路由的方式注入到 EIGRP 中。这条 EIGRP 外部路由最终更新给了 PE2。而 PE2 又从 MP-iBGP 学习到了这条路由。这里，毫无疑问是比较 AD 值的，以为是外部路由啊，前面我们已经说过了。那么最终 PE2 的路由表里，关于 7.7.7.7 存放的就是 EIGRP 路由。

事情还没完，由于 PE2 上又做了 EIGRP 到 BGP 的重发布，因此这条路由又被倒灌回了 MP-BGP。

解决的办法是，在 PE1 和 PE2 上，将 BGP 路由注入到 EIGRP 时关联一个 route-map，给路由都打上 tag，然后都在 EIGRP 向 BGP 重发布时，将这些打了 tag 的路由 deny 掉，放行其他，即可。

2. 测试 2：Site1、Site2 均使用 EIGRP，且 AS 号一致



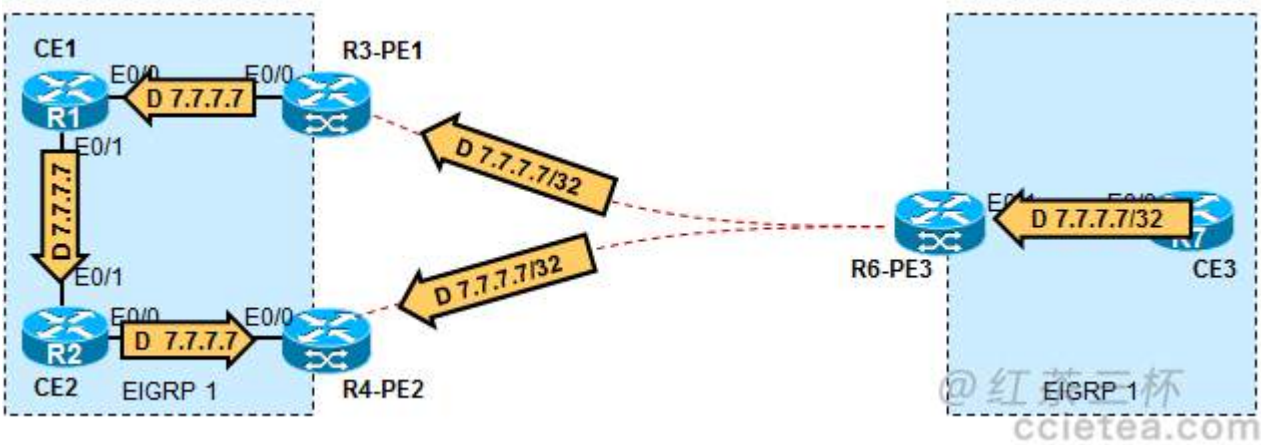
这个环境中，Site2 运行的 PE-CE 协议变成了 EIGRP，且 AS 号也是 1。

我们仍然来考虑 Site2 到 Site1 的路由传递过程。

7.7.7.7 这条 EIGRP 内部路由，被 PE3 注入到 EIGRP 后，形成一条 VPNv4 的前缀，同时，还携带着用于保留路由 EIGRP 特性的那些扩展 Community。路由分别被传递到了 PE1 及 PE2。

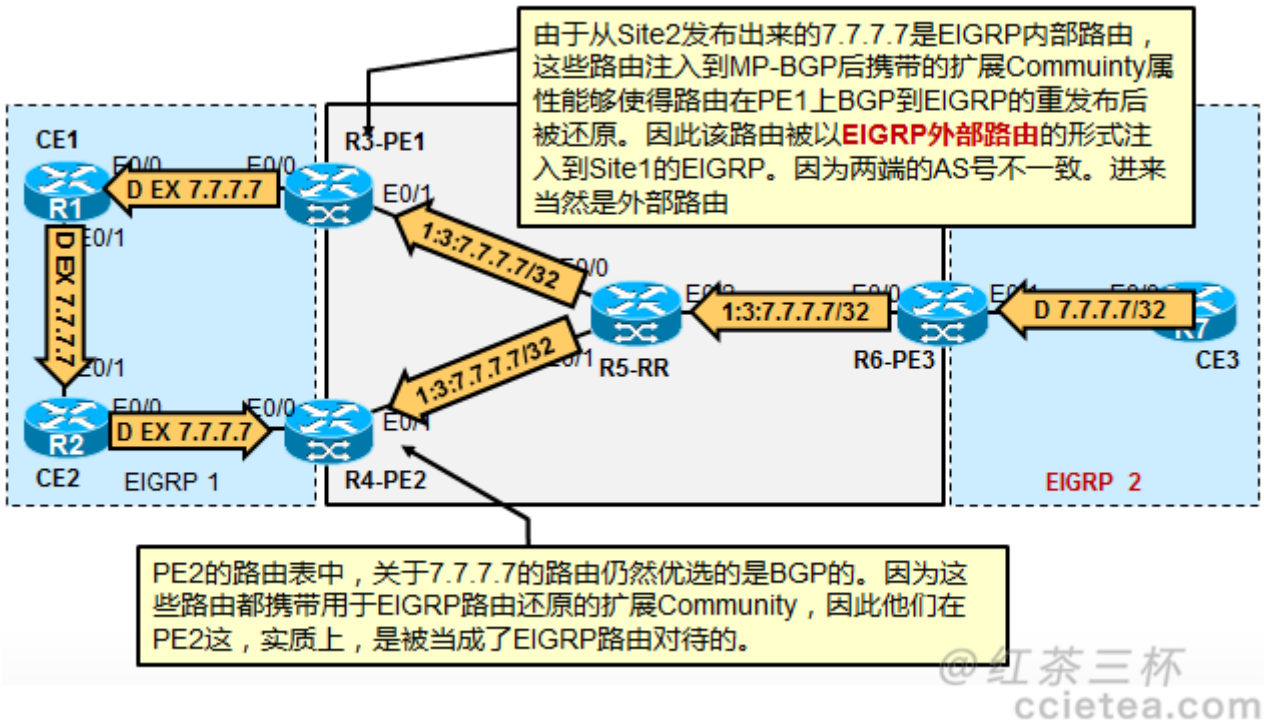
假设我们在 PE1 上先配置的 BGP 到 EIGRP 的重发布，那么些条 BGP 路由，被 PE1 注入了 EIGRP，得益于路由所携带的扩展 Community 属性 这条路由被还原成了一条 EIGRP 内部路由然后放进了 Site1 的 EIGRP 域。这样一来，这条路由就会经过 R2 最终传递给了 PE2。

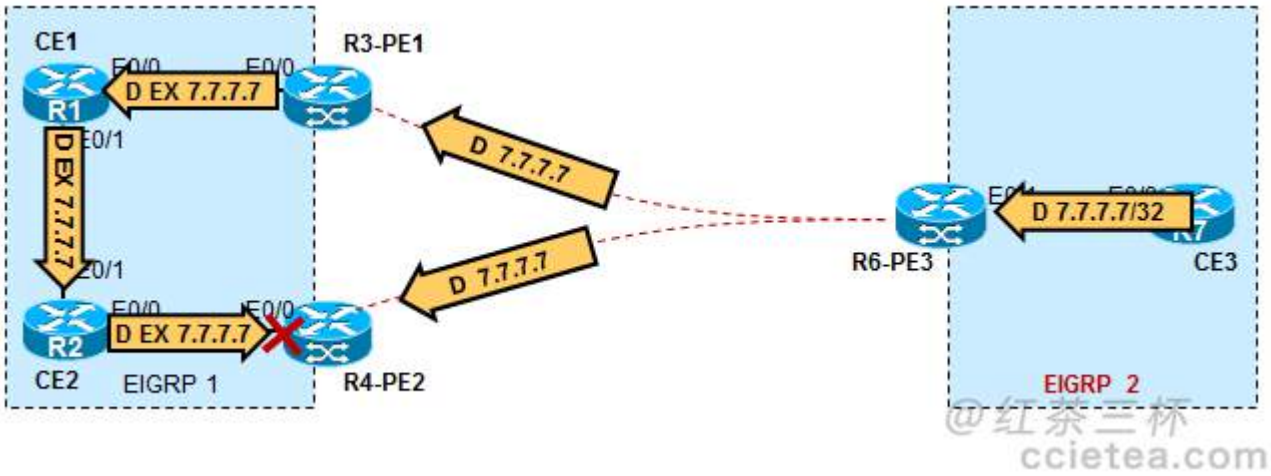
那么现在又该 PE2 纠结了，这哥们一方面，从 EIGRP 学习到了一条内部路由 7.7.7.7，另一方面呢，又从 MP-iBGP 学习到这条路由，咋办？还记得我们前面说过么？这里他不会去比较 EIGRP 和 MP-iBGP 的 AD 值，为什么？因为通过 MP-iBGP 传递过来的这条路由，携带的扩展 Community 能够使得路由在 PE2 上 BGP 重发布到 EIGRP 后被还原成了一条 EIGRP 内部路由，因此，这时候，对于 PE2 来说，MPLS VPN Backbone 等于就是完全透明的，我们可以这么来想象：



那么你说，就这个情况，PE2 该优选哪边到 7.7.7.7？毫无疑问嘛是吧？比较路由的 metric。也就是说，当 EIGRP 路由在穿越 MPLS Backbone 时，如果路由的各项特性能够很好的保存下来，那么此时 MPLS VPN Backbone 对于 EIGRP 来说相当于就是透明的。

3. 测试 3：Site1 使用 EIGRP，Site2 使用 EIGRP，且 AS 号不一致



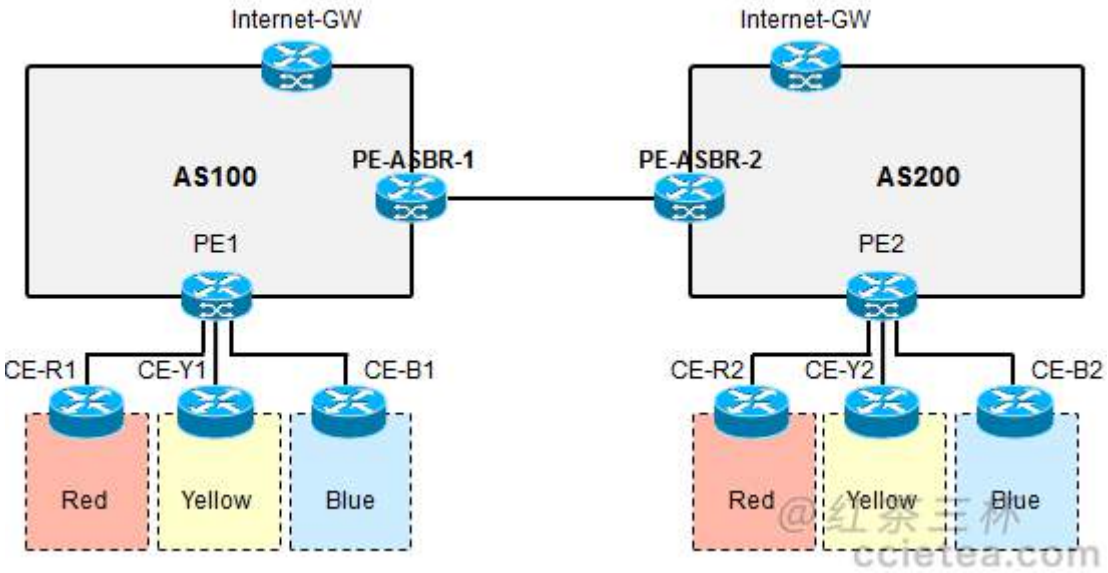


9 Inter-AS MPLS VPN

9.1 概述

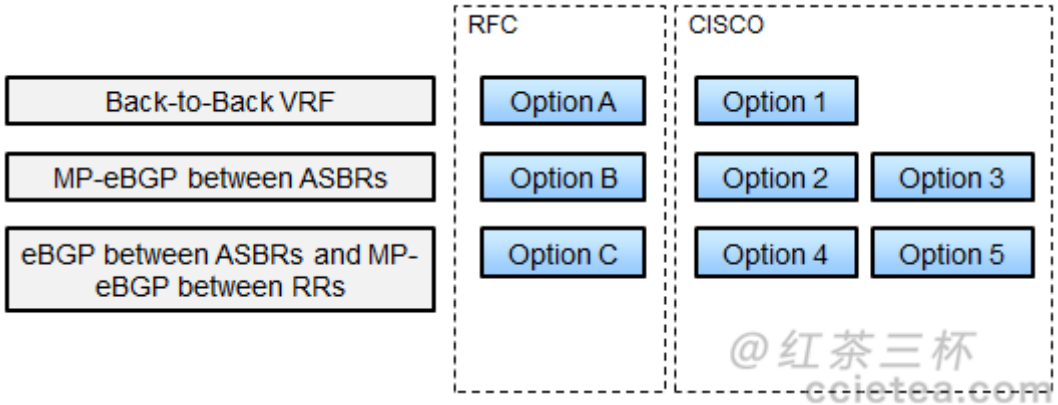
1. **关于域间 MPLS VPN**
   - Inter-AS MPLS VPN 是一种牛逼的解决方案。到目前为止，我们接触的 MPLS VPN 网络中，MPLS VPN Backbone 都是属于一个单一的 AS、单一的 ISP。那么如果我们的 VPN 站点分布在两个不同的 ISP 下属的 MPLS VPN 网络呢？就需要考虑域间的 MPLS VPN 解决方案了。
   - Inter-AS MPLS VPN 是一种基于基础 MPLS VPN 架构的扩展，在 RFC 2547bis 中定义。

2. **Inter-AS MPLS VPN overview**

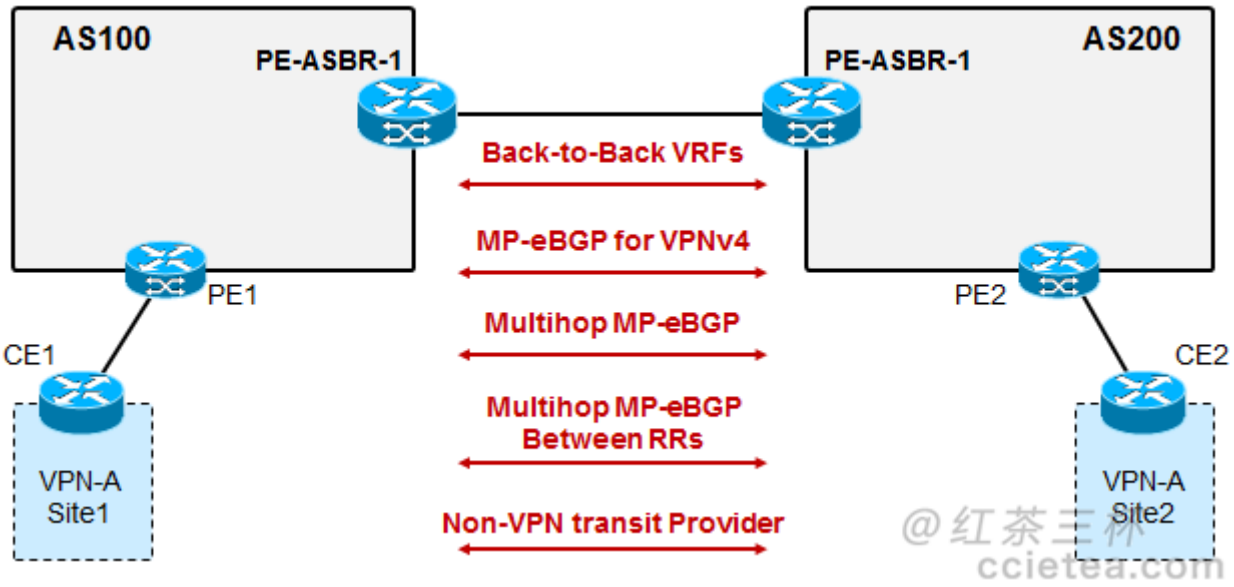


最终的目的非常简单，就是要在两个 AS 之间传递 VPN 的路由。使得各 VPN 客户的不同站点能够互访。

3. Inter-AS MPLS VPN 模型



RFC 2547bis 中定义了 Option A B C，本文档使用 Option1,2,3,4,5 的 CISCO 方式来标记各种模型。

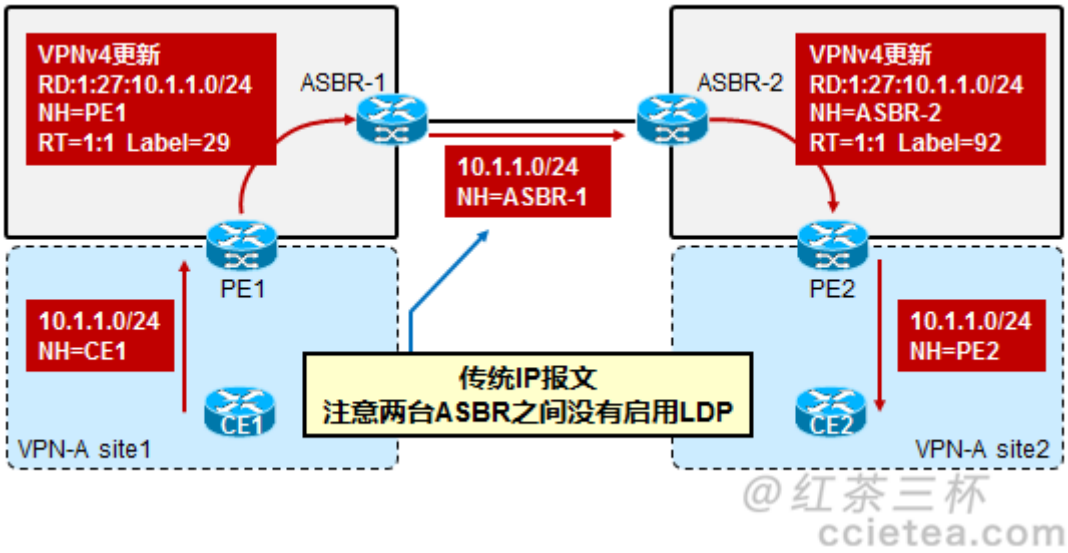


9.2 Option1：Back-to-Back VRF

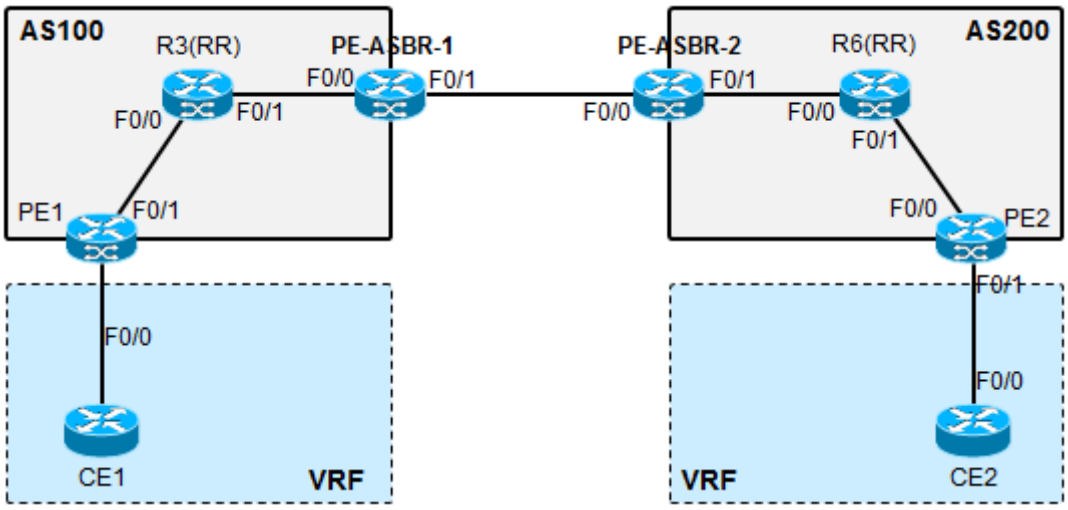
9.2.1 模型解析

1. **概述**
   - 当 PE-ASBRs 之间在一条物理连接的情况下可以考虑部署，并建议只部署在少量 VRFs 要求的环境中
   - 两 PE-ASBRs 只有一条物理连接
   - 两 PE-ASBRs 各自将对方视为一台 CE，两者之间可使用任意一种 PE-CE 动态路由协议
   - 部署时，可以为每个 VRF 创建一个子接口从而实现不同 VPN 的隔离
   - 在两 PE-ASBRs 之间的流量传递的是传统的 IP 流量，两台 ASBR 之间不维护任何 MP-BGP 邻接关系，也不维护任何 LDP 邻接关系。
   - 这种 inter-as MPLS VPN 方案最简单，但是很明显，扩展性非常差，当有大量 VRF 的情况下无法胜任

2. **数据交互过程分析**



9.2.2 实验示例



CE1、CE2 的配置非常简单，这里不做赘述，PE1-CE1 跑的是 EIGRP。PE2-CE2 跑 OSPF。

在实验环境中，设备的编号是从左到右依次为 R1、R2、.....R8，我们的 MPLS 标签空间分配也按照这个标号来，例如 PE1，实际上就是 R2，label range 为 200-299；RR1 为 R3，label range 为 300-399，这样容易看现象。

要注意 PE-ASBR1 及 ASBR2 之间，两边接口都不能运行 LDP。使用一个 OSPF 协议来作为 PE-CE 的 IGP，双方互相将对方视为自己的 VRF 客户

PE1 的配置如下：

```
ip vrf ABC
 rd 100:2
 route-target export 100:2                    !!export 的 RT 为 100:2
 route-target import 100:4                    !!这是导入由 ASBR 的 VRF 导入的、来自 AS200 的客户路由
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
interface FastEthernet0/0
 ip vrf forwarding ABC
 ip address 10.1.12.2 255.255.255.0
!
interface FastEthernet0/1
 ip address 10.1.23.2 255.255.255.0
mpls ip
!
router eigrp 100                              !!PE-CE 间运行 EIGRP 协议
 auto-summary
 !
 address-family ipv4 vrf ABC
  redistribute bgp 100 metric 1500 10 10 1 1500      !! 将 VPNv4 路由注入 EIGRP VRF
  network 10.1.12.0 0.0.0.255
  auto-summary
  autonomous-system 100            !!注意别忘了配置上 EIGRP VRF 的 AS 号，否则邻居无法建立
 exit-address-family
!
router ospf 1                                 !! Core 里的 OSPF
network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
!
```

```
router bgp 100
 no bgp default ipv4-unicast
neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf ABC
  redistribute eigrp 100                    !!将 EIGRP 路由注入 BGP，形成 VPNv4 路由
  no synchronization
 exit-address-family
!
```

RR1 的配置如下：

```
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
!
interface FastEthernet0/0
 ip address 10.1.23.3 255.255.255.0
mpls ip
!
interface FastEthernet0/1
 ip address 10.1.34.3 255.255.255.0
mpls ip
!
mpls label range 300 399
mpls ldp router-id Loopback0
!
router ospf 1                       !! core 里的 OSPF
 network 3.3.3.3 0.0.0.0 area 0
```

```
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 100
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 4.4.4.4 remote-as 100
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 2.2.2.2 route-reflector-client        !! 指定 2.2.2.2 也就是 PE1 为 RR 的 client
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
 exit-address-family
```

ASBR1 的配置如下：

```
ip vrf DEF                                       !!创建一个 VRF，用来与 ASBR（当做 CE）交互路由
 rd 100:4
 route-target export 100:4                       !!将 AS200 的路由导进来，附上 RT100:4 传递给 AS100
 route-target import 100:2                       !!将 AS100，也就是 PE1 发过来的路由导入进 VRF
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.255
!
interface FastEthernet0/0
 ip address 10.1.34.4 255.255.255.0
mpls ip
!
interface FastEthernet0/1
```

```
 ip vrf forwarding DEF                                    !! 连接 ASBR2 的接口放入 VRF DEF
 ip address 10.1.45.4 255.255.255.0
!
mpls label range 400 499
mpls ldp router-id Loopback0
!
router ospf 200 vrf DEF                    !!ASBR 之间互相将对方视为 VRF 客户
 redistribute bgp 100 subnets              !!将 vpnv4 路由注入 OSPF 200，以便 ASBR2 能接收到
 network 10.1.45.4 0.0.0.0 area 0
!
router ospf 1
network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf DEF
  redistribute ospf 200 vrf DEF match internal external 1 external 2      !!将对端 ASBR 传递过来的路由注入
BGP，以便本 AS 的 vpn 客户能够接收
  no synchronization
 exit-address-family
```

其他设备的配置就不在赘述了。配置完成后可以测试一下：

CE1#traceroute 8.8.8.8 so 1.1.1.1

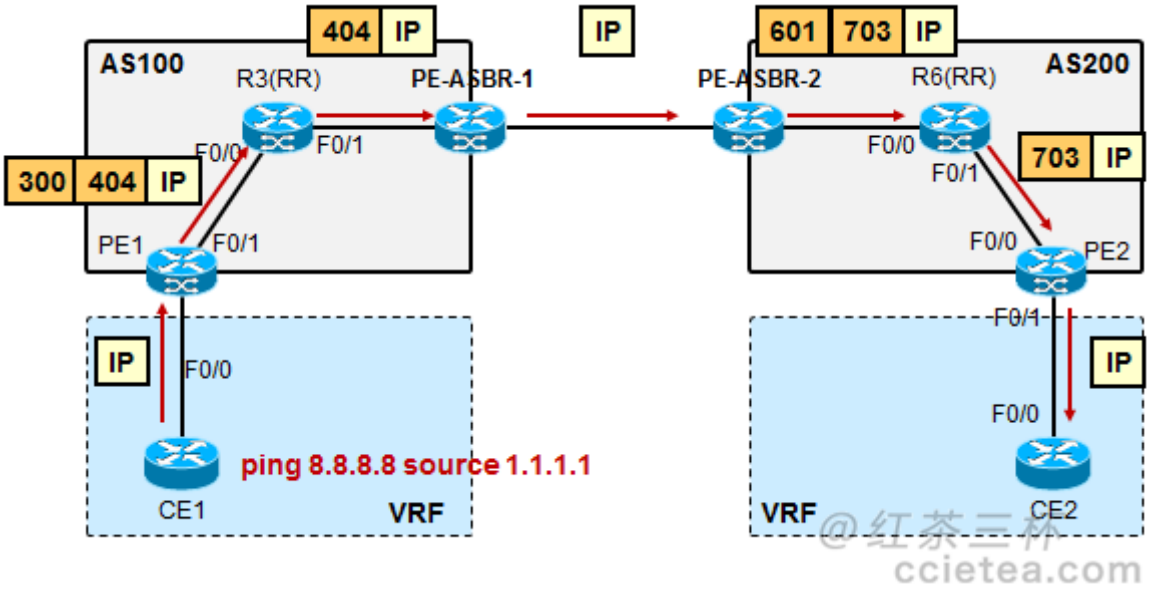
```
Type escape sequence to abort.
```

```
Tracing the route to 8.8.8.8

  1 10.1.12.2 52 msec 4 msec 64 msec
  2 10.1.23.3 [MPLS: Labels 300/404 Exp 0] 28 msec 28 msec 44 msec
  3 10.1.45.4 [MPLS: Label 404 Exp 0] 20 msec 48 msec 24 msec
  4 10.1.45.5 40 msec 12 msec 44 msec
  5 10.1.56.6 [MPLS: Labels 601/703 Exp 0] 56 msec 12 msec 32 msec
  6 10.1.78.7 [MPLS: Label 703 Exp 0] 64 msec 36 msec 40 msec
  7 10.1.78.8 72 msec *    100 msec
```

这里还有一个小点需要注意的，就是 CE1 发出来的数据是纯 IP 的，那么到了 PE1，它查什么表呢？来的是 IP 包，必然查的是 CEF 表，因此可以在 PE1 上用：

PE1#show ip cef vrf ABC 8.8.8.0 detail

```
8.8.8.0/24, version 11, epoch 0, cached adjacency 10.1.23.3
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Fa0/1, 10.1.23.3, tags imposed: {300 404}
  via 4.4.4.4, 0 dependencies, recursive
    next hop 10.1.23.3, FastEthernet0/1 via 4.4.4.4/32
    valid cached adjacency
tag rewrite with Fa0/1, 10.1.23.3, tags imposed: {300 404}
```



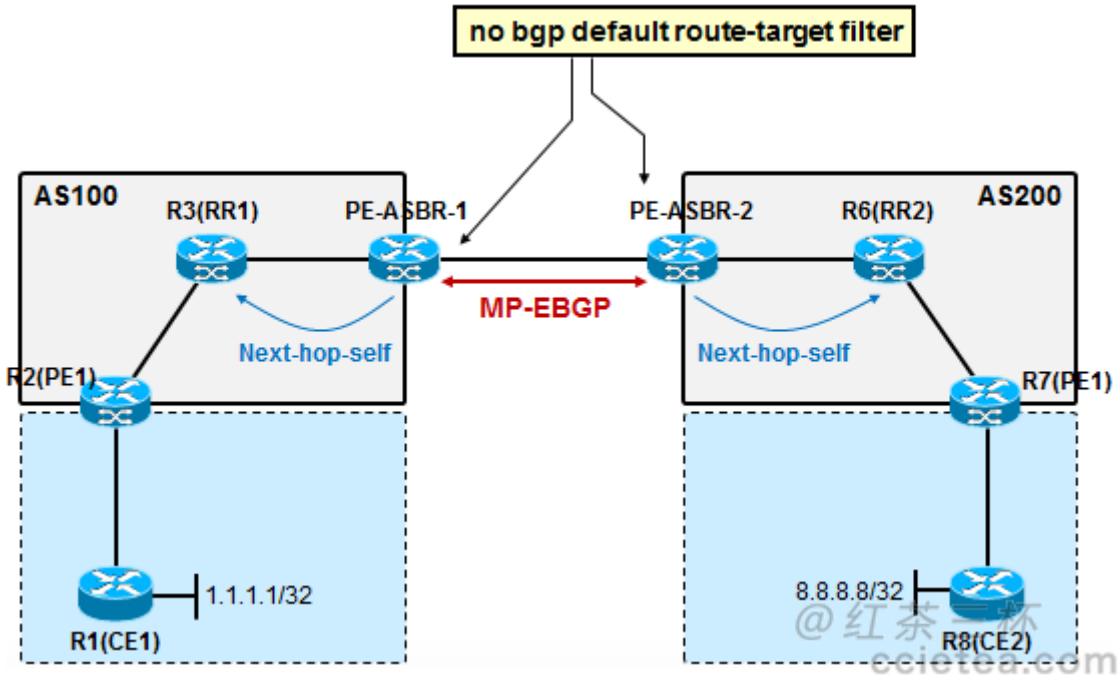
9.3 Option2：External MP-BGP for VPNv4 Prefix Exchange

9.3.1 模型解析

1. 概述

- 当需要支持大量 VRFs 互通的时候，此解决方案优于 OptionA
- 两个 ASBR 之间，建立基于直连接口的 MP-eBGP 邻居关系。
- PE-ASBRs 之间使用 MP-eBGP 直接交互路由，两者之间没有运行 LDP 或 IGP 协议
- PE-ASBRs 对自己本 AS 内的 MP-IBGP 邻居使用 next-hop-self，这会使得标签被重新产生（下一跳发生改变）
- PE-ASBRs 必须保存所有客户的 VPN 路由（使用 no default route-target filter）
- 两个 PE-ASBR 之间会产生一条对端直连 IP 的/32 主机路由

2. 注意事项

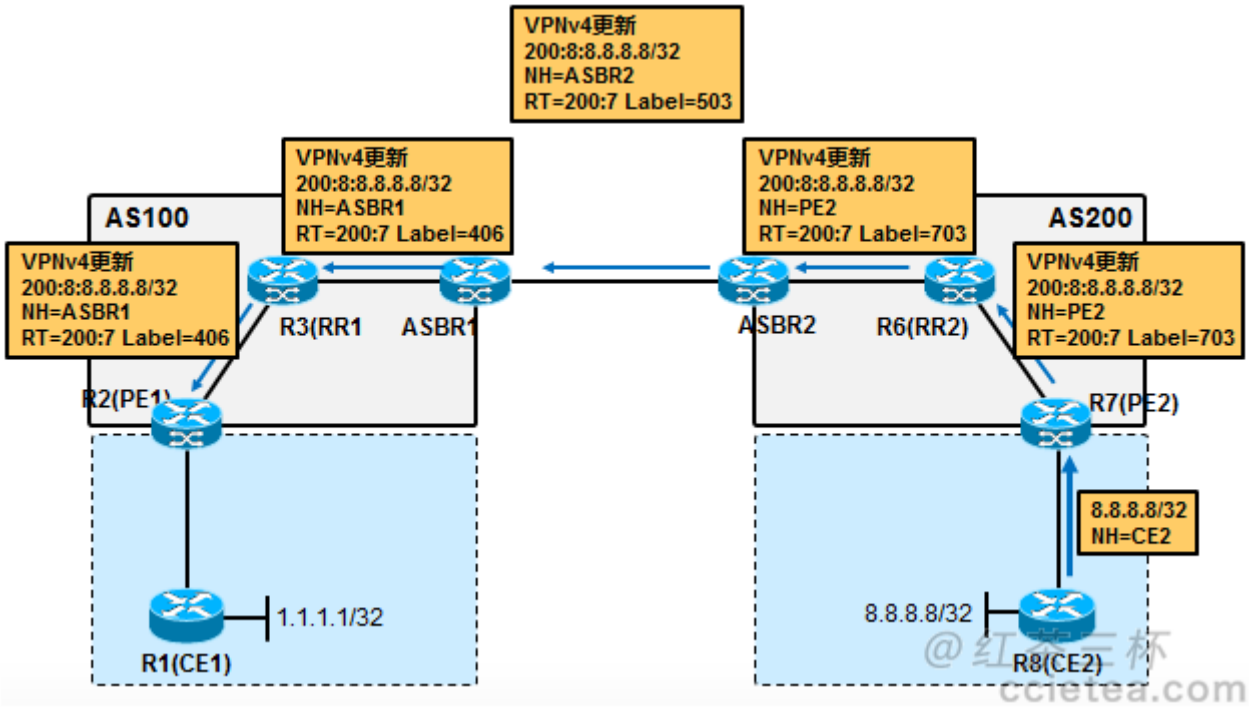


- ASBR-1 及 ASBR-2 之间不运行 IGP 协议，直接在直连口建立 MP-eBGP 邻居关系，且不维护 LDP 邻接。
- PE1 与 RR1 建立 MP-iBGP 邻居关系；RR1 与 ASBR1 建立 MP-iBGP 邻居关系，这些邻居关系都建立在 Loopback 接口上。AS200 内的情况类似这里不再赘述。
- 传统的办法是两 ASBR 之间的链路并未宣告进 AS100 及 AS200 内，那么 ASBR 上就需要对自己本 AS

内的 RR 做 next-hop-self 了。如此一来，VPNv4 标签在域间传递过程中，改变了两次。

- PE1 上，VRF RT export 100:2；PE2 上 VRF RT export 200:7
- 要实现两个 CE 能够互通。

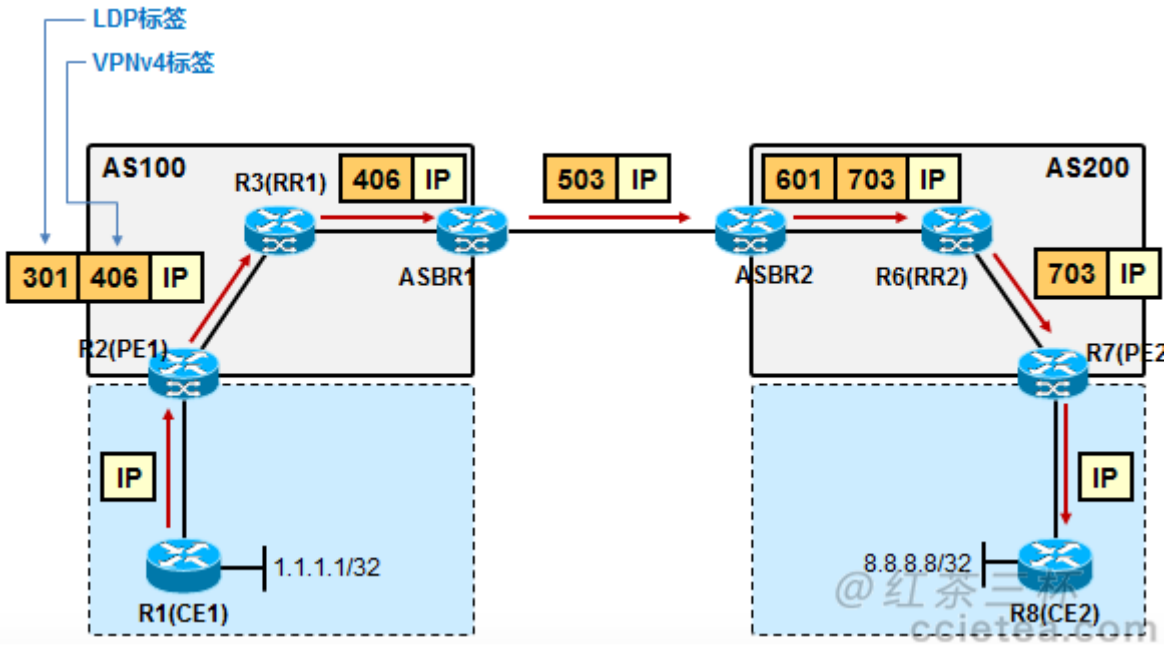
3. VPNv4 前缀的传递过程



- 我们重点关注 8.8.8.8/32 这条始发于 CE2 的路由。
- 这条路由显示被 PE2 学习到，下一跳为 CE2。PE2 将路由装载进 VRF 路由表
- IPv4 前缀 8.8.8.8/32 加上 200:8 的 RD 值 ,构成了 VPNv4 前缀 ,连同一些相关属性被一并更新给了 RR2。其中前缀的 NEXT_HOP 是 PE2 ,RT=200:7 ,标签是 703 ,这个很明显 ,是由 R7 也就是 PE2 的 MPBGP 为该 VPNv4 前缀分配的。
- VPNv4 前缀被传递到了 RR2，然后反射给了 ASBR2
- ASBR2 收到路由后，更新给了自己的 MP-eBGP 邻居 ASBR1
  现在这要格外关注，由于 ASBR1 及 ASBR2 是 MP-eBGP 邻居关系，因此 ASBR2 将 VPNv4 前缀更新给 ASBR1 的时候，NEXT_HOP 就发生了改变，变成了 ASBR2 的接口地址，于是 NH 改变，VPNv4 的标签也要变，所以 VPNv4 前缀被更新给 ASBR1 的时候，NH=ASBR1，Label=503，很明显，标签是由 R4 也就是 ASBR2 的 MP-BGP 来分配的。
- 接下去 ASBR1 将 VPNv4 前缀更新给 RR1 ,由于这个实验环境中 ,我们在 ASBR1 上 ,对本 AS 内的 RR 做了 Next-hop-self，因此它更新给 RR1 的 VPNv4 前缀 NH 也发生了改变，变成了自己的 Loopback 口地址 ,NH 改变 VPNv4 前缀标签也要跟着变 ,所以标签变成了 406。这条 VPNv4 前缀被更新给了 RR1。
- RR1 收到 VPNv4 前缀后，将路由反射给 PE1

- PE1 在本地部署了 VRF，并且 import 了 200:7，于是路由被放进了 VRF 路由表。最终被重发布动作注入了 OSPF 从而被 CE1 学习到了。

4. 数据层面的传输过程



上面是路由层面的分析，下面我们来看数据层面的分析，我们来分析一下，R1 ping 8.8.8.8 的过程。

- **R1 始发的数据当然是纯 IP 的：**

  SrcIP：1.1.1.1　DstIP：8.8.8.8

- 数据包传递到 PE-R2 后，查看的是 VRF 的 CEF 表：

  R2-PE1#show ip cef vrf cisco detail

  8.8.8.8/32, epoch 0, flags rib defined all labels
  　recursive via 4.4.4.4 label **406**　　**!!内层标签**
  　　nexthop 10.1.23.3 Ethernet0/1 label **301**　　**!!外层标签**

  从 VRF 的 CEF 表我们看到，目标网络 8.8.8.8，下一跳是 4.4.4.4，递归下一跳是 10.1.12.3。

  R2 给数据包压上两层标签：

  【301】【406】 SrcIP：1.1.1.1　DstIP：8.8.8.8

  其中，301 是 R3（RR1）为 4.4.4.4 这个地址分配的 LDP 标签。406 是 ASBR1 分配给 VPNv4 路由 200:8:8.8.8/32 的 VPN 标签。

- 数据包到了 RR1，RR1 看自己的 LFIB 表，发现入标签 301 的处理动作是 PoP，这里实际上是有个 PHP 机制(因为 RR1 的出站标签是 ASBR1 给的 ,而下一跳地址 4.4.4.4 又是 ASBR1 的直连网段 ,因此 ASBR1 分派的标签是 POP)，因此 RR1 将顶层标签 301 弹出，然后将数据丢给下一跳 ASBR1：

  【406】 SrcIP：1.1.1.1　DstIP：8.8.8.8

- 上面这个数据包到了 ASBR1，ASBR1 也是查看自己的 MPLS 转发表，发现 406 需要置换成 503，而这个 503 是 ASBR2 分配给 VPN 路由的标签，那么为什么这里标签会发生改变呢，是因为我们分别在 R4 及 R5 上对自己 AS 内的 MP-IBGP peer 做了 next-hop-self，下一跳发生改变了，标签自然也就发生了变化。还记得么？MPLS 是不会为 BGP 路由分配标签的，而是为 BGP 路由的下一跳分配标签。因此，ASBR2 将 406 置换成 503，然后转发给 ASBR2：

  【503】 SrcIP：1.1.1.1 DstIP：8.8.8.8

  PS：这里写点小插曲：在 ASBR1、ASBR2 之间，维护着基于直连接口的 MP-EBGP，他们会互相在本地建立一条到对端直连接口 IP 的/32 位的、直连主机路由，例如 ASBR1 的路由表中，会有一条 10.1.45.5/32 的路由，ASBR2 类似。

  那么为什么要生成这条/32 的主机路由呢？假设拿 ASBR1 做观察点，首先查看 ASBR 的 LFIB：

  R4-ASBR1#sh mpls for

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
406	503	200:8:8.8.8.8/32	1338	Et0/1	10.1.45.5
. …					

  LFIB 中的条目指示，如果它收到一个外层标签值为 406 的标签包，会将标签 406 替换成 503，紧接着，得看 LDP 标签啊，看下一跳 10.1.45.5 的标签，结果由于 ASBR1 上 45.0 这个接口又没有激活 LDP，那么 ASBR1 将这个数据 untag 露出里头的 IPv4 报文，送到 ASBR2 的时候，丫就傻逼了。而现在在本地产生了 10.1.45/32 的直连路由的话，ASBR1 的 LFIB 表就变了：

  R4-ASBR1#sh mpls for

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
406	503	200:8:8.8.8.8/32	1338	Et0/1	10.1.45.5
405	Pop Label	10.1.45.5/32	0	Et0/1	10.1.45.5

  我们看到，10.1.45.5/32 出现在了 LFIB 里，出站标签是 POP，注意，这个 POP 并不是让 ASBR1 把 VPNv4 标签弹出来，有一点点感觉像是将:LDP 标签弹出但这回人家也没 LDP 标签，所以这感觉，就有点像弹空气，any way，反正比 untagged 靠谱多了是吧？

  好吧，最终，ASBR1，将下面这个标签包传递给了 ASBR2

  【503】 SrcIP：1.1.1.1 DstIP：8.8.8.8

- ASBR2 收到上面的数据包后，也是查看自己的 MPLS 转发表：

  R5-ASBR2#show mpls forwarding-table detail

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop

```
503          703          200:8:8.8.8.8/32 1764          Et0/1          10.1.56.6
             MAC/Encaps=14/22, MRU=1496, Label Stack{601 703}
             0E00003010000E0000300F108847 00259000002BF000
             No output feature configured
```

于是 ASBR2 将标签包转成：

【601】【703】 SrcIP：1.1.1.1 DstIP：8.8.8.8

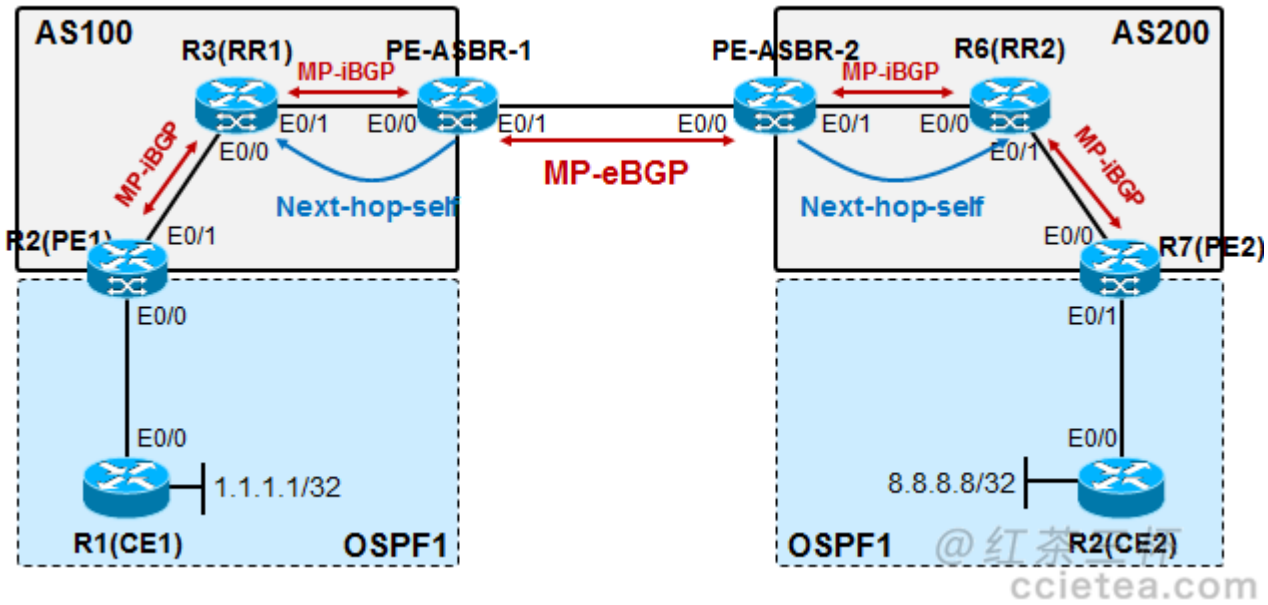
注意，这个 703 是由 R7 也就是 PE2 上 MP-BGP 分配的。这是 VPN 的标签。下一跳地址是 7.7.7.7，那么去往 7.7.7.7 ,需要有 LDP 的标签 ,这时候就是用的是 RR2 分配的 LDP 标签 ,也就是 601。最后 ASBR2 将上述标签包丢给 10.1.56.6 也就是 RR2。

RR2 收到标签包后，查找自己的 LFIB 表，根据指示，将顶层标签 POP 变成：

【703】 SrcIP：1.1.1.1 DstIP：8.8.8.8

然后丢给 PE2，那么到了 PE2 就简单了，弹掉顶层标签，然后变成纯 IP 包转发给 CE2。

9.3.2 实验示例



1. **实验环境**

- 设备互联使用 10.1.xy.0/24 的地址空间，其中 xy 为设备编号。
- 所有的设备 loopback0 的采用 x.x.x.x/32 地址空间，x 为设备编号。
- PE1-CE1 及 PE2-CE2 之间运行 OSPF。
- MP-iBGP 邻居关系如图所示。

- PE-ASBR1 及 PE-ASBR2 之间维护基于直连接口的 MP-eBGP 邻居关系，不激活 LDP。

2. 设备配置

CE1 和 CE2 的配置这里就不说了，起个 OSPF，宣告 Loopback0 即可。

PE1 的配置如下：

```
ip vrf cisco
 rd 100:1
 route-target export 100:2
 route-target import 200:7                    !!这是 PE2 的 export RT 值
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls label range 200 299
mpls ldp router-id Loopback0
!
interface Ethernet0/0
 ip vrf forwarding cisco
 ip address 10.1.12.2 255.255.255.0
interface Ethernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 1 vrf cisco
 redistribute bgp 100 subnets
 network 10.1.12.2 0.0.0.0 area 0
!
router ospf 100                              !! Backbone 内的 IGP
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
```

```
 neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family vpnv4
      neighbor 3.3.3.3 activate
      neighbor 3.3.3.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf cisco
      redistribute ospf 1 vrf cisco match internal external 1 external 2
 exit-address-family
```

R3（RR1）的配置如下：

```
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
!
mpls label range 300 399
mpls ldp router-id Loopback0
!!
interface Ethernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
interface Ethernet0/1
 ip address 10.1.34.3 255.255.255.0
 mpls ip
!
router ospf 100
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
```

```
 neighbor 2.2.2.2 remote-as 100
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 4.4.4.4 remote-as 100
 neighbor 4.4.4.4 update-source Loopback0
!
address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 2.2.2.2 route-reflector-client
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
  neighbor 4.4.4.4 route-reflector-client
 exit-address-family
```

R4（PE-ASBR1）的配置如下：

```
interface Loopback0
 ip address 4.4.4.4 255.255.255.255
!
mpls label range 400 499
mpls ldp router-id Loopback0
!
interface Ethernet0/0
 ip address 10.1.34.4 255.255.255.0
 mpls ip
interface Ethernet0/1
 ip address 10.1.45.4 255.255.255.0
!
router ospf 100
network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
```

```
 no bgp default route-target filter                    !!配上这条命令后 ,将会关闭默认的 VPNv4 前缀过滤，
如此一来这台 PE-ASBR 将拥有所有客户的 VPNv4 前缀
neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 10.1.45.5 remote-as 200
 !
address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
  neighbor 3.3.3.3 next-hop-self                       !!对 RR1 做了 next-hop-self
  neighbor 10.1.45.5 activate                          !!ASBR2
  neighbor 10.1.45.5 send-community extended
 exit-address-family
```

R2（PE-ASBR2）的配置如下：

```
interface Loopback0
 ip address 5.5.5.5 255.255.255.255
!
mpls label range 500 599
mpls ldp router-id Loopback0
!
interface Ethernet0/0
 ip address 10.1.45.5 255.255.255.0
interface Ethernet0/1
 ip address 10.1.56.5 255.255.255.0
 mpls ip
!
router ospf 100
 network 5.5.5.5 0.0.0.0 area 0
 network 10.1.56.5 0.0.0.0 area 0
!
router bgp 200
 no bgp default ipv4-unicast
```

```
 no bgp default route-target filter
 neighbor 6.6.6.6 remote-as 200
 neighbor 6.6.6.6 update-source Loopback0
 neighbor 10.1.45.4 remote-as 100
!
address-family vpnv4
  neighbor 6.6.6.6 activate
  neighbor 6.6.6.6 send-community extended
  neighbor 6.6.6.6 next-hop-self
  neighbor 10.1.45.4 activate
  neighbor 10.1.45.4 send-community extended
 exit-address-family
```

RR2 和 PE2 的配置大同小异，这里不再赘述了。

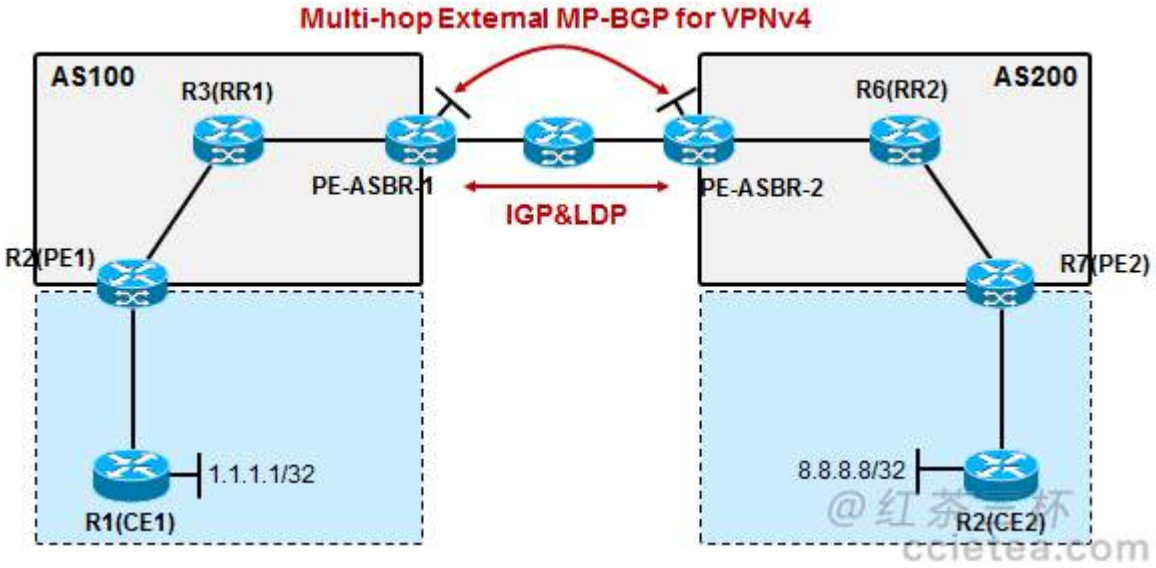
可以再分析一下，如果 ASBR 不对自己本 AS 内的 RR 做 next-hop-self 的话，标签和数据传输的过程。

9.4 Option3：Multi-hop External MP-BGP for VPNv4 prefix exchange

9.4.1 模型解析

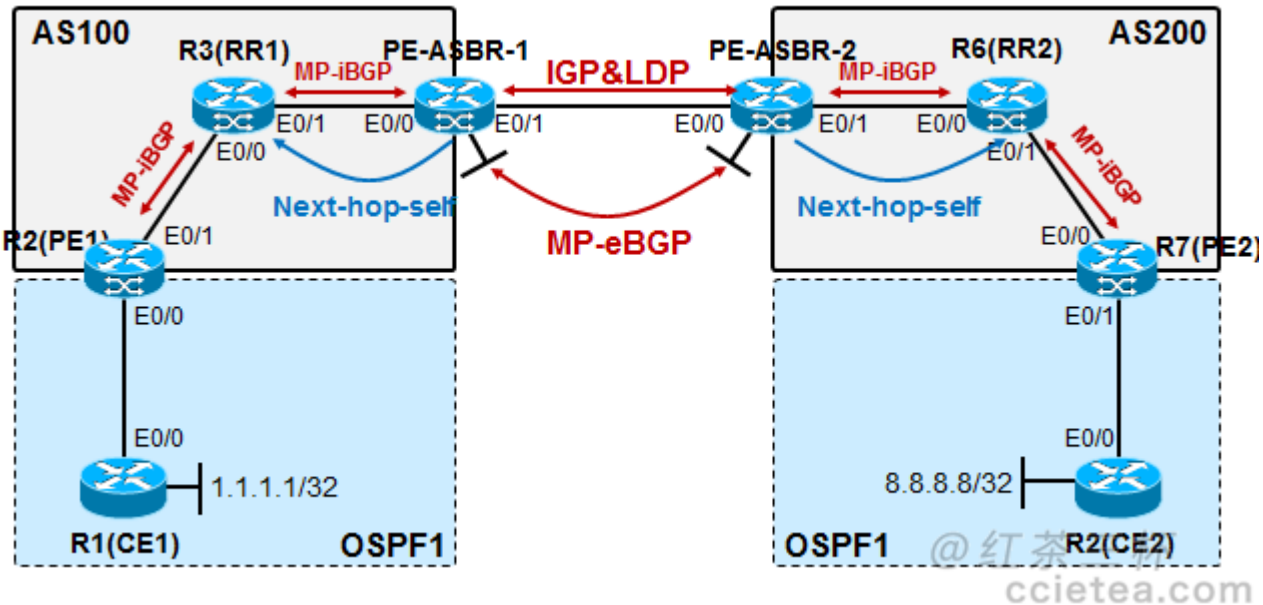
1. **概述**
   - Option3 和 option2 非常类似。
   - 两台 ASBR 之间仍然是建立 MP-eBGP 邻居关系，只不过这里不再是基于直连接口建立的邻居关系了。
   - PE-ASBR routers exchange routes across a Multi-hop BGP session
   - External MP-BGP for VPNv4 prefix exchange
   - IGP and LDP required between PE-ASBR routers to maintain the end-to-end internal LSP
   - Can use static routing to interface addresses
   - NO /32 host route created for adjacent PE-ASBR routers.

2. **注意事项**



ASBR1 及 2 之间可能有其他的路由器，也就说并非直连的。那么在 Option3 的解决方案中，两个 ASBR 之间的 MP-eBGP 邻接关系是多跳的，再者为了使得路由在两 ASBR 之间能够传播，中间可能要是一个 IGP+LDP 的环境。最简单的一种应用，就是两台 ASBR 之间的 MP-EBGP 邻接关系建立在各自的 Loopback 接口上。

9.4.2 实验示例



1. **实验环境**

- 设备互联使用 10.1.xy.0/24 的地址空间，其中 xy 为互联设备编号。
- 所有的设备 loopback0 的采用 x.x.x.x/32 地址空间，x 为设备编号。

- PE1-CE1 及 PE2-CE2 之间运行 OSPF。
- MP-iBGP 邻居关系如图所示。
- PE-ASBR1 及 PE-ASBR2 之间维护基于 Loopback0 口的 MP-eBGP 多跳邻居关系。同时在两个 ASBR 之间配置静态路由使得彼此到对方的 Loopback0 路由可达。在者两台 ASBR 的直连接口建立 LDP 邻接关系。

2. 设备配置

关键看两个 ASBR 的配置，其他设备的配置基本和 Option2 的大同小异，这里不在赘述。

ASBR1 的配置如下：

```
interface Loopback0
 ip address 4.4.4.4 255.255.255.255
!
mpls label range 400 499
mpls ldp router-id Loopback0
!
interface Ethernet0/0
 ip address 10.1.34.4 255.255.255.0
 mpls ip
interface Ethernet0/1
 ip address 10.1.45.4 255.255.255.0
 mpls ip                                    !!连接 ASBR2 的接口，要激活 LDP
!
router ospf 100
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
 no bgp default route-target filter
 neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 5.5.5.5 remote-as 200          !! 与 ASBR2 建立基于 Loopback0 的 MP-eBGP 邻接关系
 neighbor 5.5.5.5 ebgp-multihop 2        !! 这条一定要注意
```

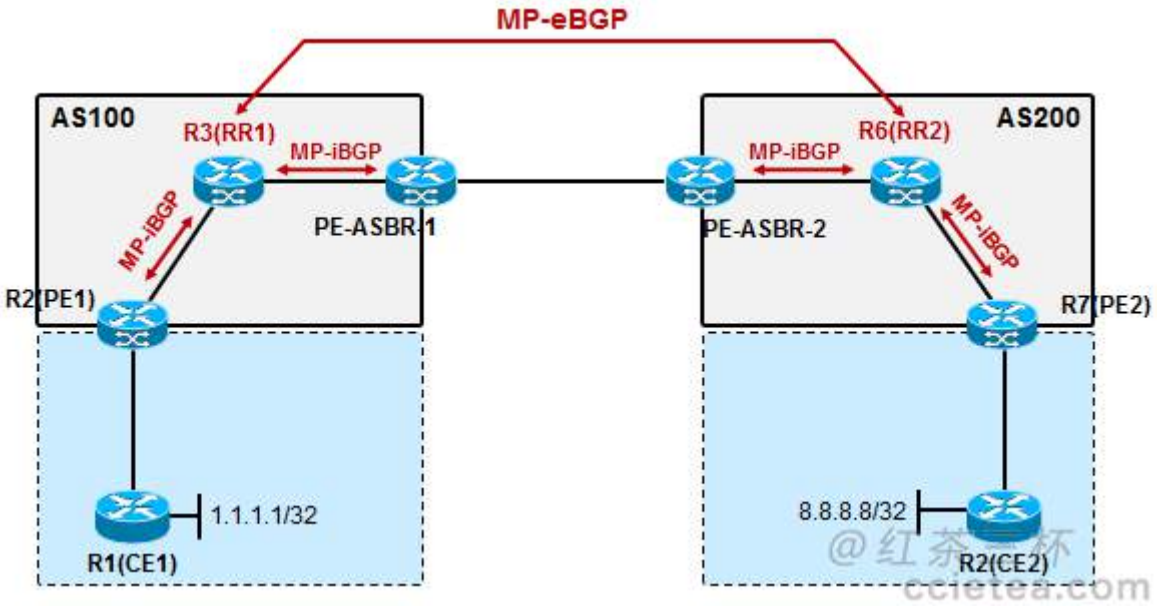
```
 neighbor 5.5.5.5 update-source Loopback0
 !
 address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
  neighbor 3.3.3.3 next-hop-self
  neighbor 5.5.5.5 activate
  neighbor 5.5.5.5 send-community extended
 exit-address-family
!
ip route 5.5.5.5 255.255.255.255 10.1.45.5        !!我们这个实验环境比较简单，ASBR1 和 2 之间没有其他设备了，所以为了让两个 ASBR 彼此都有对方的 Loopback0 口路由，就直接用了静态路由来配置。当然，如果两 ASBR 之间隔了多台路由器，可以考虑运行个 IGP 协议，在者，这些路由器必须都支持 LDP、
```

先来小结一下，Option B 的两种解决方案，也就是前面说的 Option2 和 3，存在什么问题呢？

首先两台 ASBR 是没有 VRF 的 ,并且 ,也不是 RR 的角色 ,为了使得他们都有所有客户的路由 ,我们配置了 no bgp default route-target filter。这对于两台 ASBR 来说，他们既要承担控制平面的压力，处理那么多的 VPNv4 前缀，另外，又要承担数据转发的压力，这俩哥们累啊。那么接着看 Option4 吧。

9.5 Option4：Multihop MP-eBGP for VPNv4 between RRs

9.5.1 模型概述



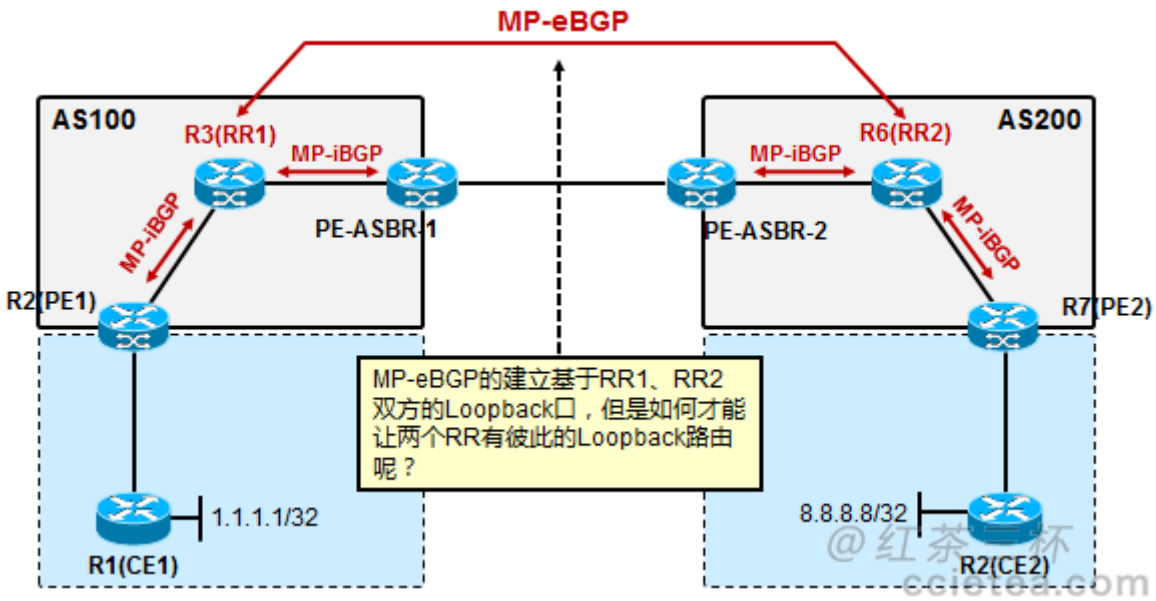
- 对于 RFC2547，其实就是 OptionC。
- 由于 RR1 及 RR2 都有所有的 VPNv4 的路由，因此可以直接在 RR 之间建立 MP-eBGP 邻居。
- **在 RR 之间建立多跳的 MP-eBGP 邻居关系从而直接交互 VPNv4 前缀，ASBR 不再需要维护 VPNv4 前缀，减轻设备负担的同时增强了网络的扩展性**
- **在 RR 上不进行 Next-hop-self 的配置**
  - 一种更优化的解决方案是，保留 VPNv4 前缀的 Next_Hop 为始发的 PE，具体怎么实现看下文。
- **在 ASBR 之间，需要使用 IPv4 eBGP 来交互携带了标签值的 IPv4 路由前缀**
  - 这需要对 BGP 进行功能上的扩展，使用 send-label 关键字
  - 在分发标签的时候，只为 PE 的 Loopback 路由分发标签，因为他们是 BGP 路由的 Next_Hop

9.5.2 模型详解

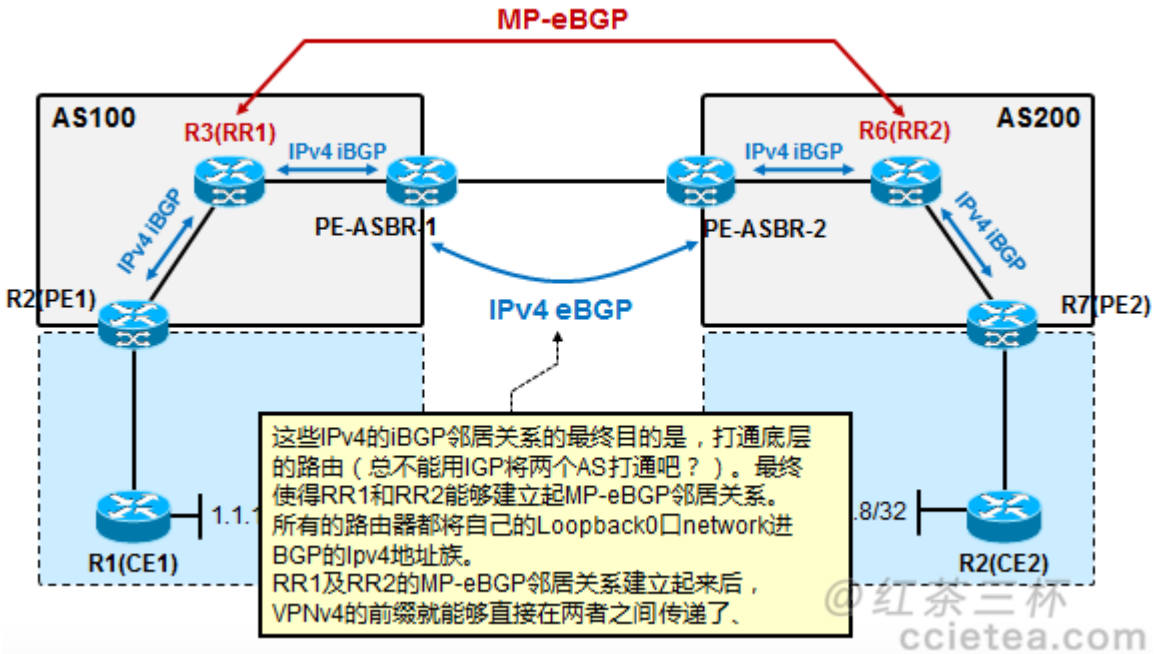
接下去我们来讲讲这个模型中，一些关键的技术点：

1. **RR1/RR2 建立 MP-eBGP 邻居关系**
2. **路由控制层面的理解**
3. **外层标签的分发问题**
4. **数据转发层面的理解**
5. **数据转发层面的优化**
6. **其他可优化的任务**

9.5.2.1 两个 AS 的 RR 建立 MP-eBGP 邻居关系



由于 RR 已经拥有本 AS 内的所有客户路由，因此直接在两个 AS 的 RR 之间建立一个 MP-eBGP 邻居关系，是一个不错的解决方案。那么 RR 之间要建立 MP-eBGP 邻居关系 ,首先两个 RR 得有到达对方 Loopback 接口的路由。



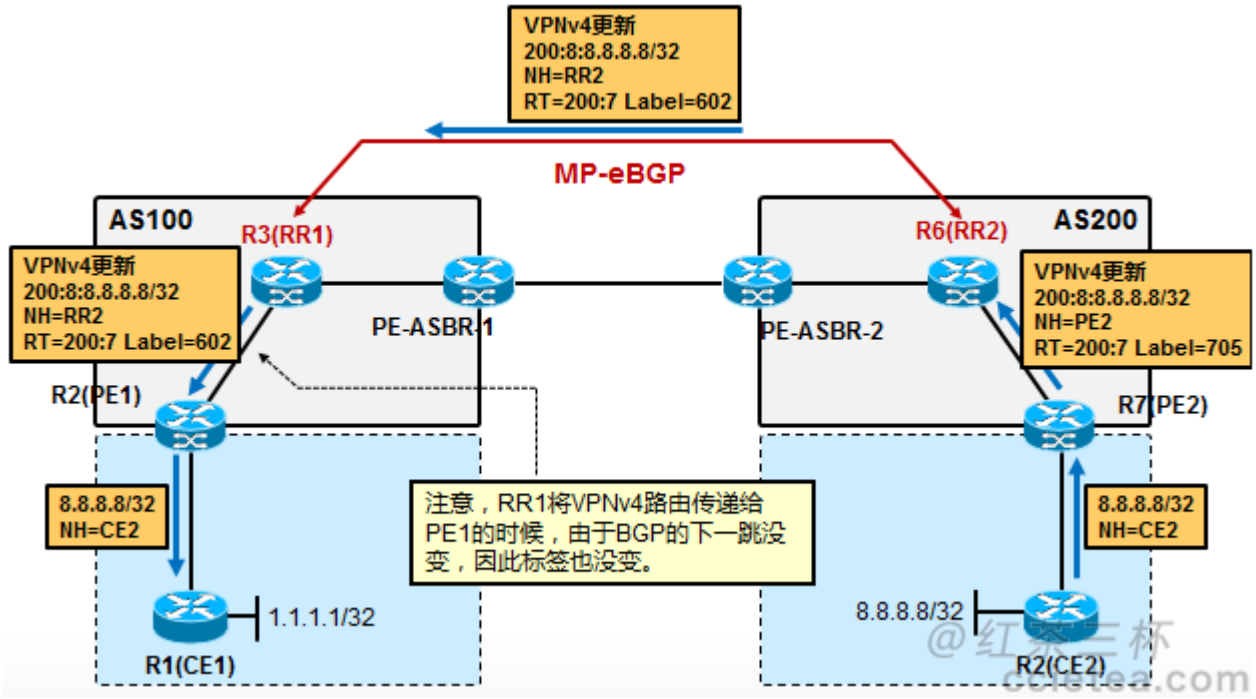
由于 AS100 及 200 内部的 Core IGP 协议，已经有了 Core 内 PE 及 P 路由器（这里的 RR）的 Loopback 路由，我们可能第一反应会想到，打通两个 AS 的 IGP 协议，但是这显然是不合理的，但却可以考虑用 BGP 来打通路由。因此如上图所示。蓝色的箭头表示了 IPv4 的 BGP 邻居关系的部署。

详细看一下，PE1、RR1、PE-ASBR1 都激活 IPv4 地址族并建立 IPv4 的 iBGP 邻居关系，同时宣告自己的 Loopback 口的网段进 BGP。AS200 Backbone 内的三台路由器同理。接下去两个 ASBR 之间维护一个 IPv4-eBGP

邻接，这样，两个 AS 的运营商路由器路由（其实主要就是各个路由器的 Loopback 地址）就都打通了，RR1 及 RR2 彼此通过 IPv4 BGP 学习到了对方 Loopback 的路由，就可以建立 MP-eBGP 的邻居关系了。

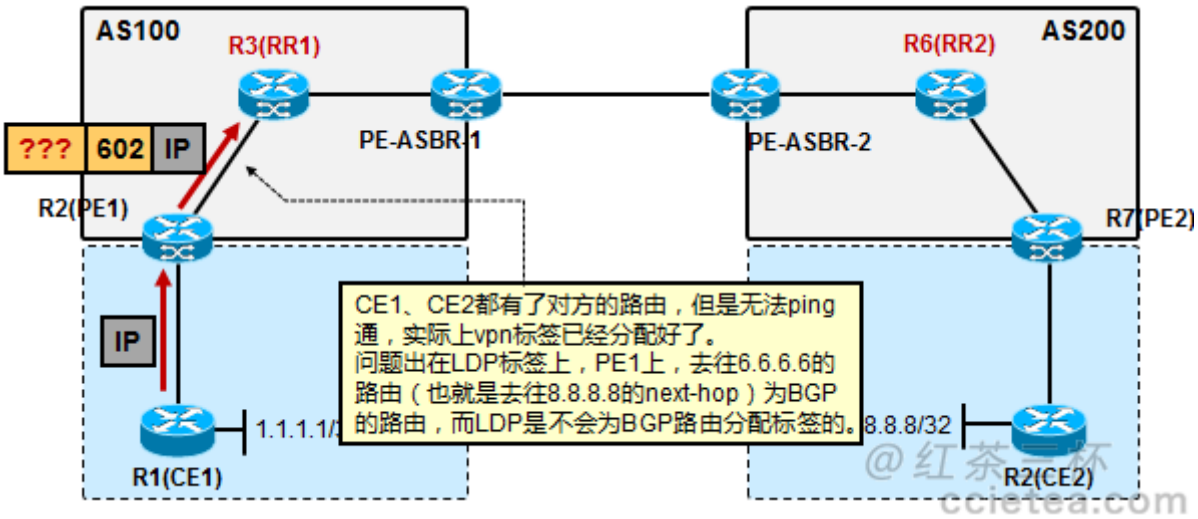
9.5.2.2 路由控制层面的理解

现在我们来看一下，在 RR1 及 RR2 之间，建立起 MP-eBGP 邻居关系之后，路由控制层面的问题。由于 RR 已经有了本 AS 内的所有客户路由，RR1-RR2 之间建立好 MP-eBGP 后，这些客户路由就能够在 RR 的 MP-eBGP 连接上传播了（以 CE2 的路由 8.8.8.8 的传播为例来讲解）：



- CE2 将客户路由 8.8.8.8/32 传递给 PE2
- PE2 通过 PE-CE 路由协议学习到，并注入了 BGP，然后形成 VPNv4 的前缀。这条 VPNv4 前缀是 RD 值 200:8 加上 32 位的 8.8.8.8/32 构成的，NEXT_HOP 是 PE2（的 Loopback 口 IP），RT 值为所设定的 200:7，MP-BGP 为这条前缀分配的标签是 705。这条 VPNv4 前缀被 PE2 传递给了 RR2。
- RR2 由于此刻已经和 RR1 建立了 MP-eBGP 邻居关系，因此它通过这条连接将 VPNv4 前缀传递给了自己的 eBGP 邻居 RR1。注意我们看这条 VPNv4 前缀，NEXT_HOP 现在变成了 RR2，下一跳变了，VPN 标签也要跟着变，所以前缀的标签就变成了 RR2 给该前缀分配的 602。
- RR1 收到了这条 VPNv4 路由，将它传递给 PE1。注意这时候，由于 RR1 收到的是一条 eBGP 路由，因此 NEXT_HOP 地址不变，仍然是指向 RR2（的 Loopback 口地址），既然下一跳不变，VPN 标签也没变，依然是 602。
- PE1 收到了这条 VPNv4 前缀，剥去 RD 值得到 32bits 的 IPv4 前缀，又加上本地 VRF 配置了 Import RT 200:7，因此 BGP 路由被装载进 VRF 路由表，最终被重发布到 PE-CE 的 IGP 路由协议中，从而被 CE1 学习到。

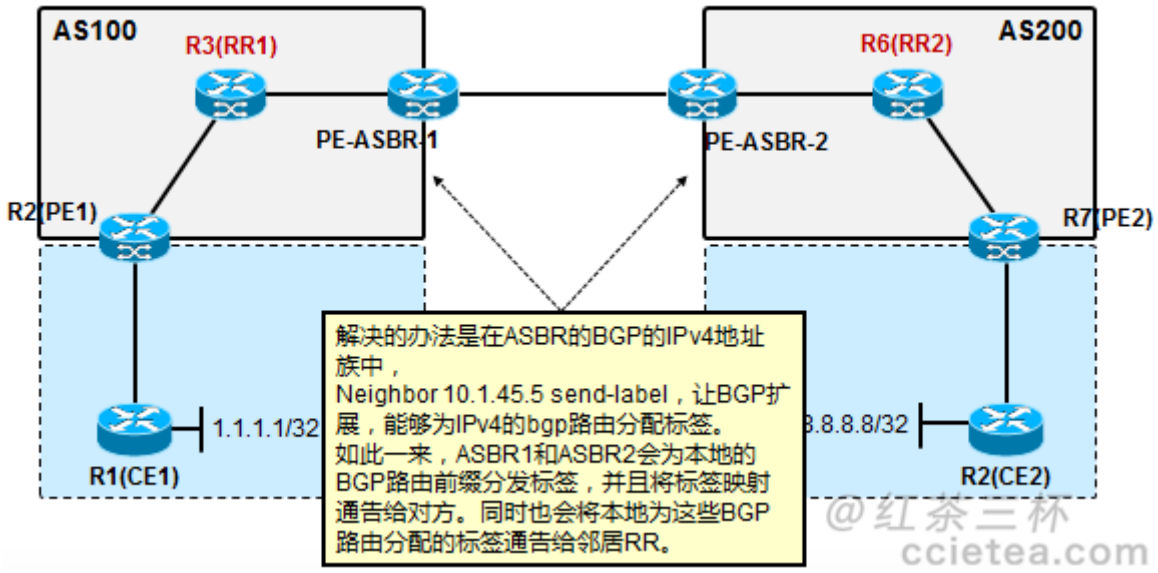
9.5.2.3 外层标签的分发问题



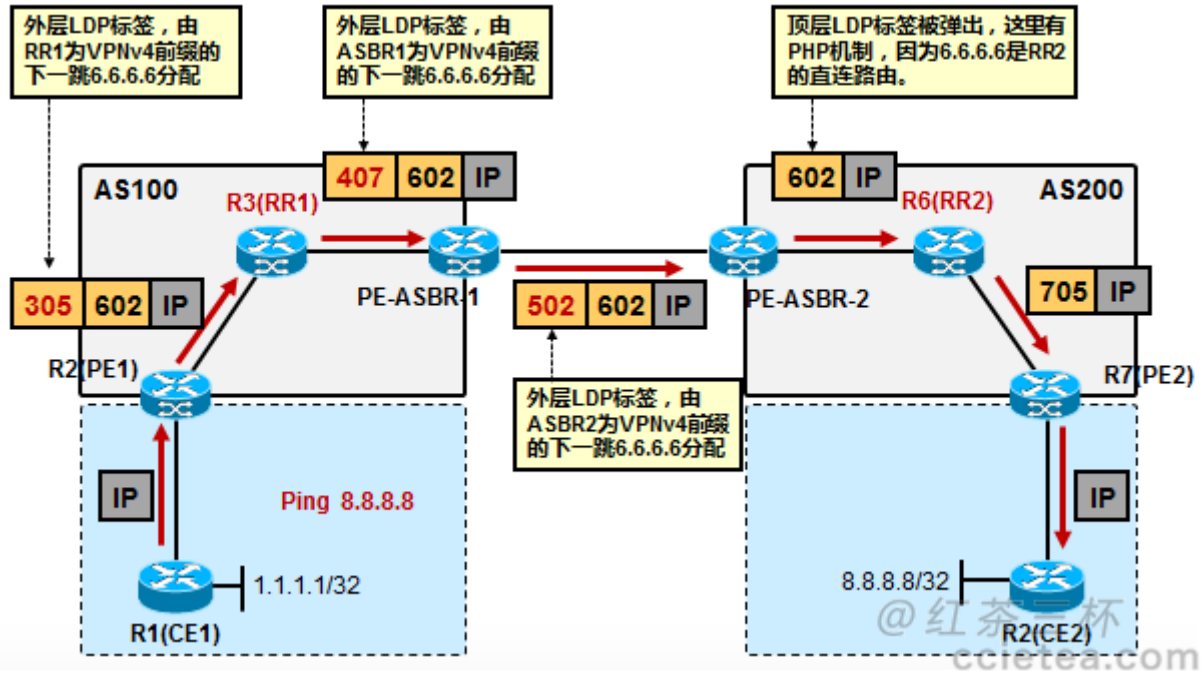
经过前面的配置，CE1 及 CE2 都已经学习到了对方的路由。我们以 CE1 为例，现在假设 CE1 要发送数据包去往 8.8.8.8，那么这个数据包到达了 PE1，首先数据包的目的地 8.8.8.8，已经获取到一个 VPN 标签.602 对吧？这是由 RR2 的 MP-BGP 分配的，只有 RR2 才能理解，那么你要将这个数据包一路传到 RR2，还需要套一层外壳，也就是 LDP 标签吧？这个标签就需要 RR1 来给（实际数据转发的下一跳设备），而需要使用的是 6.6.6.6（NEXT_HOP 地址）的标签。遗憾的是在 RR1 的路由表里，此刻 6.6.6.6 的路由是条 BGP 的路由，因此它不会为 6.6.6.6 路由分配标签，那么数据包在 PE1 这，它就不知道怎么处理了。

为了解决这个问题，我们得让 AS100 的 Backbone 内看到的 AS200 Backbone 内的那些路由（主要是 PE2 及 RR2 的 Loopback 路由）为 IGP 的，只有是 IGP 的路由 LDP 才会分配标签。所以我们在 ASBR1 上将从 IPv4 eBGP 邻居 ASBR2 学习到的 BGP 路由重发布进 OSPF 100。如此一来，在 PE1、RR1 上，获取到的关于 RR2、PE2 等设备的 loopback 路由就是 OSPF 的了，那么 LDP 标签分发就没问题了。

但是，由于 ASBR1 及 ASBR2 是重发布的部署点，在他们路由表里，关于对端 AS Backbone 内的路由可仍然是 BGP 的，于是乎，ASBR1 不会为这些 BGP 路由分配标签并且将映射消息发送给 RR1，那么在 RR1 这里，又是 untagged。在 RR2 那同理。为了解决这个问题，我们得利用一个特性。这个特性通过在 ASBR1 及 ASBR2 的 ipv4 vrf 进程里完成，例如在 ASBR1 上，neighbor 10.1.45.5 send-label，在 ASBR2 上 neighbor 10.1.45.4 send-label。如此一来，ASBR1 和 ASBR2 的邻居关系首先会重建，然后在 Open 消息中会协商新的 ability 值，这样 ASBR1 及 ABSR2 的 BGP 就得到了多协议的扩展。首先的变化是，他们都能够为本地路由表里的 BGP 路由分配标签了，同时也会将为前缀分配的标签值通过扩展的 BGP 在两个 ASBR 之间互相通告。

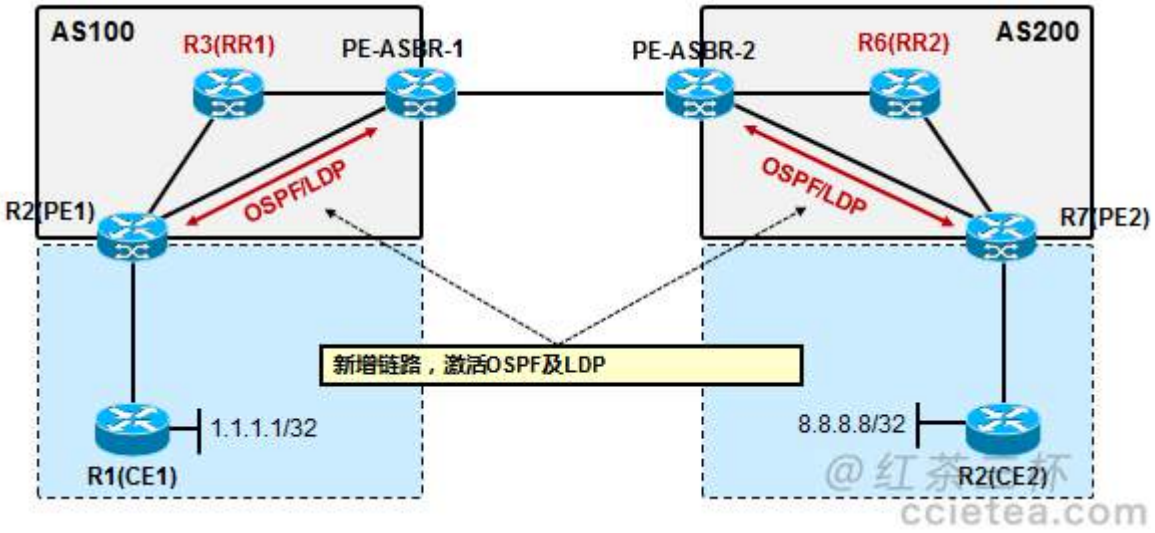


这样一来数据层面的转发工作就完全没问题了：



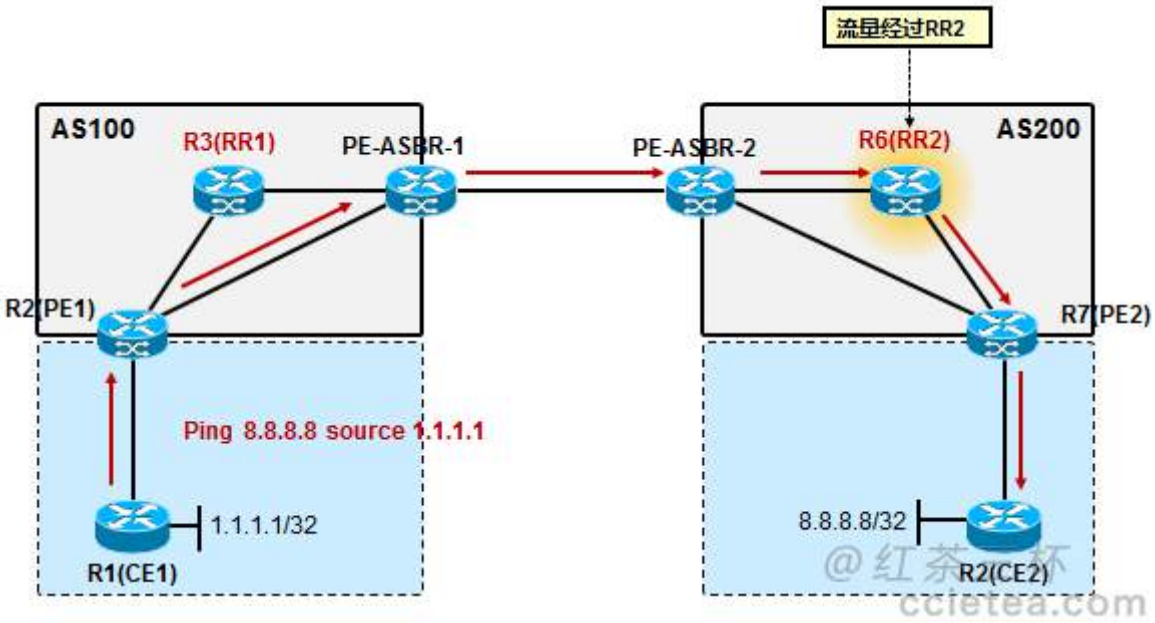
9.5.2.4 数据转发层面的优化

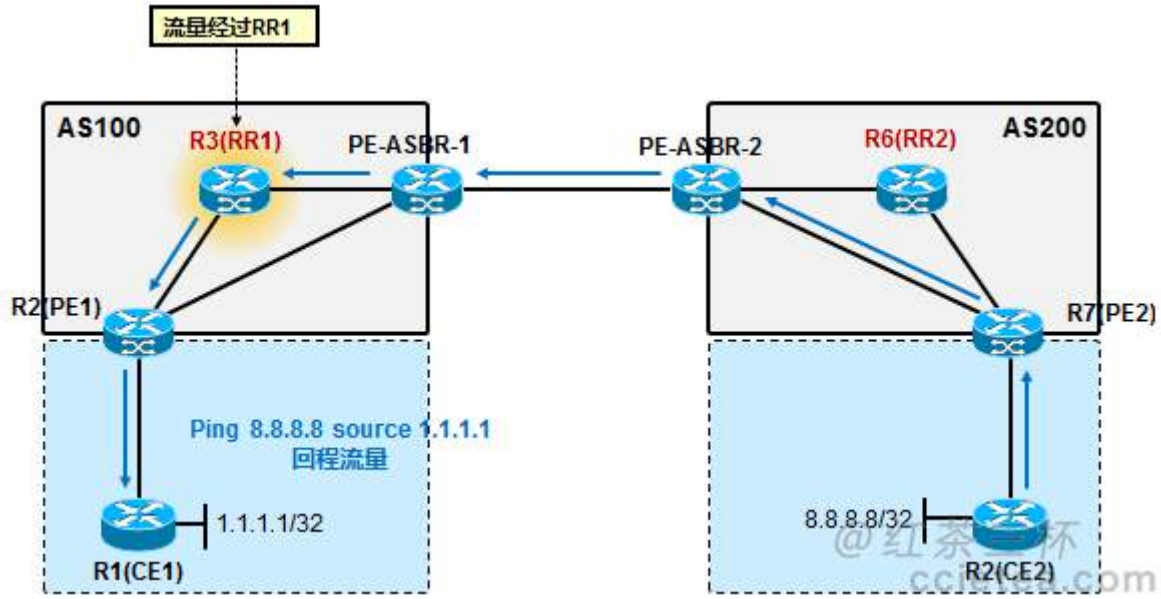
我们来考虑这样一种情况：



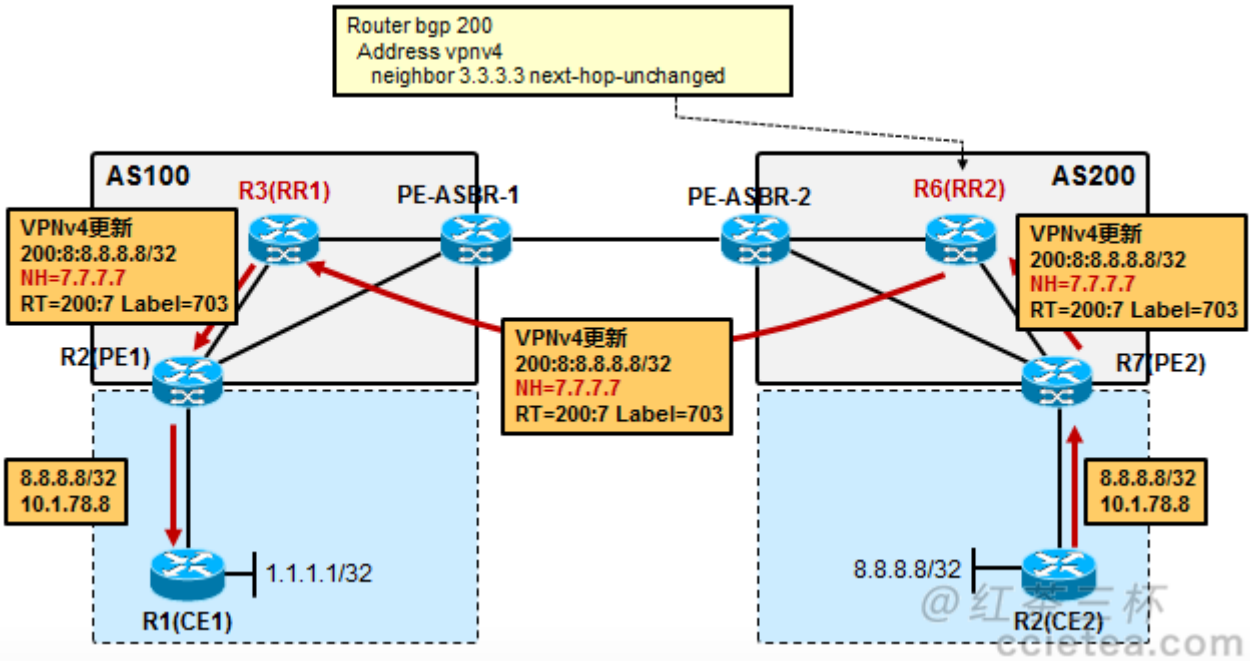
我们增加了两条链路，并且激活了 OSPF 及 LDP。

那么这样一来，当 1.1.1.1 的用户去访问 8.8.8.8 的时候，流量是怎么传输的呢？





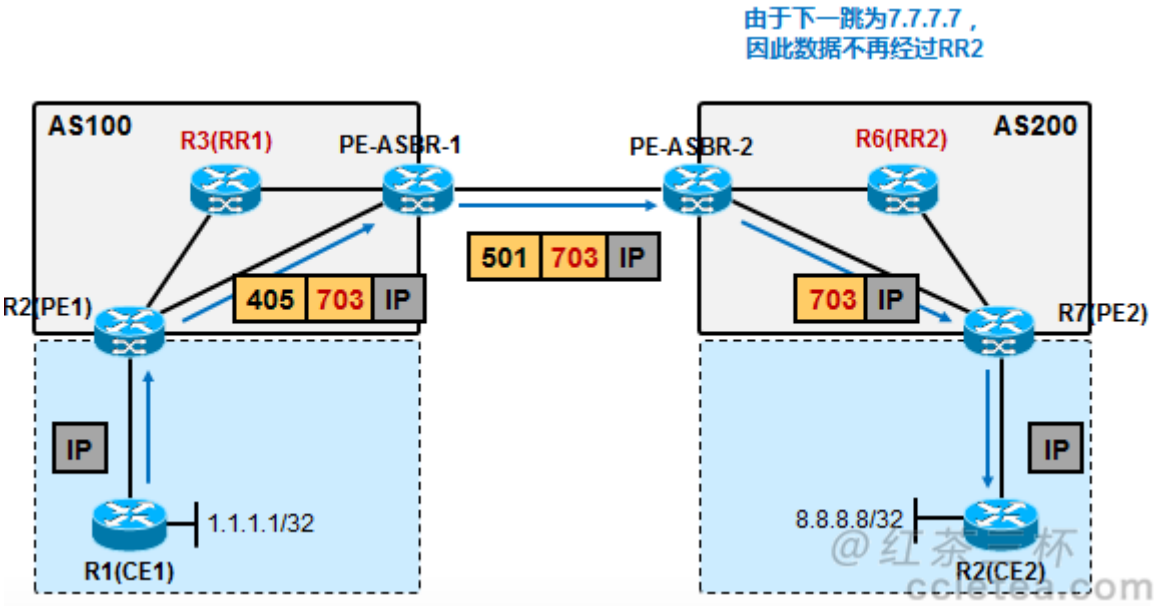
我们看到，事实上，这两个 RR，都在流量往返的过程中，各承担一次数据转发的负担。而从 BGP 网络的设计角度来说，我们更愿意或者说更希望 RR 只是扮演一个路由控制层面的把控角色，而不承担具体的用户数据转发工作。那么实现的方法很简单：



我们在 RR2 上，VPNv4 地址族内做配置：

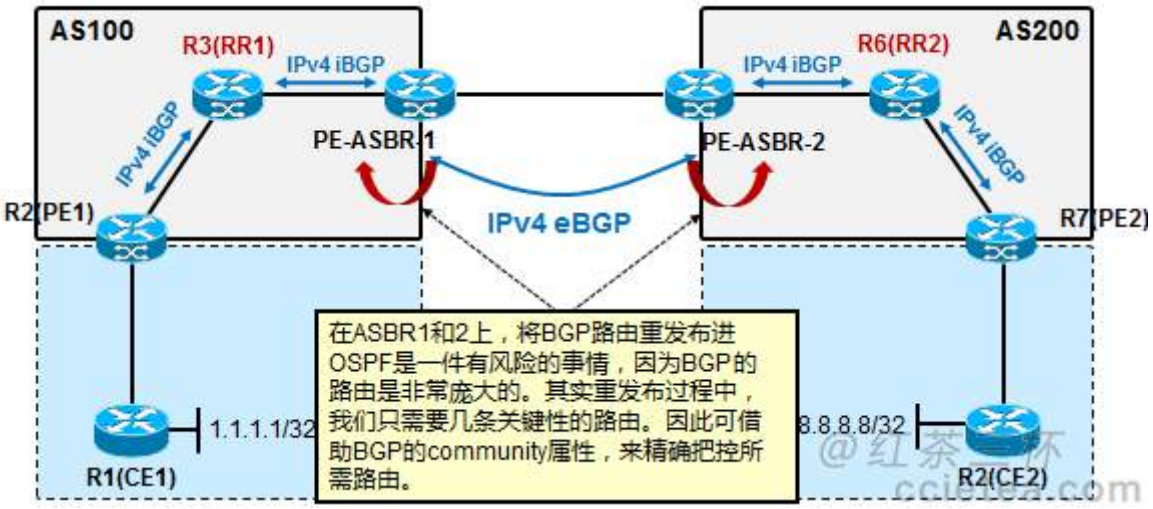
```
Neighbor3.3.3.3 next-hop-unchanged
```

这样一来，RR2 传递 VPNv4 路由给自己的 MP-eBGP 邻居 RR1 的时候，不会改变 VPNv4 前缀的 NEXT_HOP 属性，依然保持为 7.7.7.7（始发这条 VPNv4 路由的 PE2）。那么这样一来：



9.5.2.5 其他可优化的任务

1. 控制重发布到 Backbone IGP 内的 BGP 路由

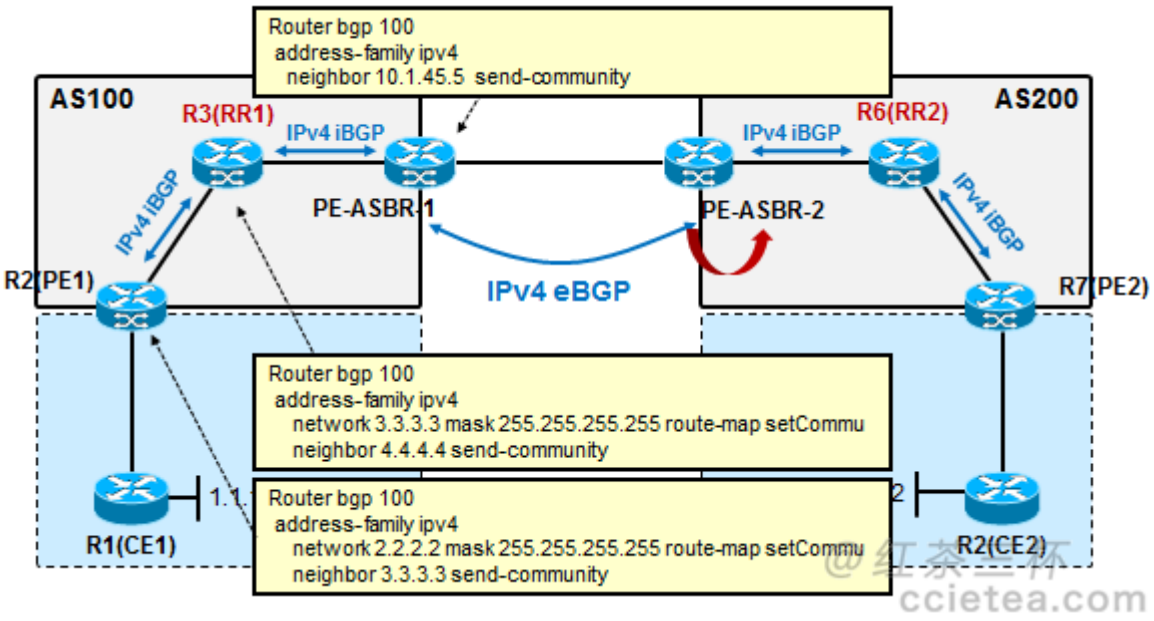


在前面的讲解中，为了让 AS100 及 AS200 内，拥有对端 AS 内关键设备的、IGP 的 Loopback 路由，使得 LDP 能够良好的工作，我们在 ASBR1 及 ASBR2 上，将 BGP 路由重发布进了 OSPF 100。这是存在风险的，因为 BGP 携带的路由前缀往往是非常庞大的。另一方面，实际上，譬如说 AS100 内的路由器，它们真正需要的也就是 RR2 的 Loopback6.6.6.6 的路由，以及 PE2 的 Loopback7.7.7.7 的路由，其他路由在本环境中意义不大。

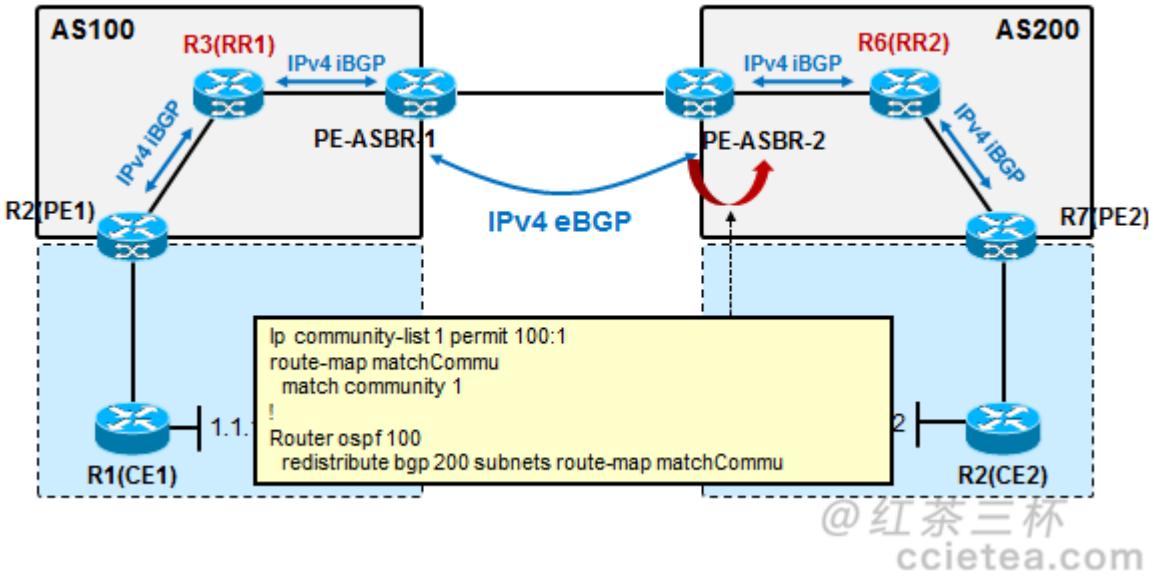
那么完成可以在 PE2 及 RR2 上、PE1 及 RR1 上，在通告自己 Loopback 路由进 BGP 的时候，打上 community，然后在 ASBR 重发布过程中，利用 community 来做路由过滤。

先定义一个 route-map

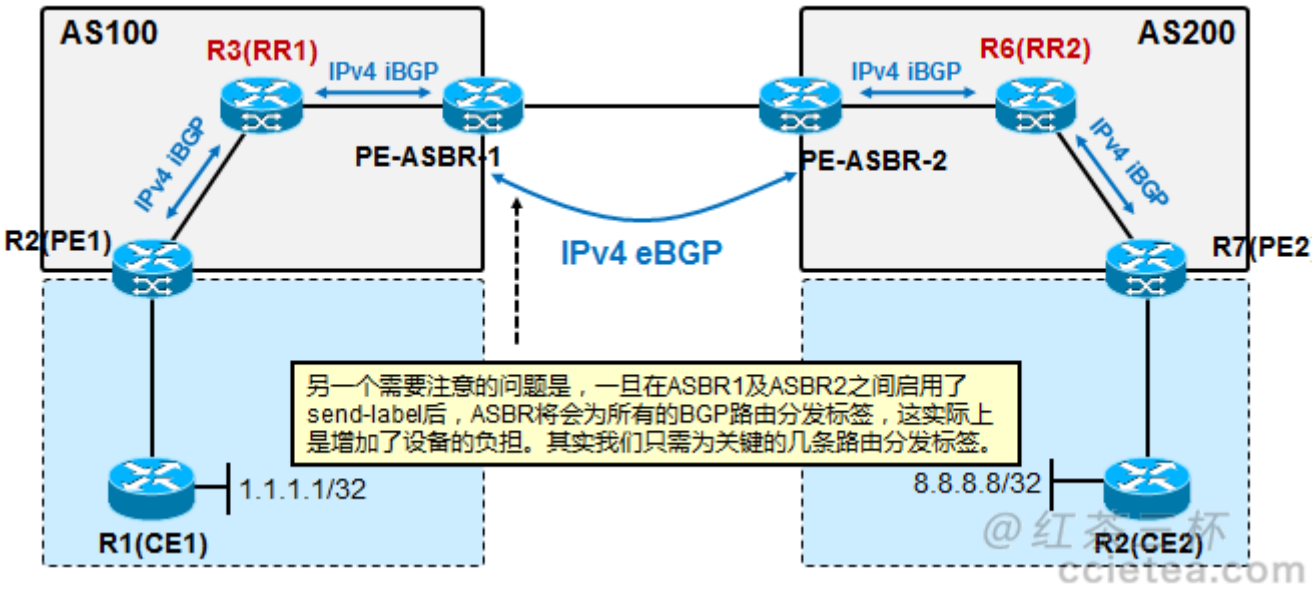
```
route-map setCommu
  set community 100:1
```



那么这样一来，我们在 BGP 到 OSPF 的重发布过程中，就可以利用这个 Community 值来执行路由的过滤。同样的，另一侧的网络配置类似。



2. 控制 BGP 路由更新及标签分发



在前面的配置中，我们在 ASBR1 及 2 之间使用 send-label 配置，使得 BGP 为 BGP 路由分发标签。拿 ASBR1 举例说，实际上 ASBR1 在将本 AS 内的 BGP 路由通告给 ASBR2 的时候，所有的被通告的 BGP 路由都捆绑了标签，那么对于 ASBR2 来说，实际上就加重了设备的负担，因为其实我们只是需要几个关键性的路由的标签。

那么我们可以在前面配置的基础上，利用已经设定好的 Community 值，来做一些策略。

假设我们在 PE1 及 RR1 通告自己 Loopback 路由进 BGP 的时候打上了 Community 值 100:1 ,那么我们可以在 ASBR1 上进一步配置：

```
Ip community 2 permit 100:1
route-map setLabel permit 10
  match community 2
  set mpls-label                    !!只对携带了 community 值 100:的路由分发标签
route-map setLabel permit 20        !!其他路由直接放行，不给标签
!
router bgp 100
  address-family ipv4
    neighbor 10.1.45.5 route-map setLabel out
```

ASBR2 上的部署方式类似。

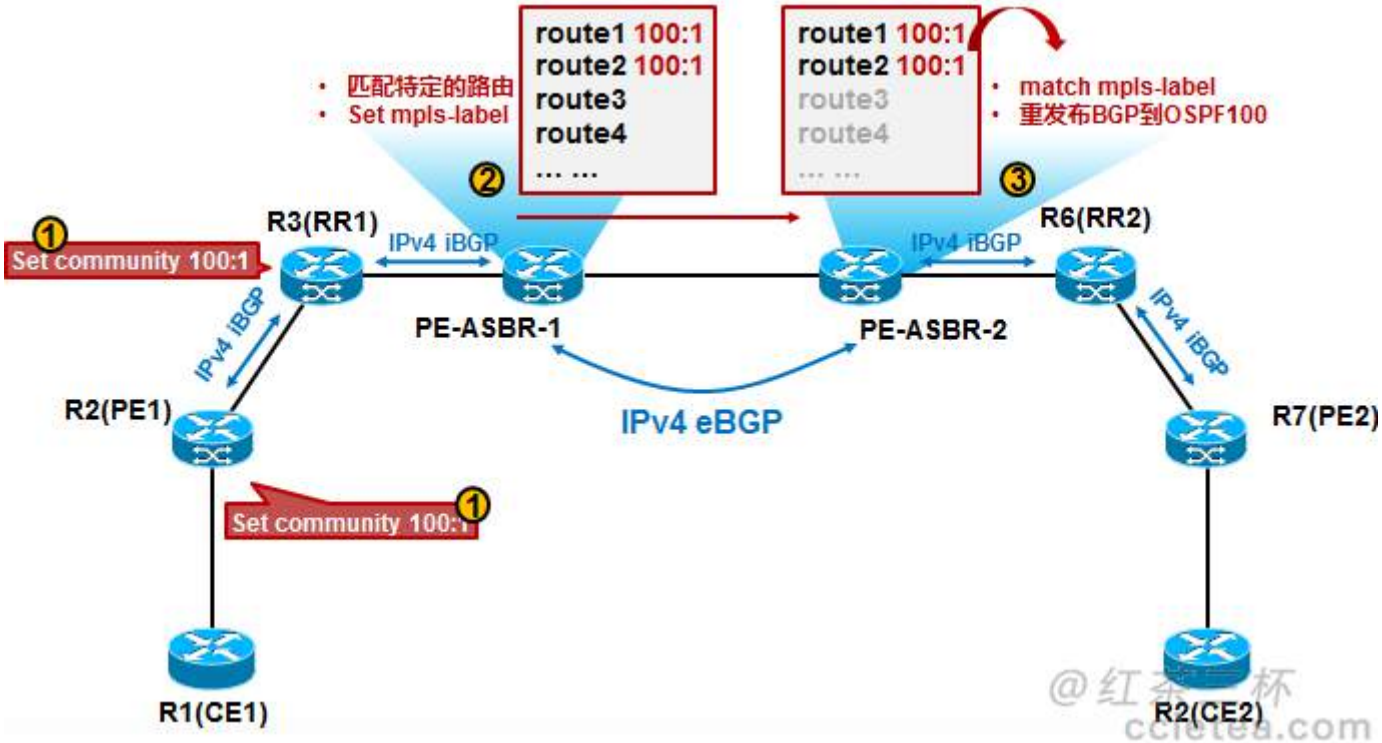
这样一来，ASBR1 只会对匹配了 Community 值 100:1 的那些 BGP 路由分发标签，然后传递给 ASBR2，其他 BGP 路由直接放行，不携带标签。

借助这个方法，我们可以改良一下前面提到的“1.控制重发布到 Backbone IGP 内的 BGP 路由”的方法。在 ASBR2 上，直接用一条 route-map，match mpls-label 命令可以帮助我们抓取那些 ASBR1 传递过来的、带

了标签的 BGP 路由，我们只需要将这些路由注入到 Backbone 的 OSPF 100 中。

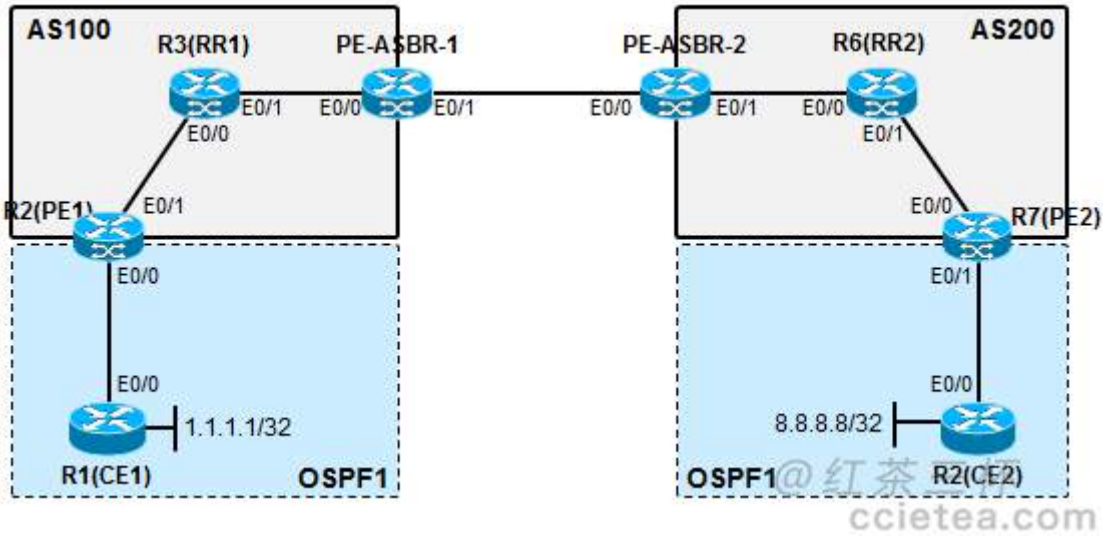
```
route-map matchLabel
   match mpls-label
!
router ospf 100
   redistribute bgp 200 subnets route-map matchLabel
```

另一边的配置类似。所以其实也就是个 match mpls-label 和 set mpls-label 的玩意儿。



9.5.3 实验示例

9.5.3.1 实验描述



1. **环境描述**
   - 设备的互联地址空间采用 10.1.xy.0/24，其中 xy 为设备编号，x 为小值，y 为大值，IP 地址最后一个八位组为本设备编号。
   - 所有设备的 Loopback0 地址空间为 x.x.x.x/32，x 为设备编号

9.5.3.2 实验过程

1. **完成设备的基本配置（hostname、IP 等）**

   这个就不说了吧？

2. **完成 AS100 及 AS200 内 MPLS VPN Backbone 的 IGP 配置**

   这里采用 OSPF，进程号都用 100。PE1、RR1、ASBR1 及 PE2、RR2、ASBR2 都宣告自己的 Loopback0 口进 OSPF 进程 100，以便 Backbone 内 IGP 互通。

   这部分配置也相对比较简单不再粘贴脚本了。

3. **完成 AS100 及 AS200 Backbone 的 LDP 配置**

   LDP routerID 及 label range、接口激活 LDP。

   例如 PE1 的配置如下：

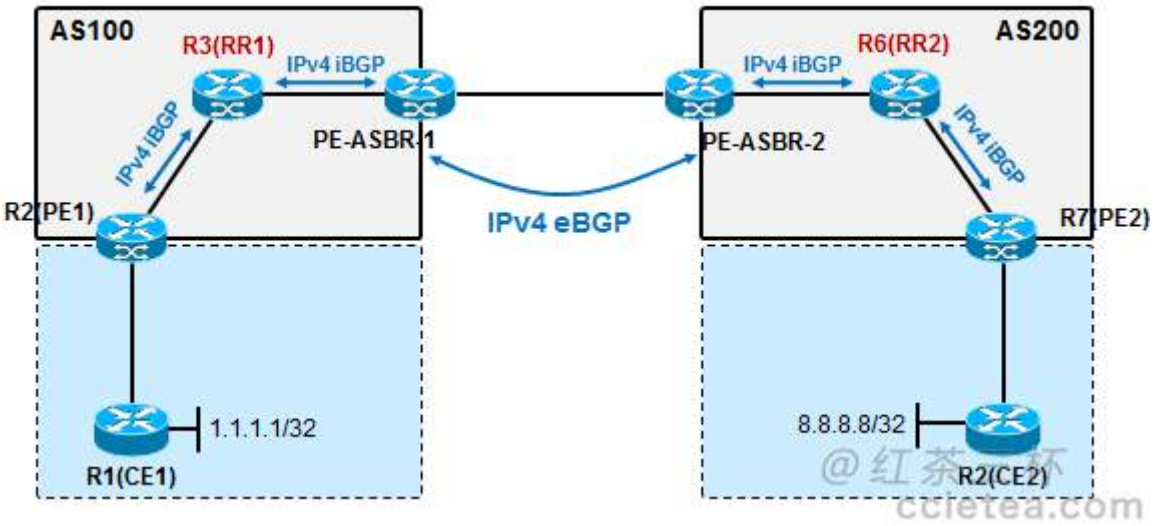
```
mpls ldp router-id loopback 0
```

```
mpls label range 200 299          !!因为 PE1 是 R2，按接口编号来设定标签范围
interface eth0/1
  mpls ip
```

其他的也不再赘述了。

4. 完成 IPv4 BGP 邻居关系的建立

建立 IPv4 BGP 邻居关系的目的是为了打通 AS100 及 AS200 的 Backbone 内的路由，使得两个 RR 之间能够建立起 MP-eBGP 邻居关系，另外也为后续的数据转发层面的优化做好准备。BGP 邻居关系请见下图：



PE1 的配置如下：

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family ipv4
  no synchronization
  network 2.2.2.2 mask 255.255.255.255
  neighbor 3.3.3.3 activate
  no auto-summary
 exit-address-family
```

RR1 的配置如下：

```
router bgp 100
 no bgp default ipv4-unicast
```

```
 neighbor 2.2.2.2 remote-as 100
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 4.4.4.4 remote-as 100
 neighbor 4.4.4.4 update-source Loopback0
 !
 address-family ipv4
  no synchronization
  network 3.3.3.3 mask 255.255.255.255
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 route-reflector-client
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 route-reflector-client
  no auto-summary
 exit-address-family
```

ASBR1 的配置如下：

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 10.1.45.5 remote-as 200
 !
 address-family ipv4
  no synchronization
  network 4.4.4.4 mask 255.255.255.255
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 next-hop-self
  neighbor 10.1.45.5 activate
  no auto-summary
 exit-address-family
```

AS200 内设备的配置类似，不再赘述。

完成这个阶段的配置后，通过 IGP 实现了 Backbone 内的路由可达性；通过 BGP 实现了 AS 之间的路由可达性。但是我们留意到在 PE1、RR1、ASBR1 去往对端 AS 内的那些路由都是 BGP 的，而 LDP 是不会为

BGP 路由分配标签的，那么这样就会出问题。举例说，RR1 上，学习到的 6.6.6.6 路由（也就是 RR2 的 Loopback 路由）是 BGP 路由，而当 PE1 要发送数据去往 CE2，那么如果 VPNv4 路由的下一跳是 6.6.6.6，PE1 就需要关于 6.6.6.6 的标签，但是 RR1 是不可能为 BGP 路由 6.6.6.6 分配标签并且通告给 PE1 的，这样 LDP 的标签分发就出问题了。

2. 在 ASBR1 及 ASBR2 上做 BGP 到 OSPF 100 的路由重发布

这个重发布的目的，是使得两个 AS 的 Backbone 内的路由器学习到的、关于对端 AS Backbone 内的路由为 IGP 路由（OSPF 路由），从而才能促使 LDP 分配标签，否则 VPN 数据传输的时候外层标签的分配就会有问题。

ASBR1 的 OSPF 进程 100 的配置如下：

```
router ospf 100
 redistribute bgp 100 subnets
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
```

ASBR2 的配置类似。

完成这个配置后，在 PE1 及 RR1 上，看到的关于 5.5.5.5、6.6.6.6、7.7.7.7 的这些 Loopback 路由就是 OSPF 外部路由了。但是还有一个问题，对于 ASBR1 来说由于他是路由重发布的部署点，这些 AS200 内的 Loopback 路由仍然是 BGP 的，同样的对于 ASBR2 来说，这些 AS100 内的 Loopback 路由也特么仍然是 BGP 的，那么 LDP 默认情况下是不会对路由表中的 BGP 路由去分发标签的。如此就造成了 RR1 上，关于 5.5.5.5、6.6.6.6、7.7.7.7 这些路由在 LFIB 中都是 untagged 的问题，RR2 当然也有类似的问题。

3. 在 ASBR1 及 ASBR2 之间完成 send-label 的配置

为了解决上面所述的问题，我们在 ASBR1 上增加如下配置：

```
router bgp 100
  address-family ipv4
    neighbor 10.1.45.4 send-label
```

ASBR2 的配置增加如下：

```
router bgp 100
  address-family ipv4
    neighbor 10.1.45.5 send-label
```

如此一来，ASBR1 及 ASBR2 的 IPv4 BGP 得到扩展，能够为 BGP 路由分配并传递标签映射，以解决外层标签的问题。例如 ASBR2 会为本地的 BGP 路由 5.5.5.5、6.6.6.6、7.7.7.7 分配标签，并且将这些标签联同

前缀通过 BGP 传递给 ASBR1。那么在 ASBR1 上：

R4-ASBR1#sh mpls forwarding-table

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
400	300	2.2.2.2/32	0	Et0/0	10.1.34.3
401	Pop Label	3.3.3.3/32	0	Et0/0	10.1.34.3
402	Pop Label	10.1.23.0/24	0	Et0/0	10.1.34.3
403	**Pop Label**	**5.5.5.5/32**	**0**	**Et0/1**	**10.1.45.5**
404	**500**	**6.6.6.6/32**	**0**	**Et0/1**	**10.1.45.5**
405	**501**	**7.7.7.7/32**	**0**	**Et0/1**	**10.1.45.5**
406	Pop Label	10.1.45.5/32	0	Et0/1	10.1.45.5

ASBR1 收到了 ASBR2 捆绑的标签，同时，它自己也为这些 BGP 路由分发了本地标签，同时通过 LDP 传递给了 RR1。这样一来相关的外层标签问题就得到了解决。

4. 完成 VRF 的配置，以及 PE-CE 路由协议的配置

CE1 及 CE2 的配置这里就不再赘述了，都是常规的 OSPF 配置。

PE1 的配置如下：

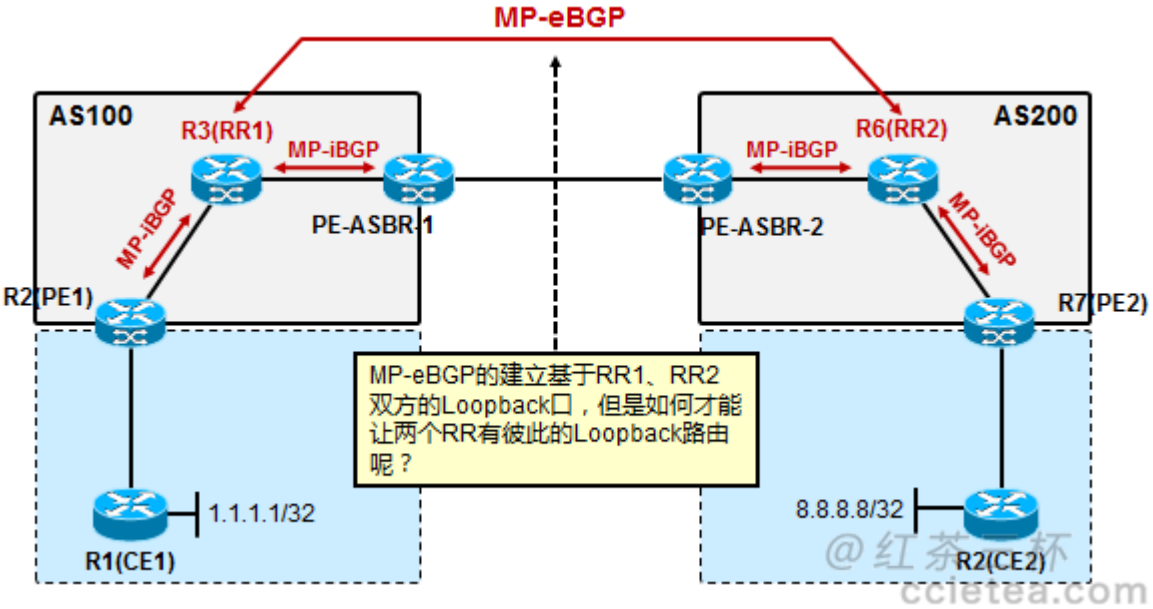
```
ip vrf cisco
 rd 100:1
 route-target export 100:2
 route-target import 200:7                    !! 这个值对应 PE2 上的 RT export 值
!
interface Ethernet0/0
 ip vrf forwarding cisco
 ip address 10.1.12.2 255.255.255.0
!
router ospf 1 vrf cisco
 network 10.1.12.2 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip vrf cisco
 rd 200:8
 route-target export 200:7
```

```
 route-target import 100:2
!
interface Ethernet0/1
 ip vrf forwarding cisco
 ip address 10.1.78,7 255.255.255.0
!
router ospf 1 vrf cisco
 network 10.1.78.7 0.0.0.0 area 0
```

5. 完成 MP-BGP 邻居关系的建立



PE1 的配置如下：

```
router bgp 100
   neighbor 3.3.3.3 remote 100
   neighbor 3.3.3.3 update-source loopback 0
   address-family vpnv4
      neighbor 3.3.3.3 activate
```

RR1 的配置如下：

```
router bgp 100
   … …省略部分配置
   neighbor 6.6.6.6 remote 200
   neighbor 6.6.6.6 update-source loopback 0
```

```
 neighbor 6.6.6.6 ebgp-multihop
 address-family vpnv4
   neighbor 2.2.2.2 activate
   neighbor 2.2.2.2 route-reflector-client
   neighbor 4.4.4.4 activate
   neighbor 4.4.4.4 route-reflector-client
   neighbor 6.6.6.6 activate                    !! RR2
```

ASBR1 的配置如下：

```
router bgp 100
 … …省略部分配置
 address-family vpnv4
   neighbor 4.4.4.4 activate
```

AS200 内相关设备的配置类似，这里不再赘述。

6. 完成 PE1 及 PE2 上 MP-BGP 与 PE-CE 的双向重发布

PE1 的配置如下：

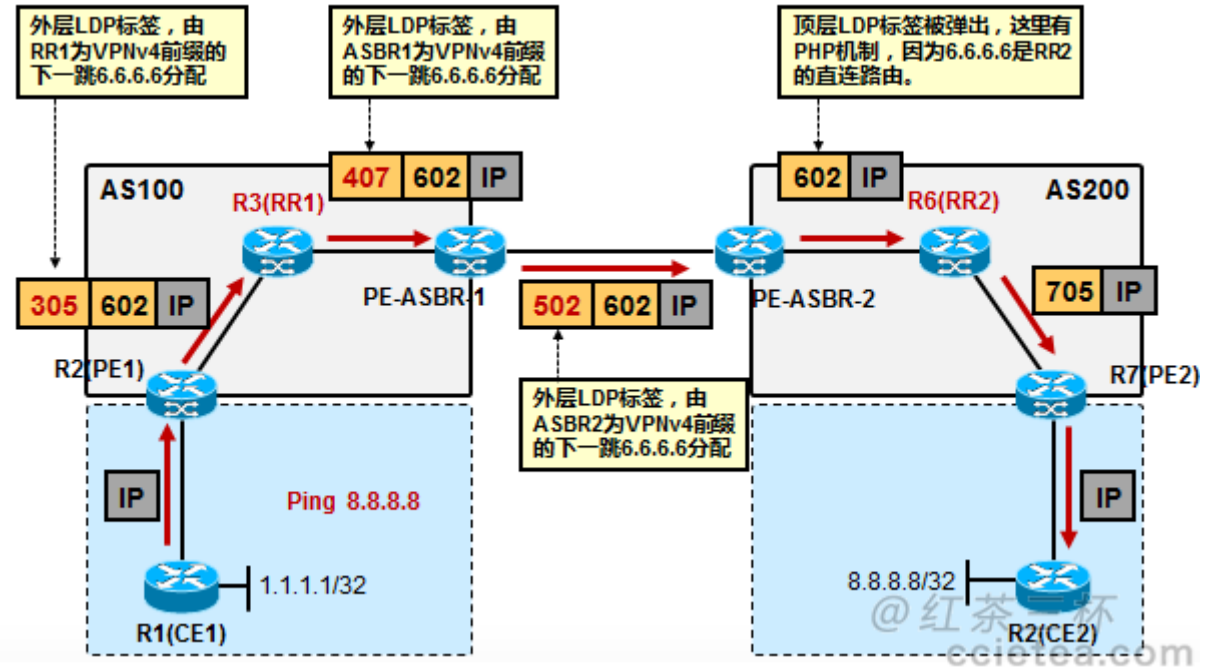
```
router ospf 1 vrf cisco
 redistribute bgp 100 subnets
router bgp 100
 address-family ipv4 vrf cisco
   redistribute ospf 1 vrf cisco match internal external
```

PE2 的配置如下：

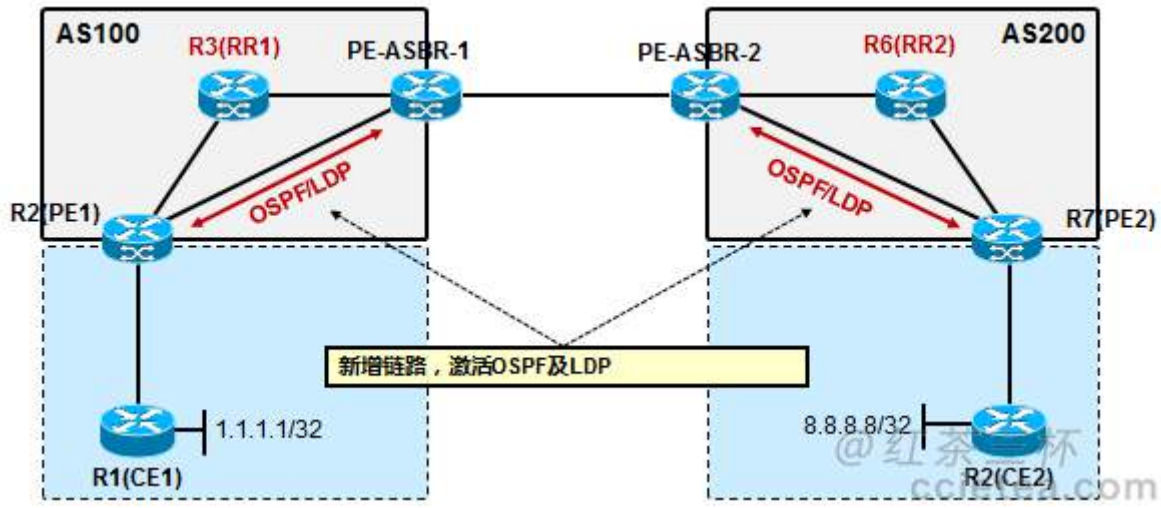
```
router ospf 1 vrf cisco
 redistribute bgp 200 subnets
router bgp 200
 address-family ipv4 vrf cisco
   redistribute ospf 1 vrf cisco match internal external
```

到此，我们的大部分配置都已经完成了，可以测试一下从 CE1 去 traceroute 8.8.8.8。

下面就是一个典型的数据层面的传输过程（标签值实际在做的时候可能和下图不大一样，各有各的情况吧）。



7. 数据转发层面的优化



前面已经说了如果在 PE1-ASBR1；PE2-ASBR2 之间增加如图的链路并激活 OSPF 及 LDP 连接，那么数据转发层面存在不优化的现象，也就是两台 RR 会承担一部分的数据转发任务。

优化的方式前面也说了，我们直接看配置：

RR1 的增补配置如下：

```
router bgp 100
  address-family vpnv4
neighbor 6.6.6.6 next-self-unchanged
```

RR2 的增补配置如下：

```
router bgp 100
  address-family vpnv4
neighbor 3.3.3.3 next-self-unchanged
```

那么现在我们再去 CE1 上 traceroute 8.8.8.8 就会发现变化了。

8. 控制 BGP 路由标签的分发以及控制 BGP 到 OSPF 的路由重发布

现在我们面临两个需要优化的地方：

- 在 ASBR1 及 ASBR2 上我们使用 send-label 关键字扩展了 BGP，使得两台 ASBR 能够为 BGP 路由分发标签，并且传递给对端。而这种分发标签的行为，会针对其本地的所有的 BGP 路由进行标签的分发，实际上，我们仅仅需要几个关键的 Loopback 路由的标签来完成数据转发任务，例如 PE1、PE2、RR1、RR2 的 Loopback 口路由 ,而对于其他的路由 ,实际上是不需要标签的。那么能否只为这些关键性的 BGP 路由来分发标签呢？
- 再者，我们在 ASBR2 及 2 上都做了 BGP 到 OSPF 的重发布，这实际上是存在巨大的风险，因为这无疑会加重 OSPF 的负担，毕竟 BGP 是用于携带大量路由前缀的，那么我们能否控制重发布，只将那些关键性的路由（实际上就是前面一点所说的，需要分发标签的 BGP 路由）注入到 OSPF 中呢？

实现的思路是这样的，我们拿 AS100 内的路由来说。

- **首先关键性的路由是 PE1 及 RR1 的 Loopback，因为他们都是有可能成为 VPNV4 前缀 NEXT_HOP 地址的路由。我们在 PE1 及 RR1 上，配置一个 route-map 用于设置 community 值 100:1，然后在宣告自己 Loopback 路由的时候进行关联。**
- **在 ASBR1 上，现在要向 ASBR2 来传递本 AS 内的那些 BGP 路由了，我们也用一个 route-map，匹配前面设置了 100:1 community 值的路由，然后使用 set mpls-label 关键字，让 BGP 只为这些感兴趣的路由分发并传递标签，然后其他的 BGP 路由放行即可。**
- **而在 ASBR2 上，收到了 ASBR1 传递过来的路由，其中关键性的那几条路由是携带了标签的，那么我们只需要将这些带了标签的路由 ,重发布进 OSPF 即可对吧？所以也用一个 route-map ,match mpls-label 这条命令可用于匹配携带了标签的 IPv4 BGP 路由前缀，然后在 BGP 到 OSPF 的重发布的时候调用这个 route-map 即可。**

顺便说一句，从 AS200 到 AS100 的路由发布原理是一样的，思路换一下就行。

在 PE1 上的增补配置如下：

```
route-map setCommu permit 10
 set community 100:1
!
router bgp 100
   address-family ipv4
      network 2.2.2.2 mask 255.255.255.255 route-map setCommu
      neighbor 3.3.3.3 send-community                    !!注意，一定要配
```

在 RR1 上增补的配置如下：

```
route-map setCommu permit 10
 set community 100:1
!
router bgp 100
   address-family ipv4
      network 3.3.3.3 mask 255.255.255.255 route-map setCommu
      neighbor 4.4.4.4 send-community
```

在 ASBR1 上增补的配置如下：

```
ip community-list 1 permit 100:1        !!匹配带了 community 100:1 的路由
!
route-map setLabel permit 10
 match community 1
 set mpls-label                         !!让 BGP 为这些路由分发标签
route-map setLabel permit 20            !!放行其他路由
!
router bgp 100
   address-family ipv4
      neighbor 10.1.45.5 send-community
      neighbor 10.1.45.5 route-map setLabel out        !!应用 route-map
```

在 ASBR2 上增补的配置如下：

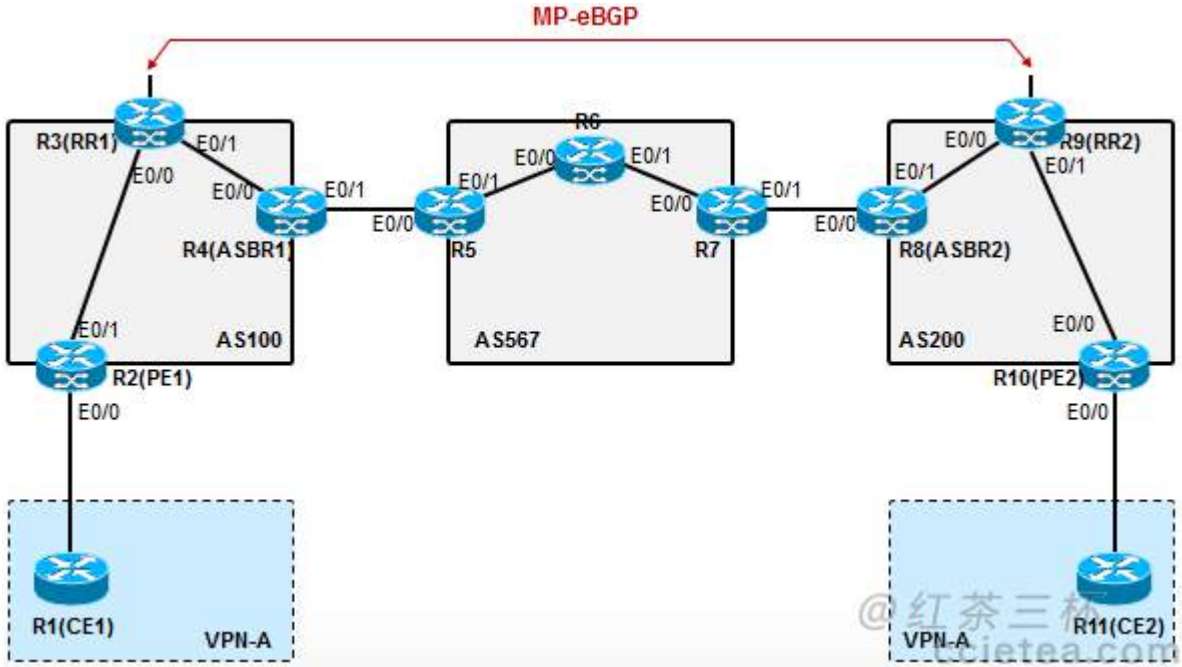
```
ip community-list 1 permit 100:1
route-map matchCommu permit 10
 match community 1
!
```

```
router ospf 100
  redistribute bgp 200 subnets route-map matchLabel
```

另一边的配置类似，不再赘述了。

9.6 Option5：Non-VPN Transit Provider

9.6.1 实验描述



1. **设备互联 IP**

   设备互联地址空间为 10.1.xy.0/24，其中 x 为设备编号较小的值，y 为大值。最后一个八位组为设备编号。

   例如 R1-R2 之间的互联，R1 的接口地址为 10.1.12.1/24，R2 的接口地址为 10.1.12.2/24，依此类推。

   R9-R10 之间的连线，R9 的地址为 10.1.90.9/24，R10 的地址为 10.1.90.10/24；

   R10-R11 之间的连线，R10 的地址为 10.1.101.10/24，R11 的地址为 10.1.101.11/24

2. **设备 Loopback 口 IP**

   所有的设备开启 Loopback0，地址为 x.x.x.x/32，x 为设备编号

3. **IGP 协议的规划**

   AS100 内的 IGP 为 OSPF，进程号 100；AS200 内的 IGP 为 OSPF，进程号 100；

   AS567 内的 IGP 为 EIGRP，进程号为 1

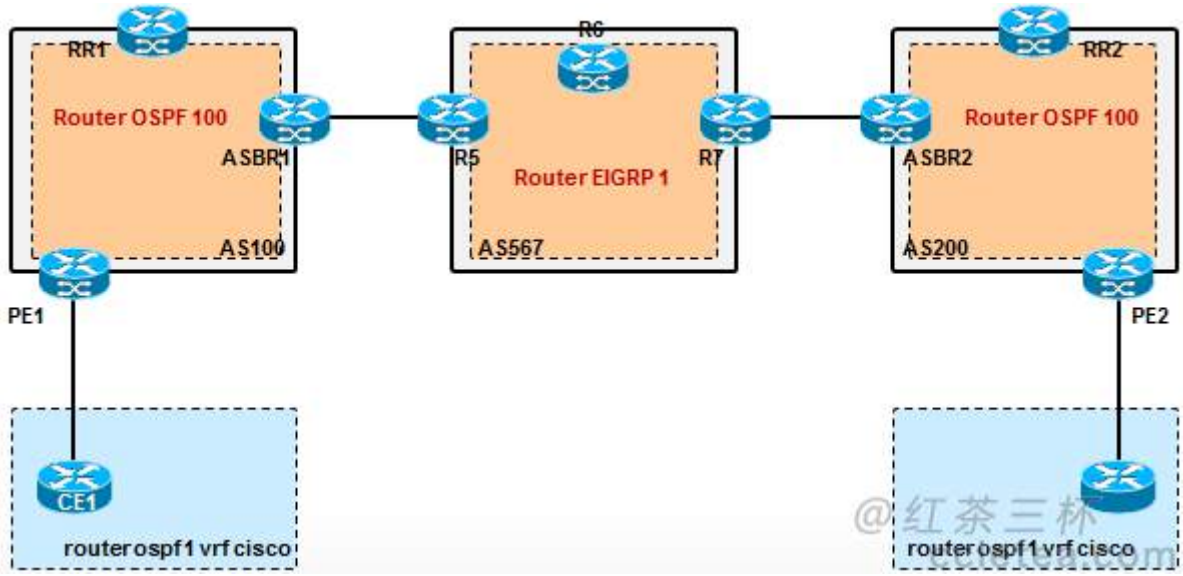
CE1-PE1 及 CE2-PE2 之间运行的 IGP 为 VRF OSPF，进程号 1

9.6.2 实验详解

1. **完成基本配置（hostname、接口 IP）**

   这就不说了吧，说得我心都碎了

2. **完成 AS100、AS200 及 AS567 内 IGP 的配置**



CE1、CE2 的配置就不说了，常规的 OSPF 配置；

PE1 的配置如下：

```
router ospf 100
  router-id 2.2.2.2
  network 10.1.23.2 0.0.0.0 area 0
  network 2.2.2.2 0.0.0.0 area 0
```

RR1 的配置如下：

```
router ospf 100
  router-id 3.3.3.3
  network 10.1.23.3 0.0.0.0 area 0
  network 10.1.34.3 0.0.0.0 area 0
  network 3.3.3.3 0.0.0.0 a 0
```

ASBR1 的配置如下：

```
router ospf 100
  router-id 4.4.4.4
  network 10.1.34.4 0.0.0.0 area 0
  network 4.4.4.4 0.0.0.0 area 0
```

R5 的配置如下：

```
router eigrp 1
  no auto-summary
  network 10.1.56.0 0.0.0.255
  network 5.0.0.0
```

R6 的配置如下：

```
router eigrp 1
  no auto-summary
  network 10.1.56.0 0.0.0.255
  network 10.1.67.0 0.0.0.255
  network 6.0.0.0
```

R7 的配置如下：

```
router eigrp 1
  no auto-summary
  network 10.1.67.0 0.0.0.255
  network 7.0.0.0
```

ASBR2、RR2、PE2 的配置大同小异，不再赘述。

3. **完成 LDP 的配置**

PE1 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 200 299
interface e0/1
```

```
  mpls ip
```

RR1 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 300 399
interface e0/0
  mpls ip
interface e0/1
  mpls ip
```

ASBR1 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 400 499
interface e0/0
  mpls ip
```

R5 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 500 599
interface e0/1                    !! 只在 E0/1 口上激活 LDP，ASBR1 与 R5 之间无需 LDP 邻接
  mpls ip
```

R6 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 600 699
interface e0/0
  mpls ip
interface e0/1
  mpls ip
```

R7 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 700 799
```

```
interface e0/0
  mpls ip
```

ASBR2 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 800 899
interface e0/1
  mpls ip
```

RR2 的配置如下：

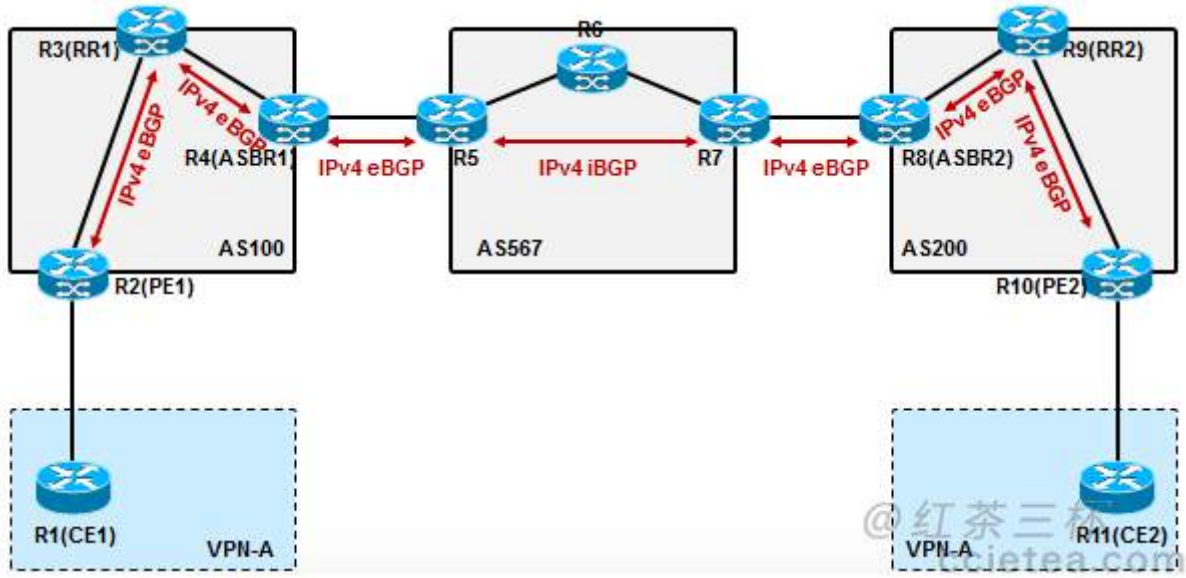
```
mpls ldp router-id loopback 0
mpls label range 900 999
interface e0/0
  mpls ip
interface e0/1
  mpls ip
```

PE2 的配置如下：

```
mpls ldp router-id loopback 0
mpls label range 1000 1099
interface e0/0
  mpls ip
```

4. 完成 IPv4 BGP 邻居关系的配置

IPv4 BGP 邻居关系规划如下图：



建立这些 IPv4 BGP 邻接关系的目的 ,是为了同步各个 AS 内部的关键性路由 ,主要就是关键设备的 Loopback 路由，为后续的数据转发，以及标签分配做准备。

PE1 上的配置如下：

```
router bgp 100
   no bgp default ipv4-unicast
   bgp router-id 2.2.2.2
   neighbor 3.3.3.3 remote-as 100
   neighbor 3.3.3.3 update-source Loopback0
   address-family ipv4
      no synchronization
      network 2.2.2.2 mask 255.255.255.255
      neighbor 3.3.3.3 activate                    !!采用规范化配置，在地址族里完成邻居的激活
      no auto-summary
    exit-address-family
```

RR1 上的配置如下：

```
router bgp 100
   bgp router-id 3.3.3.3
   no bgp default ipv4-unicast
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 update-source Loopback0
   neighbor 4.4.4.4 remote-as 100
```

```
 neighbor 4.4.4.4 update-source Loopback0
 address-family ipv4
    no synchronization
    network 3.3.3.3 mask 255.255.255.255
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 route-reflector-client
    neighbor 4.4.4.4 activate
    neighbor 4.4.4.4 route-reflector-client
    no auto-summary
  exit-address-family
```

ASBR1 上的配置如下：

```
router bgp 100
 bgp router-id 4.4.4.4
 no bgp default ipv4-unicast
 neighbor 3.3.3.3 remote-as 100
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 10.1.45.5 remote-as 567
!
 address-family ipv4
    no synchronization
    network 4.4.4.4 mask 255.255.255.255      !!在本环境中，可以不做宣告，不影响实验
    neighbor 3.3.3.3 activate
    neighbor 3.3.3.3 next-hop-self            !!注意配置，否则 iBGP 邻居收到的路由就不 best 了。
    neighbor 10.1.45.5 activate
    no auto-summary
 exit-address-family
```

R5 的配置如下：

```
router bgp 567
 bgp router-id 5.5.5.5
 no bgp default ipv4-unicast
 neighbor 7.7.7.7 remote-as 567
```

```
  neighbor 7.7.7.7 update-source Loopback0
  neighbor 10.1.45.4 remote-as 100
 !
  address-family ipv4
     no synchronization
     neighbor 7.7.7.7 activate
     neighbor 7.7.7.7 next-hop-self
     neighbor 10.1.45.4 activate
     no auto-summary
 exit-address-family
```

R7 的配置如下：

```
router bgp 567
  bgp router-id 7.7.7.7
  no bgp default ipv4-unicast
  neighbor 5.5.5.5 remote-as 567
  neighbor 5.5.5.5 update-source Loopback0
  neighbor 10.1.78.8 remote-as 200
 !
  address-family ipv4
     no synchronization
     neighbor 5.5.5.5 activate
     neighbor 5.5.5.5 next-hop-self
     neighbor 10.1.78.8 activate
     no auto-summary
 exit-address-family
```

ASBR2 的配置如下：

```
router bgp 200
  bgp router-id 8.8.8.8
  no bgp default ipv4-unicast
  neighbor 9.9.9.9 remote-as 200
```

```
 neighbor 9.9.9.9 update-source Loopback0
 neighbor 10.1.78.7 remote-as 567
!
 address-family ipv4
    no synchronization
    neighbor 9.9.9.9 activate
    neighbor 9.9.9.9 next-hop-self
    neighbor 10.1.78.7 activate
    no auto-summary
 exit-address-family
```

RR2 上的配置如下：

```
router bgp 200
 bgp router-id 9.9.9.9
 no bgp default ipv4-unicast
 neighbor 8.8.8.8 remote-as 200
 neighbor 8.8.8.8 update-source Loopback0
 neighbor 10.10.10.10 remote-as 200
 neighbor 10.10.10.10 update-source Loopback0
 address-family ipv4
    no synchronization
    network 9.9.9.9 mask 255.255.255.255
    neighbor 8.8.8.8 activate
    neighbor 8.8.8.8 route-reflector-client
    neighbor 10.10.10.10 activate
    neighbor 10.10.10.10 route-reflector-client
    no auto-summary
  exit-address-family
```

PE2 上的配置如下：

```
router bgp 200
 bgp router-id 10.10.10.10
 no bgp default ipv4-unicast
```

```
 neighbor 9.9.9.9 remote-as 200
 neighbor 9.9.9.9 update-source Loopback0
 address-family ipv4
    no synchronization
    network 10.10.10.10 mask 255.255.255.255
    neighbor 9.9.9.9 activate
    no auto-summary
  exit-address-family
```

5. 完成 VRF 的配置

PE1 的配置如下：

```
ip vrf cisco
 rd 100:1
 route-target export 100:2
 route-target import 200:10                    !! 对应 PE2 的 RT export 200:10
interface e0/0
!
interface Ethernet0/0
 ip vrf forwarding cisco
 ip address 10.1.12.2 255.255.255.0
```

PE2 的配置如下：

```
ip vrf cisco
 rd 200:1
 route-target export 200:10
 route-target import 100:2                     !! 对应 PE1 的 RT export 100:2
interface e0/0
!
interface Ethernet0/1
 ip vrf forwarding cisco
 ip address 10.1.101.10 255.255.255.0
```

6. **完成 PE-CE 路由协议的配置，并在 PE 上完成 ospf vrf 进程到 BGP 的双向重发布**

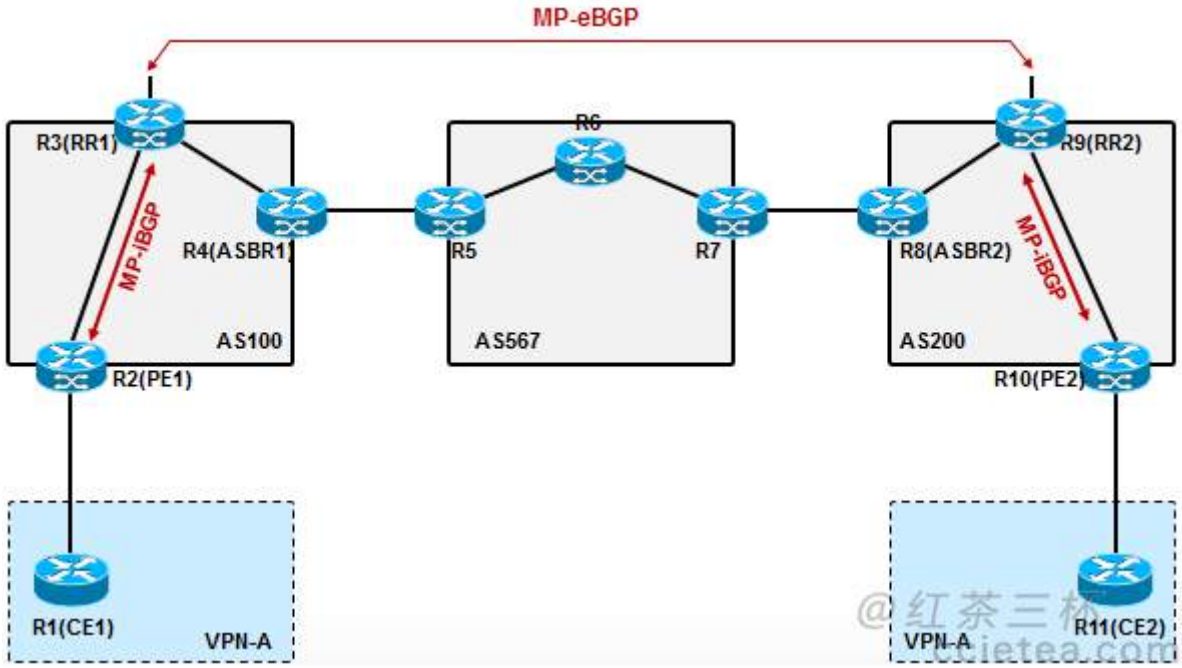
PE1 的配置如下：

```
router ospf 1 vrf cisco
   redistribute bgp 100 subnets
   network 10.1.12.2 0.0.0.0 area 0
!
router bgp 100
   address-family ipv4 vrf cisco
      no synchronization
      redistribute ospf 1 vrf cisco match internal external 1 external 2
```

PE2 的配置如下：

```
router ospf 1 vrf cisco
   redistribute bgp 200 subnets
   network 10.1.101.10 0.0.0.0 area 0
!
router bgp 200
   address-family ipv4 vrf cisco
      no synchronization
      redistribute ospf 1 vrf cisco match internal external 1 external 2
```

7. **完成 MP-BGP 邻居关系的建立**



注意 RR1 是 PE1 的 IPv4 路由反射器，同时也是 PE1 的 VPNv4 路由反射器， RR2 也类似的道理。因此在配置上这里要留意了。

PE1 的配置增加如下：

```
router bgp 100
  neighbor 3.3.3.3 remote-as 100
  neighbor 3.3.3.3 update-source Loopback0          !!上面两条其实前面已经配置了
  address-family vpnv4
    neighbor 3.3.3.3 activate
    neighbor 3.3.3.3 send-community extended
```

RR1 的配置如下：

```
router bgp 100
  neighbor 2.2.2.2 remote-as 100
  neighbor 2.2.2.2 update-source Loopback0
  neighbor 9.9.9.9 remote-as 200                    !这是 RR2
  neighbor 9.9.9.9 ebgp-multihop 255
  neighbor 9.9.9.9 update-source Loopback0
  address-family vpnv4
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 send-community extended
```

```
    neighbor 2.2.2.2 route-reflector-client
    neighbor 9.9.9.9 activate
    neighbor 9.9.9.9 send-community extended
    neighbor 9.9.9.9 next-hop-unchanged
```

PE2 的配置如下：

```
router bgp 200
  neighbor 9.9.9.9 remote-as 200
  neighbor 9.9.9.9 update-source Loopback0
  address-family vpnv4
    neighbor 9.9.9.9 activate
    neighbor 9.9.9.9 send-community extended
  exit-address-family
```

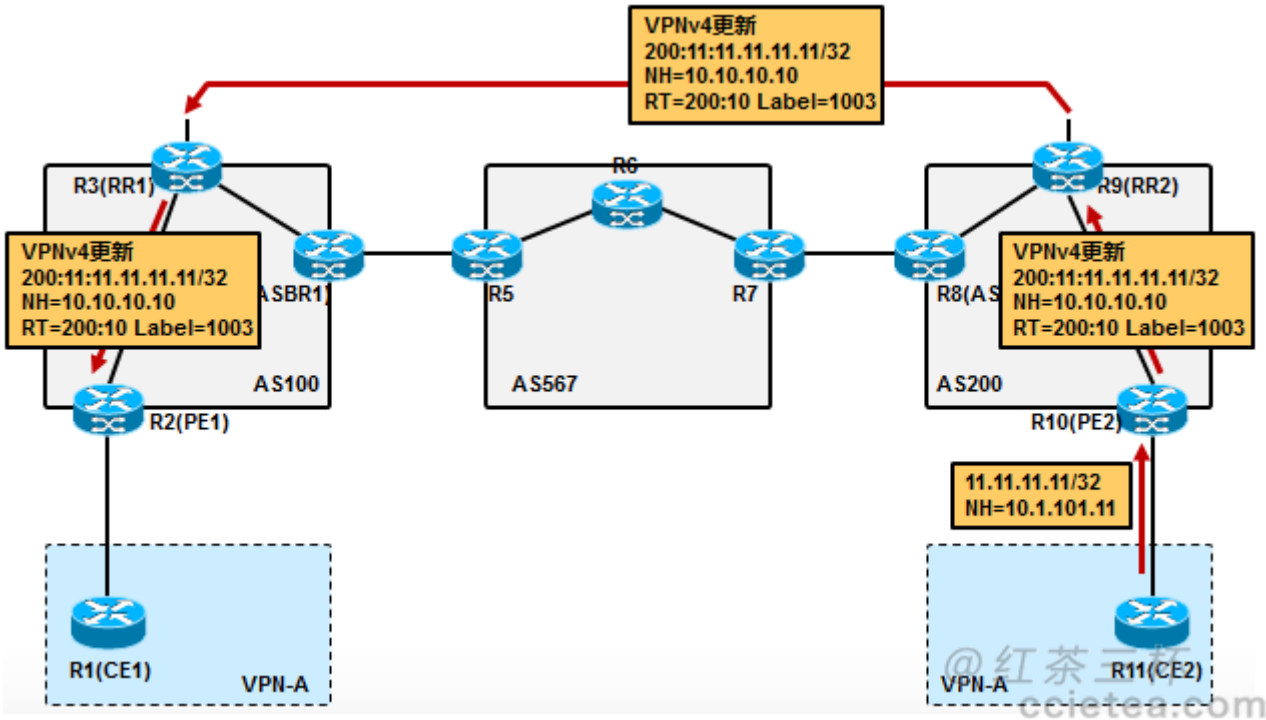
RR2 的配置如下：

```
router bgp 200
  neighbor 3.3.3.3 remote-as 100
  neighbor 3.3.3.3 ebgp-multihop 255
  neighbor 3.3.3.3 update-source Loopback0
  neighbor 10.10.10.10 remote-as 200
  neighbor 10.10.10.10 update-source Loopback0
 !
address-family vpnv4
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
  neighbor 3.3.3.3 next-hop-unchanged
  neighbor 10.10.10.10 activate
  neighbor 10.10.10.10 send-community extended
  neighbor 10.10.10.10 route-reflector-client
```

RR 之间互相配置 next-hop-unchanged，以 RR1 为例，这条命令将使得 RR1 学习到 PE1 始发的 VPNv4 路由后，更新给 RR2 时 NEXT_HOP 保持为 PE1 不变。这样有利于在冗余环境中的数据转发层面的优化。具体请参考本笔记的 Option4 部分章节。

8. 解决标签问题

完成这一步后，VPNv4 路由的传递就没有问题了，CE1 和 CE2 之间都应该能够学习到对方的路由。拿路由 11.11.11.11/32 的传递来分析：



现在路由是传递过来了，路由控制层面的问题都解决了，可是数据层面呢？

- 方法 1：可以考虑 BGP 到 OSPF 重发布，然后 ebgp 邻居之间 send-label
- 方法 2：在 IPv4 BGP 邻居之间用 send-label 来扩展 IPv4 BGP，为 BGP 路由分配标签，同时利用 BGP 来传递 BGP 的 IPv4 前缀以及所分配的标签。我们这里测试一下这种方法：

PE1 的配置如下：

```
router b 100
address-family ipv4
  neighbor 3.3.3.3 send-label
```

RR1 的配置如下：

```
router b 100
address-family ipv4
  neighbor 2.2.2.2 send-label
  neighbor 4.4.4.4 send-label
```

ASBR1 的配置如下：

```
router b 100
address-family ipv4
  neighbor 3.3.3.3 send-label
  neighbor 10.1.45.5 send-label
```

R5 的配置如下：

```
router b 567
address-family ipv4
  neighbor 10.1.45.4 send-label
  neighbor 7.7.7.7 send-label
```

R7 的配置如下：

```
router b 567
address-family ipv4
  neighbor 10.1.78.8 send-label
  neighbor 5.5.5.5 send-label
```

ASBR2 的配置如下：

```
router b 200
address-family ipv4
  neighbor 10.1.78.7 send-label
  neighbor 9.9.9.9 send-label
```

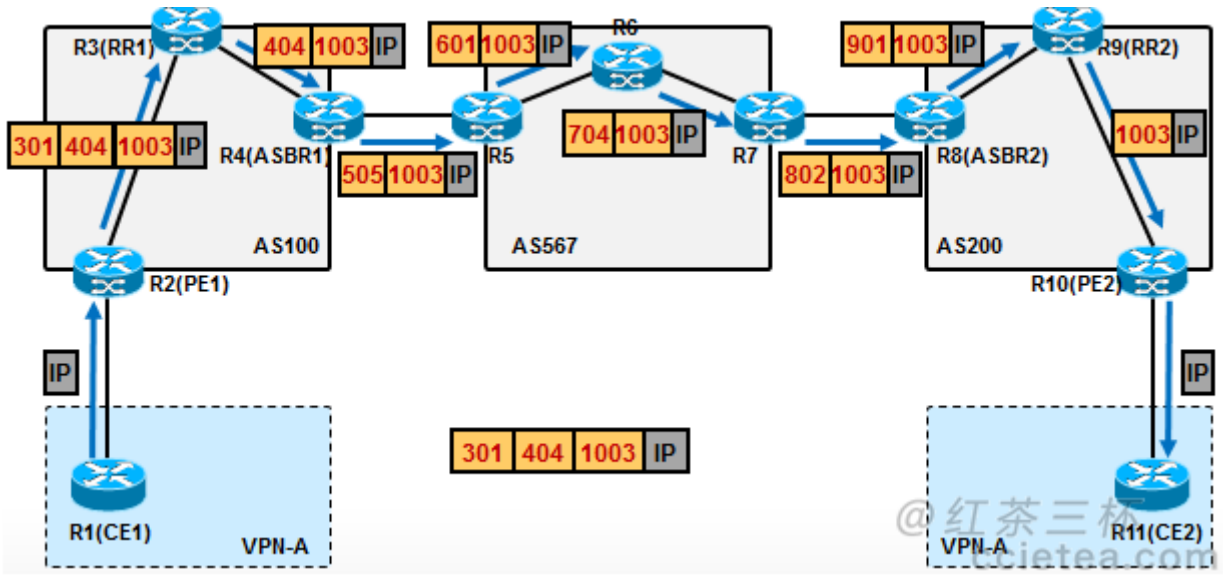
RR2 的配置如下：

```
router b 200
address-family ipv4
  neighbor 8.8.8.8 send-label
  neighbor 10.10.10.10 send-label
```

PE2 的配置如下：

```
router b 200
address-family ipv4
  neighbor 9.9.9.9 send-label
```

好，现在我们来看一下数据转发层面：



上面，是我们使用 CE1 去 traceroute 11.11.11.11 后得出的。

IP 数据包从 CE1 始发，到达 PE1，它查找的是 FIB 表：

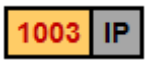
R2-PE1#show ip cef vrf cisco detail

```
11.11.11.11/32, epoch 0, flags rib defined all labels
  recursive via 10.10.10.10 label 1003
    recursive via 4.4.4.4 label 404
      nexthop 10.1.23.3 Ethernet0/1 label 301
```

注意这里出现个很奇妙的现象，CE1 去往 11.11.11.11 的数据包，被 PE1 压上了三层标签，从上面的 CEF 表项我们可以看出，其实这就是个递归的问题。由于本地去往 11.11.11.11 的路由是 BGP 的路由，递归一次得到下一跳 10.10.10.10，再递归一次得到下一跳 4.4.4.4。我们来详细看一下：

11.11.11.11/32 首先由始发的 PE2 给了一个标签，这是底层标签也就是 VPN 标签：1003；

好，那么先来一层标签：



这个标签是谁给的？PE2 给的，只有他认识，那数据转发的途中，其他路由器都不识别啊，怎么办？

我先看 11.11.11.11/32 路由的下一跳：

R2-PE1#show ip bgp vpnv4 all

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i11.11.11.11/32	**10.10.10.10**	0	100	0	200 ?

那么我还需要关于这个下一跳的路由 10.10.10.10 的标签，这是一条 BGP 路由，下一跳是 4.4.4.4，也就是 R4-ASBR1，那么我就需要 R4（以下一跳的身份）为我这条路由 10.10.10.10 分配一个标签，R4 分了标签

404 ,注意 BGP 标签分发不像 LDP 哪样是逐条进行的 ,R4 为 BGP 路由 10.10.10.10 的标签一路送到了 PE1。那么 PE1 再压上一层标签 404。

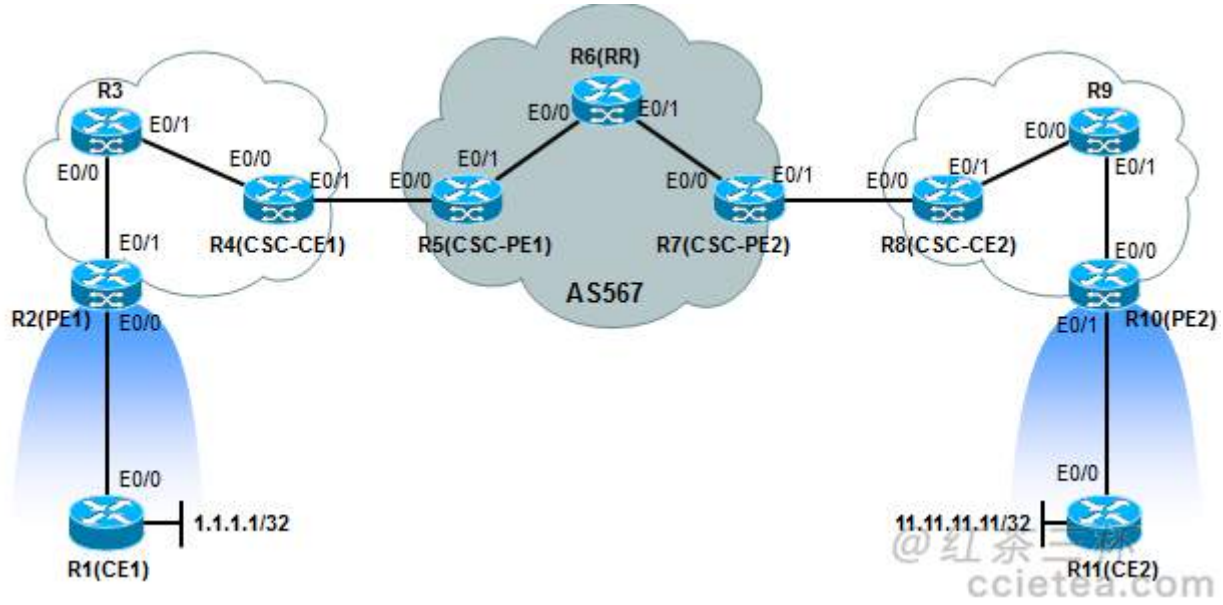
但是这个事情还没完 ,10.10.10.10 这条路由 ,还是 BGP 路由 ,又做了一次递归 ,这次得到的地址是 4.4.4.4，而这个地址或者说这条路由，是 IGP 可达的（通过 OSPF 进程 100），因此我的 R3（RR1）为 4.4.4.4 这条前缀分配了标签 301 然后传递给了 PE1，那么 PE1 又再压入一层标签 301。最终就成了这副德行。

于是这个标签包被交给了 R3-RR1，那么它查找 LFIB，发现 301 的入站标签要 POP，因此将顶层标签 POP 掉，然后交给 R4-ASBR1，R4 进一步查找 LFIB，然后对标签进行交换，接下去大家就这么重复查找-交换的动作最终把标签包传到 PE2。中间的过程我就不罗嗦了，到了这个阶段，大家应该能明白。

9.7 Carrier supporting Carrier

9.7.1 概述

9.7.2 实验示例



1. 实验环境

1) 设备互联 IP

设备互联地址空间为 10.1.xy.0/24 ,其中 x 为设备编号较小的值 ,y 为大值。最后一个八位组为设备编号。

例如 R1-R2 之间的互联，R1 的接口地址为 10.1.12.1/24，R2 的接口地址为 10.1.12.2/24，依此类推。

R9-R10 之间的连线，R9 的地址为 10.1.90.9/24，R10 的地址为 10.1.90.10/24；

R10-R11 之间的连线，R10 的地址为 10.1.101.10/24，R11 的地址为 10.1.101.11/24

2) 设备 Loopback 口 IP

所有的设备开启 Loopback0，地址为 x.x.x.x/32，x 为设备编号

3) IGP 协议的规划

ISP1 的 Backbone IGP 为 OSPF，使用进程号 100；

ISP2 的 Backbone IGP 为 OSPF，使用进程号 100；

AS567 内的 IGP 为 EIGRP，AS 号为 1；

CSC-PE1 与 CSC-CE1 之间运行的 VRF 路由协议是 OSPF，PE1 使用的进程号为 100；

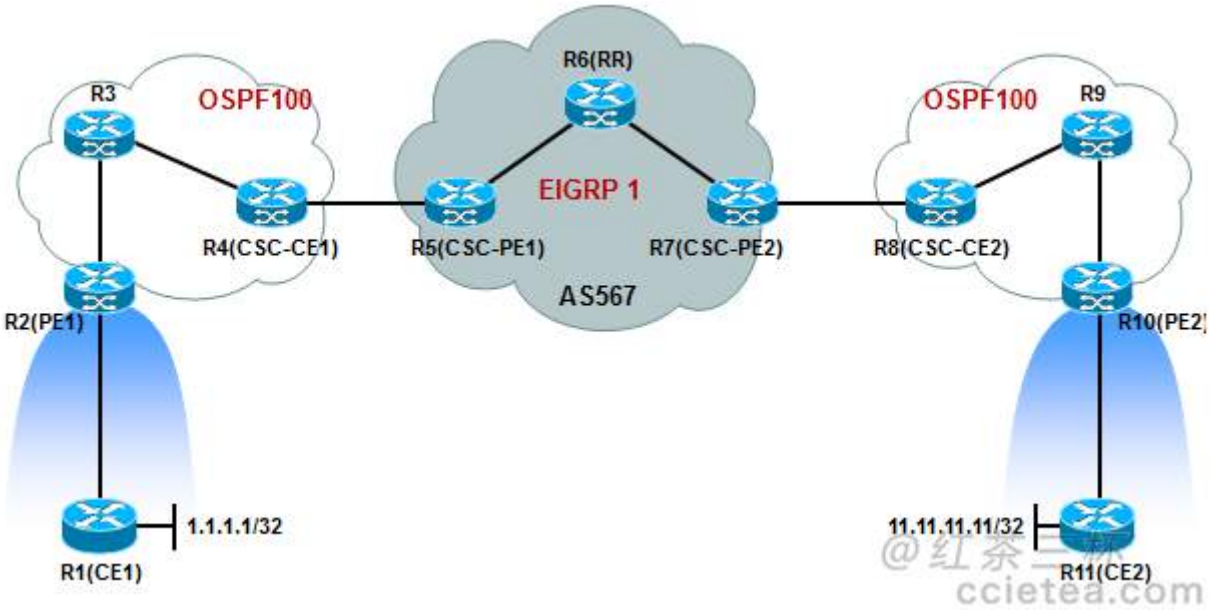
CSC-PE2 与 CSC-CE2 之间运行的 VRF 路由协议是 OSPF，PE2 使用的进程号为 100；

PE1-CE1 及 PE2-CE2 之间运行的 VRF 路由协议是 OSPF，使用的进程号为 1.

2. 实验步骤

a) 完成基本配置（Hostname、IP 地址等）

b) 完成各运营商 Backbone 内 IGP 协议的配置



PE1 的配置如下：

```
router ospf 100
   network 2.2.2.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
```

R3 的配置如下：

```
router ospf 100
   network 3.3.3.3 0.0.0.0 area 0
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
```

R4-CsC_CE1 的配置如下：

```
router ospf 100
   network 4.4.4.4 0.0.0.0 area 0
   network 10.1.34.4 0.0.0.0 area 0
   network 10.1.45.4 0.0.0.0 area 0          !! 宣告与 CsC-PE1 的直连接口
```

PE2 的配置如下：

```
router ospf 100
   network 10.1.90.10 0.0.0.0 area 0
   network 10.10.10.10 0.0.0.0 area 0
```

R9 的配置如下：

```
router ospf 100
   network 9.9.9.9 0.0.0.0 area 0
   network 10.1.89.9 0.0.0.0 area 0
```

```
 network 10.1.90.9 0.0.0.0 area 0
```

R8-CsC_CE2 的配置如下：

```
router ospf 100
 network 8.8.8.8 0.0.0.0 area 0
 network 10.1.78.8 0.0.0.0 area 0
 network 10.1.89.8 0.0.0.0 area 0
```

R5-CsC_PE1 的配置如下：

```
router eigrp 1
 network 5.0.0.0
 network 10.1.56.0 0.0.0.255
 no auto-summary
```

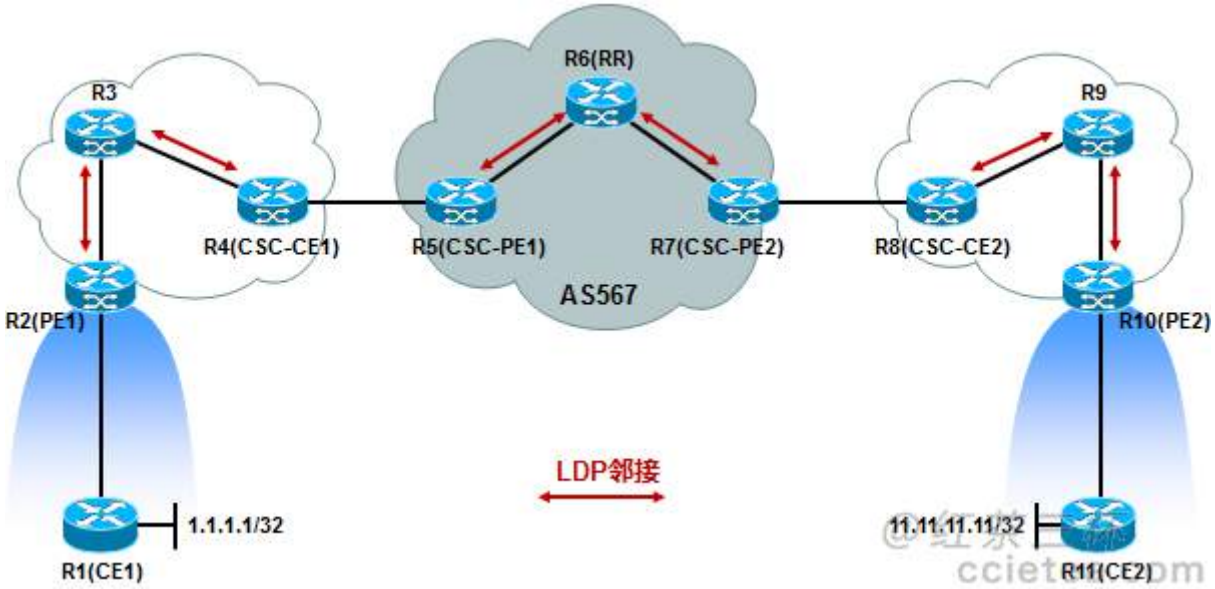
R6 的配置如下：

```
router eigrp 1
 network 6.0.0.0
 network 10.1.56.0 0.0.0.255
 network 10.1.67.0 0.0.0.255
 no auto-summary
```

R7-CsC_PE2 的配置如下：

```
router eigrp 1
 network 7.0.0.0
 network 10.1.67.0 0.0.0.255
 no auto-summary
```

c) 激活三个运营商 Backbone 内的 LDP



R2-PE1 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 200 299
interface eth0/1
  mpls ip
```

R3 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 300 399
interface eth0/0
  mpls ip
interface eth0/0
  mpls ip
```

R4-CsC_CE1 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 400 499
interface eth0/0
  mpls ip
```

R5-CsC_PE1 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 500 599
interface eth0/1
```

```
  mpls ip
```

R6 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 600 699
interface eth0/0
  mpls ip
interface eth0/1
  mpls ip
```

R7-CsC_PE2 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 700 799
interface eth0/0
  mpls ip
```

R8-CsC_CE2 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 800 899
interface eth0/1
  mpls ip
```

R9 的配置如下：

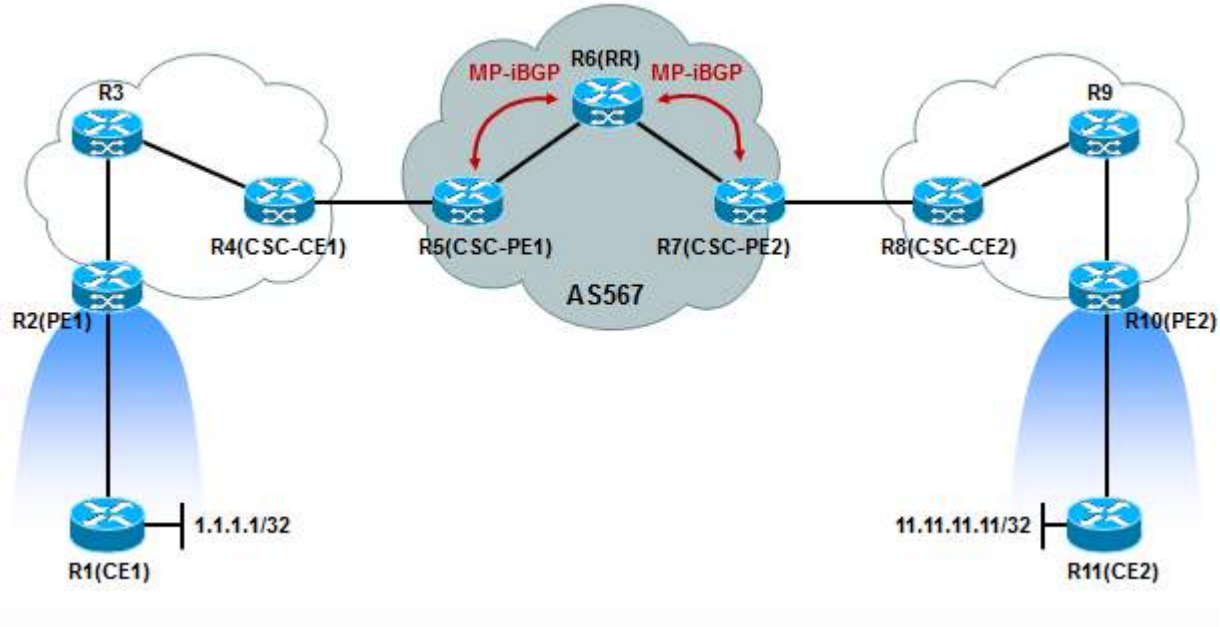
```
mpls ldp router-id Loopback0
mpls label range 900 999
interface eth0/0
  mpls ip
interface eth0/1
  mpls ip
```

R10-PE2 的配置如下：

```
mpls ldp router-id Loopback0
mpls label range 1000 1099
interface eth0/0
  mpls ip
```

d) 完成运营商 AS567 的 MPLS VPN 配置

在我们这个实验中，站在运营商 AS567 的角度，ISP1 和 ISP2 都是我的客户，那么 R5、R7 此刻的角色就是 PE ,而 R4 和 R8 此刻的身份是 CE。ISP1 及 ISP2 的 Backbone 内的路由首先通过 AS567 的 MPLS VPN Backbone 网络来运载，所以这一步我们的目标是，要使得 ISP1 和 ISP2 内的路由能够互通。



我们的主要任务有：

- 在 CsC-PE1 及 CsC-PE2 上完成 VRF 的配置
- 在 CsC-PE1 及 CsC-PE2 上完成 PE-CE 路由协议的配置
- 在 CsC-PE1 及 CsC-PE2、R6（RR）上完成 MP-iBGP 邻居关系的建立
- 在 CsC-PE1 及 CsC-PE2 上完成 VRF OSPF 进程到 MPBGP 的双向重发布

R5-CsC_PE1 的配置如下：

```
ip vrf cisco
   rd 567:1
   route-target export 567:5
   route-target import 567:7
!
interface eth 0/0
   ip vrf forwarding cisco
   ip add 10.1.45.5 255.255.255.0
!
router ospf 1 vrf cisco
   network 10.1.45.5 0.0.0.0 area 0
```

```
 redistribute bgp 567 subnets
!
router bgp 567
 no bgp default ipv4-unicast
 neighbor 6.6.6.6 remote-as 567
 neighbor 6.6.6.6 update-source Loopback0
!
 address-family vpnv4
  neighbor 6.6.6.6 activate          !!与 R6-RR 建立 MP-iBGP 邻居关系
  neighbor 6.6.6.6 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf cisco
  no synchronization
  redistribute ospf 1 vrf cisco match internal external 1 external 2
 exit-address-family
```

R6-RR 的配置如下：

```
router bgp 567
 no bgp default ipv4-unicast
 neighbor 5.5.5.5 remote-as 567
 neighbor 5.5.5.5 update-source Loopback0
 neighbor 7.7.7.7 remote-as 567
 neighbor 7.7.7.7 update-source Loopback0
 !
 address-family ipv4
  no synchronization
  no auto-summary
 exit-address-family
 !
 address-family vpnv4
  neighbor 5.5.5.5 activate
  neighbor 5.5.5.5 send-community extended
  neighbor 5.5.5.5 route-reflector-client
```

```
    neighbor 7.7.7.7 activate
    neighbor 7.7.7.7 send-community extended
    neighbor 7.7.7.7 route-reflector-client
  exit-address-family
```

R7-CsC_PE2 的配置如下：

```
ip vrf cisco
  rd 567:2
  route-target export 567:7
  route-target import 567:5
!
interface Ethernet0/1
  ip vrf forwarding cisco
  ip address 10.1.78.7 255.255.255.0
!
router ospf 1 vrf cisco
  redistribute bgp 567 subnets
  network 10.1.78.7 0.0.0.0 area 0
!
router bgp 567
  no bgp default ipv4-unicast
  neighbor 6.6.6.6 remote-as 567
  neighbor 6.6.6.6 update-source Loopback0
 !
  address-family ipv4
    no synchronization
    no auto-summary
  exit-address-family
 !
  address-family vpnv4
    neighbor 6.6.6.6 activate
    neighbor 6.6.6.6 send-community extended
  exit-address-family
 !
```

```
 address-family ipv4 vrf cisco
    no synchronization
    redistribute ospf 1 vrf cisco
 exit-address-family
```

完成这一步配置后，我们来看一下：

R5-CsC-PE1#show ip bgp vpnv4 al

```
BGP table version is 19, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 567:1 (default for vrf cisco)
*> 2.2.2.2/32       10.1.45.4                31         32768 ?
*> 3.3.3.3/32       10.1.45.4                21         32768 ?
*> 4.4.4.4/32       10.1.45.4                11         32768 ?
*>i8.8.8.8/32       7.7.7.7                  11    100        0 ?
*>i9.9.9.9/32       7.7.7.7                  21    100        0 ?
*> 10.1.23.0/24     10.1.45.4                30         32768 ?
*> 10.1.34.0/24     10.1.45.4                20         32768 ?
*> 10.1.45.0/24     0.0.0.0                   0         32768 ?
*>i10.1.78.0/24     7.7.7.7                   0    100        0 ?
*>i10.1.89.0/24     7.7.7.7                  20    100        0 ?
*>i10.1.90.0/24     7.7.7.7                  30    100        0 ?
*>i10.10.10.10/32   7.7.7.7                  31    100        0 ?
Route Distinguisher: 567:2
*>i8.8.8.8/32       7.7.7.7                  11    100        0 ?
*>i9.9.9.9/32       7.7.7.7                  21    100        0 ?
*>i10.1.78.0/24     7.7.7.7                   0    100        0 ?
   Network          Next Hop            Metric LocPrf Weight Path
*>i10.1.89.0/24     7.7.7.7                  20    100        0 ?
*>i10.1.90.0/24     7.7.7.7                  30    100        0 ?
```

```
*>i10.10.10.10/32     7.7.7.7                  31       100          0 ?
```

CsC-PE1 已经学习到了 ISP1 内的路由，同时 ISP2 的内的路由也过来了。

通过查看 PE1 及 PE2 的路由表，会发现，两个 ISP：ISP1 及 ISP2 都已经学习到了对端的路由。

但是能否 ping 通呢？在 PE1 上我们去 ping 10.10.10.10 也就是 PE2 的 Loopback ,发现能够 ping 通(这里其实是有一点潜在问题的，等会我们再分析，虽然是 ping 通了，但是大家可以自己去琢磨一下，看看标签，这里可能会有什么潜在问题)。

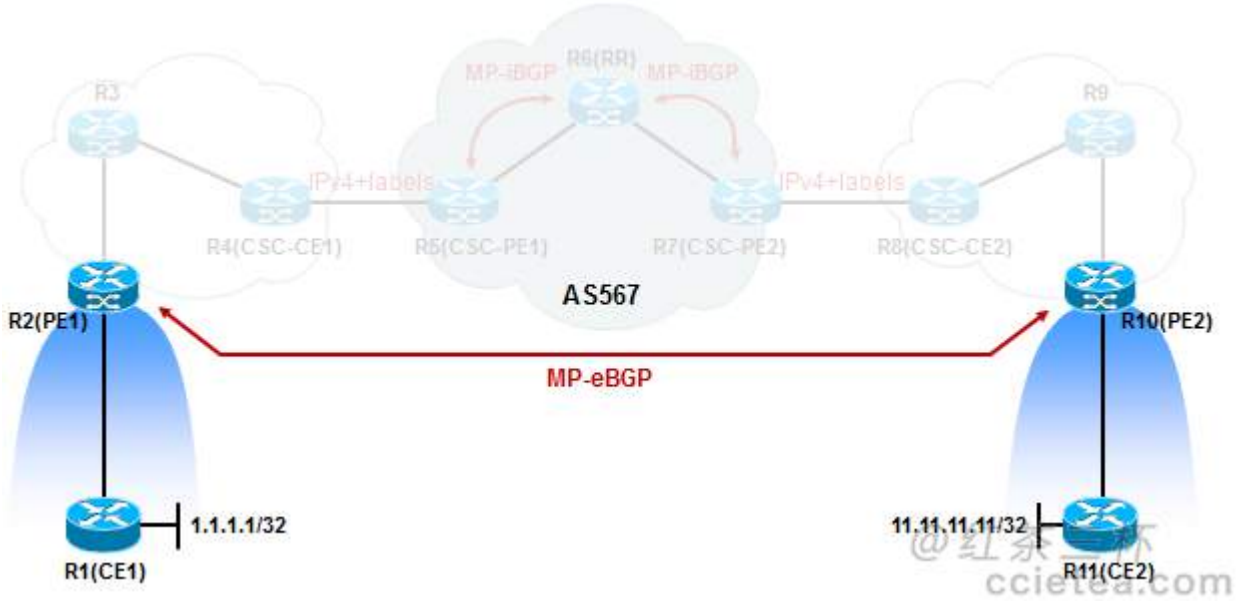
e) **完成 PE1 及 PE2 的 MP-eBGP 邻居关系的建立**

R2-PE1 的配置如下：

```
router bgp 234
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 10.10.10.10 remote-as 890
  neighbor 10.10.10.10 ebgp-multihop 255                    !!注意 ebgp 多条的配置
  neighbor 10.10.10.10 update-source Loopback0
 !
  address-family ipv4
     no synchronization
     no auto-summary
  exit-address-family
 !
  address-family vpnv4
     neighbor 10.10.10.10 activate                          !!激活与 PE2 之间的 VPNv4 地址族
     neighbor 10.10.10.10 send-community extended
  exit-address-family
```

如此一来，PE1 与 PE2 之间的 MP-eBGP 邻居关系就建立好了。

f) **完成 PE1、PE2 上的 VRF 配置；完成 PE1-CE1、PE2-CE2 的 VRF OSPF 进程配置；完成 PE1、PE2 上的 VRF IGP 协议与 MP-BGP 协议的双向重发布。**



我们的主要任务有：

- 在 PE1 及 PE2 上完成 VRF 的配置
- 在 PE1 及 PE2 上完成 PE-CE 路由协议的配置
- 在 PE1 及 PE2 上完成 VRF OSPF 进程到 MPBGP 的双向重发布

PE1 的配置如下：

```
ip vrf cisco
   rd 234:1
   route-target export 234:2
   route-target import 890:10
!
interface Ethernet0/0
   ip vrf forwarding cisco
   ip address 10.1.12.2 255.255.255.0
!
router ospf 1 vrf cisco
   network 10.1.12.2 0.0.0.0 area 0
   redistribute bgp 234 subnets
!
router bgp 234
   address-family ipv4 vrf cisco
      redistribute ospf 1 vrf cisco match internal external
```

CE1 的配置如下：

```
interface Ethernet0/0
 ip address 10.1.12.1 255.255.255.0
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
router ospf 1
 network 1.1.1.1 0.0.0.0 area 0
 network 10.1.12.1 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip vrf cisco
   rd 890:2
   route-target export 890:10
   route-target import 234:2
!
interface Ethernet0/1
   ip vrf forwarding cisco
   ip address 10.1.101.10 255.255.255.0
!
router ospf 1 vrf cisco
   network 10.1.101.10 0.0.0.0 area 0
   redistribute bgp 234 subnets
router bgp 890
   address-family ipv4 vrf cisco
      redistribute ospf 1 vrf cisco match internal external
```

CE2 的配置如下：

```
interface Ethernet0/0
 ip address 10.1.101.11 255.255.255.0
!
interface Loopback0
 ip address 11.11.11.11 255.255.255.255
```

```
!
router ospf 1
 network 11.11.11.11 0.0.0.0 area 0
 network 10.1.101.10 0.0.0.0 area 0
```

完成这一步的配置后，我们在 CE1 上看一下：

R1-CE1#sh ip ro

```
O IA        10.1.101.0/24 [110/11] via 10.1.12.2, 00:00:29, Ethernet0/0
O IA        11.11.11.11 [110/21] via 10.1.12.2, 00:00:29, Ethernet0/0
```

CE1 已经学习到 VPN site2 传递过来的路由了。说明，我们两个客户的路由已经都传递到位了。也就是说，路由控制层面的问题我们都已经解决了。

那么 CE1、CE2 是否就能互访了呢？

CE1 上，我们去 ping 11.11.11.11 发现并不通。为什么不通呢？我们的分析一下。

- **首先 CE1 将数据包交到了 PE1，那么 PE1 查找自己的 FIB 表：**

  ```
  R2-PE1#sh ip cef vrf cisco detail
  11.11.11.11/32, epoch 0, flags rib defined all labels
    recursive via 10.10.10.10 label 1005
      nexthop 10.1.23.3 Ethernet0/1 label 308
  ```

  PE1 根据 FIB 表项的指示，将原始 IP 数据包先压入 1005 的底层标签，再压入 308 的顶层标签，然后将数据丢给 23.3 也就是 R3。

- **R3 查看自己的 LFIB 表：**

  R3#show mpls forwarding-table

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
…	…				
308	408	10.10.10.10/32	366	Et0/1	10.1.34.4

  根据条目所示，R3 将顶层标签 308 置换成 408，然后交给 34.4 也就是 R4-CsC_CE1。

- **标签包到了 R4，R4 也是查看自己的 LFIB 表：**

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
…	…				
408	**No Label**	10.10.10.10/32	254	Et0/1	10.1.45.5

结果，408 的入站标签，对应的出站动作是 untagged，这就是问题所在了，那么为什么会出现这种现象呢，回顾一下此前我们的配置，其实在 CsC-CE1 与 CsC-PE1 之间，以及 CsC-CE2 和 CsC-PE2 之间，我们是没有激活 LDP 邻接的，并且在我们此前的实验中，PE-CE 之间，也不需要激活 LDP 邻接，但是在这里不同了，我们需要 10.10.10.10 的标签，但是可惜，CsC-PE1 有路由但是下游邻居没给标签，那么我就只能 untagged 了。

所以，现在我们要激活 CsC-CE1 与 CsC-PE1 之间，以及 CsC-CE2 和 CsC-PE2 之间的 LDP 邻居关系。

R4- CsC-CE1 的补充配置如下：

```
Interface eth0/1
  mpls ip
```

R5-CsC_PE1 的补充配置如下：

```
Interface eth0/0
  mpls ip
```

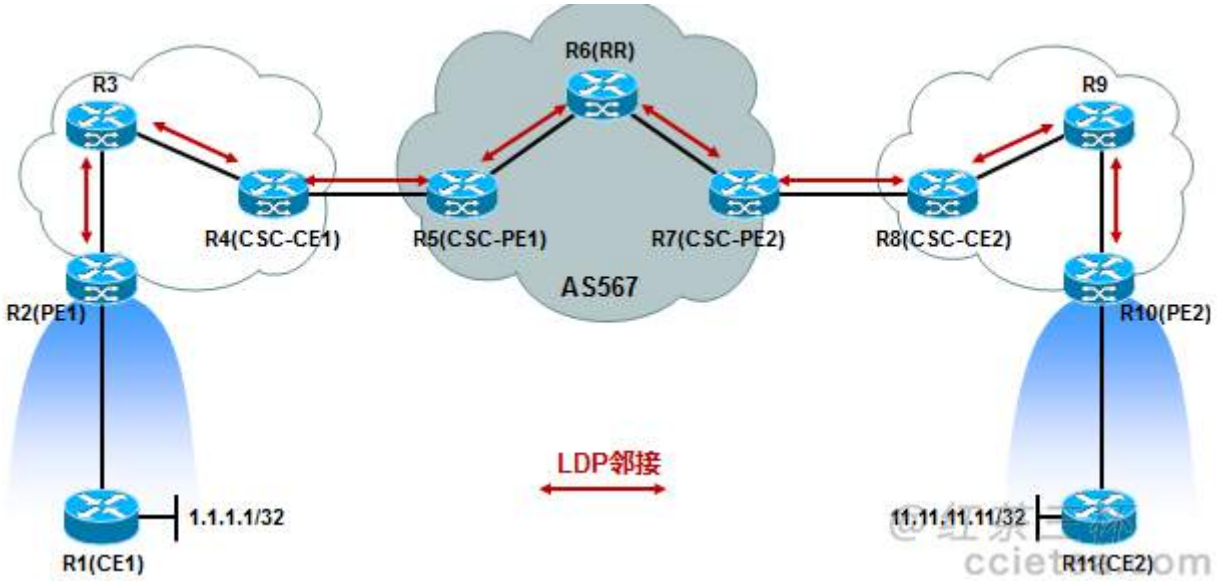
R7- CsC-PE2 的补充配置如下：

```
Interface eth0/1
  mpls ip
```

R8-CsC_CE2 的补充配置如下：

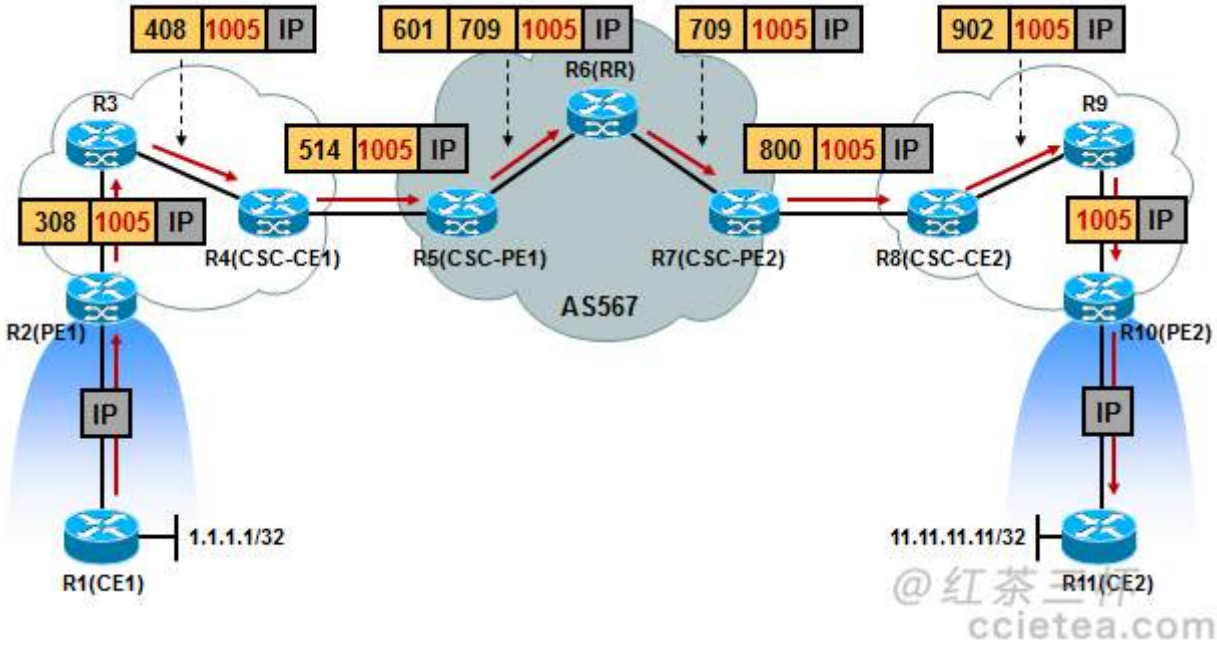
```
Interface eth0/0
  mpls ip
```

那么到目前位置，我们的 LDP 邻接关系如下：



3. **理解数据转发层面，与路由控制层面**

我们现在 CE1 上来 traceroute 一下 11.11.11.11：



- **发往 11.11.11.11 的 IP 数据包送到了 PE1**

  PE1 仍然是查看自己的 FIB 表：

  R2-PE1#sh ip cef vrf cisco detail

  11.11.11.11/32, epoch 0, flags rib defined all labels

  recursive via 10.10.10.10 **label 1005**

  nexthop 10.1.23.3 Ethernet0/1 **label 308**

  表项显示，数据包需要压入两层标签：

  内层标签是由 PE2 分配给 VPNv4 前缀 890:2:11.11.11.11/32 的、通过 MP-eBGP 传递过来的；

  顶层标签是由 R3 分配给 VPNv4 前缀 890:2:11.11.11.11/32 的 NEXT_HOP：10.10.10.10 的。

  于是【308】【1005】【IP-_-】上路了。

- **标签包到达了 R3**

  R3 查看自己的 LFIB 表：

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
308	**408**	10.10.10.10/32	10618	Et0/1	10.1.34.4

  于是将顶层标签置换成 408，得到：

  【408】【1005】【IP-_-】，然后交给 R4-CsC_CE1。

- **标签包到达了 R4-CsC_CE1**

  这哥们也查看自己的 LFIB 表：

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
408	**514**	10.10.10.10/32	8924	Et0/1	10.1.45.5

于是将顶层标签置换成 514，得到：

【514】【1005】【IP-_-】，然后交给 R5-CsC_PE2。

- **标签包到了 R5-CsC_PE2**

注意，这时候，R5 是从自己的 VRF 接口上收到的这个数据包。

那么 R5 会怎么做呢？

首先根据 LFIB 表项的内容：

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
514	**709**	10.10.10.10/32[V]	0	Et0/1	10.1.56.6

入站标签 514 需要置换成 709，而这个 709 的标签是一个为 VPNv4 前缀所分配的标签，为哪个 VPNv4 前缀？为 10.10.10.10/32 对应的 VPNv4 前缀？由谁分配的？由 R7-CsC_PE2 分配的。那么如果直接将 514 标签置换成 709，然后扔出去，这个标签包就有可能在 AS567 的茫茫大海中被丢弃，因为 Backbone 里的 P 路由器是无法理解 VPN 标签的。所以，还需要给这层标签再套上一层 IGP 标签，用于标签包在 Backbone 内的传输对吧？

其实，这时候，整个 AS234 和 AS890 在 AS567 的眼里就是客户网络，那么 AS890 作为一个 MPLS VPN BACKBONE 来说，收到一个 VPN 客户的数据，按我们前面做的那些个实验，不都是要压入两层标签的么？

所以 R5 手打这个标签包【514】【1005】【IP-_-】，首先将顶层标签置换成 709，然后还需要一层 IGP 标签，这层标签是数据转发的下一跳路由器 R6 为 7.7.7.7（也就是 10.10.10.10 的下一跳）所分配的。最后得到的标签包就是：

【601】【709】【1005】【IP-_-】，然后交给 R6

- **R6 收打了标签包后，查看自己的 LFIB 表：**

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
600	Pop Label	5.5.5.5/32	21463	Et0/0	10.1.56.5
601	**Pop Label**	7.7.7.7/32	18143	Et0/1	10.1.67.7

于是它将顶层标签 601 弹出，然后将弹出后的标签包【709】【1005】【IP-_-】交给 R7。

这里实际上是有个次末跳弹出的机制，大家应该都能理解，因为 7.7.7.7 是 R7 的直连口。

- **R7 收打了标签包后，查看自己的 LFIB 表：**

Local Label	Outgoing Label	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
709	800	10.10.10.10/32[V]	0	Et0/1	10.1.78.8

顶层标签 709 被置换成了 800，得到【800】【1005】【IP-_-】然后交给 R8

- **R8 根据自己的 FLIB 表继续对顶层标签进行置换，得到：**
  【902】【1005】【IP-_-】然后交给 R9

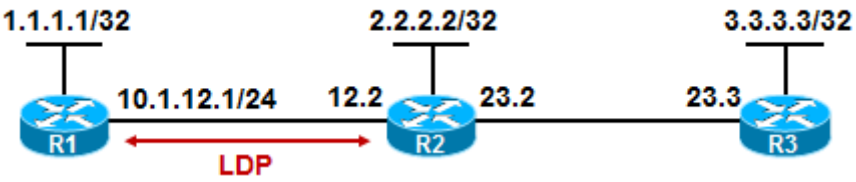
- **R9 根据自己 LFIB 表的查找结果。对顶层标签进行 POP，得到：**
  【1005】【IP-_-】 然后交给 R10

- **R10 收到数据包，将标签栈移除，然后将里头的【IP-_-】萌货交给 CE2 任由她处置。**

10疑难杂症

10.1 关于标签的分发问题

【实验 1】 LDP 是否会为没有激活 LDP（配置 MPLS ip）的接口所在网段的路由前缀分配标签



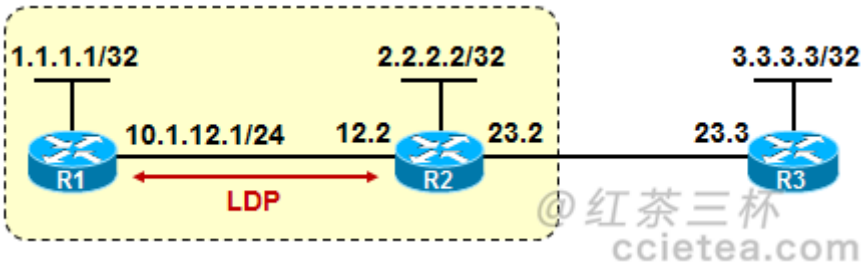
实验背景：

- 我们重点看 R1 和 R2，全网没有任何动态路由协议、没有配置静态路由
- R1、R2 之间激活 LDP 邻居关系。注意两者的 LDP routerID 都手工改成了自己直连接口的 IP 因此不用配置到对方 RouterID 的路由即可建立起 LDP 邻接
- 两者的 Loopback0 是都没有激活 LDP 的

实验结论：

- LDP 会为路由表中的所有路由分配标签（除了 BGP 路由）。
- 例如 R1，会为 1.1.1.1/32 及 10.1.12.0/24 分配标签，并且将标签映射传递给 R2。为这两个直连路由分配的标签都是 3，也就是 POP。虽然说 R2 收到些个标签映射，没有任何效果。因为对于 R2 而言，收到了 R1 捆绑给 1.1.1.1/32 的标签，但是本地路由表没有 1.1.1.1/32 的路由，因此直接忽略，所以 LFIB 表中看到一个条目的前提是，路由表里得有这个前缀。

【实验 2】 从上游 IGP 邻居那收到了路由，但是没有收到该邻居为该路由捆绑的标签



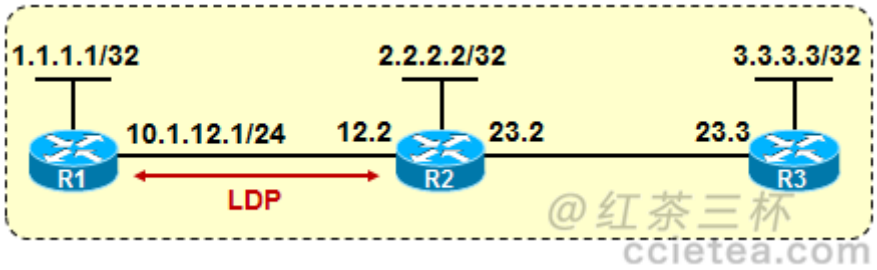
实验背景：

- R1、R2 建立 OSPF 邻居关系，同时通告各自的 Loopback 口
- R1、R2 激活 LDP 邻接；

实验结论：

- 正常情况下，R2 上学习到来自 R1 的路由 1.1.1.1/32，路由的下一跳是 R1 没错，同时也收到了这个下一跳为前缀 1.1.1.1/32 捆绑的标签（POP），因此此刻 R2 的 LFIB 表正常。Outgoing 标签为 POP
- 现在我们保持现状，但是在 R1 上，使用 no mpls ldp advertise-labels，不让丫分发任何标签，那么这样一来，R2 将收不到来自 R1 的关于 1.1.1.1 分配的标签，这就是说，丫给了路由但是丫没给这路由分标签，以为着什么？意味 R2 认为，既然给了路由没给标签，有可能通过 IP 可达，于是 R2 将 1.1.1.1/32 的 outgoing 动作设置为 untagged，也就是弹出所有标签栈，然后进行 IP 转发。

【实验 3】 从非 LDP 邻居那学习到路由，这条路由在 LFIB 中的状态？



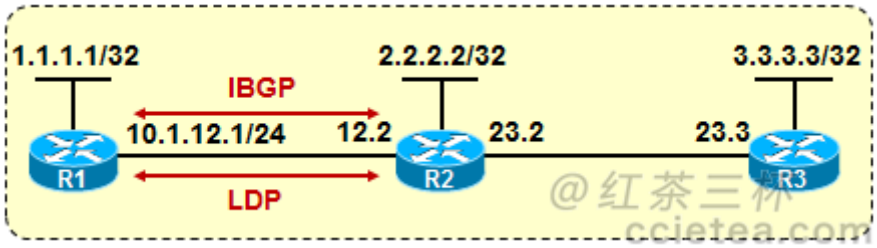
实验背景：

- R1、R2、R3 运行 OSPF，全网可达
- R1、R2 激活 LDP 邻接；R2、R3 之间没有 LDP 连接

实验结论：

- R2 能学习到 R3 传递过来的关于 3.3.3.3/32 的路由，并放入全局路由表
- 路由表有了这条前缀，那么 LFIB 表里也就有了条目，但是有了前缀，标签呢？这条路由的下一跳 R3 没有给我标签啊，因此此刻 LFIB 表中，3.3.3.3/32 这条路由，outgoing 是 untagged 的。这个实验测试有点~基本上跟实验 2 的思想是一样的，随便鼓捣鼓捣。

【实验 4】 MPLS 不会为 BGP 路由分配标签



实验背景：

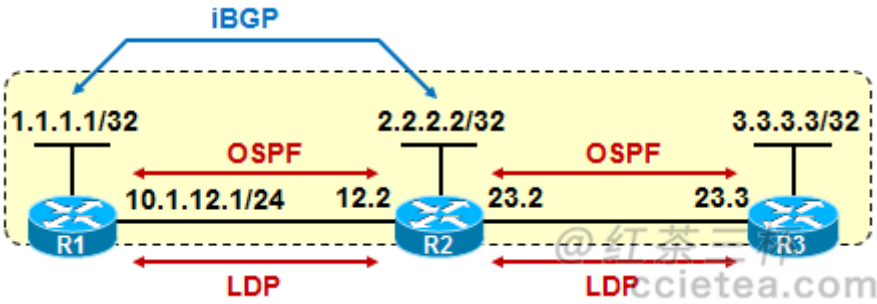
- R1、R2、R3 运行 OSPF，宣告各自的 Loopback0（如图中标识），全网路由可达
- R1、R2 激活 LDP 邻接
- R1-R2 之间建立基于 Loopback 口的 iBGP 邻居关系。
- R1 上新开一个 Loopback1，地址为 11.11.11.11/32，将该直连路由 network 进 BGP
- R2 通过 iBGP 学习到了 11.11.11.11/32，路由表里为 B 11.11.11.11/32

实验结论：

- 拿 R2 来说，它能通过 iBGP 学习到 1.1.1.1/32 的路由，并且装载进了路由表，表里是 B 11.11.11.11/32，下一跳是 1.1.1.1。

- 注意此时 R1 与 R2 之间有 LDP 的邻居关系 ,R1 为本地直连 11.11.11.11/32 分配了 POP 标签并且将标签映射传递给了 R2
- 但是在 R2 的 LFIB 表中，压根就没 11.11.11.11/32 的影儿。也就是说如果是 LSR 的路由表里有条 BGP 路由，LDP 压根就不会在本地给这条前缀分配 LOCAL 标签，而且也不会在 LFIB 表里体现这条 BGP 路由。

【实验 5】 让 MPLS 为 BGP 路由分配标签

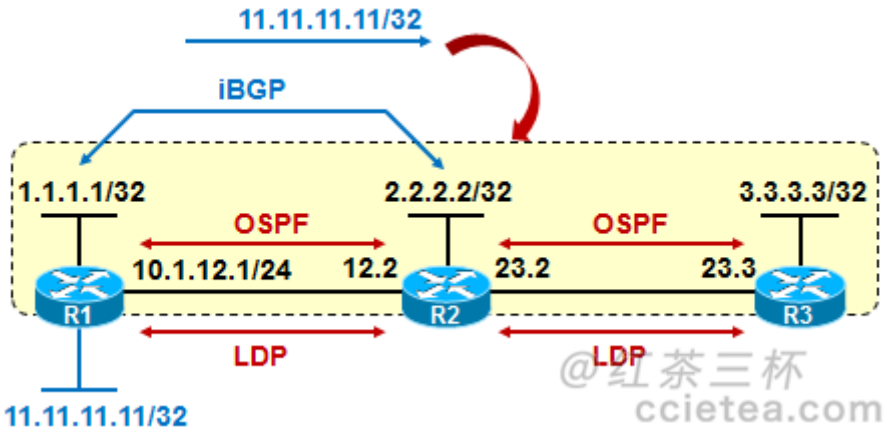


实验背景：

- R1、R2、R3 运行 OSPF，宣告各自的 Loopback0（如图中标识），全网路由可达
- R1-R2 激活 LDP 邻接；R2-R3 激活 LDP 邻接
- R1-R2 之间建立基于 Loopback 口的 iBGP 邻居关系。
- R1 上新开一个 Loopback1，地址为 11.11.11.11/32，将该直连路由 network 进 BGP
- R2 通过 iBGP 学习到了 11.11.11.11/32，路由表里为 B 11.11.11.11/32

实验结论：

- 现在，在 R2 上，将学习到的 BGP 路由 11.11.11.11/32 重发布进 OSPF，让 R3 也能学习到。



- R3 学习到这条外部路由 OE2 11.11.11.11/32。那么 此刻在 R3 的 LFIB 表里 ,关于 11.11.11.11/32 的路由，就是 Untagged 的，为啥？因为给它路由的 R2 并没有给路由捆绑标签啊（在 R2 上该

路由是 BGP 的），有路由没标签，那我就 untagged 了。而为啥 R2 不给标签，因为我们前面说了，LDP 不会给 BGP 路由分配标签的。

- 此刻我们在 R2 上：

R2#sh ip b 11.11.11.11

```
BGP routing table entry for 11.11.11.11/32, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    1.1.1.1 (metric 2) from 1.1.1.1 (11.11.11.11)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

从 R1 收到的 11.11.11.11 条目是这样的，先马克一下。

- 接下去，我们在 R1-R2 之间配置 send-label

R1 的配置如下：

```
router bgp 12
  neighbor 2.2.2.2 send-label
```

R2 的配置如下：

```
router bgp 12
  neighbor 1.1.1.1 send-label
```

现在我们再去看看：

R2#sh ip b 11.11.11.11

```
BGP routing table entry for 11.11.11.11/32, version 6
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  Local
    1.1.1.1 (metric 2) from 1.1.1.1 (11.11.11.11)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      mpls labels in/out 202/imp-null
```

这条 send-label 命令使得 BGP 进行了多协议的扩展，BGP 开始为 IPv4 前缀分配标签了。

R1 发送给 R2 的 BGP update 包里，采用了 MP_REACH_NLRI，就包含了标签信息：

```
Path attributes
  ORIGIN: IGP (4 bytes)
  AS_PATH: empty (3 bytes)
  MULTI_EXIT_DISC: 0 (7 bytes)
  LOCAL_PREF: 100 (7 bytes)
  MP_REACH_NLRI (20 bytes)
    Flags: 0x80 (Optional, Non-transitive, Complete)
    Type code: MP_REACH_NLRI (14)
    Length: 17 bytes
    Address family: IPv4 (1)
    Subsequent address family identifier: Labeled Unicast (4)
    Next hop network address (4 bytes)
    Subnetwork points of attachment: 0
    Network layer reachability information (8 bytes)
      Label Stack=3 (bottom) IPv4=11.11.11.11/32
        MP Reach NLRI Prefix length: 56
        MP Reach NLRI Label Stack: 3 (bottom)          标签
        MP Reach NLRI IPv4 prefix: 11.11.11.11 (11.11.11.11) 前缀
```

当然，R1 更新给 R2 的 LDP 地址映射消息里，也就开始携带 11.11.11.11 的前缀和分配给该前缀的标签了。

现在我们再来看看 R2 上的 LFIB 表：

R2#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	1.1.1.1/32	0	Fa0/0	10.1.12.1
201	Pop tag	3.3.3.3/32	0	Fa1/0	10.1.23.3
202	**Pop tag**	**11.11.11.11/32**	**0**	**Fa0/0**	**10.1.12.1**

11.11.11.11/32 威武的出现了，下一跳 10.1.12.1 也就是 R1 分配的标签是 POP，然后 R2 本地为该前缀分配的标签是 202。好了，那么 R2 给 BGP 路由 11.11.11.11/32 分配了标签，它就会将标签映射消息通过 LDP 发送给 R3。

```
Forwarding Equivalence Classes TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Forwarding Equivalence Classes TLV (0x100)
    TLV Length: 8
  FEC Elements
    FEC Element 1
      FEC Element Type: Prefix FEC (2)
      FEC Element Address Type: IPv4 (1)
      FEC Element Length: 32
      Prefix: 11.11.11.11
Generic Label TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Generic Label TLV (0x200)
    TLV Length: 4
    Generic Label: 202
```

如此一来，在 R3 上：

R3#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
300	200	1.1.1.1/32	0	Fa0/0	10.1.23.2
301	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.23.2
302	Pop tag	10.1.12.0/24	0	Fa0/0	10.1.23.2
303	**202**	**11.11.11.11/32**	**0**	**Fa0/0**	**10.1.23.2**

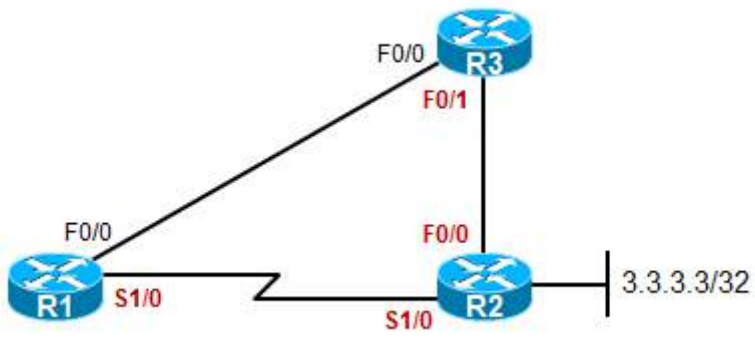
也就有了出站标签了。

- 所以在 R1、R2 之间互相配置 send-label 后，R1-R2 的 IPv4 BGP 邻居关系会重建，Open 消息中会包含新的 ability，也就是对 labeled unicast 支持能力的地址族。

  这个特性使得 BGP 得到多协议的扩展，以便对 BGP 路由分配标签，并通过 LDP 的支持将标签映射传递给其他的非 BGP 的 LDP 邻居。

10.2 Untagged、Aggregate

1. 关于 untagged



实验环境：

- 拓扑如上图所示，设备互联网段 IP 地址空间采用 10.1.xy.0/24，其中 x 及 y 为设备编号，IP 地址的最后一个 8 位组为设备编号。所有设备的 loopback0 地址为 x.x.x.x/32

- 全网运行 OSPF 协议，全网路由可达
- 拓扑中，接口编号为红色粗体的接口，激活 LDP
- 配置比较简单，我这里就不贴配置了，关键点是注意 1：全网互通，2：跑 MPLS，但是 R1 的 F0/0 及 R3 的 F0/0 不运行 MPLS 也就是不激活 LDP。

实验现象：

完成实验后，R1 的 LFIB 表如下：

R1#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	**Untagged**	2.2.2.2/32	0	Fa0/0	10.1.13.3
101	**Untagged**	3.3.3.3/32	0	Fa0/0	10.1.13.3
102	**Untagged**	10.1.23.0/24	0	Fa0/0	10.1.13.3

我们发现，2.2.2.2、3.3.3.3、10.1.23.0 这个前缀，我本地都分配了标签，但是，outgoing 标签却是 untagged。为什么会出现这样的问题呢？

我们知道 MPLS 的工作是基于 IP 路由表及 CEF 表、LIB 表进行的，LSR 标签交换路由器会为自己本地路由表的路由产生标签，并且将路由前缀与自己为该前缀所分配的标签的映射发给 LDP 邻居。那么当一台 LSR 在收到 LDP 邻居发送给我的标签映射后，我会进行一系列的检查。首先这条路由前缀在我的路由表里要有，在者，路由的下一跳是否就是传标签映射给我的这个 LDP 邻居。一台 LSR 有可能会从多个 LDP 邻居收到为同一个路由前缀捆绑的标签，而我只会使用去往该路由前缀的下一跳（路由器）分配的标签。

那么回到上面的拓扑，对于 R1 而言，2.2.2.2、3.3.3.3、10.1.23.0 这三条路由前缀，R1 会从 R2 及 R3 都学习到，但是由于 COST 值的关系，R1 会优选 R3 的路由，也就是 R3 是 R1 去往该三个网段的实际下一跳。然而，可惜的 R1 及 R3 之间并非 LDP 邻居，可不就是世界上最遥远的距离么？R2 呢，为这三条路由捆绑了标签，并且将标签映射发给了 R1，可惜啊，R1 的心这会儿在人家 R3 上，它就忽略了 R2 的标签映射（通过 debug mpls ldp bindings 可以看到 R1 直接 omit 了 R2 发过来的标签映射），因此，就造成了 R1 的 LFIB 的这个状态。

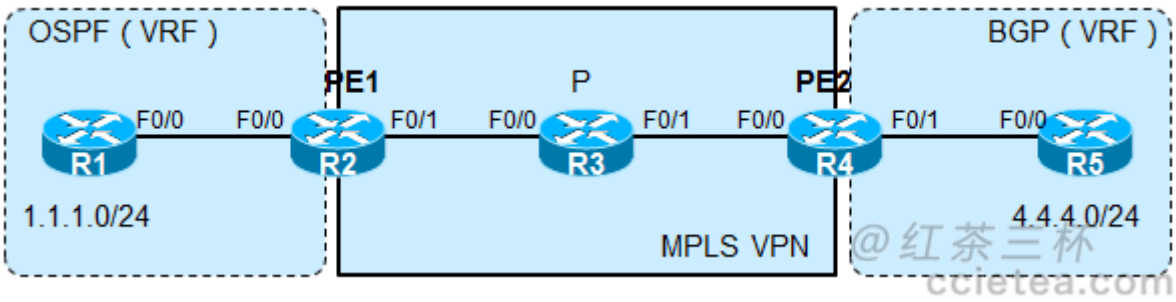
这是一个产生 untagged 标签动作的典型问题。

实验小结：

那么 untagged 是一个怎样的操作呢？当一台 LSR 收到一个标签包，它会去查看 LFIB 表，如果 outgoing 动作是 untagged，那么意味着该 LSR 认为这条路由 IP 可达，并且下一跳可能并不支持 MPLS，只是 IP 设备，于是它会将标签包的所有标签全部弹出，然后根据 LFIB 表里所示的 ougoing interface 及 next-hop 将该 IP 包扔出去，注意这时候不需要拿着这个 IP 包去查 CEF 表，因为 LFIB 表中已经有我们所需的转发信息了。

这在我们这个实验环境中看似没什么问题，但是，在大多数 MPLS 环境中，例如 MPLS VPN 环境中，这个问题就大条了。

2. 关于 aggregate



实验环境：

- R1、R5 为 CE 路由器。R1 运行 OSPF；R5 运行 EBGP
- R2、R4 为 PE 路由器。R2 运行 VRF 的 OSPF，以获取 R1 的客户路由，重点关注 1.1.1.0/24；R4 运行 VRF 的 EBGP，以获取 R5 的客户路由，重点关注 4.4.4.0/24
- R2、R3、R4 为 MPLS VPN Backbone ,Core 内运行 OSPF ,使得三台路由器都能获取到各自的 Loopback 路由。Loopback 均使用 x.x.x.x/32。R3 为 P 路由器，同时是 RR。
- PE1 的 Fa0/1、R3 的 F0/0 及 F0/1、R4 的 F0/0 激活 LDP

实验现象

观察一下 R2 的 LFIB 表：

R2#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	3.3.3.3/32	0	Fa0/1	10.1.23.3
201	Pop tag	10.1.34.0/24	0	Fa0/1	10.1.23.3
202	301	4.4.4.4/32	0	Fa0/1	10.1.23.3
204	**Aggregate**	**10.1.12.0/24[V]**	**0**		
205	**Untagged**	**1.1.1.0/24[V]**	**0**	**Fa0/0**	**10.1.12.1**

首先我们看到，在 R1 的 LFIB 表中，vpnv4 前缀 1.1.1.0/24 的 outgoing 动作是 untagged，为什么呢？经过前面对 untagged 的分析，很容易得出，1.1.1.0 的这条前缀来自 CE 路由器 R1，R1 是我 R2 去往 1.1.1.0 的实际下一跳，但是该下一跳并没有激活 LDP，也就没有给我标签映射，因此，这条路由是 untagged 的。那么当 R2 收到一个带着标签“205”的数据，我就会将该标签包的所有标签栈全部弹出，然后丢给 F0/0 口的 10.1.12.1。

然后我们又看到 10.1.12.0/24 这条 vpnv4 前缀，它的动作的 aggregate，因为这条路由是我本地的 VRF 直连路由，既然是我的直连，那么如果有数据要到达该网段，那到我这应该是最后一跳了吧？肯定是要移除标签栈，并且做 IP 查找的，这就是 aggregate 的动作。要注意和本地的全局路由表的直连路由区分开，R2 为自己的全局路由表的直连路由 2.2.2.2/32 分配的是 pop 标签，并且将标签映射传递给了 R3。而这里 10.1.12.0 的 VRF 的直连路由，所以实际上在弹出标签之后，还有其他的查找动作要做。

我们知道 untagged 的 LFIB 表项，里头是有出接口及下一跳信息的，但是留意一下 aggregate 表项，出接口及下一跳的信息都是空的，正是因为这是 VRF 的路由，因此，当我收到一个标签包，经过本地 LFIB 查找后发现是 aggregate，我会先移除标签，然后做进一步的 CEF 查找，而这个 CEF 查找又很有可能是 VRF 的 CEF。

另一个产生 aggregate 的诱因是路由汇总，例如在上图中，PE1 也就是 R2，收到了来自 R1 的 VPN 路由 1.1.1.0/24。如果在 R2 的 BGP 进程中，address-family ipv4 vrf xxx 下，对 1.1.1.0/24 进行手工的 BGP 汇总，那么 R2 将本地始发一条 BGP 的汇总路由，如 1.0.0.0/8，那么对于 R2 而言这条路由是本地始发的路由，因此也是 aggregate 动作。如下：

```
router bgp 234
<!—省略部分配置-->
address-family ipv4 vrf ABC
  redistribute ospf 1 vrf ABC match internal external 1 external 2
aggregate-address 1.0.0.0 255.0.0.0 summary-only as-set
  no synchronization
 exit-address-family
```

R2#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	3.3.3.3/32	0	Fa0/1	10.1.23.3
201	Pop tag	10.1.34.0/24	0	Fa0/1	10.1.23.3
202	301	4.4.4.4/32	0	Fa0/1	10.1.23.3
203	**Aggregate**	**1.0.0.0/8[V]**	**0**		
204	Aggregate	10.1.12.0/24[V]	0		
205	Untagged	1.1.1.0/24[V]	0	Fa0/0	10.1.12.1

3. Aggregate 及 untagged 的区别及共同点

个人觉得：

- 与 pop 不同，pop 是有特定的标签值对应的，而 untagged 及 aggregate 是没有特定的标签的

- Untagged 动作，是弹出所有标签栈，并且根据 LFIB 表提示的出接口和下一跳做转发。而 Aggregate 动作则是弹出标签栈，同时再做进一步的 IP 查找。
- Untagged 及 aggregate 的诱因不同。前面已经分析过了两者的诱因。

11参考书籍

MPLS 技术架构

MPLS 和 VPN 体系结构

MPLS Fundamentals

OSPF 笔记

红茶三杯 CCIE 学习文档

文档版本： 3.0　　更新时间： 2013-01-26
文档作者： 红茶三杯　　文档地址： http://ccietea.com
文档备注： 请关注文档版本及更新时间

1 基础知识

1.1 知识点

1. **COST 的计算**

   接口 COST=100M /接口带宽，一条路由的 COST 由该路由从来源一路过来的所有入口方向的接口 cost 值的总和。这意味着更高带宽的传输介质在 OSPF 协议中将会计算出一个小于 1 的分数，这在 OSPF 协议中是不允许的（会被四舍五入为 1）。可使用 auto-cost reference-bandwidth xx 来更改参考值。注意如果要调整，就要在全 OSPF 域的所有 OSPF router 上都进行配置。

2. **组播地址**

   224.0.0.5 （所有的 OSPFrouter）来向所有其他 router 发送链路状态信息。

   224.0.0.6 用来表示所有的指定 router

3. **数据库类型**

   在同一个 OSPF 区域中的每台 Router 都相同的数据状态数据库，带有多个接口的 router 可以加入多个区域（必须至少有一个借口属于 area0）,这些 router 称为 Area Border Routers，它会给每个区域维护一个单独的拓扑数据库。

毗邻数据库 adjacency database	列出所有已经与 router 建立双向通信关系的 router
链路状态数据库 LSDB link-state database	列出关于网络中所有其他 router 的信息，该数据库显示出了网络的拓扑结构，一个区域中所有的 router 都有相同的该数据库
路由选择表	Routing table

Forward database	

1.2 RouterID

OSPF Router_ID 的选定是若有 loopback 口，选最大的 loopback 口地址，若无则选活动的物理接口中 IP 地址最大的作为 RouterID。

在路由器运行了 OSPF 并确定了 ROUTER-ID 之后，如果该 ID 对应的接口 DOWN 掉或者接口消失或出现一个更大的 IP，OSPF 仍然使用该 ROUTER-ID（也就是说，非抢占，稳定第一），即使此时 clear ip ospf process 重启 OSPF 进程，RouterID 也不会发生改变；除非重新手工配置 RouterID（OSPF 进程下手工敲 router-id xxx），并重启 OSPF 进程方可。

1.3 Timers

Hello Interval	CISCO 设备在广播型网络上默认 10s，非广播型网络默认 30s，修改 HELLO 时间，Dead 时间默认也会随之修改
Routerdead interval	多长时间没收到邻居的分组，认为其失效。默认为 hello interval 的 4 倍
LSA refresh timer	默认每隔 30 分钟发送链路状态条目的摘要，刷新之后 LSA 序列号加 1
LSAmaxAge	60 分钟，该时间内如果没有刷新，则从 LSDB 中删除
Wait timer	等待计时器 在开始选取 DR 和 BDR 之前,路由器等待邻居路由器的 HELLO 数据包通告 DR 和 BDR 的时长（收集所有候选人的信息），等待计时器的时间长度就是 RouterDeadIntervd 的时间
Retransmit	指在没有得到确认的情况下,路由器重传 OSPF 数据包将要等待的时间，缺省 5s
pollInterval	这个值只用于 NBMA 网络上相关的邻居路由器。因为在 NBMA 网络上,邻居路由器可能无法自动地 本地路由器发现，因此，如果邻居状态是失效 DOWN 的，那么路由器将每经过 pollInterval 的时间就会发送一个 Hello 数据包给它的邻居路由器。这里的 pollInterval 的时长比 HelloInterval 的时间要长一些。

1.4 DR BDR

1. 选举过程

当一台 OSPF 路由器有效(active)并去发现它的邻居路由器时，它将去检查有效的 DR 和 BDR 路由器。如果 DR 和 BDR 路由器存在的话，这台路由器将接受已经存在的 DR 和 BDR 路由器。如果 BDR 路由器不存

在，将执行一个选取过程，选出具有最高优先级的路由器作为 BDR 路由器。如果存在多台路由器具有相同的优先级，那么在数值上具有最高路由器 ID 的路由器将被选中。如果没有有效的 DR 路由器存在，那么 BDR 路由器将被选举为 DR 路由器，然后再执行一个选取过程选取 BDR 路由器。

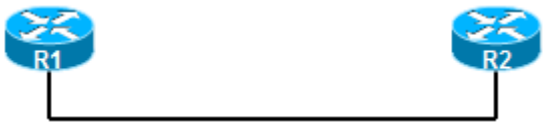
每台路由器的每一个多点访问接口都有一个路由器的优先级 0-255 默认 1。

2. 要点总结

- 如果在一个多路访问网络上只有惟一的一台具有选取资格的路由器相连，那么这台路由器将成为 DR 路由器，而且在这个网络上没有 BDR 路由器。其他所有的路由器都将只和这台 DR 路由器建立邻按关系
- 如果没有具有选取资格的路由器和一个多路访问网络相连 那么这个网络上将没有 DR 或者 BDR 路由器，而且也不建立任何邻接关系。
- DR、BDR 非抢占。当 DR 失效，BDR 成为 DR，同时再产生 BDR。
- DR 与 BDR、DR 与 Drother、BDR 与 Drother 之间都是 FULL 状态，Drother 之间是 2way 状态。所有的 Drother 都只和 DR 以及 BDR 建立全毗邻关系。

3. 验证实验

【实验 1】



- R1、R2 接口 ip ospf pri 0，则 R1、R2 不建立邻居关系
- R1 接口优先级为 0，R2 为 1，则 R2 成为 DR，并且该多路访问网络中没有 BDR
- 在上一步的基础上，R1 接口优先级改为 100，对网络没有影响（非抢占）
- 在上一步的基础上，R2（此时为 DR）重启 OSPF 进程，则 R1 成为 DR，R2 成为 BDR

【实验 2】

在上图中，如果 R1、R2 接口优先级相等，但 R2 的 routerid 大，一般情况下 R2 应该是 DR，但如果 R1 先配置，并且 R2 在 R1 配置后的 40s 之后才配置，那么这个时候即使 R2 的 routerid 大，R1 已经成为了 DR。原因在于 Wait timer 等待计时器，这个计时器的是在开始选取 DR 和 BDR 之前，路由器等待邻居路由器的 HELLO 数据包通告 DR 和 BDR 的时长（收集所有候选人的信息），等待计时器的时间长度就是 RouterDeadIntervd 的时间。因此只要 R2 在 40s 之后才配的，那么在这 40s 内，R1 就抢夺了 DR 的身份。

1.5 序列号

DBD 的序列号用于 master 和 slave 的同步，LSA 的序列号用来比较 LSA 的新或旧，这里讨论的是 LSA 的序列号，

是一个 4 个字节的数字。

1. **线性空间**

   从 0x80000001 到 0x7fffffff

2. **循环序列号空间**

   循环序列编号建立了一个不合逻辑的奇特的位。如果 x 是 1 到 4294 967 295 之间任意的一个数,那么 0<x<0。在运行正常的网络中通过声明两条规则可以维持这种条件,其中声明的规则用来确定什么时候一个序列号大于或小于另一个序列号。假设序列号空间为 n,有两个序列号 a 和 b,如果满足以下任意一种条件,则认为曰更新(数量更大)

   a>b 且 (a-b) <= n/2

   a<b 且 (b-a) > n/2

   例如假设使用一个 6 位序列号空间

   n=2 的 6 次方 = 64　　那么 n/2=32

   若有 3 和 48 两个序列号，则根据规则 2,可得 3 较新

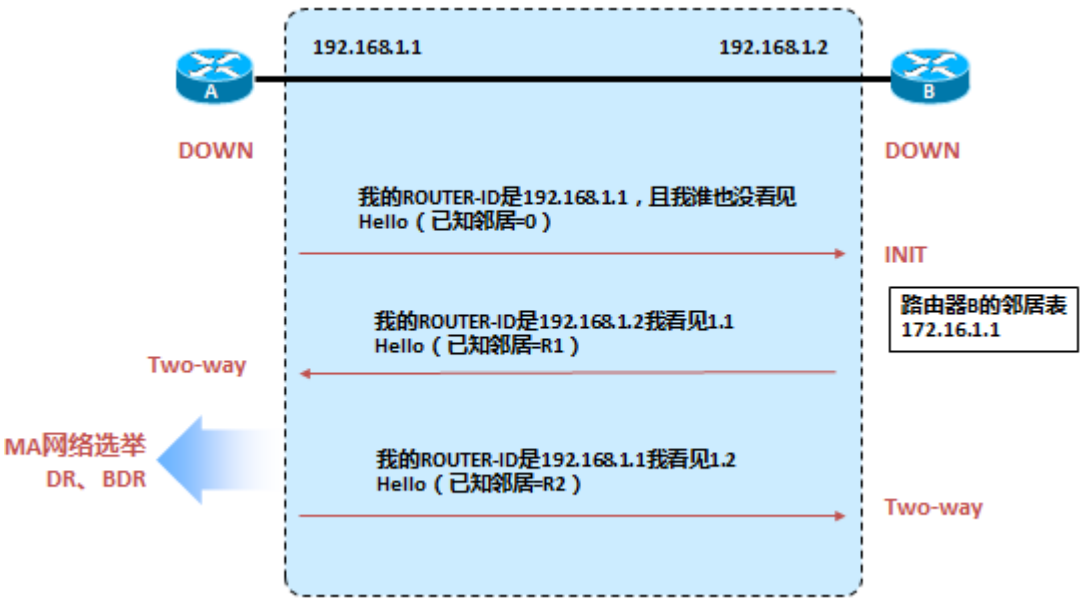
3. **棒棒糖**

   圆形空间的缺点是不存在一个数小于其他所有的数。

   线性空间的缺点是不能循环使用序列号，即序列号是有限的

2 邻居关系建立及维护

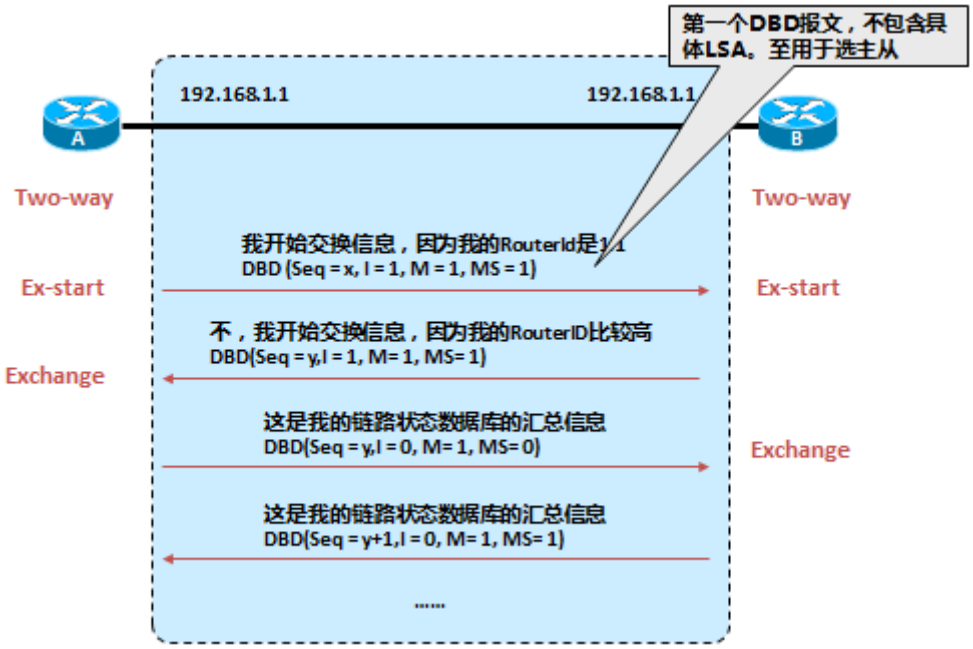
2.1 邻居建立过程



在初始情况下，A、B 在某个接口激活了 OSPF 后，都会开始在这个接口上去发组播的 HELLO 包，目的是发现 OSPF 邻居。HELLO 包里，有个 active neighbor 字段，用来存储路由器在某个 OSPF 接口上发现的邻居，当然，初始情况下，这个 HELLO 包里是不包含任何活跃的邻居的（也就没有 active neighbor 字段），因为他谁也没发现。

当 OSPF 路由器（B）在某个 OSPF 接口上收到邻居发来的 HELLO 包（里面没有装 active neighbor），它会记录下 A（在自己的 OSPF 接口数据结构中）并且将 A 的状态视为 **init**，然后将 A 的 RouterID 存储在自己将要发送的 HELLO 包的 active neighbor 字段里发送出去，这样 A 就会收到这个 hello 包，并且在这个 hello 包当中找到自己的 RouterID，那么 A 会认为，与 B 已经完成了双边关系的建立，因此 A 会将 B 的邻居状态置为 **two-way**。

与此同时，A 也会继续发送 HELLO 包，并且将 B 的 routerID 放置于 HELLO 包中，而 B 收到这个 hello 包并看见了自己的 RouterID 后，B 也会将 A 的状态置为 two-way，至此 OSPF 的第一个稳态就达到了。



接下去 A、B 会进入 **ex-start** 状态并开始进行 master、slave 的协商，协商 M/S 的目的是为了决定在后续的 LSA 交互中，谁来决定 DD 的序列号，而 RouterID 大的那个 OSPF 路由器的接口将会成为 master，由它来决定 DD seq，对端成为 slave。这里要注意 master 不是 DR，要注意与 DR 的概念进行区分。这个协商过程，是由交互 DBD 包实现的，注意这里使用的是空的 DBD 包，也就是不包含任何 LSA 头部的 DBD 包，这个包当中，有三个位非常关键：I、M、MS。用于 ex-start 阶段协商 master、slave 的 DBD 包，I 位（或叫做 init 位）都是置 0 的，另外 MS 位如果置 1，表示 DBD 报文始发路由器认为自己的 master，当然起初大家都这么认为，在一系列 DBD 交换后，就会得到选举结果，被选举为 slave 的 OSPF 接口，会将发送的 DBD 包 MS 位置为 0；另外 M 位表示 more，如果一个 OSPF 接口发送的 DBD 包 M 位置 1，在表示这不是最后一个 DBD，后续还有 DBD 包待发送。

当 OSPF 接口收到一个 DBD 包且其中 I 位置 0 的时候，它就知道与该邻居的 ex-start 阶段已经过去了，于是将邻居的状态置为 **ex-change**，并存储对端发来的 DBD 包所包含的 LSA 头部，当然，他自己也发送关于自己 OSPF DB 的摘要给邻居。如此一来，双方都能通过 DBD 的交互，了解到对方 OSPF DB 中的摘要情况。在这个过程中，可能交互数个 DBD 报文，并要注意，这些报文的 I 位都置 0，且 M 位一般也置 0，除非这是某个 OSPF

接口发送的最后一个 DBD 包。

当 Router A收到一个 M 位置 0 的 DBD 包的时候，它就知道，这是邻居发来的最后一个 DBD 包了，如果它搜集完这个邻居（假设是 B）发来的 DBD 并且发现，这些 DBD 里有它感兴趣的 LSA，它期望更详细的 LSA 信息时，它将 B 置为 **Loading** 状态，并且开始发送 LSR 报文去请求特定 LSA 的详细信息。B 收到这个 LSR 后，会以 LSU 进行回应，其中就包含了对方请求的 LSA 详细信息，因此，只有在 LSU 报文中，才能看到 LSA 的完整信息。收到 LSU 后，A 将 LSU 中所包含的 LSA 放进自己的 LSDB，并且给 B 发一个 Lsack 进行确认。当 OSPF 接口上所有的待请求的 LSA 全部收到更新后，它会将邻居置为 FULL。至此，OSPF 邻接关系的建立达到全毗邻。

在这里我们有个地方需要留意，我们通常说，OSPF 路由器 A 与 B 进入了 xx 状态，其实这句话并不严谨，原因之一，是因为 OSPF 是接口敏感型协议，许多的操作都是以 OSPF 接口作为立足点去考虑的，譬如邻居关系的建立，再如 DR 和 BDR，我们不能说一台路由器是 DR，准确的说，应该是某路由器的某个接口是 DR；再者，说两台路由器之间是 xx 状态，这个也不严谨，所谓的邻居状态，必须是以某台路由器为观察点，在其某个接口上观察到的某个邻居的状态，因此可能出现的一个情况是，在 A 上，看到的 B 的状态为 Loading，但是在 B 上，A 的状态已经是 FULL 了。

2.2 状态机

1. **Down**

   在 DOWN 状态下，OSPF 接口仍然有尝试发现邻居的意愿，因此会不断的发送组播 hello 包。

2. **Attempt**

   如果一个路由器，它邻居处于这种状态，则表示它从邻居没有收到任何信息，但是做了努力来与邻居联系。仅在 NBMA 网络上存在，当 NBMA 网络上具有 DR 选取资格的路由器和其邻居路由器相连的接口开始变为有

效(Activ/e)时,或者当这台路由器成为 DR 或 BDR 时,这台具有 DR 选取资格的路由器将会把邻居路由器的状态转换到 Attempt 状态。在 Attempt 状态下,路由器将使用 hellointerval 时间代替 pollinterval 的时间来作为向邻居发送 hello 数据包的时间间隔。

3. init

当 OSPF 接口收到链路上某个邻居发来的第一个 HELLO 包的时候，它会在接口上将该邻居置为 init 状态，注意这个 hello 包中可能并未包含任何的邻居信息。但是这至少证明，我这个 OSPF 接口在这个链路上，至少有个活的邻居。

下面是一个没有发现任何 active neighbor 的 hello 包：

```
Open Shortest Path First
⊟ OSPF Header
    OSPF Version: 2
    Message Type: Hello Packet (1)
    Packet Length: 44
    Source OSPF Router: 1.1.1.1 (1.1.1.1)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0xea9c [correct]
    Auth Type: Null
    Auth Data (none)
⊟ OSPF Hello Packet
    Network Mask: 255.255.255.0
    Hello Interval: 10 seconds
  ⊞ Options: 0x12 (L, E)
    Router Priority: 1
    Router Dead Interval: 40 seconds
    Designated Router: 0.0.0.0
    Backup Designated Router: 0.0.0.0
⊞ OSPF LLS Data Block
```

4. Two-way

当 OSPF 路由器在某个链路上发现了邻居后，它自己发送的 hello 包里就会增加 active neighbor 字段，用于存储在该链路上发现的 OSPF 邻居。当一台 OSPF router 看到自己（的 RouterID）出现在邻居发过来的的 hello 分组中，它就会将该邻居置为 Two-way。该状态是 OSPF 邻居之间可以具有的最基本的关系，也是第一个稳态，但是此时两者还不能共享路由信息。

下面是一个已经在链路上发现了邻居 1.1.1.1 的 hello：

```
Open Shortest Path First
⊟ OSPF Header
    OSPF Version: 2
    Message Type: Hello Packet (1)
    Packet Length: 48
    Source OSPF Router: 2.2.2.2 (2.2.2.2)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0xe694 [correct]
    Auth Type: Null
    Auth Data (none)
⊟ OSPF Hello Packet
    Network Mask: 255.255.255.0
    Hello Interval: 10 seconds
  ⊞ Options: 0x12 (L, E)
    Router Priority: 1
    Router Dead Interval: 40 seconds
    Designated Router: 0.0.0.0
    Backup Designated Router: 0.0.0.0
    Active Neighbor: 1.1.1.1
⊞ OSPF LLS Data Block
```

5. ExStart

一台 OSPF 路由器在将某个邻居置为 2way 状态后，就开始发送空的 DBD 包，用于协商 master/slave。这个就是 ex-start 状态。两台 router 间用空的 BDB 分组确定 master 和 slave 关系（注意不是 DR 和 BDR），

在 DBD 包中有 3 个标记位用来管理邻接关系的建立过程：

- I 位　　或称为初始位（initial bit） 用于 ex-start 协商主从关系的初始化协商的 DBD 包，该位置 1
- M 位　　或称为后继位（More bit） 如果这不是 OSPFrouter 发送的最后一个 DBD，该位置 1
- MS 位　或称为主/从位（Master/slave bit） 如果始发路由器是 Master，则该位置 1

如果某台 OSPF router 收到邻居发来的 DBD，I 位也就是 init 位置 0，则意味着 ex-start 状态结束，并且 MS/slave 已经选出来了，那么该路由器会将邻居置为 exchange 状态，开始用包含 LSA 头部的 DBD 交换各自的 LSBD。

下面是一个用于初始化协商的 DBD 消息：

```
Open Shortest Path First
⊞ OSPF Header
⊟ OSPF DB Description
    Interface MTU: 1500
  ⊞ Options: 0x52 (O, L, E)
  ⊟ DB Description: 0x07 (I, M, MS)
      .... 0... = R: OOBResync bit is NOT set
      .... .1.. = I: Init bit is SET
      .... ..1. = M: More bit is SET
      .... ...1 = MS: Master/Slave bit is SET
    DD Sequence: 7994
⊞ OSPF LLS Data Block
```

6. Exchange 状态

这个过程，双方使用包含自己 LSA 头部的 DBD 报文进行交互，并且将对方发过来的 LSA 头部、并且自己感兴趣的 LSA（或自己没有的 LSA）存储在一个本地 OSPF 接口的队列里，以便在下一个阶段进行 LSA 详细

信息的请求。当某个 OSPF 接口收到邻居发来的 DBD，M 位置 0，则表示对方已经发完 DBD 了，与此同时，如果该路由器的这个 OSPF 接口上存在待请求的 LSA，那么它会将这个邻居置为 loading 状态。

下面是一个装载了 LSA 头部的 DBD 消息：

```
Open Shortest Path First
⊞ OSPF Header
⊟ OSPF DB Description
    Interface MTU: 1500
  ⊞ Options: 0x52 (O, L, E)
  ⊟ DB Description: 0x02 (M)
      .... 0... = R: OOBResync bit is NOT set
      .... .0.. = I: Init bit is NOT set
      .... ..1. = M: More bit is SET
      .... ...0 = MS: Master/Slave bit is NOT set
    DD Sequence: 7994
⊟ LSA Header                                    LSA头部
    LS Age: 19 seconds
    Do Not Age: False
  ⊞ Options: 0x22 (DC, E)
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 1.1.1.1
    Advertising Router: 1.1.1.1 (1.1.1.1)
    LS Sequence Number: 0x80000002
    LS Checksum: 0x3e21
    Length: 48
⊞ OSPF LLS Data Block
```

7. loading 状态

OSPF router 使用 LSR 去请求 LSA 的详细信息，对方使用 LSU 发来更新，因此只有 LSU 里才有 LSA 的完整信息。在收到 LSU 后，一方面本地使用 LSAack 进行确认，另一方面将 LSU 中包含的 LSA 装载进自己的 LSDB。

以下是一个 LSR 消息，非常的简单：

```
Open Shortest Path First
⊞ OSPF Header
⊟ Link State Request
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 1.1.1.1
    Advertising Router: 1.1.1.1 (1.1.1.1)
```

接着是一个 LSU 消息，里头包含了 LSA 的完整信息，LSA 这里暂时不做详细介绍，请看下文：

```
Open Shortest Path First
⊞ OSPF Header
⊟ LS Update Packet
    Number of LSAs: 1
  ⊞ LS Type: Router-LSA
```

8. Full Adjacency 状态

Loading 状态结束后，也就是本地 OSPF 接口上再没有待更新的 LSA 队列后，将邻居置为 FULL。

2.3 邻居数据结构

Router#sh ip ospf neighbor 192.168.1.2

Neighbor 192.168.1.2, interface address 9.9.12.2

In the area 0 via interface Serial2/0

Neighbor priority is 1 (configured 100), State is FULL, 18 state changes

DR is 9.9.12.1 BDR is 9.9.12.2

Poll interval 120

Options is 0x52

LLS Options is 0x1 (LR)

Dead timer due in 00:01:53

Neighbor is up for 00:00:19

Index 1/1, retransmission queue length 0, number of retransmission 2

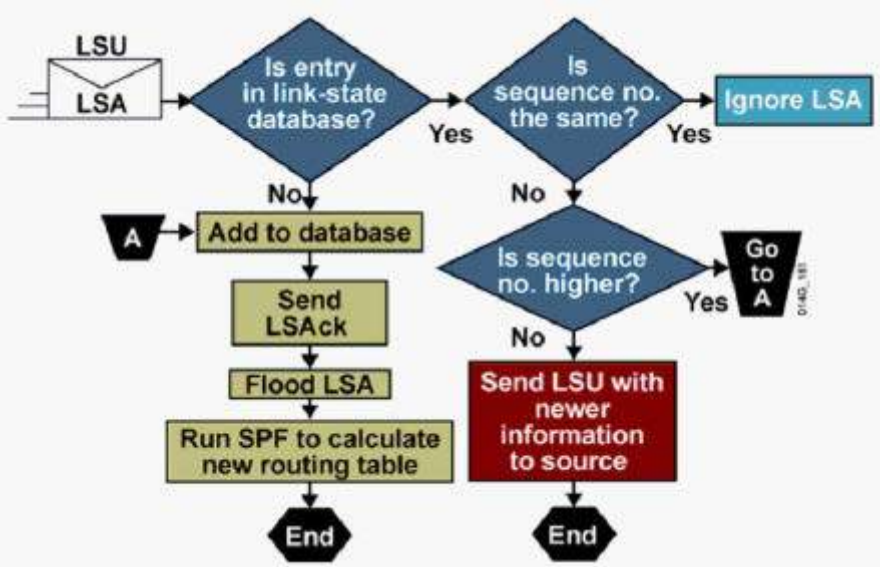
First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)

显示的是 NBMA 环境下，OSPF 邻居的数据结构，其中邻居的实际接口优先级是 1，

本地使用 neighbor 9.9.12.2 priority 100

本命令只对 DR、BDR 选举过程有效

2.4 LSA 的更新



LSA 可以通过下面两种方法之一来确认：

显式确认(Explcit Acknowledgment) 确认收到包含这个 LSA 头部的链路状态确认数据包。

隐式确认(Implcit Acknowledgment) 确认收到包含这个 LsA 的相同实例(没有其他更加新的 LSA)的更新数据包。

主路由器将控制数据库的同步过程，并确保每次只有一个数据库描述数据包是未处理的。当从路由器收到一个从主路由器发出的 DBD 后，从路由器将通过发送一个具有相同序列号的 DBD 报文来确认(这个过程一直延续，直到主路由器发出 m 位置 0 的报文)。如果 rxmtInterval 到期后，从路由器还没确认，主路由器会重传。

LSDB 里每个 LSA 都有一个序列号，用来标示 LSA 的新旧，从 0x80000001 到 0x7fffffff，当 LSA 触发更新的时候，其泛洪 LSA 的时候会增加序列号，且随着 30min 一次的泛洪，序列号也会加 1（由产生该 LSA 的 router）. 每个 LSA 条目都有一个老化时间 aging timer ，存储在 age 字段中，默认的 30 分钟。LSA 失效后，最初发送该条目的路由器会发送一个 LSU，其中包含序列号更高的 LSA。

2.5 OSPF 邻居关系建立中的问题

1. **MTU 问题**

   ip os mtu-ignore

   在接口模式下配置可以忽略 MTU ,注意是在 MTU 小的那一个 OSPF 路由器接口上(如果两端 MTU 不一样)

2. **接口掩码问题**

   a) **在一个 MA 网络中，如果邻居两端接口掩码不一致，则邻居建立不起来。**

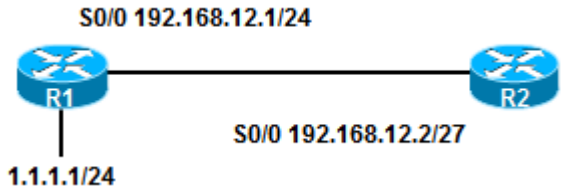
   ```
   *Mar  1 09:20:00.173: OSPF: Mismatched hello parameters from 192.168.12.1
   *Mar  1 09:20:00.173: OSPF: Dead R 40 C 40, Hello R 10 C 10  Mask R 255.255.255.0 C 255.255.255.224
   ```

   原因在于 2 类 LSA 中，是有掩码字段的

   ```
   Routing Bit Set on this LSA
   LS age: 251
   Options: (No TOS-capability, DC)
   LS Type: Network Links
   Link State ID: 192.168.23.2 (address of Designated Router)
   Advertising Router: 192.168.12.2
   LS Seq Number: 80000001
   Checksum: 0x144
   Length: 32
   Network Mask: /24          // 如果掩码不一致，DR 就无法用 2 类 LSA 来描述你这台路由器
   Attached Router: 192.168.12.2
   Attached Router: 192.168.23.3
   ```

b) 在 p2p 的网络中，如果邻居两端接口掩码不一致，邻居能建立，并且路由也能学习到

玩玩而已，别真特么去搞两个掩码不一样。



则 R1 的路由表：

192.168.12.0/24 is variably subnetted, 2 subnets, 2 masks

O 192.168.12.0/27 [110/128] via 192.168.12.2, 00:01:34, Serial0/0

C 192.168.12.0/24 is directly connected, Serial0/0

1.0.0.0/24 is subnetted, 1 subnets

C 1.1.1.0 is directly connected, Loopback0

则 R2 的路由表：

192.168.12.0/24 is variably subnetted, 2 subnets, 2 masks

C 192.168.12.0/27 is directly connected, Serial0/0

O 192.168.12.0/24 [110/128] via 192.168.12.1, 00:03:14, Serial0/0

1.0.0.0/32 is subnetted, 1 subnets

O 1.1.1.1 [110/65] via 192.168.12.1, 00:03:14, Serial0/0

3. 网络类型问题

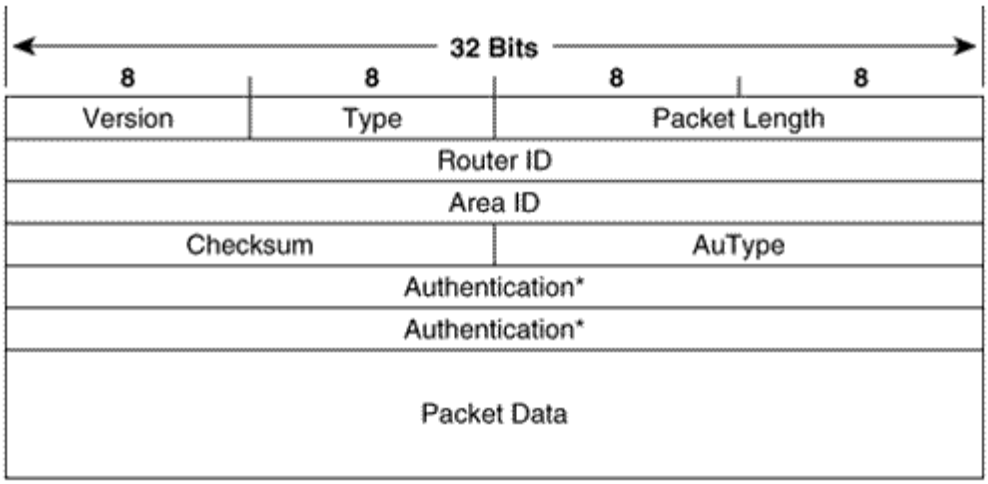
两端如果 OSPF 网络类型不同 ,那么邻居关系能建立起来 ,路由器甚至能收到对方的 LSA ,但是路由学不到。

3 OSPF 数据包

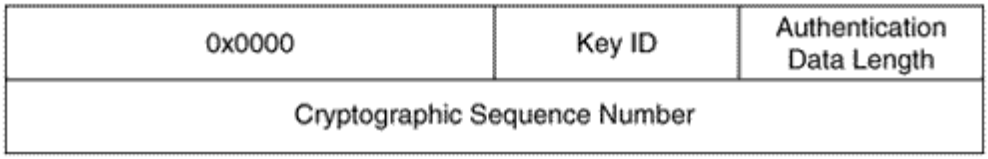
3.1 数据包类型

Hello	与邻居建立毗邻关系。默认每 10s 交换一次 HELLO 分组
DBD 数据库描述分组	描述一个 OSPF router 连路状态数据库的内容
LSR 链路状态请求	请求邻居 router 发送其链路状态数据库中的特定项
LSU 链路状态更新	向邻居 router 发送链路状态通告
LSACK 链路状态确认	由于 OSPF 使用 IP 分组，并不使用 TCP 或 UDP，而其又对可靠性由要求，因此增加该包来保证可靠性。

3.2 数据包头部



*If AuType = 2, the Authentication field is:

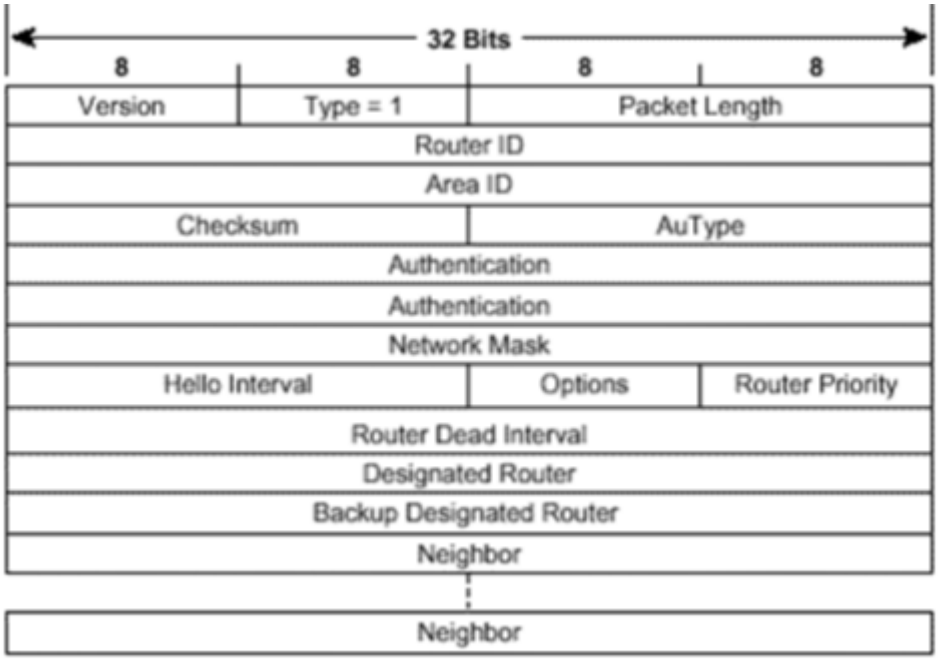


AreaID 是一个点分十进制表示的数在这里，如 area256，则报文中存放的是 0.0.1.0 （转换成二进制看）

Autype=0 不认证 =1 明文 =2 密文

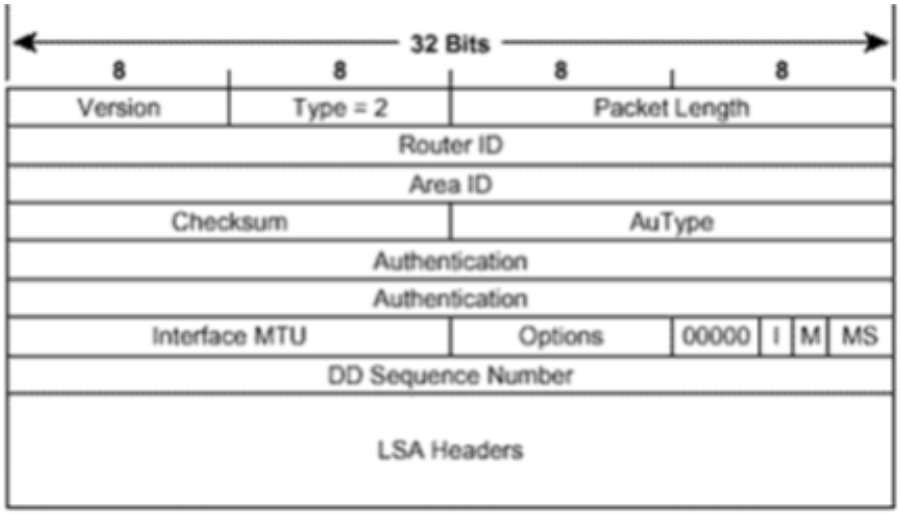
3.3 数据包格式

1. HELLO 包格式



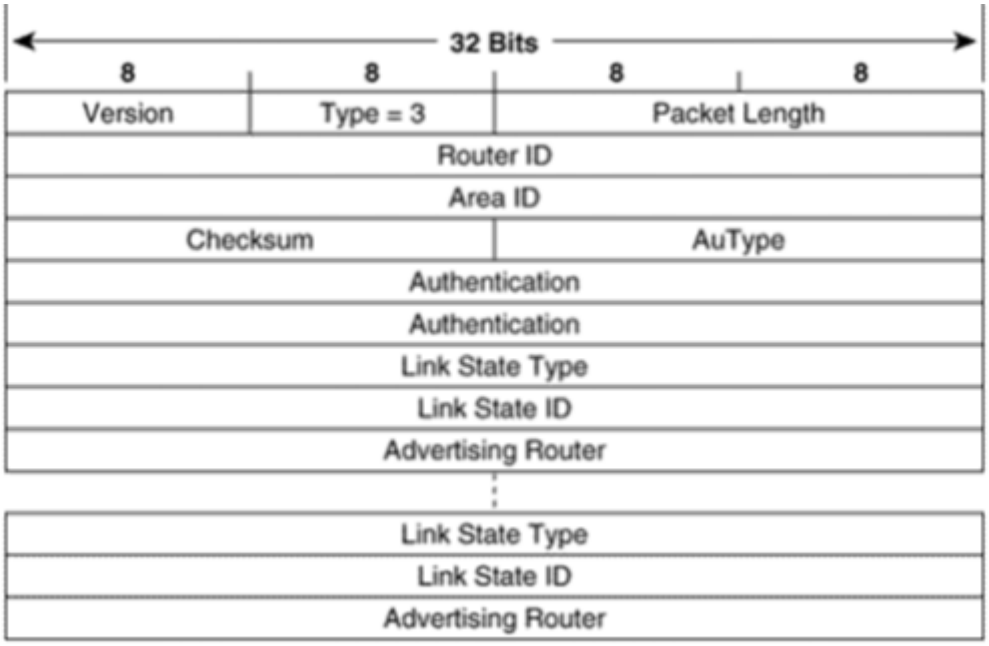
```
⊟ OSPF Hello Packet
    Network Mask: 255.255.255.0
    Hello Interval: 10 seconds
  ⊟ Options: 0x12 (L, E)
      0... .... = DN: DN-bit is NOT set
      .0.. .... = O: O-bit is NOT set
      ..0. .... = DC: Demand circuits are NOT supported
      ...1 .... = L: The packet contains LLS data block
      .... 0... = NP: Nssa is NOT supported
      .... .0.. = MC: NOT multicast capable
      .... ..1. = E: ExternalRoutingCapability
    Router Priority: 1
    Router Dead Interval: 40 seconds
    Designated Router: 0.0.0.0
    Backup Designated Router: 0.0.0.0
```

2. DBD 报文格式



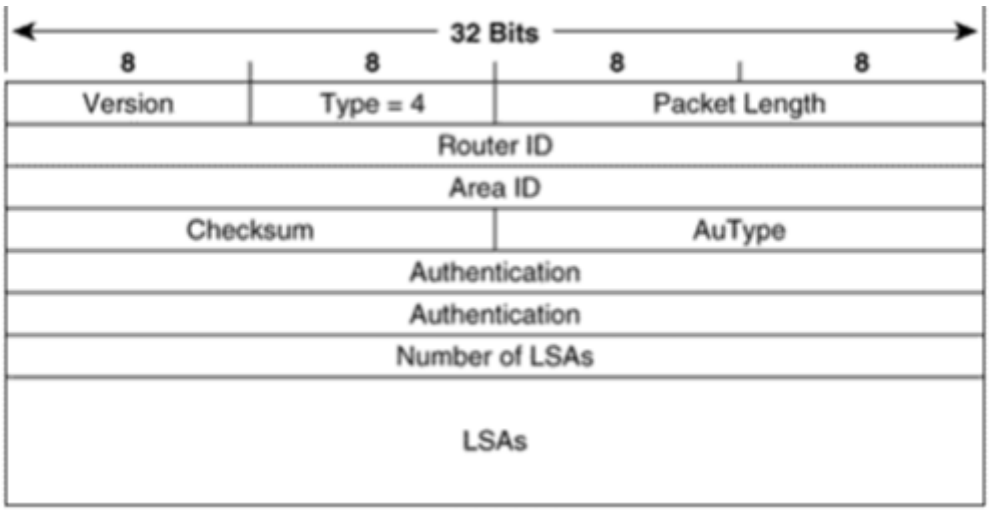

```
OSPF DB Description
  Interface MTU: 1500
⊟ Options: 0x52 (O, L, E)
    0... .... = DN: DN-bit is NOT set
    .1.. .... = O: O-bit is SET
    ..0. .... = DC: Demand circuits are NOT supported
    ...1 .... = L: The packet contains LLS data block
    .... 0... = NP: Nssa is NOT supported
    .... .0.. = MC: NOT multicast capable
    .... ..1. = E: ExternalRoutingCapability
⊟ DB Description: 0x07 (I, M, MS)
    .... 0... = R: OOBResync bit is NOT set
    .... .1.. = I: Init bit is SET
    .... ..1. = M: More bit is SET
    .... ...1 = MS: Master/Slave bit is SET
  DD Sequence: 6444
```

3. LSR 报文格式




```
Link State Request
  Link-State Advertisement Type: Router-LSA (1)
  Link State ID: 9.9.12.2
  Advertising Router: 9.9.12.2 (9.9.12.2)
```

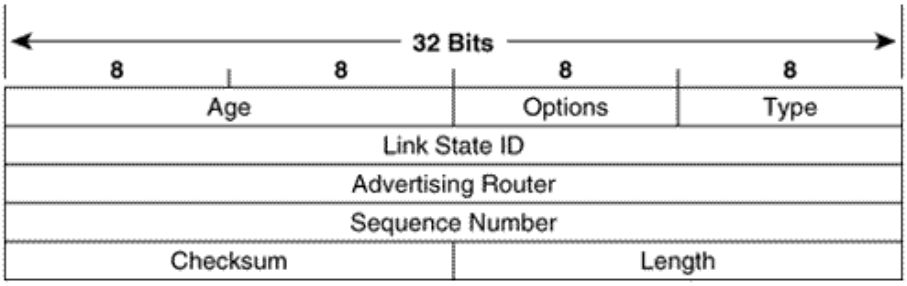
4. LSU 报文格式



```
LS Update Packet
  Number of LSAs: 1
⊟ LS Type: Router-LSA
    LS Age: 1 seconds
    Do Not Age: False
  ⊞ Options: 0x22 (DC, E)
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 9.9.12.2
    Advertising Router: 9.9.12.2 (9.9.12.2)
    LS Sequence Number: 0x80000003
    LS Checksum: 0xd54a
    Length: 48
  ⊞ Flags: 0x00 ()
    Number of Links: 2
  ⊞ Type: PTP      ID: 1.1.1.1        Data: 9.9.12.2       Metric: 64
  ⊞ Type: Stub     ID: 9.9.12.0       Data: 255.255.255.0  Metric: 64
```

3.4 LSA 格式

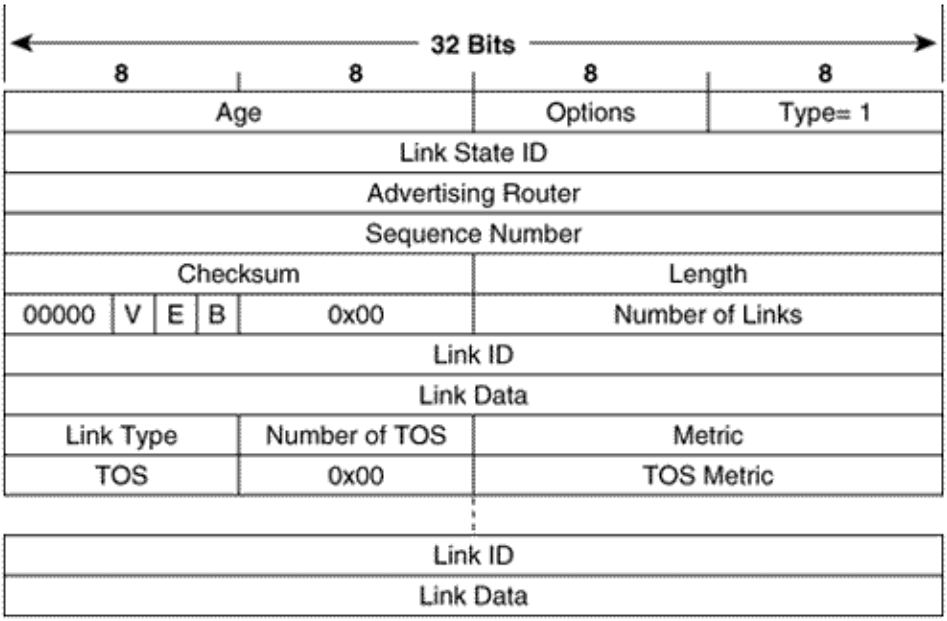
LSA 头部格式如下：



LSA 头部后面跟的，就是具体的 LSA 了，具体如下：

1. Router LSA

这个最基本的 LSA 通告列出了路由器所有的链路或接口，并指明了它们的状态和沿每条链路方向出站的 cost。



LinkstateID 始发路由器的 RouterID

V 位 置为 l 时说明始发路由器是一条或多条具有完全邻接关系的虚链路的一个端点 ,这里被描述的区域是传送区域。

E 位 当始发路由器是一个 ASBR 路由器时,设置该位为 1

B 位 当始发路由器是一个 ABR 路由器时,设置该位为 1

从链路 ID 到链路数据字段，这段即为具体的 LSA 内容

LINKID、Link Date 的取值根据 OSPF link 类型不同而不同：

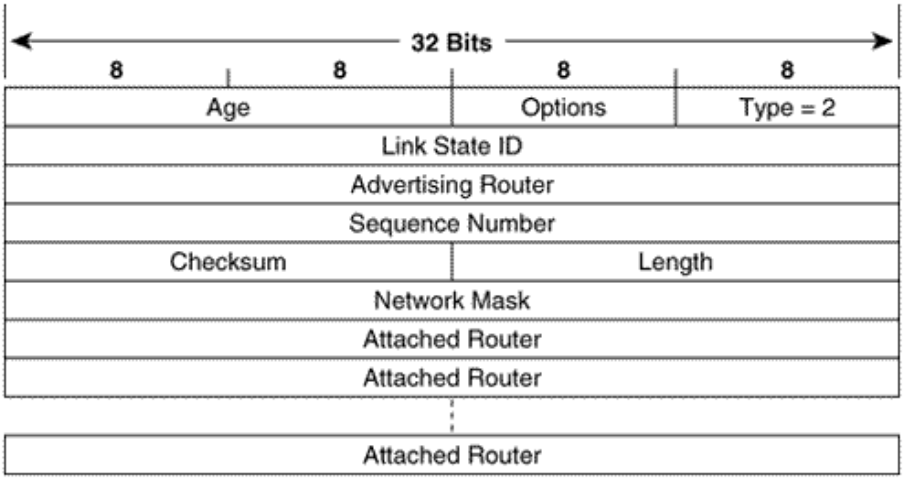
链路类型	连接	链路 ID 的值	链路数据
1	点到点连接到另一台路由器	邻居路由器的路由器 ID	和网络相连的始发路由器接口的 IP 地址
2	连按到一个传送网络（常见以太网）	DR 路由器的接口的 IP 地址	和网络相连的始发路由器接口的 IP 地址
3	连按到-个末梢网络	IP 网络或子网地址	网络的 IP 地址或子网掩码
4	虚链路	邻居路由器的路由器 ID	始发路由器接口的 MIB-II ifIndex 值

TOS 号 RFC2328 不在支持。如果没有 Tos 度量和一条链路相关联,那么这个字段就设置为 0x00

METRIC 度量值

TOS CISCO 只支持为 0

2. Network LSA



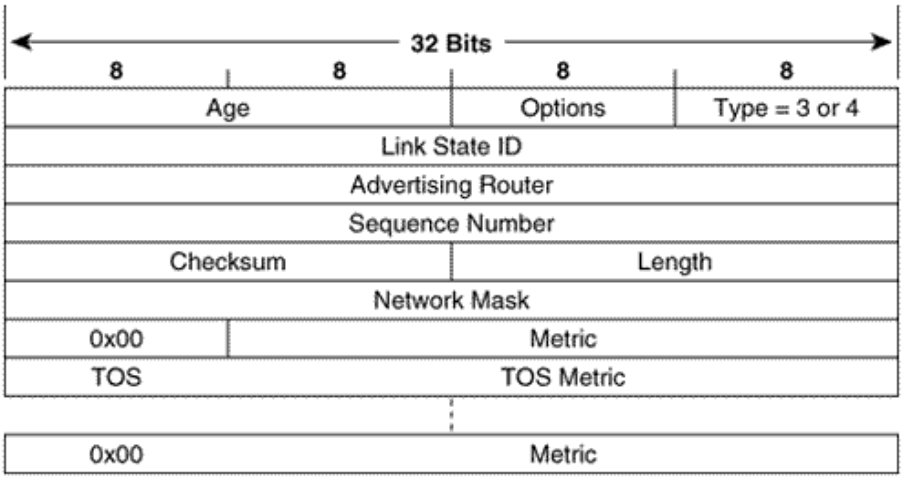
链路状态 ID（LINK stat ID） DR 路由器接口上的 IP 地址

相连的路由器 相连的路由器的 routerID（包括 DR 他自己）

注意：2 类 LSA 是不带 metric 的

3. Network summary LSA 和 ASBR summary LSA

两者有同样的格式，惟一的不同之处是它们所指的类型和链路状态 ID。

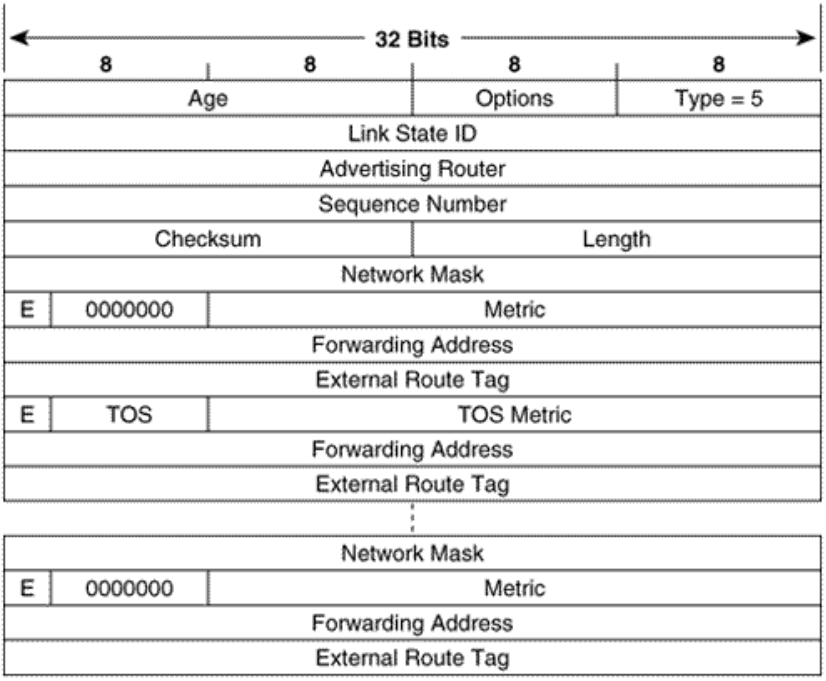


链路状态 ID(Link state ID) 对于类型 3 的 LSA 来说,它是所通告的网络或子网的 IP 地址。

对于类型 4 的 LSA 来说,链路状态 ID 是所通告的 ASBR 路由器的路由器 ID

3 类 LSA 是由 ABR 产生的，ABE 指的是，必须有一个接口属于区域 0，并且有另外一个接口属于其他区域

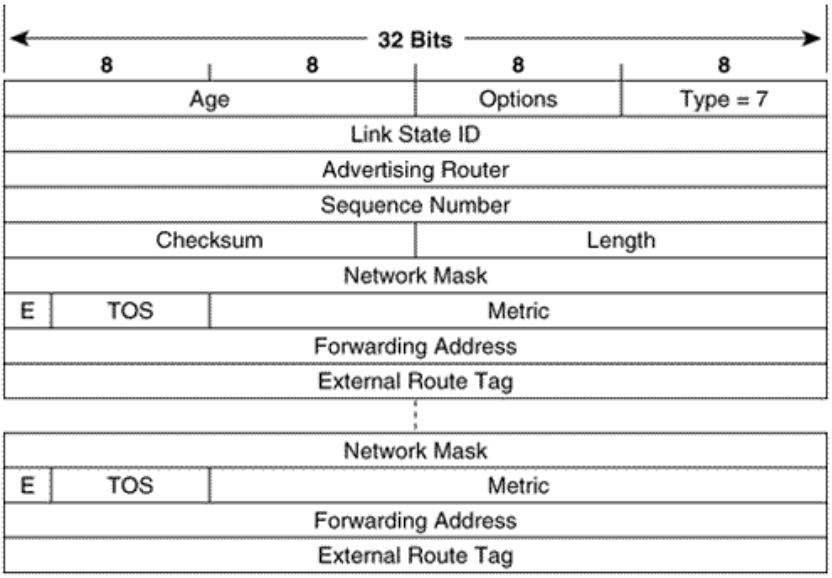
4. 自治系统外 LSA



链路状态 ID AS 外部 LSA 的链路状态 ID 是指日的地的 IP 地址

E 或称外部度量位 设置为 1，则为 E2 类型；为 0 则 E1 类型

5. NSSA 外部 LSA



3.5 OPTION 字段

Option 字段，出现在每一个 HELLO、DBD 以及每一个 LSA 中。

•	O	DC	EA	N/P	MC	E	T

```
Options: 0x52 (O, L, E)
  0... .... = DN: DN-bit is NOT set
  .1.. .... = O: O-bit is SET
  ..0. .... = DC: Demand circuits are NOT supported
  ...1 .... = L: The packet contains LLS data block
  .... 0... = NP: Nssa is NOT supported
  .... .0.. = MC: NOT multicast capable
  .... ..1. = E: ExternalRoutingCapability
```

1. N/P 位（RFC1587）

其中 N/P 位为 1 位，N 位及 P 位分别只出现在 HELLO 及 LSA 报文中：

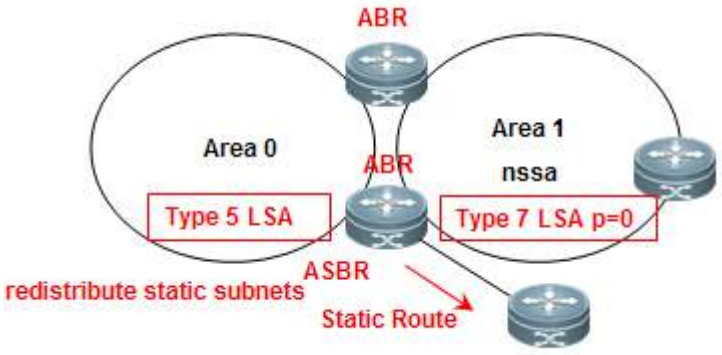
- **在 Hello 报文中：**N bit 指示该路由器为 NSSA 区域路由器。当 N bit 被置 1 时 E bit 就必须被清零。
- **在 LSA 报文中：**P bit 仅在 7 类 LSA 中出现

1) 置 1 时指示 NSSA 区域的 ABR 能够将这条 7 类 LSA 转成 5 类 LSA。

```
r1#sh ip os da nssa-external
OSPF Router with ID (1.1.1.1) (Process ID 1)
Type-7 AS External Link States (Area 1)
LS age: 9
Options: (No TOS-capability, No Type 7/5 translation, DC)    //即 P 位被置 0
```

2) 为 0 时，ABR 将不能将该 7 类 LSA 转换成 5 类 LSA

只有 NSSA 区域的 ABR 重发布路由时，通告的 7 类 LSA 中的 P bit 才为 0



NSSA 区域 ABR 在重发布时，将 7 类 LSA 的 P 位置 0，通知其他 NSSA 区域 ABR 不要对该 LSA 进行 7 转 5，因为骨干区域内的其他路由器已经从骨干区域内收到该路由的 5 类 LSA，因此 NSSA 区域其他 ABR 执行 7 转 5 是没有意义的。

2. E 位

当始发路由器具有接受 AS 外部 LSA 的能力时,该位将被设置。

在所有的 AS 外部 LSA 和所有始发于骨干区域以及非末梢区域的 LSA 中，该位将设置为 1。而在所有始发于末梢区域的 LSA 当中，该位设置为 0。另外，可以在 Hello 数据包中使用该位来表明一个接口具有接收和发送类型 5 的 LSA 的能力。

E 位配置错误的邻居路由器将不能形成邻接关系，这个限制可以确保一个区域的所有路由器都同样地具有支持末梢区域的能力。

3. **MC 位**

该位描述是否组播扩展 OSPF

4 OSPF 网络类型及链路类型

4.1 网络类型 network-type

指的**是 OSPF 协议在接口上针对不同的二层数据链路层介质或封装而定义的**，例如如果接口二层封装协议是以太，那么 OSPF 在这个接口的网络类型为 broadcast，如果接口的二层封装是 HDLC 或者 PPP，那么 OSPF 的网络类型是 P2P。OSPF 在不同的接口网络类型下，操作方式是不尽相同的。

使用 show ip ospf interface x 可以查看到接口的网络类型，如下：

```
R4#sh ip os int s 0/0
Serial0/0 is up, line protocol is up
  Internet Address 10.1.24.4/24, Area 1
  Process ID 1, Router ID 4.4.4.4, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:03
```

OSPF 定义了如下几种网络类型：

- 点到点 P2P
- 广播 Broadcast
- 非广播 Non-Broadcast

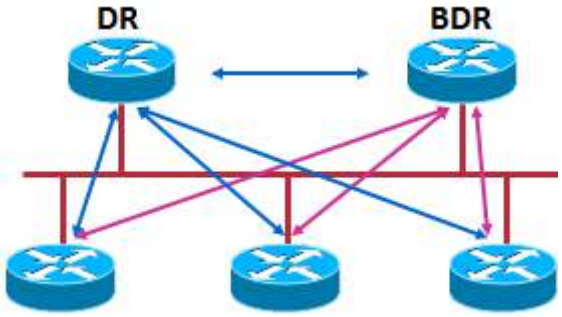
  非广播又包括了 5 种运行模式：
  - NBMA (RFC)
  - P2MP (RFC)

- P2MP nonbroadcast(CISCO)
- Broadcast(CISCO)
- P2P(CISCO)

1. 点到点类型

- 如果二层的协议为 PPP、HDLC 等，则 OSPF 网络类型为 P2P
- 如果帧中继子接口类型为 P2P 的，则 OSPF 网络类型也为 P2P
- 不选举 DR、BDR
- 使用组播地址 224.0.0.5
- OSPF 能够根据二层封装自动检测到 P2P 网络类型

2. 广播多路访问型



- 通常出现在以太网
- 选举 DR、BDR
- 所有路由器均与 DR 及 BDR 建立邻接关系
- 使用组播地址 224.0.0.5 及 224.0.0.6

3. 非广播

详见红茶三杯 OSPF 在 NBMA 环境下的操作

4.2 链路类型 link-type

OSPF 除了定义网络类型，还定义了链路类型，注意链路类型与网络类型是两个概念，不要混淆。链路类型主要用于描述 OSPF 路由器的接口或邻居。在 1 类 LSA 中，可以看到始发该 LSA 的路由器所连接的所有链路（Link）、链路的类型以及相关的内容。

LS age: 1355

Options: (No TOS-capability, DC)

LS Type: Router Links

Link State ID: 1.1.1.1

Advertising Router: 1.1.1.1

LS Seq Number: 8000001F

Checksum: 0xFF44

Length: 48

Area Border Router

AS Boundary Router

Number of Links: 2

Link connected to: **another Router (point-to-point)**

(Link ID) Neighboring Router ID: 2.2.2.2

(Link Data) Router Interface address: 10.1.12.1

Number of TOS metrics: 0

TOS 0 Metrics: 64

Link connected to: **a Stub Network**

(Link ID) Network/subnet number: 10.1.12.0

(Link Data) Network Mask: 255.255.255.0

Number of TOS metrics: 0

TOS 0 Metrics: 64

1 类 LSA 中，用于描述 Link 的 LINKID、Link Date 的取值根据 OSPF link 类型不同而不同：

链路类型	连接	链路 ID 的值	链路数据
1	点到点连接到另一台路由器	邻居路由器的路由器 ID	和网络相连的始发路由器接口的 IP 地址
2	连按到一个传送网络（常见以太网）	DR 路由器的接口的 IP 地址	和网络相连的始发路由器接口的 IP 地址
3	连按到-个末梢网络	IP 网络或子网地址	网络的 IP 地址或子网掩码
4	虚链路	邻居路由器的路由器 ID	始发路由器接口的 MIB-II ifIndex 值

OSPF 链路类型分为以下几种：

1. **Stub Network Link**

在一个网段中只有一台 OSPF 路由器的情况下，该网段被 OSPF 链路类型定义为 Stub Network Link；因为一个网段中只有一台 OSPF 路由器，所以在这个网段就不可能有 OSPF 邻居，一个接口被通告进 OSPF，无论其二层链路是什么介质，只要在该接口上没有 OSPF 邻居，那么就是 Stub Network Link；Loopback 接口永远被定义为 Stub Network Link，默认使用 32 位掩码表示，无论将 Loopback 接口改为哪种 OSPF 网络类型（Network Type），始终改变不了它的 OSPF 链路类型（Link Type）属性，但可以改变它在 LSA 中的掩码长度。

2. **Point-To-Point Link**

OSPF 网络类型（Network Type）为 Point-To-Point 的接口，OSPF 链路类型（Link Type）为 Point-To-Point Link，但 Loopback 接口除外；而网络类型为点到多点（Point-To-Multipoint）的接口，同样链路类型也为 Point-To-Point Link。

Point-To-Point Link 可以是手工配置的地址（Numbered），也可以是借用的地址（Unnumbered），也可以是物理接口或逻辑子接口。

3. **Transit Link**

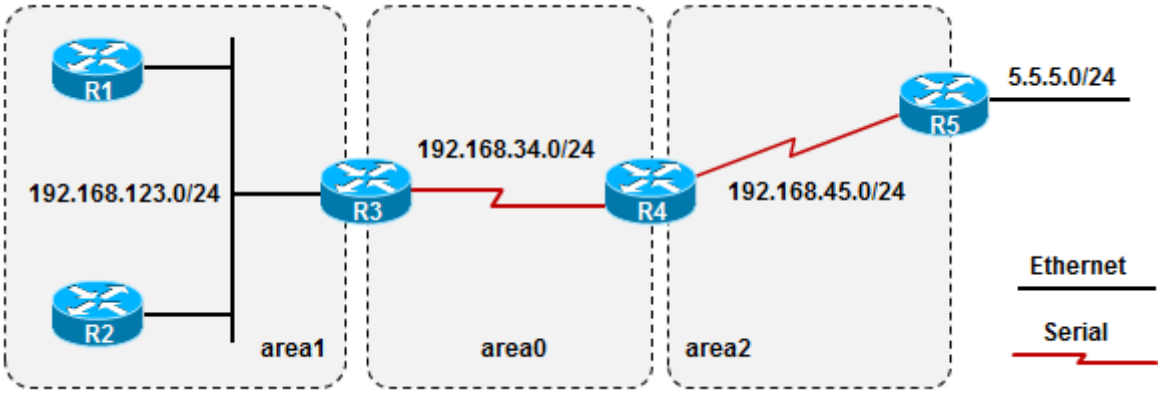
拥有两台或两台以上 OSPF 路由器的链路，简单理解为有邻居的 OSPF 接口就是 Transit Link，但网络类型为 Point-To-Point 和点到多点（Point-To-Multipoint）的接口除外，因为它们被定义为 Point-To-Point Link。

4. **Virtual link**

就是 OSPF 虚链路(Virtual Link),但稀奇的是 ,虚链路(Virtual Link)被定义为手工配置的地址(Numbered)的 Point-To-Point Link。

5 LSA 详解

5.1 各类 LSA 解析



我们本次 LSA 详解采用的拓扑结构相对比较简单，如上图所示，R1、R2、R3 之间连接到一个广播多路访问的介质上（以太网交换机），网段为 192.168.123.0/24，R1、R2、R3 的接口 IP 分别是.1、.2、.3；R3、R4 以及 R4、R5 之间通过串行链路直连；所有路由器上均开启 LOOPBACK 接口，R1 的 LOOPBACK 接口 IP 为 1.1.1.1，R2 为 2.2.2.2，其他设备依次类推；所有的上述 LOOPBACK 接口仅作为路由器 Router-ID 的选取使用，并不做 network 宣告；R5 上将 LOOPBACK 接口重发布进 OSPF；

设备互联地址如上图所示。我们开始来逐一认识每种 LSA：

1. **Type 1 LSA：Router LSA**

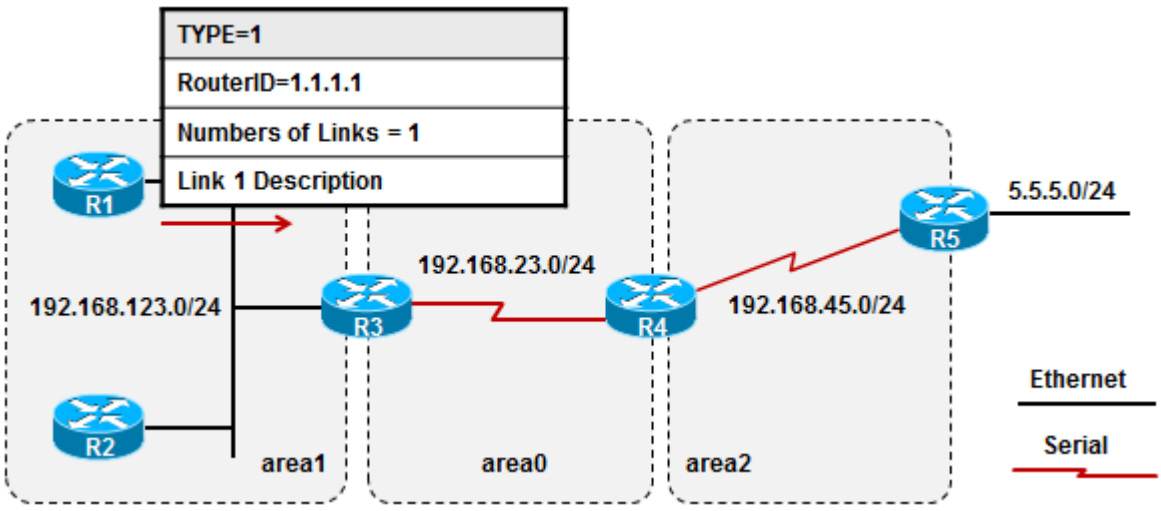
   每一个运行 OSPF 的路由器均会产生 1 类 LSA，1 类 LSA 怎么理解？其实很简单，就是每台路由器描述一下自己“家门口的状况”，并且只会告诉给“全村的人”（本区域内泛洪）。

   主要的功能有以下两点：

   - **表示路由器的特殊角色，如 Virtual-link、ABR、ASBR**

     这是通过 1 类 LSA 中相关的 V、B、E 位置为来实现的

   - **描述本路由器在某个区域内部与的直连链路（接口）及接口 COST 值**



在这个环境中，每台路由器均会产生 1 类 LSA，并且这些 1 类 LSA 只会在本区域内泛洪。譬如 area1 内，泛洪的 1 类 LSA 有哪些呢？

R1#sh ip ospf database

```
            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Router Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum Link count
1.1.1.1         1.1.1.1         59          0x80000002 0x00213B 1
2.2.2.2         2.2.2.2         60          0x80000002 0x00E270 1
3.3.3.3         3.3.3.3         60          0x80000003 0x00A5A2 1
```

可以看到，R1、R2、R3 均在 area1 内产生了各自的 1 类 LSA，每个 1 类 LSA 中均包含 1 个 LINK，也即三台路由器各有一个直连接口连接到这个区域内。

接着来看 1 类 LSA 详细内容，譬如 R1，R1 有一个直连接口，连接到一个 MA 网络 192.168.123.0，因此 R1 产生的 1 类 LSA（在 area1 内泛洪）中，包含一条链路，

详细输出如下：

R1#sh ip ospf database router 1.1.1.1

```
  LS age: 244
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 1.1.1.1                  // 产生该 LSA 的路由器的 Router-ID
  Advertising Router: 1.1.1.1
  LS Seq Number: 80000002
  Checksum: 0x1F3C
  Length: 36
  Number of Links: 1                      // R1 连入本区域的接口（链路）数量
```

```
    Link connected to: a Transit Network          // R1 通过以太网连接到一个 MA 网络 123.0
     (Link ID) Designated Router address: 192.168.123.3      // DR 的接口 IP
     (Link Data) Router Interface address: 192.168.123.1      //连接到这个 MA 的本地接口 IP
      Number of TOS metrics: 0
       TOS 0 Metrics: 1                            //  接口 COST=1
```

这就是 1 类 LSA，“Link connected to”开始就是该路由器的所有直连链路的相关信息。

R2 及 R3 在 Area1 中的 1 类 LSA 大体上类似，那么 R3 在 area0 中泛洪的 1 类 LSA 是如何？

R1#sh ip ospf database router 3.3.3.3

```
            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Router Link States (Area 1)

 Routing Bit Set on this LSA
 LS age: 584
 Options: (No TOS-capability, DC)
 LS Type: Router Links
 Link State ID: 3.3.3.3
 Advertising Router: 3.3.3.3
 LS Seq Number: 80000003
 Checksum: 0xA5A2
 Length: 36
 Area Border Router          //标识这台路由器为 ABR，这其实是 LSA1 报文的 B 位置 1
 Number of Links: 1

   Link connected to: a Transit Network
    (Link ID) Designated Router address: 192.168.123.3
    (Link Data) Router Interface address: 192.168.123.3
     Number of TOS metrics: 0
      TOS 0 Metrics: 1
```

我们注意到 R3 产生的 1 类 LSA 中，有着“Area Border Router”字样，标识该路由器是 ABR，而这，正是使得其产生的 1 类 LSA 中 B 位=1

再来看一下 R3 在 area0 中泛洪的 1 类 LSA：

R3#show ip ospf database router 3.3.3.3

```
            OSPF Router with ID (3.3.3.3) (Process ID 1)


                Router Link States (Area 0)
```

LS age: 1933

Options: (No TOS-capability, DC)

LS Type: Router Links

Link State ID: 3.3.3.3

Advertising Router: 3.3.3.3

LS Seq Number: 80000009

Checksum: 0x3B1E

Length: 48

Area Border Router

Number of Links: 2

Link connected to: another Router (point-to-point)

(Link ID) Neighboring Router ID: 4.4.4.4

(Link Data) Router Interface address: 192.168.34.3

Number of TOS metrics: 0

TOS 0 Metrics: 64

Link connected to: a Stub Network

(Link ID) Network/subnet number: 192.168.34.0

(Link Data) Network Mask: 255.255.255.0

Number of TOS metrics: 0

TOS 0 Metrics: 64

R3 在 area0 中只有一个接口，由于是 serial 接口，OSPF 用了两条 LINK 来描述。

我们来总结一下，针对不同的链路类型，OSPF 1 类 LSA 的有不同的内容：

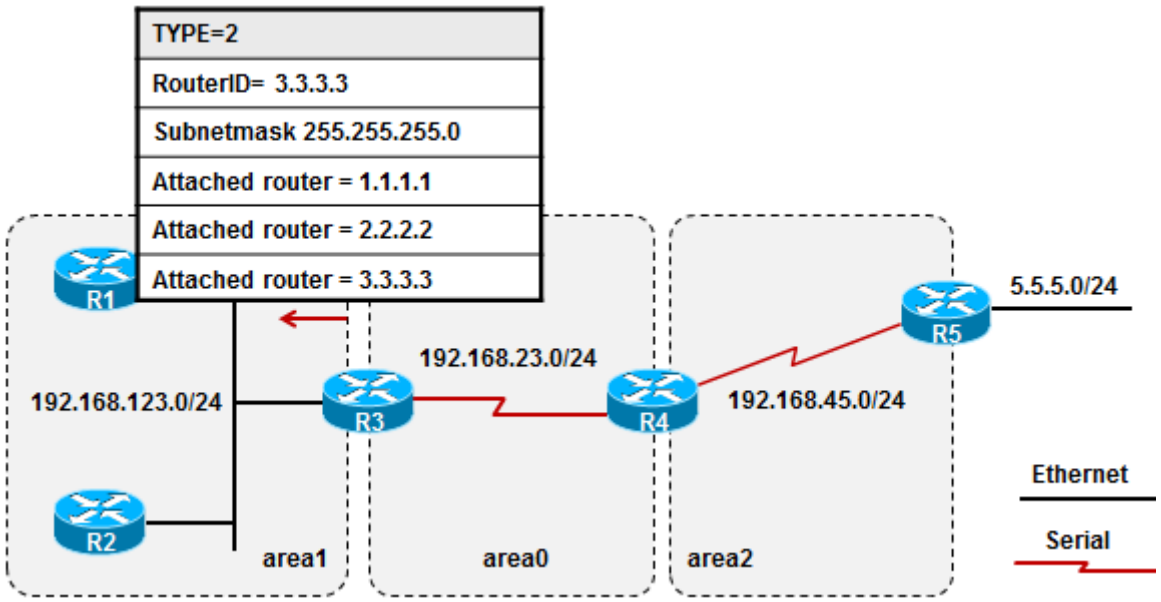
链路类型	连接	LINK ID	LINK DATA
1	点到点连接到另一台路由器	邻居路由器的路由器 ID	和网络相连的始发(本地)路由器接口的 IP 地址
2	连按到一个传送网络（常见以太网）	DR 路由器的接口的 IP 地址	和网络相连的始发路由器接口的 IP 地址
3	连按到-个末梢网络	IP 网络或子网地址	网络的 IP 地址或子网掩码
4	虚链路	邻居路由器的路由器 ID	始发路由器接口的 MIB-II ifIndex 值

2. Type 2 LSA：Network LSA

在 MA 网络中，会选举 DR、BDR，而所有的 Drother 都只能和 DR 及 BDR 建立邻接关系.

从某种层面上说，DR 实际上代表了这个 MA 网络，在本区域内泛洪 2 类 LSA，来呈现该 MA 网络中的所有路由器。因此 2 类 LSA 仅存在于有 MA 网络的区域中，并且由 DR 发送，用来描述这个 MA 网络中的所有路

由器（的 Router-ID）。



在上例中，R3 作为 123.0 网络的 DR，因此 R1、R2 都只和 R3 建立邻接关系。这时候 R3 就成了这个 MA 网络的代表者，其发送 2 类 LSA，该 LSA 包含的内容如图，详细信息见下，注意该 2 类 LSA 也只是在 area1 内泛洪。

R1#show ip ospf database network

```
            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Net Link States (Area 1)

  Routing Bit Set on this LSA
  LS age: 952
  Options: (No TOS-capability, DC)
  LS Type: Network Links
  Link State ID: 192.168.123.3 (address of Designated Router)
  Advertising Router: 3.3.3.3
  LS Seq Number: 80000001
  Checksum: 0x4CDD
  Length: 36
  Network Mask: /24
        Attached Router: 3.3.3.3
        Attached Router: 1.1.1.1
        Attached Router: 2.2.2.2
```

总结一下：

LSA2 描述 TransNet（包括 Broadcast 和 NBMA 网络）网络信息；

由 DR 产生，描述其在该 MA 网络上连接的所有路由器的 RouterID（包括 DR）以及该 MA 网络的掩码

LSA 类型 2 只在本区域 Area 内洪泛，不允许跨越 ABR；

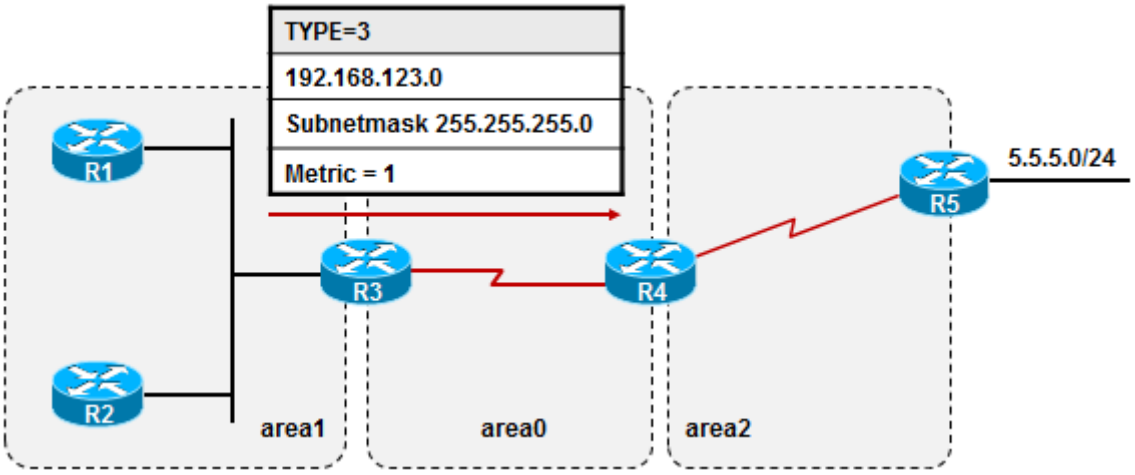
LSA 类型 2 中没有 COST 字段（因此需借助 1 类 LSA 来进行 SPF 算法）

得益于 1、2 类 LSA，OSPF 在一个区域内的路由计算就没有问题了，由此产生的路由，在路由表中的标记为“O”，表示本区域内的路由。

3. Type 3 LSA：Network summary LSA

前两类 LSA 解决了区域内路由计算的问题，那么区域间呢？如果路由器需要访问其他区域呢？这时就需要 3 类 LSA。3 类 LSA 是网络汇总 LSA，这里的汇总，其实翻译为归纳更贴切，它和路由汇总是完全不同的概念。由 ABR 同时属于两个以上的区域（其中必须有骨干区域），它知道这些区域的 1、2 类 LSA，那么就能做一件事：将某个区域的 1、2 类 LSA，做个归纳，然后为其他区域生成 3 类 LSA 并泛洪到其他区域中，如此一来，区域间的路由计算就没问题了。.

因此 3 类 LSA，由 ABR 产生：



上图中 ,R3 两个接口分属 area1 及 area0 ,它是一台 ABR ,因此它将 area1 内的 LSA 做个归纳 ,为 area0 生成 3 类 LSA，实际上就是 192.168.123.0 这个网段的信息，然后泛洪到 area0 中，接着，再被 R4 泛洪到 area2 中。这样，R4、R5 就都能根据这个 LSA3 计算到 123.0 的路由。

当然，R1、R2 也都能通过 R3 泛洪过来的 3 类 LSA 了解到 34.0、45.0 的路由信息。

R3#show ip ospf database

```
.....
                Summary Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
192.168.45.0    4.4.4.4         1533        0x80000001 0x00F065
192.168.123.0   3.3.3.3         1562        0x80000001 0x003912
```

Area0 中，有 2 个 3 类 LSA 在泛洪，分别描述的是 192.168.45.0、192.168.123.0 两个网络。

可以再详细点查看一下 3 类 LSA 的内容：

R3#show ip ospf database summary 192.168.123.0

```
                OSPF Router with ID (3.3.3.3) (Process ID 1)
                    Summary Net Link States (Area 0)
  LS age: 1886
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(Network)
  Link State ID: 192.168.123.0 (summary Network Number)
  Advertising Router: 3.3.3.3
  LS Seq Number: 80000001
  Checksum: 0x3912
  Length: 28
  Network Mask: /24
        TOS: 0  Metric: 1
```

以上是在 area0 中泛洪的 192.168.123.0 这条三类 LSA 的详细内容，注意到 3 类 LSA 的 link state ID 也就是这个 LSA 所描述的子网号，同时 LSA 中还描述了这个子网的掩码以及到达该子网的 metric，也就是 cost 值，这里 cost=1，其实就是 R3 的 fast0/0 口的接口 cost。

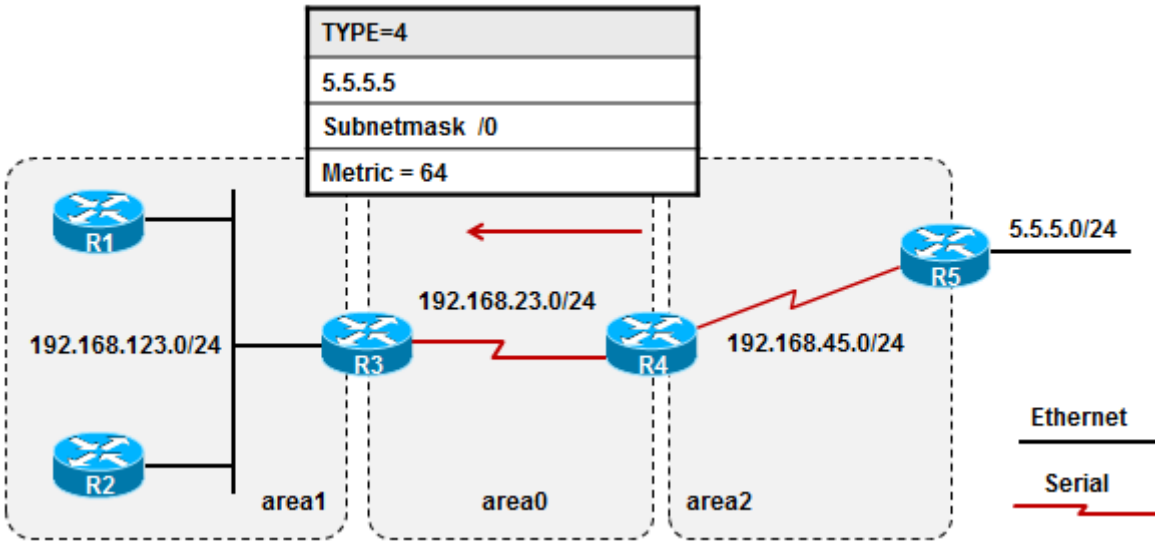
4. Type 4 LSA：ASBR summary LSA

4 类 LSA 是一个指向 ASBR 的 LSA，由该 ASBR 所在的区域中的 ABR 产生（这点要格外留意）。ASBR 作为域边界路由器，将外部的路由通过重发布的方式注入了 OSPF 域，这些外部路由在 OSPF 中进行传递（这些外部路由是以 5 类 LSA 的形式在域内传播），而我们 OSPF 内部的路由器如果想前往这些外部网段，则需要同时具备两个条件：

- **知道外部的路由（这通过重发布的动作，已经完成了注入，借助 5 类 LSA 完成传播）**
- **知道完成这个重分发动作的 ASBR 的位置**

关键在于第二点，与 ASBR 在同一区域的区域内部路由器（例如本实验中的 R4），能通过 ASBR 产生的 1 类 LSA 知道该 ASBR 的位置（1 类 LSA 中 E 位=1，所以与 ASBR 同区域的路由器都知道），但是问题来了，1 类 LSA 的泛洪范围是本区域内，那么该区域外的路由器，如何得知这台 ASBR 的位置呢？那么就需要借助 4 类 LSA 了。

因此 4 类 LSA 由 ABR 产生，用来告诉与 ASBR 不在同一个区域内的其他 OSPF Router 关于 ASBR 的信息。



上图中 R5 做为 ASBR ,在了路由重发布 ,将 5.5.5.0/24 的直连网段发布进了 OSPF。那么 R1、R2、R3、R4 如何能够访问 5.5.5.0 呢？上面已经说了，条件 1，是需要 5.5.5.0 的路由信息，这通过重发布将这些外部路由引入就 OK。条件 2，是需要了解 ASBR，也就是 R5 的信息。

R5 作为 OSPF 路由器，其本身会泛洪 1 类 LSA（在 area2 内部），并且该 LSA1 的 E 位被置 1，用以表示其本身是一台 ASBR，如此一来，R4 就能通过该 1 类 LSA，知道 ASBR 的存在，那么 R4 本身去往 5.5.5.0 就没问题了，但是 R1、R2、R3 呢？他们是无法收到上述的 1 类 LSA 的。因此 R4，作为 ASBR 所在区域的 ABR，就承担起解决这个问题的关键，它会向 area0 下发一条 4 类的 LSA，该 4 类 LSA 会进一步被泛洪，那么 R1、R2、R3 就都能了解到 R5，也就是 ASBR 了，所以条件 2 就具备了，问题也解决了。下面的输出，即是 4 类 LSA：

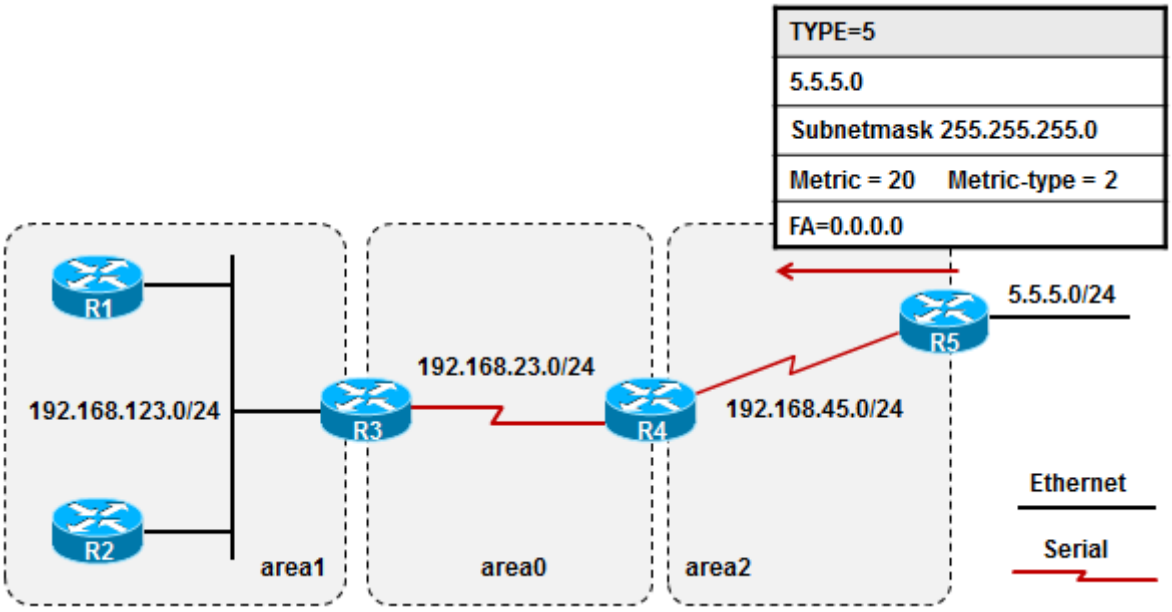
R4#show ip ospf database asbr-summary

```
            OSPF Router with ID (4.4.4.4) (Process ID 1)

                Summary ASB Link States (Area 0)

 LS age: 448
 Options: (No TOS-capability, DC, Upward)
 LS Type: Summary Links(AS Boundary Router)
 Link State ID: 5.5.5.5 (AS Boundary Router address)      // ASBR 的 RouterID
 Advertising Router: 4.4.4.4          //通告该 4 类 LSA 的路由器的 routerID，也就是 R4
 LS Seq Number: 80000002
 Checksum: 0x9C3A
 Length: 28
 Network Mask: /0
        TOS: 0   Metric: 64          //到达 ASBR 的开销
```

5. Type 5 LSA：AS external LSA

外部路由，被重发布进 OSPF 以后，将产生 5 类的 LSA，在 OSPF 与中进行传递；

因此 5 类 LSA 由 ASBR 产生



上图中，R5 做为 ASBR 将 5.5.5.0 这条直连网段重发布进了 OSPF，因此 R5 产生一个关于该外部路由的 5 类 LSA，其实就是描述的这条外部路由 5.5.5.0/24，体现如下：

R4#show ip ospf database external

```
            OSPF Router with ID (4.4.4.4) (Process ID 1)
                Type-5 AS External Link States
Routing Bit Set on this LSA
LS age: 584
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 5.5.5.0 (External Network Number )        //外部路由的网络号
Advertising Router: 5.5.5.5                              //通告该外部路由的 ASBR 的 routerID
LS Seq Number: 80000002
Checksum: 0x9AE1
Length: 36
Network Mask: /24                                        //外部路由的掩码
      Metric Type: 2 (Larger than any link state path)   //外部路由的 metric-type
      TOS: 0
      Metric: 20                                         //metric，在重发布的时候指定
      Forward Address: 0.0.0.0                           //FA 地址，是一个知识难点
```

External Route Tag: 0

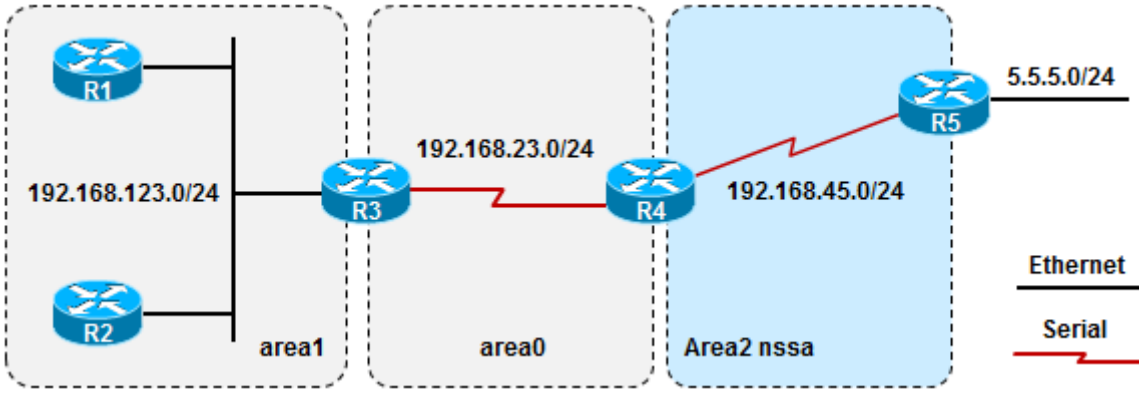
关于 type5 LSA 的 metric type，E1 及 E2 的区别请见下一小节

6. Type 6 LSA：Group-membership-LSA

这里不做讨论

7. Type 7 LSA ：NSSA external LSA

7 类 LSA 是一种非常特殊的 LSA 要注意的是这种 LSA 作为一种描述外部路由的 LSA 它只能被在 NSSA 中进行泛洪，不能跨越 NSSA 进入常规区域。NSSA 区域阻挡从骨干区域 area0 中过来的 5 类 LSA 进入，同时允许 NSSA 本地始发外部路由，这些外部路由以 7 类 LSA 的形式在本地 NSSA 中进行泛洪，当这些 7 类 LSA 到达 NSSA 的 ABR 时，由该 ABR 负责将这些 7 类 LSA 转换成 5 类 LSA，方可注入骨干区域。



留意一下上图，在上图中，我们将 area2 配置为 NSSA，接下去再观察 R5 的 LSDB：

R5#show ip ospf da

OSPF Router with ID (5.5.5.5) (Process ID 1)

Router Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
4.4.4.4	4.4.4.4	73	0x80000006	0x0048E8	2
5.5.5.5	5.5.5.5	71	0x80000007	0x00E249	2

Summary Net Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum
192.168.34.0	4.4.4.4	83	0x80000004	0x000A4E
192.168.123.0	4.4.4.4	83	0x80000004	0x003DC0

Type-7 AS External Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum Tag
5.5.5.0	5.5.5.5	75	0x80000001	0x00C90E 0

我们看到，重发布注入进来的 5.5.5.0 这条外部路由，是 7 类 LSA 所描述的。再看一下详细的内容：

R5#show ip ospf database nssa-external

```
            OSPF Router with ID (5.5.5.5) (Process ID 1)
                Type-7 AS External Link States (Area 2)
  LS age: 144
  Options: (No TOS-capability, Type 7/5 translation, DC)
  LS Type: AS External Link
  Link State ID: 5.5.5.0 (External Network Number )
  Advertising Router: 5.5.5.5
  LS Seq Number: 80000001
  Checksum: 0xC90E
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 192.168.45.5
        External Route Tag: 0
```

我们发现 7 类 LSA 的结构与 5 类 LSA 并无明显差异，其实在数据包方面，也并没有明显的差别。当然，他们的差异从应用或者规划的角度而言，是非常明显的，这里不做详细描述。

我们再看一下 R4 的 OSPF LSDB

R4#show ip ospf database

```
            OSPF Router with ID (4.4.4.4) (Process ID 1)

                Router Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum Link count
3.3.3.3         3.3.3.3         93          0x80000004 0x004519 2
4.4.4.4         4.4.4.4         29          0x80000005 0x00E86D 2

                Summary Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
192.168.45.0    4.4.4.4         49          0x80000003 0x00EC67
```

```
192.168.123.0     3.3.3.3          93          0x80000003 0x003514

                  Router Link States (Area 2)
Link ID           ADV Router       Age         Seq#       Checksum Link count
4.4.4.4           4.4.4.4          18          0x80000006 0x0048E8 2
5.5.5.5           5.5.5.5          18          0x80000007 0x00E249 2

                  Summary Net Link States (Area 2)
Link ID           ADV Router       Age         Seq#       Checksum
192.168.34.0      4.4.4.4          29          0x80000004 0x000A4E
192.168.123.0     4.4.4.4          29          0x80000004 0x003DC0

                  Type-7 AS External Link States (Area 2)
Link ID           ADV Router       Age         Seq#       Checksum Tag
5.5.5.0           5.5.5.5          24          0x80000001 0x00C90E 0

                  Type-5 AS External Link States
Link ID           ADV Router       Age         Seq#       Checksum Tag
5.5.5.0           4.4.4.4          10          0x80000001 0x007C69 0
```

我们看到 R4 收到 R5 始发的 7 类 LSA 后，装入的自己的 LSDB，同时他还做了一件利国利民的事情，那就是将这条 LSA“转换”成了 5 类 LSA，并将它传递给骨干区域，好让其他区域的 OSPF 路由器知道这条外部路由。当然，在 R4 的路由表中，该条 5.5.5.0 的外部路由：

R4#show ip route

```
O IA 192.168.123.0/24 [110/65] via 192.168.34.3, 00:09:01, Serial0/0
C       192.168.45.0/24 is directly connected, Serial0/1
        4.0.0.0/24 is subnetted, 1 subnets
C          4.4.4.0 is directly connected, Loopback0
        5.0.0.0/24 is subnetted, 1 subnets
O N2       5.5.5.0 [110/20] via 192.168.45.5, 00:09:01, Serial0/1
C       192.168.34.0/24 is directly connected, Serial0/0
```

5.2 E1、E2 的比较（N1 及 N2 类似）

Type5 LSA 由 ASBR 产生，告诉相同自治区的路由器通往外部 AS 的路由。

1. **E1/E2 路由的优选问题：**

E1 永远优于 E2 路由，也就是说，如果一台 OSPF 路由器，同时学习到去往同一个目的地的两条外部路由，一条为 OE1 一条为 OE2，则不论各自 metric 如何，OE1 的路由永远优选。

- **E1 路由之间比较**

当本地接收到多条描述同一外部路由信息的类型 1 外部 LSA（都是 OE1）时,需要比较每条外部 LSA 中携带的 Metric 与本地到达产生该 LSA 的 ASBR 的 Metric 之和（即路由表中显示的 Metric），并优选值小的类型 1 外部 LSA

如果比较的结果 Metric 之和相等，则产生路由的负载均衡。

- **E2 路由之间比较**

当本地接收到多条描述同一外部路由信息的类型 2 外部 LSA(都是 OE2)时 ,按照如下顺序比较不同的 Metric 值，从而实现优选：

1、首先比较外部 LSA 本身携带的 Metric，优选值小的外部 LSA。

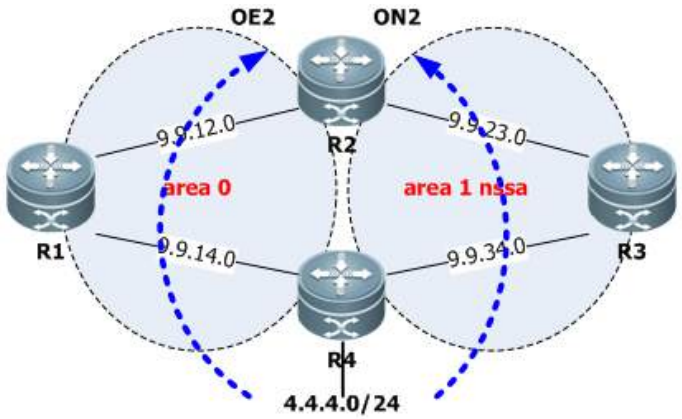
2、如果外部 LSA 携带的 Metric 值相同，则继续按照下面原则比较

3、比较本地到达通告每条 LSA 的 ASBR/FA 地址的 Metric，优选值小的。

4、如果本地到达通告每条 LSA 的 ASBR/FA 地址的 Metric 值相同，则产生路由负载均衡。

2. **ON2 和 OE2 的选择问题**

当 metric 不一样时，比较各自到 ASBR/FA 地址的内部 metric， 如果 metric 一样，CISCO 优选择 OE2

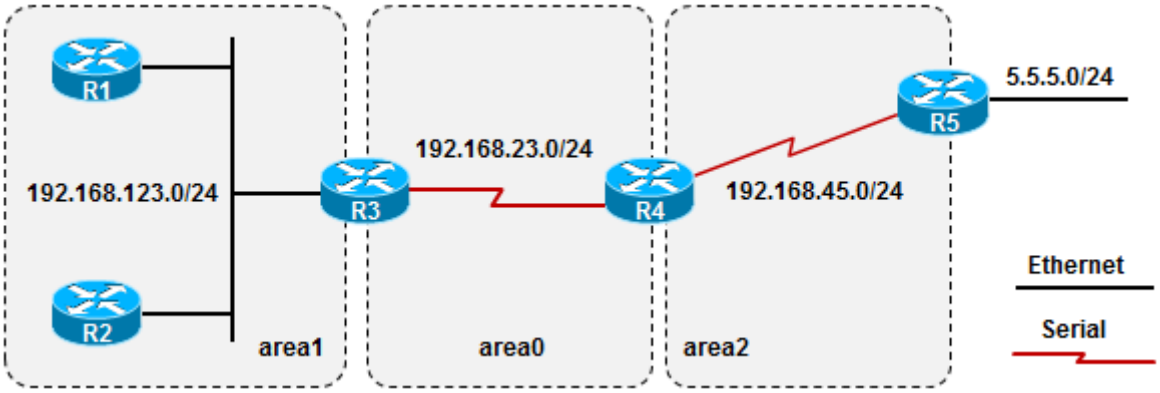


如图 R2 收到 4.4.4.0 的 OE2 和 ON2 ,在两者 metric 相等时 ,CISCO 默认选择 OE2 ,但是更改 R2 接口 cost 后，可以看到相应变化。

当上面的 OE2 和 ON2 两 metric 相等，各自到 ASBR 或 FA 地址的 metric 相当时，CISCO 默认优选 OE2，当两 metric 相等，但各自到 FA 或 ASBR 的 metrci 不等时，优选这个距离小的。

5.3 查看 LSDB

经过前面的讲解，相信大家对于各种类型的 LSA 都能掌握了，那么下面，我们就来全面的分析一下实验环境中的 LSDB。OSPF 最底层的东西就是 LSA ,LSA 在 OSPF 路由选择域中泛洪并发挥着自己的作用 ,OSPF Router 搜集到 LSA 后都存储在自己的 LSDB（链路状态数据库）里，并且根据这些 LSA 进行 SPF 算法的计算，最终得出前往每个目的地最优的路径。因此在从事 OSPF 的网络建设及维护过程中，学会读懂 LSDB，学会读懂每一类的 LSA 是非常关键的。



还是这张拓扑，完成基本配置后，我们看看 R1 的 LSDB。

R1#show ip ospf database

```
            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Router Link States (Area 1)          // 区域 1 中泛洪的 1 类 LSA
Link ID         ADV Router      Age         Seq#       Checksum  Link count
1.1.1.1         1.1.1.1         1203        0x80000003 0x001F3C  1
2.2.2.2         2.2.2.2         1099        0x80000003 0x00E071  1
3.3.3.3         3.3.3.3         1050        0x80000004 0x00A3A3  1

                Net Link States (Area 1)             // 区域 1 中泛洪的 2 类 LSA
Link ID         ADV Router      Age         Seq#       Checksum
192.168.123.3   3.3.3.3         1050        0x80000002 0x004ADE

                Summary Net Link States (Area 1)     // 区域 1 中泛洪的 3 类 LSA
Link ID         ADV Router      Age         Seq#       Checksum
192.168.34.0    3.3.3.3         1050        0x80000002 0x0086DD
192.168.45.0    3.3.3.3         1050        0x80000002 0x008F89

                Summary ASB Link States (Area 1) // 区域 1 中泛洪的 4 类 LSA
```

Link ID	ADV Router	Age	Seq#	Checksum
5.5.5.5	3.3.3.3	1050	0x80000002	0x003D5D

Type-5 AS External Link States // 5 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum Tag
5.5.5.0	5.5.5.5	1007	0x80000002	0x009AE1 0

R1 属于常规区域内部路由器，因此其 LSDB 相对比较好读，分别有 1、2、3、4、5 类 LSA；其中 1 类 LSA 产生自本区域（area1）的三台路由器 R1、R2、R3。2 类 LSA 产生自本 MA 网络的 DR 也就是 R3。3 类 LSA 产生自本区域的 ABR 也就是 R3，目的是告诉本区域内的路由器 OSPF 域内其他 area 的网络信息。4 类 LSA 也由 ABR R3 产生，用于告知本区域内的路由器关于 ASBR 的信息。5 类 LSA 是用于描述域外路由的，由 ASBR R5 产生并最终被泛洪到本区域。R2 的 LSDB 应该和 R1 类似，这里不再赘述。

R3#show ip ospf database

OSPF Router with ID (3.3.3.3) (Process ID 1)

Router Link States (Area 0) // 区域 0 中泛洪的 1 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum	Link count
3.3.3.3	3.3.3.3	1338	0x80000003	0x004718	2
4.4.4.4	4.4.4.4	1311	0x80000003	0x00E673	2

Summary Net Link States (Area 0) // 区域 0 中泛洪的 3 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum
192.168.45.0	4.4.4.4	1311	0x80000002	0x00EE66
192.168.123.0	3.3.3.3	1338	0x80000002	0x003713

Summary ASB Link States (Area 0) // 区域 0 中泛洪的 4 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum
5.5.5.5	4.4.4.4	1311	0x80000002	0x009C3A

Router Link States (Area 1) // 区域 1 中泛洪的 1 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum	Link count
1.1.1.1	1.1.1.1	1494	0x80000003	0x001F3C	1
2.2.2.2	2.2.2.2	1388	0x80000003	0x00E071	1
3.3.3.3	3.3.3.3	1338	0x80000004	0x00A3A3	1

Net Link States (Area 1) // 区域 1 中泛洪的 2 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum
192.168.123.3	3.3.3.3	1338	0x80000002	0x004ADE

Summary Net Link States (Area 1) // 区域 0 中泛洪的 3 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum
192.168.34.0	3.3.3.3	1340	0x80000002	0x0086DD
192.168.45.0	3.3.3.3	1340	0x80000002	0x008F89

Summary ASB Link States (Area 1) // 区域 0 中泛洪的 4 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum
5.5.5.5	3.3.3.3	1340	0x80000002	0x003D5D

Type-5 AS External Link States // 5 类 LSA

Link ID	ADV Router	Age	Seq#	Checksum	Tag
5.5.5.0	5.5.5.5	1297	0x80000002	0x009AE1	0

R3 的 LSDB 与 R1 有一个很直观的变化，那就是 R3 作为 ABR，同时连接到了 area0 及 area1，因此 R3 需为两个区域同时创建并泛洪 LSA。这里注意到 area0 中，并没有 2 类 LSA 存在，这是因为 2 类 LSA 只存在与 MA 网络中，而 area0 中并没有 MA 网络，因此没有 2 类 LSA，而 area1 中的 2 类 LSA，自然是不能超越区域泛洪到 area0 中的，当然，也没有这个必要。另外 R3 关于 area1 的 LSA，与 R1 并无两样，这里不再赘述。

R4#show ip ospf database

OSPF Router with ID (4.4.4.4) (Process ID 1)

Router Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
3.3.3.3	3.3.3.3	1661	0x80000003	0x004718	2
4.4.4.4	4.4.4.4	1631	0x80000003	0x00E673	2

Summary Net Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum
192.168.45.0	4.4.4.4	1631	0x80000002	0x00EE66
192.168.123.0	3.3.3.3	1661	0x80000002	0x003713

Summary ASB Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum
5.5.5.5	4.4.4.4	1632	0x80000002	0x009C3A

Router Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
4.4.4.4	4.4.4.4	1632	0x80000003	0x00A299	2
5.5.5.5	5.5.5.5	1616	0x80000004	0x0043F1	2

Summary Net Link States (Area 2)

Link ID	ADV Router	Age	Seq#	Checksum
192.168.34.0	4.4.4.4	1632	0x80000002	0x0068F7
192.168.123.0	4.4.4.4	1633	0x80000002	0x009B6A

Type-5 AS External Link States

Link ID	ADV Router	Age	Seq#	Checksum	Tag
5.5.5.0	5.5.5.5	1618	0x80000002	0x009AE1	0

我们再观察一下 R4 的 LSDB，R4 作为一台 ABR，LSDB 中自然也有两个区域的 LSA 描述信息。细心的朋友一定会发现，在 R4 LSDB 中，area2 里并没有 Summary ASB Link States 也就是 4 类 LSA，原因上面已经讲过了，这是因为 R4 与 ASBR R5 同属一个区域，通过 R5 发出的 1 类 LSA 即可知道 ASBR 的所在。

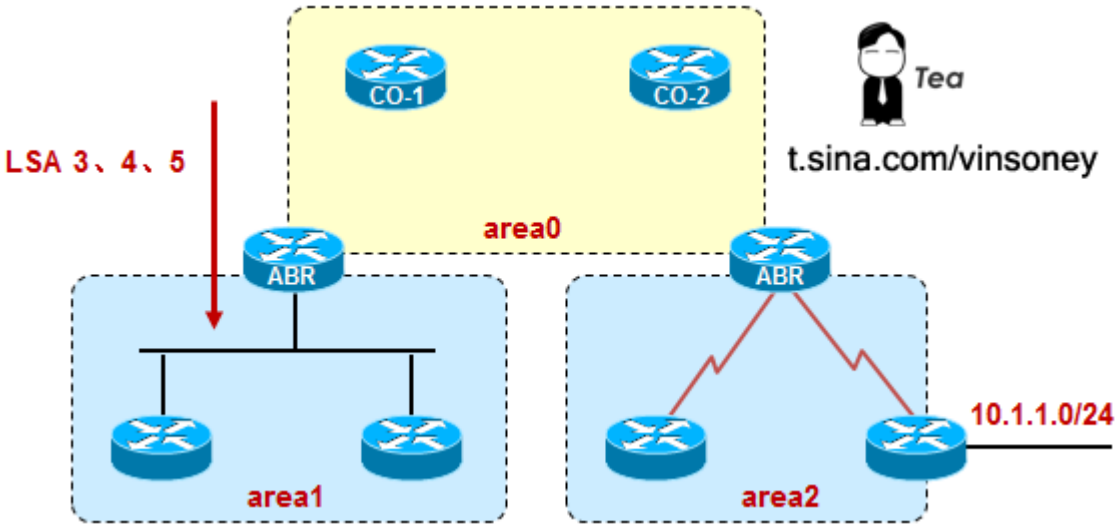
5.4 各区域中的 LSA

各区域内所允许出现的 LSA 总结如下：

- 骨干区域： 1、2、3、4、5
- 标准区域： 1、2、3、4、5
- Stub 区域： 1、2、3、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）
- 完全 Stub 区域： 1、2、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）
- NSSA 区域： 1、2、3、7
- 完全 NSSA 区域： 1、2、7、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）

6 OSPF 特殊区域

6.1 认识特殊区域

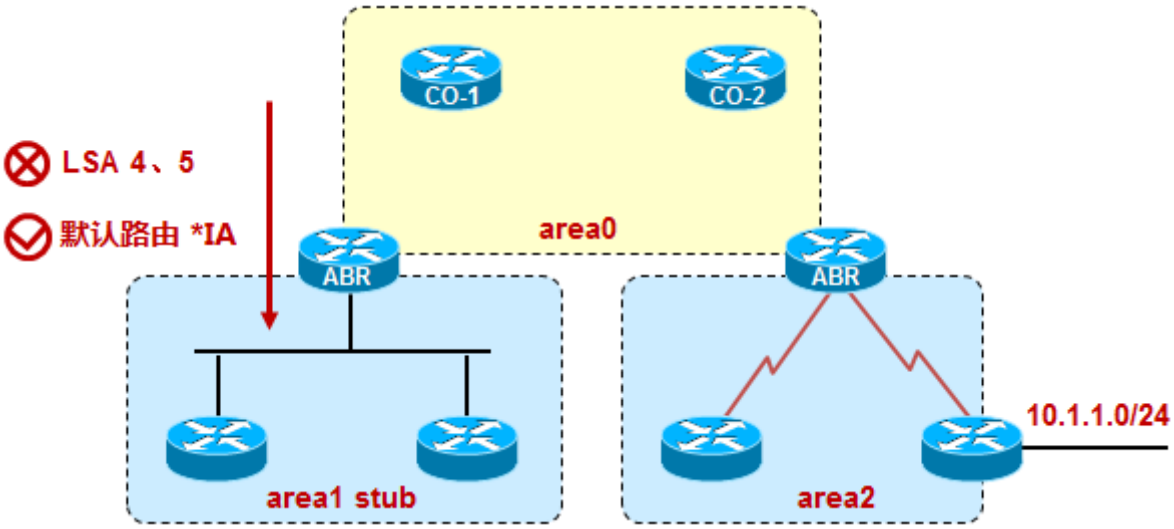


为了让我们的讲解更加的通俗易懂 ,我设计了上面这个拓扑 ,这是一个根据客户业务逻辑结构所涉及的 OSPF 网络，共有三个区域（实际上远远不止），骨干区域 area0 为一级行及二级行所运行，三级行运行的是 OSPF 的常规区域，为了保证网络的畅通，我们将网络中的各个角落都宣告进了 OSPF，感觉上很爽，但是其实路由器很压抑，毕竟随着设备越老越多、网络前缀越来越多，路由条目势必逐渐增多，那么路由器就亚历山大了，毕竟庞大的路由表及 LSA 在极大地消耗着路由器的资源。

从网络优化的角度，我们一直在试图在保证网络通畅的情况下减少网络中传递的路由条目及 LSA 的数量，路由汇总就是一种很好的方式，当然，从 OSPF 的设计规划角度，我们还有特殊区域可供我们灵活运用，下面来看看 OSPF 特殊区域是如何帮助我们减少 LSA 泛洪的。

我们拿 area1 做参考区域，当 area1 为常规区域时，区域中会有多少种 LSA 在泛洪呢？1 类是必然有的，由于 area1 中存在 MA 网络，因此 2 类 LSA 也有。其他区域的 3 类 LSA 被 ABR 也都注入进了本区域。另外，由于 area2 的 ASBR 引入的外部路由（5 类 LSA）也会被泛洪进 area1，当然 4 类 LSA 也跟着来了。那么如此一来，area1 中就有 1、2、3、4、5，共计 5 种类型的 LSA，齐活了。但是仔细一想我们就发现，其实 area1 作为“叶”区域，没必要知道外部路由的详细情况，我只需要知道有那么一条路，让我到达域外即可，因此，第一种特殊区域：末梢区域 stub area.

6.2 末梢区域 stub area



我们可以通过配置，将一个常规区域设置为 stub 区域。

Stub 区域将禁止 4、5 类 LSA 进入该区域，同时该区域的 ABR 将会自动下发一条默认路由（3 类 LSA）进该区域，以确保数据通路没有问题。这可以形象的理解为: “外面的世界再怎么精彩，你不用告诉我细节，只要让我出去就行了”。这就是 stub area 的设计思路。当引入大量的外部路由进 OSPF，适当的规划某些区域为 stub，可以起到不错的网络优化作用。

有一点值得注意，你不能将骨干区域 area0 配置为 stub 区域，同时，让一个区域被指定为 stub，区域内将不允许注入外部路由，也就是不能做重发布。

配置命令：

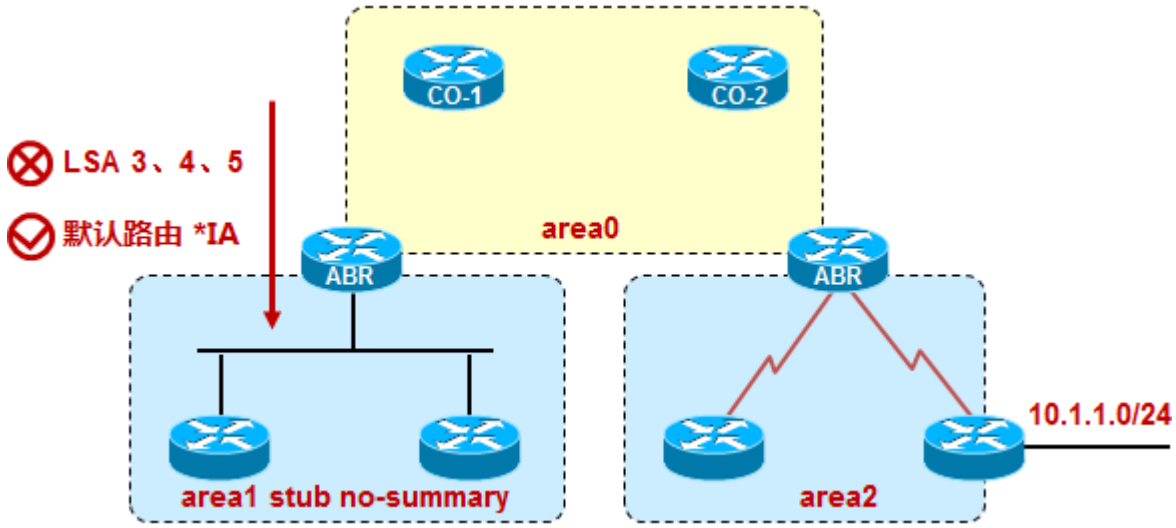
```
router ospf 1
area 1 stub
```

需要注意的是，该命令要配置在 stub 区域中的所有路由器上，如果某台路由器没有配置，那么它将无法去其他 stub area router 建立邻接关系。

实现效果：

Area1 中将不会在有 4、5 类 LSA，也就是 area2 重发布进来的路由，被 ABR 过滤掉了，同时 area1 中的路由器将获取到一条 3 类 LSA 的默认路由，该默认路由是由 area1 的 ABR 自动下发。

6.3 完全末梢区域 totally stub



通过将区域规划为 stub area，可以起到一定的网络优化作用，但是感觉上还不够彻底，除了外部路由，其他区域的路由（LSAs）其实我也没必要知道太多细节，完全用一条默认路由替代也行嘛，那么你可以将 area1 配置为完全末梢区域（totally stub area），当一个区域被配置为完全末梢区域，这个区域将：

- **阻挡 3、4、5 类 LSA 进入本区域**
- **区域的 ABR 自动下发一条 3 类 LSA 的默认路由进入本区域**

这么一来，area1 内路由器收到的 LSA 将进一步减少，在存储 LSA 及进行 SPF 算法运算的时候，耗费的资源自然也就减少了，另外当区域外拓扑出现变更的时候，对本区域的影响也将变为最小。

与 stub 区域类似，你无法将骨干区域 area0 配置为 totally stub area，当然，如果一个区域被指定为 totally stub area，你将不能在区域中的路由器上做路由重发布动作。

配置命令：

```
router ospf 1
area 1 stub
```

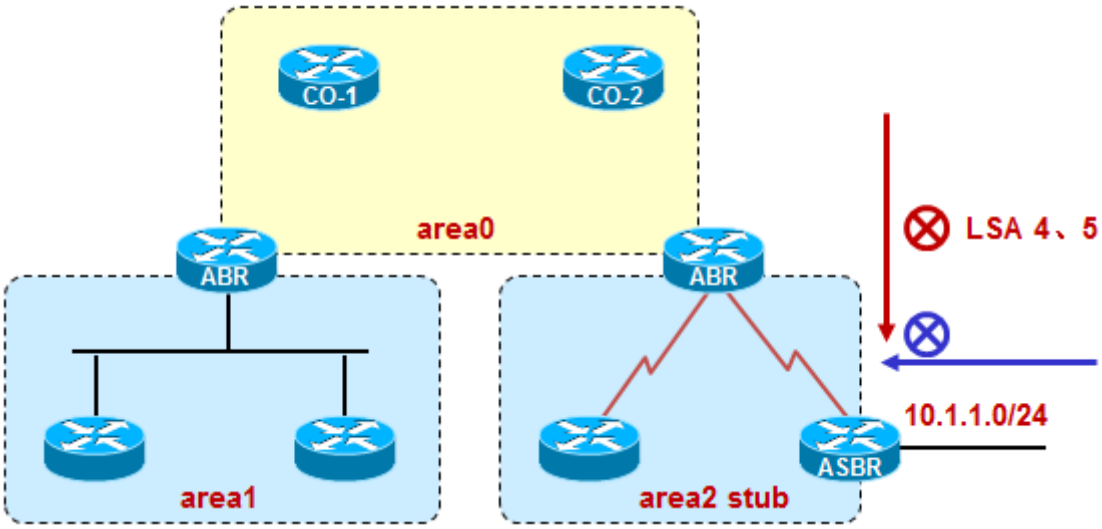
在完全末梢区域内的所有路由器，都配置上述命令，与 stub 区域的区别在于 ABR 的配置，下面是配置在完全末梢区域的 ABR 上的。

```
router ospf 1
area 1 stub no-summary
```

实现效果：

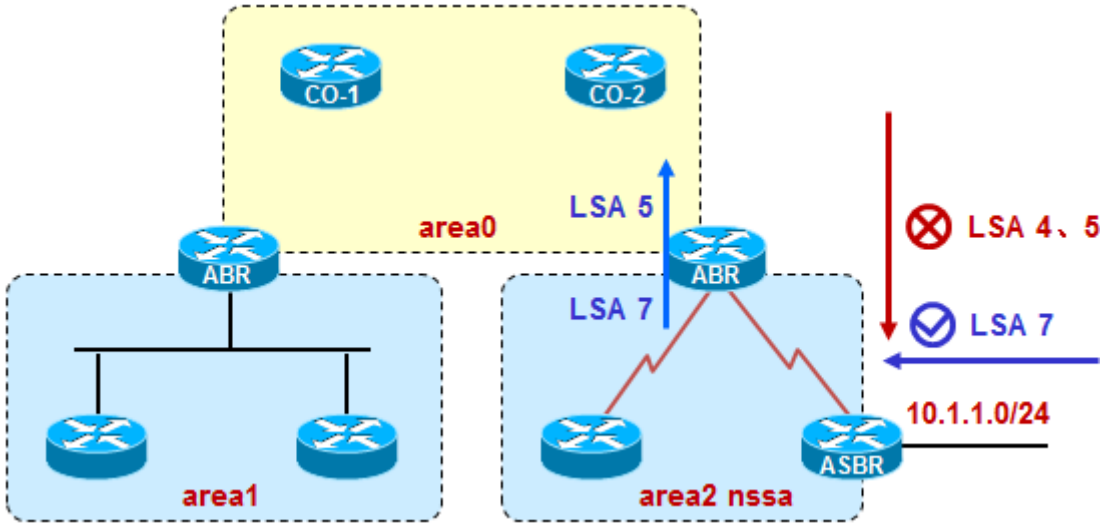
完成上述配置后，area1 内的路由器将只有本区域内的路由（ABR 除外），同时都能获取到 ABR 下发的 3 类的默认路由。也就是说其他区域的路由以及外部注入的路由都被 ABR 阻挡在外，取而代之的是一条默认路由。

6.4 非完全末梢区域 NSSA



在前面的知识基础上，我们已经了解到，在保证网络连通性的情况下，减少 LSA 的泛洪以及精简路由表，我们可以将特定区域配置为末梢区域或完全末梢区域。看上面的拓扑，我们将 area2 配置为 stub，那么，这个区域一来将阻挡来自其他区域的 4、5 类 LSA，同时区域内的路由器禁止重发布外部路由，那么如果此时我期望这个区域保持“阻挡其他区域过来的 4、5 类 LSA”这个特性，同时允许我在区域本地重发布路由呢？

例如，假设 area2 原先是作为一个末梢区域运行的，但突然有一个外部网络，需要接入到我们这个 OSPF 网络中，并且连接在 area2 中，那么这个时候，为了保证路由的可达，就必须向 area2 中注入外部路由了，而这又违法了 stub area 的规则。



这里就引入 NSSA（not-so-stubby-area）的概念，中文翻译过来，可以理解为非完全末梢区域，当你将一个区域配置为 NSSA，那么这个区域一来将阻挡骨干区域过来的 4、5 类 LSA，同时允许区域本地注入外部路由，这些外部路由以一种特殊的 LSA 类型—7 类 LSA 在 NSSA 中泛洪，并且 7 类 LSA 不允许进入骨干区域或常规区域，NSSA 的 ABR 会负责将 7 类 LSA“转换”成 5 类 LSA，从而在常规区域中进一步泛洪。上面的“允许区域本地注入”的意思是，NSSA 这个区域的路由器配置重发布。

值得注意的是，与 stub area 及 totally stub area 不同的是，如果你将一个区域配置为 NSSA，默认情况下 NSSA 的 ABR 不会自动下发默认路由进 NSSA，因此在 NSSA 环境下，需留意网络连通性问题。

配置命令：

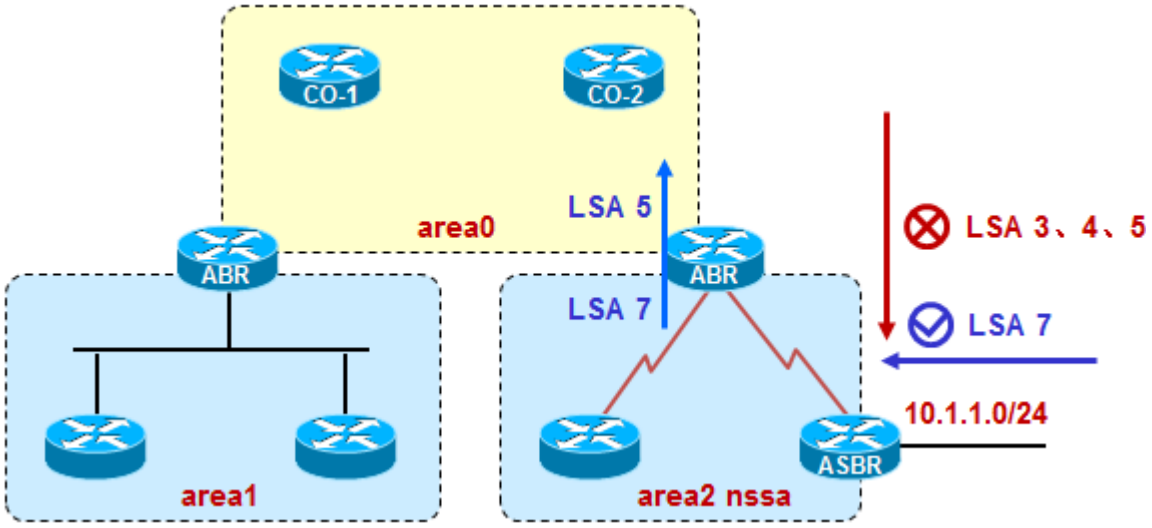
```
router ospf 1
area 2 nssa
```

上述命令需配置在 NSSA 内的所有路由器上。

实现效果：

将 area2 配置为 nssa 后，从骨干区域过来的 4、5 类 LSA 将无法进入 NSSA，也就是说如果 area1 做了重发布，那么这些重发布的外部路由将无法进入 NSSA（这是因为 NSSA 中不允许出现 5 类 LSA），与此同时，area2 允许本地的路由器做重发布动作，重发布进来的路由，以 LSA7 在 NSSA 中泛洪，大家在路由表中看到的这些外部路由，标记为“O N”，而这些 7 类 LSA 在“穿越”NSSA 的 ABR 进入骨干区域之前，由 ABR 负责将 7 类 LSA “转换”成 5 类 LSA。最终 area0 及 area1 也都能学习到这些外部路由，只不过，他们路由表中呈现的是“O E”标记。

6.5 Totally NSSA



Totally NSSA，这是一个不太好用中文翻译的词汇，完全 非完全末梢区域？显得有点尴尬，不过这个概念的理解并不像其名字那么唬人，Totally NSSA 是在 NSSA 区域的基础之上，进一步阻挡 NSSA 区域外的其他区域过来的 3 类 LSA，同时 ABR 自动下发一条 3 类的默认路由进 NSSA 区域。

配置命令：

```
router ospf 1
area 2 nssa
```

上述命令需配置在 NSSA 中的所有路由器上，ABR 的配置有所不同，如下：

```
router ospf 1
area 2 nssa no-summary
```

实现效果：

Area2 之外的其他 OSPF area 过来的 3、4、5 类 LSA，都会被阻挡在 NSSA 之外，同时 ABR 会自动下发默认路由进 NSSA，另外，NSSA 内的路由器做重发布动作，区域内的其他路由器将会学习到 7 类的外部 LSA，这些外部 LSA 会被 ABR 转成 5 类 LSA 并注入进骨干区域。

6.6 特殊区域总结

- **骨干区域　Backbone Area 0**
  本身是一个标准区域，负责连接非骨干区域，其它区域（非骨干区域）必须保证和骨干区域有直接的物理连接，因为区域之间的 LSA 必须经过骨干区域中转
- **常规(标准)区域　Standard Area**
  一个区域缺省是常规区域
- **末梢区域　Stub Area**
  把一个区域配成存根区域的好处是，阻挡 LSA4、LSA5 外部路由进入本地区域，从而精简路由表；同时 ABR 会自动产生 3 类的默认路由 LSA 注入进该区域。
- **完全末梢区域 Totally Stubby Area**
  完全末梢区域是一种对末梢区域的改进，进一步精简路由表；
  阻挡 LSA3、4、5 进入该区域，同时 ABR 自动下发 3 类默认路由。
- **非完全末梢区域　Not-so-stubby Area**
  即想阻挡 LSA5，自身又想引入外部路由，stub 的变种；
  NSSA 既阻挡外部 LSA5 的进入，同时区域内的路由器又可以引入外部路由 LSA7；
  LSA7 在 NSSA 内洪泛，通过 ABR 时转换为 LSA5 并被注入进骨干区域。
  ABR 不会缺省生成 0/0 默认路由进入本地区域，需手工配置
- **完全 NSSA　totally Not-so-stubby Area**
  在 NSSA 的基础上 ,进一步阻挡骨干区域过来的 3、4、5 类 LSA ,并且 ABR 自动下发 3 类的默认路由进 NSSA。

各区域内所允许出现的 LSA 总结如下：

- 骨干区域：　1、2、3、4、5
- 标准区域：　1、2、3、4、5
- Stub 区域：　1、2、3、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）
- 完全 Stub 区域：　1、2、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）
- NSSA 区域：　1、2、3、7

- 完全 NSSA 区域： 1、2、7、3 类 0.0.0.0/0（ABR 向区域内发起的一条 3 类缺省路由 LSA）

6.7 NSSA 详解

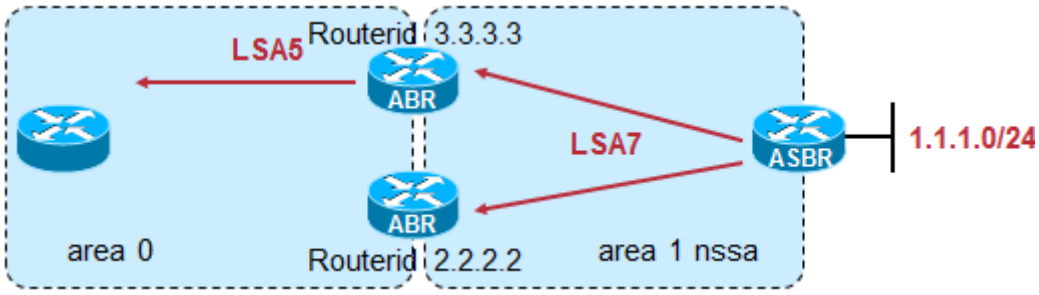
1. NSSA ABR 上的路由汇总动作

NSSA ABR 会将 7 类 LSA 转换为 5 类 LSA，在 7 转 5 的过程中可以使用 summary-address 通告汇总的 5 类 LSA，注意，这里汇总的是针对 NSSA 区域外部引入对汇总，但是如果仍想对 area 1 nssa 区域自己内部对路由做汇总，则使用 area range。

2. NSSA 的 ABR，在某种程度上可理解为常规区域的 ASBR，它不会为常规区域生成 4 类 LSA。

NSSA 的 ABR 产生了 5 类 LSA(7 转 5)并通告进骨干区域，因此这台 NSSA 的 ABR 对于骨干区域而言就是一台 ASBR。由于存在这个 7 转 5 的过程，真正的 ASBR(NSSA 区域中执行重发布的那台)信息就不需要被骨干区域路由器所知道，因此 NSSA 区域的 ABR 也不会为这个区域里的 ASBR 产生 4 类 LSA 并通告进骨干区域。

3. NSSA 双 ABR 问题



上图中，两个 ABR 都会收到 7 类 LSA，但只有 router-id 大的 ABR 执行 7 转 5。

两台 ABR 都会在 NSSA 区域泛洪 1 类 LSA，并从中了解到对方的存在。

4. N/P 位

Hello 报文以及 LSA 报文中的 Option 字段

•	O	DC	EA	N/P	MC	E	T

其中 N/P 位为 1bit，在 HELLO 和 LSA 中都有携带 option 字段，N/P 位在两者中分别有不同意义

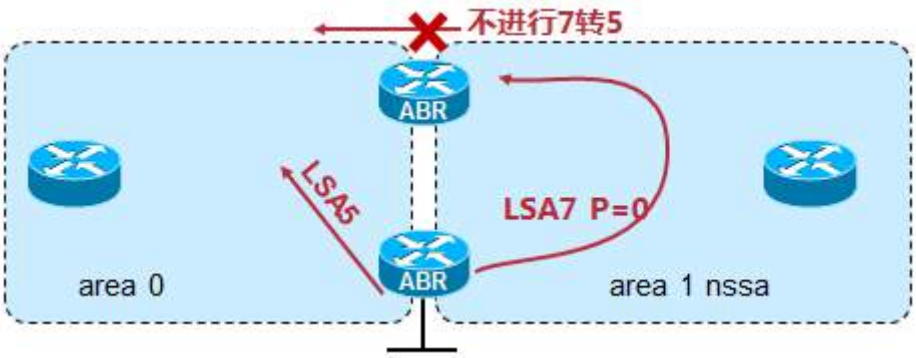
在 Hello 报文中：N bit 指示该路由器为 NSSA 区域路由器，当 N bit 被置 1 时 E bit 就必须被清零。

在 LSA 报文中：P bit 仅在 7 类 LSA 中出现，置 1 时指示 NSSA 区域的 ABR 能够将这条 7 类 LSA 转成 5 类 LSA。

```
r1#sh ip os da nssa-external
                OSPF Router with ID (1.1.1.1) (Process ID 1)
                Type-7 AS External Link States (Area 1)
            LS age: 9
            Options: (No TOS-capability, No Type 7/5 translation, DC)    //即 P 位被置 0
```

- **P bit 为 0 时，ABR 将不能将该 7 类 LSA 转换成 5 类 LSA**

  只有 NSSA 区域的 ABR 重发布路由时，通告的 7 类 LSA 中的 P bit 才为 0



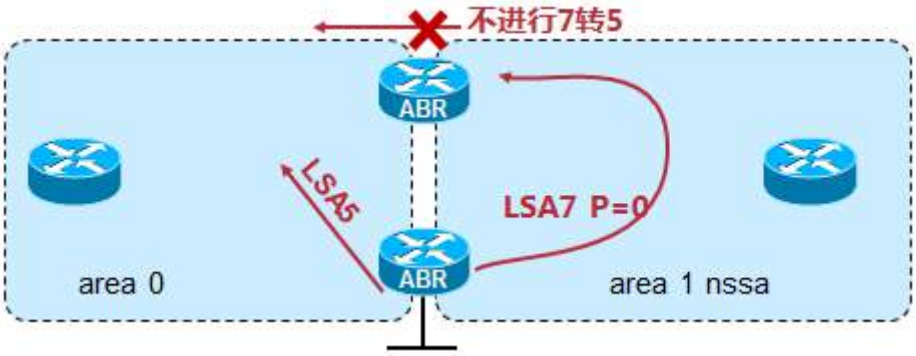

```
Open Shortest Path First
⊞ OSPF Header
⊟ LS Update Packet
    Number of LSAs: 1
  ⊟ LS Type: NSSA AS-External-LSA
      LS Age: 1 seconds
      Do Not Age: False
    ⊟ Options: 0x20 (DC)
        0... .... = DN: DN-bit is NOT set
        .0.. .... = O: O-bit is NOT set
        ..1. .... = DC: Demand Circuits are supported
        ...0 .... = L: The packet does NOT contain LLS data block
        .... 0... = NP: NSSA is NOT supported
        .... .0.. = MC: NOT Multicast Capable
        .... ..0. = E: NO External Routing Capability
        .... ...0 = MT: NO Multi-Topology Routing
      Link-State Advertisement Type: NSSA AS-External-LSA (7)
      Link State ID: 2.2.2.0
      Advertising Router: 2.2.2.2 (2.2.2.2)
      LS Sequence Number: 0x80000001
      LS Checksum: 0x1754
      Length: 36
      Netmask: 255.255.255.0
      External Type: Type 2 (metric is larger than any other link sta
      Metric: 20
      Forwarding Address: 10.1.24.2
      External Route Tag: 0
```

NSSA 区域 ABR 在重发布时，将 7 类 LSA 的 P 位置 0，通知其他 NSSA 区域 ABR 不要对该 LSA 进行

7 转 5，因为骨干区域内的其他路由器已经从骨干区域内收到该路由的 5 类 LSA，因此 NSSA 区域其他 ABR 执行 7 转 5 是没有意义的.

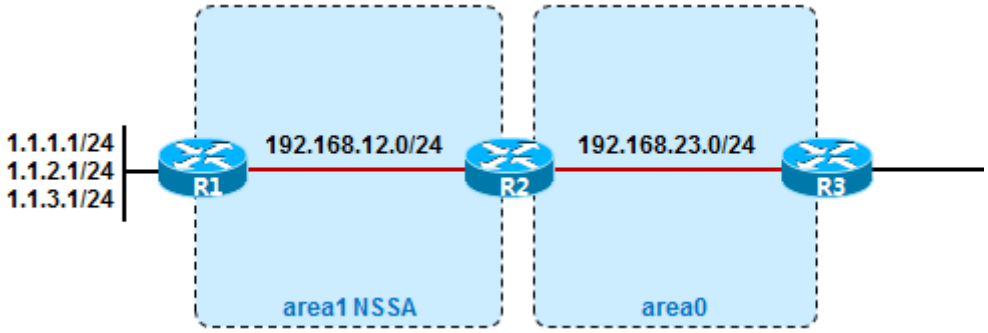
- **在 ABR 上重发布，默认情况，通告 7 类 LSA 进 NSSA 区域，通告 5 类 LSA 进入骨干区域**

  如果使用 area x nssa no-redistribution，则不通告 7 类 LSA 进入 NSSA 区域



这个一般在不希望 NSSA 学到太多对路由明细对情况下使用，比如网点路由器。 这个时候可在此配置基础上再在 ABR 上对 NSSA 区域发布一条默认路由，从而最大程度上对减少路由表。

5. NSSA 中默认路由的传递问题



- **NSSA 区域中的 ABR（R2）通告默认路由**
  - 将区域类型配置为完全 NSSA，自动下发 3 类 LSA 的默认路由；同时阻塞 LSA3；
  - 在进程下面使用 area x nssa default-information-originate 命令（本地无需默认路由的存在），自动下发 7 类 LSA 的默认路由，这时候 ABR 并不阻塞从骨干区域来的 LSA3 类进入 NSSA 区域
- **NSSA 区域中的 ASBR（R1）通告默认路由**

  在进程下面使用 area x nssa default-information-originate 命令，不会自动下发 7 类 LSA 的默认路由，必须事先在路由表中存在一条默认路由。

  【注意】在 NSSA 区域中，无论是 ASBR 上，还是 ABR 上，使用静态默认路由加上 default-information-originate [always]命令均无法注入默认路由，原因是采用该命令注入的默认路由，为 5 类 LSA，而 5 类 LSA 在 NSSA 中无法存活。

7 OSPF 配置

7.1 基本配置

1. **修改 OSPF 接口优先级（0-255）**

   取值范围为 0-255，默认为 1，优先级为 0 的 OSPF 接口在任何情况下都不具备选举资格

   ```
   Router (config-if)# ip ospf priority x
   ```

   DR 不会因为新加入的接口在其 HELLO 分组中报告了更高的优先级而放弃其地位，这是 DR 的非抢占性。

2. **修改 OSPF 接口 cost**

   OSPF 使用 cost 作为 metric （10E8 / 带宽）

   ```
   Router (config-if) # ip ospf cost x
   ```

3. **配置 OSPF 定时器**

   在缺省的情况下，失效时间间隔是 HellO 时间间隔的 4 倍

   ```
   Router (config-if) # Ip ospf hello-interval seconds
   Router (config-if) # Ip ospf dead-interval seconds
   ```

4. **配置 LSDB 过载保护**

   ```
   max-lsa xx [ yy% ] [ waning-only ] [ ignore-time  aaa ] [ ignore-count  bbb ] [ reset-time  zzz ]
   ```

   当 LSA 的个数超过定义的这个阀值，将会……

   xxx 为 LSA 最大个数

   yy 为百分比，默认 75%

   waning-only 超过最大值，只是警告（仍然收发 LSA）。默认关闭

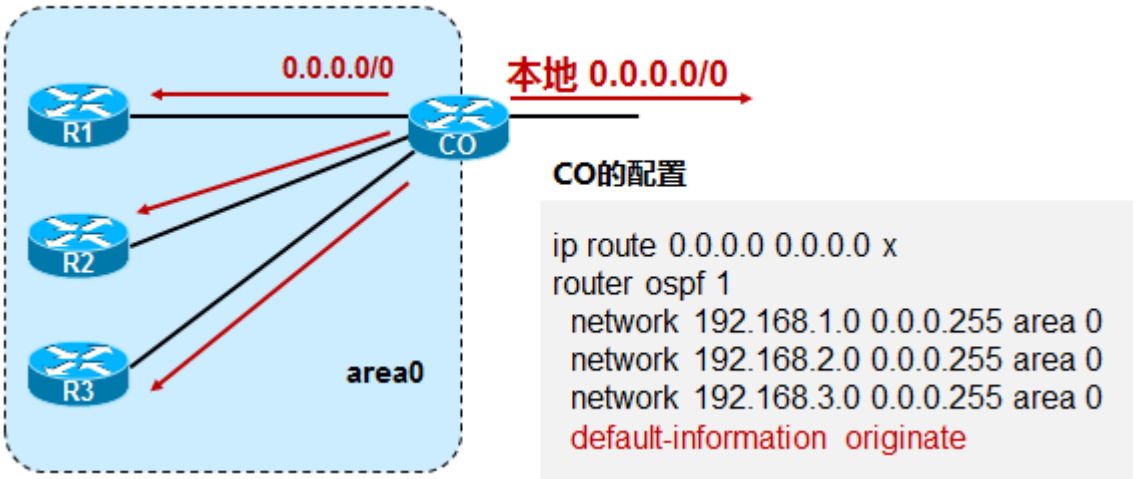
   Ignore-time 超过所定义的最大值后，超过多长时间，路由器才进入 ignore 状态。默认 5 分钟。Ignore 状态下会清除所有邻居及 LSA，不会尝试去建立邻居

   Ignore-count 默认 5 次，超过这个次数，则 OSPF 永远处于 ignore 状态，除非手工重启 ospf 进程

   Reset-time 默认 10 分钟，ignore 多长时间，脱离 ignore 状态

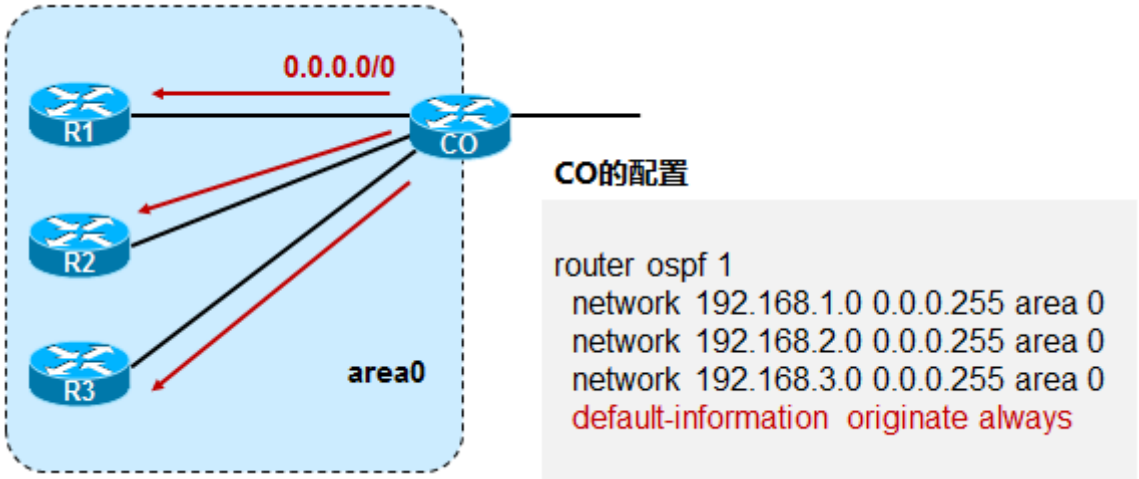
7.2 注入默认路由

- **OSPF 默认路由注入方式一：default-information originate**



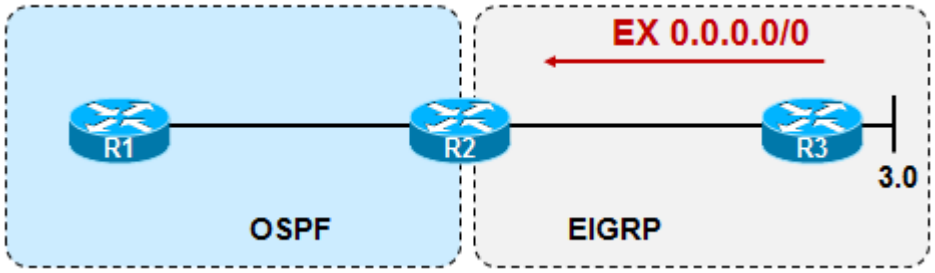
CO 本地必须有一条默认路由，该默认路由可以是静态的，也可以是通过其他动态路由协议获取到的，同时搭配上 default-information originate，即可向 OSPF 域中注入 OE 类型的默认路由，且该默认路由在 CO 本地这条默认失效后也随之 DOWN 掉。

- **OSPF 默认路由注入方式二：default-information originate always**



本地无需任何形式的默认路由，使用 default-information originate always 将始终向 OSPF 域中注入一条默认路由，并且该默认路由只要 CO 还 UP，则不会 DOWN。

- **使用 Redistribute 方式尝试从其他动态路由协议注入默认路由是徒劳的**



R1、R2 跑 OSPF；R2、R3 跑 EIGRP：

首先 R2 上配置 ip route 0.0.0.0 0.0.0.0 10.1.23.3，并将静态路由重发布进 OSPF，默认路由重发布失败

其次 R3 上向 R2 传递 EIGRP 默认路由及 3.0 网络 ,然后在 R2 上将 EIGRP 重发布进 OSPF ,发现 R3 上 3.0 网络及 R2、R3 之间的链路所在网段都被重发布进 OSPF，但是默认路由重发布失败。

由此可见，使用重发布的方式，无法将本地的静态默认路由或从其他协议学习到的默认路由注入 OSPF

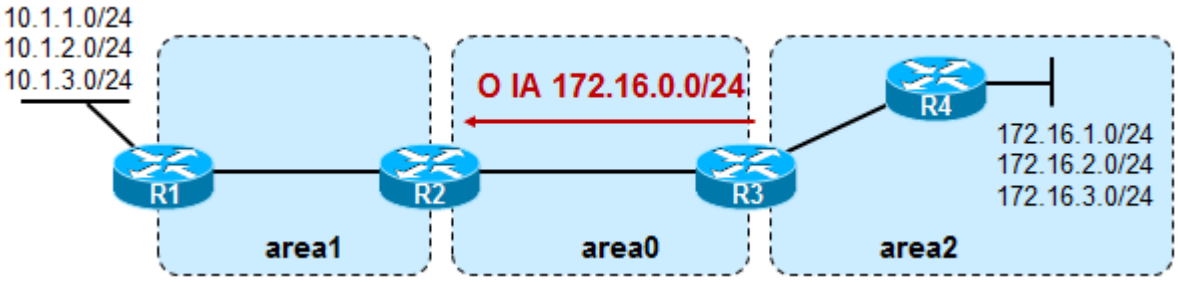
7.3 路由汇总

1. 区域间路由汇总

Router (config-router)#area 1 range 172.9.0.0 255.255.0.0 ?

cost	User specified metric for this range	//调整汇总的 3 类 LSA 的 Metric
advertise	Advertise this range (default)	//通告汇总但抑制明细 ,默认就携带了这个关键字
not-advertise	DoNotAdvertise this range	//抑制产生该汇总的 3 类 LSA 以及相关明细 3 类 LSA 也就是汇总和明细都没了，可用于过滤明细

注意该 area range 只对 1 类、2 类 LSA 产生作用，对 3 类 LSA 是不起作用对，也就是对 O IA 是无法汇总的，只对本区域内始发的路由起作用，并且必须配置在 ABR 上，产生的汇总路由将传递给其他区域。

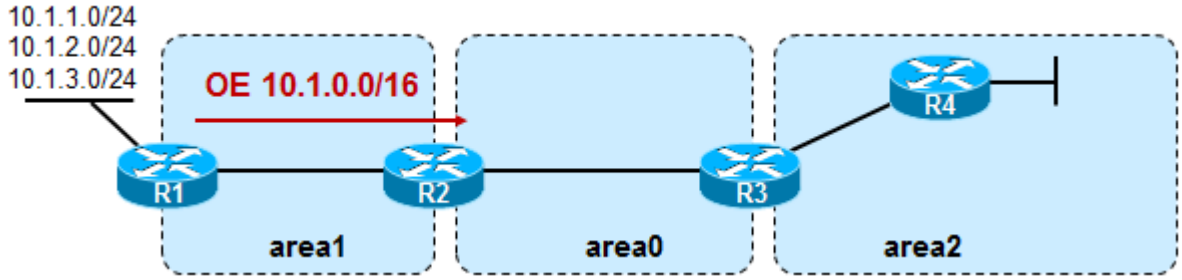


配置如下：

```
router ospf 1
  area 2 range 172.16.0.0 255.255.0.0
```

另外一点， 产生的这条汇总路由，metric 是多少呢？譬如 area2 内，由 R4 传递给 ABR R3 的明细路由如果 cost 各不相等，那么 R3 产生的这条汇总路由的 metric 即为这些明细路由中，cost 最小的那个值再加上 R3 路由入接口的 cost，两者之和就是 R3 产生的汇总路由的 metric。

2. 外部路由汇总




```
router ospf 1
summary-address 10.1.0.0 255.255.0.0 ?
  not-advertise          Do not advertise when translating OSPF type-7 LSA
  tag                    Set tag
```

对重发布进 OSPF 域的外部路由，需使用上述命令进行汇总，且注意配置的路由器必须是引入这些外部路由的那台 ASBR。

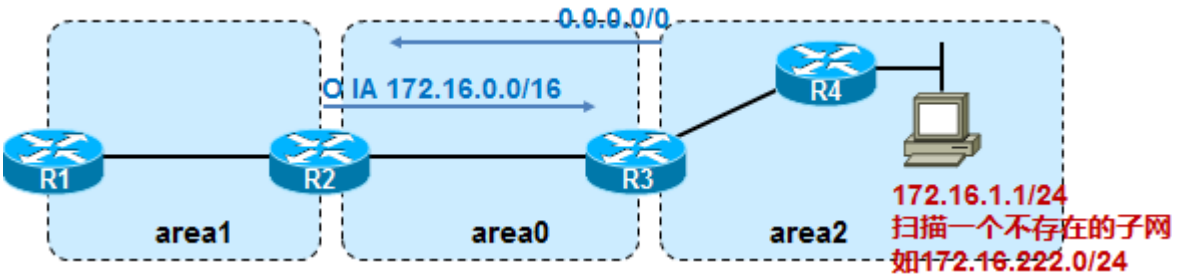
另外，当在 NSSA 中外部路由时，这些 7 类的 LSA 在穿越 NSSA 的 ABR 时，由 ABR 进行 7 转 5，这些 5 类 LSA 的路由，由于是由 NSSA 的 ABR 产生的，这时候此台 ABR 就有点 ASBR 的味道了，因此，亦可使用此命令针对这些路由进行汇总。

3. 关于汇总路由的防环问题

许多路由协议在手工汇总后，在执行汇总的 Router 的路由表里会出现一条指向 NULL0 的路由：

```
O    172.16.0.0/16 is a summary, 00:19:40, Null0
```

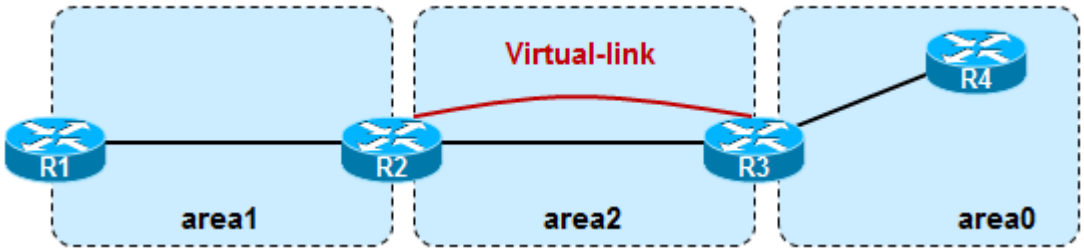
如果没有这个机制，看看会发生什么：



R3、R4 都有默认路由出去，R3 作为 ABR，将 area2 内 172.16.0.0 子网进行汇总，将汇总路由传递给了骨干区域。此时 R4 下某个子网里，有 PC 在扫描一个不存在的子网内的 IP，如 172.16.222.0/24，这些数据包会被默认路由匹配一路传递到 R2，而 R2 上由于收到 R3 传递过来的汇总路由，因此又把这些数据包丢回去给 R3，R3 又丢回 R2，如此反复，直到报文 TTL 为 0。

OSPF 为了解决这个问题，在进行汇总时，OSPF 在 R3 上（汇总路由的产生地）会在本地自动产生一条指向 Null0 的汇总路由，这样一来当再类似事件发生，数据包将在 R3 这就被丢弃。

7.4 virtual-link



OSPF 规定每个区域都必须与区域 0 相连，当出现了以上情况，可以在区域 1 中建立一条虚连接来过渡。

Area 过渡区域的 ID virtual-link 虚链路对端 router 的 RouterId

虚链路用 hello 包建立，单播。一旦建立，hello 将不再发送。凡是通过虚链路学到的 LSA，都会标 DNA，表示永远不老化。

例如在 R2 上看到的部分 LSDB：

Router Link States (Area 0)

Link ID	ADV Router	Age		Seq#	Checksum	Link count
2.2.2.2	2.2.2.2	270		0x80000002	0x00C38D	1
3.3.3.3	3.3.3.3	1	**(DNA)**	0x80000003	0x002161	3
4.4.4.4	4.4.4.4	87	**(DNA)**	0x80000001	0x00E775	2

R2 上的邻居表：

R2#sh ip os nei

Neighbor ID	Pri	State	Dead Time	Address	Interface
3.3.3.3	**0**	**FULL/ -**	**-**	**192.168.23.3**	**OSPF_VL0**
1.1.1.1	0	FULL/ -	00:00:35	192.168.12.1	Serial0/0
3.3.3.3	0	FULL/ -	00:00:31	192.168.23.3	Serial0/1

虚链路的网络类型是 p2p；中转区域不能是 stub 区域；另外 vl 只是一种临时或者割接的手段，不建议运用在常规网络的实施当中。

注意：虚连接是作为主干区域的一部分，因此如果在 area 0 上配置了验证，那么 virtual-link 也要做。

如上图，area0 开启了认证，那么由于 R2、R3 之间的虚链路在此前已经建立，所以暂时不受影响（不超时），但重启 OSPF 后，邻居关系则无法建立，此时需要分别在 R2 及 R3 上开启虚链路认证，例如 R2 上的配置：

```
area 2 virtual-link 3.3.3.3 authentication-key spoto
area 2 virtual-link 3.3.3.3 authentication
```

当然，如果 area0 不验证，虚链路上自己做认证，那验证的范围就仅在虚链路上

Virtual-link 具有可传递性。

7.5 OSPF 认证

一共三种，链路认证、区域认证、虚链路认证，每种认证都支持明文及 MD5 认证

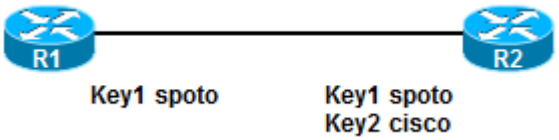
1. **链路（接口）认证**

明文认证：

```
Ip ospf authentication-key spoto      //接口下配置明文密钥
Ip ospf authentication                //为该接口开启明文认证
```

密文认证：

```
Ip ospf message-digest-key 1 md5 spoto      // 1 为 key ID， 接口下配置密文密钥
Ip ospf authentication message-digest       // 开启密文认证
```



上图中 ,R2 配置了 2 个 key ,R2 会将 key1 及 key2 给 R1 ,R1 使用 key1 与 R2 完成认证 ,当 R1 配置上 key2 cisco 后，R1 R2 之间会自动切换到使用 cisco 密钥进行认证，并且邻居关系不受影响，key1 也就不用了。

2. **区域认证**

一旦给某区域开启认证，则属于给区域的所有接口都要配置密钥（不要求所有的密码一致，只需要直连接口双方一致），这种验证方式相当于区域内的所有接口都开启接口验证。

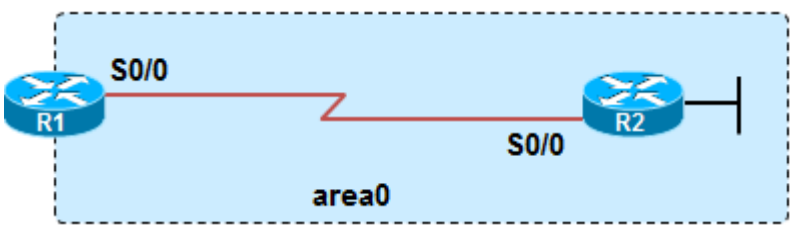
明文认证：

```
Ip ospf authentication-key spoto      //在所有属于开启认证区域的接口下配置明文密钥
area 0 authentication                 //在 ospf 进程中开启区域明文认证
```

密文认证：

```
Ip ospf message-digest –key 1 md5 spoto      //在所有属于开启认证区域的接口下配置密文密钥
area 0 authentication messae-digest          //开启区域密文认证
```

如果区域 0 开启了认证，则虚链路也要起认证，只要虚链路路由器上 area 0 authentication



下图中，R1 及 R2 启用了区域验证后，R1 及 R2 的 s0/0 口都要配置 key，而 R2 的 loopback 接口没有接任何邻居，仅是一个末梢网段，因此该接口即使不配置验证 key，R1 仍然能学习到关于该接口的 LSA，因此也能学习到该 loopback 的路由

3. **虚链路的认证**

明文认证

```
Area 1 virtual-link 2.2.2.2 authentication-key xx        // 配置明文密钥
Area 1 virtual-link 2.2.2.2 authentication               // 开启明文认证
```

MD5 认证

```
area 1 virtual-link 2.2.2.2 message-digest-key 1 md5 cisco
area 1 virtual-link 2.2.2.2 authentication message-digest
```

7.6 LSA 的过滤

```
area 1 filter-list prefix                    //在区域间过滤 LSA
```

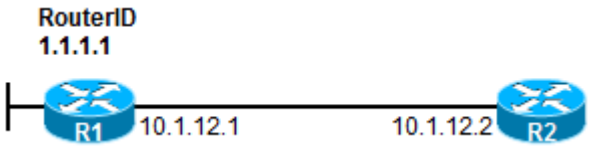
7.7 调整管理距离

distance AD 值 路由更新源 IP 反掩码 ACL

distance ospf external ad1 **inter-area** ad2 **intra-area** ad3

第一条命令能根据路由更新源并用 ACL 匹配路由条目进行设置 AD；这个路由更新源 IP 对于 OSPF 来说，指的就是 OSPF 的 routerID。

第二条命令中的 inter-area 针对的是 O IA 路由，intra-area 针对的是本区域内的路由



如上图，R2 能学习到邻居 R1（routerID1.1.1.1）发来的路由，如果希望在 R2 修改来自 R1 的路由的 AD 值使用 distance 222 1.1.1.1 0.0.0.0，可行，使用 distance 222 10.1.12.1 0.0.0.0，无效，因此这个更新源 IP，对于 OSPF 而言，实际上就是 OSPF ROUTER ID

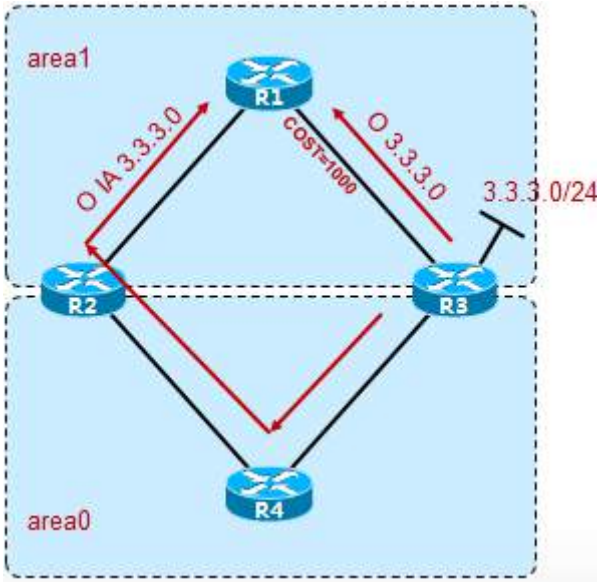
8 OSPF 路由优选规则

8.1 基本优选规则

1. 直连路由：本路由器发起的 LSA 1、2；
2. 区域内路由：O； LSA 1、2；
3. 区域间路由：O IA； LSA 3；
4. 1 类外部路由：O E1； LSA 5 类型 1；
5. 2 类外部路由：O E2： LSA 5 类型 2；
6. 1 类 NSSA 路由：O N1； LSA 7 类型 1；
7. 2 类 NSSA 路由：O N2； LSA 7 类型 2。

注意：E1 绝对优于 E2，而 E1 之间、E2 之间的比较，则需要严格注意（见本文档 LSA 详解部分章节的讲解）。另外，OE 及 ON 的优选，要看厂商，不同的厂商，优选方式不同。

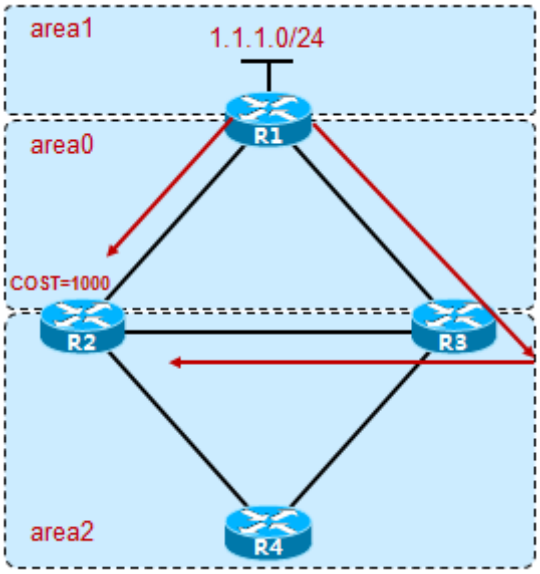
- **验证实验：O 及 O IA 的比较**



在这个图中，R1 同时从 R2 及 R3 学习到 3.3.3.0 的路由，一条是 O，一条是 OIA，R1 将绝对优选 R3 作为前往 3.3.3.0 的下一跳，而不看 metric。

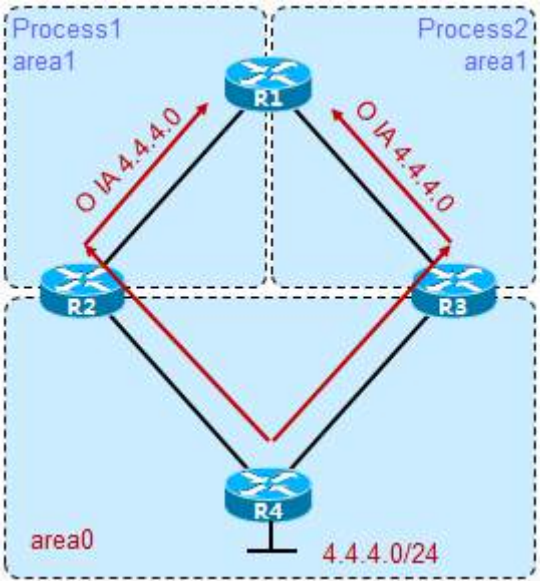
- **验证实验：OIA 之间的比较**

一般来说，OIA 之间比较 metric，谁小取谁，但是也有特例，如下：



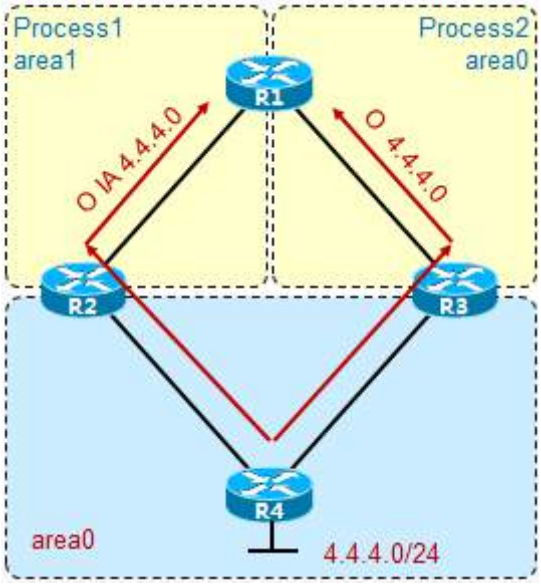
上图中，R2 上，能学习到由 ABR R1 及 R3 发出的关于路由 1.1.1.0 的 3 类 LSA，两条路由都是 O IA，但是 R2 将恒定优选 R1 作为去往 1.1.1.0 的下一跳，而不管这条路由的 metric 多大。

- **验证实验：多进程 OSPF 问题 1**



同一台路由器，本地如果跑两个 OSPF 进程，则各自独立，路由信息不会自动相互注入；另外如果同时从两个进程学习到同一条路由，则不管 metric 如何，先到先用，而不会负载均衡。

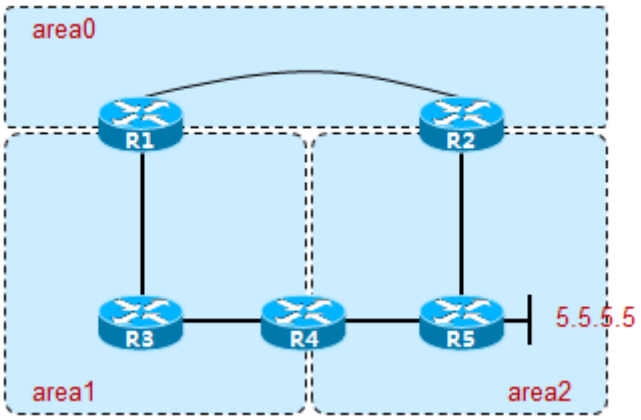
- **验证实验：多进程问题 2**



由于是两个 OSPF 进程，因此依然是先到先得，R1 上，如果率先与 R2 建立邻居关系，那么 4.4.4.0 的路由在路由表中，仍然是 O IA 的。

即使通过 R3 过来的是 O ,但是由于是不同的进程 ,因此只看谁先来 ,不管 metric 也不管什么 O>IA>OE1>OE2

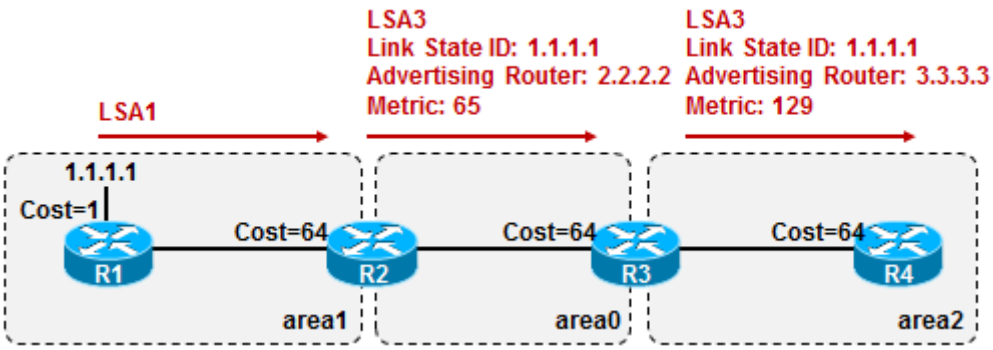
8.2 OSPF 的距离矢量特征



上图中，假设所有的 OSPF 接口 cost=1，那么 R5 更新 5.5.5.0 这条路由，R3 是能学习到的，R3 会怎么走呢？直观的来看，R3 去往 5.5.5.0 走 R4，cost 会更小，然而实际的结果是，R3 去 R5 会走 R1。

因为 R1 作为 ABR 产生的 3 类 LSA，而 R4 不是 ABR（还记得 ABR 的概念么？）无法下发 3 类 LSA，那么 R3 将只会从 R1 收到 3 类 LSA，因此不管从 R4 走 cost 如何小，R3 将只能通过 R1 到 5.5.5.0 网络。

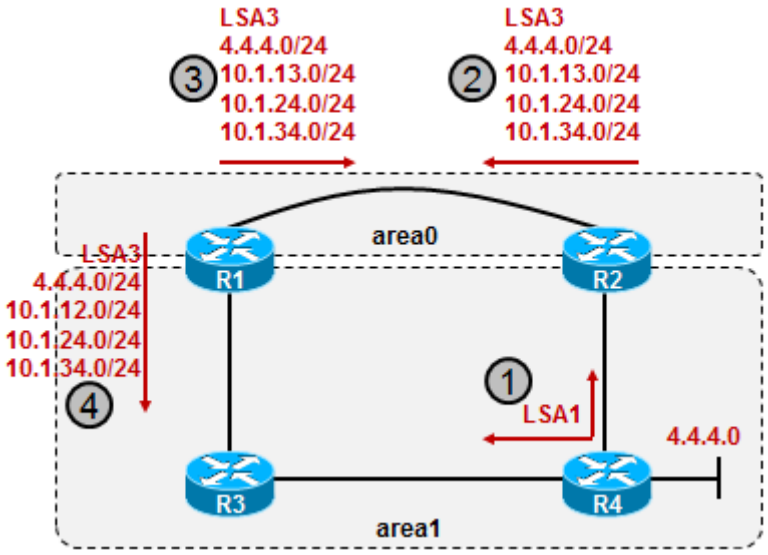
其实，OSPF 在区域内部，的确是链路状态协议的处理方式，然而在区域之间，OSPF 的对路由的处理却有典型的距离矢量特性。



我们看上面的图，R1 宣告 1.1.1.1/32 这条直连链路，R1 将在 area1 内泛洪自己的 LSA1，其中就包含这段直连链路，这是典型的链路状态协议的特征。随后 R2 搜集到 LSA1，并且为 area0 生成 LSA3，注意，这时候 LSA3 的通告路由器就变成了 2.2.2.2 也就是 R2，同时 R2 为路由加上自己的接口 metric，64+1=65。随后这条 LSA3 被 R3 接收，R3 将信息改写，仍然是这条路由，通告路由器变成了 3.3.3.3 也就是 R3，同时，R3 在自己接收到的 metric 65 的基础上，累加上自己接口的 cost，得到 129，并将这条 LSA3 再注入到 area2 – 上述动作，是典型的距离矢量路由协议的特征，RIP 不就是这么干的么？路由一跳一跳的传递，每跳路由器累加一跳然后发出去。好了那么 RIP 作为距离矢量路由协议，有水平分割等防环机制，**那么 OSPF 呢？OSPF 则要求，所有的非骨干区域的 3 类 LSA 的传递必须经过 area0 来中转，也就是说，所有常规区域必须与骨干区域 area0 直连，以此来达到防止环路的目的。**

这么一来本小节的第一个图就能理解了，既然 3 类 LSA 只能通过 area0 进行中转，且中转的动作必须由 ABR 来完成，那么 R4 这个屌丝自然无法为 area1 或 area2 去生成 3 类 LSA。

接下去，我们来看一个典型的问题：



上图中，编号 1、2、3、4 并非实际的协议运行步骤，只是我们观察点的一个转移而已。R1、R2、R3、R4 都会在 area1 内泛洪自己的 LSA，从而描述自己所有直连的 OSPF 接口（Link），因此 R3 将 4.4.4.0 的路由装载进路由表，这是经过 SPF 算法计算得出的最短路径，路由的类型为 O，区域内的路由。接下去我们将视角切换到

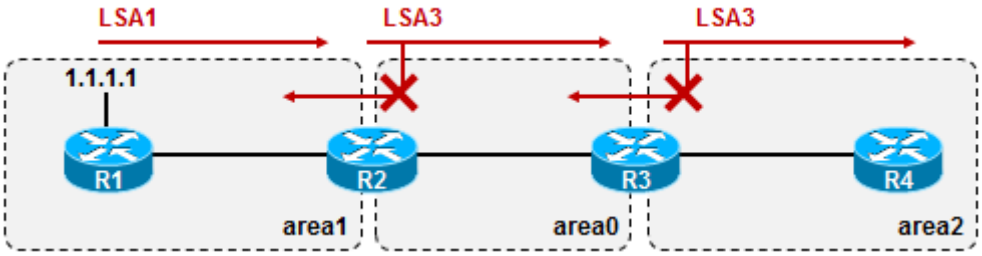
R1、R2，这两个 ABR 都会收到 area1 内路由器的 1LSA 的泛洪，它两会各自将 area1 内的 LSA1 及 LSA2（若有）搜集后归纳成 LSA3，并且在 area0 内进行泛洪，LSA3 的 LINKID 就是其描述的路由的前缀，在 R1 及 R2 上查看 OSPF database 可以看到，在 area0 中，R1 及 R2 都会各自泛洪 4 条 3 类 LSA，分别描述 10.1.13.0、10.1.34.0、10.1.24.0、4.4.4.0 这四个网段，如此一来，在 area0 中，就有共计 8 条 3 类 LSA 在泛洪。接下去我们将视角放到 R1，R1 从 R2 收到这四个网段的 3 类 LSA，会将这些 LSA 的通告路由器改为 R1 自身，并且进一步在 area1 内泛洪，如此一来 R3 将从 R1 收到四条 3 类 LSA，尽管 R3 并不需要这些 LSA，R3 会将忽略它们，甚至不会安装进自己的 LSDB。

OK，那么现在我们在 R3 上 定义个 ACL，匹配 4.4.4.0，随后 debug ip routing access-list x 去跟踪这条路由的更新，然后在 R4 上 DOWN 掉 4.4.4.0，这个时候会出现个诡异的现象：

```
*Mar  1 11:08:00.057: RT: del 4.4.4.4/32 via 10.1.34.4, ospf metric [110/65]
*Mar  1 11:08:00.057: RT: delete subnet route to 4.4.4.4/32   // 路由 delete，因为 R4 的告知
*Mar  1 11:08:00.057: RT: NET-RED 4.4.4.4/32
*Mar  1 11:08:00.121: RT: SET_LAST_RDB for 4.4.4.4/32
  NEW rdb: via 10.1.13.1

*Mar  1 11:08:00.121: RT: add 4.4.4.4/32 via 10.1.13.1, ospf metric [110/193]
*Mar  1 11:08:00.125: RT: NET-RED 4.4.4.4/32    //路由竟然又回来了，因为 R1 还没收敛，仍然通告 LSA3
R3#
*Mar  1 11:08:05.125: RT: del 4.4.4.4/32 via 10.1.13.1, ospf metric [110/193]
*Mar  1 11:08:05.125: RT: delete subnet route to 4.4.4.4/32   //路由 delete R1 收敛了
*Mar  1 11:08:05.129: RT: NET-RED 4.4.4.4/32
*Mar  1 11:08:05.129: RT: SET_LAST_RDB for 4.4.4.4/32
  NEW rdb: via 10.1.34.4

*Mar  1 11:08:05.133: RT: add 4.4.4.4/32 via 10.1.34.4, ospf metric [110/321]   //又回来了，R4 凌乱了
*Mar  1 11:08:05.133: RT: NET-RED 4.4.4.4/32
*Mar  1 11:08:05.265: RT: del 4.4.4.4/32 via 10.1.34.4, ospf metric [110/321]
*Mar  1 11:08:05.265: RT: delete subnet route to 4.4.4.4/32
*Mar  1 11:08:05.265: RT: NET-RED 4.4.4.4/32                //终于平静了
```

我们可以看到，R3 上关于这条路由经历了上面这个诡异的过程，造成这种现象的原因，还是在于 LSA3，OSPF 处理 LSA3 的行为，是典型的距离矢量的行为，是不是有感觉了？

当然，对于 LSA3，OSPF 也是存在类似水平分割的机制，如下图：



R2 将 area1 内的路由归纳成 LSA3 注入 area0，R3 收到后，将这些 LSA3 的信息修改，通告路由器变成自身，同时累加路由的 metric，然后再传入 area2，那么在这个过程中，R3 是不会将这些 LSA3 再注入回 area0 的。我们刚才那个的双 ABR 的环境比较特殊，因此存在上述的问题。

9 Forward Address

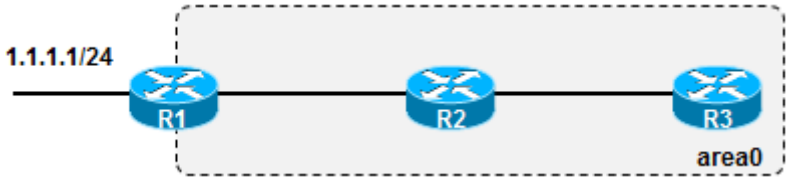
9.1 全 0 FA

The forwarding address is set to 0.0.0.0 if the ASBR redistributes routes and OSPF is *not enabled* on the next hop interface for those routes.

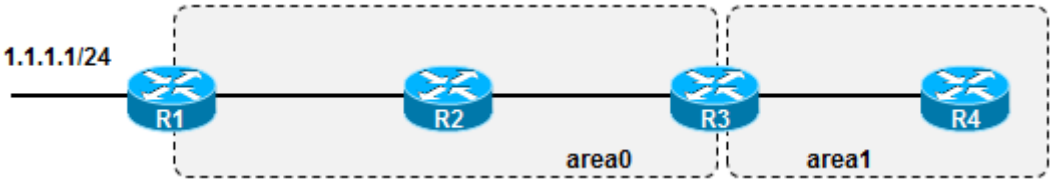
其他路由器在收到 5 类 LSA 时，必须在数据库中存在关于产生该 5 类 LSA 的 ASBR 信息。

ASBR 信息不是指路由表中存在 ASBR Router-id 的路由信息，而是数据库中存在 ASBR 产生的 Router LSA。

若数据库中存在 ASBR 产生的 Router LSA，则该路由器可以加载 5 类 LSA 进入路由表,其下一跳为到达 ASBR 的最近一跳



R2、R3 收到 R1 产生的 5 类 LSA 的 FA 为全 0，并且 R2、R3 都存在 R1 通告的 Router LSA，因此 R2、R3 都可以将 1.1.1.0 的外部路由加载进路由表。



当接收到 5 类 LSA 的路由器 R4 与产生该 5 类 LSA 的 ASBR 不在同一区域内时，由于 R1（ASBR）产生的 Router LSA 不会泛洪进 Area 1（1 类 LSA 只在本区域内泛洪），R4 的数据库中不存在 R1（ASBR）产生的 Router

LSA，因此将无法加载 5 类 LSA 进路由表，因为他不知道 ASBR 在哪。为了解决 ASBR 信息的问题，ABR（本例中的 R3）会向它所连接的其他区域（非 ASBR 所在的区域）通告一条 4 类 LSA，以标识 ASBR 信息（其中包含 ABR 到达该 ASBR 的距离）

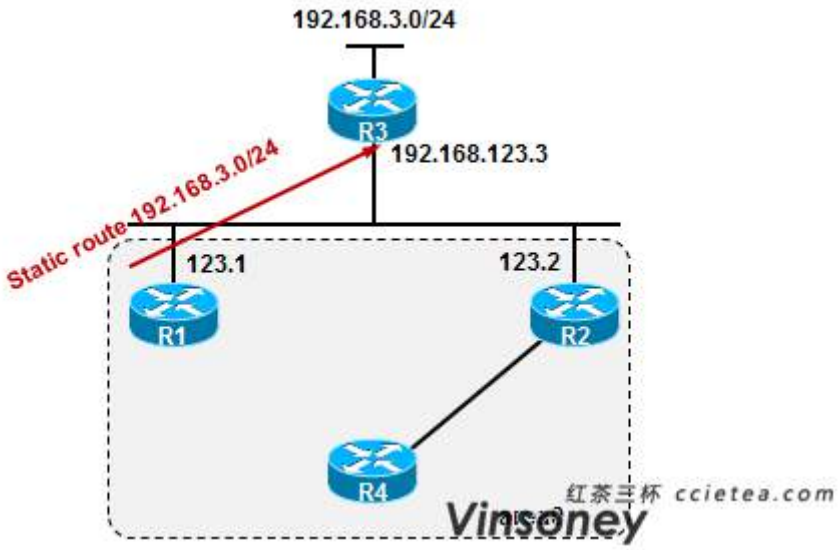
如此一来，R4 既有了关于 1.1.1.0 路由的 5 类 LSA，又有了通告这条外部路由的 ASBR 信息，因此可以将这条外部路由装载进路由表。

9.2 非 0 FA(no NSSA)

非 NSSA 区域 ASBR 产生的 5 类 LSA，其 FA 非 0 的条件（四个条件需全部满足，缺一不可）：

- 外部路由的下一跳接口启动 OSPF
- 外部路由的下一跳地址在 network 范围内
- 外部路由的下一跳接口没有被设置为被动接口
- 外部路由的下一跳接口的网络类型不是 P2P 或者 P2MP

当上述条件都满足，ASBR 在产生 5 类时就将 FA 置为外部路由的下一跳。

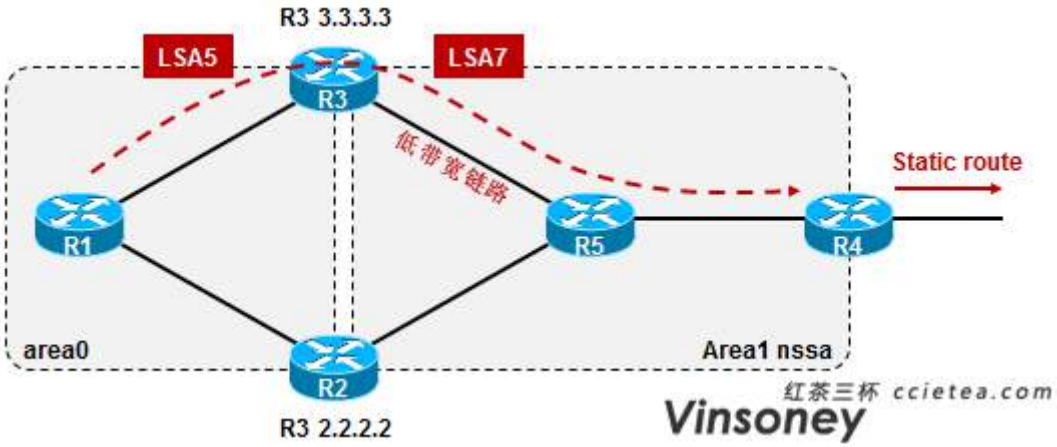


为什么需要 FA 呢？考虑一下如果没有 FA 的情况下回如何：R1 重发布静态路由(去往 3.0 ,下一跳是 123.3)，R2 接收到 R1 产生的 5 类 LSA，其 FA 为 0，因此加载该外部 LSA 进入路由表，下一跳为到达 ASBR（R1）最近的下一跳也就是 192.168.123.1 ,R4 接收到 R1 产生的 5 类 LSA ,其 FA 为 0 ,因此加载该外部 LSA 进入路由表，下一跳为到达 ASBR 最近的下一跳，结果是 R4 访问 192.168.3.0 是用的次优路径，R2---R1---R3 这样走。

解决办法： 使用非 0 FA，R1 始发的这条关于外部路由 192.168.3.0 的 5 类 LSA，其 FA 设置为 192.168.123.3，R2 接收到 R1 产生的 5 类 LSA，其 FA 非 0（符合上述的四个条件），FA 地址为 192.168.123.3，该地址通过直连路由可达，因此加载 5 类 LSA 进路由器，其下一跳为到达 FA 的下一跳。如此一来，次优路径的问题就解决了。

9.3 非 0 FA (NSSA)

- **验证实验 1**

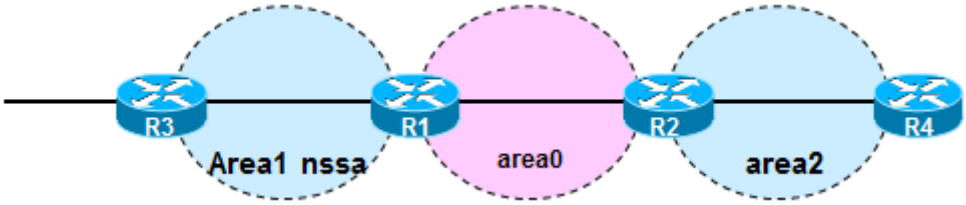


在 NSSA 区域存在多 ABR 时，只有 router-ID 大的 ABR 才会进行 7 转 5 的动作，这个我们已经知道鸟。

上图中，R4 重发布静态路由，只有 R3(Router-ID 大的)进行了 7 转 5，并且在 7 转 5 的过程中，没有产生 Type 4 LSA（描述 ASBR R4，因为右边是 NSSA 区域，R3 虽为 NSSA 的 ABR，但是同时也兼具 ASBR 的属性，这些 7 转 5 后生成的 5 类 LSA，对于 area0 而言，是产生自 R3 的），R1 无法掌握真实的 ASBR（R4）的信息，因此在路由表加载时，选择了一条次优路径，R1-R3-R5-R4。

使用 FA 可以解决这个问题，R4 在重发布进静态路由时，7 类 LSA 携带 FA =192.168.45.4，这个 FA 伴随着 R3 的 7 转 5 的动作传递到 R1，而 R1 去往这个外部路由，就看本地前往该 FA 的内部路由（最短路径），因此走 R2、R5、R4。

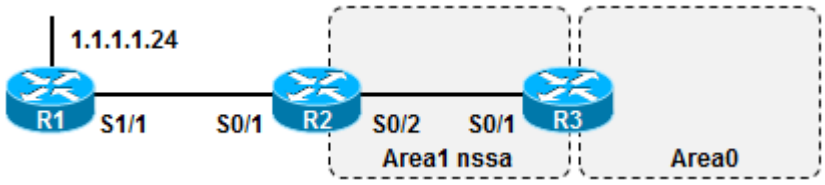
- **验证实验 2**



由于 r3 在重发布直连时，产生的 7 类 LSA 的 FA 被填充为一个非 0 的 IP 地址（172.16.13.3），R1 负责 7 转 5,FA 不变仍为 172.16.13.3，R4 在收到 5 类 LSA 后，发现其 FA 非 0，因此就首先在路由表中查找是否有到达 172.16.13.3 的 OSPF 内部路由，结果查找不到，5 类 LSA 所指示的网段 192.168.3.0/24 就无法加载进路由表。

解决办法：使用 area 1 nssa translate type7 suppress-fa ，R1 进行 7 转 5 时，会将 FA 地址置为全 0.

- **总结实验**

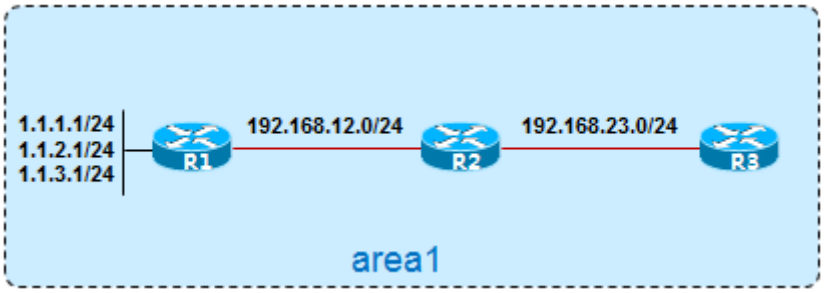


关于 NSSA 相关的 FA，具体填充的是什么地址，需要通过实验验证，并且不同的厂商有所不同。

1. **R2 作为 NSSA 区域的 ASBR，在 R2 上如果 1.1.1.0 是通过其他协议学习到的，比如 rip ，EIGRP，无**论是内部路由还是外部路由，在重分发进 OSPF 的时候：
   1) 如果 R2 的 S0/1 不宣告进 OSPF，那么 FA 将被设置为 192.168.23.2，即 R2 的 S0/2 的接口地址
   2) 如果 R2 的 S0/1 被宣告进 OSPF 且 R2 与 R1 之间链路类型为广播型链路，那么 FA 是 R1 的 S1/1 地址 192.168.12.1
   3) 如果 R2 的 S0/1 被宣告进 OSPF 且 R2 与 R1 之间链路类型为点到点型链路 那么 FA 是 R2 的 S0/1 地址 192.168.12.2
2. **如果被重分发进 OSPF 的网络不是通过其他协议学习到的，而是本地直连的**
   1) 如果除了 R2 的 S0/2 被宣告进 OSPF 之外，还有其他接口也被宣告进了 OSPF，那么选取其他接口的地址作为 FA，多个接口被宣告，选择最大的宣告 IP。
   2) 除了 R2 的 S0/2 没有其他接口被宣告进 OSPF，那么选取 R2 的 S0/2 的地址作为 FA

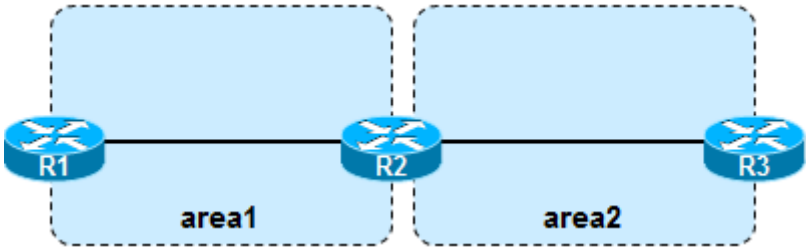
10 CASE 分析

1. 单区域非 area0 的情况



当只有一个区域的时候，这个区域就未必一定要是 area0，可以是任意的常规区域，由于单区域的话，LSA1、2 的泛洪都没有问题，且借助 LSA1、2 区域内的路由器都能了解到整个区域的所有链路，因此全网路由都是可达的。

2. 关于多区域无 area0 的情况



R1 及 R2，R2 及 R3 都能建立起邻居关系，但是 R1 无法学习到 area2 的 LSA，R3 同样无法获取 area1 的 LSA，因为没有 ABR，ABR 必须是与 area0 及非骨干区域直连的路由器，显然 R2 不是，既然不是 ABR，也就没有权利泛洪 type 3 LSA。

R1 宣告直连进 area1，则 R2 能学习到（单区域内的 LSA1、2 泛洪没有问题），R3 无法学习到。

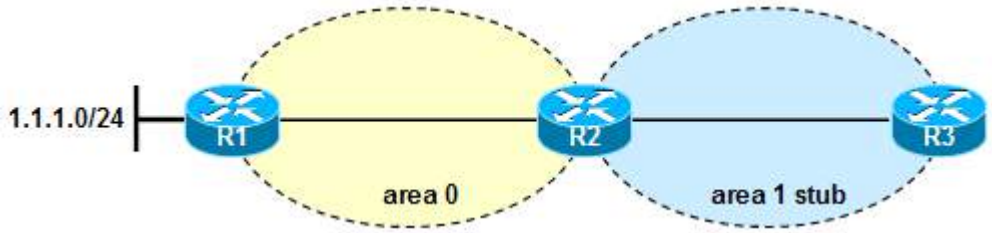
3. 关于辅助地址

只有在主网络或子网也运行 OSPF 协议的时候，OSPF 才会通告一个辅助的网络或子网。

OSPF 将把辅助地址看作是末梢网络(这些网络上没有 OSPF 邻居),从而不会在这些网络上发送 Hdlo 数据包。因此,在辅助网络上就无法建立邻接关系。

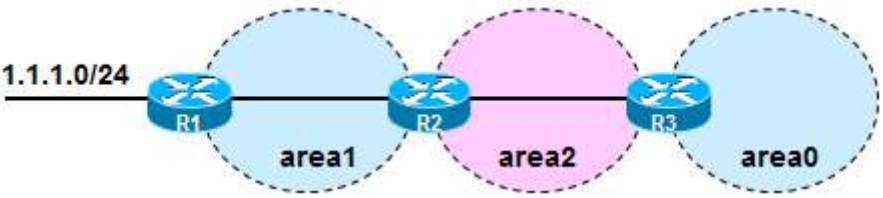
如果需要通过辅助地址与其他路由器建立邻居关系，如上图，R2 可使用子接口与 R1 建立邻居关系

4. 关于 4 类 LSA 的生成



R1 重发布 1.1.1.0 入 OSPF，此时 R1 为 ASBR，而 R2 为 ABR，但是目前的拓扑结构中，R2 又不能向 area1 注入 4 类 LSA，因为 area1 为 stub，所以这个时候 R2 的 database 里并没有 4 类 LSA，倘若 R2 上再挂一个常规区域 area2，则 R2 会为 area2 生成 LSA3 类，指向 R1。

5. 关于 4 类 LSA 的生成 2



R3 能收到 1.1.1.0 的 5 类 LSA，但是不会生成对应的路由条目，因为 R3 上没有关于 R1 这个 ASBR 的 4 类 LSA。原因是 R2 并不认为自己的 ABR（没有任何一个接口属于 area0），因此 R2 不会产生 4 类 LSA

11 参考书目

1. **BOOKS**

TCP/IP 卷一

疑难解析

Switching 技术笔记

红茶三杯 CCIE 学习文档

文档版本： 2.0　　更新时间： 2013-04-26

文档作者： 红茶三杯　　文档地址： http://ccietea.com

文档备注： 请关注文档版本及更新时间

1 以太网

1.1 基础知识

1. **About Ethernet**

Ethernet(以太网)于 20 世纪 70 年代中期 ,由 Xerox 公司分部 Palo Alto 研究中心(PARC)开发的。Xerox 最早发明的是一个 2Mbps 的以太网,后来又和 Intel 和 DEC 合作开发了出了 10Mbps 的以太网,俗称(Ethernet II 或Ethernet DIX).后来IEEE 通过802 委员会(802 Committee)把Ethernet标准化为IEEE 802.3。它和 Ethernet II 十分相似。

在 TCP/IP 中 ,以太网的 IP 数据报文的封装格式由 RFC 894 定义 ,IEEE802.3 网络的 IP 数据报文封装由 RFC 1042 定义。当今最常使用的封装格式是 RFC894 定义的格式，通常称为 Ethernet II 或者 Ethernet DIX。

2. **管理 MAC 表**

show mac address-table

clear mac address-table

绑定一个 mac 地址到一个接口

Switch(config)# mac address-table static 机器的 mac 接口 vlan vlan 号

要取消用 no mac address-static ……

1.2 以太网的数据链路层

在以太网中，针对不同的双工模式，提供不同的介质访问方法：

- 在半双工模式下采用的是 CSMA/CD 的访问方式。
- 而在全双工模式下则可以直接进行收发，不用预先判断链路的忙闲状态。

半双工和全双工是物理层的概念，而针对物理层的双工模式提供不同访问方式则是数据链路层的概念，这样就形成了以太网的一个重要特点：数据链路层和物理层是相关的。

由于以太网的物理层和数据链路层是相关的，针对物理层的不同工作模式，需要提供特定的数据链路层来访问。这给设计和应用带来了一些不便。

为此，一些组织和厂家提出把数据链路层再进行分层，分为逻辑链路控制子层（LLC）和媒体访问控制子层（MAC）。这样不同的物理层对应不同的 MAC 子层，LLC 子层则可以完全独立。如图 1-4 所示。

1. MAC 子层

MAC 子层负责如下任务：

- 提供物理链路的访问。
- 链路级的站点标识：在数据链路层识别网络上的各个站点。
- 也就是说，在该层次保留了一个站点地址，即 MAC 地址，来标识网络上的唯一一个站点。
- 链路级的数据传输：从 LLC 子层接收数据，附加上 MAC 地址和控制信息后把数据发送到物理链路上；在这个过程中提供校验等功能。

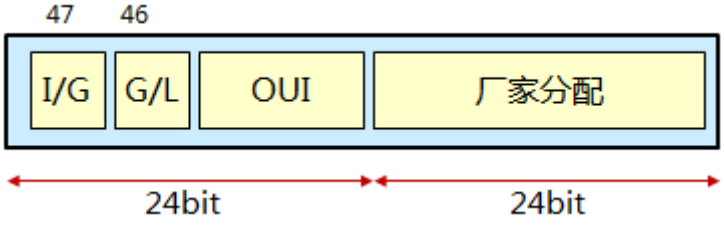
MAC 子层是物理层相关的，也就是说，不同的物理层有不同的 MAC 子层来进行访问。在以太网中，主要存在两种 MAC：

- 半双工 MAC：物理层运行模式是半双工时提供访问。
- 全双工 MAC：物理层运行模式是全双工时提供访问。

这两种 MAC 都集成在网卡中，网卡初始化的时候一般进行自动协商，根据自动协商的结果决定运行模式，然后根据运行模式选择相应的访问 MAC。

MAC 地址

MAC 地址是烧录在网卡（Network Interface Controller,NIC）的 ROM 里的



高位是 individual/group 位，当它的值为 0 时，就可以认为这个地址实际上是设备的 MAC 地址。当它的值为 1 时，就可以认为这个地址表示以太网中的广播地址或组播地址，或者表示 TR 和 FDDI 中的广播地址或功能地址。下一位是 G/L 位（也称为 U/L,这里的 U 表示全局）。当这一位设置为 0 时，就表示一个全局管理地址（由 IEEE 分配），当这一位为 1 时，就表示一个在管理上局部本地的地址（就像在 DECnet 中一样）。以太网一直使用全局唯一地址。

2. 以太网帧格式

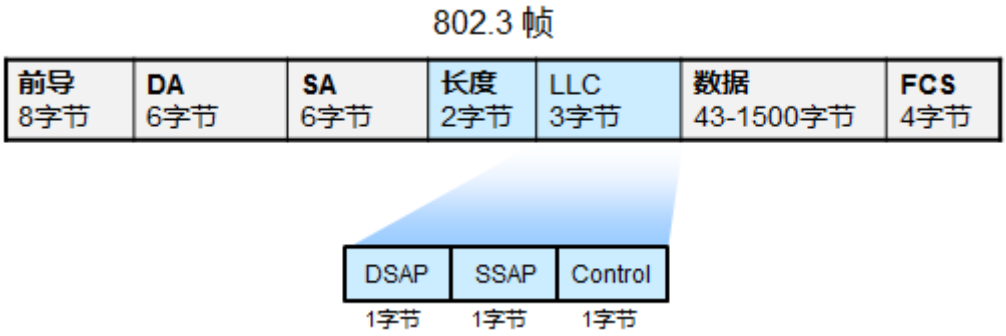
Ethernet II 帧

前导 8字节	DA 6字节	SA 6字节	类型 2字节	数据 46-1500字节	FCS 4字节

前导（Preamble）	包括 7 个字节的前导码（一串 1、0 间隔，用于信号同步）及 1 个字节的帧起始定界符（10101011）
类型（Type）	802.3 使用长度字段，但 Ethernet 帧使用类型字段来识别网络层的协议。在 EthernetII 帧中，两字节的类型字段用于标识数据字段中包含的高层协议，也就是说，该字段告诉接收设备如何解释数据字段。 在以太网中，多种协议可以在局域网中同时共存。因此，在 Ethernet II 的类型字段中设置相应的十六进制值提供了在局域网中支持多协议传输的机制。 • 类型字段取值为 0800 的帧代表 IP 协议帧。 • 类型字段取值为 0806 的帧代表 ARP 协议帧。 • 类型字段取值为 0835 的帧代表 RARP 协议帧。 • 类型字段取值为 8137 的帧代表 IPX 和 SPX 传输协议帧。 802.3 不能识别上层协议，且必须与专用的 LAN(比如 IPX)一起使用。
数据（DATA）	46-1500 字节；在接口下设置　mtu xxxx，指的就是这个，并且一般不允许手动修改。Ip mtu 指的是 ip 报文的最大值
帧校验序列 FCS（Fralue check sequence）	存储 CRC 循环冗余校验的结果

PS：在以太网中，由于冲突的存在，共享介质上两台主机同时发 frame，将产生冲突。根据特定的算法，以太网中，frame 的最小长度为 64 字节。

PS：目前我们所使用到的以太网帧基本都是 Ethernet II 帧



Length	Length 字段定义了 Data 字段包含的字节数。
LLC	LLC(Logical Link Control)由目的服务访问点 DSAP(Destination Service Access Point)、源服务访问点 SSAP（ Source Service Access Point ）和 Control 字段组成。

IEEE802.3 帧根据 DSAP 和 SSAP 字段的取值又可分为以下几类：

- 当 DSAP 和 SSAP 都取特定值 0xff 时，802.3 帧就变成了 Netware-ETHERNET 帧，用来承载 NetWare 类型的数据。
- 当 DSAP 和 SSAP 都取特定值 0xaa 时，802.3 帧就变成了 ETHERNET_SNAP 帧。
  ETHERNET_SNAP 帧可以用于传输多种协议。因此，SNAP 可以被看作一种扩展，它允许厂商创建自己的以太网传输协议。
  ETHERNET_SNAP 标准由 IEEE802.1 委员会制定 ,以保证 IEEE802.3 局域网和以太网之间的互操作性。
- DSAP 和 SSAP 其他的取值均为纯 IEEE802.3 帧。

3. LLC 子层

在前文的介绍中提到了 MAC 子层形成的帧结构，包括 IEEE802.3 的帧和 ETHERNET_II 帧。

在 ETHERNET_II 帧中，由 Type 字段区分上层协议，这时候就没有必要实现 LLC 子层，仅包含一个 MAC 子层。而 IEEE802.3 帧中的 LLC 子层除了定义传统的链路层服务之外，还增加了一些其他有用的特性。这些特性都由 DSAP、SSAP 和 Control 字段提供。

例如以下三种类型的点到点传输服务：

- **无连接的数据包传输服务**
  目前的以太网实现就是这种服务。
- **面向连接的可靠的数据传输服务**
  预先建立连接再传输数据，数据在传输过程中可靠性得到保证。
- **无连接的带确认的数据传输服务。**
  该类型的数据传输服务不需要建立连接，但它在数据的传输中增加了确认机制，使可靠性大大增加。

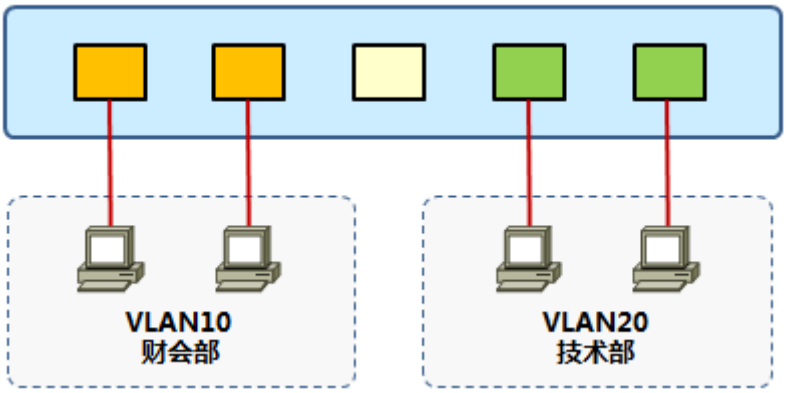
下面通过一个例子来说明 SSAP 和 DSAP 的应用。假设终端系统 A 和终端系统 B 要使用面向连接的可靠的数据传输服务，这时候会发生如下过程：

1. A 给 B 发送一个数据帧，请求建立一个面向连接的可靠连接。
2. B 接收到以后，判断自己的资源是否够用(即是否建立了太多的连接)，如果够用，则返回一个确认信息，该确认信息中包含了识别该连接的 SAP 值。
3. A 接收到回应后，知道 B 已经在本地建立了跟自己的连接。A 也开辟一个 SAP 值，来表示该连接，并发一个确认给 B，连接建立。
4. A 的 LLC 子层把自己要传送的数据进行封装，其中 DSAP 字节填写的是 B 返回的 SAP，SSAP 字节填写的是自己开辟的 SAP，然后发给 MAC 子层。
5. A 的 MAC 子层加上 MAC 地址和 LENGTH 字段之后，发送到数据链路上。
6. B 的 MAC 子层接收到该数据帧之后，提交给 LLC 子层，LLC 子层根据 DSAP 字段判断出该数据帧属于的连接。
7. B 根据该连接的类型进行相应的校验和确认，通过这些校验和确认后，才向上层发送。
8. 数据传输完毕之后，A 给 B 发送一个数据帧来告诉 B 拆除连接，通信结束。

2 二层交换

2.1 VLAN

2.1.1 VLAN 基本概念



- 一个 VLAN 中所有设备都是在同一广播域内，不同的 VLAN 为不同的广播域
- VLAN 之间互相隔离，广播不能跨越 VLAN 传播，因此不同 VLAN 之间的设备一般无法互访，不同 VLAN 间需通过三层设备实现相互通信
- 一个 VLAN 一般为一个逻辑子网，由被配置为此 VLAN 成员的设备组成
- VLAN 中成员多基于交换机的端口分配，划分 VLAN 就是对交换机的接口划分
- VLAN 工作于 OSI 参考模型的第二层
- VLAN 是二层交换机的一个非常根本的工作机制

2.1.2 VLAN 基本通信原理

为了提高处理效率，交换机内部的数据帧一律都带有 VLAN Tag，以统一方式处理。当一个数据帧进入交换机端口时，如果没有带 VLAN Tag，且该端口上配置了 PVID(Port VLAN ID)，那么，该数据帧就会被标记上端口的 PVID。如果数据帧已经带有 VLAN Tag，那么，即使端口已经配置了 PVID，交换机不会再给数据帧标记 VLAN Tag。

PVID 是“端口缺省 VLAN ID”的意思，即一个端口缺省属于的 VLAN。

由于端口类型不同，交换机对帧的处理过程也不同。下面根据不同的端口类型分别介绍。

1. Access 端口处理帧的过程

Access 端口处理 VLAN 帧的过程如下：

1) 收到一个二层帧。
2) 判断帧是否有 VLAN Tag。
   - 没有 Tag，则标记上 Access 端口的 PVID，进行下一步处理。
   - 有 Tag，则比较帧的 VLAN Tag 和端口的 PVID，两者一致则进行下一步处理；否则丢弃帧。
3) 二层交换机根据帧的目的 MAC 地址和 VLAN ID 查找 VLAN 配置信息，决定从哪个端口把帧发送出去。
4) 交换机根据查到的出接口发送数据帧。
   - 当数据帧从 Access 端口发出时，交换机先剥离帧的 VLAN Tag，然后再发送出去。
   - 当数据帧从 Trunk 端口发出时，直接发送帧。
   - 当数据帧从 Hybrid 端口发出时，交换机判断 VLAN 在本端口的属性是 Untag 还是 Tag。如果是 Untag，先剥离帧的 VLAN Tag，再发送；如果是 Tag，直接发送帧。

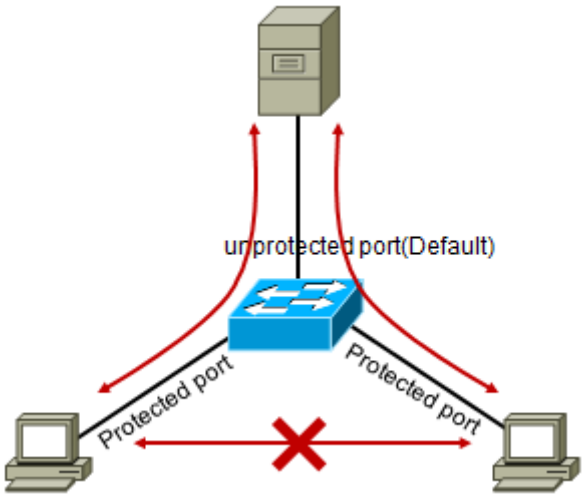
2. Trunk 端口处理帧的过程

Trunk 端口处理 VLAN 帧的过程如下：

1) 收到一个二层帧。

2) 判断帧是否有 VLAN Tag。
   - 没有 Tag，则标记上 Trunk 端口的 PVID，进行下一步处理。
   - 有 Tag，则判断该 Trunk 端口是否允许该 VLAN 帧进入。允许则进行下一步处理，否则丢弃帧。
3) 二层交换机根据帧的目的 MAC 地址和 VLAN ID ,查找 VLAN 配置信息 ,决定从哪个端口把帧发送出去。
4) 交换机根据查到的出接口发送数据帧。
   - 当数据帧从 Access 端口发出时，交换机先剥离帧的 VLAN Tag，然后再发送出去。
   - 当数据帧从 Trunk 端口发出时，直接发送帧。
   - 当数据帧从 Hybrid 端口发出时 ,交换机判断 VLAN 在本端口的属性是 Untag 还是 Tag。如果是 Untag，先剥离帧的 VLAN Tag，再发送;如果是 Tag，直接发送帧。

2.1.3 Protected port



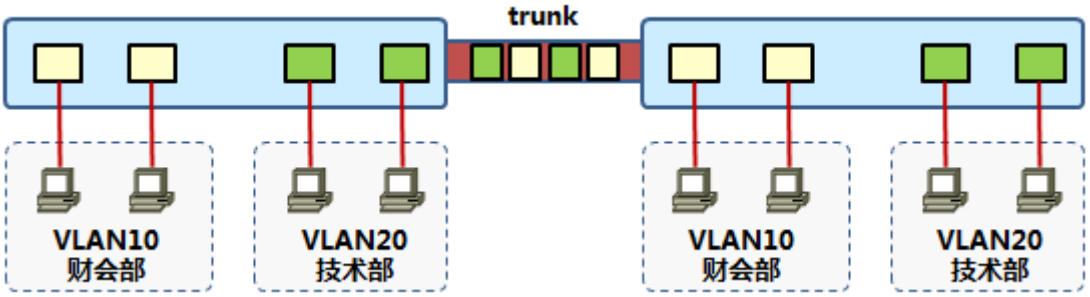
- Protected port 虽然同处一个 VLAN，但是彼此无法互相通信
- Protected port 只能与 unprotected port（默认）互相通信
- Protected port 特性无法跨交换机实现

配置非常简单：

```
Switch(config-if)# switchport protected
```

2.2 TRUNK

2.2.1 Trunk 概述



- 当一条链路，需要承载多 VLAN 信息的时候，需使用 trunk 来实现
- Trunk 两端的交换机需采用相同的干道协议
- 一般见于交换机之间或交换机与路由器、服务器之间

2.2.2 封装协议

ISL	802.1Q
CISCO私有	公有标准
采用封装的方式	采用tag的方式
在原始帧的基础上封装上新帧头及新的FCS	在原始的帧头中插入tag字段，去掉原有FCS，重新计算FCS

1. **baby giant frame 大于标准的 MTU1500 字节，但是小于 2000 字节**

   对于采用 ISL 封装的，MTU=1548 (下文有解释)

   对于 Dot1Q　　MTU=1522

2. **Vlan 范围和映射**

   ISL 支持的 vlan 编号是 1-1005（默认允许正常的），802.1q 是 1-4094（默认允许所有正常和扩展的）

   所以当穿过 802.1q 和 ISL 的干道的时候就需要映射。

   （1） 单台交换机上最多允许 8 个 802.1q 到 ISL vlan 的映射

   （2） 只能映射到 ethernet 的 vlan

   （3） 该被映射的 ethernet vlan 将被阻塞

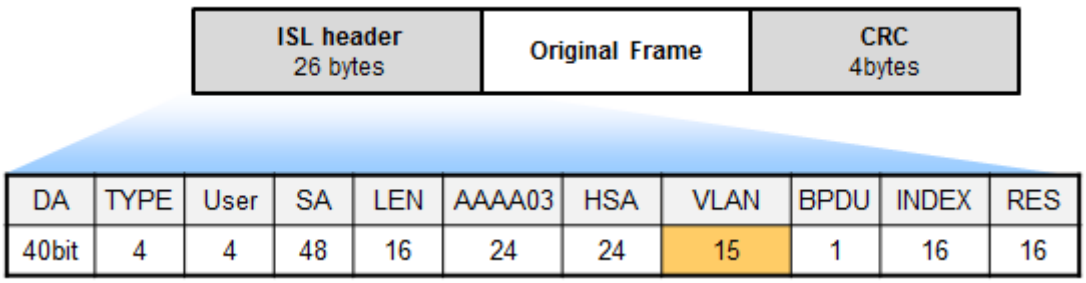
（4） 映射仅在本地有效

3. 链路聚集模式

- trunk 永久链路聚集模式，强制 trunk，发送 DTP 帧
- Nonegotiate 永久链路聚集模式，必须手动将邻居配为干道口，不发送 DTP 帧。一般用于对端设备不支持 DTP 的情况
- Desirable 主动尝试将链路成为干道（默认模式）发送 DTP 帧 ,如果邻接接口为 trunk、desirable、或 AUTO，那么此接口成为 Trunk。
- Auto 接口愿意成为 trunk ,如果邻接接口被设置为 trunk 或 desirable ,那么接口就成为 trunk
- Access 永久的 nontrunking 模式，并且与对端接口协商，使其成为 nontrunking 链路

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	不建议
Acces	Access	Access	不建议	Access

2.2.3 ISL



- CISCO 私有协议
- 支持 PVST
- 在原始的数据帧基础上封装上 ISL 头及新的 FCS
- 没有修改原始的数据帧，因此处理效率比 802.1Q 高
- VLAN 字段，15 个比特目前用了 10 个，那么最多支持 2 的 10 次方=1024 个 VLAN

- “原始以太网帧”最大是 1518 个字节 ,1500 的 IP MTU ,加上源目的 MAC 地址共 12 ,类型字段 2 个，CRC4，再加上 30 字节的 ISL 封装，就是 1548 字节了

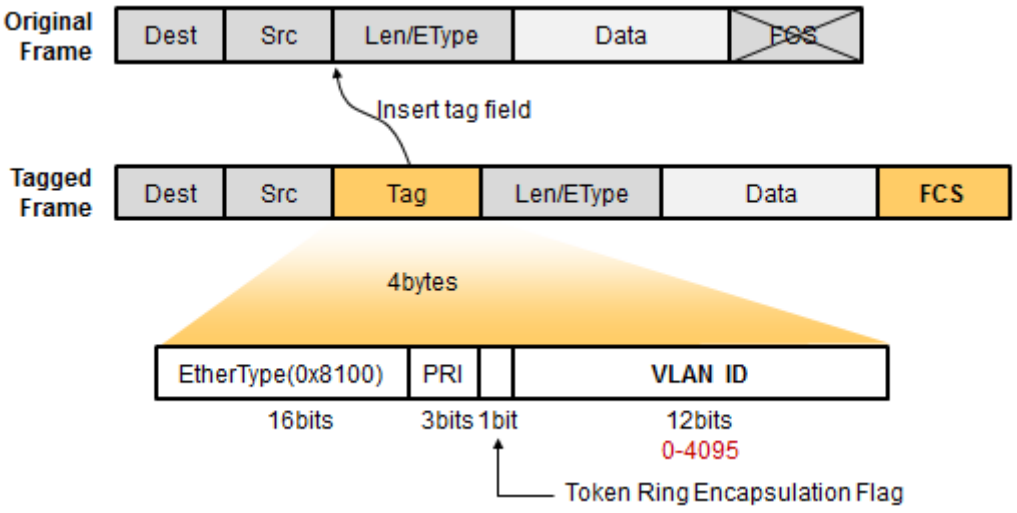
上图中几个字段（ISL 头）的描述如下：

- DA　　40bit 的组播地址用于标示这个 FRAME 是 ISL 的
- TYPE　　标示这个帧是什么类型的，如以太、令牌环等
- SA　　发送帧的原交换机 MAC
- AAAA03　　SNAP（固定值）
- VLAN　　15 个比特目前用了 10 个，那么最多支持 2 的 10 次方=1024 个 VLAN
- INDEX　　这个帧的对端交换机来源端口

所以 ISL 帧最大 1548bytes（1518+26+4）

2.2.4 Dot1q

1. **帧格式**



802.1Q Tag 包含 4 个字段，其含义如下：

- **EtherType**

  长度为 2 字节，表示帧类型。取值为 0x8100 时表示 802.1Q Tag 帧。如果不支持 802.1Q 的设备收到这样的帧，会将其丢弃。

- **PRI**

  Priority，长度为 3 比特，表示帧的优先级，取值范围为 0～7，值越大优先级越高。

用于当交换机阻塞时，优先发送优先级高的数据包。

- **CFI**

  Canonical Format Indicator，长度为 1 比特，表示 MAC 地址是否是经典格式。

  CFI 为 0 说明是经典格式，

  CFI 为 1 表示为非经典格式。用于区分以太网帧、FDDI（Fiber Distributed Digital Interface）帧和令牌环网帧。

  在以太网中，CFI 的值为 0。

- **VID**

  VLAN ID ,长度为 12 比特 ,表示该帧所属的 VLAN。在 VRP 中 ,可配置的 VLAN ID 取值范围为 1～4094。

2. 优缺点

缺点是破坏了原始以太帧以及重新计算 FCS，ISL 是直接封装头和尾。DOT1q 公用，ISL 私有

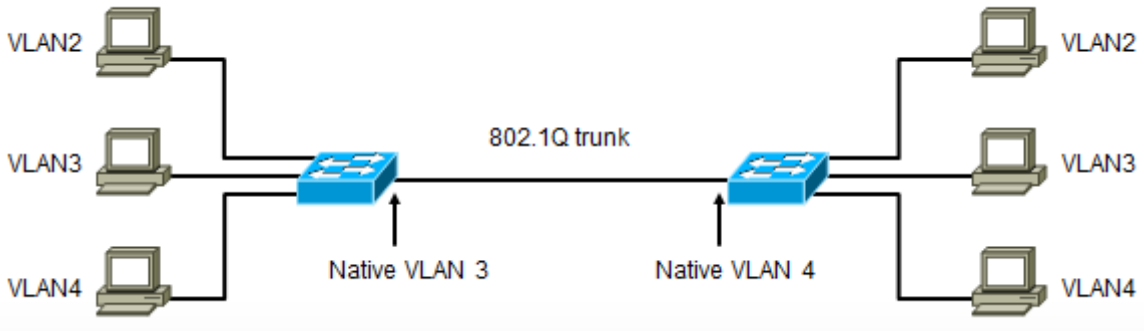
802.1Q 支持 4096 个 VLAN

最大帧：1518+4=1522

3. Native Vlan

在 802.1q 的 native vlan 是不打标签的 ,使用 Dot1q 的交换机把所有未被标记的 frame 转发到 native vlan 中，而 ISL 会对所有的数据帧，包括 native vlan 进行封装，因此如果收到没有封装的数据帧它会丢弃（ISL 没有 native VLAN 的概念）。

- Native VLAN 所属的帧在经过 trunk 时不打标签
- Native VLAN 在 Trunk 两端必须匹配，否则会出现 VLAN 流量互串
- 默认的 native vlan 是 vlan 1
- 建议将一个生僻的 VLAN 配置为 Native vlan



我们看上面这个图，两台交换机 trunk 两端 native vlan 不一样，会有什么问题？首先两端的 vlan2 通信肯定是没有问题的，但是 vlan3 和 vlan4 通信就有问题了，左边 vlan3 的用户发出来的数据帧从左交换机出去上 trunk，是不打标签的，但是这些数据帧到了右交换机，它会认为这些数据帧是属于 vlan4 的，这就出现问题了。

```
Switch(config-if)# switchport trunk native vlan ?
```

在 trunk 上设置 native vlan

```
Switch(config)# vlan dot1q tag native
```

上述命令将对 native vlan 也打标签

4. Vlan 范围

VLAN范围	作用
0，4095	保留，系统使用
1	Cisco默认vlan
2-1001	For Ethernet VLANs
1002-1005	Cisco默认为FDDI及TokenRing定义
1006-4094	只能为Ethernet使用，在一些特殊平台被保留使用

2.2.5 DTP

- Trunk 可以手工静态配置或者通过 DTP 进行协商
- DTP 使得交换机之间能够进行 trunk 协商

trunk	永久链路聚集模式，强制trunk，发送DTP帧
Nonegotiate	不发送DTP帧。一般用于对端设备不支持DTP的情况；需搭配手工指定的trunk或acess模式命令
Desirable	主动尝试将链路成为干道 （默认模式），发送DTP帧，如果邻接接口为trunk、desirable、或AUTO，那么此接口成为Trunk。
Auto	接口有意愿成为trunk，但是不会主动发送DTP。如果邻接接口被设置为trunk或desirable，那么接口就成为trunk，会被动响应DTP协商数据帧
Access	永久的nontrunking模式，并且与对端接口协商，使其成为nontrunking链路

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	不建议
Acces	Access	Access	不建议	Access

2.2.6 Trunk 配置

```
Switch(config-if)# switchport mode access
```

- 将接口设置为 access 模式

```
Switch(config-if)# switchport mode encapsulation {dot1q | ISL}
```

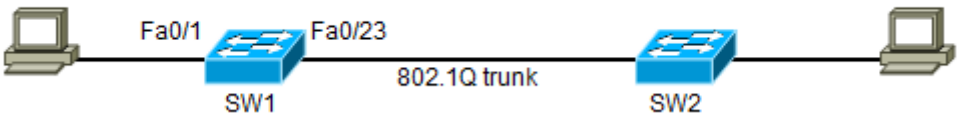
- 如果接口为 trunk，设置干道协议类型

```
Switch(config-if)# switchport mode dynamic {auto | desirable}
```

- 将接口设置为 DTP 动态协商，可选 auto 或 desirable

```
Switch(config-if)# switchport nonegotiate
```

- 将接口设置为 nonegotiate，不发送 DTP 帧，如果配置为非协商，那么就必须手工配置接口模式，为 access 或 trunk




```
SW1(config)# interface fast0/23
SW1(config-if)# switchport trunk encapsulation dot1q
SW1(config-if)# switchport mode trunk
SW1(config-if)# switchport native vlan 1
```

```
SW1(config-if)# switchport nonegotiate
SW1(config-if)# switchport trunk allowed vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in trunking mode
  add     add VLANs to the current list
  all     all VLANs
  except  all VLANs except the following
  none    no VLANs
  remove  remove VLANs from the current list
```

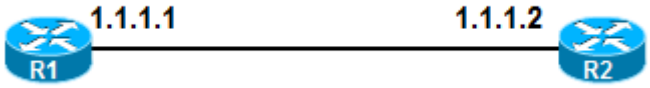
上图中 ,两端 PC 都属于 vlan10d 的话 ,如果 SW1 的 fa0/23 口将 vlan10remove 掉 ,那么 PC 肯定就无法通信了。

2.2.7 MTU 问题

MTU 就是最大传输单元，不同的系统对于 MTU 的设定和理解是不同的。

CISCO IOS 上，interface x 接口模式下：

- **MTU ?** 指的是二层的 MTU，这是接口 MTU，指的是不包含二层帧头的、Payload 的 MTU，这个 MTU 值一般是不能手工修改的，默认是 1500 字节。如此一来 CISCO 路由器支持的二层数据帧最大值就是 1500 的 payload 加上二层帧头及二层 FCS：目的 mac**6** 字节+源 mac**6** 字节+类型字段 **2** 字节+FCS**4** 字节，所以总的就是 1518 字节。
- **Ip mtu ?** 指的是三层的 MTU ,这个值可以手工修改 ,但是最大值必须小于接口的二层 MTU 值也就是 1500。这个 MTU 指的是三层 IP 包的总大小，如果接口发出的包大于这个接口的 ip mtu，那么这个 IP 包将被分片



做个测试：上图中，R1 的 fa0/0 口 ip mtu 为 1500，

我们去 ping 1.1.1.2 repeat 1 size 1500，我们会发现 R1 直接将一个 ICMP 包发出去了，没有分片，报文如下：

```
Internet Protocol Version 4, Src: 1.1.1.1 (1.1.1.1), Dst: 1.1.1.2 (1.1.1.2)
  Version: 4
  Header length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-E
  Total Length: 1500
  Identification: 0x0000 (0)
⊞ Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
⊞ Header checksum: 0xb21c [correct]
  Source: 1.1.1.1 (1.1.1.1)
  Destination: 1.1.1.2 (1.1.1.2)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xeb07 [correct]
  Identifier (BE): 0 (0x0000)
  Identifier (LE): 0 (0x0000)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
  [Response In: 5]
⊞ Data (1472 bytes)
```

从报文中我们可以看到，这个 IP 包的大小为 1500 字节。

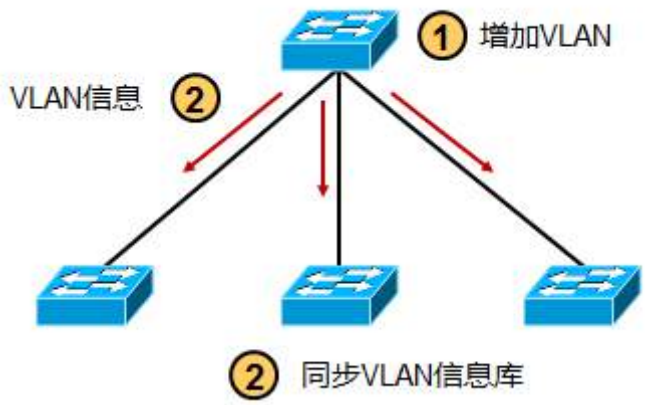
其中 IP 报头 20 字节，ICMP 报头 8 字节，ICMP data 荷载 1472 字节，刚好 1500 字节。

因此在 CISCO IOS 设备上，ping 后面跟着的 size 指的就是发出去的 IP 包整个的大小。

而在 windows PC 的 cmd 下，ping 后跟的包大小就是 ICMP data 大小，ping –l 1472，产生的包就是 1500 字节

还是上面的例子，如果我们在 R1 上，ping 1.1.1.2 repeat 1 size 1501，这个时候，R1 由于产生的这个 IP 包大于 mtu 1500，因此会被分片，然后在 R2 上，这两个分片被重组。

2.3 VTP



- 简化大型园区网中 VLAN 信息库同步的问题（同一个 VTP 管理域中）

- 只同步 VLAN 信息
- 需要交换机之间的 trunk 链路支持

1. VTP mode

Server	Client	Transparent
• 可创建vlan • 可修改vlan • 可删除vlan • 发送/转发信息宣告 • 同步 • VLAN信息存储于NVRAM • Catalyst交换机默认是server模式	• 不能创建、修改、删除VLAN • 发送/转发信息宣告 • 同步 • VLAN信息不会存贮于NVRAM	• 可创建vlan • 可修改vlan • 可删除vlan • 转发信息宣告 • 不同步 不始发 • VLAN信息存贮于NVRAM

Transparent 模式的 VTP 配置修订号始终为 0

2. VTP 的运作

- VTP 协议通过组播地址 0100-0CCC-CCCC 在 Trunk 链路上发送 VTP 通告；
- VTP Server 和 clients 通过最高的修订号来同步数据库；
- VTP 协议每隔 5 分钟发送一次 VTP 通告或者有变化时发生；

3. VTP 的配置

```
Switch(config)# vtp domain cisco
```

- **配置 VTP 域名**

```
Switch(config)# vtp mode {server | client | transparent}
```

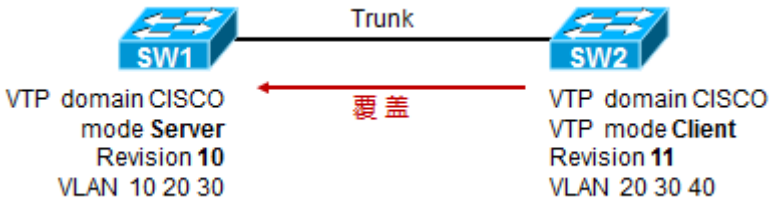
- **配置本机 VTP 模式**

```
Switch(config)# vtp password x
```

- （可选）配置密码

```
Switch# show vtp password
```

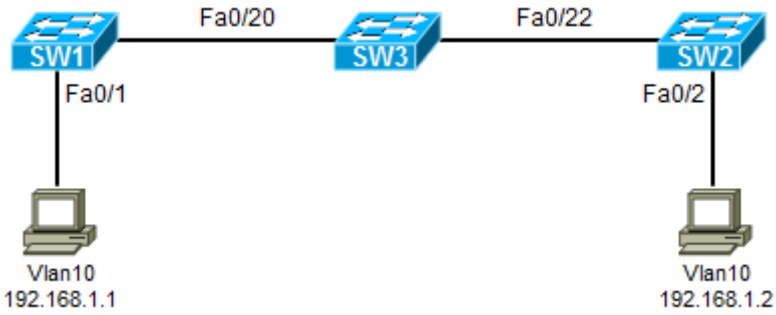
4. VTP 的几个问题



如果 VTP client 的配置修订号比 Server 的高，那么 client 也是能够将 server 的 vlan 信息覆盖掉的。

上图中，server 及 client 的配置修订号相同，但是 vlan 信息则不同，这时候就会报错，提示 md5 digest checksum mismatch

5. VTP pruning



1) 测试目的：VLAN 信息传递

SW1 及 SW2 配置为 VTP mode client；SW3 配置为 VTP mode Server

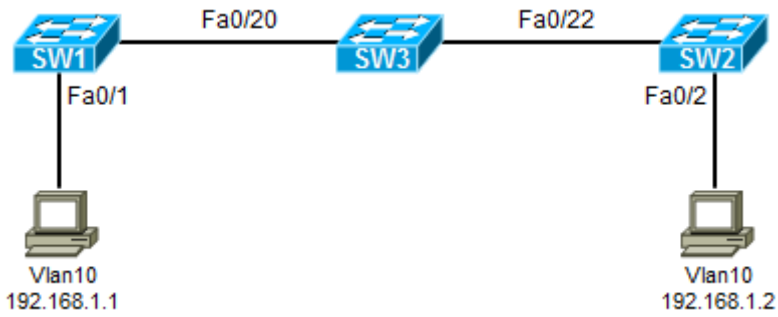
在 SW3 上创建 10、20、30 三个 VLAN

SW1 及 SW2 都能学习到，三岁小孩都能做，就不多说了

```
SW3#sh vtp status
VTP Version capable             : 1 to 3
VTP version running             : 1
VTP Domain Name                 : ccnp
VTP Pruning Mode                : Disabled      !! 默认 VTP prunnig 是关闭的
VTP Traps Generation            : Disabled
```

```
Device ID                          : 000a.8a07.8280
Configuration last modified by 0.0.0.0 at 3-5-93 00:11:48
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                 : Server
Maximum VLANs supported locally    : 1005
Number of existing VLANs           : 8
Configuration Revision             : 15
MD5 digest                         : 0x49 0x3C 0x1F 0x79 0x15 0x00 0xC7 0xAE
                                     0x0E 0xDC 0xDD 0xEF 0x93 0xA5 0xB6 0x26
```

2) 测试目的：trunk allowed vlan



继续吧，将 SW1 及 SW2 的连接 PC 的接口划入 vlan10

完成后两个 PC 就能够互相通信了。

这时候我们在 SW1 上看一下 trunk 的状态：

```
SW1#sh int trunk
Port        Mode         Encapsulation  Status        Native vlan
Fa0/20      on           802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/20      1-4094

Port        Vlans allowed and active in management domain
Fa0/20      1,10,20,30
```

Port Vlans in spanning tree forwarding state and not pruned

Fa0/20 1,10,20,30 !!在关闭 VTP prunnig 的情况下 ,trunk 默认放行所有 VLAN 的流量 ,这在 SW2 及 SW3 上情况一样

接下去我们在 SW1 的 Fa0/20 接口上，做 switchport trunk allowed vlan remove 10，将 vlan10 的流量修剪掉，这时候 PC 之间就无法 ping 通了，再去 R1 上看一下：

SW1#sh int trunk

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1

Port Vlans allowed on trunk

Fa0/20 1-9,11-4094

Port Vlans allowed and active in management domain

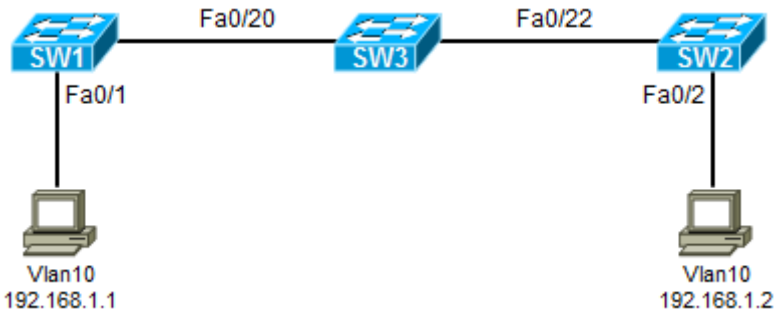
Fa0/20 1,20,30

Port Vlans in spanning tree forwarding state and not pruned

Fa0/20 1,20,30

注意，此时此刻虽然 SW1 在 fa0/20 口上将 vlan10 的流量修剪了，但是 SW3 的 fa0/20 口确实依然放行该流量的，不过不管怎样，PC 之间是无法互访了，这里只是做个演示，知道一下 allowed vlan 的作用。当然，这是手工修剪的方式。

3) 测试目的：VTP pruning



去掉步骤 2 中配置在 SW1 上的命令，还原实验环境。

接下去我们在 SW3，也就是 VTP 的 server 上开启 vtp pruning。

SW3#sh vtp st

VTP Version capable : 1 to 3

VTP version running : 1

VTP Domain Name : ccnp

VTP Pruning Mode : Enabled

VTP Traps Generation : Disabled

Device ID : 000a.8a07.8280

Configuration last modified by 0.0.0.0 at 3-5-93 02:01:49

Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:

VTP Operating Mode : Server

Maximum VLANs supported locally : 1005

Number of existing VLANs : 8

Configuration Revision : 17

MD5 digest : 0x73 0x52 0x60 0xE7 0x4D 0xA5 0xC7 0x4F

0xCA 0x3D 0x6F 0x1D 0x3F 0x23 0x03 0xBB

在 VTP server mode 的 SW3 上开启 VTP pruning 后，Client 的 SW1 及 SW2 都会学习到并且也开启自己的 VTP pruning。

如此一来，三台交换机都会进行 VTP 报文的交互，告知自己本地存在接入用户的 VLAN。对于本地没有用户的 VLAN，将会被自动修剪掉，我们看一下：

SW1#show int trunk

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1

Port	Vlans allowed on trunk
Fa0/20	1-4094

Port	Vlans allowed and active in management domain
Fa0/20	1,10,20,30

Port	Vlans in spanning tree forwarding state and not pruned
Fa0/20	**1,10**

!! 在 R1 上，Fa0/20 口修剪得只剩下了 vlan 1 和 10，因为本地只有 vlan1 和 vlan10

SW2、SW3 也是类似的情况 ;注意 ,此刻我们是没有在 trunk 口上做任何静态配置的 allowed vlan 配置，

这些都是 VTP pruning 自动协商完成的。

接下去，我们在 SW2 上，将 vlan10 的端口移除，可直接在 fa0/2 上 no switchport access vlan
如此一来 SW2 上就没有 vlan10 的用户了，sw2 会泛洪这一消息以便其他交换机知道。
SW3 收到这一消息后，知道 SW2 不再有 vlan10 的用户，也就不再需要 vlan10 的流量了，于是在自己的 FA0/22 口上将 vlan10 修剪掉：

```
SW3#sh int tru
Port        Mode             Encapsulation  Status        Native vlan
Fa0/20      on               802.1q         trunking      1
Fa0/22      desirable        n-isl          trunking      1

Port        Vlans allowed on trunk
Fa0/20      1-4094
Fa0/22      1-4094

Port        Vlans allowed and active in management domain
Fa0/20      1,10,20,30
Fa0/22      1,10,20,30

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/20      1,10
Fa0/22      1                    !! vlan10 被修剪
```

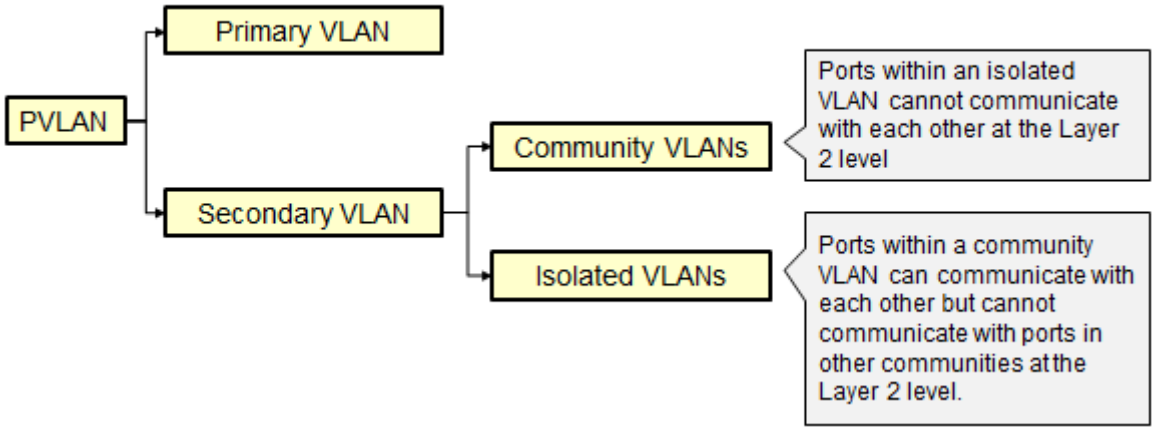
最后注意：

- Vtp pruning 只能在 server mode 上开启
- Server mode 上开启后，client 都会自动开启 pruning
- Vtp pruning 默认是关闭的

2.4 私有 VLAN（PVLAN）

1. **私有 VLAN 的概念**
   - 将一个 VLAN 划分成几个单独的 VLAN，这些 VLAN 都使用同一个 IP 网段。

- 可以提高安全性，降低子网数目，提高 IP 利用率
- 尽管网络设备处于同一个子网中，但是他们属于不同的 pVlan，pVlans 之间的通信还是必须通过默认网关来实现。
- 每个 pVLAN 包括一个**主 VLAN** 以及多个**辅助 VLAN**。所有的辅助 VLAN 都映射到主 VLAN。
- **辅助 VLAN 分为团体 VLAN 和隔离 VLAN。**
- 相同团体 VLAN 能够互相通信，但是团体 VLAN 之间必须通过设置 SVI 或者路由器接口才能通信。
- 相同隔离 VLAN 内部以及隔离 VLAN 之间都是不能够互相通信的，只能与混杂接口通信。
- **一个主 VLAN 只能有一个 Isolate vlan**
- 混杂端口能够与 pVLAN 中的任何设备通信，不管对方是处于主 VLAN，还是辅助 VLAN.



2. PVLAN 端口类型

Isolated

Communicates with only promiscuous ports

Promiscuous

Communicates with all other ports

Community

Communicates with other members of community and all promiscuous ports

3. PVLAN 注意事项

具体支持的设备，见 CISCO 官方文档。

关于 PVLAN 的配置：

http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.2/31sga/configuration/guide/pvlans.html

pVLAN 必须配置在透明模式的交换机上。而且要求 VTP 版本 1 或 2。

禁止将第三层的 VLAN 接口配置为辅助 VLAN。

Etherchannel 端口不支持 pvlan

4. PVLAN 的配置

创建主 VLAN：

```
Vlan 100
  Private-vlan primary
```

创建辅助 VLAN：

```
Vlan 101
  Private-vlan community
Vlan 102
  Private-vlan ioslate
```

配置主 VLAN，将二层辅助 VLAN 关联到主 VLAN

```
Vlan 100
  Private-vlan association 101,102
```

将辅助 VLAN 映射到主 VLAN 的 SVI 接口，从而允许 pVLAN 入口流量的三层交换。

```
Interface vlan 100
 Private-vlan mapping add 101,102
```

配置接口：

- **如果将接口配置为主机接口**

```
interface f0/1
   Switchport mode private-vlan host
   Switchport private-vlan host-association 100 101     //关联主 VLAN 和辅助 VLAN 到接口
Interface f0/2
   Switchport mode private-vlan host
   Switchport private-vlan host-association 100 102
```

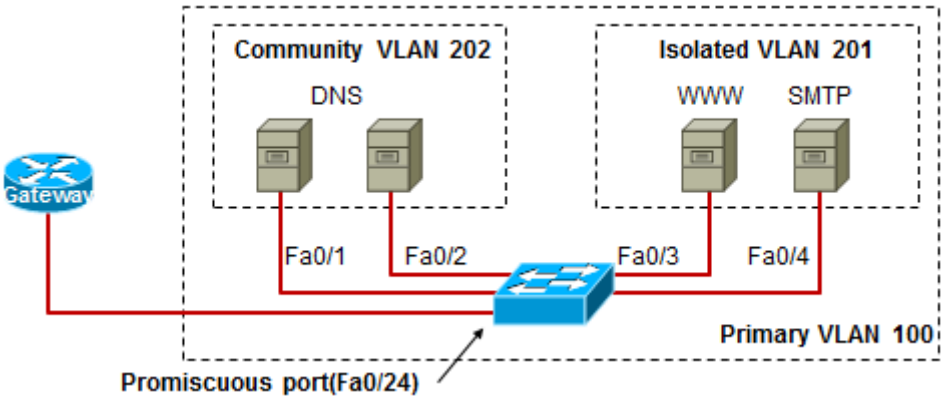
- **如果将接口配置为混杂接口**

```
Interface f0/3
   Switchport mode private-vlan promiscuous
   Switchport private-vlan mapping add 100 101          //将端口映射到 pVLAN
```

查看及验证：

```
show pvlan mapping
```

5. PVLAN 的配置示例



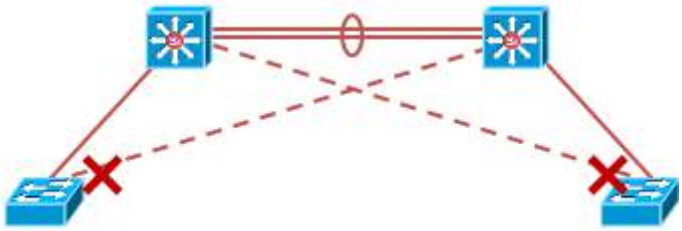
- DNS、WWW、SMTP 服务器同属一个子网
- 两台 DNS 服务器同属一个 community VLAN，彼此之间能够互相通信
- WWW 及 SMTP 属于 Isalated VLAN，因此彼此之间无法互相访问；WWW 及 SMTP 服务器都只能与混杂端口，也就是与路由器通信

```
sw(config)#vtp transparent
sw(config)#vlan 201
sw(config-vlan)#private-vlan isolated
sw(config)#vlan 202
sw(config-vlan)#private-vlan community
sw(config)#vlan 100
sw(config-vlan)#private-vlan primary
sw(config-vlan)#private-vlan association 201,202
!
sw(config)#interface fa0/24
sw(config-if)#switchport mode private-vlan promiscuous
sw(config-if)#switchport private-vlan mapping 100 201,202
sw(config)#interface range fa 0/1-2
sw(config-if)#switchport mode private-vlan host
sw(config-if)#switchport private-vlan host-association 100 202
sw(config)#interface range fa 0/3-4
sw(config-if)#switchport mode private-vlan host
```

```
sw(config-if)#switchport private-vlan host-association 100 201
```

3 Spanning-tree

3.1 协议概述



- 如果接入层交换机单链路上联，则存在单链路故障另一个问题的单点故障，如果任意一个汇聚设备宕机，将直接导致下联的接入网络挂掉
- 与是接入层交换机采用双链路上联到两台汇聚设备，构成一个物理链路冗余的二层环境，解决了单链路故障问题
- 但是二层环境存在环路
- 生成树用于解决这个问题
- 通过生成树协议，在逻辑上将特定端口进行 Block，从而实现物理上存在冗余环境，而二层上又阻止环路的产生
- 当拓扑发生变更的时候，生成树协议能够探测到这些变化，并且及时自动的调整接口状态，从而适应网络拓扑的变化，实现链路冗余

3.2 协议基础

1. 生成树协议

PROTOCOL			Describe
STP (PVST)	Public	802.1 D	
PVST+	Cisco Private		Portfast,uplinkfast, backbonefast
RSTP	Public	802.1W	集成了 pvst 的功能并公有化
MST	Public	802.1S	

2. 参数

- **网桥 ID （8 字节）= 网桥优先级(2 字节)+网桥 MAC (6 字节)**

  缺省优先级 32768,范围 0-65535 （前 4bit 表示优先级,后 8bit 作为 system id,为协议扩展用，越小越好,4096 的倍数）

- **端口 ID （2 字节）= 端口优先级(1 字节)+端口 ID(1 字节)**

  缺省优先级 128，范围 0-255，越小越好

  Catos 交换机端口 ID 中优先级默认 32（优先级 6 个 bit），IOS 交换机默认 128（优先级字段 8 个比特）

- **根路径开销**

  本交换机到达根交换机路径的总开销

  越小越好，与接口带宽有关

Link Speed	Cost (New IEEE Specification)	Cost (Old IEEE Specification)
10 Gb/s	2	1
1 Gb/s	4	1
100 Mb/s	19	10
10 Mb/s	100	100

3.3 802.1D

3.3.1 BPDU 报文

BPDU 有两种类型：配置 BPDU 及 TCN

1. **配置 BPDU**

在网络刚开始运行的阶段，所有交换机都会从所有端口发送 BPDU，大家都认为自己是 root，随着 BPDU 泛洪和收集，根据 BPDU 中所含信息，大家 PK 出来个结果，root 被选举出来了。在此之后由 Root 以默认 2S 为周期发送 BPDU，所有的非 root 交换机从自己的根端口收到 BPDU，再从自己的指定端口产生 bpdu 发出去。这就有点像我们从 root 倒一盆水下来，水顺着这颗无环的树从上往下不断的流。另外，被 block 的非指定端口会源源不断的收到链路上的 bpdu 并一直侦听，当其在一定时间内没有再收到 bpdu，则认为链路出现了故障，开始进入新的收敛阶段。

"When a switch receives a configuration BPDU that contains superior information (lower bridge ID, lower path cost, and so forth), it stores the information for that port. If this BPDU is received on the root port of the

switch, the switch also forwards it with an updated message to all attached LANs for which it is the designated switch.

If a switch receives a configuration BPDU that contains inferior information to that currently stored for that port, it discards the BPDU. If the switch is a designated switch for the LAN from which the inferior BPDU was received, it sends that LAN a BPDU containing the up-to-date information stored for that port. In this way, inferior information is discarded, and superior information is propagated on the network."

字节	字段	描述
2	协议	代表上层协议（BPDU），该值总为 0
1	版本	（802.1D 的总为 0）
1	TYPE	"配置 BPDU"为 0x00、"TCN BPDU"为 0x80
1	标志	LSB 最低有效位表示 TC 标志；MSB 最高有效位表示 TCA 标志
8	**根 ID**	根网桥的桥 ID
4	**路径开销**	到达根桥的 STP cost
8	**网桥 ID**	BPDU 发送桥的 ID
2	**端口 ID**	BPDU 发送网桥的端口 ID（优先级+端口号）
2	消息寿命 Message age	从根网桥发出 BPDU 之后的秒数（这个 BPDU 存活了多长时间了），每经过一个网桥都减 1，所以它本质上是到达根桥的跳数。
2	最大寿命 Max age	当一段时间未收到任何 BPDU，生存期到达 MAX age 时，网桥认为该端口连接的链路发生故障。也可以理解为这个 BPDU 的最大寿命，默认 20S
2	HELLO 时间	根网桥连续发送的 BPDU 之间的时间间隔。默认 2S
2	转发延迟	在监听和学习状态所停留的时间间隔。默认 15S

Catos 交换机端口 ID 中优先级默认 32（优先级 6 个 bit），IOS 交换机默认 128（优先级字段 8 个比特）

2. TCN BPDU

在网络拓扑变化的时候产生。

TYPE 字段　为 0x80

Flag 字段　　LSB=TC　　MSB=TCA

当网络拓扑发生变化的时候，最先意识到变化的交换机会从根端口发送 TCN（TYPE 字段=0x80）到上一层交换机（朝向 ROOT 的方向），一直到根交换机，上一层交换机除了向其上一层交换机发送 TC 外，也回一个 TCA（MSB 置位）的确认信息给前一个交换机。当根接收到 TC 后发送 TCA 回最开始的发送交换机并向所有网桥发送 TC（LSA 置位）。交换机收到 ROOT 发出来的 TC 后，会将 MAC 地址表的老化时间减少为 15s（转发延迟），这个 TC 一直会持续 35s（20+15）

3.3.2 STP 的运行

1. **STP 采用四部来解决二层环路：**
   1) 在一个交换网络中选举一个 root bridge
   2) 在每个非根桥上选举一个根端口
   3) 为每个 Segment 选举一个指定端口
   4) 阻塞非指定端口

2. **比较原则**

STP 需要网络设备相互交换消息来检测桥接环路，该消息称为**网桥协议数据单元 BPDU,** 阻塞端口也会不断收到 BPDU，以保证故障发生的时候，仍然可以计算出一棵新的 STP。要理解 STP 的工作过程，非常重要的一点是要理解 BPDU 中各字段的含义，因为这些都是 STP 赖以工作的根本。

生成树构造一个无环路拓扑时，总是使用相同的 4 步来判定：

- **最低根桥 ID**
- **到根桥的最低路径成本**
- **最低的发送者网桥 ID**
- **最低的发送者端口 ID**

网桥使用这 4 步来保存各个端口接收到的“最佳”的 BPDU 的一个副本。每个 BPDU 到达时，都会按照这个 4 步判决步骤来进行检查，看它是否该端口保存的 BPDU 更优，如果是，则会更新。

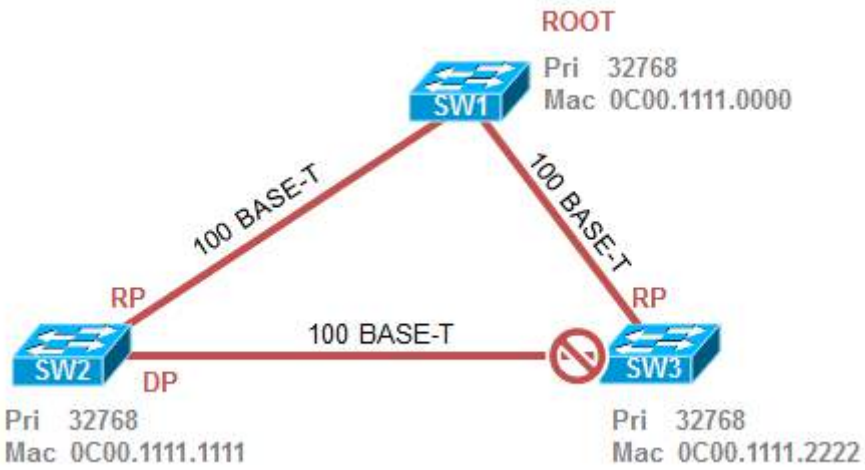
当一个网桥开始活动时， 他的每个端口都是每 2s 发送一个 BPDU，而当一个端口收到一个比现在发送的更优的 BPDU，则本地端口会停止发送，如果在一段时间内 （却省为 20s）后他不再收到来自邻居的更优 BPDU，则他将再次发送。因此对于 802.1D 来说，根桥会不停的向所有接口发送 BPDU，而非根桥会从自己的根端口收到 BPDU，并且向自己的指定端口去发 BPDU，非指定端口是不会发送 BPDU 的，只会侦听。

3. **注意事项：**
   - 根桥的角色是可抢占的
   - 桥 ID 中的 MAC，是交换机的背板 MAC，端口 ID 中的 MAC 是交换机的端口 MAC，查看交换机上的所有 mac 可用

     show interface | include bia
   - 二层交换机的端口 mac 就在这用了：)

3.3.3 案例分析

- **CASE1：**

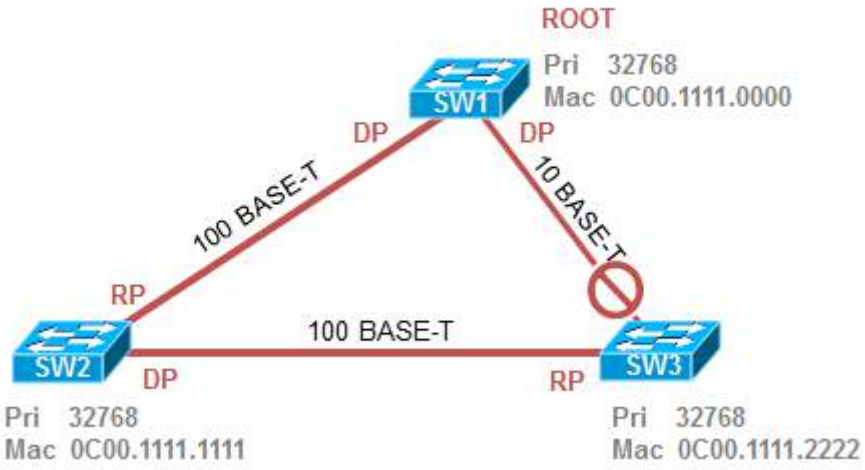


首先选 root，交换机 MAC 最小的 SW1 胜出，成为 root

其次在非根桥 SW2 及 SW3 上选择一个 root port，如图所示，因为这些端口到达 SW1 开销最小

最后在每个 segment 上选择一个指定端口 DP，sw1 是根桥，所有接口都是 DP，最后比较 SW2、SW3 之间直连链路的两个接口。SW2 会从接口上收到 SW3 发来的 BPDU，SW3 也会收到 SW2 发来的 BPDU 各自比较自己和收到 BPDU 中的桥 ID（也就是所谓的比较发送桥 ID），最终 SW2 由于 MAC 地址较小胜出。因此 Block 掉 SW3 的那个接口。

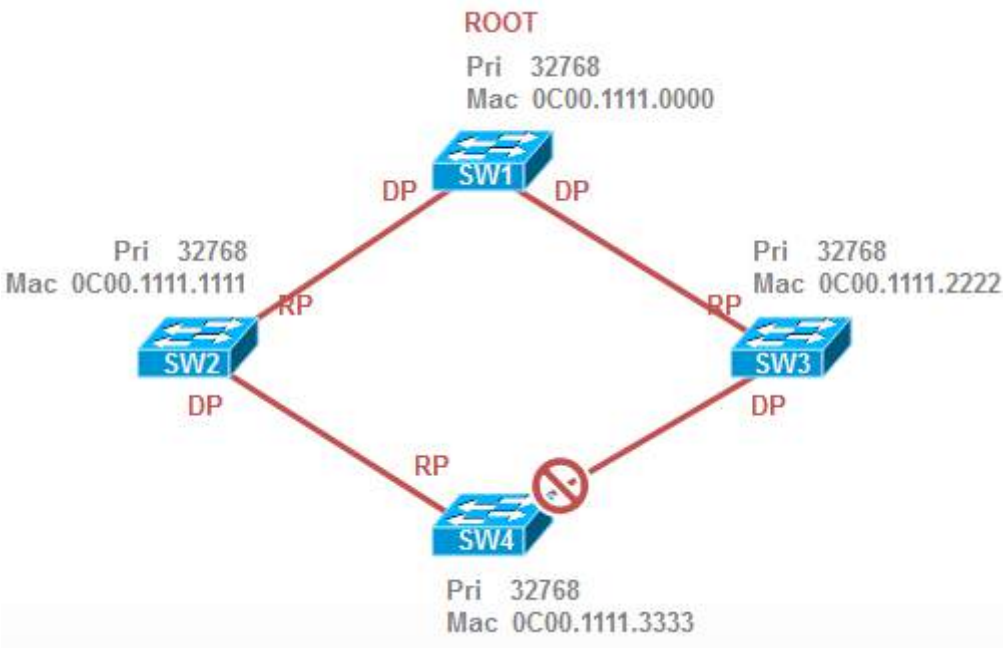
- **CASE2：**



根桥的选举就不说了，接下去看 RP，SW2 两个接口，都会收到 BPDU，而上联到 root 的接口收到的 BPDU

中到 root 的开销最小，所以上联口为 RP。 SW3 却不一样，由于连接到 root 的接口带宽仅为 10M，因此连接 SW2 的接口胜出为 RP。最后选择 DP。简单不赘述了。

- **CASE3：**

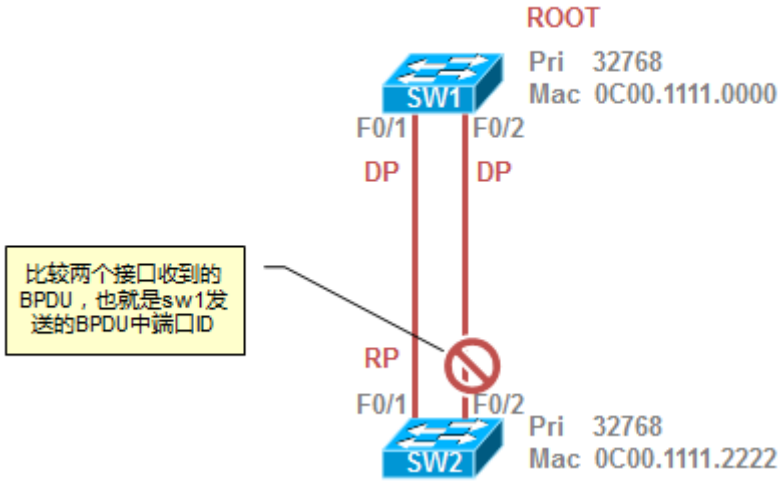


根桥的选举就不说了

根端口的选举，SW2 和 SW3 都比较简单。关键看 SW4，有两个接口，这两个接口都会收到 BPDU，首先看到 root 的开销，由于这两个 BPDU 的到 root 开销相等，加上 SW4 这两个接口的带宽也相等，因此这一步比较不出来，接着看发送者桥 ID，也就是收到的这两份 BPDU 的 BridgeID，比较后发现 SW2 的 MAC 更小（比 SW3 小），因此 SW4 上连接 SW2 的接口胜出，成为 RP。

最后看 DP，也比较简单如图所示。我们拿 SW3 及 SW4 之间的 segment 举例，由于 SW3 及 SW4 都会产生 BPDU，而 SW3 发给 SW4 的 BPDU 明显要优于 SW4 自己产生的从这个接口发送的 BPDU，因此最终 SW4 的接口胜出。

- **CASE4：**



根桥的选举就不说了。

接下去看 RP，SW2 有两个接口，这两个接口都会收到 BPDU，都来自 SW1，因此这两个 BPDU 首先到 root 的开销都是 0，相等；其次发送者的桥 ID，由于是来自同一台交换机，因此也相等，也比较不出来；再比较这两个 BPDU 的端口 ID，也就是 SW1 上这两个接口的端口 ID，我们假设两个接口优先级相等，那么就比这两端口的接口 ID，最终选出 SW2 上的根端口 F0/1。

注意，这时候如果试图在 SW2 上将 F0/2 的接口优先级改小，也是没用的，因为看的是发送者的端口 ID。所以如果在 SW1 上，将 F0/2 的端口优先级调小。那么 SW2 上，F0/2 就会胜出成为根端口。

3.3.4 STP 端口状态

1. **写在前面的话：**

   由于网络设备存在固有的滞后，所以交换网络中也就存在传播延迟。基于上述原因，拓扑变更就可能发生在交换网络中的不同时间和不同的网段。如果 2 层接口直接从生成树的 block 状态切换到转发状态，就可能会出现暂时的数据环路。为了缓解这种问题，在开始转发数据帧之前，端口应当等待新的拓扑信息传播到整个交换网络中。

2. **计时器**

   阻塞到转发状态通常要 30-50s（默认 50S，即 20+15+15），这个时间也可以通过配置生成树计时器来调整。

   Hello 时间　　根网桥发送配置 BPDU 的时间间隔　　缺省为 2 s

   转发延迟时间　　侦听到学习状态， 或者学习状态转换到转发状态所需要的时间　　缺省为 15s。

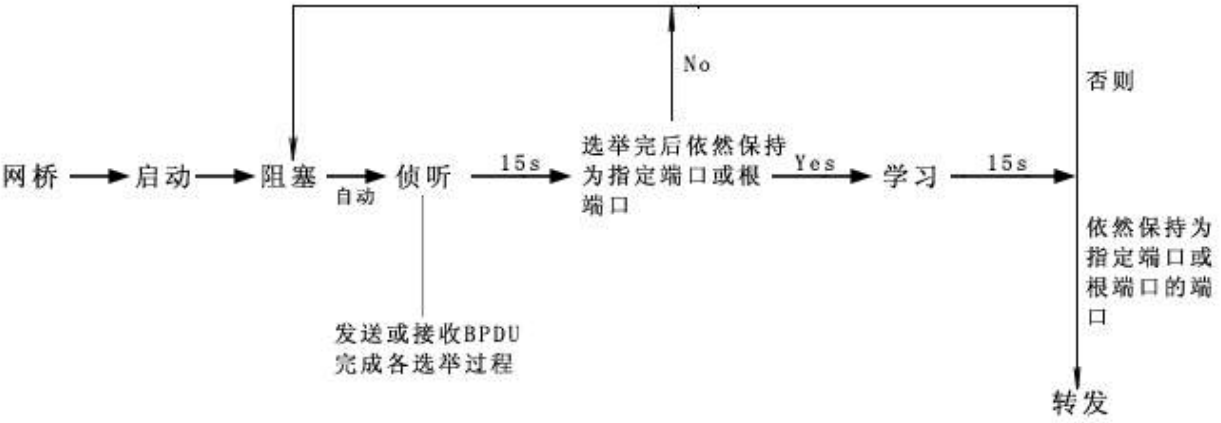
   最大存活期　　在丢弃 BPDU 之前 ,网桥用来存储 BPDU 的时间 ,缺省为 20s。如果连续收不到 10 个 bpdu（20s 的时间），开始进入 listening 状态

网络中的生成树拓扑依附于根网桥的计时器，root 将 BPDU 中的计时器传递给第二层的所有交换机。

3. STP 各状态如下：

Disable	不收发任何报文
Blocking	不接收也不转发帧，接收但不发送BPDU，不学习MAC地址
Listening	不接收也不转发帧，接收并且发送BPDU，不学习MAC地址
Learning	不接收也不转发帧，接收并且发送BPDU，学习MAC地址
Forwarding	接收并转发帧，接收并且发送BPDU，学习MAC地址

4. STP 端口状态转换过程



3.3.5 STP 拓扑变更

1. TCN BPDU 概述

当网络拓扑出现变更的时候，最先意识到变化的交换机将发送 TCN BPDU。

在发生以下时间时，交换机发送 TCN：

- 对于处于转发和监听状态的接口，过渡到 Block 状态（链路故障的情况）
- 端口进入转发状态，并且网桥已经拥有指定端口
- 非 root 网桥在它的指定端口收到 TCN

2. TCN BPDU

TCN BPDU 包含 3 个字段，它与配置 BPDU 除了 type 字段之外的前 3 个字段完全相同。

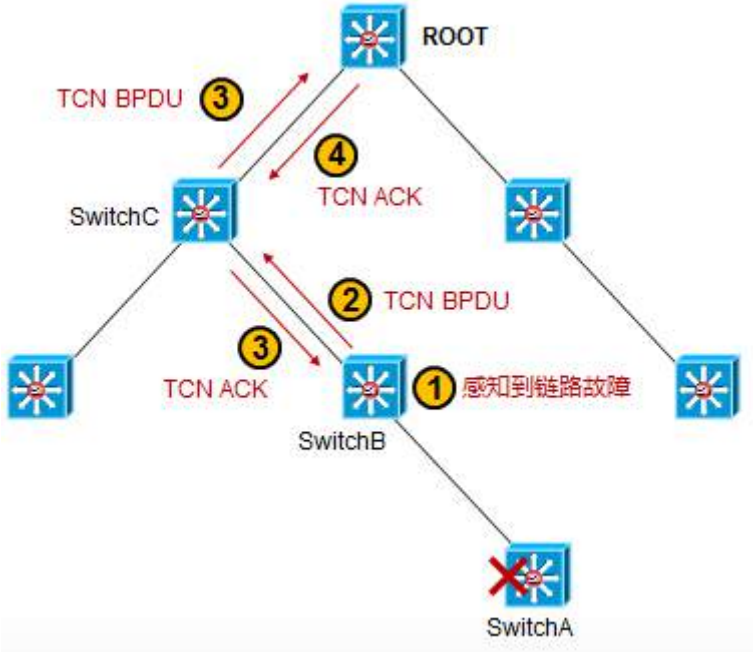
3. 拓扑变更过程

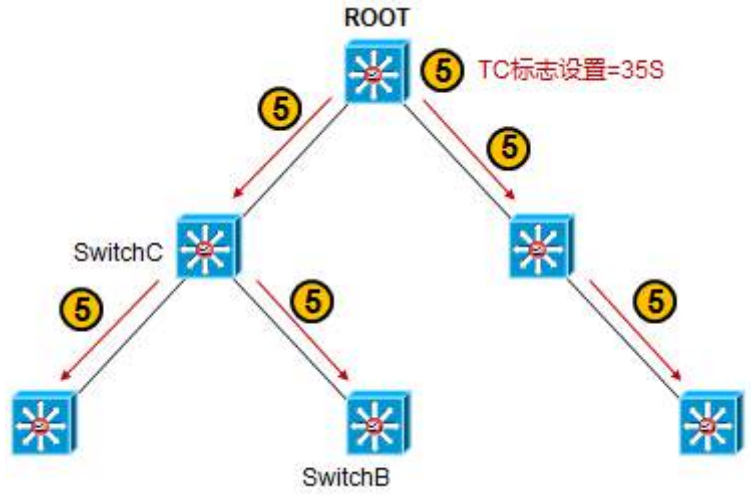
最先意识到拓扑变更的交换机发送 TCN ,指定网桥接收 TCN 并且通过立即回送一个 TCA 被置位的正常 BPDU 来确认，在该网桥确认这个 TCN 之前，负责通知拓扑变更的网桥将持续发送 TCN BPDU。

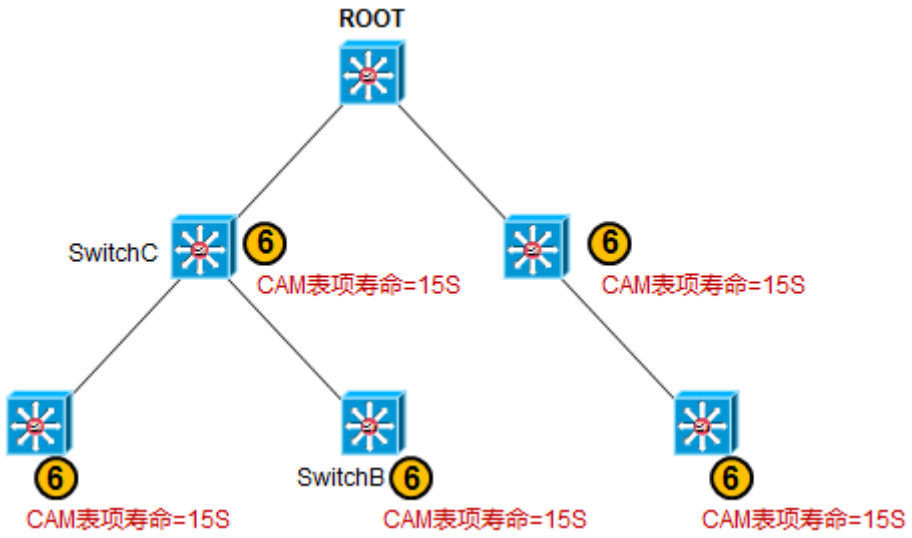
接下去该指定网桥将为自己的根端口产生另外的 TCN，并且这个过程一路发到 root 为止。

一旦网桥意识到网络中发生拓扑变更的情况，它将发送 TC 被置位的配置 BPDU，网络中的每台交换机都将传递这些被置位的 BPDU，进而便于每个单独的网桥都意识到拓扑变更的情况。

4. 拓扑变更过程 范例







1) SwitchA 挂掉
2) SwitchB 最先检测到拓扑变化，于是产生 TCN BPDU 并从根端口发送出去（因为根端口是朝着根桥的方向），B 将连续发送 TCN BPDU 直到指定交换机 C 发送 TCN ACK 进行确认
3) SwitchB 收到这个 TCN BPDU ,回送一个 TCN ACK 进行确认 ,同时向自己的根端口转发这个 TCN BPDU
4) Root 收到这个 TCN，回送一个 TCN ACK 给 C。
5) Root 修改自己的配置 BPDU，以此来通告整个交换网络关于拓扑变更的情况。 Root 在配置 BPDU 中设置一段时间的拓扑变更（TC 标志），这段时间等于转发延迟+Max. Age，默认 35S
6) 当交换机收到 Root 发出的这个 TC 标志置位的配置 BPDU，它们使用转发延迟计时器（默认 15S）来更新其 MAC 地址表中的条目。也就是说条目的寿命由原来的 300S 的默认值变成 15S，这样能保证 MAC 地址条目更快速的刷新。交换机将持续这个过程，直到不再从 Root 收到 TC BPDU 消息为止。

我们会发现当拓扑变更的时候，就会产生 TCN，然而有些情况下 TCN 的过渡泛洪可能会对网络才成不必要的影响，通过在接入层交换机上、连接 PC 终端设备的接口设置为 portfast 可以在一定程度上优化网络，防止由于 PC 的开关机导致的接入交换机端口 updown 而产生过多的 TCN。

3.3.6 Spanning-tree 特性

1. Portfast

portfast 端口一旦接了设备，接口可绕过 listening 和 learning 状态直接进入 forwarding 状态。

```
Switch(config)# spanning-tree portfast default
```

（全局配置命令）将所有非 trunk 接口激活 portfast 特性

```
Switch(config-if)# spanning-tree portfast [trunk]
```

将特定接口激活 portfast 特性

Spanning-tree portfast 特性不能直接配置在 trunk 模式的接口上，否则即使配上去了，CISCO IOS 也不生效，除非该接口变成 access 模式。如果确实需要在 trunk 接口上配置，例如该接口连接了一台支持 trunk 的服务器，那么就在 Spanning-tree portfast 命令上增加 trunk 关键字。

```
Switch(config-if)# switchport mode host
```

一条宏命令，指定接口 mode 为 access 并且开启 portfast 特性

```
Switch(config)# spanning-tree portfast bpduguard default
```

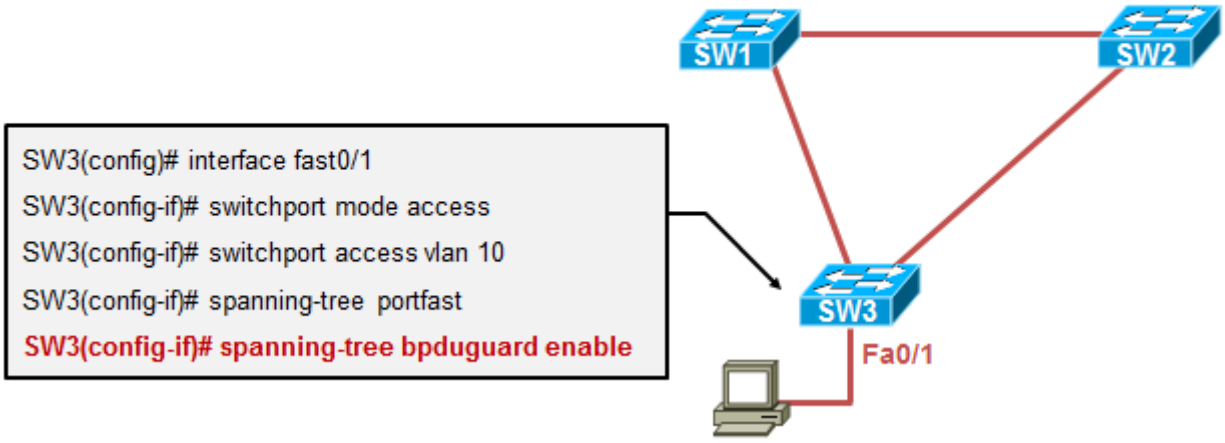
将所有激活了 portfast 的接口激活 bpduguard

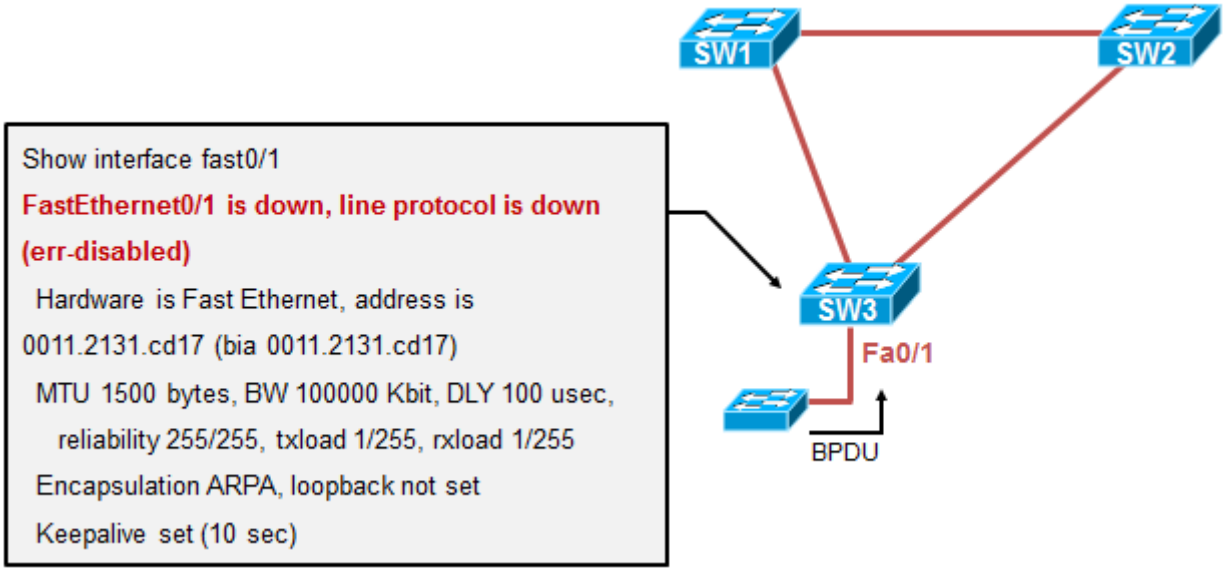
注意，虽然配置了 portfast 的接口在 no shutdown 或者 link 一旦 up 的时候，会 jump to forwarding 状态，绕过 listening 和 learning，但是，这个接口建议必须是连接路由器或 PC 的。如果连的是交换机，这个接口仍然要接受 spanning-tree 的计算结果，如果计算结果是 block，那么这个接口仍然会被 block。这就是为什么你在 CISCO IOS 上敲入该条命令，IOS 会提示你，说可能会造成短暂的桥接环路。

2. **BPDUguard**

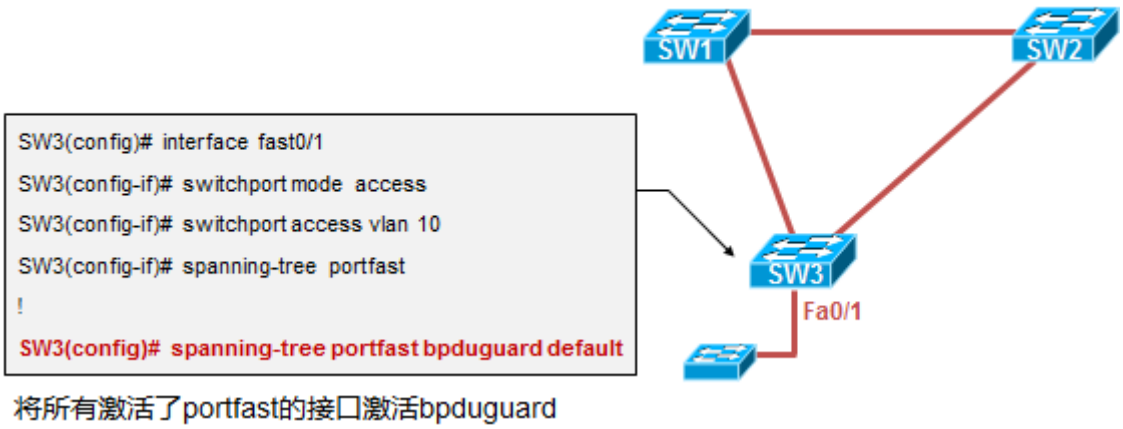
- **该接口在收到 BPDU 报文后，会立即切换到 err-disable 状态**
- **常搭配 portfast 特性在接口上一起使用，用于连接主机**
- **可在接口模式上激活，也可在全局模式上配置，两者有所不同。**

在接口模式下配置，该接口收到 BPDU 报文后，立即切换到 err-disable 状态：





在全局下配置，使用命令：SW3(config)# spanning-tree portfast bpduguard default，该命令会在所有激活 portfast 特性的接口上激活 bpduguard：



使用 err-disable 命令可以修改 err-diasable 状态的持续时间，默认是 300S

3. BPDUFilter

可以在全局模式下配置，也可以在接口模式下配置，区别如下：

全局配置：spanning-tree portfast bpdufilter default

a) 启用了 portfast 的接口将激活 bpdufilter 特性

b) 上述接口在 link up 后瞬间会发送 BPDU（a few），此后不在发送任何 BPDU

c) 上述接口在收到 BPDU 后立即丢失 portfast 及 bpdufilter 特性，成为一个普通的 spanning-tree 接口

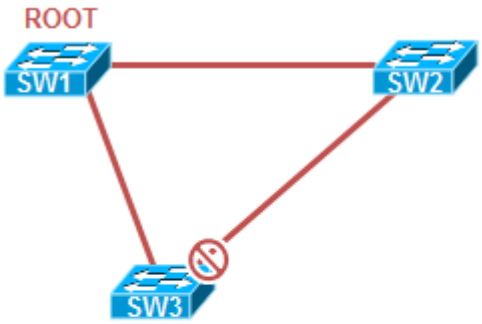
接口配置： spanning-tree bpdufilter enable

d) 该接口将不会发送 BPDU，也忽略接收到的 BPDU

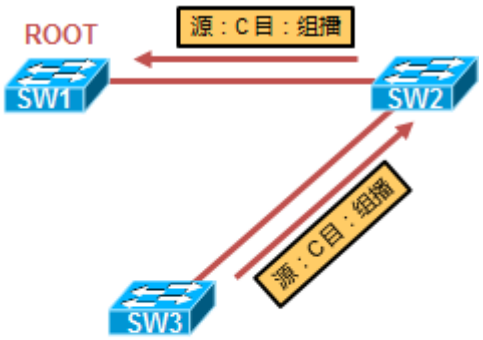
e) Enabling BPDU filtering on an interface is the same as disabling spanning tree on it and can result in spanning-tree loops.

f) 在接口上配置，不一定必须 portfast 特性，可独立实施。当然，建议搭配 portfast 特性使用。

4. UplinkFast



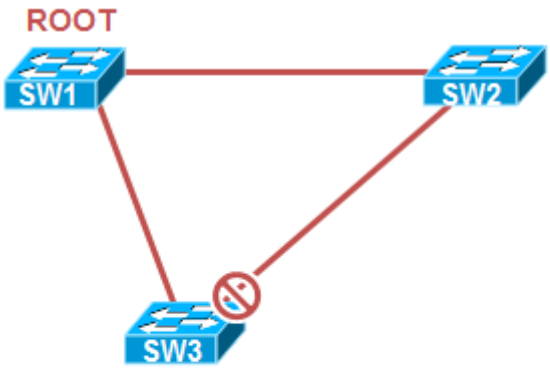
- 激活uplink特性
- SW3跟踪去往根桥的所有可用路径
- 当端口DOWN掉，则立即进行切换，跳过30S的延时



- SW3会代表下联的PC去发送一个用于更新其他交换机CAM表的报文，使得其他交换机能够更新自己的MAC表

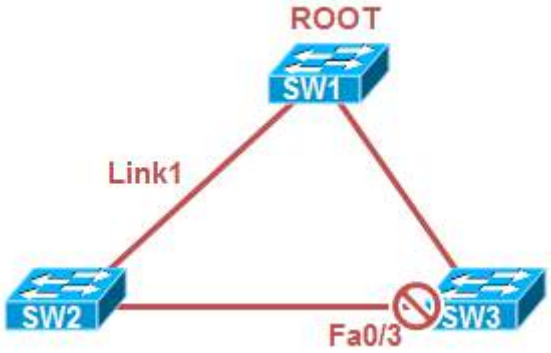
- **配置 Uplinkfast 的交换机需为末梢交换机，或者网络三层结构中的接入层交换机，不能为根桥**
- **在激活 Uplinkfast 特性后，交换机会自动调整一些参数：**
  - 交换机的默认桥优先级会增加到一个比缺省值更高的值（32768->49152），以使得该交换机不会成为 Root
  - 交换机的所有端口的默认 COST 值会增加 3000，以使得该交换机的端口不被选举为指定端口
  - 非默认的（手工配置的非默认）priority 与 cost 不变

另外：When the spanning tree reconfigures the new root port, other interfaces flood the network with multicast packets, one for each address that was learned on the interface. You can limit these bursts of multicast traffic by reducing the max-update-rate parameter (the default for this parameter is 150 packets per second). However, if you enter zero, station-learning frames are not generated, so the spanning-tree topology converges more slowly after a loss of connectivity.



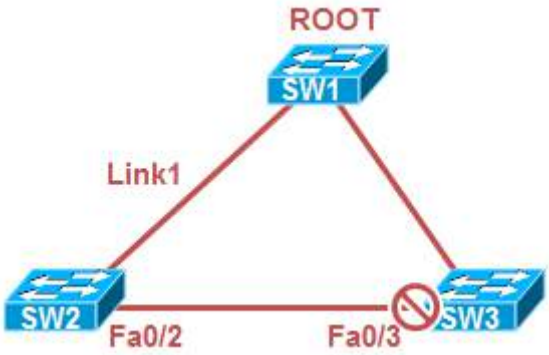

```
SW3(config)# spanning-tree uplinkfast
SW3# show spanning-tree uplinkfast
```

5. BackboneFast



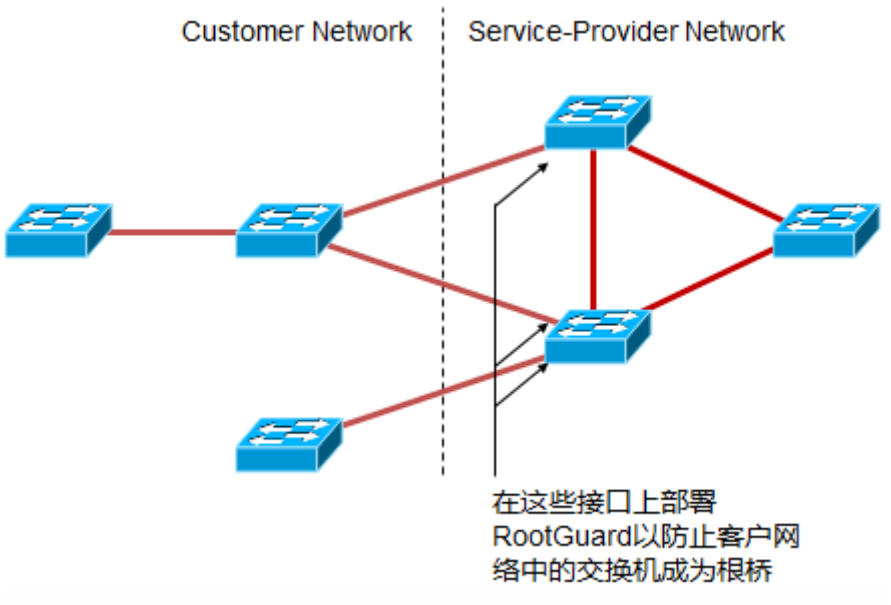
- 拓扑正常的情况下，SW3 有一个接口被 BLOCK，这个接口将不会发送 BPDU 报文
- 当 LINK1 DOWN 掉后，SW2 将无法从 ROOT 收到 BPDU，于是它认为自己的 Root，并且开始向 SW3 发送 BPDU
- SW3 收到这个 BPDU，但是发现这个 BPDU 比之前自己本地存储的 BPDU 更不优，因此忽略该 BPDU
- MAX_AGE（20S）计时器超时后，SW3 上 Fa0/3 存储的 BPDU 老化，该端口进入侦听状态，并发送和接收 BPDU
- SW2 收到 SW3 发送的 BPDU 后，就停止发送它自己的 BPDU
- SW3 的 Fa0/3 从 Listening 到 Forwarding，需花费 20+30S

接下来看看 BackboneFast 的特性：

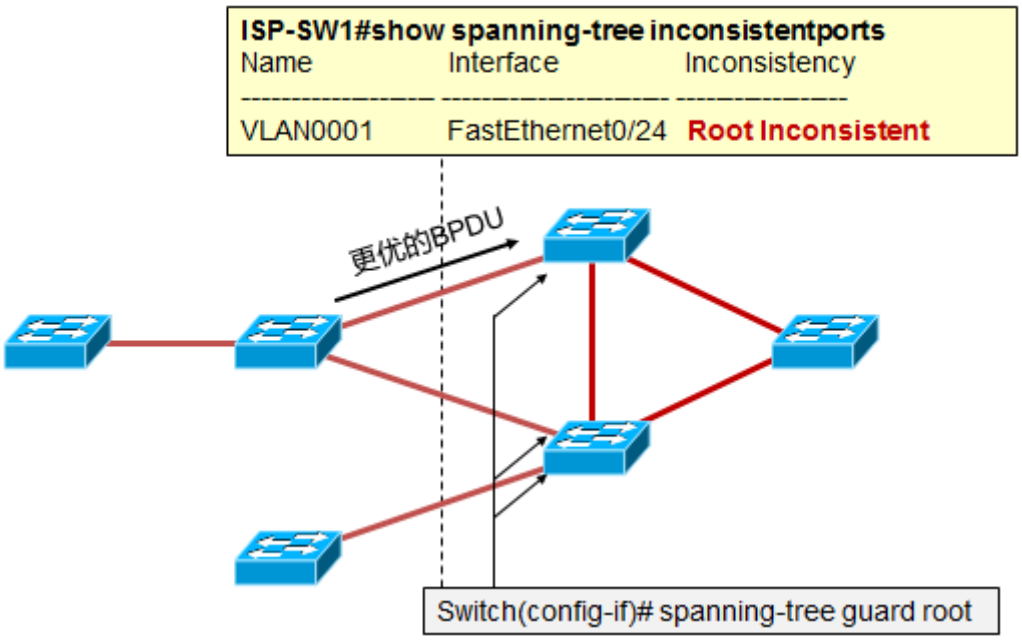


- 在部署了 Backbone Fast 后，SW3 一到收到 SW2 发送来的次优 BPDU，会即刻进行一系列步骤以重新计算根端口。SW3 会从根端口向根桥发送 RLQ 请求
- Root SW1 收到 RLQ 请求，立刻以 RLQ 响应进行回复，以告知自己仍然存活
- SW3 立即老化掉存储在 Fa0/3 上的 BPDU，端口 Fa0/3 进入 listening 状态并开始发送 BPDU
- SW2 从 SW3 收到 BPDU，经过计算得出自己的 Fa0/2 为根端口

6. RootGuard



上图中，客户与运营商网络之间是二层链路连接，如果客户的设备由于某种原因被选举为根桥，那就麻烦了，因此可以使用 rootguard 特性，在运营商连接客户的接口上部署。那么当这些接口收到来自客户的、更优的 BPDU 后，当运营商设备经过 STP 计算后得出这些接口即将成为根端口，此时设备立即将接口置为 root-inconsistent (blocked)状态。因此使用该特性，可以防止客户设备成为 Root，或者防止运营商设备上、连接客户设备的这些接口成为根端口。



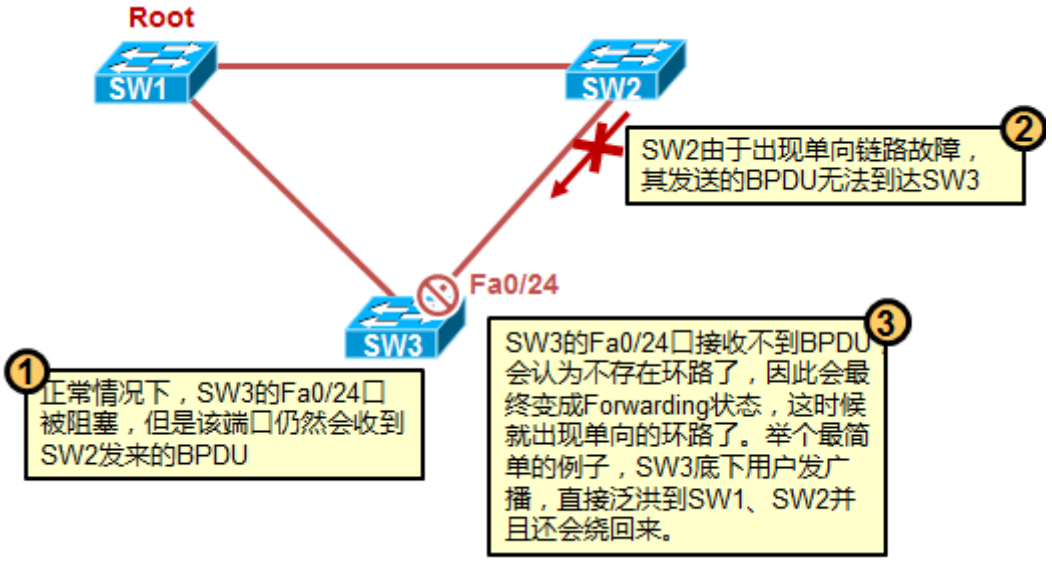
使用 show spanning-tree inconsistentports 命令查看到相关的表项，

注意 inconsistent 跟 err-diasble 的区别是，err-disable 会 disable 掉整个接口，而 inconsistentport 是针对特定的 VLAN 的。

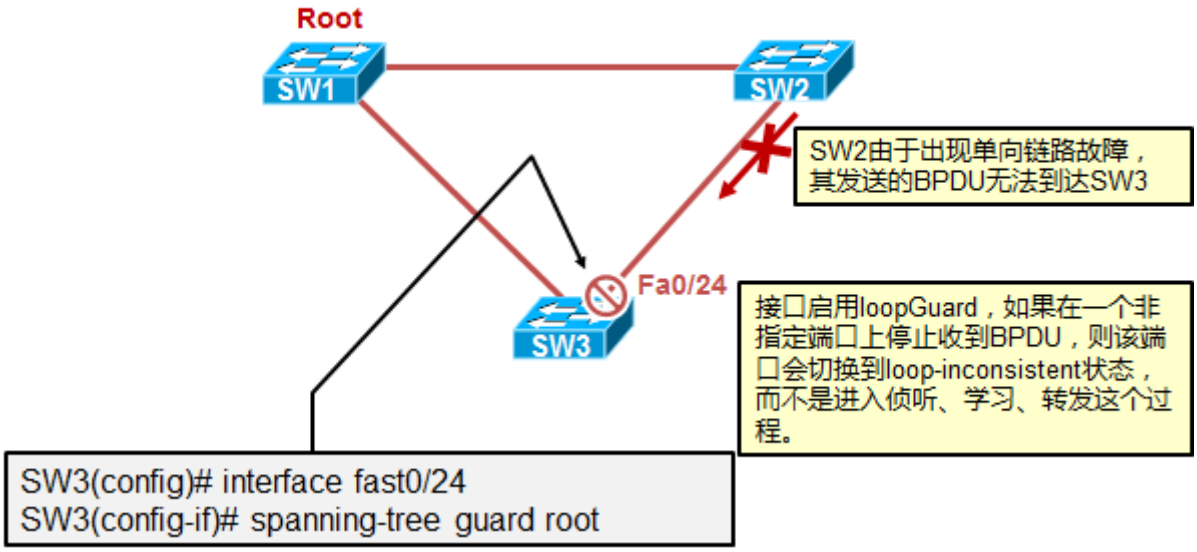
7. LoopGuard

You can use loop guard to prevent alternate or root ports from becoming designated ports because of a failure that leads to a unidirectional link. This feature is most effective when it is enabled on the entire switched network. Loop guard prevents alternate and root ports from becoming designated ports, and spanning tree does not send BPDUs on root or alternate ports.

You can enable this feature by using the **spanning-tree loopguard default** global configuration command.

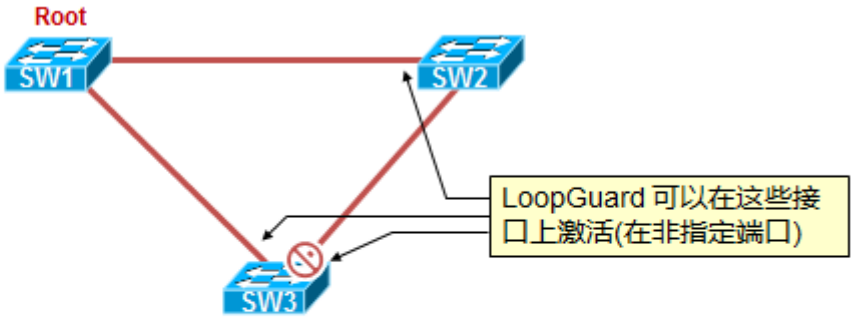


部署 loopguard 后：



注意事项：

- **与 Root Guard**
  - LoopGuard 与 RootGuard 是无法同时启用的




```
SW3(config)# spanning-tree global-default loopguard enable
SW3(config-if)# spanning-tree guard root
```

8. **UDLD**
   - **关于 UDLD**

     用于单向链路检测，主要用于光纤链路。需要链路两端的设备都支持 UDLD

     当 UDLD 检测到单向链路故障，它 administratively shuts down 掉这个接口，并且提示用户。

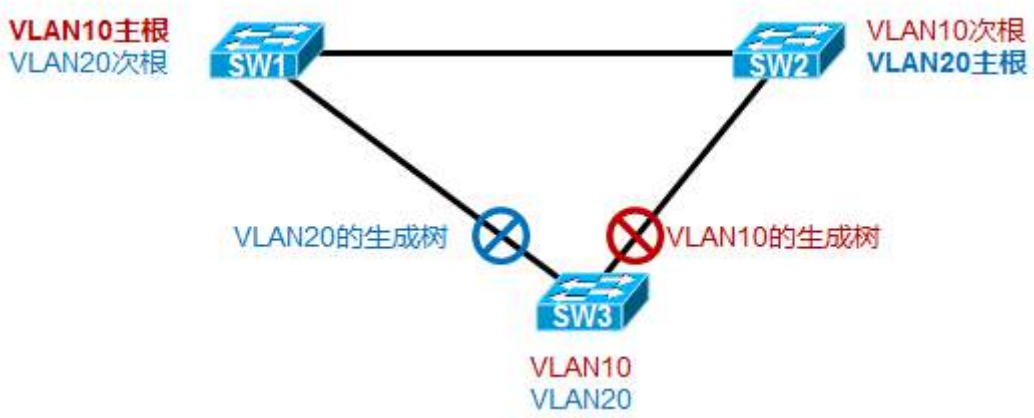
     单向链路会引发各种问题，其中包括 spanning-tree 拓扑环路

   - **UDLD 操作模式**

     两种操作模式：normal (the default) 和 aggressive

3.4 PVST+

3.4.1 基本概念



传统的 STP，也就是 802.1D，是所有的 VLAN 共用一棵生成树，这样一来的好处是交换机不用耗费过多的资源去计算多棵生成树，但是，缺陷是，通过 block 固定的端口，导致网络中的流量只走一侧，而部分链路一丁点流量都没有，浪费了带宽。

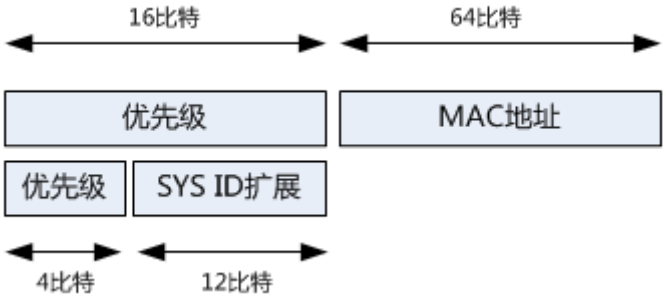
PVST+在这方面做了改进，perVLAN 的意思是，一个 VLAN 对应一棵生成树，如此一来交换机将基于 VLAN 计算生成树，我们可以通过对单独的 VLAN 生成树调节优先级等相关参数，从而影响生成树的计算，最终实现合理的链路带宽利用。例如上图，内网存在两个 VLAN，10 和 20，我们让 SW1 成为 VLAN10 的主根（全网交换机优先级最低），SW2 成为 VLAN10 的次根，SW3 优先级最高，如此一来我们将 BLOCK 掉 SW3 上连接 SW3 的上联口。相对的，VLAN20 也最特定的调节，这样，VLAN10 的用户出去将走左侧链路，VLAN20 的用户将走右侧，链路都得到了合理的运用，实现了负载均衡。

3.4.2 MAC 地址分配和缩减

- CISCO CATALYST 交换机的 MAC 地址池最多可以容纳 1024 个地址，交换机的型号决定了可用 MAC 的数目，并不是所有 catalyst 交换机都能支持到这么多个 MAC
- 这些 MAC 地址作为 VLAN 生成树中的网桥 ID 的 MAC 地址部分。不同的交换机型号支持不同的可用 MAC 地址数目。交换机依照次序分配 MAC 地址
- Show run int | include bia 能看到所有的 MAC，其中第一个 MAC 将被生成树使用，也就是 CPU 的 MAC。接下去就是每个以太网接口的 MAC。
- 我们知道交换机能够支持的 VLAN 的数据是很庞大的，如果开启 PVST+，每个 VLAN 一棵生成树，而没棵生成树都要有一个独立的标识，都需要耗费一个 MAC 的话，那么 MAC 地址池肯定是无法承受的。
- 因此需要使用到 MAC 地址缩减方案

在 PVST+中，一个 VLAN 一棵生成树，在每台交换机上，对于每一棵生成树需要有一个唯一的标示符，也就是网

桥 ID 要唯一，网桥 ID 前面说了，是由优先级和 MAC 地址构成，在 MAC 地址缩减方案中，同一台交换机的所有生成树的桥 ID 中，MAC 地址都使用自己交换机 CPU 的 MAC，同时将 16bits 的优先级进行扩展，变成 4bits 的优先级+12bits 的系统 ID ,通过这个系统 ID 来识别不同的 VLAN。在 CISCO IOS 中 ,这个系统 ID 用的就是 VLAN ID。



如此一来，优先级就成了最高的 4bits，也就有了为什么优先级必须的 4096 的倍数（2 的 12 次方=4096）。另外系统 ID ,取值 VLAN ID ,例如 vlan10 的生成树 ,在本交换机的桥 ID 就是" 4bits 的优先级+10+交换机的 MAC" SYSID 还可作为 VLAN 或 MST 实例的标识，比如 VLAN100 的系统 ID 扩展是 100

3.4.3 配置

Switch(config)# **spanning-tree** *vlan-id*

- 激活特定 VLAN 的 STP，如需关闭 STP，加 no

Switch(config)# **spanning-tree vlan** *vlan-id* **priority** *pri*

- 配置桥优先级

Switch(config)# spanning-tree vlan *vlan-id* root {primary | secondary} [diameter *diameter*]

- 设置主根、次根。用 primary 则在全网默认优先级（都是 32768）的情况下，该交换机生成树 vlan 优先级调整为 24576，如果是 secondary，则为 28672。但是如果网络中存在优先级为最低：4096 的交换机，那么这条命令也没则。也就是实际上这条命令就是检测目前网络中的其他设备的网桥优先级，并且来调整自己的优先级并保证最低，就这么简单。

  Diameter 指的是一个直径值，默认 7。这个值关系到 STP 计时器。不同的 diameter 值配置上去，将导致交换机自动调整 hello 及 max age 计时器，这条命令一般在 root 上配置，这样计时器的值会通过 bpdu 传递到其他交换机。

  因此其实这就是条宏命令

Switch(config-if)# **spanning-tree** [**vlan** *vlan-id*] **cost** *cost*

- 设置接口 cost

Switch(config-if)# **spanning-tree** [**vlan** *vlan-id*] **port-priority** *pri*

- 设置接口优先级，CISCO IOS 默认 128；CatOS 默认 32

Switch(config)# **spanning-tree** [**vlan** *vlan-id*] **hello-time** *sec*

Switch(config)# **spanning-tree** [**vlan** *vlan-id*] **forward-time** *sec*

Switch(config)# **spanning-tree** [**vlan** *vlan-id*] **max-age** *sec*

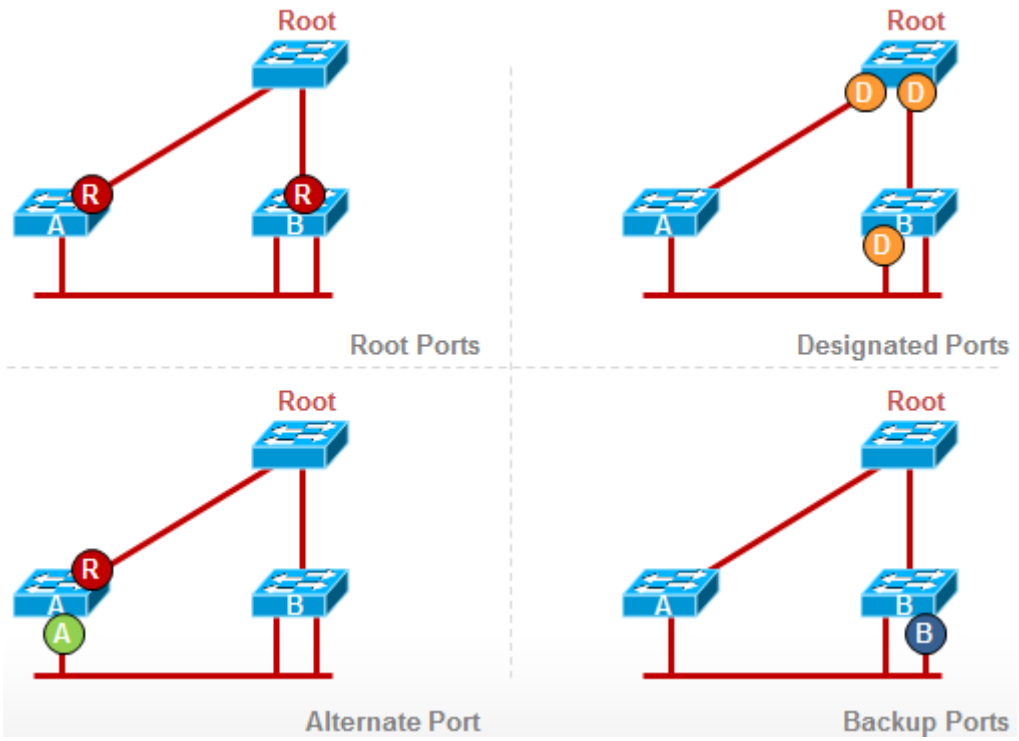
- 设置 STP 的 timer

3.5 RSTP (802.1W)

3.5.1 基础知识点

1. **RSTP 端口状态**
   - 丢弃（也就是 802.1D 中的禁用、阻塞、侦听的合并）
   - 学习
   - 转发

2. **端口角色：**
   - **根端口：　收到最优的 BPDU 的接口就是根端口。这是距离 Root 最近的（cost 最小）的接口。**
   - **指定端口：　在每一个 segment 上选择一个指定端口，该端口将发送这个 segment 上最优的 BPDU。**
   - **替代端口：**　丢弃状态。本交换机除了根端口外，其他到根路径的端口，如果活跃的根端口发生故障，替代端口将成为根端口，所以替代端口可以理解为根端口的可替代者
   - **备份端口：**　丢弃状态。指定端口的备份，出现在一台交换机有两个端口连接到同一个共享介质时。
   - **禁用端口**



STP 与 RSTP 的端口对比：

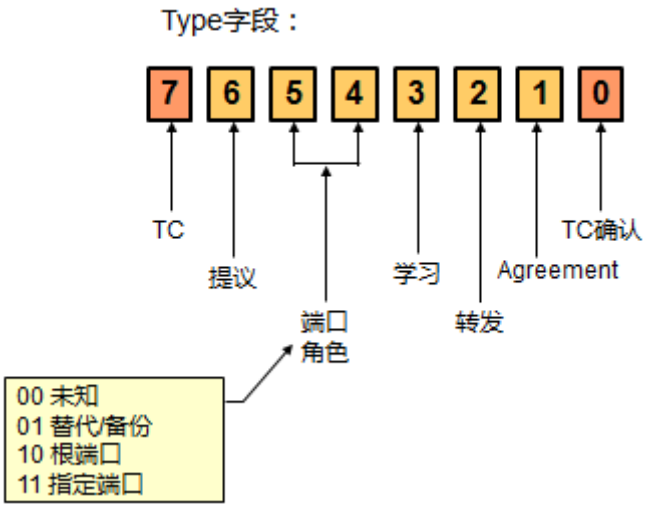
STP端口角色	STP端口状态	RSTP端口角色	RSTP端口状态
Root Port	Forwarding	Root Port	Forwarding
Designated Port	Forwarding	Designated Port	Forwarding
Nondesignated Port	Blocking	Alternative or Backup Port	Discarding
Disabled	-	Disabled	Discarding
Transition	Listening or Learning	Transition	Learning

3.5.2 BPDU 格式和操作

1. **802.1W BPDU 格式**

   RSTP 只在 802.1D 基础上对 BPDU 做了少量修改：

   在 802.1D 中，TYPE 字段只使用了最高位和最低位，来表示 TC 和 TC 确认，RSTP 对该字段进行的扩展：



RSTP BPDU 的协议是 2，版本是 2。

2. BPDU 操作

在 802.1D 中，非根交换机只有从根端口收到根桥发送的 BPDU，自己才能产生 BPDU。而在 RSTP 中，即使非根交换机没有从根交换机处收到 BPDU，其自己也以“hello 间隔”为周期（默认 2S）发送 BPDU。

3. Faster Aging of Information

在特定的接口，如果连续三个周期没有收到 BPDU（或者 max-age 超时），接口上的 STP 协议数据将迅速老化，如此一来，BPDU 又有点类似交换机之间的 keep-alive 机制。这种快速老化的机制有助于对拓扑变化的快速响应。

4. Accepts Inferior BPDUs

这个机制与 CISCO 的 BackboneFast 特性非常类似。

当交换机从其他指定交换机或根桥收到次优 BPDU，802.1D 遇到这种情况是首先忽略这些次优 BPDU，而 RSTP 是立即接受这些次优 BPDU 同时回传一个更优的 BPDU。

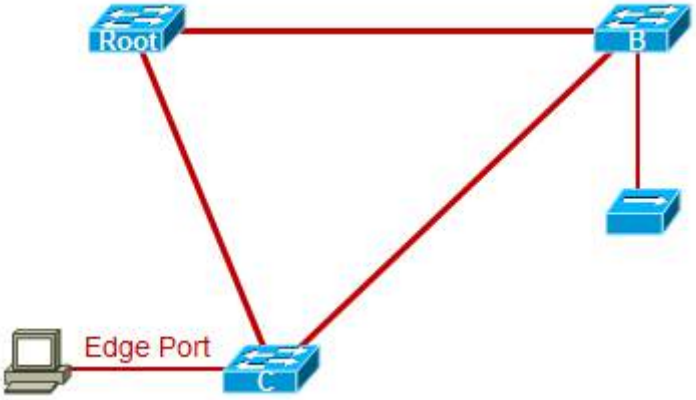
看上图，初始情况下，C 的一个接口被选举为非指定端口被 Block，B 会从指定端口发送 BPDU 给 C。我们先考虑一下 802.1D 的情况，当 Root 及 B 之间的链路故障了，由于 C 上连接 B 的接口被 BLOCK，它不发送 BPDU 给 B，因此，B 此刻认为自己是 Root，于是向 C 发送自己为 Root 的 BPDU。而由于这个时候 C 的接口上还存储着之前 B 发给自己的 BPDU ,而这个 BPDU 相比与 B 后来发送给自己的 BPDU 更优 ,因此 C 直接忽略这些次优 BPDU，一直到 Max-Age 超时，C 的接口上存储的此前 B 发过来的 BPDU 才会老化，这时候接口进入 LST，才开始发送 BPDU，指示根为 Root。而此刻 B 才接受事实，将自己连接 C 的接口置为根端口。

那么对于 RSTP 情况就不一样了，C 在收到次优 BPDU 后，将立即回送自己的 BPDU，好让 B 了解拓扑情况。这个机制跟 BackboneFast 非常类似。

3.5.3 Rapid Transition to Forwarding State

RSTP 的一个重要的改进是端口的快速过渡。传统的 STP 算法在将一个接口过渡到 forwarding 状态之前，需要经历几个计时器。为了获得网络的快速收敛，我们可能会去调整计时器，然而这种方式有可能影响网络的稳定性。RSTP 的设计，使得我们不用依赖调整计时器，并且可以使得接口可以安全的过渡到转发状态。为了实现接口上的快速收敛，RSTP 引入了一些新的概念：

1. **边缘端口 edge ports**



RSTP 定义的这种端口类型与 Portfast 十分类似。因为这些接口用于连接主机，所以一般不会产生环路。这些端口可以跳过 LST 或 LRN 直接过渡到转发状态。并且当这些接口 up down 的时候不会引起拓扑变更。另外，边缘端口一旦收到 BPDU，则立即丢失边缘端口的特征，变成一个普通的 spanning-tree 接口。

在 catalyst 交换机上，可以用 portfast 关键字来进行手工配置。

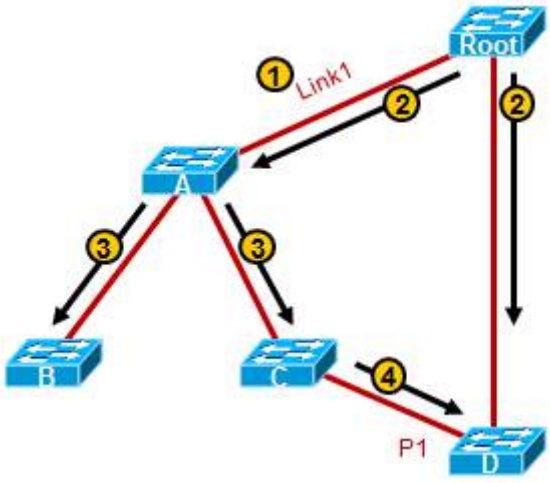
2. 链路类型 Link types

RSTP 能够在边缘端口及 point2piont 链路上快速过渡。RSTP 的链路类型是通过接口的双工状态自动获取的，如果接口是半双工，那么链路类型就是 shared port，如果是全双工，那么就是 point2point。

当然，接口的链路类型可以通过命令修改，接口模式下：spanning-tree link-type ?

3. 802.1D 与 RSTP 的收敛对比

- **802.1D 的情形：**

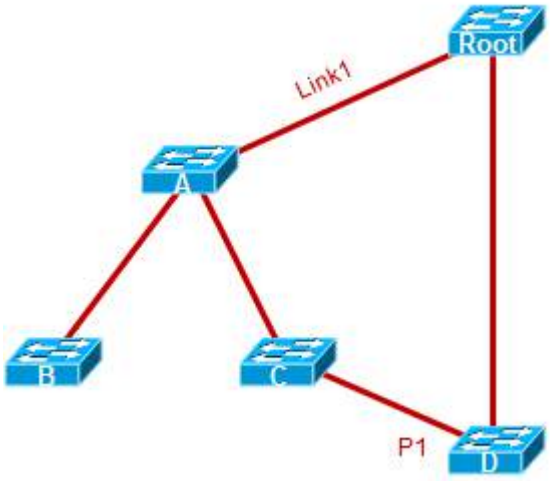


1. A 及 Root 之间新增链路 Link1
2. A 及 Root 在 Link1 两端的接口都进入 Listening 状态，A 将收到 Root 发出来的 BPDU
3. A 将 BPDU 从自己的指定端口发送出去，BPDU 被泛洪到网络中

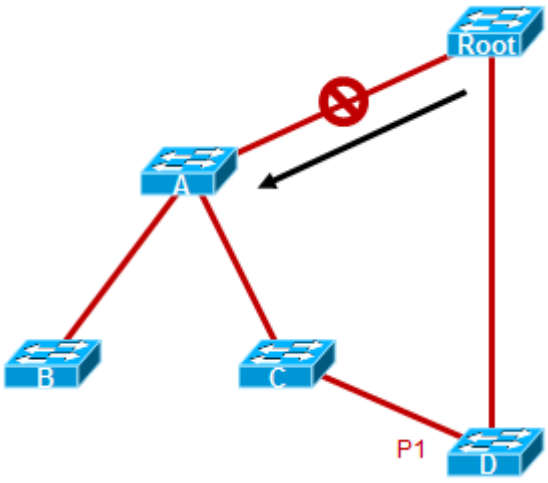
4. B 和 C 收到这个更优的 BPDU，继续向网络中泛洪
5. 数秒后，D 收到这个 BPDU，Block 掉端口 P1

由于缺乏 feedback 机制，A 连接 Root 的接口从 Listening 到 Forwarding，需要经历 15*2s 的延迟。此时 A、B、C 下联的用户流量就出现问题了（因为 D 在收到更优的 BPDU 后，将 P1 口 block 了，这时候 ABC 相当于在 A 的根端口过渡到 forwarding 之前都处于网络的"隔离地带"）

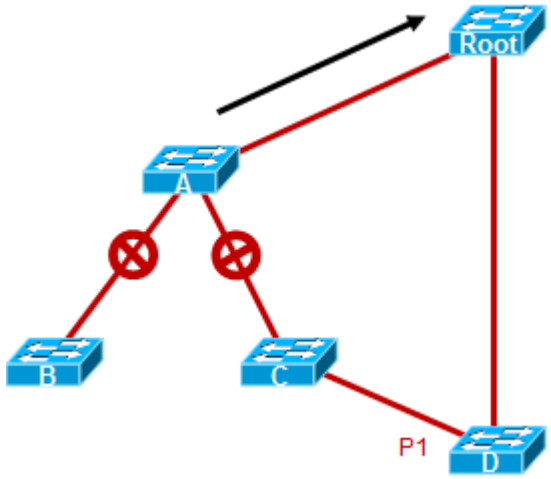
- **RSTP 的情形：**



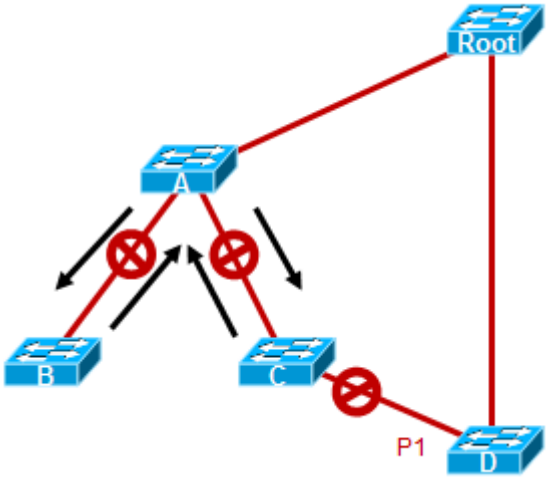
1. A及Root之间新增链路Link1



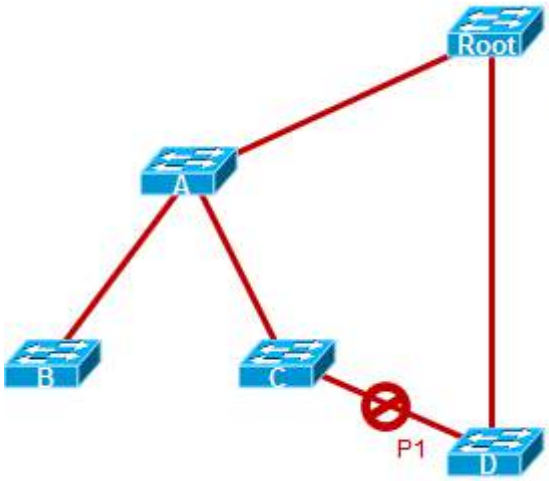
2. A及Root在Link1两端的接口在up之后进入designated blocking。然后双方交互BPDU消息。这个过程实际上是一个协商的过程



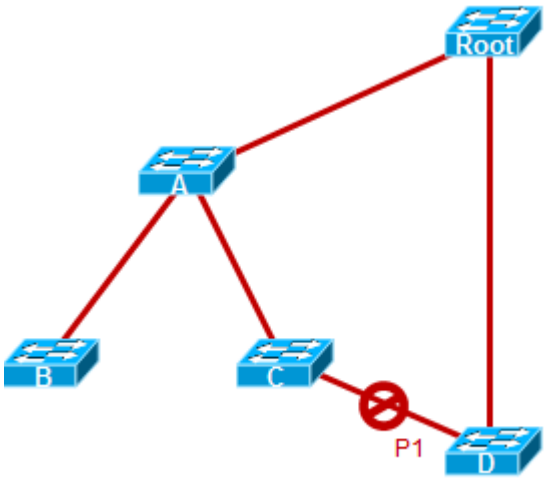
3. A在收到Root发送的BPDU后，将自己的所有非边缘端口Block（这个过程称为同步sync），并且回送一个agreement消息给Root
4. 在此之后，Root及A在link1上的端口立即都过渡到转发状态。而网络目前是没有环路的，A往下的网络目前是切断的



5. A与B和C之间，开启一轮新的协商，BC收到A发送的BPDU后，完成同步Sync过程，将自己的非边缘端口BLOCK掉，然后都向A回送agreement消息。同时，ABC互联的接口进入转发状态。在BC同步操作过程中，B下联全是主机，因此没有端口被Block（已经完成同步）；而C要BLOCK掉连接D的端口。



7. 完成上一步之后，生成树状态如图：
8. 最终BPDU到达D，D将P1口Block掉



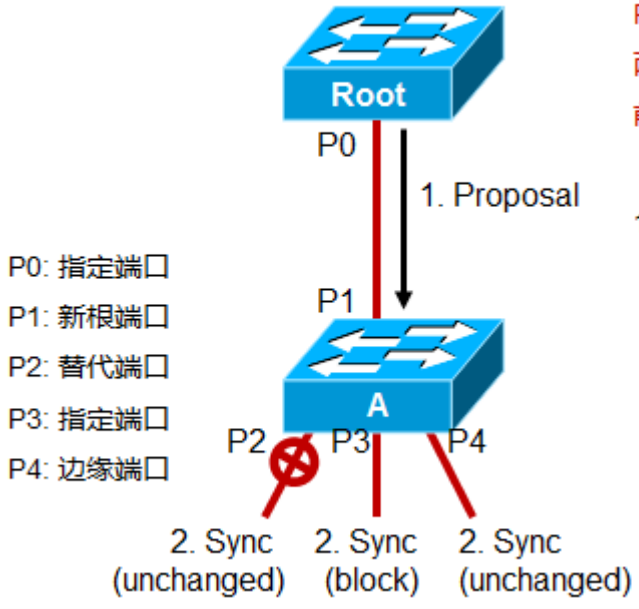
在RSTP收敛过程中，耗费是时间仅仅是BPDU从Root泛洪到网络末端的时间，不用受到任何Timer的限制，直接绕过两个转发延迟时间。因此收敛速度更快。

有两点需注意：

- 交换机之间的这种协商机制只在P2P链路上被执行
- 边缘端口的配置非常重要，如果配置不当，有可能会在同步过程中被BLOCK。

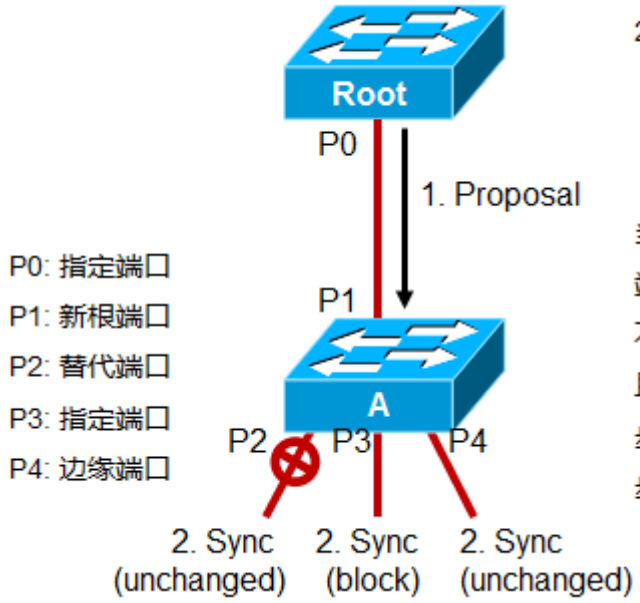
4. Proposal/Agreement Sequence

对于 802.1D 来说，当一个端口被选举为指定接口，它从 blocking 到 forwarding 至少需要 30S 的时间。然而在 RSTP 中，proposal/Agreement 机制使得接口能够在几秒内完成迅速、可靠的过渡。



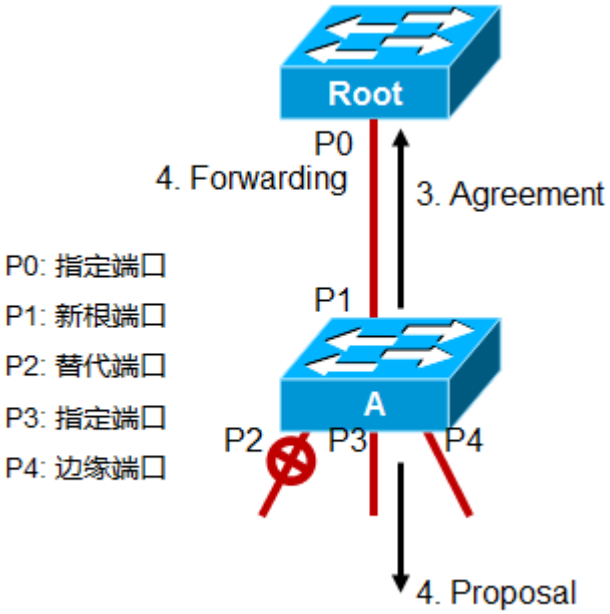
Root和A之间新增了一条链路，链路两端的接口在收到对方发送的BPDU前是designated blocking状态

1. 当一个被选举为指定端口的接口在discarding或learning状态（且只在这个状态），它在其发送的BPDU中进行proposal bit置位。这就是步骤1的P0的情况



2. A收到一个最优的BPDU，它立刻知道P1就是新的根端口。接下去A启动一个同步进程sync，A的所有端口将会促发这个进程。

当一个接口为Blocking状态或为边缘端口，则该接口已同步。而如果接口不满足上述两个条件，它将被block并且进入discarding状态已达同步。在步骤2中，P2、P3、P4都已经完成同步了。



3. 在A完成同步后，A就可以将新选出的根端口unblock并且发送一个agreement消息给Root。这个agreement消息是A的proposal消息的拷贝，但是agreement bit置位了。
4. 如此一来，P0就收到了应答，立即转为forwarding。注意这时候P3接口仍处于designated discarding状态，于是它向它的邻居网桥去发送proposal，而且也在积极等待回传的agreement以便进入forwarding状态.

小结：

- proposal agreement 的机制是非常快速的，因为它们不用受限于任何计时器，这个握手机制会迅速的蔓延到整个交换网络末梢，并且能在拓扑发生变更的时候迅速收敛。
- 如果一个 designated discarding 接口在发出 proposal 后 ,没有收到 agreement ,它将慢慢的过渡到 forwarding 状态，这个过程是传统的 802.1D 的 listening-learning-forwarding 过程。这种情况有可能发生在对端交换机不理解 RSTP BPDUs 或者对端交换机的端口被 block 的情况。

3.5.4 拓扑变更机制

1. **拓扑变更机制（检测）**
   - 在 RSTP 中，只有当非边缘端口过渡到 forwarding 状态才会触发拓扑变更。也就是说，一个端口如果丢失了连接，则不再认为是一次拓扑变更，这与 802.1D 是有区别的。
   - 当一个 RSTP 交换机检测到一次拓扑变更它将：
     - 为根端口及所有的非边缘指定端口启动一个 TC while timer，timer 的值等于 2 倍的 hello-time 计时器
     - 向上述端口泛洪 MAC 表
     - 注意主要 TC while timer 在端口上计时，端口发送出去的 BPDU 就会进行 TC bit 置位，该 BPDU 也会从根端口往外发送

2. **拓扑变更机制（传递）**

   当一台 RSTP 交换机收到 TC bit 置位的 BPDU，它将：

   1) 清除从所有接口学习到的 MAC 表项，除了收到 TC BPDU 的那个接口。这个动作虽然有可能导致网络中存在短暂的突发性泛洪，但是也有利于刷新 CAM 表、清除 stale 表项。
   2) 激活 TC while timer ，然后从所有的非边缘指定端口及根端口往外发送 TC 置位的 BPDU。通过这种方式，拓扑变更信息会迅速在网络中泛洪。

   通过这种方式，TC 消息会被迅速的泛洪到整个网络。而不用像 802.1D 哪样，把消息传递到 root，再由 root 来同时拓扑变更。

4 三层交换

4.1 CAM 及 TCAM

1. **CAM 表**

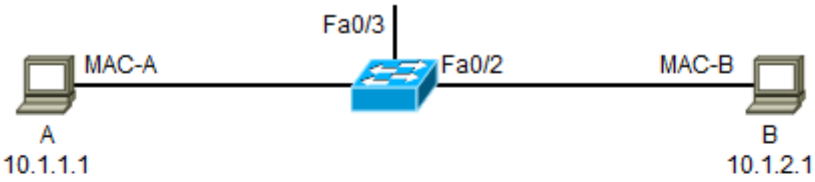
   CAM 是交换机用于二层交换时所查的表

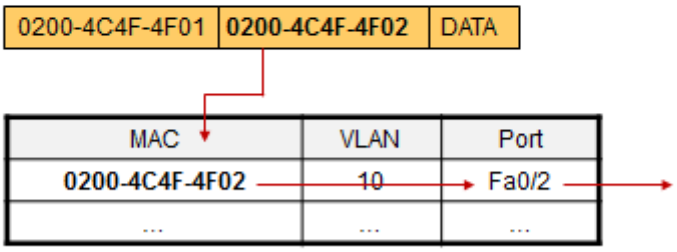
   a) Content Addressable Memory Table 内容可寻址内存

   b) 在查表时，使用二进制的 0、1 位进行匹配，并且需严格匹配，也就是目的 MAC 与 CAM 表中的 MAC 需

完全匹配

c) 查找的结果如果发现完全匹配项，则根据返回的端口号将数据转发出去





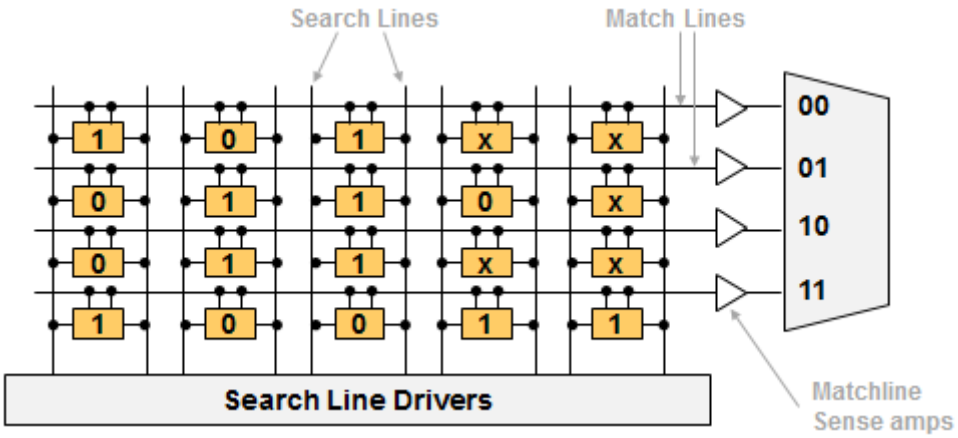
CAM表
（CAM表项实际上是MAC、端口、VLAN号等的HASH结果）

2. TCAM 表

TCAM 是路由模块或路由器用于三层转发时所查的表

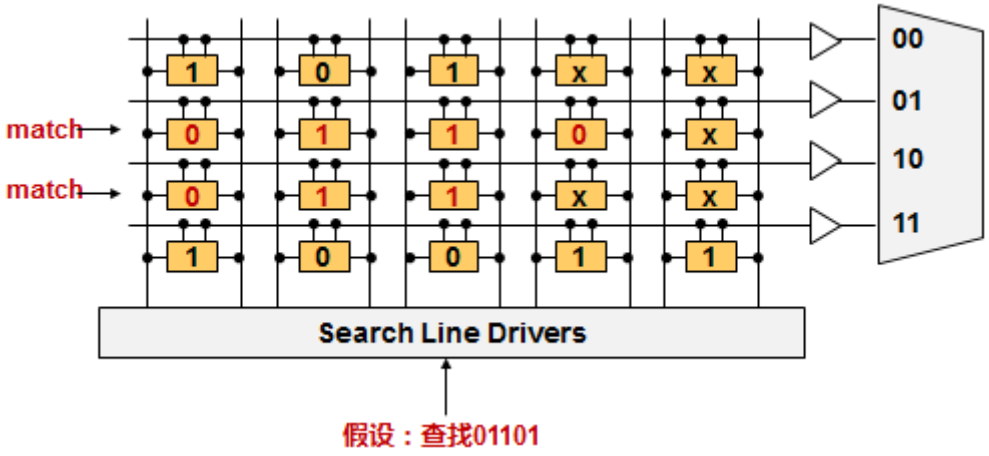
a) 匹配表项中的 0、1、x（x 为无所谓）

b) 最长匹配原则

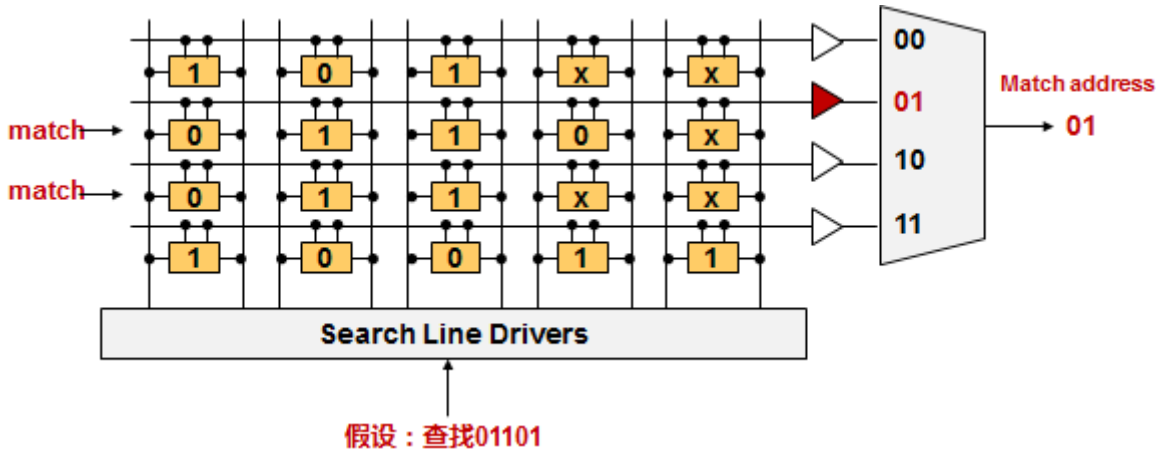


将所有的路由条目+掩码放进 TCAM 表项，就是图中的“行”，二进制的 0 和 1 都要匹配，x 为无所谓。

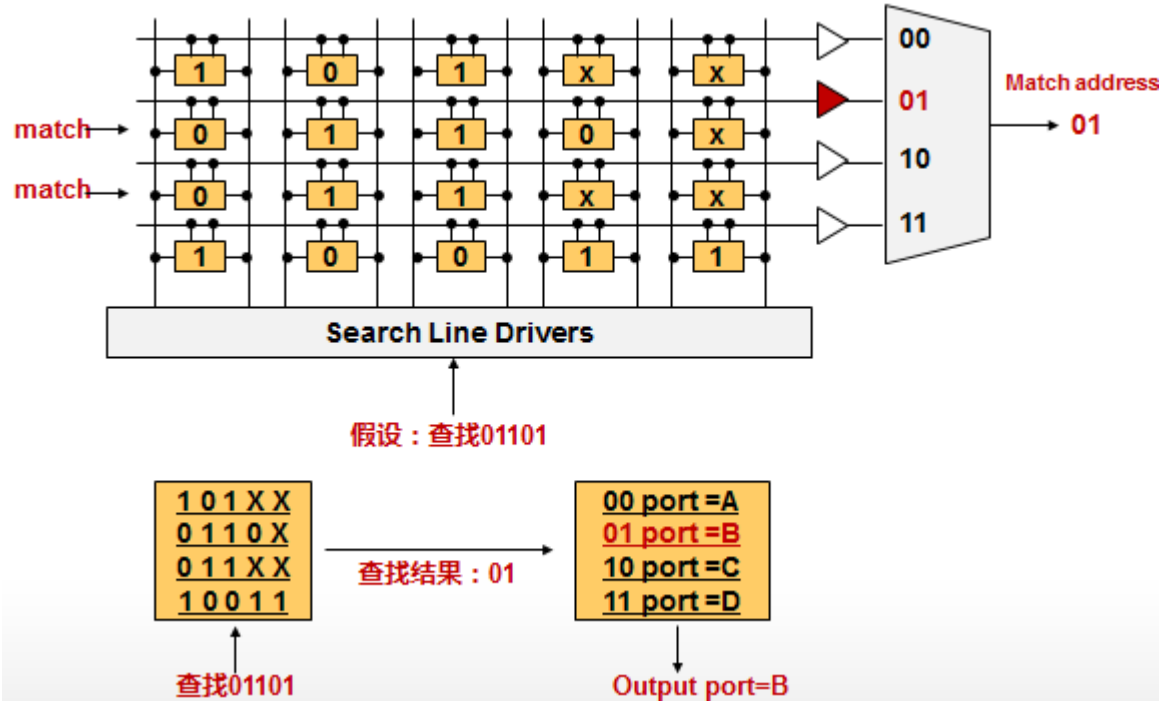
当我查找一个目的地的时候，例如查找“01101”：



那么会给 search lines 加电进行检测，结果发现第 2、3 行匹配，但是第 2 行的匹配长度最长，因此最终结果：

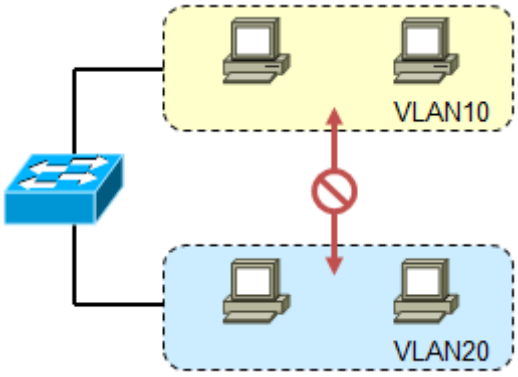


查找出来的结果，可以理解为一个指针，用于找到关联的出接口：

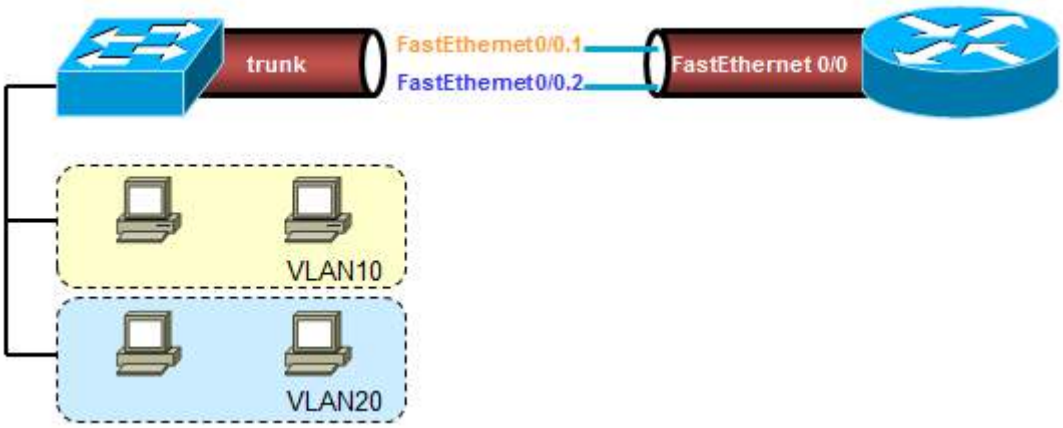


4.2 VLAN 间路由

4.2.1 单臂路由



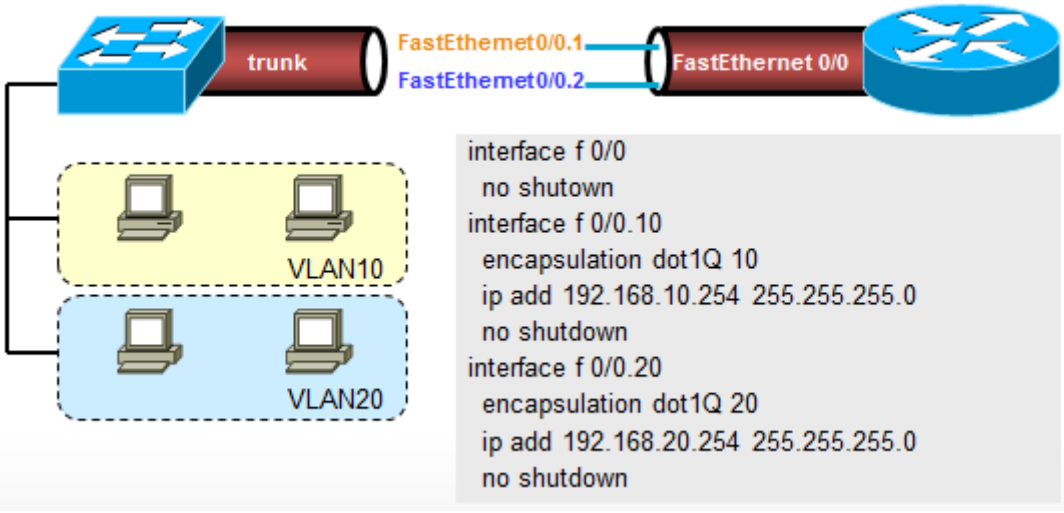
我们知道，在二层交换环境下，一个 VLAN 就是一个广播域，不同 VLAN 不同的广播域，一般也是不同的逻辑子网，而且相互隔离，互相是无法互访的，这样能起到隔绝广播的作用。但是实际网络中往往 VLAN 有互访的需求，例如同一公司不同的部门划分在不同的 VLAN，那么如果这些部门之间有数据往来的需求呢？那么二层交换机就无法实现了，需要借助三层设备。一个最简单的方法，就是使用路由器，用单臂路由的解决方案：



所谓单臂路由，就是在路由器的以太网口上（必须是 100M 接口以上），来承载 VLAN 流量，让路由器和交换机跑一个 Trunk，使用 dot1Q 的封装，这时候，为了让路由器的以太网口支持 Dot1Q 及识别并承载 VLAN 流量，那么需对物理接口进行子接口的划分，如上图。

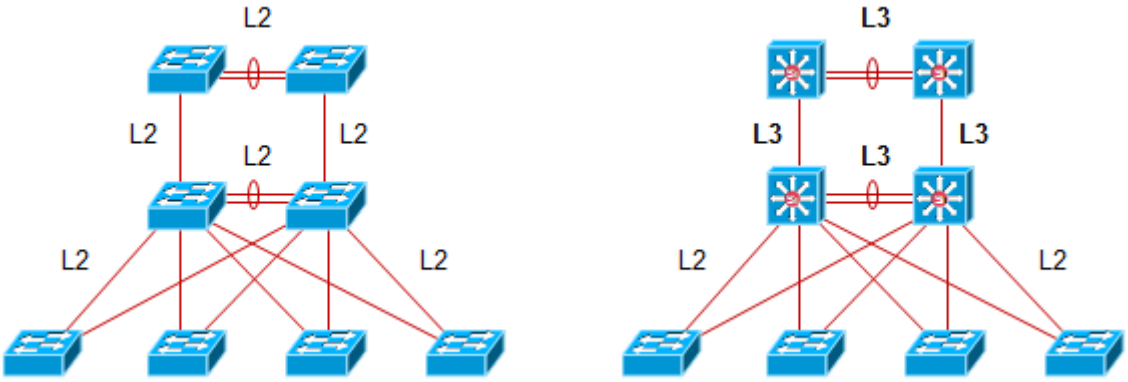
将 Fa0/0 接口划分成两个子接口，并且配置封装协议 Dot1Q，同时为流量分别打上 VLAN 的 tag，这样一来，交换机和路由器之间就起了一个 trunk。路由器的这两个子接口分别配置两个 VLAN 的网关 IP，作为 VLAN 用户的网关。

配置方式如下：



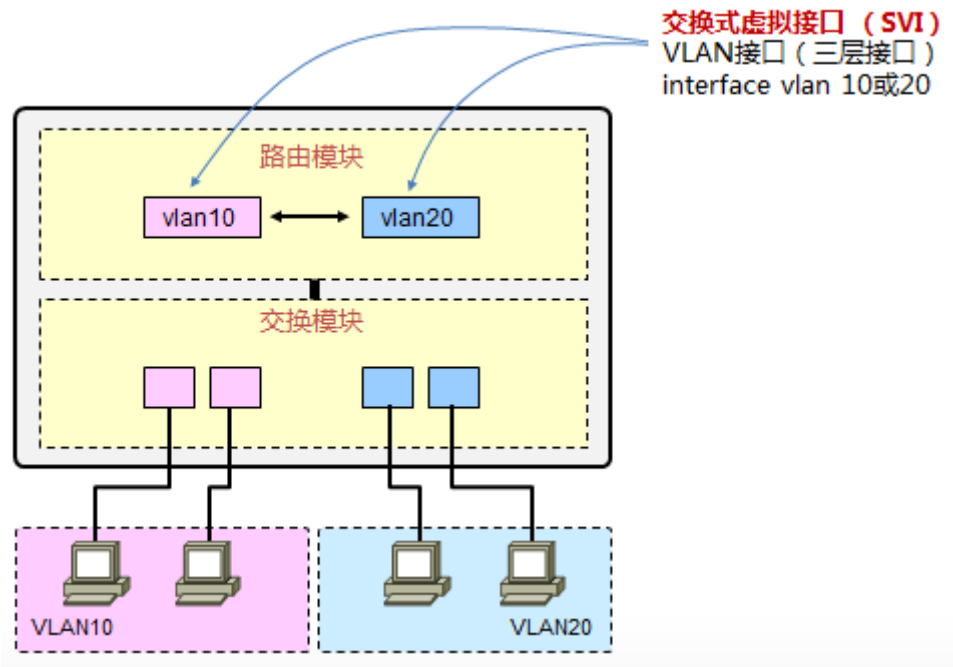
使用单臂路由确实能解决 VLAN 间数据互访的问题，但是却也存在种种弊端，例如路由器的接口，负载太大，流量需二次进出接口或链路，导致干道链路负载也过重，而且扩展性特别差。

4.2.2 路由 VS. 交换的园区网架构



- 在过去，交换是基于硬件的转发，而路由是基于软件的转发，因此园区网络更多的采用交换网络的设计
- 而如今，路由已经几乎与交换一样快，也能够基于硬件做转发，与此同时路由的设计很好的解决了交换网络的二层环路问题，以及 LAN 的隔离问题

4.2.3 Switch Virtual Interfaces（SVI）

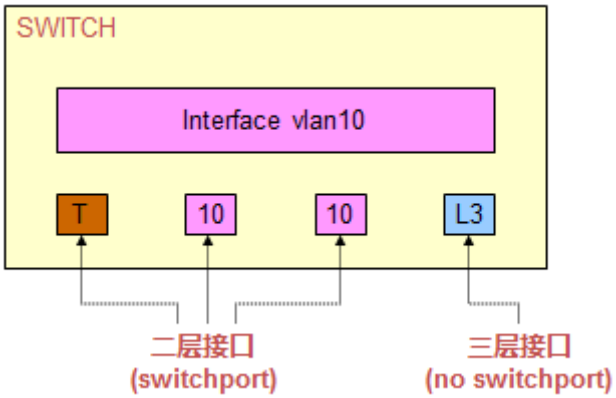


OK，在理解了单臂路由之后，我们再来看看三层交换机是如何实现 VLAN 间的数据互访的，我们从这里为切入点，开始理解并部署三层交换。我们知道二层交换机是可以实现二层交换的，它看的是数据帧，对帧头的二层信息进行读取并且根据自己的 CAM 表进行转发。而三层交换机相当于在二层交换机的基础上，多了个路由模块，于是乎它就能支持路由功能了 :支持路由选择协议、支持三层数据的转发、支持 IP 路由查找、支持三层接口等等。

先来认识一下第一种三层接口：SVI 交换式虚接口，SVI 的一个逻辑接口，也就是说不是一个物理接口，当我们在交换机上创建了一个 VLAN 之后，紧接着就可以创建一个与这个 VLAN 对应的 SVI 接口，例如我们创建了 VLAN10，那么 VLAN10 对应的 SVI 接口就是 interface vlan10 或者叫 SVI10，这个 SVI10 是一个三层接口，你可以为这个 SVI 口配置 IP 地址，与 VLAN10 内的 PC 用户的 IP 地址同一网段，那么这样一来，VLAN10 内的用户就能够将网关指向这个 SVI 接口，当 VLAN10 的 PC 需要访问本网段以外的网络时他们将数据交给网关，也就是 SVI10，再由 SVI 去做路由查找及数据转发。实际上，在这个理解过程中，我们可以拿单臂路由那个模型对类比。

所以看上面这图，在三层交换机上创建了两个 VLAN：10 和 20，同时为两个 VLAN 的 SVI 分配了地址作为各自 VLAN 的用户网关，这样一来，这台交换机的路由表里就有了两个 VLAN 网段的路由。那么当两 VLAN 之间要互访时，VLAN10 的用户将数据丢给自己的网关，也就是 VLAN10 的 SVI，数据到了 SVI10 之后，三层交换机查表，发现目的地是 VLAN20 的所在网段，因此将数据从 VLAN20 扔出去，最终抵达目的地的 VLAN20 的 PC。

4.2.4 三层交换机的各类端口



总的来说，三层交换机包含两类端口：二层接口 L2 和 三层接口 L3

- **二层接口(switchport)：access 模式、trunk 模式**
- **三层接口：路由接口（no swithport 或称为 routed port）、SVI 接口**

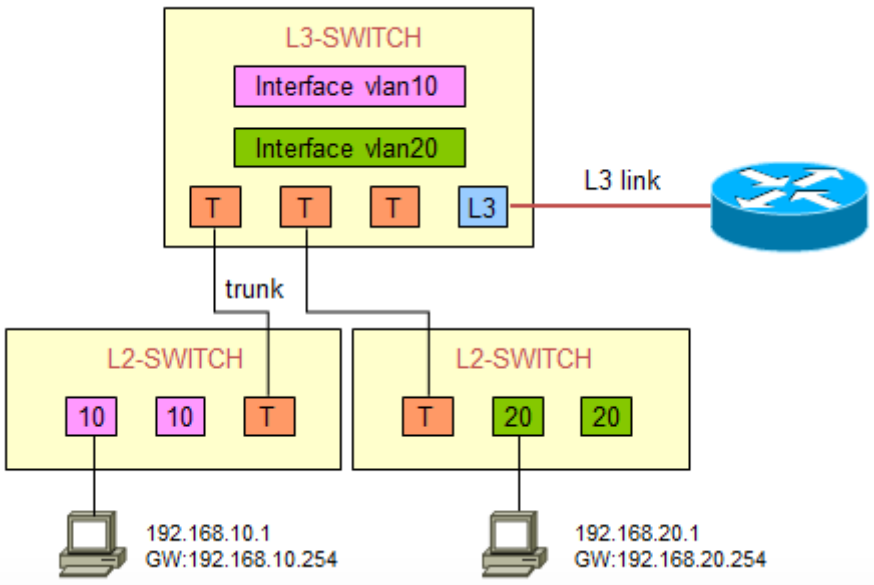
其中二层接口包含我们非常熟悉的 access mode 和 trunk mode。交换机的所有物理接口默认是二层接口，也就是 switchport。

三层接口有两种，一种是 SVI 上面我们已经讨论过了，另一种是 routed port，或者叫 no switchport。注意 SVI 口是一个虚拟接口，而 routed port 是物理接口。三层交换机支持将物理接口变成一个类似路由器物理接口的三层接口，具体的配置就是进入特定接口的配置模式后，使用 no switchport 命令，该接口就变成了一个 L3 的路由口，你可以给他配置 IP，就像操作路由器的一个以太网口哪样来操作它。

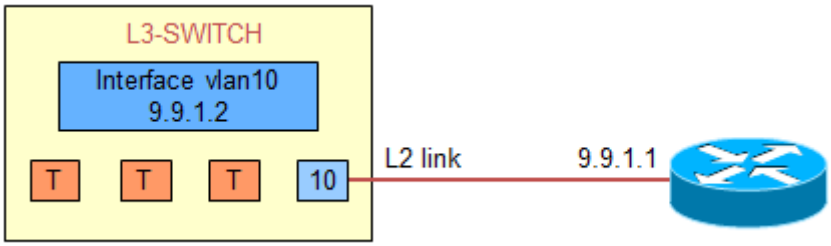
正是由于三层交换机支持这么多种类型的接口，使得三层交换网络的部署更加的灵活和可扩展。

下面我们来看一个三层交换部署的简单案例：

- **综合示例 1：**



- **综合示例 2：**



4.3 三层交换机基本配置

```
Switch(config)# ip routing
```

- 开启三层交换机的路由功能

```
Switch(config)# vlan 10
Switch(config-vlan)# name Classroom
```

- 创建 VLAN

```
Switch(config)# interface vlan 10
Switch(config-if)# ip address 192.168.10.254 255.255.255.0
Switch(config-if)# no shutdown
```

- 配置 VLAN 对应的 SVI 接口

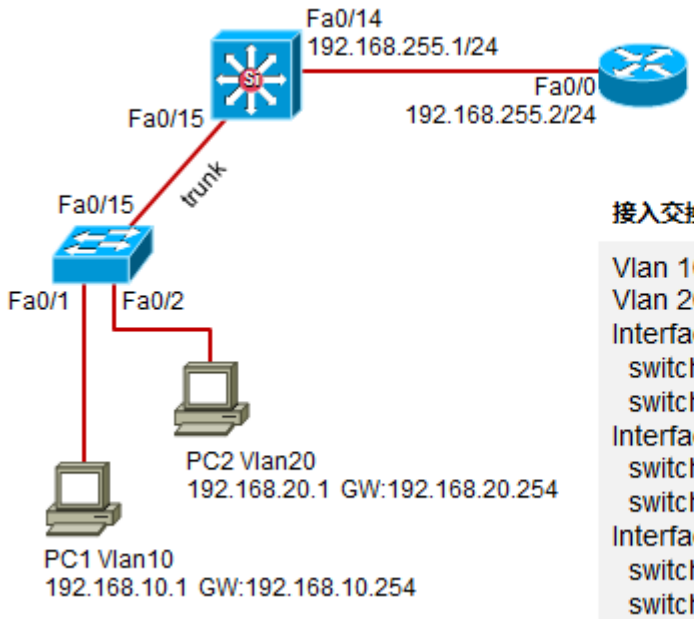
```
Switch(config)# interface fast 0/1
Switch(config-if)# no switchport
Switch(config-if)# ip address 192.168.255.1 255.255.255.0
Switch(config-if)# no shutdown
```

- 配置 no switchport（routed port）三层接口

```
Switch(config)# ip route 0.0.0.0 0.0.0.0 192.168.255.2
```

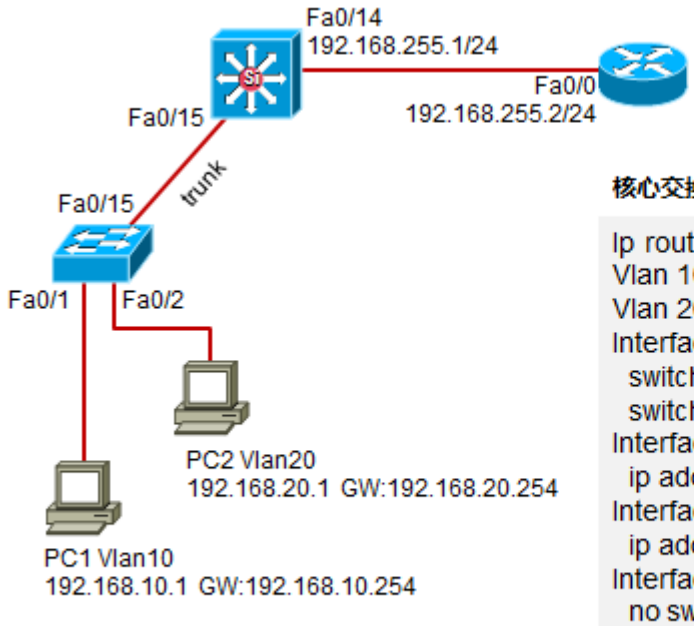
- 配置静态路由

配置示例：



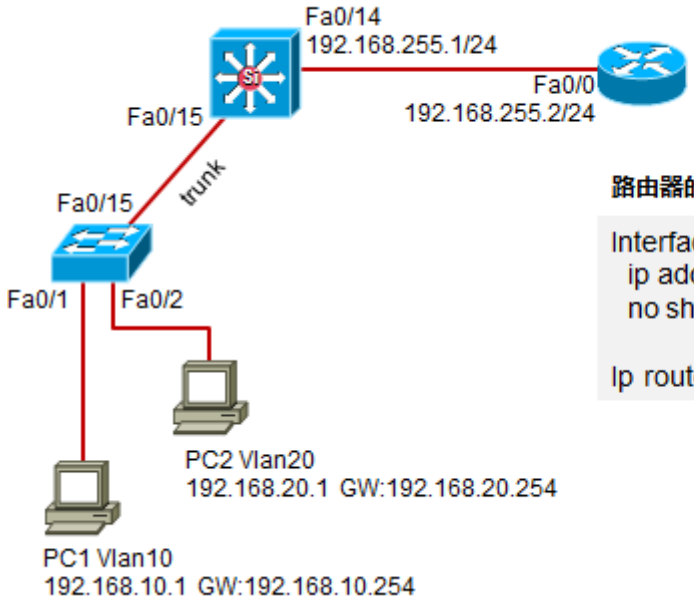
接入交换机的配置如下：

```
Vlan 10
Vlan 20
Interface fast0/1
  switchport mode access
  switchport access vlan 10
Interface fast0/2
  switchport mode access
  switchport access vlan 20
Interface fast0/15
  switchport trunk encapsulation dot1q
  switchport mode trunk
```



核心交换机的配置如下

```
Ip routing
Vlan 10
Vlan 20
Interface fast0/15
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
Interface vlan 20
  ip address 192.168.20.254 255.255.255.0
Interface fast0/14
  no switchport
  ip address 192.168.255.1 255.255.255.0
Ip route 0.0.0.0 0.0.0.0 192.168.255.2
```



路由器的配置

```
Interface fast0/0
  ip address 192.168.255.2 255.255.255.0
  no shutdown

Ip route 192.168.0.0 255.255.0.0 192.168.255.1
```

4.4 交换网络的管理

4.4.1 二层交换机的管理 VLAN

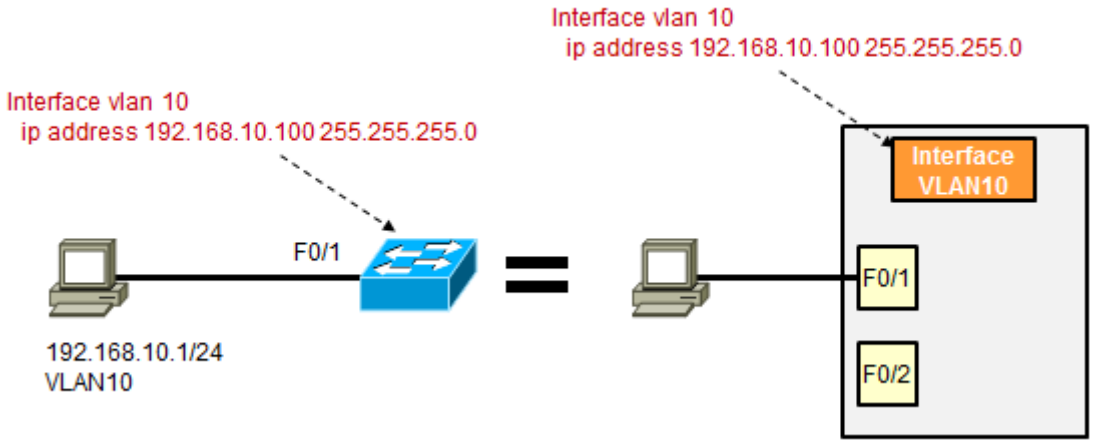
我们知道，在一个园区网中，数量最多的设备，一般而言应该是交换机（二层及三层交换机），其中又以二层交换机的数量居多，二层交换机在典型的三层网络结构中处于“接入层”，主要的任务是为终端的 PC 和用户提供接入，同时划分 VLAN 隔离广播域，再者运行 STP 来提供二层的防环机制。一个中小型的园区网，二层交换机的数量往往是上百台，这些设备在刚刚在客户现场被拆箱后，一般是由工程师现场用 Console 线缆一台台的调试的（不要惊讶，笔者的记录是一个下午的时间，个人完成近 100 台交换机的调试任务，当然，有集成商的兄弟帮忙安放设备、加点、贴标签）。这些交换机在调试好之后，就会被安装到客户现场的各个机房或者弱电间去，一切妥当后就正式上线运行了。

那么这就有一个问题，在设备上线后，如果我们要变更设备的配置、要管理这些交换机怎么办？难道要拿着笔记本电脑带着 console 线下到机房去现场调试么？这种屌丝级的方法完全不可理喻嘛，对了，可以通过 telnet 或者其他方式来远程管理。路由器上的 telnet 我们已经很熟悉了，只要是三层可达，就能 telnet 到路由器，那么接下去我们来看看，如何管理交换机。

这里我们讲的是二层交换机，大家都知道，二层交换机是无法识别三层报文的，它压根不会去看三层的 I 头，但是这不影响二层交换机自己拥有一个 IP 地址。在路由器上，我们是给路由器的物理接口配置 IP 地址，而二层

交换机，我们是在 VLAN 接口上配置 IP 地址，VLAN 接口我们也称为 SVI 交换式虚拟接口，是跟 VLAN 对应的一个逻辑的、虚拟的接口。一台二层交换机，只能够给一个 VLAN 接口分配 IP 地址。

考虑一个最简单的模型：

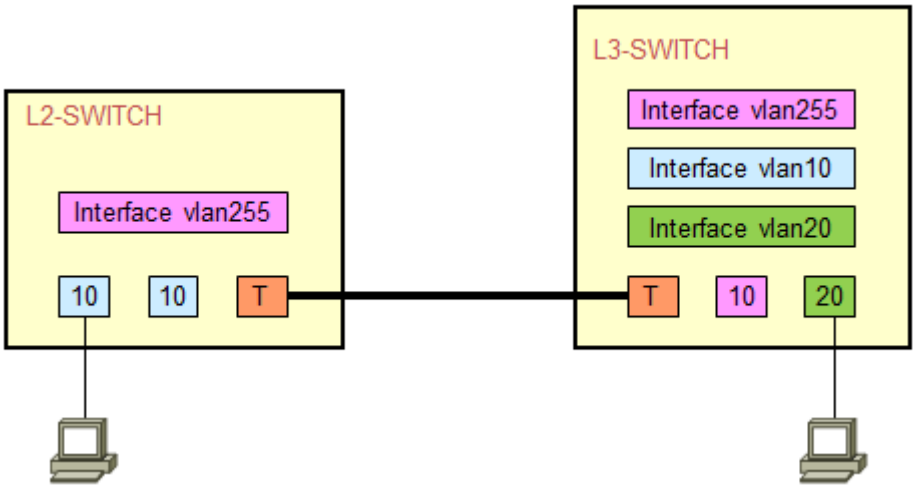


如上图左侧，PC 要想 Telnet 交换机，首先 PC 要能 ping 通交换机，其次交换机上要激活 VTY 并配置密码。那么我们在二层交换机上创建一个 VLAN10，将 F0/1 口划入 VLAN10，同时给二层交换机的 VLAN10 的逻辑接口配置一个 IP，与 PC 在同一个网段的 IP。这样一来，PC 就能访问到交换机了。可是问题来了，这样一来 PC 与交换机就在同一个网段，同一个 VLAN 了，万一下面有 PC 配置的 IP 地址与交换机有冲突那可就麻烦了，因此我们可以考虑给交换机划分一个单独的 VLAN，用于管理这些交换机，这个 VLAN 适用于整个交网络，统一的 VLAN 统一的 IP 规划，它就是管理 VLAN，因此管理 VLAN 并不是一个特定的 VLAN，更不是 VLAN1，这是许多人的误解。一般情况下，我们会使用一个较为“生僻的”VLAN ID 和 IP 编制，例如本实验中的 VLAN255，以及网段 192.168.255.0/24。

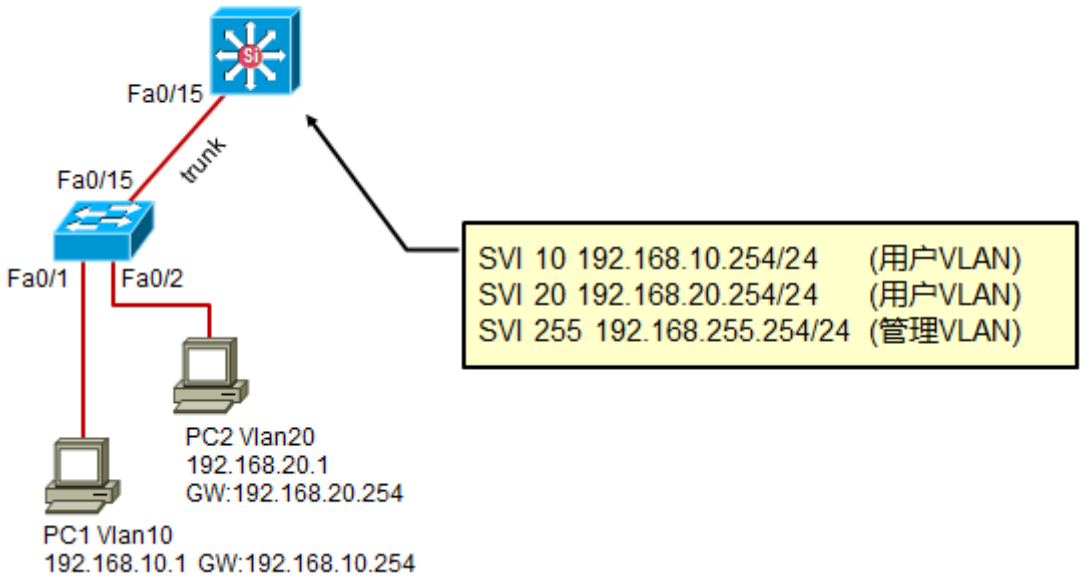
问题又来了，给交换机弄一个单独的设备管理 VLAN 固然可以起到与用户 VLAN 隔离的作用，但是这样一来用户不就无法访问到交换机了么？这就需要借助三层设备—例如路由器或者三层交换机了。与此同时，由于二层交换机没有路由功能，无法像路由器哪样拥有一个 IP 路由表，因此，你还需给交换机配置一个默认网关，就像你用路由器模拟 PC 那样哦亲。

关于具体的三层交换机管理 VLAN 的实验，请见《CCNA 实验手册 By 红茶三杯》

4.4.2 三层交换机环境下的管理 VLAN



配置示例：



接入交换机的配置：

```
Switch(config)# vlan 10
Switch(config)# vlan 20
Switch(config)# vlan 255
Switch(config)# interface fast0/1
Switch(config-if)# switchport access vlan 10
Switch(config)# interface fast0/2
Switch(config-if)# switchport access vlan 20
Switch(config)# interface fast0/15
```

```
Switch(config-if)# switchport mode trunk
Switch(config)# interface vlan 255
Switch(config-if)# ip address 192.168.255.1 255.255.255.0
Switch(config)# ip default-gateway 192.168.255.254     !!为接入交换机自身设置的网关
```

核心交换机的配置：

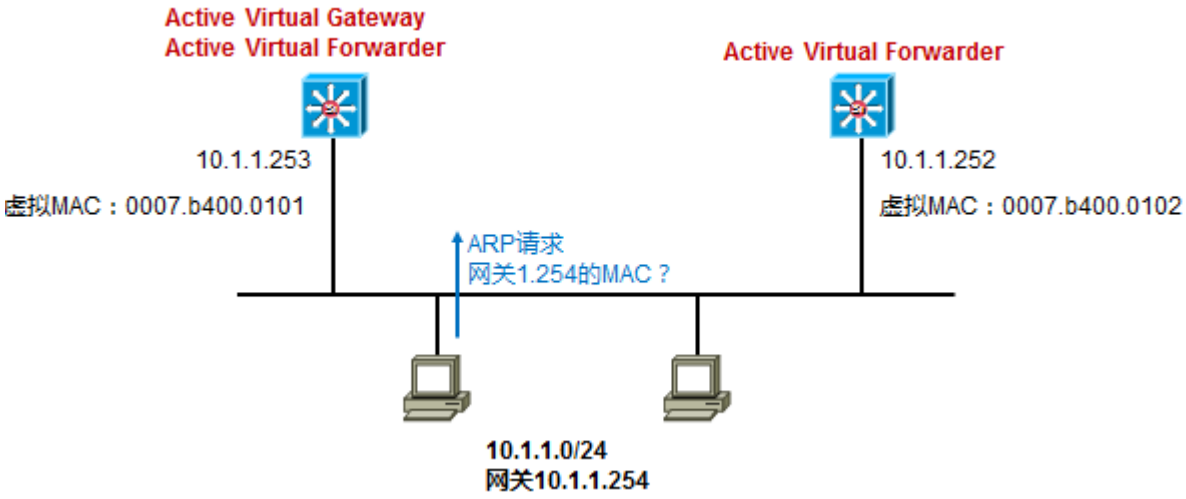
```
Switch(config)# vlan 10
Switch(config)# vlan 20
Switch(config)# vlan 255
Switch(config)# interface fast0/15
Switch(config-if)# switchport mode trunk
Switch(config)# interface vlan 10
Switch(config-if)# ip address 192.168.10.254 255.255.255.0
Switch(config)# interface vlan 20
Switch(config-if)# ip address 192.168.20.254 255.255.255.0
Switch(config)# interface vlan 255
Switch(config-if)# ip address 192.168.255.254 255.255.255.0
```

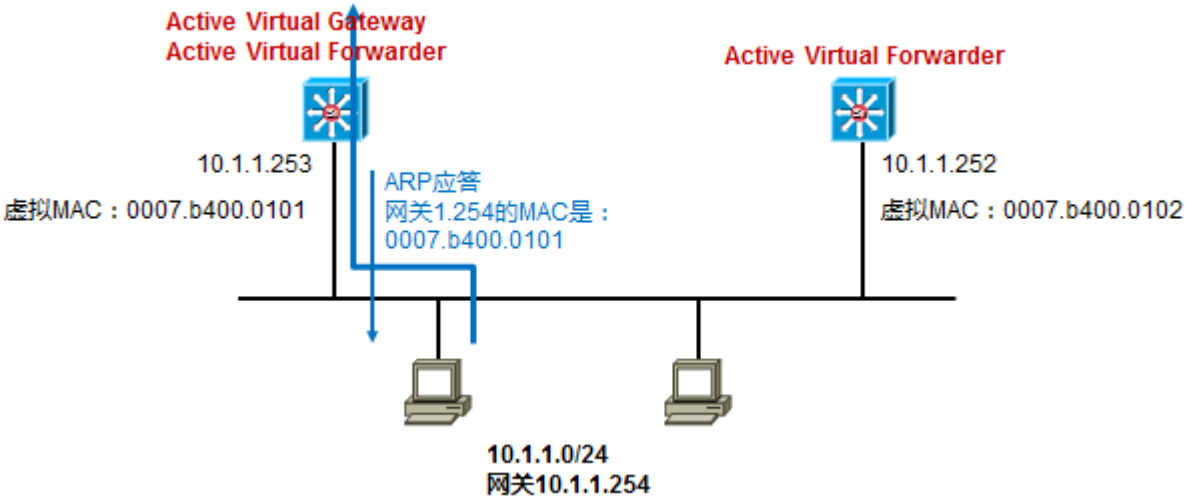
4.5 网关冗余技术

4.5.1 GLBP

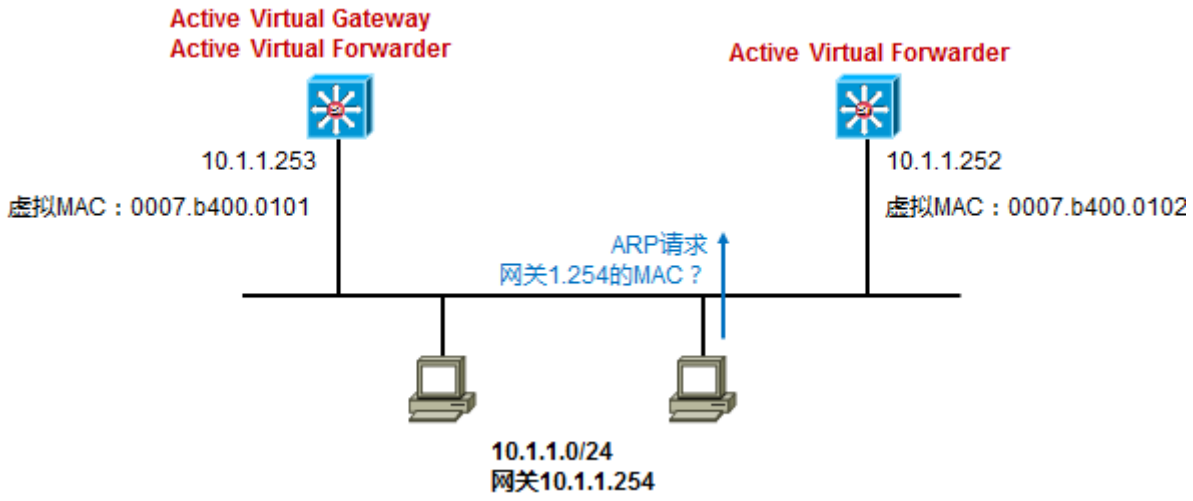
1. **协议概述**
   - 使得同一时间可使用多个网关，并且自动检测活跃网关。
   - 每组 GLBP 最多可以有 4 台作为 ip 默认网关的成员路由器，这些网关被称为 **AVF (active virtual forwarder)**
   - GLBP 自动管理虚拟 MAC 地址的分配，决定谁来负责处理转发的工作，这些功能由 **AVG(active virtual gateway 实现)**
   - 因此 AVG 负责分发虚拟 MAC，AVF 负责转发数据

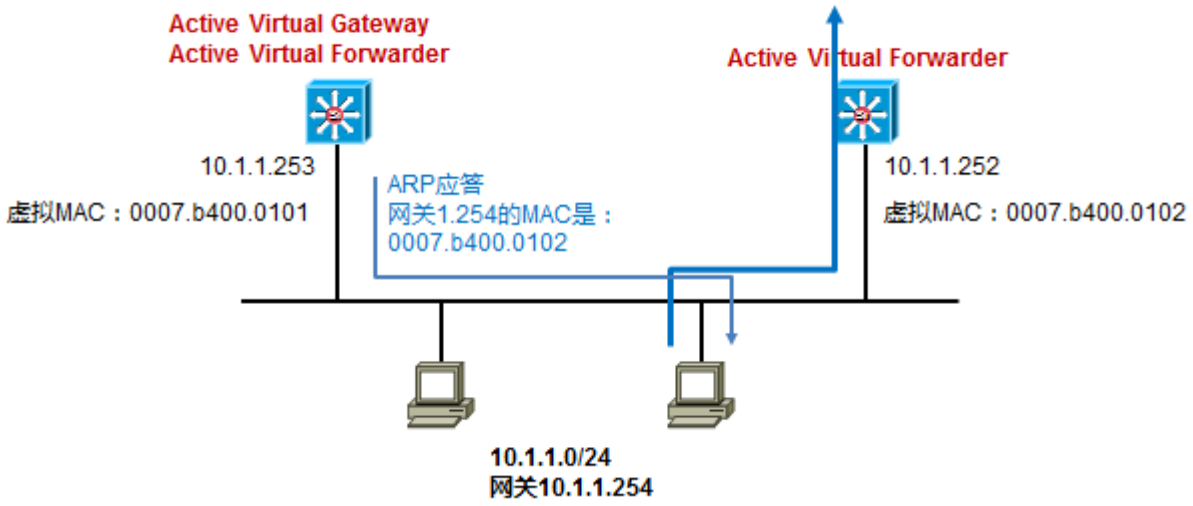
2. 工作机制



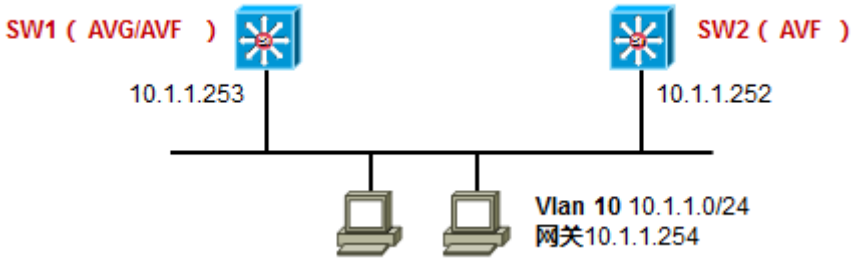


由 AVG 来负责响应 PC 对网关的 ARP 请求，同时 AVG 负责虚拟 MAC 的分配。AVG 可以根据不同的负载均衡模式灵活的对 PC 的 ARP 请求进行响应。





3. 配置及实现




```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
```

```
SW1(config-if)# glbp 1 priority 120    !! 优先级默认 100，优先级高的成为 AVG，优先级相等的情况下，最大 IP 的接口成为 AVG。默认“AVG 抢占”关闭
```

```
SW2(config)# interface vlan 10
SW2(config-if)# ip address 10.1.1.252 255.255.255.0
SW2(config-if)# glbp 1 10.1.1.254
```

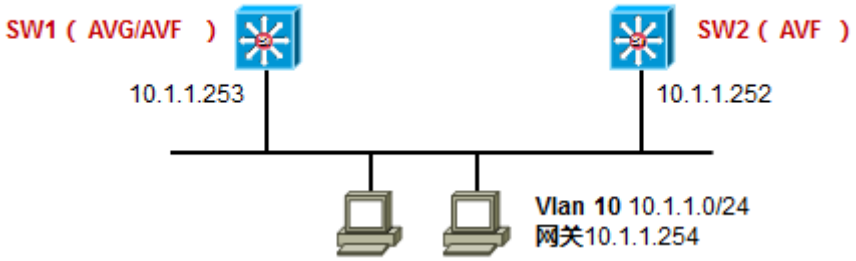
R1#show glbp

```
FastEthernet0/0 - Group 1
  State is Active
    2 state changes, last state change 00:07:34
  Virtual IP address is 10.1.1.254
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 1.280 secs
  Redirect time 600 sec, forwarder time-out 14400 sec
  Preemption disabled
  Active is local
  Standby is 10.1.1.252, priority 100 (expires in 7.572 sec)
  Priority 120 (configured)
  Weighting 100 (default 100), thresholds: lower 1, upper 100
  Load balancing: round-robin
  Group members:
    cc00.043c.0000 (10.1.1.253) local
    cc01.043c.0000 (10.1.1.252)
  There are 2 forwarders (1 active)
 There are 2 forwarders (1 active)
  Forwarder 1
    State is Active
      1 state change, last state change 00:07:24
    MAC address is 0007.b400.0101 (default)
    Owner ID is cc00.043c.0000
    Redirection enabled
    Preemption enabled, min delay 30 sec
    Active is local, weighting 100
```

```
    Arp replies sent: 3
  Forwarder 2
    State is Listen
    MAC address is 0007.b400.0102 (learnt)
    Owner ID is cc01.043c.0000
    Redirection enabled, 597.032 sec remaining (maximum 600 sec)
    Time to live: 14397.032 sec (maximum 14400 sec)
    Preemption enabled, min delay 30 sec
    Active is 10.1.1.252 (primary), weighting 100 (expires in 7.028 sec)
    Arp replies sent: 1
```

4. 负载均衡

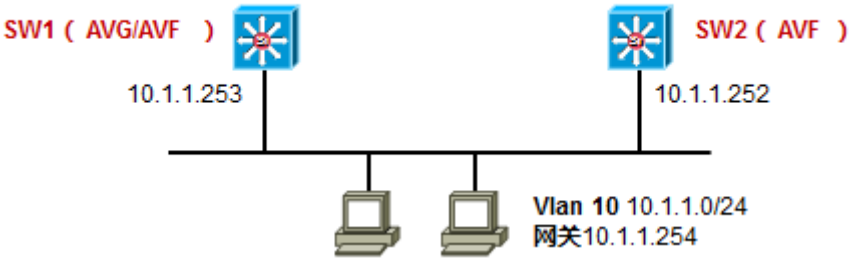
实验一：host-dependent




```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing host-dependent   !! 同一台 PC( 源 MAC )分配到固定的网关虚拟 MAC
```

- 负载均衡在 AVG 上修改即可生效，但是建议在所有路由器上都做配置
- 默认的负载均衡方式是 round-robin

实验二：weighted

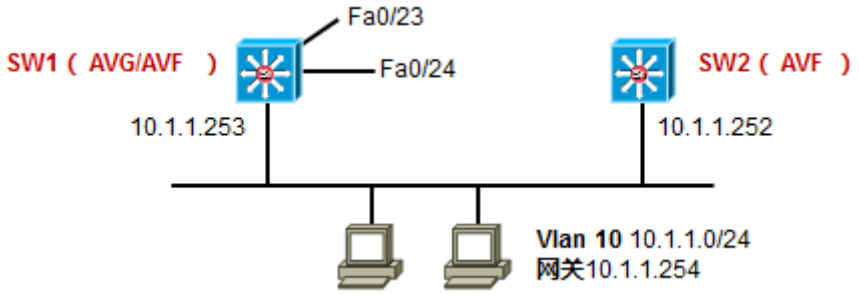



```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 200
```

```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 100
```

2:1

实验三：weighted 扩展




```
SW1(config)# track 10 interface fa0/23 line-protocol
SW1(config)# track 20 interface fa0/24 line-protocol
!
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 110 lower 85 upper 105
SW1(config-if)# glbp 1 weighting track 10 decrement 10
SW1(config-if)# glbp 1 weighting track 20 decrement 20
```

当weight低于85，则不再分配任何流量。只有流量再高于105，才会转发流量

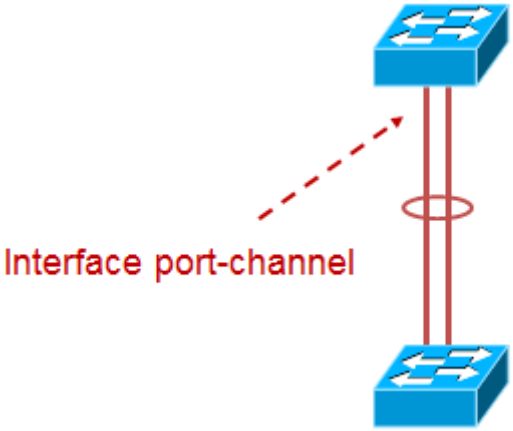
Track失败，weight减去一个值

4.5.2 HSRP

5 EtherChannel

1. 协议概述

- Solution to provide more bandwidth
- Logical aggregation of similar links
- Viewed as one logical link
- Provides load balancing and redundancy
- Supported for switch ports and routed ports



2. EtherChannel 协议

PAgP is a Cisco propretary protocol

LACP is IEEE 802.3ad standard

PAgP 的模式：

- On: channel member without negotiation（no protocol）
- Desirable: actively ask if the other side can/will
- Auto: passively wait for other side to ask
- Off: EtherChannel not configured on interface

on --- **Channel** ---- on

On/Auto/Desirable --- No Channel ---- Off

Auto/Desirable --- **Channel** ---- Desirable

Auto --- No Channel ---- Auto

LACP 的模式：

- On: channel member without negotiation（no protocol）
- Active: actively ask if the other side can/will
- Passive: passively wait for other side to ask
- Off: EtherChannel not configured on interface

on --- **Channel** ---- on

On/Passive/Active --- No Channel ---- Off

Passive/Active --- **Channel** ---- Active

Passive/On --- No Channel ---- Passive

3. 实施要点

- Port-channel 接口一旦建立完成后，就形成了一个逻辑的接口，后续针对该接口的配置在 port-channel 逻辑接口中完成
- Port-channel 接口不能成为 SPAN 的目的接口
- 隶属于一个 port-channel 的物理接口需有相同的如下配置
  - 相同的 speed 和 duplex
  - 相同的接口模式（access、trunk）
  - 如果是 trunk 模式，那么 native vlan 及 allowed vlan 需相同
  - 如果是 access 模式，所属 vlan 需相同

4. **配置及实现**

1) **选择用于 Channel 的端口**

2) **选择 PAgP 或 LACP**

3) **在接口上配置 channel-group**

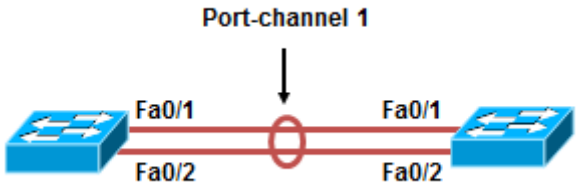
a) 设置 channel-group ID

b) 根据特定的协议，选择接口模式

4) **完成上述步骤后，逻辑的 etherchannel 接口就建立好了**

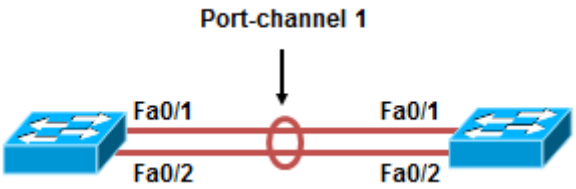
可以进一步对这个逻辑的 etherchannel 接口进行配置

- **二层捆绑**




```
Sw1(config)#interface range fastEthernet 0/1 – 2   //进入接口范围
Sw1(config-if-range)#switchport                    //将接口配置为二层接口
Sw1(config-if-range)#switchport trunk encapsulation dot1q
        //trunk 封装协议为 dot1q
Sw1(config-if-range)#switchport mode trunk
        //设置接口模式为 Trunk 模式
Sw1(config-if-range)#channel-protocol pagp/lacp
Sw1(config-if-range)#channel-group 1 mode desirable
        //配置 etherchannel，ID 为 1，模式为 desirable
//查看命令：Show etherchannel 1 summary
```

- **三层捆绑**

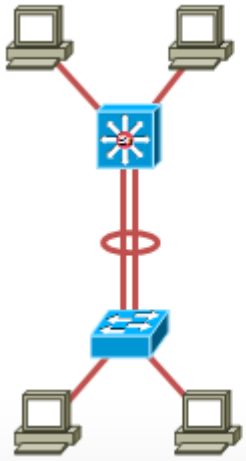



```
Sw1(config)#interface range fastEthernet 0/1 – 2
```

```
Sw1(config-if-range)#no switchport //配置 L3 接口
Sw1(config-if-range)#no ip address
Sw1(config-if-range)#channel-group  1 mode desirable
Sw1(config-if-range)#no shutdown
Sw1(config)#interface port-channel  1  //进入 port-channel 组 1 配置 IP
Sw1(config-if)#ip address  172.16.10.1 255.255.255.0
Sw1(config-if)#no shu
查看命令：Show eth 1 summary / show ip route / show ip int brief
```

5. **负载均衡**

- **EtherChannel 支持在同一个 port-channel 的链路中执行负载均衡**
- **负载均衡动作可以基于 MAC、端口、IP（源 IP、目的 IP 或两者）**
- **默认的行为：源目 IP 地址对（src-dst-ip）**



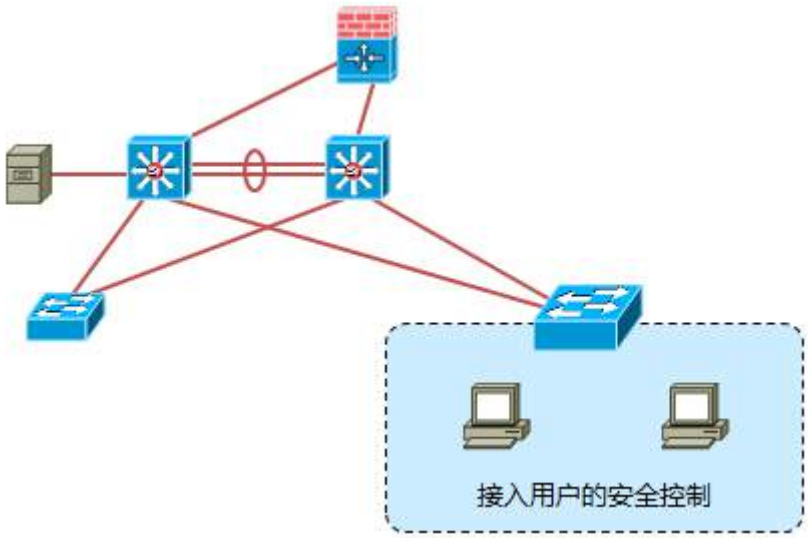
```
Switch(config)# port-channel load-balance type
```

```
Switch# show etherchannel load-balance
```

6 交换网络安全

6.1 Port-Security

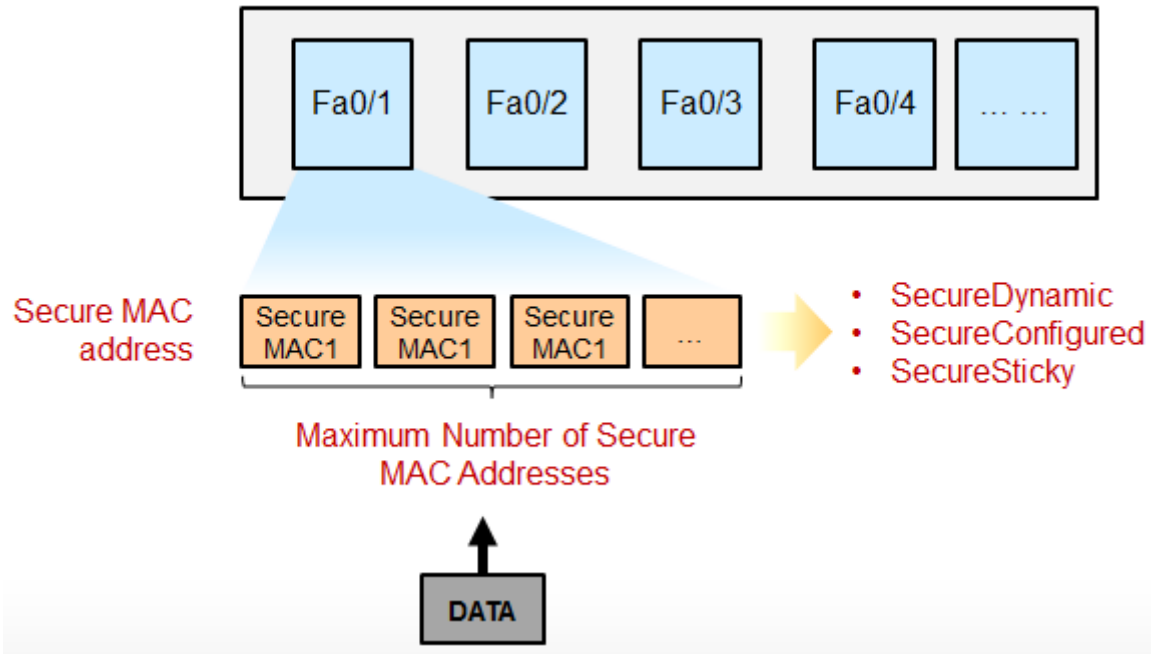
6.1.1 Port-Security 概述



在部署园区网的时候，对于交换机，我们往往有如下几种特殊的需求：

- 限制交换机每个端口下接入主机的数量（MAC 地址数量）
- 限定交换机端口下所连接的主机（根据 IP 或 MAC 地址进行过滤）
- 当出现违例时间的时候能够检测到，并可采取惩罚措施

上述需求，可通过交换机的 Port-Security 功能来实现：



6.1.2 理解 Port-Security

1. **Port-Security 安全地址：secure MAC address**

在接口上激活 Port-Security 后，该接口就具有了一定的安全功能，例如能够限制接口（所连接的）的最大 MAC 数量，从而限制接入的主机用户；或者限定接口所连接的特定 MAC，从而实现接入用户的限制。那么要执行过滤或者限制动作，就需要有依据，这个依据就是**安全地址** – secure MAC address。

安全地址表项可以通过让使用端口**动态学习到的 MAC（SecureDynamic），或者是手工在接口下进行配置（SecureConfigured），以及 sticy MAC address（SecureSticky） 三种方式进行配置。**

当我们将接口允许的 MAC 地址数量设置为 1 并且为接口设置一个安全地址 那么这个接口将只为该 MAC 所属的 PC 服务，也就是源为该 MAC 的数据帧能够进入该接口。

2. **当以下情况发生时，激活惩罚（violation）：**

- 当一个激活了 Port-Security 的接口上，MAC 地址数量已经达到了配置的最大安全地址数量，并且又收到了一个新的数据帧，而这个数据帧的源 MAC 并不在这些安全地址中，那么启动惩罚措施
- 当在一个 Port-Security 接口上配置了某个安全地址，而这个安全地址的 MAC 又企图在同 VLAN 的另一个 Port-Security 接口上接入时，启动惩罚措施

当设置了 Port-Security 接口的最大允许 MAC 的数量后，接口关联的安全地址表项可以通过如下方式获取：

- 在接口下使用 switchport port-security mac-address 来配置静态安全地址表项
- 使用接口动态学习到的 MAC 来构成安全地址表项
- 一部分静态配置，一部分动态学习

当接口出现 up/down，则所有动态学习的 MAC 安全地址表项将清空。而静态配置的安全地址表项依然保留。

3. **Port-Security 与 Sticky MAC 地址**

上面我们说了，通过接口动态学习的 MAC 地址构成的安全地址表项，在接口出现 up/down 后，将会丢失这些通过动态学习到的 MAC 构成的安全地址表项，但是所有的接口都用 switchport port-security mac-address 手工来配置，工作量又太大。因此这个 sticky mac 地址，可以让我们将这些动态学习到的 MAC 变成“粘滞状态”，可以简单的理解为，我先动态的学，学到之后我再将你粘起来，形成一条”静态 “（实际上是 SecureSticky）的表项。

在 up/down 现象出现后仍能保存。而在使用 wr 后，这些 sticky 安全地址将被写入 start-up config，即使设备重启也不会被丢失。

6.1.3 默认的 Port-Security 配置

Port-Security	默认关闭

默认最大允许的安全 MAC 地址数量	1
惩罚模式	shutdown（进入 err-disable 状态），同时发送一个 SNMP trap

6.1.4 Port-Security 的部署注意事项

1. Port-Security 配置步骤

1) 在接口上激活 Port-Security
   Port-Security 开启后，相关参数都有默认配置，需关注
2) 配置每个接口的安全地址（Secure MAC Address）
   可通过交换机动态学习、手工配置、以及 stciky 等方式创建安全地址
3) 配置 Port-Security 惩罚机制
   默认为 shutdown，可选的还有 protect、restrict
4) （可选）配置安全地址老化时间

2. 关于被惩罚后进入 err-disable 的恢复：

如果一个 psec 端口由于被惩罚进入了 err-disable，可以使用如下方法来恢复接口的状态：

- 使用全局配置命令：err-disable recovery psecure-violation
- 手工将特定的端口 shutdown 再 noshutdown

3. 清除接口上动态学习到的安全地址表项

使用 clear port-security dynamic 命令，将清除所有 port-security 接口上通过动态学习到的安全地址表项
使用 clear port-security sticky 命令，将清除所有 sticky 安全地址表项
使用 clear port-security configured 命令，将清除所有手工配置的安全地址表项
使用 clear port-security all 命令，将清除所有安全地址表项
使用 show port-security address 来查看每个 port-security 接口下的安全地址表项

4. 关于 sticky 安全地址

Sticky 安全地址，是允许我们将 Port-Security接口通过动态学习到的 MAC 地址变成“粘滞”的安全地址，从而不会由于接口的 up/down 丢失。然而如果我们希望在设备重启之后 ,这个 sticky 的安全地址表项仍然存在，那么就需要 wr 一下。将配置写入 start-up config 文件。Sticky 安全地址也是一个简化我们管理员操作的一个很好的工具，毕竟现在不用再一条条的手工去绑了。

5. port-security 支持 private vlan

6. **port-security 支持 802.1Q tunnel 接口**
7. **port-security 不支持 SPAN 的目的接口**
8. **port-security 不支持 etherchannel 的 port-channel 接口**
9. **在 CISCO IOS 12.2(33)SXH 以及后续的版本，我们可以将 port-security 及 802.1X 部署在同一个接口上。而在此之前的软件版本：**
   - 如果你试图在一个 port-security 接口上激活 8021.X 则会报错，并且 802.1X 功能无法开启
   - 如果你试图在一个 802.1X 接口上激活 port-security 则也会报错，并且 port-security 特性无法开启

10. **Port-Security 支持 nonegotiating trunk 接口**
    - Port-Security 支持在如下配置的 trunk 上激活

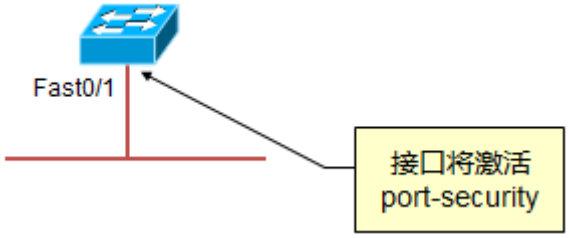
```
switchport
switchport trunk encapsulation ?
switchport mode trukn
switchport nonegotiate
```

    - If you reconfigure a secure access port as a trunk, port security converts all the sticky and static secure addresses on that port that were dynamically learned in the access VLAN to sticky or static secure addresses on the native VLAN of the trunk. Port security removes all secure addresses on the voice VLAN of the access port.
    - If you reconfigure a secure trunk as an access port, port security converts all sticky and static addresses learned on the native VLAN to addresses learned on the access VLAN of the access port. Port security removes all addresses learned on VLANs other than the native VLAN.

11. **Flex links 和 Port-Security 互不兼容**

6.1.5 Port-security 的配置

1. **激活 Port-Security (在 access 接口上)**



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
```

接口的 Port-Security 特性一旦激活后，默认的最大安全地址个数为 1，也就是说，在不进行手工配置安全地址的情况下，这个接口将使用其动态学习到的 MAC 作为安全地址，并且，这个接口相当于被该 MAC（所属的设备）独占。而且默认的 violation 是 shutdown

SW1#show port-security interface f0/1

```
Port Security              : Enabled
Port Status                : Secure-up        !!接口目前的状态是 up 的
Violation Mode             : Shutdown         !!违例后的惩罚措施，默认为 shutdown
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1                !!最大安全地址个数，默认为 1
Total MAC Addresses        : 0
Configured MAC Addresses   : 0                !!手工静态配置的安全 MAC 地址，这里没配
Sticky MAC Addresses       : 0                !!sticky 的安全地址，这里没有
Last Source Address:Vlan   : 00b0.1111.2222:10   !!最近的一个安全地址+vlan
Security Violation Count   :  0               !!该接口历史上出现过的违例次数
```

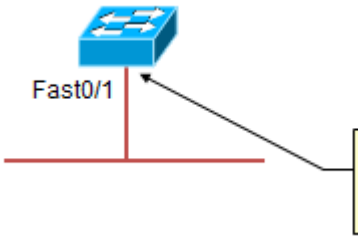
这个时候，如果另一台 PC 接入到这个端口上，那么该 port-security 接口将会收到一个新的、非安全地址表项中的 MAC 地址的数据帧，于是触发的违例动作，给接口将被 err-disable 掉。同时产生一个 snmp trap 消息，另外，接口下，Security Violation Count 将会加 1

2. 激活 Port-Security（在 trunk 接口上）

- 注意该trunk接口需为DTP非协商模式
- 同时由于默认的port-security最大允许的安全地址表项为1且violation为shutdown，因此在激活port-security前强烈建议先修改最大允许的安全地址数量

```
Switch(config)# interface fast0/24
Switch(config-if)# switchport
Switch(config-if)# switchport trunk encapsulation {dot1q | isl}
Switch(config-if)# switchport mode trunk
Switch(config-if)# switchport nonegotiate
Switch(config-if)# switchport maximum ?
Switch(config-if)# switchport port-security
```

3. Port-Security violation 惩罚措施




```
Switch(config)# interface fast0/1
Switch(config-if)# switchport port-security violation {protect | restrict | shutdown}
```

- Protect　　仅丢弃非法的数据帧
- Restrict　　丢弃非法的数据帧，同时产生一个syslog消息
- Shutdown　　将端口置为err-disable，接口不可用，同时产生一个syslog消息

默认的 violation 是 shutdown。如果是 protect，那么惩罚就会温柔些，对于非法的数据帧（例如数据帧的源 MAC 不在安全地址表项中的、且安全地址已经达到最大数），这些非法数据将仅仅被丢弃，合法数据照常转发，同时不会触发一个 syslog 消息，另外接口下的“Security Violation Count”也不会加 1。而如果是 restrict，那么非法数据被丢弃，同时触发一个 syslog 消息，再者，Security Violation Count 加 1，合法的数据照常转发。

4. 配置 Port Security Rate Limiter

（注意，在 6509 交换机，truncated switching 模式下不支持该功能）

在交换机接口上开启 Port-Security 是会耗费资源的，Port-Security 会检测每一个进入接口的数据帧，以判断流量是否合法，或者是否存在违例行为。当一个安全接口设置的 violation 为 shutdown 的时候，该接口在违

例后触发惩罚机制，进入 err-diasble 状态，这样可以有效的方式有效的防止交换机由于处理违例事件导致交换机的 CPU 利用率过高。然而 protect 和 restict 的惩罚措施，则不会将端口关闭，端口依然可用，那么这就可能导致在违例事件发生的情况下交换机的 CPU 利用率飙高(例如大量的非法数据涌入)。因此当使用 protect 和 restrict 这两种违例惩罚措施事，可以通过 Port-Secuirty rate limiter 来防止 CPU 飙高。

Switch(config)# mls rate-limit layer2 port-security rate_in_pps [burst_size]

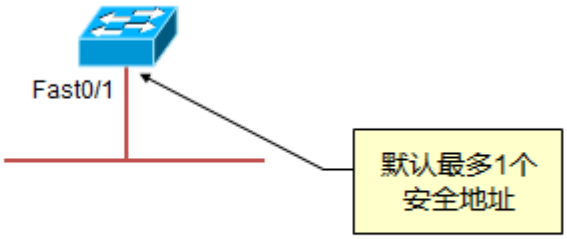
关于 rate_in_pps 参数：

- **范围是 10- 1000000**
- **没有默认值**
- **值设置的越低，对 CPU 的保护程度就越高，这个值对惩罚措施发生前、后都生效，当然这个值也不能设置的过低，至少要保障合法流量被处理吧。一般低于 1000 就差不多。**

关于 burst-size 参数：

- **范围是 1-255**
- **默认是 10，这个默认值一般就够用了。**

5. 配置 Port-Security 最大允许的安全地址数量




```
Switch(config)# interface fast0/1
Switch(config-if)# switchport port-security maximum ?
```

最大安全地址数量，不同的软件平台允许的上限值有所不同，但是默认都是 1。

在 trunk 口上，前面说了，也是可以激活 port-security 的，而在 trunk 口上配置最大安全地址数量，可以基于接口配置（对所有 VLAN 生效），也可以基于 VLAN 进行配置。如下：

switchport port-security maximum 1

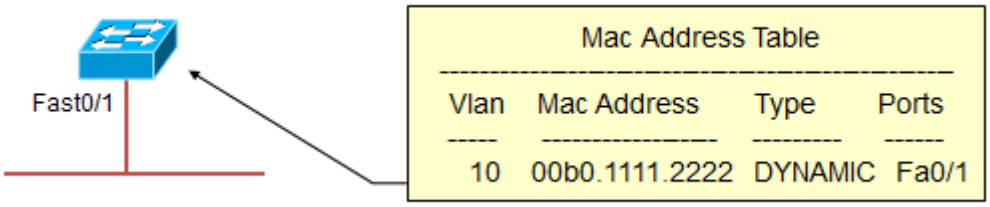
switchport port-security maximum 1 vlan 10,20,30 !!可以关联多个 VLAN

6. 在 port-security 接口上手工配置安全地址


```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum  3
Switch(config-if)# switchport port-security mac-address xxxx  [vlan id]
```

- 上述配置中，最大安全地址数设置为 3，然后使用手工配置了一个安全地址，那么剩下 2 个，交换机可以通过动态学习的方式来构建安全地址。
- 在 trunk 接口上手工配置安全地址，可关联 vlan 关键字。如果在 trunk 接口上手工配置安全地址，没有关联 vlan 关键字，那么该安全地址将被关联到 trunk 的 native vlan 上

7. 在 port-security 接口上使用 sticky MAC 地址



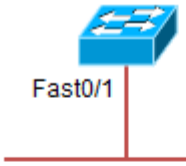

```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security mac-address sticky
```

我们知道，构成安全地址表项的方式有好几种，其中一种是使用 switchport port-security mac 来手工配置，但是这种方式耗时耗力，更需要去 PC 上抄 MAC，工作成本比较高。而另一种构成安全地址的方式是让交换机使用动态学习到的 MAC，然而这些安全地址在接口一旦 up/down 后，将丢失，更别说重启设备了。因此可以使用 sticky mac 的方式，这种方式激活后，交换机将动态学习到的 MAC “粘起来”，具体的动作很简单，就是在动态学习到 MAC（例如一个 00b0.1111.2222）后，如果我激活了 sticiky MAC address，则在接口下自动产生一条命令：

```
interface FastEthernet0/1
 switchport access vlan 10
```

```
switchport mode access
switchport port-security
switchport port-security mac-address sticky
switchport port-security mac-address sticky 00b0.1111.2222          !! 自动产生
```

这样形成的安全地址表项（是 SecureSticky 的），即使在接口翻动，也不会丢失。在者如果 wr 保存配置，命令写入 config.text，那么设备即使重启，安全地址也不会丢失。




```
SW1#show port-security address
                Secure Mac Address Table
------------------------------------------------------------------------
Vlan   Mac Address       Type            Ports   Remaining Age
                                                      (mins)
-----  -----------       ----------      -----   -------------
  10   00d0.1111.2222    SecureSticky    Fa0/1   -
```

```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security mac-address sticky
Switch(config-if)# switchport port-security mac-address sticky 00b0.1111.2222
```

当在接口上激活了 port-security mac-address sticky，那么：

- 该接口上所有通过动态学习到的 MAC，将被转成 sticky mac address 从而形成安全地址
- 接口上的静态手工配置的安全地址不会被转成 sticky mac address
- 通过 voice vlan 动态学习到的安全地址不会被转成 sticky mac address
- 命令配置后，新学习到的 MAC 地址，也是 sticky 的

当此时又敲入 no port-secuirty mac-address sticiky ,则所有的 sticky 安全地址条目都变成动态的安全地址条目（SecureDynamic）

8. 配置安全地址老化时间

配置的命令比较简单：

```
Switch(config-if)# switchport port-security aging type {absolute | inactivity}
```

配置老化时间的类型，如果选择 absolute，也就是绝对时间，需要搭配 aging time 命令设定老化时间的具体值，命令一旦敲下去后，所有的通过动态学习的 MAC 构建的安全地址表项将开始以 aging time 倒计时。如果是 inactivity 关键字，则只在该动态安全地址表项不活跃的时候（譬如主机离线了或者挂掉了）才开始倒计时。

```
Switch(config-if)# switchport port-security aging time ?
```

设定老化时间

```
Switch(config-if)# switchport port-security aging static
```

使用前面两条命令，老化时间是不会影响那些使用静态手工配置的安全地址表项的，当然 sticky 表项也不会受影响，这些表项都是永不老化的，但是如果搭配上上面这条命令，则手工配置的安全地址表项也受限于老化时间了。

不过对于 sticky 表项，则始终不会激活 aging time，它是不会老化的。

- **示例 1：**


```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 3
Switch(config-if)# switchport port-security aging type absolute
Switch(config-if)# switchport port-security aging time 50
```

将安全地址老化时间设置为 50min。针对动态学习到的 MAC 构成的安全地址有效

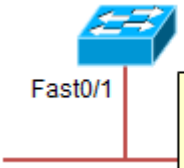
50min 是一个绝对时间，配置完成后开始倒计时，无论该 MAC 是否依然活跃，都始终进行倒计时

- **示例 2：**


```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 3
Switch(config-if)# switchport port-security aging type inactivity
Switch(config-if)# switchport port-security aging time 50
```

针对动态学习到的 MAC 构成的安全地址有效，如果该 MAC 在 50min 内一直处于失效状态（例如主机离线了），那么该安全地址在 aging time 超时后被清除

- **示例 3：**



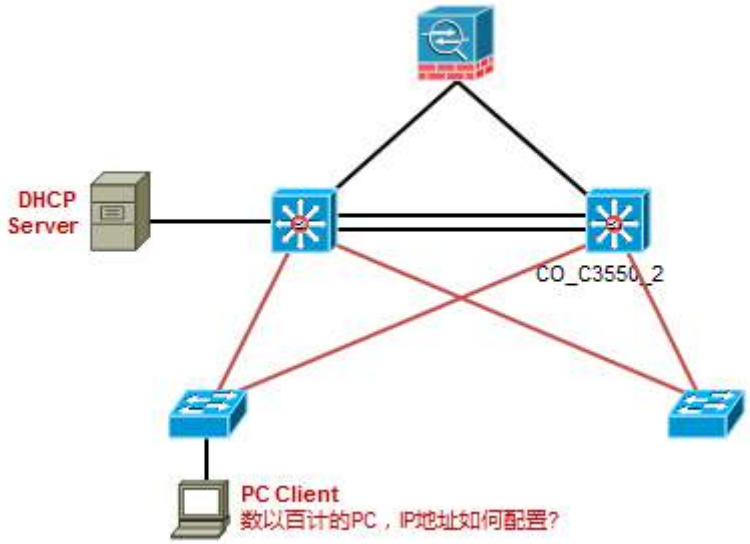

```
SW1#sh port-security address
                Secure Mac Address Table
------------------------------------------------------------
Vlan    Mac Address       Type                  Ports   Remaining Age
                                                        (mins)
----    -----------       --------------------  -----   -------
  10    00b0.9999.9999    SecureConfigured      Fa0/1      47
  10    00d0.2222.2222    SecureDynamic         Fa0/1      47
```

注意，上述两种配置方式，对手工配置 switchport port-security mac-address 00b0.9999.9999 的安全地址无效。也就是采用上述方法配置的静态安全地址表项永不超时。

- 如果增加 switchport port-security aging static 命令，则手工静态配置的安全地址也的 aging time 也开始计时
- 注意，对于 sticky mac address，安全地址的老化时间无效

6.2 DHCP

6.2.1 DHCP 概述



- Dynamic Host Configuration Protocol
- 用于动态地址分配
- 基于 UDP 协议 端口 67 及 68
- bootPC:67（客户端端口号）；bootPS：68（服务端端口号）

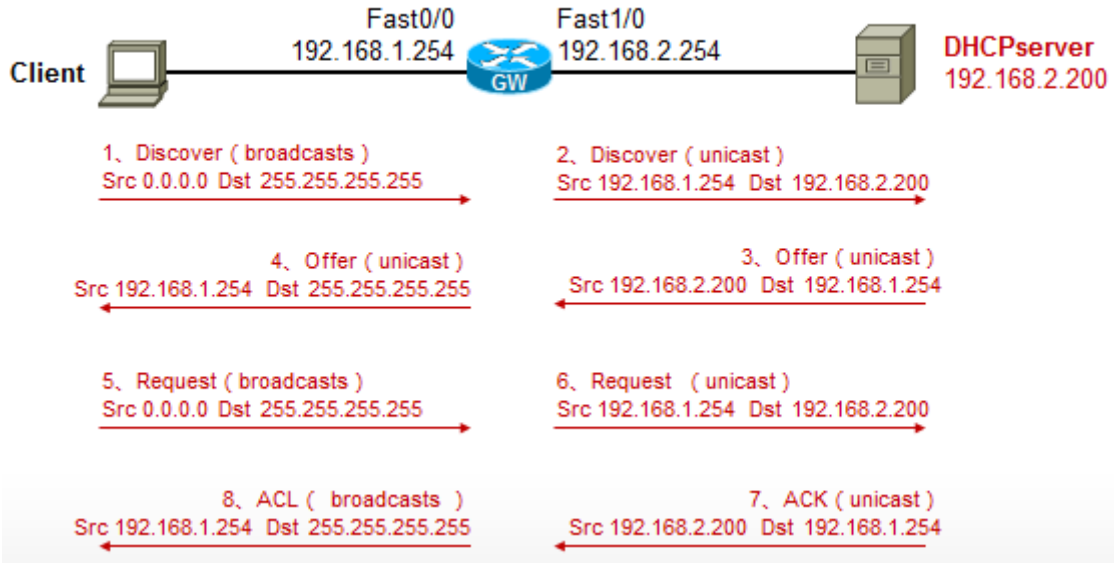
6.2.2 协议机制

1. 报文交互过程

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover
2	1.984000	192.168.1.254	255.255.255.255	DHCP	DHCP Offer
3	2.033000	0.0.0.0	255.255.255.255	DHCP	DHCP Request
4	2.129000	192.168.1.254	255.255.255.255	DHCP	DHCP ACK

注意，不同的产品，数据交互过程有所不同，上述过程是在 CISCO 设备（服务端）上测试的结果。

2. DHCP 中继



6.2.3 配置及实现

1. 基本配置

Router(config)#server *dhcp*

开启 DHCP 服务。By default, the Cisco IOS DHCP server and relay agent features are enabled on your router

Router(config)#ip dhcp pool *[pool name]*

定义 DHCP 地址池，一个网段对应一个地址池

Router(config-dhcp)#network *[network address][subnet mask]*

定义地址池关联的网段

Router(config-dhcp)#default-router *[host address]*

定义分配给客户端的网关 IP

Router(config)#ip dhcp excluded-address *172.16.1.100 172.16.1.103*

将一个或多个地址排除在地址池之外（如网关 IP 等），以避免分配给客户端

2. 查看及验证

```
show ip dhcp database
```

Displays recent activity on the DHCP database

```
show ip dhcp server statistics
```

Shows count information about statistics and messages sent and received

```
show ip route dhcp
```

Displays routes added to the routing table by DHCP

```
Show ip dhcp binding
```

Show dhcp binding on DHCP server

3. 验证

```
debug dhcp detail
```

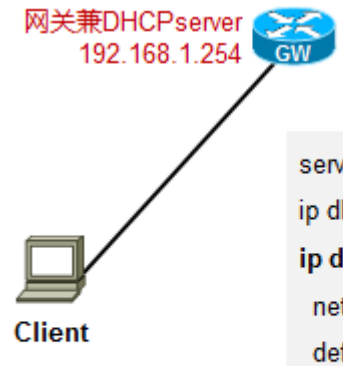
Enables debugging on the DHCP client

```
debug ip dhcp server {events | packets | linkage}
```

Enables debugging on the DHCP server

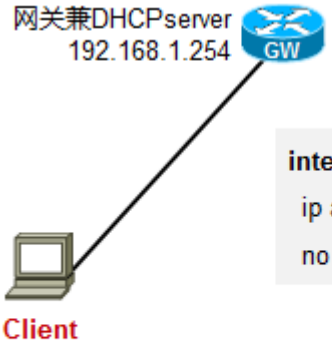
6.2.4 配置示例

1. 基础实验



在CISCO IOS 路由器上部署DHCP服务

```
service dhcp
ip dhcp excluded-address 192.168.1.254
ip dhcp pool Userpool
 network 192.168.1.0 255.255.255.0
 default-router 192.168.1.254
 domain-name global.com
 dns-server 192.168.200.200 192.168.200.201
 netbios-name-server 192.168.200.198 192.168.200.199
```



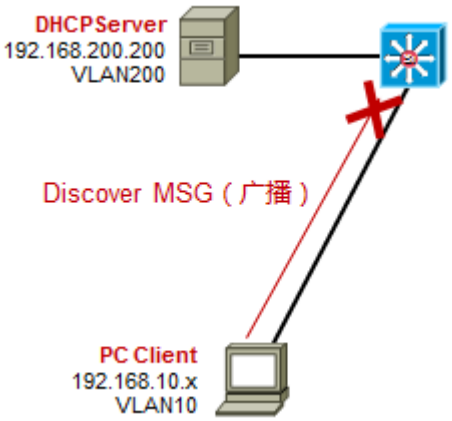
在CISCO IOS 路由器上部署DHCP客户端

```
interface fast0/0
 ip address dhcp
 no shutdown
```

在PC客户端上配置DHCP自动获取

直接在对应网卡上修改即可

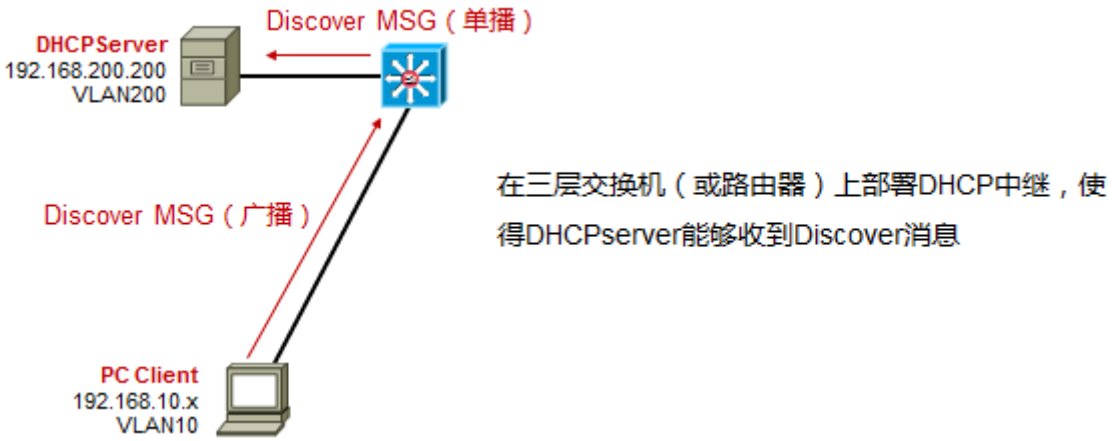
2. DHCP 中继实验



PC客户端与DHCPserver不在同一个网段

PC发出的广播DHCP discover包到达三层设备，

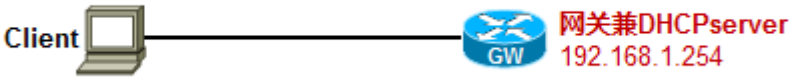
被三层接口丢弃，因此数据无法到达DHCP服务器



注意，中继在配置的时候：

- Ip helper-address 命令是配置在收到 client 发出的广播 discovery 报文的那个三层接口上 ,或者说 ,是“阻挡”这个广播 DHCP 报文的接口上的，例如，如果是三层交换机用 access 接口连接 PC，那么就应该配置在该 VLAN 的 SVI 接口上。
- Ip helper-address 命令只需配置在第一个收到 client 发出的广播 discovery 报文的那个三层接口上 ,在此之后这些广播的 DHCP 报文将被中继成单播包，就是走 IP 路由了。
- 注意中继设备需开启 DHCP 服务： service dhcp
- 注意 DHCPserver 与中继接口之间必须三层能通 ,因为中继后的单播包 ,源地址是配置 ip helper-address 的那个中继接口 ,所以 DHCPserver 在回包的时候 ,报文的目的 IP 就是这个配置 ip helper-address 的接口的接口 IP。

3. DHCP 静态地址绑定



- Manual bindings are IP addresses that have been manually mapped to the MAC addresses of hosts that are found in the DHCP database. Manual bindings are stored in NVRAM on the DHCP server. Manual
- bindings are just special address pools. There is no limit on the number of manual bindings, but you can only configure one manual binding per host pool.

配置命令如下：

```
ip dhcp pool BIND
  host 192.168.1.111 255.255.255.0
  client-identifier unique-identifier
```

- DHCP clients require client identifiers. The unique identification of the client is specified in dotted

hexadecimal notation, for example, 01b7.0813.8811.66, where 01 represents the Ethernet media type.

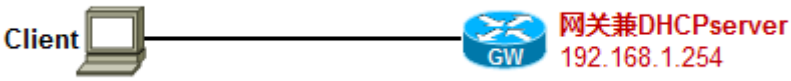
- 也可以在服务器上通过 show ip dhcp bingding 抓取客户端的 client-identifier

如果客户端发出的 DHCP discovery 报文中不包含 client-identifier，则可使用 hardware address 来识别终端设备。配置命令如下：

```
ip dhcp pool BIND
  host 192.168.1.111 255.255.255.0
  hardware-address hardware-address type        ！可选
```

上面命令中的 type 参数，用来指示这个硬件地址的硬件类型，如 type 为 1，则类型为 ethernet，如果 type 为 6，则类型为 802，默认是 ethernet

4. DHCP 配置 手工绑定 从网络读取绑定



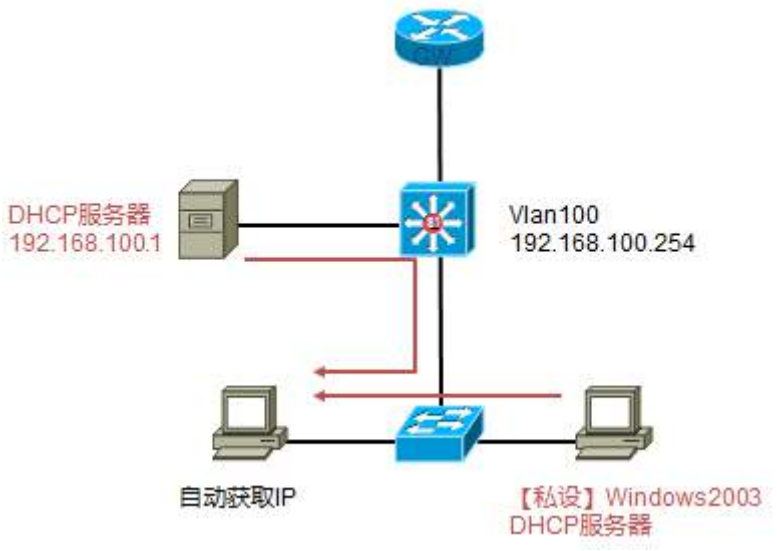

```
ip dhcp pool BIND
  origin file flash:CCIE.txt
```

CCIE.txt 格式：

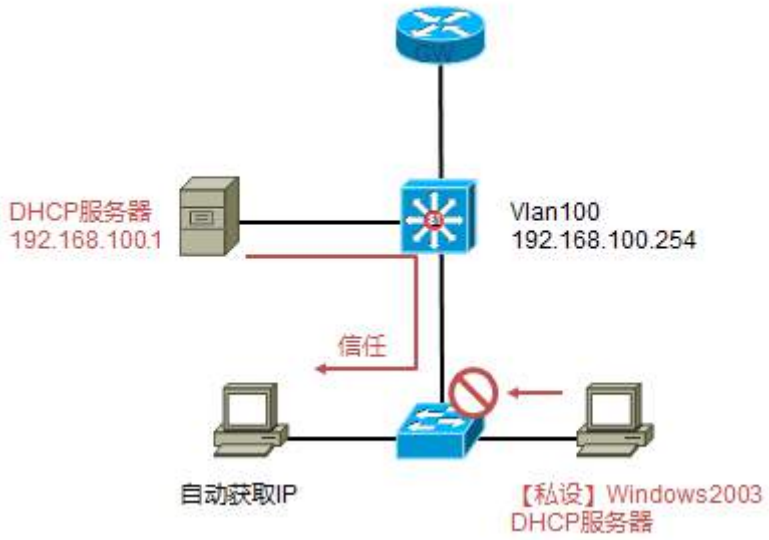
```
*time* Nov 05 2011 07:52 PM
*version* 2
!IP address          Type      Hardware address        Lease expiration
30.30.44.45 /24       id       0100.e01e.6012.89              Infinite
*end*
```

6.3 DHCPsnooping

6.3.1 机制概述



DHCP 都非常熟悉了 ,对于 DHCP 客户端而言 ,初始过程中都是通过发送广播的 DHCP discovery 消息寻找 DHCP 服务器，然而这时候如果内网中存在私设的 DHCP 服务器，那么就会对网络造成影响，例如客户端通过私设的 DHCP 服务器拿到一个非法的地址，最终导致 PC 无法上网。



在 DHCP snooping 环境中（部署在交换机上），我们将端口视为 trust 或 untrust 两种安全级别，也就是信任或非信任接口。在交换机上，将连接合法 DHCP 服务器的接口配置为 trust。只有 trust 接口上收到的来自 DHCPserver 的报文(如 DHCPOFFER, DHCPACK, DHCPNAK, 或者 DHCPLEASEQUERY)才会被放行 ,相反，在 untrust 接口上收到的来自 DHCPserver 的报文将被过滤掉 ,这样一来就可以防止非法的 DHCPserver 接入。同时在部署了 DHCP Snooping 了交换机本地，还能维护一张 **DHCPsnooping 的绑定数据库**（binding database），用于保存侦听到的 DHCP 交互的表项，**信息包括 (针对 untrust 接口的) : MAC 地址、IP 地址 (DHCP 分配的)、租期、绑定类型、VLAN 号、接口编号(DHCP 客户端也就是连接客户端 PC 的 untrust 接口)**。这个 DHCP snooping banding databse 除了可以做一些基本的安全接入控制，还能够用于 DAI 等防 ARP 欺骗的解决方案。

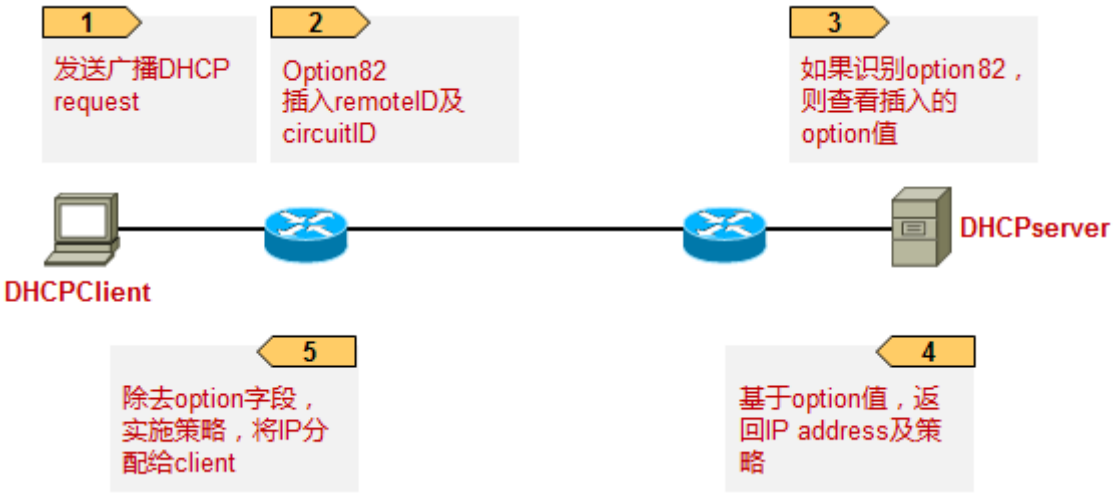
一台支持 DHCP snooping 的交换机，如果在其 **untrust** 接口上，收到来自下游交换机发送的、且带有 option82 的 DHCP 报文，则默认的动作是丢弃这些报文。如果该交换机开启了 DHCP snooping 并且带有 option82 的 DHCP 报文是在 **trusted** 接口上收到的，则交换机接收这些报文，但是不会根据报文中包含的相关信息建立 DHCP bingding databse 表项。

如果交换机确实通过一个 untrust 接口连接了下游交换机，并且希望放行该接口收到的、下游交换机发送出来的带有 option82 的 DHCP 报文，则可使用全局命令：ip dhcp snooping information option allow-untrusted，同时由于是通过 untust 接口收到的 DHCP 报文，因此会根据侦听的结果创建 DHCP snooping binding database 表项。

6.3.2 Option82

DHCP option 82 又称为 DHCP 中继代理信息选项（Relay Agent Information Option），是 DHCP 报文中的一个选项，其编号为 82

CISCO 注释：The DHCP option-82 feature is supported only when DHCP snooping is globally enabled and on the VLANs to which subscriber devices using this feature are assigned.



基本的交互过程如下（下面描述的过程与上面的配图无关）：

1. DHCP client 广播 DHCP request
2. 交换机收到 DHCP request，它在报文中插入 option82 信息（当然，要你先做了相应的配置后），在插入的信息中，remote-ID suboption 是该交换机的 MAC；circuit-ID suboption 是收到 DHCP request 报文的接口 ID。
3. 如果交换机上还部署了 DHCP relay，那么交换机将中继地址也添加进 DHCP 报文中
4. 交换机将携带了 option82 信息的 DHCP 报文转发给 DHCP 服务器

5. DHCP 服务器接收到了 DHCP 报文，如果这台服务器是能够识别 option82 的，那么它将根据 option82 中携带的信息 ,来为客户端分配 IP 地址 ,以及选择相应的策略。然后服务器回应一个 DHCP 包 ,其中也包含 option82 信息。
6. 交换机收到该 DHCP 报文后，发现报文的 option82 信息正是从本地始发，交换机将 option82 信息移除后，将 DHCP 报文转发给 DHCP client。

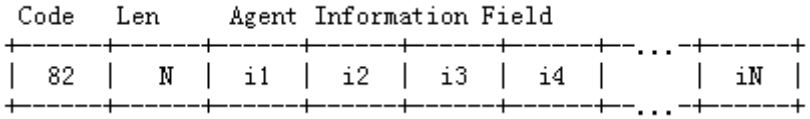
在上述事件的发生过程中，下面字段的内容不会发生改变：

- **Circuit-ID suboption fields**
  - Suboption type
  - Length of the suboption type
  - Circuit-ID type
  - Length of the circuit-ID type
- **Remote-ID suboption fields**
  - Suboption type
  - Length of the suboption type
  - Remote-ID type
  - Length of the remote-ID type

In the port field of the circuit-ID suboption, the port numbers start at 3.也就是说这个字段中，端口号是从 3 开似乎的，第一个端口号就是 3，依次类推。例如，一个带了 SFP 插槽的 24 口交换机，port3 是 fast x/0/1，port4 是 fast x/0/2，以此类推，port 27 就是 sfq 模块的 x/0/1 口。其中 x 为堆叠成员编号。

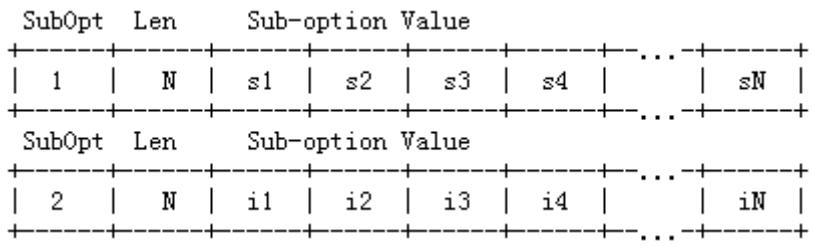
Option 报文结构：

DHCP option 82 又称为 DHCP 中继代理信息选项（Relay Agent Information Option），是 DHCP 报文中的一个选项，其编号为 82。rfc3046 定义了 option 82，选项位置在 option 255 之前而在其它 option 之后。



- CODE：82
- LEN： Agent Information field 的字节数，不包含 code 及 len 字段的长度

Option82 可包含多个 suboption，rfc3046 定义了如下两个：



- Subopt： 子选项编号，如果是 circuit ID 则值为 1，remote ID 则值为 2
- Len： Sub-option Value 的字节个数，不包括 SubOpt 和 Len 字段的两个字节

option82 子选项 1：即 Circuit ID，它表示接收到的 DHCP 请求报文来自的电路标识，这个标识只在中继代理节点内部有意义 ,在服务器端不可以解析其含义 ,只作为一个不具含义的标识使用。一般情况下 ,默认是接收到 DHCP 请求报文的接入交换机 VlanID 加接入二层端口名称，如 Vlan2+Ethernet0/0/10，也可以由用户指定自己的代理电路 ID。通常子选项 1 与子选项 2 要共同使用来标识 DHCP 客户端的信息。

option82 子选项 2：即 Remote ID，一般情况下为插入该 option82 信息的接入层交换机的 MAC。

CISCO 注释 :If the DHCP relay agent is enabled but DHCP snooping is disabled, the DHCP option-82 data insertion feature is not supported.

6.3.3 DHCP Snooping Binding Database

在交换机上激活 DHCP snooping 后，交换机会使用在 untrust 接口上“侦听到的“DHCP 交互内容形成一个表：dhcp snooping binding database。如下：

SW1#show ip dhcp snooping binding

MacAddress	ipAddress	Lease(sec)	Type	VLAN	Interface
00:B0:64:04:09:99	192.168.10.1	86025	dhcp-snooping	10	FastEthernet0/1

Total number of bindings: 1

默认情况下，这个数据库是动态的，也就是说当交换机重启，数据库内的表项全部丢失。这个事情的结果可能会造成网络的中断。因此我们可以将这个数据库以文件的形式存储起来 ,这个就是 DHCP snooping database agent。配置命令非常简单，全局命令：

SW2(config)#ip dhcp snooping database ?

```
  flash:        Database agent URL
```

```
  ftp:            Database agent URL
  http:           Database agent URL
  https:          Database agent URL
  rcp:            Database agent URL
  scp:            Database agent URL
  tftp:           Database agent URL
  timeout         Configure abort timeout interval
  write-delay     Configure delay timer for writes to URL
```

由于 flash 空间非常有限，因此建议将文件存储在 TFTP 服务器上。当部署在 TFTP 服务器上时，注意先创建一个空的文件以对应配置命令中的 URL，这个文件用于 dhcp snooping bingding database 的写入（视具体设备而定）。

形成的文件的格式如下：

```
<initial-checksum>
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
<entry-1>     <checksum-1>
<entry-2>     <checksum-1-2>
...
...
<entry-n>     <checksum-1-2-..-n>
END
```

下面是一个例子：

```
2bb4c2a1
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
192.1.168.1 3 0003.47d8.c91f 2BB6488E interface-id 21ae5fbb
192.1.168.3 3 0003.44d6.c52f 2BB648EB interface-id 1bdb223f
192.1.168.2 3 0003.47d9.c8f1 2BB648AB interface-id 584a38f0
END
```

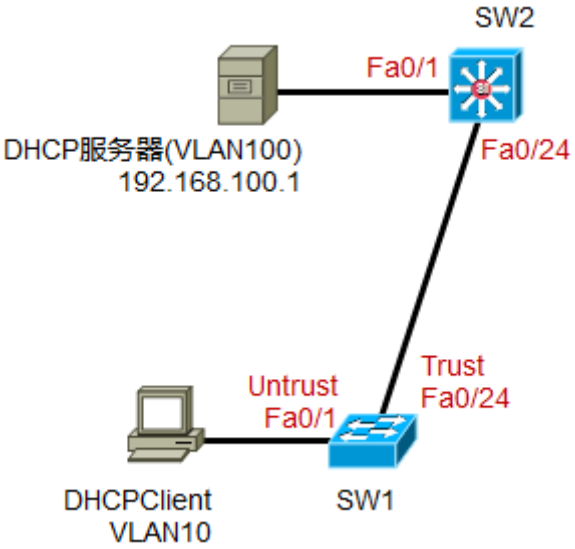
6.3.4 配置示例

1. **实验 1：基础实验**

PC 属于 VLAN10，网关在 SW2 上。DHCPserver 属于 VLAN100，网关在 SW2 上。

SW1 为接入层交换机，部署 DHCP snooping，将 Fa0/24 定义为 trust 接口。

SW2 为核心层交换机，部署 DHCP relay



DHCPserver 的配置如下：

```
no ip routing
ip default-gateway 192.168.100.254
Interface fast0/0
  ip address 192.168.100.1 255.255.255.0
  no shutdown
  exit
Service dhcp
Ip dhcp pool vlan10
  network 192.168.10.0 /24
  default-router 192.168.10.254
```

SW1 的配置如下：

```
vlan 10
!
ip dhcp snooping                          !! 全局开启 DHCP snooping
ip dhcp snooping vlan 10                  !! vlan10 激活 DHCP snooping
no ip dhcp snooping information option    !! 不写入 option82
!
Interface fast0/1
```

```
  switchport access vlan 10
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
  ip dhcp snooping trust
```

SW2 的配置如下：

```
vlan 10
vlan 100
  name Server
!
Interface fast0/1
  switchport access vlan 100
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
  ip helper-address 192.168.100.1
Interface vlan 100
  ip address 192.168.100.254 255.255.255.0
```

实验结果：

PC 获取到了地址，在 SW1 上查看 dhcp snooping bingding databsae：

SW1#show ip dhcp snooping binding

```
MacAddress          ipAddress       Lease(sec)  Type           VLAN  Interface
------------------  --------------  ----------  -------------  ----  --------------------
00:B0:64:04:09:99   192.168.10.1    86025       dhcp-snooping    10  FastEthernet0/1
Total number of bindings: 1
```

2. **实验 2：接入交换机插入 option82，核心交换机做中继**

在 SW1 上开启 DHCP snooping，同时不去 no ip dhcp snooping information option，也就是插入 option82。其他设备配置与上一个实验相同，发现 PC 机无法获取地址，通过在网关设备上，也即是 SW2 上 debug 会发现报错 relay information option exists, but giaddr is zero。giaddr 字段意思为该报文做经过的第一个 DHCP 中继代理的 IP 地址。

在 SW2 上增加配置：(全局配置模式) ip dhcp relay information trust-all

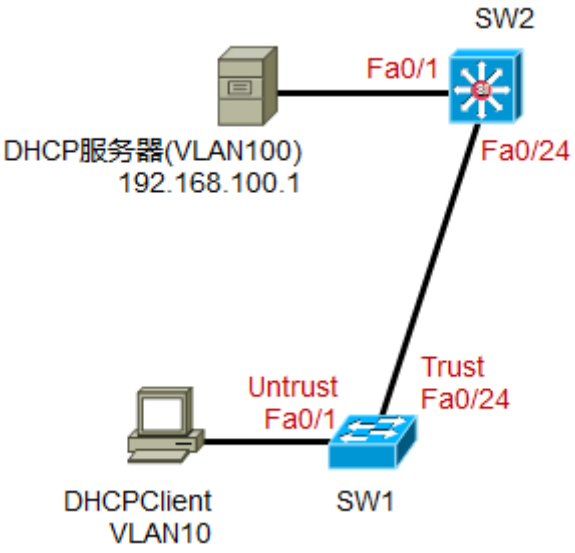
或：

Interface vlan 10

ip dhcp relay information trusted

上述命令，将使交换机接受携带 option82 信息，同时 giaddr 为 0 的 dhcp 报文，然后做中继。

如此一来即可成功获取地址。



SW1 的配置如下：

```
vlan 10
!
ip dhcp snooping
ip dhcp snooping vlan 10
Interface fast0/1
  switchport access vlan 10
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
  ip dhcp snooping trust
```

SW2 的配置如下：

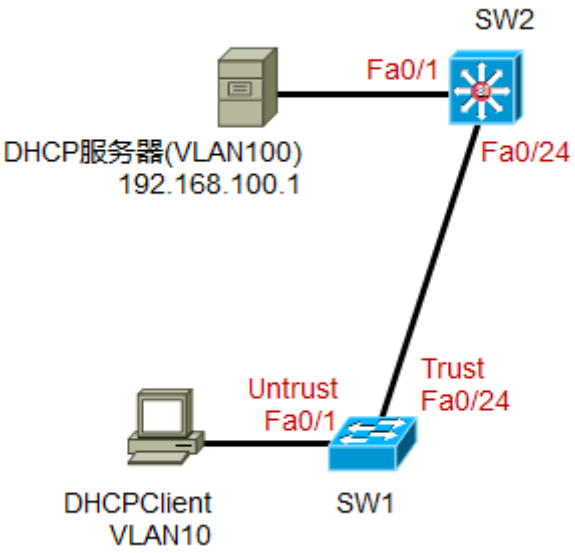
```
vlan 10
vlan 100
  name Server
!
```

```
ip dhcp relay information trust-all
Interface fast0/1
  switchport access vlan 100
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
  ip helper-address 192.168.100.1
Interface vlan 100
  ip address 192.168.100.254 255.255.255.0
```

3. **实验 3：接入层交换机插入 option82，上游核心交换机也开启 dhcp snooping**

接入层交换机 SW1 开启 dhcp snooping，插入 option82；

核心层交换机 SW2 也开启 dhcp snooping，默认所有端口都是 untrust 的，这时如果 SW2 的 fast0/24 口收到携带了 option82 的 dhcp 报文，则丢弃。解决办法之一，是将 SW2 的 fast0/24 口配置为 trust 接口，但是这么一来通过该 trust 接口收到的 dhcp 报文，是不会创建 dhcp snooping binding 表项的。另一个方法是，使用命令：ip dhcp snooping information option allow-untrusted。这条命令放行携带了 option82 的 DHCP 报文，同时会在本地创建 dhcp snooping binding 表项。



SW1 的配置如下：

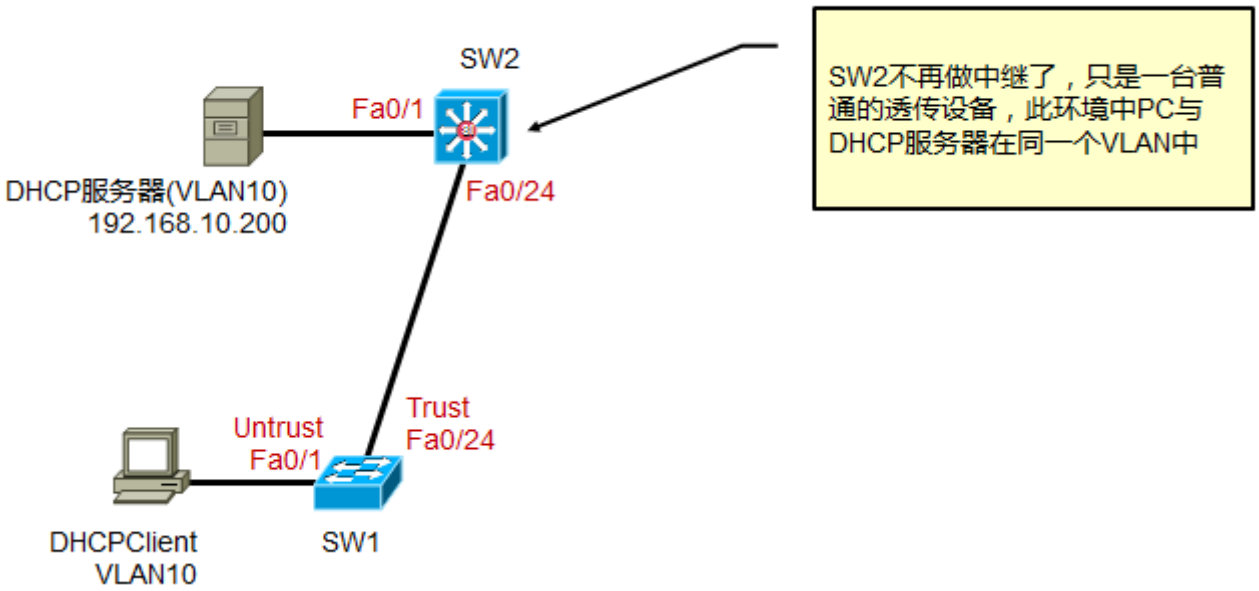
```
vlan 10
!
ip dhcp snooping
```

```
ip dhcp snooping vlan 10
Interface fast0/1
  switchport access vlan 10
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
  ip dhcp snooping trust
```

SW2 的配置如下：

```
vlan 10
vlan 100
  name Server
!
ip dhcp snooping information option allow-untrusted
ip dhcp relay information trust-all
                                        !!注意区分上述两条命令
Interface fast0/1
  switchport access vlan 100
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
  ip helper-address 192.168.100.1
Interface vlan 100
  ip address 192.168.100.254 255.255.255.0
```

4. 无中继环境中，option82 及相关问题



这个环境比较简单，PC 与 DHCP 服务器同处一个 VLAN，我主要想证实一下普通交换机对携带了 option82 的 DHCP 包的处理情况。

SW1 仍然开启 dhcp snooping，同时插入 option82

SW2 将服务器划入与 PC 相同的 VLAN，相当于纯做透传。

其中 SW1 的配置如下：

```
vlan 10
!
ip dhcp snooping
ip dhcp snooping vlan 10
Interface fast0/1
  switchport access vlan 10
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
  ip dhcp snooping trust
```

SW2 的配置如下：

```
vlan 10
!
Interface fast0/1
  switchport access vlan 10
```

```
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
```

DHCP server 的配置就是常规配置，这里不再赘述。

实验的结果是，PC 无法获取地址。

我们分析一下，SW1 将 PC 发出来的 DHCP 请求消息插入 option82，随后发给 SW2。通过 SW2 上的 debug 信息我们发现：

*Mar 16 17:09:21.858: DHCPD: inconsistent relay information.

*Mar 16 17:09:21.858: DHCPD: relay information option exists, but giaddr is zero.

SW2 意识到，DHCP 报文中有中继信息选项，也就是 option82，但是 giaddr 是全 0。SW2 虽然意识到了这个问题，但还是乖乖将这个数据交给了 DHCPserver，通过在 DHCPserver 上的 debug 信息：

*Mar 4 07:02:23.473: DHCPD: inconsistent relay information.

*Mar 4 07:02:23.473: DHCPD: relay information option exists, but giaddr is zero.

显然，DHCP 服务器也意识到了上面的问题，它采取的动作是，忽略这个 DHCP 请求。因此 PC 无法获取地址。

解决的办法之一，是 SW1 不插入 opiton82，另一个方法，是在 DHCPserver 上，使用配置：

（全局配置模式）ip dhcp relay information trust-all

或：

```
Interface fast0/0
  ip dhcp relay information trusted
```

思路和实验 2 一样。这样一来 SW2 会接受这些 DHCP 报文并进行处理，PC 就能够获取到地址了。

5. 其他配置

```
ip dhcp snooping limit rate rate
```

注意，在交换机开启了 DHCP snooping 后，由于对 dhcp 报文都要进行窥探，因此当网络中存在 DHCP 攻击行为，将会严重消耗交换机的性能，因此，可以通过该命令限制接口上允许收到的 DHCP 报文数量。Configure the number of DHCP packets per second that an interface can receive. The range is 1 to 2048. By default, no rate limit is configured.

```
ip dhcp snooping verify mac-address
```

Configure the switch to verify that the source MAC address in a DHCP packet received on untrusted ports matches the client hardware address in the packet. The default is to verify that the source MAC address matches the client hardware address in the packet.

ip dhcp snooping binding *mac-address* vlan *vlan-id ip-address* interface *interface-id* expiry *seconds*

静态配置一条 dhcp snooping bingding dabase 表项，该命令是在特权模式下配置。

6.4 DAI - Dynamic ARP Inspection

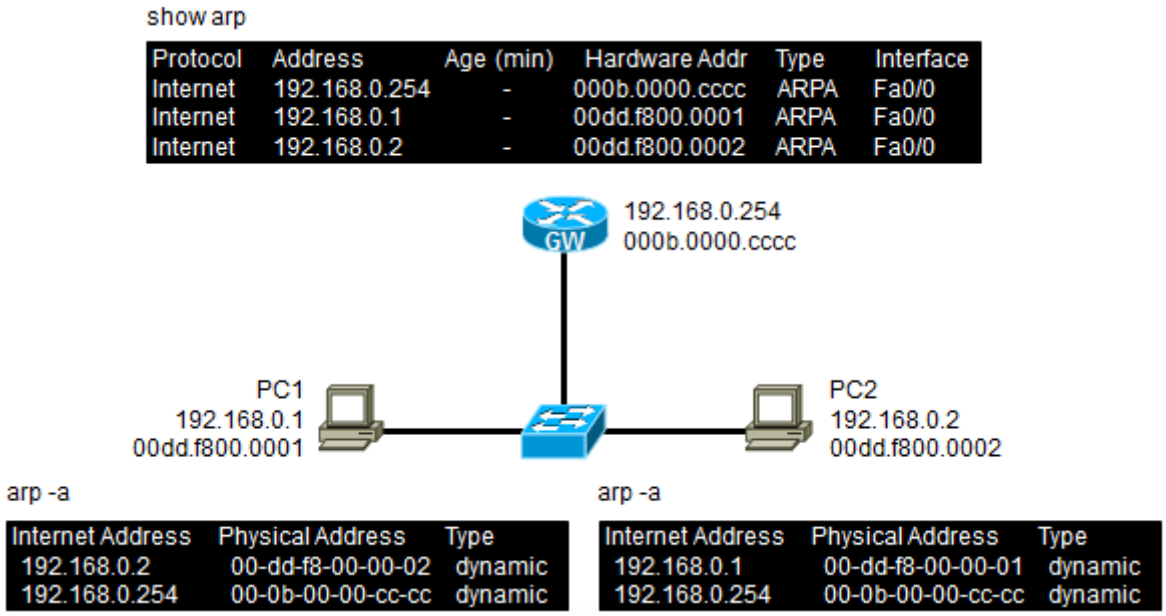
6.4.1 ARP 协议原理

1. **协议概述**
   - Address Resolution Protocol
   - 在以太网环境中，节点之间互相通信，需知晓对方的 MAC 地址
   - 在现实环境中，一般采用 IP 地址标示通信的对象，而 ARP 的功能就是将 IP“解析”到对应的 MAC 地址。

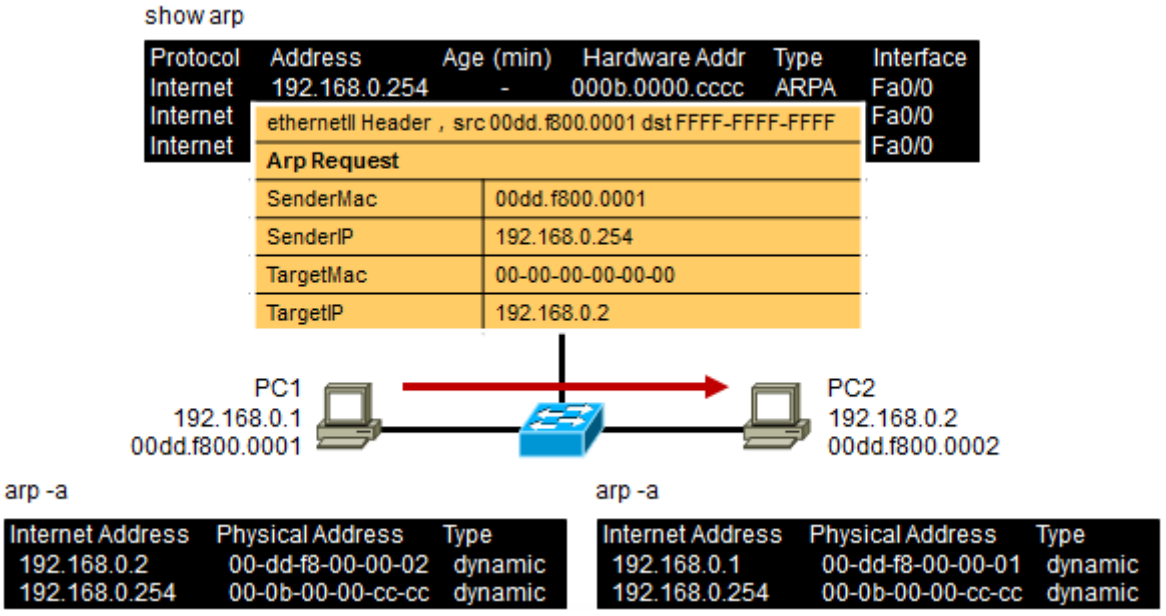
2. **协议漏洞**
   - 基于广播，不可靠
   - ARP 响应报文无需请求即可直接发送，这给攻击者留下巨大漏洞
   - 没有确认机制，任何人都可以发起 arp 请求或 response
   - IPv6 中通过特定的机制规避掉 ARP 以及其漏洞

6.4.2 ARP 欺骗原理

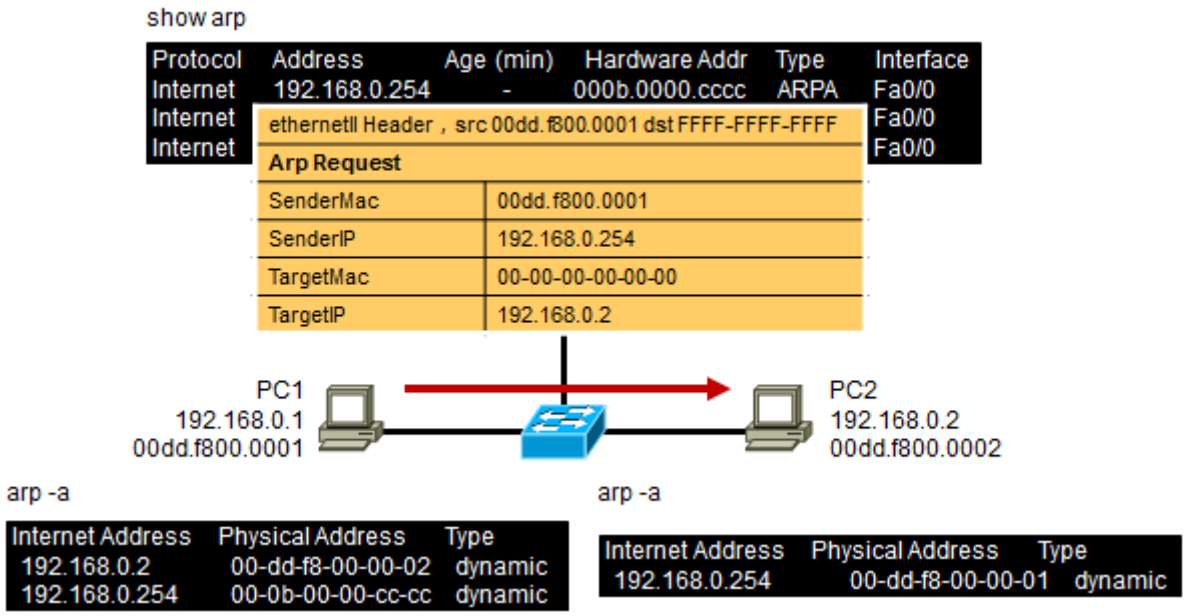
1. **正常情况下各设备的 ARP 表项**



2. 欺骗过程 ARP Request

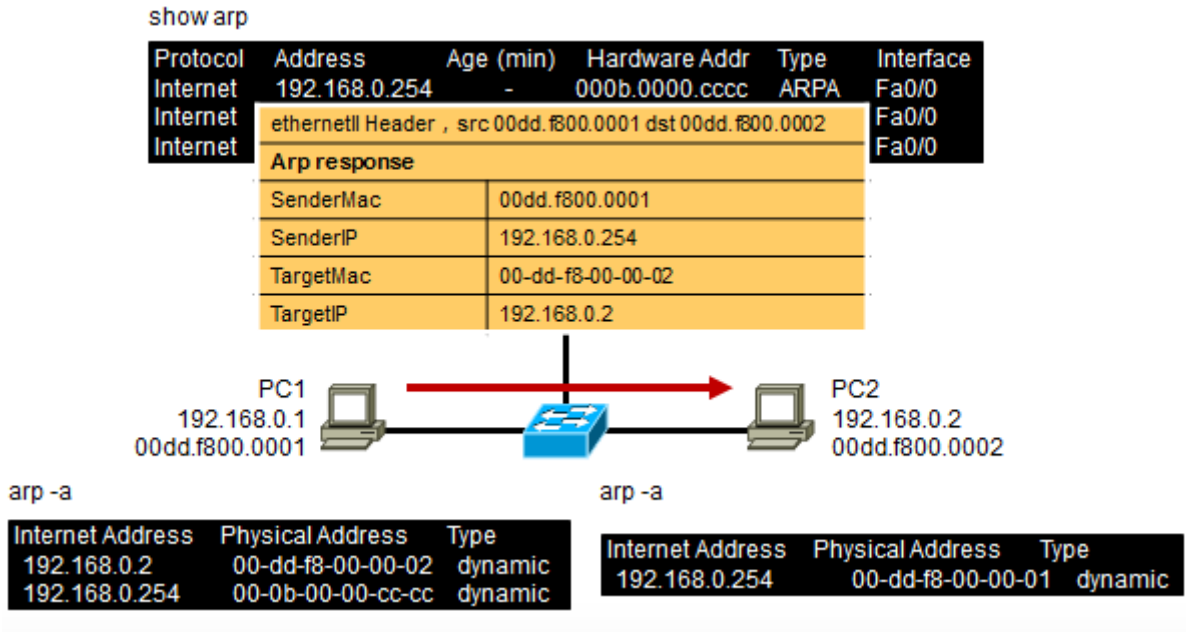


PC1 要欺骗 PC2，通过造包工具，如 anysend 等，发出一个 ARP request 广播，这个 ARP 数据帧的以太网头部中，源 MAC 为 PC1 的 MAC，目的 MAC 为全 F，关键内容在于 ARP 协议报文（ARP body），也就是这个数据帧的以太网帧头后的内容，这才是 ARP 的真正内容，里头包含的主要元素有：发送者 mac、发送者 IP、接收者 MAC 及接收者 IP。PC1 发出的这个 ARP 数据帧中，sender MAC 为自己的 MAC 地址，但是 SenderIP 为网关 IP 即 192.168.0.254，此刻，PC2 收到这个 arp 消息后，将更新自己的 arp 表项，如下：



如此一来，PC2 发送数据到自己的网关 0.254 时，由于网关的 MAC 已经被欺骗成了 PC1 的 MAC，因此数据都被转发给了 PC1，那么 PC2 就断网了。如果 PC1 再机灵点，将 PC2 发过来的数据再转给真实的网关，同时在本地运行一个报文分析工具窥探 PC2 发过来的数据，那么就可以在 PC2 不断网的情况下，神不知鬼不觉的窥探 PC2 的上网流量，这就是 the man in the middle，中间人攻击。

3. 欺骗过程 ARP Response



由于 ARP 的 response 并不需要 arp request 为先决条件即可直接发送 ,因此攻击者可以构造 arp reply 消息，并发送给被攻击对象，从而刷新被攻击者的 arp 表，同样能达到 arp 欺骗的目的。

4. 理解 invalid arp 表项

什么是 invalid arp 表项：

ARP 表中的 MAC 地址为全零（Windows 主机）或 “No completed”（网络设备）

```
Ruijie#show arp
Total Numbers of Arp: 5
Protocol  Address          Age(min)  Hardware        Type   Interface
Internet  192.168.1.1      <---->    <No completed>  arpa   VLAN 1
Internet  192.168.1.254    --        001a.a908.9f0b  arpa   VLAN 1
Internet  192.168.0.2      9         00d0.f800.0002  arpa   VLAN 26
Internet  192.168.0.1      2         00d0.f800.0001  arpa   VLAN 26
Internet  192.168.0.254    --        001a.a908.9f0b  arpa   VLAN 26
```

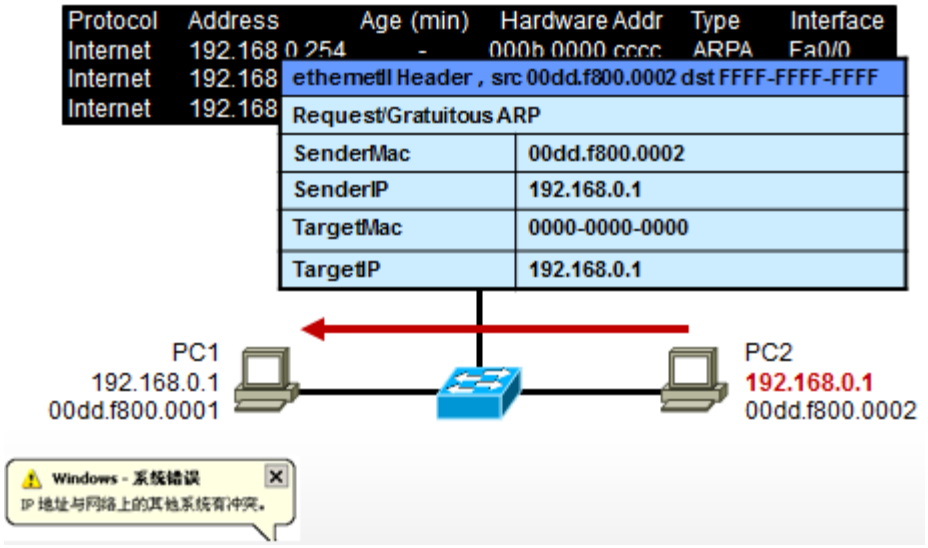
产生的原因：发送 ARP Request 后，为接收 ARP Reply 做准备

大量存在的原因：同网段扫描（主机）、跨网段扫描（网络设备）

5. Gratuitous ARP

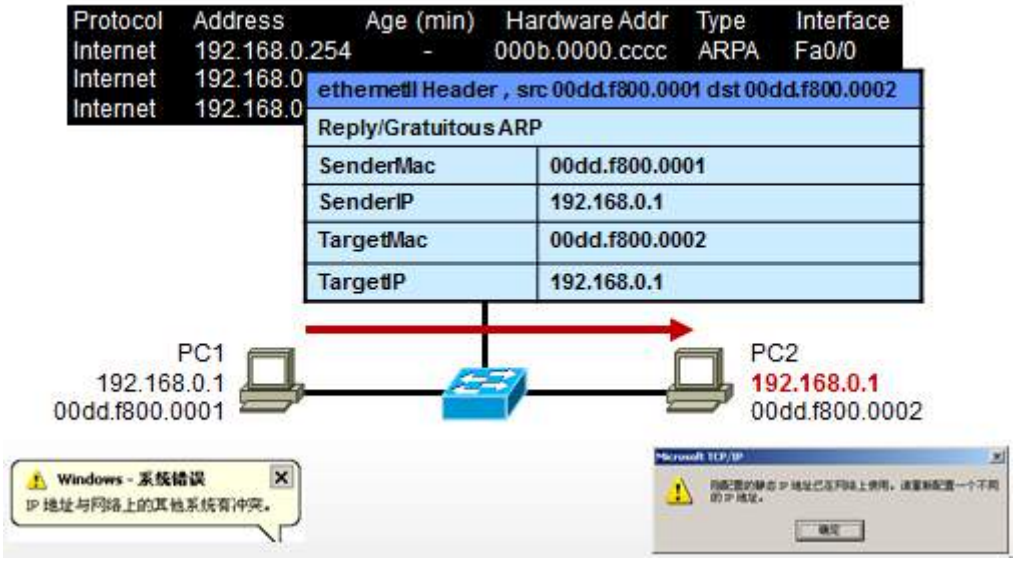
Gratuitous ARP，免费 ARP 是一种特殊的 ARP Request/response 报文，即 Sender IP 与 Target IP 一致（一般用于 IP 冲突检测）

我们看下面的例子：



PC2 是新接入的主机，配置了 192.168.0.1 的地址，完成地址配置后，PC2 向网络中发送一个 Gratuitous ARP，以防内网中有人使用相同的 IP 地址，这个 Gratuitous ARP 的 senderIP 和 target IP 都是自身，senderMAC 是自己的 MAC。

那么当 PC1 收到这个 Gratuitous ARP 后，由于 senderIP 和自己冲突了，于是在本地弹出一个气泡（如果是 windows 系统），同时回复一个 Gratuitous ARP response 以告知对端出现了 IP 地址重复。



对端收到这个 Gratuitous ARP 消息后，知道内网中已经存在这个 IP 地址的使用者了，于是告警。

Gratuitous ARP 的另外一个作用是，某些厂家用于防止 ARP 欺骗。例如在某些路由器上，配置了基于 Gratuitous ARP 的防 arp 欺骗解决方案，则该路由器（的接口）将以一定的时间间隔发送 Gratuitous ARP，目的就是刷新底下 PC 的 arp 表，以保持网关 IP 对应的 MAC 始终是正确的，当然，这种防 arp 的解决方案挺土鳖的。

6. 如何判断内网中是否存在 arp 欺骗

现在市面上其实已经有许多 arp 防火墙可以识别到 ,当然有一些简单的方法 ,例如 arp –a 看 MAC 是否正确。或者抓包等。

7. Arp 欺骗的解决方案

解决方案 1：手工绑定 IP+MAC

例如在 PC 上 ,用 arp –s 静态绑定网关 IP 及网关的 MAC ,以防止 arp 欺骗 ,或者在关键节点的网络设备上，使用命令 arp 进行绑定。

显然这种方式扩展性太差。

解决方案 2：免费 arp

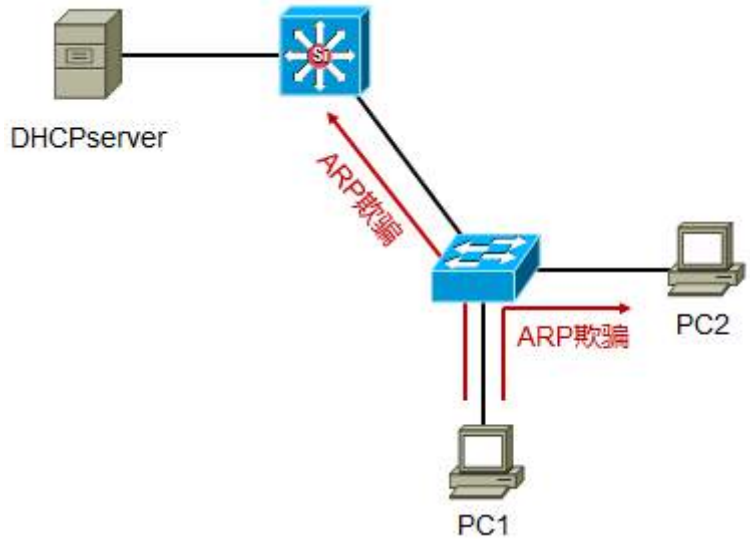
这个前面已经说过了

解决方案 3：DAI

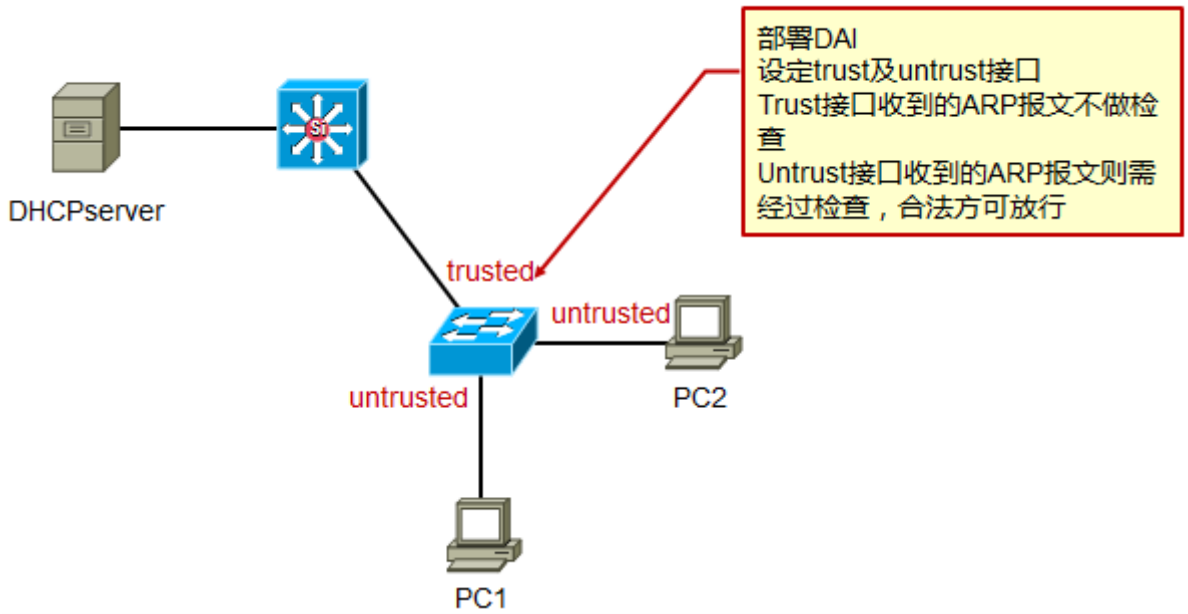
其他，其实防 ARP 欺骗的解决方案业内还是有许多的，每个厂家都有自己的方法。

6.4.3 DAI 工作机制

1. 机制概述

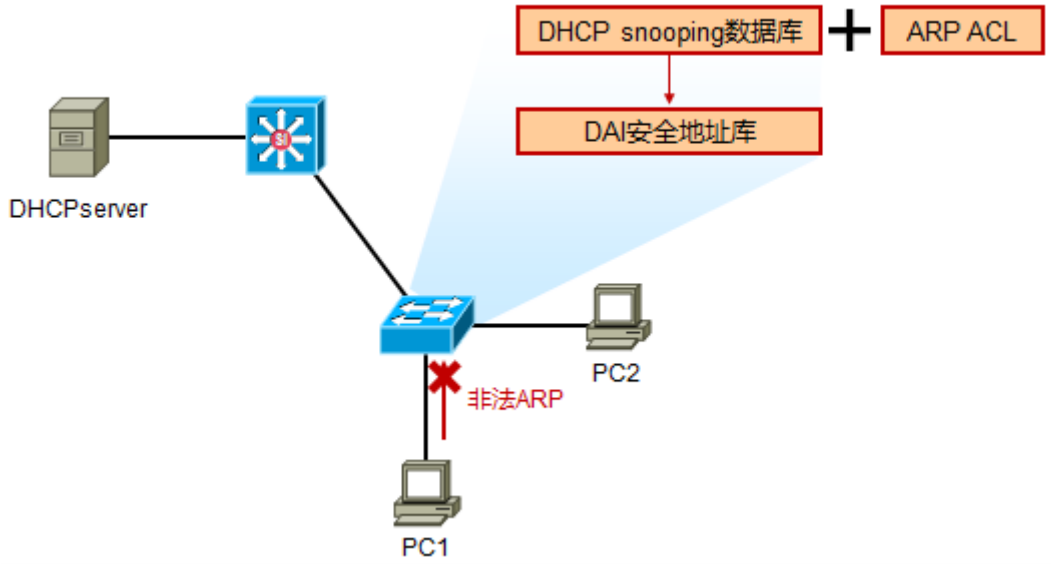


部署 DAI 前，内网如果出现 ARP 欺骗行为，例如 PC 发送非法的 ARP 报文，对于交换机而言，是无法检测并作出防御动作的。



部署 DAI 后，我们可以定义交换机接口的信任状态：trusted，或 untrusted，对于信任端口，将不对收到的

ARP 报文做检测，直接放行。而对于 untrusted 接口，ARP 报文在收到后回进行检查，只有合法的 ARP 报文才会被放行，如果出现非法的 ARP 报文，则会被 log，同时丢弃。



DAI 依赖 DHCP snooping 技术，在一个用户动态获取 IP 地址的网络环境中，我们通过部署 DHCP snooping 一来可以起到防御 DHCP 欺骗的效果，另一方面，会在交换机上得到 DHCP snooping binding database ,这个数据库是 DHCP Snooping 侦听 DHCP 交互过程后的信息记录 ,里头包含客户端的 IP、MAC、所属 VLAN 等等相关信息，而这些信息，恰恰可以作为 ARP 合法性校验的依据。

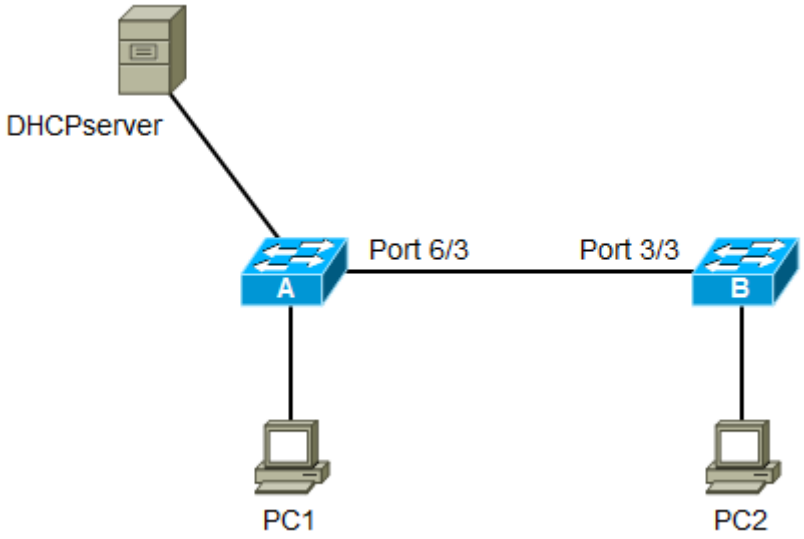
DAI 部署于交换机上，用于确保合法的 ARP request 或 response 被放行而非法的 ARP 消息被丢弃。交换机部署 DAI 后的主要动作如下：

- 在 DAI untrust 接口上（注意与 DHCP snooping 的 untrust 接口区分开）阻拦一切 ARP requests 或 response 消息并作校验
- 在更新自己 ARP 表或将收到的 ARP 消息转发出去之前先进行合法性校验，主要看 ARP 报文中的 IP 及 MAC 对应关系是否合法
- 丢弃非法的 ARP 报文，交换机会在丢弃非法的 ARP 后进行 log

前面说了 DAI 借助于 DHCP snooping 的绑定数据库进行 ARP 合法性校验。另一个合法性校验的依据是手工配置的 ARP ACL。

你也可以配置 DAI 来丢弃那些以太网帧头源 MAC 与 ARP body 里的 MAC 不一致的非法 ARP。这个后面有做进一步的介绍。

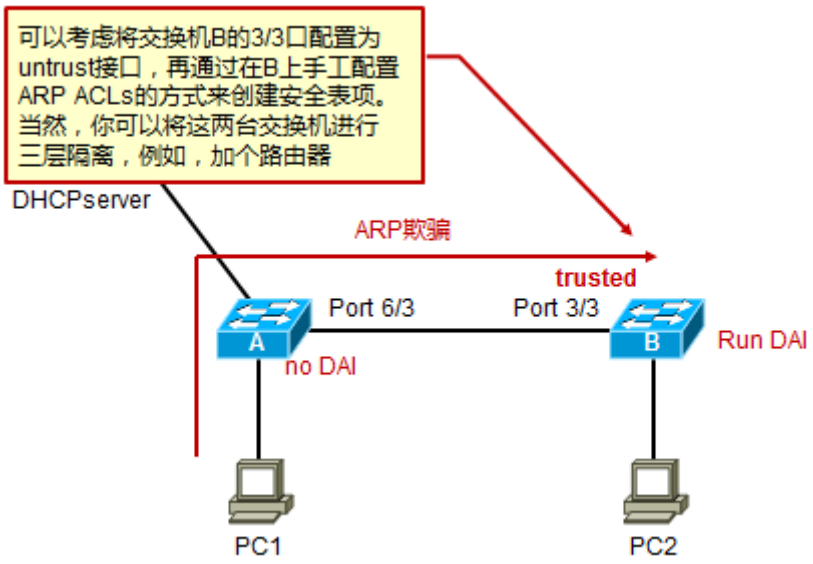
2. 接口信任状态及网络的安全问题



一般来说，我们认为攻击行为多来自于终端的 PC，可能是中毒或者黑客行为等。所以我们常会将连接终端 PC 的接口配置为 DAI untrust 接口，而交换机之间的接口配置为 trust 接口。

看上图，我们假设交换机 A 和 B 都运行 DAI，而 PC1 及 PC2 都通过 DHCP 服务器获取地址，那么这时候对于 PC1 而言，只有在交换机 A 上的 DHCP snooping database 中才有其 IP+MAC 的绑定，交换机 B 则没有。因此如果 A 和 B 之间的接口为 DAI 的 untust，则来自 PC1 的 ARP 报文，将被交换机 B 丢弃，因为它认为这些报文是非法的，如此一来，PC1 和 2 的通信就出问题了。所以建议 A 和 B 之间的接口设置为 DAI 的 trust。

但是，并非所有的交换机互联接口设置为 DAI 的 trust 都会没有问题，在特定的环境下，这会留下一个安全隐患。例如，仍然是上面的图，假设 A 交换机不支持 DAI，如果 B 的 3/3 口配置为 trust，那么 PC1 就可以大摇大摆的去对 B 和 PC2 进行 ARP 欺骗了，即使 B 运行了 DAI。所以 DAI 仅仅是保证运行 DAI 的交换机本身所连接的终端 PC 不能进行非法的 ARP 动作。像刚才所述的情况，可以考虑将交换机 B 的 3/3 口配置为 untrust 接口，再通过在 B 上手工配置 ARP ACLs 的方式来创建安全表项。当然，你可以将这两台交换机进行三层隔离，例如，加个路由器神马的。



3. ARP 报文的 rate limit

对于 DAI 而言 ,是需要损耗交换机 CPU 资源的 ,如果开启了 DAI ,就有可能会成为拒绝服务攻击的对象。所以我们对于运行了 DAI 的交换机接口 ,有 ARP 报文的限制。默认 DAI untrust 接口的 rate limit 是 15 个 P/S 也就是 15pps，trust 接口则完全没有限制。可以通过 ip arp inspection limit 这条接口级的命令来修改。

当接口收到的 ARP 报文超出这个阀值，接口将进入 err-disable。可以使用 shutdown no shutdown 的方式手工重新恢复这个接口。或者，使用全局命令 the errdisable recovery 来让接口在一定的时间间隔后自动恢复。

4. ARP ACL 及 DHCP snooping database 条目的相对优先级

ARP ACL 的优先级高于 DHCP snooping database 条目。在你使用全局命令 ip arp inspection filter 指定了 ARP ACL 后，如果 ARP 报文被 ARP ACL deny 掉了，那么这些报文就被直接丢弃，即使在 DHCP snooping database 中有合法的表项匹配这些 ARP 报文。

6.4.4 DAI 的配置方针

1. DAI 是 ingress 安全特性，不会做 egress 的安全校验
2. DAI 的工作需依赖 DHCP snooping
3. 如果 DHCP snooping 被关闭或者，这是一个无 DHCP 的网络环境，例如纯静态 IP 地址的环境，使用 ARP ACLs 来放行或丢弃 ARP 报文

4. DAI 在 access 接口, trunk 接口, EtherChannel 接口, 及 private VLAN 接口上都支持

6.4.5 DAI 配置命令

1. 基本配置

ip arp inspection vlan {vlan_ID | vlan_range}

激活特定 VLAN 的 DAI

ip arp inspection trust

接口模式，将接口配置为 DAI trusted，默认为 untrusted

2. 应用 ARP ACLs 到 DAI

ip arp inspection filter arp_acl_name **vlan** {vlan_ID | vlan_range} **[static]**

将配置好的 ARP ACL 应用到 DAI

如果不使用 static 关键字，则 ARP 报文会先被 ARP ACL 匹配，如果没有任何匹配项，则再被 DHCP Snooping database 的安全表项再做一次校验，如果有合法表项，则放行，如果没有，则丢弃。

如果使用 static 关键字 ,则意味着 ARP ACLs 启用隐含 deny any 机制。也就是说如果 ARP 报文与 ARP ACLs 匹配，而结果是没有任何匹配项，则直接丢弃，不管 DHCP Snooping database 里是否有合法的条目。

3. 配置 ARP 报文 rate limit

ip arp inspection limit {rate pps **[burst interval** seconds**] | none}**

接口级命令。通过限制接口上收到的 ARP 报文的数量，可以有效的防止开启 DAI 的交换机被 DoS 攻击。

注意事项：

- 默认在 DAI untrusted 接口上 rate 是 15pps，而 trusted 接口则无限制
- Ip arp inspection limit rate none，是无限制
- Burst interval ，默认是 1s，可选配置。这是一个连续的检测时间段，用来检测在这个时间段内的 ARP 报文数量，默认是 1S，所以就是说 1S 内，如果超出了 15 个（默认）ARP，则违例 。这个时间可选区间为 1-15S
- 当出现违例，接口进入 err-disable 状态

errdisable recovery cause arp-inspection

激活由于 arp-inspection 违例导致的 err-disable 接口的自我恢复。

4. 配置附加的校验动作

ip arp inspection validate { [dst-mac] [ip] [src-mac] }

我们可以通过配置一些额外的选项，让 DAI 做进一步的校验。

有三个可选关键字，dst-mac、ip、src-mac

- dst-mac　　检测收到的 ARP 报文中，以太网帧头的目的 MAC 与 ARP body 里的 target MAC 是否一致，这个 check 针对 ARP response。当开启这个选项后，如果两个 MAC 不一致，则 ARP 报文被丢弃
- ip　　检测 ARP body 里的 IP 地址是否是无效或非预期的，如 0.0.0.0、255.255.255.255，以及组播 IP 地址，这些都被认为是无效的 IP。SenderIP 是无论报文为 ARP request 或 response 都会进行校验。而 targetIP 只有在报文是 ARP response 时才会进行校验
- src-mac　　检测收到的 ARP 报文中 以太网枕头的源 MAC 与 ARP body 里的 sender MAC 是否一致。这个校验动作在 ARP request 及 response 报文中都会进行。

5. 配置 DAI logging

当 DAI 丢弃一个 ARP 报文，交换机会在 log buffer 里存储一条信息，随后产生一条 system message 系统信息，在这条系统信息产生后，交换机从 log buffer 里清除之前存储的条目。每一个条目都包含例如 VLAN、端口号、源目的 IP 地址、源目的 MAC 等信息。

一个存储在 log buffer 中的条目可以代表多个报文 ,例如交换机从同一个接口同一个 VLAN 收到具有相同 ARP 元素的 ARP 包，那么这些报文对应一个条目，以节省 buffer 的占用。同时只会产生一条系统信息。

ip arp inspection log-buffer entries number

配置 DAI 占用 log buffer 的大小（最大条目数量），区间是 0-1024

默认是 32 条

ip arp inspection log-buffer logs number_of_messages **interval** length_in_seconds

配置 DAI loggin system message，系统消息将以每 length_in_seconds 发送 number_of_messages 的速率来约束系统消息的发送。number_of_messages 默认为 5，范围是 0-1024，如果配置为 0 则表示条目只出现在 log buffer 中而不产生系统消息；length_in_seconds 默认是 1S，范围是 0-86400（1 天），如果配置为 0 则表示系统消息将会立即产生，而 log buffer 中不再存储条目，如果将 length_in_seconds 配置为 0，则 number_of_messages 也会变为 0。

ip arp inspection vlan *vlan_range* **logging {acl-match {matchlog | none} | dhcp-bindings {all | none | permit}}**

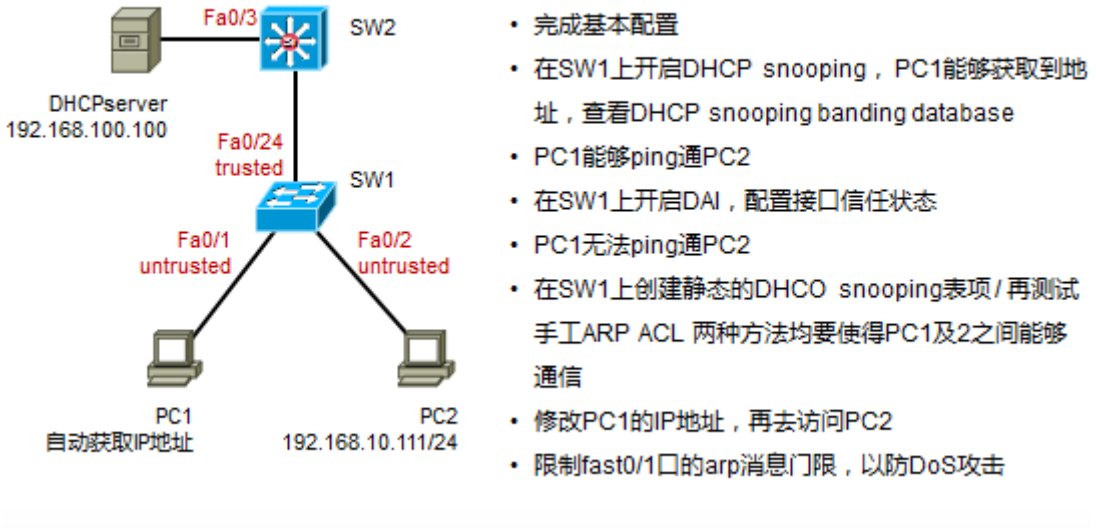
默认情况下，只要 DAI 丢弃报文，则会被 logged

- **acl-match matchlog**—Logs packets based on the DAI ACL configuration. If you specify the
- **matchlog** keyword in this command and the **log** keyword in the **permit** or **deny** ARP access-list
- configuration command, ARP packets permitted or denied by the ACL are logged.
- **acl-match none**—Does not log packets that match ACLs.
- **dhcp-bindings all**—Logs all packets that match DHCP bindings.
- **dhcp-bindings none**—Does not log packets that match DHCP bindings.
- **dhcp-bindings permit**—Logs DHCP-binding permitted packets.

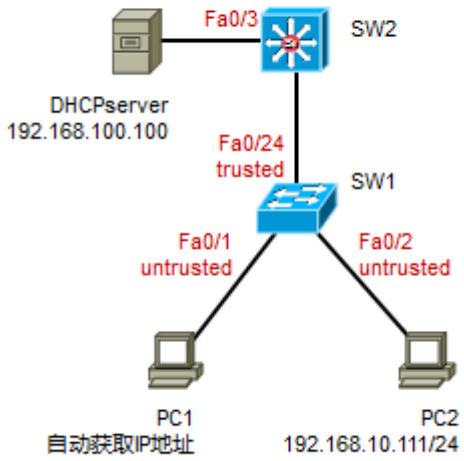
ip arp inspection vlan 100 logging acl-match none

意思是针对 VLAN100 中的流量，不去 log 被 ARP ACL 匹配丢弃的报文

6.4.6 DAI 配置范例

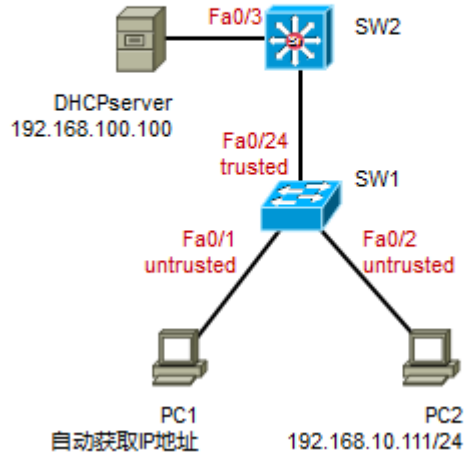


基本配置如下：



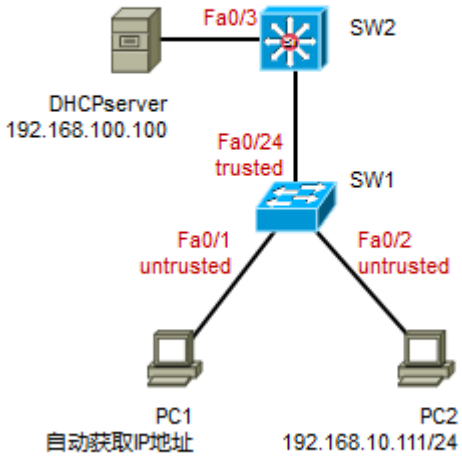
DHCPserver的配置如下：

```
no ip routing
!
ip default-gateway 192.168.100.254
!
Interface fast1/0
  ip address 192.168.100.100 255.255.255.0
  no shutdown
  exit
!
Service dhcp
!
Ip dhcp pool vlan10
  network 192.168.10.0 /24
  default-router 192.168.10.254
```



SW2的配置如下：

```
vlan 10
vlan 100
!
Interface fast0/3
  switchport mode access
  switchport access vlan 100
interface fast0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
Interface vlan 10
  ip address 192.168.10.254 255.255.255.0
  ip helper-address 192.168.100.100
Interface vlan 100
  ip address 192.168.100.254 255.255.255.0
```



SW1的配置如下：

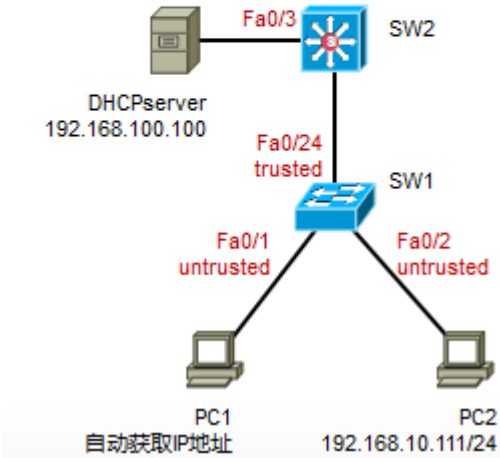
```
vlan 10
!
Interface range fast 0/1 -2
  switchport mode access
  switchport access vlan 10
Interface fast 0/24
  switchport trunk encapsulation dot1q
  switchport mode trunk
!
ip dhcp snooping
ip dhcp snooping vlan 10
no ip dhc snooping information option
!
interface fast0/24
  ip dhcp snooping trust
```

基本配置完成后，来看一下：

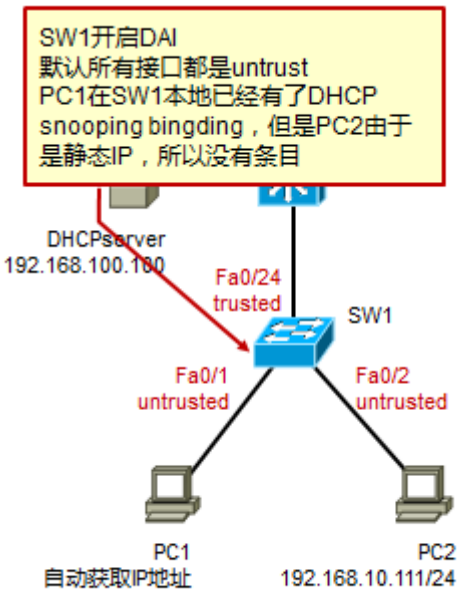
SW1#sh ip dhcp snooping binding

MacAddress	IpAddress	Lease(sec)	Type	VLAN	Interface
00:B0:64:04:09:A1	192.168.10.2	86366	dhcp-snooping	10	FastEthernet0/1

Total number of bindings: 1



接下去在 SW1 上开启 DAI：



SW1的增加配置如下：

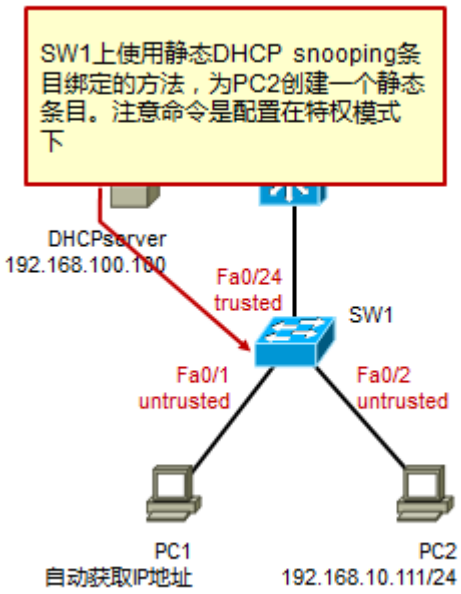
```
Ip arp inspection vlan 10
!
Interface fast 0/24
  ip arp inspection trust
```

SW1上立即出现如下系统消息：

*Mar 1 00:25:18.827: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Fa0/2, vlan 10.([000b.5f35.70a1/192.168.10.111/00b0.6404.09a1/192.168.10.2/00:25:18 UTC Mon Mar 1 1993])

PC1与PC2无法通信

为了使得 PC2 发出的 ARP 消息能够被放行，我们首先使用手工配置 DHCP snooping bingding 表项的方法：



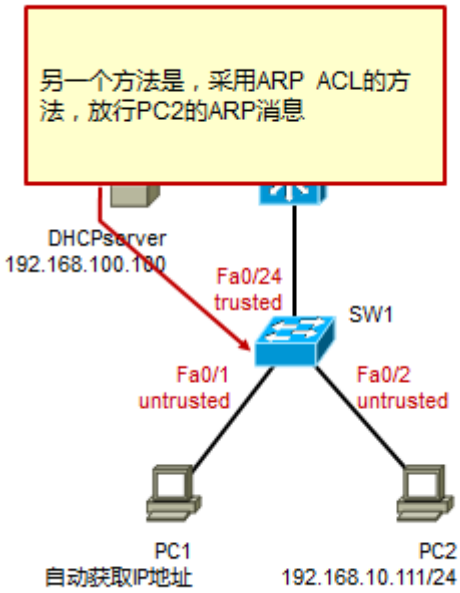
SW1的增加配置如下：

```
ip dhcp snooping binding 000b.5f35.70a1
vlan 10 192.168.10.111 interface fast 0/2
expiry 1000
```

SW1上使用show ip dhcp snooping bingding
可以看到手工创建的绑定

如此一来PC1即可获取到PC2的MAC，从而ping通PC2

另一个方法，是用 ARP ACL：



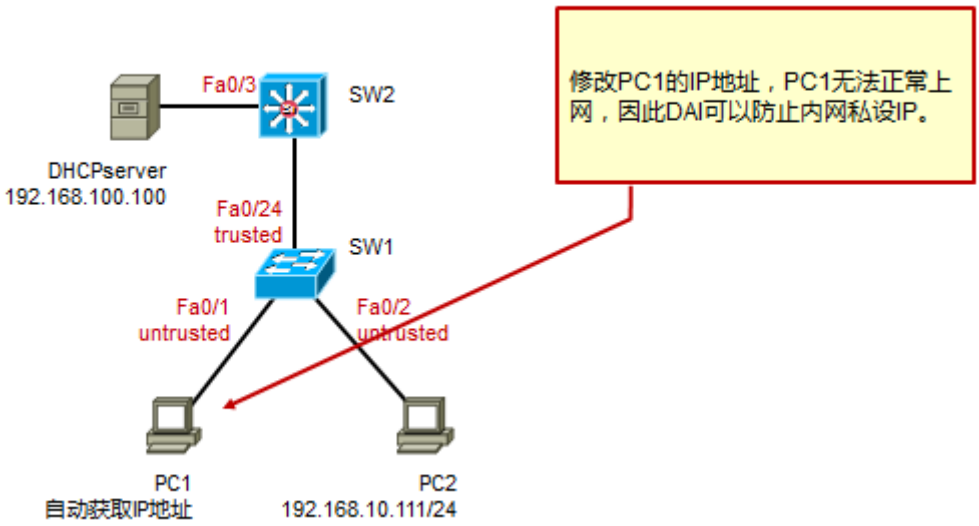
SW1的增加配置如下：

```
arp access-list ccie
 permit ip host 192.168.10.111 mac host 000b.5f35.70a1
!
ip arp inspection filter ccie vlan  10
```

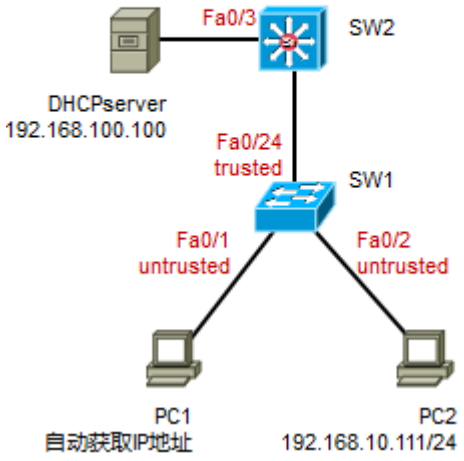
有了ARP ACL，同时应用到DAI上，PC2的ARP消息在进入untrusted 接口fa0/2后，会被ARP ACL ccie所匹配并放行。

PC1及PC2即可通信

再来做一个测试，就是 PC1 私自修改 IP 地址：



由于 DAI 开启后，下联的 PC 如果段时间发送大量的无效 ARP，可能会导致交换机性能大量损耗，也就是 DoS 攻击。那么建议在接口上限定 ARP 消息门限。



SW1的增加配置如下：

```
errdisable recovery cause arp-inspection
errdisable recovery interval 30
Interface fast 0/1
  ip arp inspection limit rate 10
```

为了防止开启了DAI的交换机受到DoS攻击，建议在接口上限制ARP消息的数量，当超出这个阀值，接口回被err-disable，搭配errdisable recovery 命令，可以使得交换机接口能够在一定时间后自我恢复（前提是违例行为停止）。

什么是网络割接

红茶三杯 CCIE 学习文档

文档版本：	1.0	更新时间：	2013-03-26
文档作者：	红茶三杯	文档地址：	http://ccietea.com
文档备注：	请关注文档版本及更新时间		

1 写在前面的话

最近经常有同学问道，网络工程业务有哪些，什么是网络割接等问题。今儿个得空，写个小文说说网络割接的那些事儿。本文主要介绍的内容：

- 网络工程涉及的业务类型（技术相关）有哪些
- 什么是网络割接
- 举例说一下网络割接（不涉及技术细节）
- 网络割接过程中需要注意哪些事项
- 网络割接的文档交付材料如何撰写

2 关于网络工程业务

网络工程行业，业务种类是非常繁多的，单从技术（不区分售前和售后）的角度看，常见业务就有：

- 售前测试
- 售前解决方案咨询
- 网络解决方案可行性分析
- 新项目实施（如建网等）
- 网络改造或扩容、网络维护
- 故障处理
- 软件升级
- 技术、项目、产品培训

- 巡检、网络安全及健康状况分析
- 驻场服务
- 其他

所以，网络工程行业技术业务的种类是非常繁多的，具体要看岗位及客户要求、公司的全局安排。

3 网络割接

我想再次明确一点，在这里我们探讨的“网络”的概念，不是指小型的家用宽带这种小规模网络，而是大、中型的工业级别网络，如企业、金融机构、政府机构、大型教育园区网等类型项目。小型家用网络，自己瞎鼓捣鼓捣就成了，大不了自家上不了网，但是工业、商业、事业用途的网络，就需要专业的技术服务了。

假设我们有这么一个客户，客户有个新的园区刚刚建成，园区内包括建筑物若干，地理覆盖面也较广。园区土建施工等这块我们就不说了，园区建成后，肯定是需要一个专用网络的，用于承载公司的业务流量，可能是无线，也可能是有线，或者是有线、无线的融合网络。不管怎样，肯定是需要一个大规模的园区网络来承载电子化的业务交互数据的，最少，园区内的用户总要上网的吧？

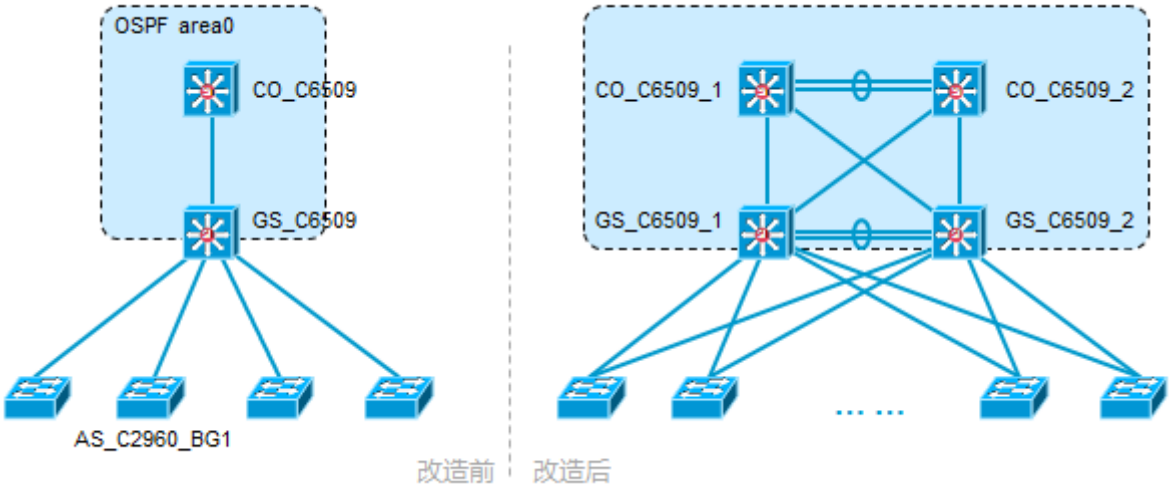
那么有建网需求，就有网络产品或解决方案的销售行为，这里我们直接忽略销售环节，现在假设所有的售前和销售环节都定下来了。那么接下去就是**新网实施**。网络建成后，一般来说服务交付中还有一块，就是给**客户培训**，培训内容是关于网络就相关产品的实施、使用情况、维护方法等。紧接着网络就开始用了，使用过程中，可能会有些小的调整性需求，例如 VLAN、IP 地址段的新增等，这就是**网络维护**。再有，网络难免出现故障，那么就需要做**故障处理**。

那么如果网络在运行一段时间后，需要对网络进行改造、升级、迁移等变更，同时这些网络操作行为，又是发生在一个正在承载业务流量的网络上，那么这种行为，就称为割接。

网络割接动作，可能是为了调整网络结构、新增或者替换网络设备、更换线路、更改设备配置或者其他针对网络的变更需求。

网络割接是一个相对难度较高的动作，尤其是在应对营运商、金融、政府或者大型企业的核心网络，每一个割接的动作都需要非常谨慎，因为如若操作失败，造成的影响是非常恶劣的。所以网络割接，对专业工程师的技术、技能、体力，脑力，经验等等都提出了一定的要求。

4 网络割接范例



这里来看一个典型的案例，我们重点理解网络割接这个行为。一个网络在改造前，网络结构见“改造前”，可以看得出，网络结构比较简单，而且存在单一设备、单一链路的缺陷。设备和链路均没有冗余。那么这个网络运行一段时间后，网络可能会时常出问题，例如接入交换机如果上联链路出现故障，那么整台接入交换机上所连接的用户都无法上网了。于是客户提出网络的改造需求，总体的目标是 1、新增汇聚及核心层设备；2、接入交换机全改为双链路上联到汇聚交换机；3、实施二层、三层冗余技术以提高网络的冗余度和可靠性；4、重新规划 OSPF 网络模型；6、调整数据流走向。

这个网络，可能每时每刻都有业务流量在上面被转发，那么割接的操作就需要充分考虑到这些流量，以及可能性的风险，并且做好充分的准备，以及与客户的充分沟通。尽量做好平滑的切换，降低对业务的影响。

针对这个改造目标及客户的需求，工程师要做的工作主要有如下几点：

1) 彻底摸清现网网络环境，包括但不限于
   网络物理、逻辑拓扑；网络设备的配置以及相关的运行参数； 业务数据流走向等
2) 了解清楚客户对改造的期望及需求
3) 针对客户需求，撰写网络改造（或割接）方案，关于报告涉及的内容，在下一小节探讨
4) 在制定割接方案的过程中，需要充分考虑到每个割接步骤对业务流量的影响，并且为每个步骤考虑回退方案。是否会断网？如果会，断多长时间，是否需要客户配合下发通知等；
5) 将撰写好的割接方案交予给客户并与客户进行充分的沟通，并最终确定割接方案、割接时间，以便做好相关的准备工作，一般来说，割接的动作如果太大，往往会选择业务流量较小的时候进行，例如半夜~
6) 如果必要的话，还可选个时间在现网中做小范围测试

所以，对于网络割接我们可以看出下面几个特点：

1) 这玩意儿需要细心，细心，再细心
2) 技术的角度，各种问题需要考虑清楚，例如如果是路由复杂的环境，那么割接过程中，以及完成后，是否有可能出现路由环路之类的问题。
3) 割接的准备工作非常重要
4) 割接往往是发生在一个已经在承载业务流量的网络中，为了减少割接对业务的影响，一般会选择业务流量较小的时候进行，例如半夜，或者放假时间等。从事运营商网络建设的兄弟，就经常半夜工作，非常辛苦。所以身体非常重要，再者，有时长时间的割接在机房中进行，机房，你懂的，注意保温措施，带上点小面包，有时也是必要的。

5 割接方案需注意什么

方案只是割接思想的一个交付物，有的时候是给客户看的，有的时候是给领导看的。当然，关键是自己要完全清楚整个割接过程，一般来说，一份割接方案会包括如下内容（当然，根据具体情况而定）：

1) 文档目的
2) 项目背景
3) 网络现网
   a) 网络拓扑
   b) IP、VLAN 信息
   c) 路由及数据走向
   d) 设备配置备份
4) 割接目标
   a) 改造后的网络拓扑
   b) IP、VLAN 信息
   c) 路由模型
   d) 数据走向
   e) 新增设备型号及版本信息
   f) 设备管理信息
5) 实施工艺
   a) 步骤一：xxxx；（同时包含回退方案，下同）
   b) 步骤二：
   c) 步骤三：
   d) 步骤四：
   e) .....
6) 割接进度及各方职责

7) 测试环境及脚本

8) 其他交付物

红茶三杯

网络工程 | 项目管理 | IT 服务管理 | CCIE 培训

学习 沉淀 成长 分享

微博：http://weibo.com/vinsoney

博客：http://blog.sina.com.cn/vinsoney

站点：http://ccietea.com

网络工程技术文档编写规范

版本 **V2.1**

密级 □开放 ☑内部 □机密

类型 □讨论版 □测试版 ☑正式版

修订记录				
修订日期	**修订人**	**版本号**	**审核人**	**修订说明**
2010-04-22	**红茶三杯**	V1.0		
2012-05-27	**红茶三杯**	V2.0		

1 前言

1.1 文档目的

为提高我司技术服务工程师工程文档编写能力、统一文档呈现界面、统一化文档管理、提高文档专业性及可读性，特此下发此规范。

良好的文档呈现能力，能够：

- 更好的沉淀、传播知识
- 加快个人、组织的“知识管理”进程；提高知识沉淀质量
- 更好地体现工作价值及提升个人品牌形象，从而提升团队或公司的专业形象
- 提高工作效率
- 保障优先沟通

1.2 网络工程文档类型

我公司层面文档类型主要分为：

解决方案类文档（面向内部、面向客户）

工程实施类文档（面向交付）

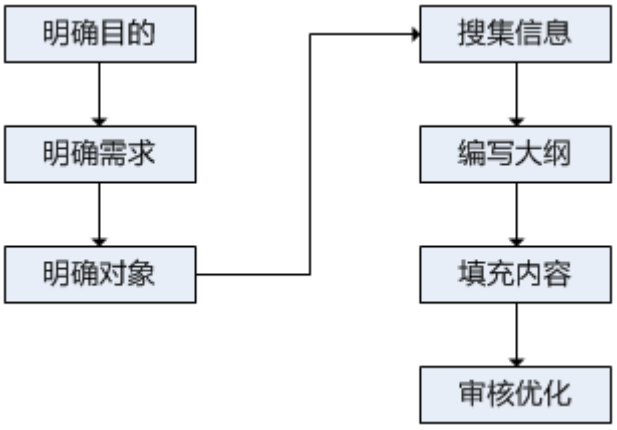
技术研发类文档（面向内部、知识分享、面向客户培训）

知识沉淀类文档（面向内容）

产品展示类文档（面向客户、面向网络）

2 文档编写规范

2.1 文档编写思路



2.2 步骤分解

2.2.1 明确目的

明确文档编写的目的，将直接影响整个文档撰写过程以及交付的成果，因此，务必要摸清楚交付目的：

例如：

1. **交付一份测试手册，关于 S5750 设备的 Ipv6 自动 6to4 隧道的功能，用于指导一线工程师工作**

   手册的目的是：指导一线工程师工作，那么这份文档，就要站在一线的角度去写，理解一线需要什

么，或者一线的工作习惯等等。

2. **交付一份网络培训 PPT，用于在项目实施后对客户进行培训**

PPT 的目的是对客户进行项目实施后的培训，那么文档就要积极呈现服务的价值，以及公司的解决方案价值。同时帮助客户认识自己的网络，以及学习如何维护网络。

2.2.2 明确需求

这里的需求，并非简单意义上的概括，例如：“写一个项目实施报告，把这个项目的实施过程写一下”这个需求就太笼统了。

一个科学的需求，应该具备“SMART”原则，例如：

“写一份项目割接方案，详细描述网络现状、网络割接后的模型、割接步骤以及回退方案、IP 及 VLAN 的割接前后规划、各方的职责、工时及进度表等等，下周二之前提交”

另一点要注意的是，一定要“找对人”确认需求，例如一份文档，如果使用对象是客户，但是交付对象是主管，那么这里需求应该和谁去确认呢？要注意的一点是，如果需求不明确，很可能产生无用功，浪费人力物力及时间。

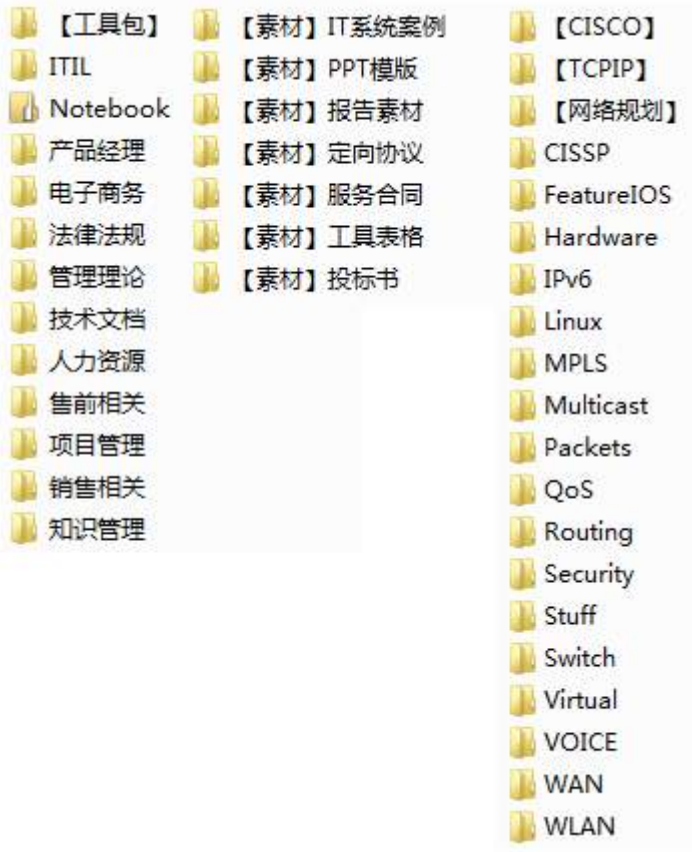
2.2.3 明确对象

关注的对象应该有：文档的交付对象（提交给谁）、文档的使用人（谁用）明确对象，有助于找准准心。对象的明确，能够使得文档呈现更加的准确。

例如提交一份《故障处理报告》给内部，那么自然是实事求是，无论是产品或是环境问题可尽数呈现，但是如若是同一个内容，提交给客户，那么就要考虑下呈现的方式了，是否需要规避对品牌或者产品不利的因素等。再如，如若提交《实施手册》给内部工程师，那么可加强文档的专业度和针对性，如若提交给集成商，则需考虑集成商的技术水平，可适当加强文档的易用性和通俗性。

2.2.4 搜集信息

搜集文档所需的信息：例如撰写技术文档时，可能需要用到的设备参数、图片素材、网络环境素材等等。建议在日常工作或学习过程中，建立自己的 KM，也即“知识管理”体系，将自己的所有“知识”，例如工作的积累、学习的材料，显性（看的见摸得着的）以及隐形（逻辑的如方法或技巧）保持阶段性整理和沉淀的习惯，例如笔者的个人电脑，建立知识管理目录：



如此一来信息的定位就会非常快速。关于知识管理的进一步内容，详见 bbs.SPOTO.net 雏鹰部落论坛的 BOOTCAMP 板块中“知识管理”一帖。

2.2.5 编写大纲

大纲的编写是文档成型非常关键的第一步，明确了大纲，文档的总体架构就出来了。同时，把握好文档大纲，能够使得文档与需求的匹配度更高、文档的返修率降低等。

在文档撰写初期，强烈建议先编写大纲，并且与文档干系人，尤其是使用人确认大纲信息。

例如，撰写一个《xx 网络割接方案》，大纲可参考如下：

- 方案背景
- 现网环境
- 改造后环境
- 割接过程（含回退方案、二三层信息规划）
- 配置规范
- 实施进度
- 需配合项

当然，实际工程中，大纲的层级要远远比这复杂，最重要的是在明确需求和背景的情况下，保证文档有清晰的逻辑机构。

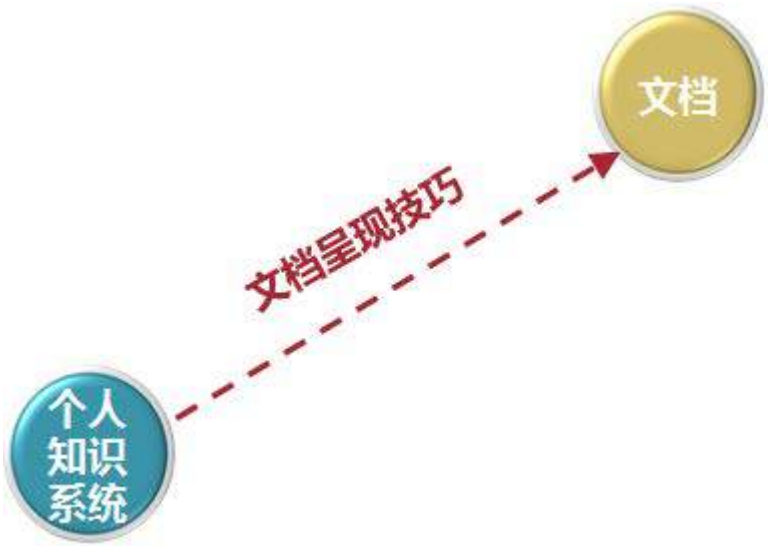
2.2.6 填充内容

在完成大纲的基础上，填充文档的主题内容，这一步是工作量最大、最耗时的环节却也是最重要的环节。在内容呈现上，可参考本文档“呈现规范”一节，内有详细描述及呈现案例。

一个好的文档，给人的感觉是专业、规范、工整、简明扼要、可读性高、使用方便、维护方便等。

要编写好一份文档，除了上述说的种种，下面谈的云云，**关键还在于个人对于所呈现的知识的理解程度、个人知识储备情况以及非常重要的一点，呈现技能**。

编写一份项目实施方案，首先要做到的是，对于该网络、该方案，你是否足够的理解，是否完全消化，当然这不仅仅是单一的 CASE，甚至可能要求到全产品线亦或是解决方案的理解。个人知识的储备情况包含的方面可能就更加多元化，例如撰写《XX 银行 X 省网点改造计划》，则需要的不仅仅是对于网络或产品以及技术的理解，同时还需要对于金融行业背景（银行的组织架构、办事流程、实施规范等）、技术服务行业背景（工程师的服务规范、服务成本及价值传递等）。最后，文档呈现，即是将对本 CASE 的理解，加之自身的知识储备，通过呈现技巧的运用将内容表现出来。个人的呈现技巧只能是知行合一，多看多练。在一个具备知识管理意识的团队中，例如厂商内，文档或知识往往有比较丰富的储备和统一的规范，因此多看多模仿是一种比较快捷的学习手段，在此基础上进行大量的编写进行积累和提高，最终形成自己的个人品牌形象。毫不夸张的说，在熟练掌握文档呈现技能后，任何书面的，甚至是口头的表达，都会变得有章法、条理清晰。



2.2.7 审核优化

在组织内部，文档和知识有一个统一化的管理，部分文档的发布有一套审核流程。

例如大项目的实施方案，在落地前，都需要有关部门，如工程管理部、二线或产品部门审批；又如部分产品手册或 FAΘ、升级报告等由于涉及机密，可能会处于受控状态，不得随意发放。

2.3 命名规范

使用同一的文档命名规则定义公司的知识文档：

【工程拓扑】xx 政府新大楼改造项目 20100324

【实施报告】xx 政府新大楼改造项目 20100324

【故障处理】xx 大学 ASA 出口策略故障 20100324

2.4 呈现规范

1. 统一的文档格式；
2. 图文并茂，避免大面积的文字堆积；合理运用表格来呈现内容或数据；
3. 文字描述言简意赅，不繁琐、不罗嗦；
4. 书面化用语；不携带任何情感因素；避免过分主观的描述；
5. 合理断句、合理使用标点符号；

6. 注意段落结构、善用项目编号式的文字呈现方式；
7. 合理控制段落及行间距，增加文档可读性

2.5 文档案例

【案例 1 】 常见写法：

网络结构采用接入交换机双链路上联、双核心热备的解决方案。接入层双链路上联并且使用 MSTP 实现 2 层冗余，双核心之间运行 VRRP 热备份协议为内网用户提供冗余网关，同时各自上联至透传的防火墙，由防火墙将数据透传至出口设备。

改良写法：

本项目拓扑结构描述如下：

- 接入交换机双链路上联至两台核心交换机
- 核心与接入之间使用 MSRP+VRRP 解决方案使得链路及网关都能实现冗余热备
- 核心两台三层交换机各自通过 xx 链路上联至防火墙。防火墙使用透传模式
- 防火墙通过 x 链路上联至出口路由器

问题描述：

尽量避免语句过长，否则会在一定程度上会增加阅读困难。合理使用断句，合理使用项目编号式的描述方式。

【案例 2】 常见写法：

故障发生后，客户的网路挂了，业务根本办不了

改良写法：

故障发生后，客户网络瘫痪，业务停滞

问题描述：

避免口语化，避免情绪化

【案例 3】 常见写法：

客户的网络为双核心，两台 C6509，各自一条光纤上联到一台防火墙，防火墙再各自上联到出口路由器。内网有两台 WEB 服务器，在出口路由器上，分别对这两台服务器进行映射。网络有 2 个出口，电信及网通，出口路由器分别连接到电信及网通出口，在两台

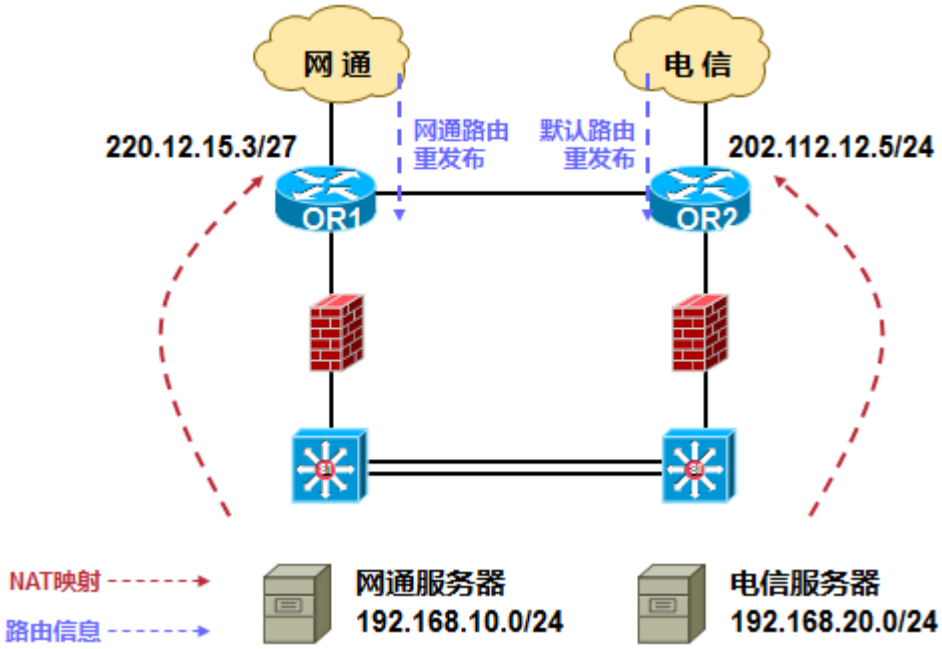
路由器上将内网的两台服务器进行映射到出口。内网运行 OSPF，在网通出口路由器上定义网通静态路由，并重发布进 OSPF，电信出口路由器重发布默认路由进 OSPF。

问题描述：

大面积文字堆积，流水账式的描述方式，逻辑紊乱，无段落或表达层次。

无配图，非常不直观

改良写法：



现场环境如上图：

1、 内网核心层为两台 C6509 交换机，分别下挂 2 台服务器。这两台服务器分别对外面向电信及网通用户提供 WEB 服务；

2、 两台 C6509 各自单链路上联至防火墙，经由防火墙透传至出口路由器；

3、 两台出口路由器分别连接网通及电信出口线路。网通路由器上，使用静态路由指向网通网络资源，电信路由器上则配置默认路由。内网运行 OSPF，网通出口路由器将网通路由明细注入 OSPF，电信出口路由器则注入默认路由；

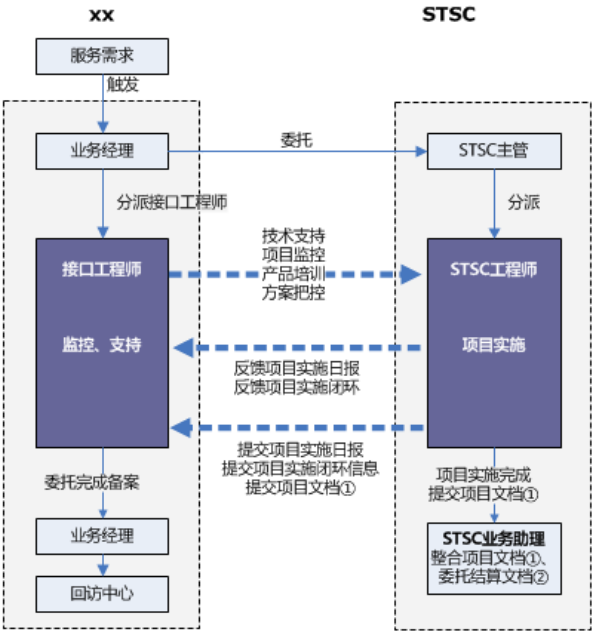
4、 分别在网通及电信出口路由器上，对内网的网通及电信服务器做静态 NAT 映射，使得外网能够访问服务器的 WEB 服务。

【案例 4】 **常见写法：**

流程如下：yy 客户触发项目需求，xx 业务经理向 STSC 省区主管下发服务委托，并分派

给双方项目工程师……

改良写法：



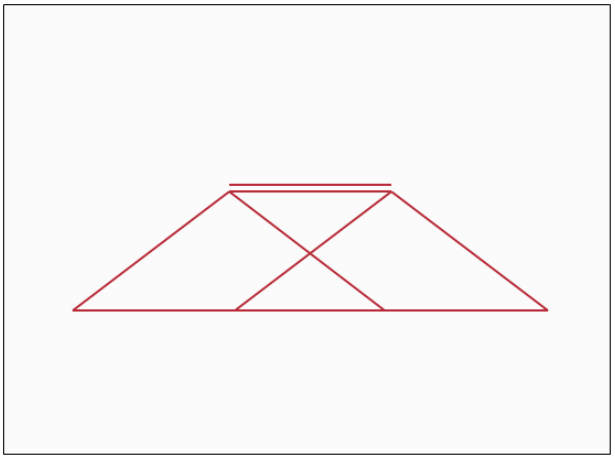
问题描述：

合理利用图表或流程图来呈现内容

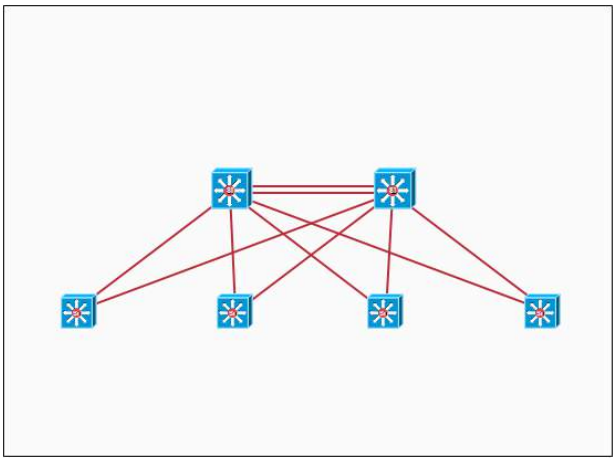
3 工程拓扑绘制规范

3.1 绘制步骤

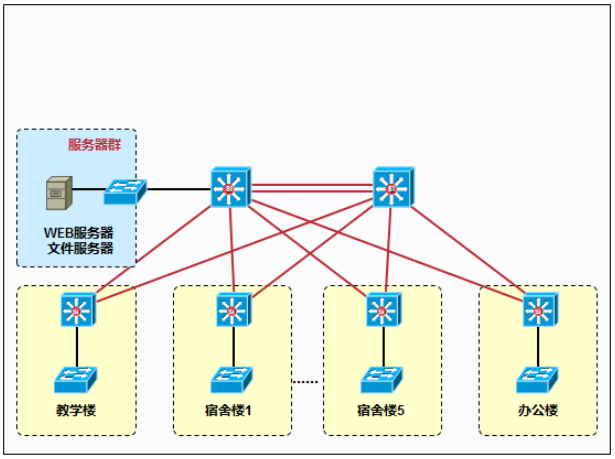
1. **在纸上绘制拓扑草稿**
2. **在工具环境中，描绘拓扑架构，可借助参考线、色块、线条等等**



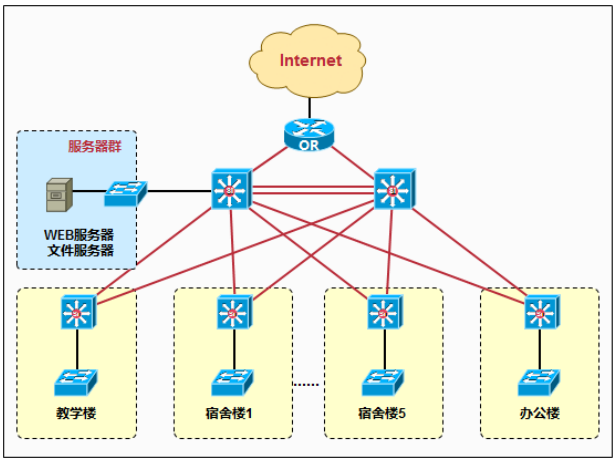
3. 放置拓扑骨干设备图标、完成设备互联



4. 继续完成拓扑、标记设备、标记拓扑



5. 完成终端及扩展信息、辅助信息



注意，这只是一个小型网络拓扑的示例，大型工程拓扑的绘制可能会更加复杂，需要考虑的元素也更多，因此，切不可死记硬背，方法要懂得变通。

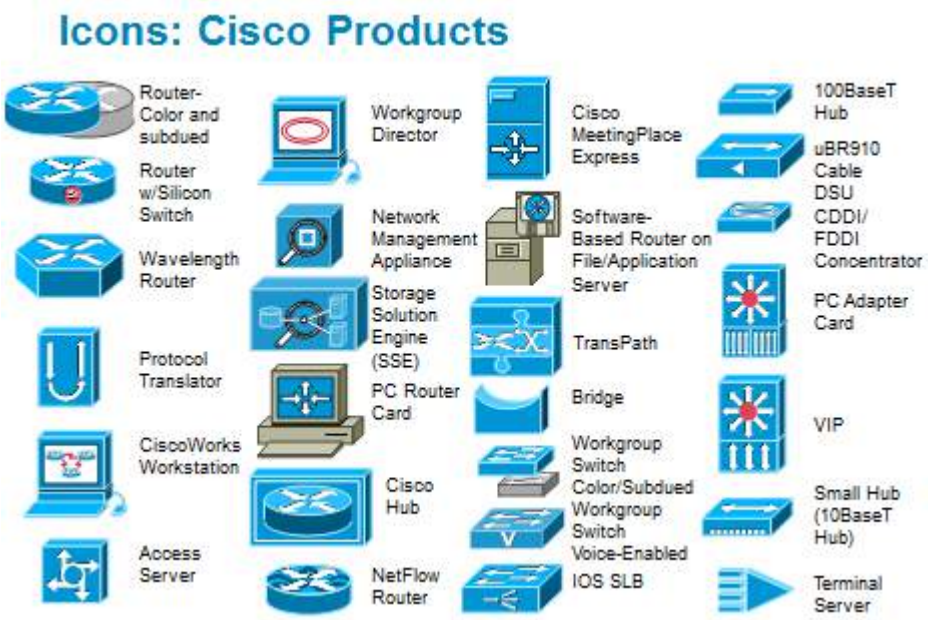
3.2 绘制工具

Powerpoint	PPT 适合在技术文档中呈现网络拓扑，例如解决方案、产品演示、项目培训等场合
Visio	适合呈现专业的大型工程拓扑；专业化工具；适合彩绘

3.3 绘制规范

3.3.1 图标规范

使用统一的图标库，如 CISCO Icons



3.3.2 图标标识规范

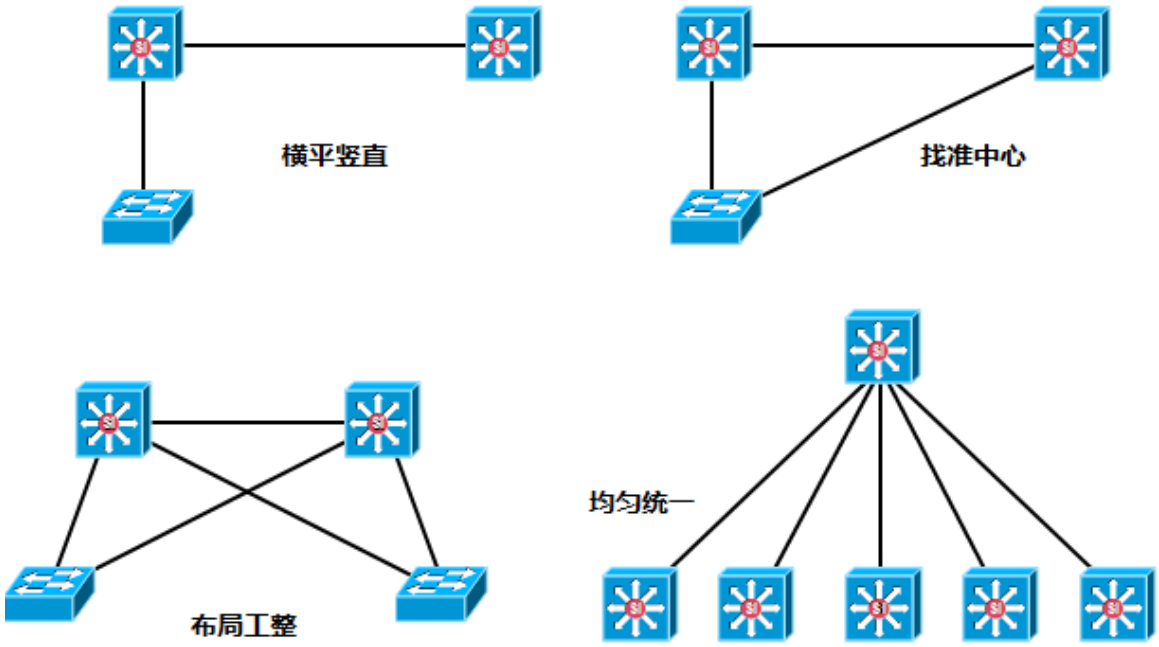
3.3.3 互联线条规范

使用统一的线条规范，并且在工程拓扑中适当位置给出图例，例如（仅供参考）：

- 百兆网线：0.75 磅，黑色实线
- 千兆网线：1.5 磅，黑色实线。千兆光纤：1.5 磅，红色实线
- 万兆网线：2.25 磅，黑色实线。万兆光纤：2.25 磅，红色实线
- 广域网线路：1.5 磅，蓝色实线

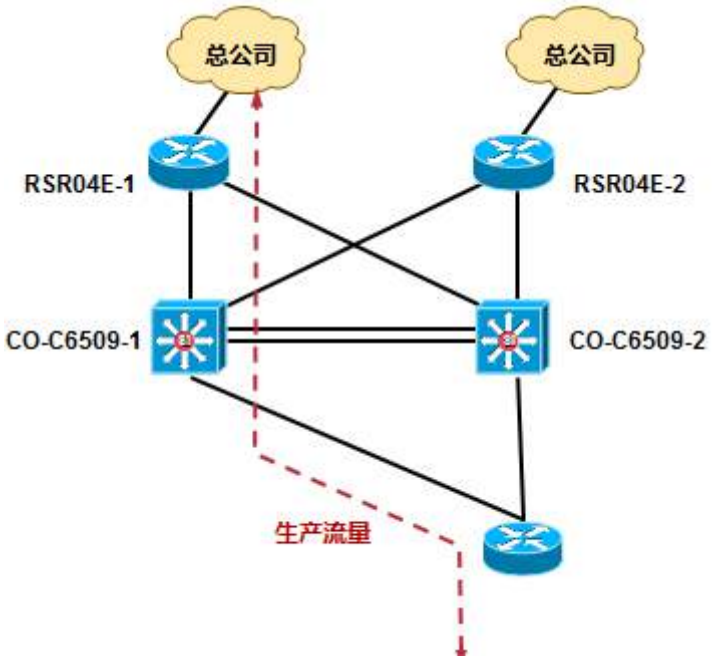
一般情况下，在工程拓扑的适当位置，需给出图例及描述：

3.3.4 设备分布规范



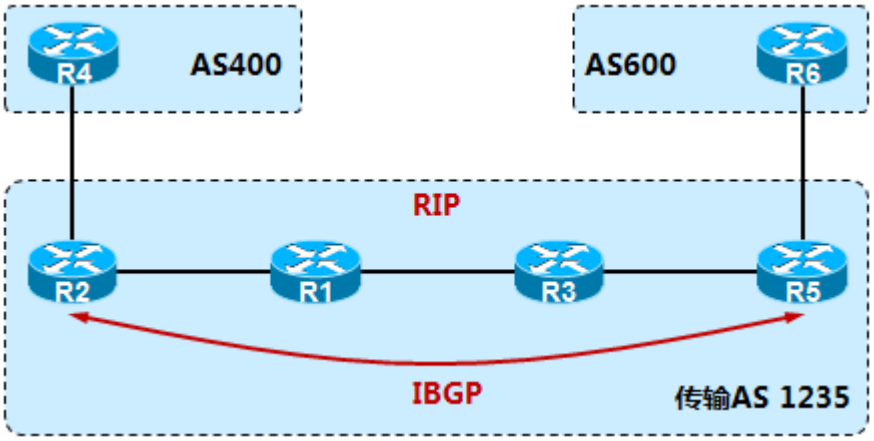
3.3.5 箭头使用规范

箭头在拓扑绘制中用途较多，如：标记信息、标识数据流量等

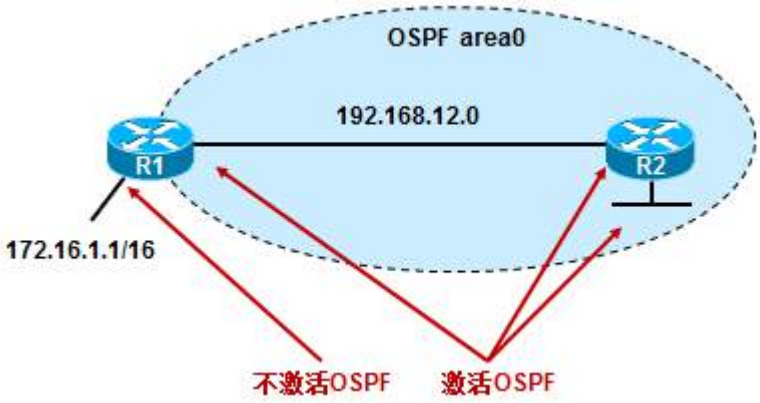


3.3.6 路由选择域

使用虚线框的填充块来表示动态路由选择域，不同的路由协议可使用不同的颜色进行区分；

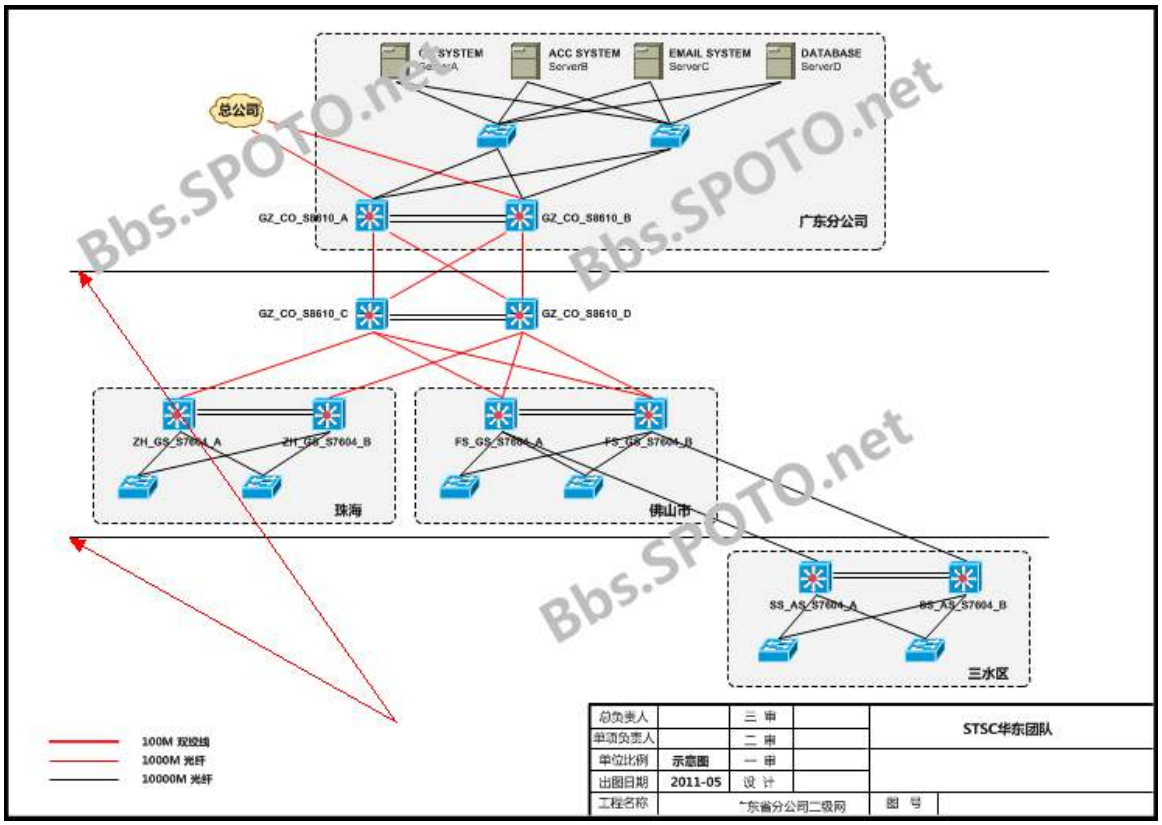


落在虚线框范围内的设备接口，才会激活动态路由协议，因此，注意虚线框的覆盖。



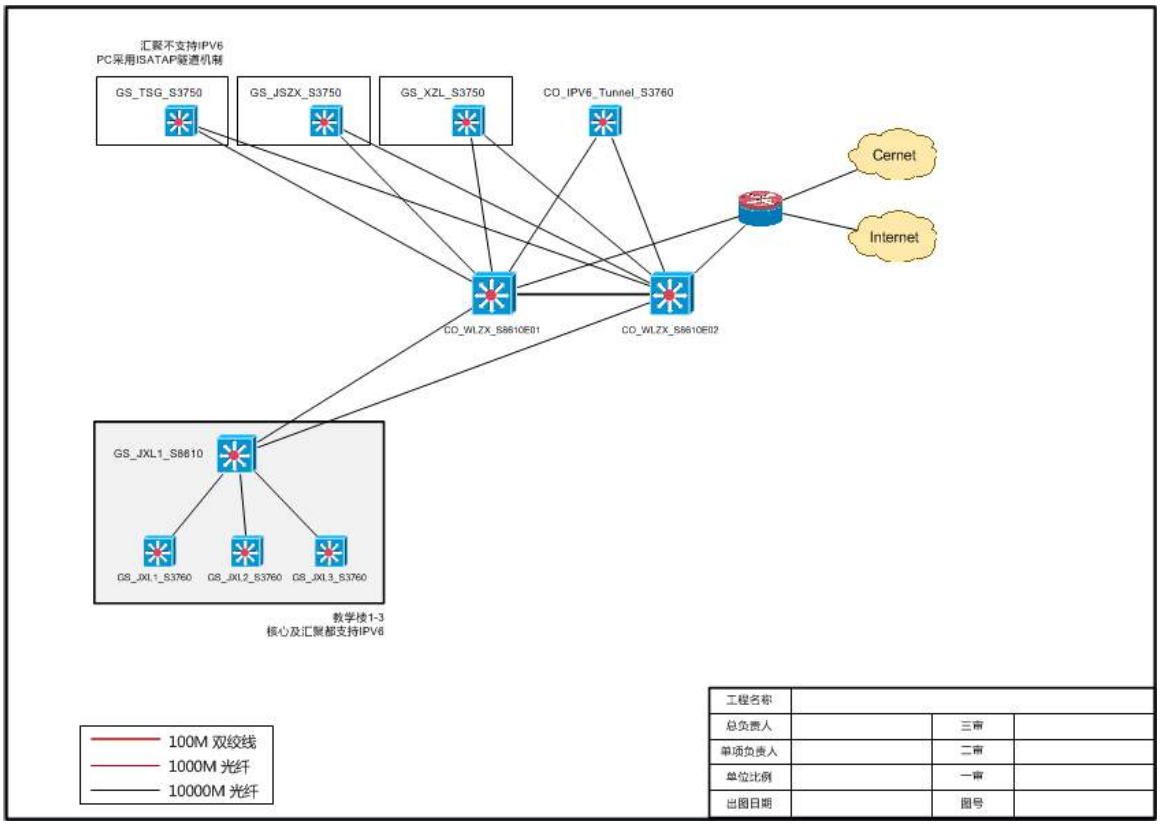
3.3.7 拓扑构图

可借助辅助线、色块、线条等手段帮助完成拓扑构图。

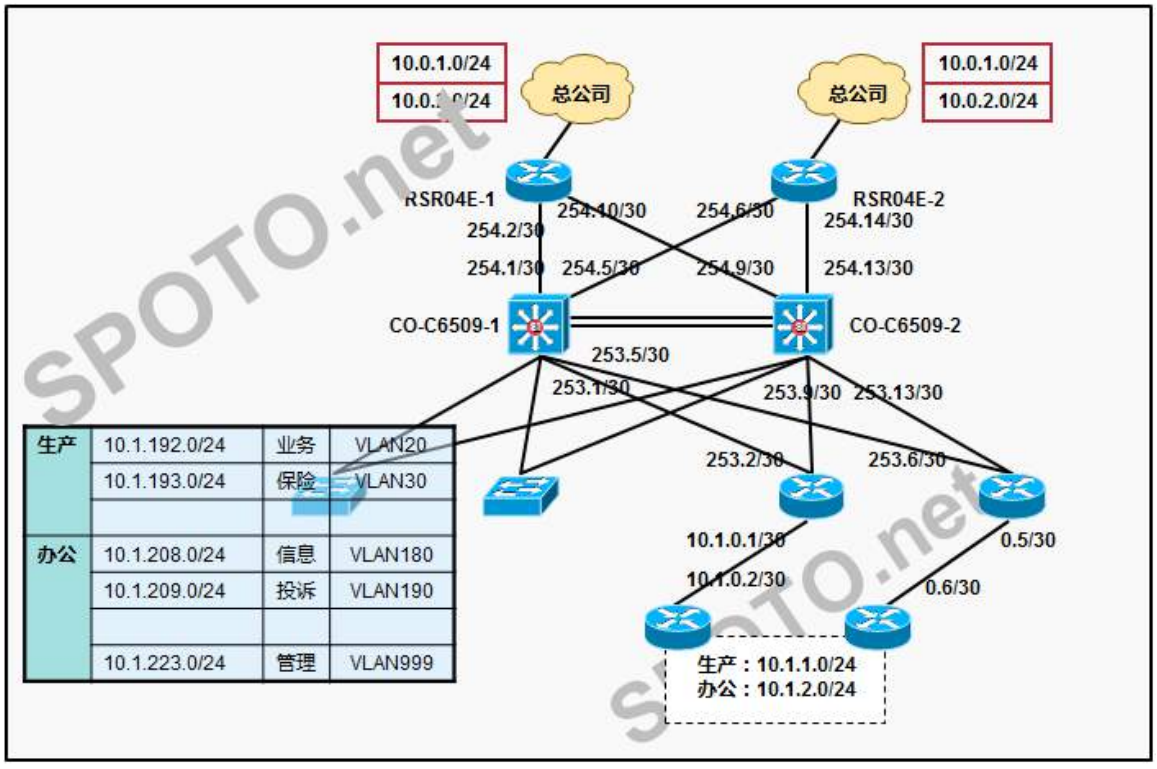


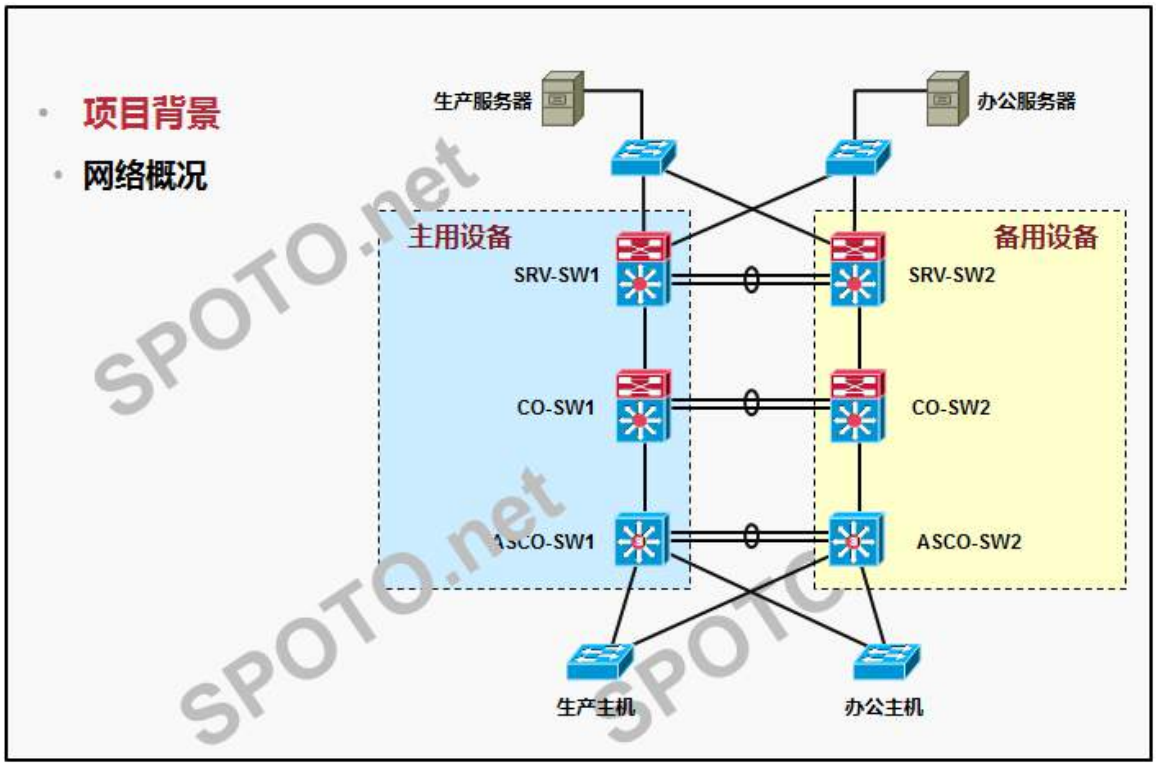
3.3.8 拓扑模板

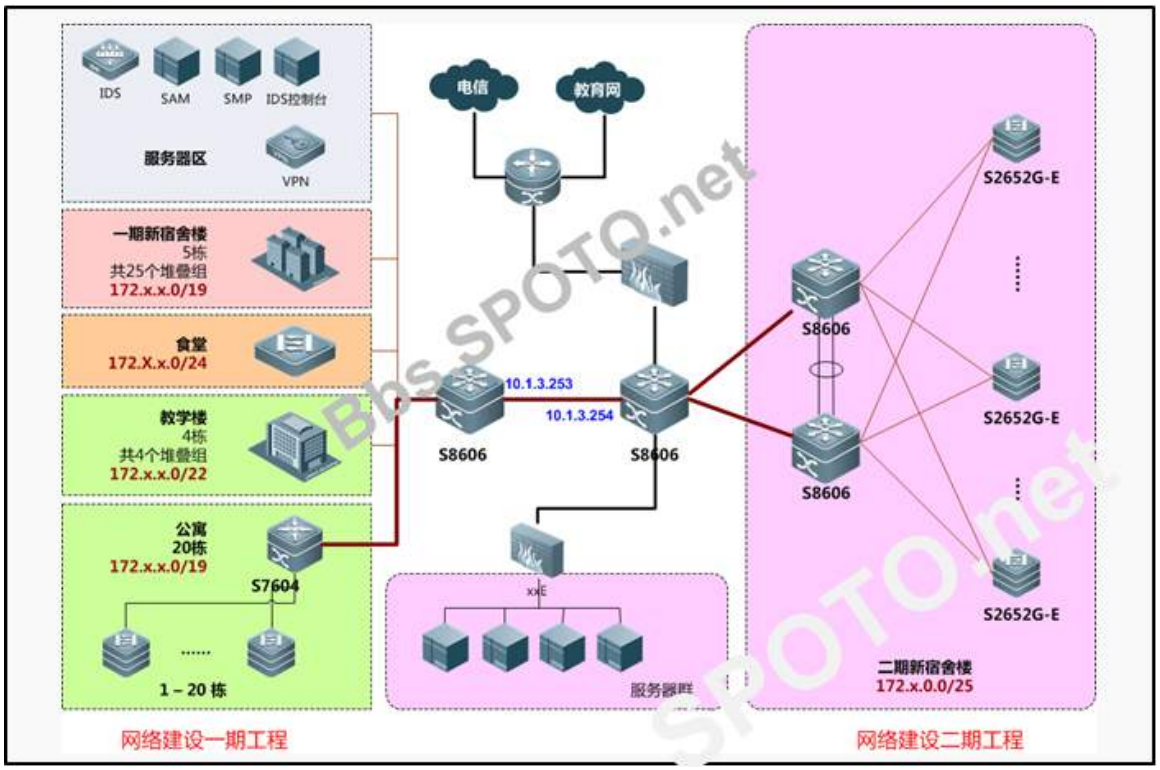
注意，工业中的工程拓扑有较为严格的绘图规范：


汇聚不支持IPV6
PC采用ISATAP隧道机制
GS_TSG_S3750
GS_JSZX_S3750
GS_XZL_S3750
CO_IPV6_Tunnel_S3760
Cernet
Internet
CO_WLZX_S8610E01
CO_WLZX_S8610E02
GS_JXL1_S8610
GS_JXL1_S3760
GS_JXL2_S3760
GS_JXL3_S3760
教学楼1-3
核心及汇聚都支持IPV6
100M 双绞线
1000M 光纤
10000M 光纤
工程名称
总负责人
单项负责人
单位比例
出图日期
三审
二审
一审
图号

3.4 效果展示


10.0.1.0/24
10.0.2.0/24
总公司
总公司
10.0.1.0/24
10.0.2.0/24
RSR04E-1
RSR04E-2
254.10/30
254.6/30
254.14/30
254.2/30
254.1/30
254.5/30
254.9/30
254.13/30
CO-C6509-1
CO-C6509-2
253.5/30
253.1/30
253.9/30
253.13/30
253.2/30
253.6/30
10.1.0.1/30
10.1.0.2/30
0.5/30
0.6/30
生产：10.1.1.0/24
办公：10.1.2.0/24
生产 10.1.192.0/24 业务 VLAN20
10.1.193.0/24 保险 VLAN30
办公 10.1.208.0/24 信息 VLAN180
10.1.209.0/24 投诉 VLAN190
10.1.223.0/24 管理 VLAN999
SPOTO.net


项目背景
网络概况
生产服务器
办公服务器
主用设备
备用设备
SRV-SW1
SRV-SW2
CO-SW1
CO-SW2
ASCO-SW1
ASCO-SW2
生产主机
办公主机


IDS
SAM
SMP
IDS控制台
服务器区
VPN
一期新宿舍楼
5栋
共25个堆叠组
172.x.x.0/19
食堂
172.X.x.0/24
教学楼
4栋
共4个堆叠组
172.x.x.0/22
公寓
20栋
172.x.x.0/19
S7604
1 – 20 栋
电信
教育网
10.1.3.253
10.1.3.254
S8606
S8606
S8606
S8606
S2652G-E
S2652G-E
S2652G-E
二期新宿舍楼
172.x.0.0/25
网络建设一期工程
网络建设二期工程

4 笔者的话

注：本文档为 SPOTO IT 人才机构“**BOOTCAMP 网络工程师带薪实战集训营**”的 培训材料。

如有任何意见或建议，可至 SPOTO 雏鹰部落发帖交流：http://bbs.spoto.net

或粉本人微博：http://weibo.com/vinsoney

或移步本人博客：http://blog.sina.com.cn/vinsoney

网络工程初学者的学习方法及成长之路

文档版本： 1.0
更新时间： 2013-01-22
文档作者： 红茶三杯
文档地址： http://ccietea.com
文档备注： 请关注文档版本及更新时间

红茶三杯
网络工程 | 项目管理 | IT 服务管理 | CCIE 培训
学习 沉淀 成长 分享
微博：http://weibo.com/vinsoney
博客：http://blog.sina.com.cn/vinsoney
站点：http://ccietea.com

现在基本每天都会在微博和 QQ 上收到来自各地网友的提问，关于技术、项目、职业规划等等；有大学生，有在职的，也有不少技术狂热者。网络工程师是一个热爱技术、热爱生活，渴望自我完善和成长的敏捷型群体，这种交流让我受益匪浅，份外珍惜。

还记得自己最初接触网络工程相关专业知识的时候，尚在读大一，忘记通过什么渠道了解到思科，怀着对网络技术的狂热，开始这趟旅程并一发不可收拾。母校南昌大学是思科网院，大二那年果断报了个 CCNA 课程开始学习，一头扎进书本和实验里，没心没肺的学，没心没肺的做实验，没心没肺的记笔记，孤独但快乐，充实又不寂寞，这种对新事物及新技术（对于我而言）的热衷和对自我成长的渴望，伴随至今，不曾消散。遗憾的是，南昌的 IT 氛围和 IT 人才需求有限，偌大个省会城市当时竟然开不起一个 CCNP、CCIE 班（没人学），苦苦等待无果，我只能开始在大学期间的自学。

相信我的经历，跟许多网友类似，自学，一切靠自己，缺乏科学的、系统性的指导，缺乏资源，只能自己慢慢摸索。虽然在自学过程中沉淀了不少珍贵的技能：自学能力、信息检索能力、实践能力、知识管理意识等等，但是弯路是避免不了的，而且，学习没有计划，缺乏紧凑度，没有圈子，这些都是自学的短板。

正因如此，多年之后，利用点业余时间，自己还是弄了个小站：ccietea.com，正是希望能够利用这些个年头积累的教学材料以及个人知识笔记帮助到与我有类似经历的技术爱好者、初学者们，让大家有迹可循少走弯路。羞愧的是个人时间和精力有限，小站几乎荒废，远没有达到自己预期的效果。只能是点到为止了。

这里还是想跟大家聊聊网络工程初学者的学习方法和成长问题，帖子写的有点啰嗦，现在人多浮躁，不知道有多少人能看完，又能够帮助到多少人，也罢，就当是随笔唠叨。

1 知识参考体系

任何一门专业知识，如果有个科学的、系统的、完善的参考体系，那么学习起来将更有针对性，更系统。CISCO 认证及技术培训是本人这些年专注的领域，因此向大家推荐 CISCO 的认证及知识体系，当然，目前业内还是有不少类似的知识及认证体系，例如 Juniper、华为等，大家可以根据自己的需要进行选择。CISCO 认证体系为网络工程领域的从业者规划了一个动人的蓝图，无论最终是否考取思科认证，思科的这套体系依然是经典，依然是值得学习和研究的。

对于网络工程初学者而言，CCNA 是一个绝佳的入门课程，从 NA 到 IE，似乎给我们指了一个方向，事实上正是如此，对于初学者完全可以参照这个路线来逐步的完善自己的理论体系。当然，有条件的童鞋可以考虑找个靠谱儿的培训机构进行学习，本文着重针对没有条件或者想自学的童鞋提点小小的建议，仅供参考。

2 学习资源

1. 专业教材及书籍（或业人士推荐度较高的书籍）

专业性的书是非常重要的，应该说是最重要的知识来源之一，当然，强烈建议大家在学习过程中不要贪多，有些书本是要精读深读的。举些例子：CCNA 阶段我们推荐大家阅读《CCNA 学习指南》，进入 CCNP 我则会推荐经典书籍 :《TCP/IP 路由技术》卷一、卷二，以及 CCNP 的相关学习指南，而针对各个技术专题有又相应的经典书目。建议在攻克一门课程或一个技术专题的时候，选择一到两本业内好评度较高的书籍着重研读。

2. 教学视频

互联网及移动互联网的发展，极大的改变着我们的生活，也带来了许许多多的便利。现如今互联网的资源浩瀚无边，基本上你能想到的，在互联网上都能找到相关的内容。许多培训平台、讲师都将自己的授课内容录制下来，压制成视频并且公开到互联网上供大家学习和下载。大部分视频是免费的，也有一部分高质量的收费视频。总的来说，现在你不管在哪儿都能足不出户的享受较高质量的远程教学资源（不像我当时那么苦逼，这方面资源非常稀缺）。

如上所示，我自己在抽空填充 ccietea.com 小站上的内容，希望这个小站能够陪伴大家从零基础走到 CCIE。当然，网上还有更优秀和经典的教学材料、视频，大家可以到各大 IT 论坛相应的板块去看看。尽量搜集到一整套视频。还是那句话，不要贪多。

3. 技术文档

书本+教学视频，配合上大量的实验，能将你的基础知识框架搭建的不错了，但是仍然有许多东西书本并未细致描绘，例如一些协议底层的工作机制、协议特性、产品特性等，这些就需要我们利用好互联网平台通过各方渠道去检索相关文档。掌握搜索引擎的用法是一个基本功，这个大家自己慢慢琢磨。

Cisco.com 思科官网上有着丰富的技术材料，在我自己钻研技术或授课过程中，经常在官网上检索文档做分析和讲解，应该说，这里的技术文档是海量的。

值得一提的是，许多优秀的文档材料是英文的，因此建议大家提高一下英文水品。如果没有英文阅读习惯，初次尝试阅读英文技术文档可能会感觉非常不适应，但是，相信我，一定要强迫自己坚持下来。关于如何提升自己的英文水平，我写过一个小文可以浏览一下：http://blog.sina.com.cn/s/blog_5ec35371010196nx.html

再次强调一下，信息搜集能力是一项非常重要的基本技能，步入职场，很多时候碰到问题都要靠自己解决，因此要掌握基本的寻找资源的方法，例如一个项目里可能涉及自己并未解除过的技术或产品，那么如果你能快速定位到有效的资源，这将是你的职场优势。

4. 技术笔记

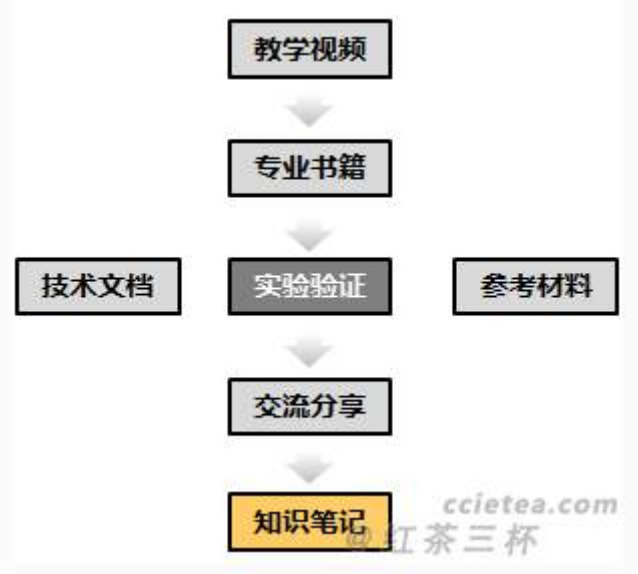
技术笔记是一种沉淀知识的很不错的习惯。一个枯燥的技术点不同的人可能会有不用的解析和分析的角度，以及方式。因此阅读技术笔记，无论是自己的，还是他人的，也是一种帮助理解和加深知识的方法。本人已将自己的部分技术笔记做了归纳和整理，分享在了 ccietea.com 上，有兴趣的朋友可以下载来看看。当然，还是建议大家养成自己的知识记录和整理习惯。

5. 行业圈子

互联网时代，圈子是一个非常重要的概念。无论是学生，还是职业人士，都应该注重行业圈子的寻找、建立和融入。如果你在自学网络技术，若能找到或建立个行业圈子，一来有对口专业的人能共同讨论、交流、提升；二来圈子能够贡献给内里的人多多少少关于行业的视野，让你不至于做个井底蛙；三来圈子里偶尔出现福利（如项目或就业机会）神马的，总而言之，人多好办事儿。YY 群、QQ 群、IT 论坛等等都可以多去看看去找找。

3 学习方法

从宏观层面上 ,我会建议初学者按照 CCNA -> CCNP -> CCIE 的路子来逐步搭建自己的专业知识体系 ,CCNA 到 CCNP 阶段建议仍然以标准的课程框架为基础进行学习 ,在 CCIE 阶段 ,开始以专题为单位进行攻克 ,例如 RIP、OSPF、EIGRP、ISIS、BGP、路由策略、二层交换、生成树、IPv6、Multicast、QoS、MPLS、MPLS VPN、feature 等等。逐个专题进行突破和攻克。下面我们来看一下微观层面：



举个例子，CCNA 课程的自学。我个人比较建议的学习方法如上图：

首先是找到一套 CCNA 教学视频（别问我怎么找的了，网上这方面的资源太多了，建议大家到国内的一些 IT 论坛上去看看，大家在看什么视频在推荐什么视频）。视频这种多媒体材料是学习知识的一种便捷、高效的方法，画面、声音、板书再搭配上讲师的讲解，可回放可暂停，一套高质量的教学视频可以大大的缩短你学习的周期甚至给你带来新的视野。

通过一整套 CCNA 教学视频的学习，你已经基本了解了 CCNA 知识的框架，在这个过程中，选定一本书籍，推荐《CCNA 学习指南》，然后开始研读书籍。有一点值得提醒的是，我经常看到不少同学，贪多喜大，CCNA 第一节课后围着我就开始拷文档教材，一下子几十 G 的材料这么拷走，但是真正去看的有几个？这几十个 G 最终只是在那么一瞬间贡献了一点点虚假的充实感而已。因此当学习一样东西的时候，集中精力攻克一本书，一套教材，这叫“主”，与此同时在关注某个技术细节的时候如果描写有限可以考虑参考其他书籍，这叫“辅”，有主有辅，狠抓主线打根基才是硬道理，千万别贪多，把自己搞乱搞晕了还整个毛线啊。

上面说到视频和书籍，在学习过程中还有一个非常重要的功课，就是实验。网络技术这门学科需要大量的实验来帮助我们消化知识、加深理解。每一个技术细节或者知识点务必通过实验进行验证。实验要做到一定境界，我经常开玩笑的说，做到自己想吐为止。

养成一个好的学习习惯，在学习知识、理解知识、消化知识的过程中，将知识用自己的习惯的、喜好的方式整理并记录下来。网络工程的领域是相当广的，路由、交换、安全、语音、无线、存储、DC 等等，要看的要学的实在太多了，很多东西你一段时间不看就容易遗忘，这时候如果有自己的一套知识笔记，相信回顾起来会非常的高效。这是一种个人知识管理技能。我个人非常建议大家用电子化的方式来记录笔记，因为那不容易丢失，而且便于检索和翻阅。我倾向使用 word 和 onenote 做笔记，用 word 记录笔记已经有近 10 年的历史了，这让我获益匪浅，我的文档功底得到了极大的历练，同时，也逐渐形成了一套自己的知识沉淀风格。关于知识笔记的建议我另寻他文来做分享，本文暂不详释。

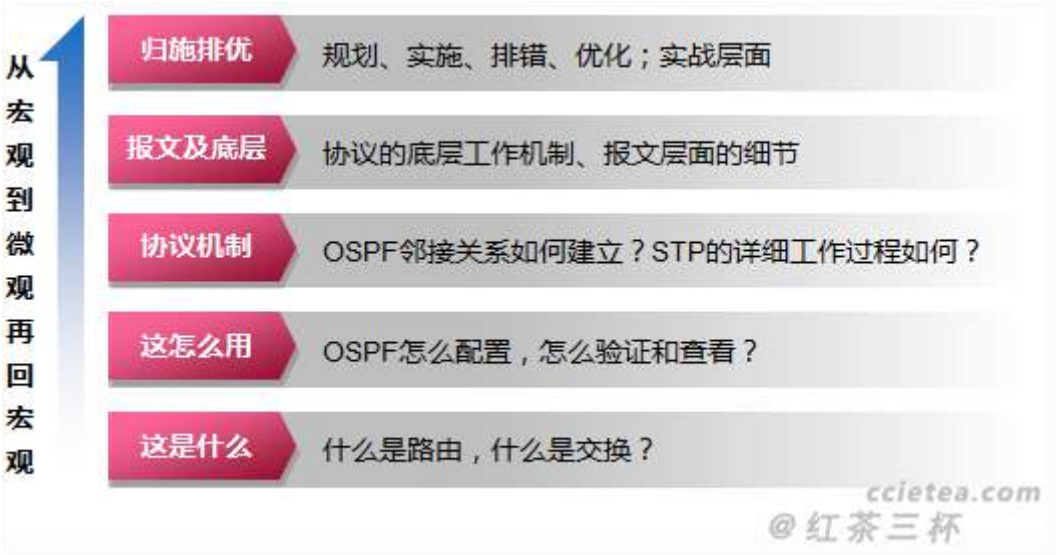
因此我们的主线是先看视频，搭配书本进行知识细节的学习，然后进行大量实验，通过实验加深理解，接着将自己的理解及实验结果记录成笔记。同时在学习过程中，有疑问，则可上网寻找相关技术文档或参考资料进一

步学习，或者通过 IT 论坛（如 bbs.spoto.net）、QQ 群等圈子提问或交流。如果有时间有精力，开个技术博客，将自己的学习成果发布到博客上。现在估计也没多少人有耐心写博客了，不过坚持写博客，在多年后你会发现它的价值，这里不再赘述。

这里再强调一点，在我教过的学生中，有部分同学是在校生，经常会发现一种现象。暑期过来集训，学了 2 个月，基础打的还算不错，转眼就开学回校了，到了寒假回来忘的干干净净全尼玛还给我了。学习一定要有强烈的主观能动性，要有强烈的紧迫感。适当的给自己施加点压力这会让你未来少点压力。再者，作为自学的童鞋，要懂得给自己定学习计划，例如两星期自学完 CCNA，半年自学完 CCNP，接下去以专题的形式，一个月过一个专题。虽然我不愿承认但是还是不的不说，能真正将学习计划执行下来的估计不超过两成，然而如若你真的能扛下来，我敢说你绝对菜不了。学习要紧凑，尤其是网络技术专题的学习，可以保持长时间的研究和钻研，但是在初学的这段时间，我个人还是建议把学习时间弄紧凑，而不要拉得太长。

4 成长里程碑

网络工程行业初学者成长之路，我喜欢用一个模型来描述，我将其整理如下：



入门阶段，例如在 CCNA 的学习过程中，我们要了解“这玩意儿是什么”，例如什么是路由，什么是交换，他们的概念是什么，这要理解。

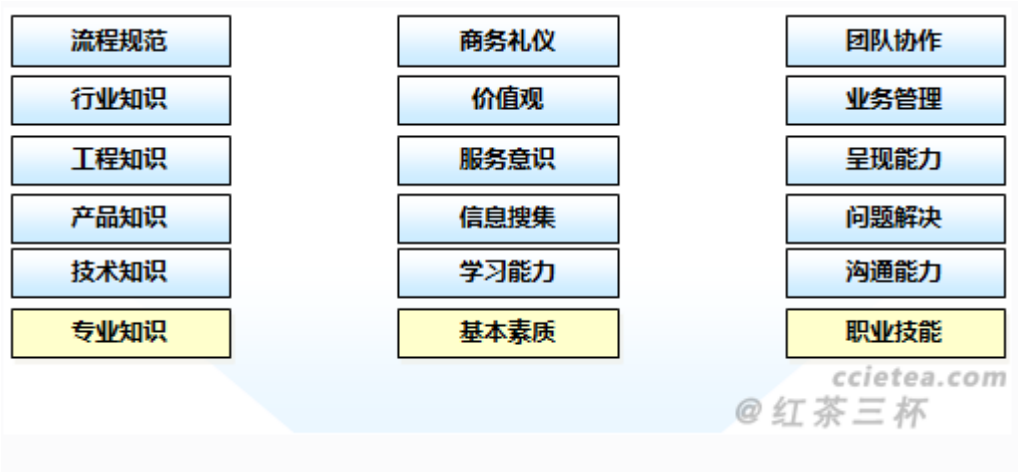
紧接着就是要掌握这东西怎么用，或者说，怎么配置怎么查看和验证，例如 OSPF 你掌握了它的基本概念以及大概了解了它的工作原理，那么怎么配置呢？

上面两个里程碑基本上是让你对特定技术和协议有个宏观上的认识，你已经能配置或者使用这些技术了。接下去就要慢慢的渗透到微观层面。你要掌握这些技术和协议的底层工作机制或原理，否则你的技术永远是半桶水。例如 OSPF ,它的邻居关系建立过程是如何进行的 ,可能会遇到什么问题？OSPF 的几种常见 LSA 的理解？HSRP 协议的工作机制如何等等。

了解了工作机制，是否就可以成为技术牛人呢？还未必，要成为技术牛人，你还需要关注协议的报文。在此之前你可能已经知道，网络协议的工作需要依赖于网络设备之间交互特定的协议报文，这些报文中包含的字段内容将支撑协议的工作，例如 OSPF 协议，在 OSPF 网络中，所有的 OSPF 路由器都将交互 OSPF 报文，那么这些报文都是什么？有着什么样的格式？关键字段里包含的内容有什么意义呢？这些将是我们在这个里程碑需要去掌握的。久而久之我们希望大家养成一个习惯，学习任何一个网络技术或者一个网络协议，采用“从宏观到微观，从抽象到具象”的思维来进行，正如上文所述。因此，在微观层面，你将和报文打交道。理解协议报文的交互及关键字段对协议的贡献，这将把你的技术及视野推升到一个新的高度。

在经过上面几个里程碑后，你已经成长为一个技术牛人了，然而这是不够的，毕竟我们学技术不是为了装逼不是为了做实验或泡妹纸啥的，而是为了应用为了实践的，因此宏观到微观之后，又回宏观 -- 现在你要掌握的是实战层面或者工业层面的内容：规划、实施、排错、优化……，也就是一个技术在实际的业务环境中如何被运用，可能会出现什么问题等等，这些玩意儿可就不是纸上谈兵了，需要大量的项目来历练和积累。

事实上宏观层面的回归还有些后续的内容没有体现在图表上，那是因为我相信，完成上述几个里程碑的飞跃，大家都应该知道下面的路子该如何走了，正如我前面所说，网络工程师是一个渴望自我提升自我成长和自我完善的敏捷性群体，每一个工程师的职业成长道路都可以理解为每个人的综合层面修炼过程。



总结一下几点建议：

- 主线明确，学习有重点
- 善用多媒体资源
- 紧凑型、有计划的学习
- 注意积累知识，模式适合自己的知识管理方法
- 注意交流，找到圈子
- 坚持，坚持，坚持