Embracing Concurrency

for Fun, Utility & Simpler Code

Michael Sparks
BBC R&D, http://www.kamaelia.org/Home

Ignite Leeds, Jan 2009

Embracing Concurrency

for Fun, Utility & Simpler Code

Or "what we've learnt as a part of the Kamaelia project about making concurrency something that's fun and useful, and usable by novice and advanced developers alike... ..rather than a pain in the neck"

Michael Sparks
BBC R&D, http://www.kamaelia.org/Home

Ignite Leeds, Jan 2009

Why?

Opportunity!

Hardware finally going massively concurrent ...

.... PS3, high end servers, trickling down to desktops, laptops)

“many hands make light work” but **Viewed** as Hard

... do we just have crap tools?

Problems

“And **one** language to in the darkness bind them”

... **can** just we **REALLY** abandon 50 years of code for Erlang, Haskell and occam?

We're Taught Wrong

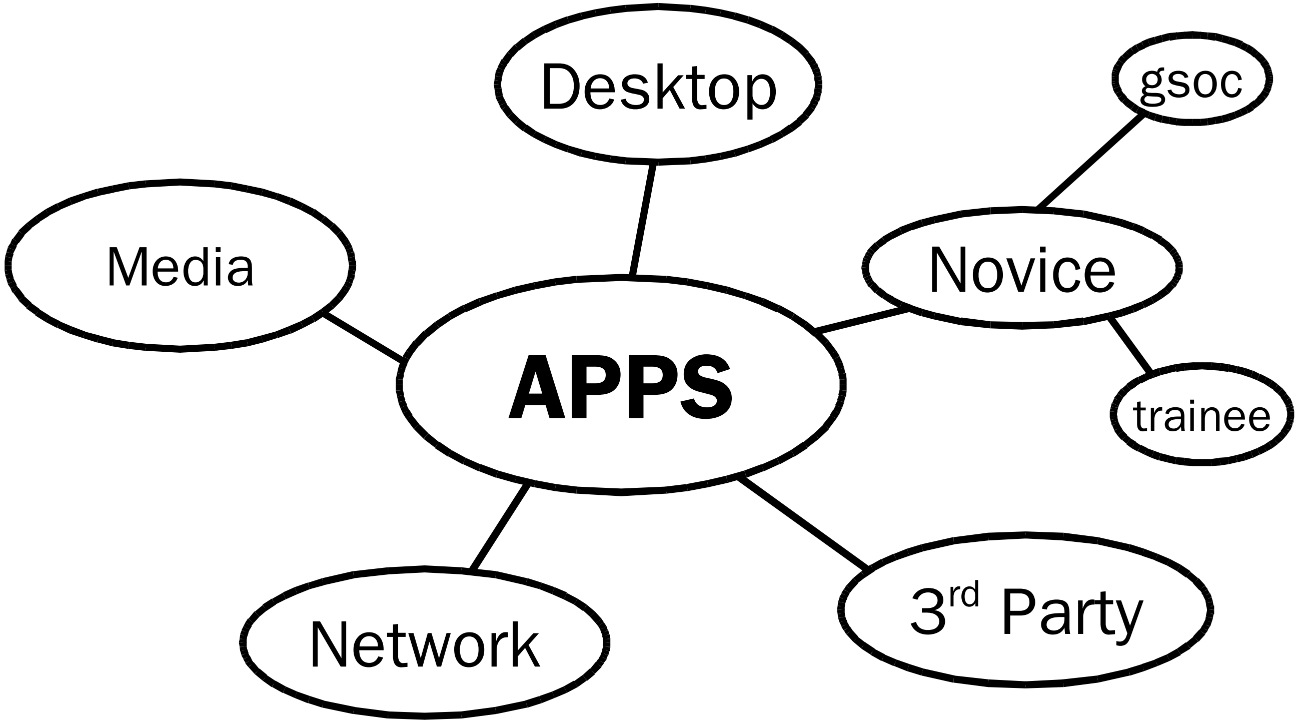
Fundamental Control Structures

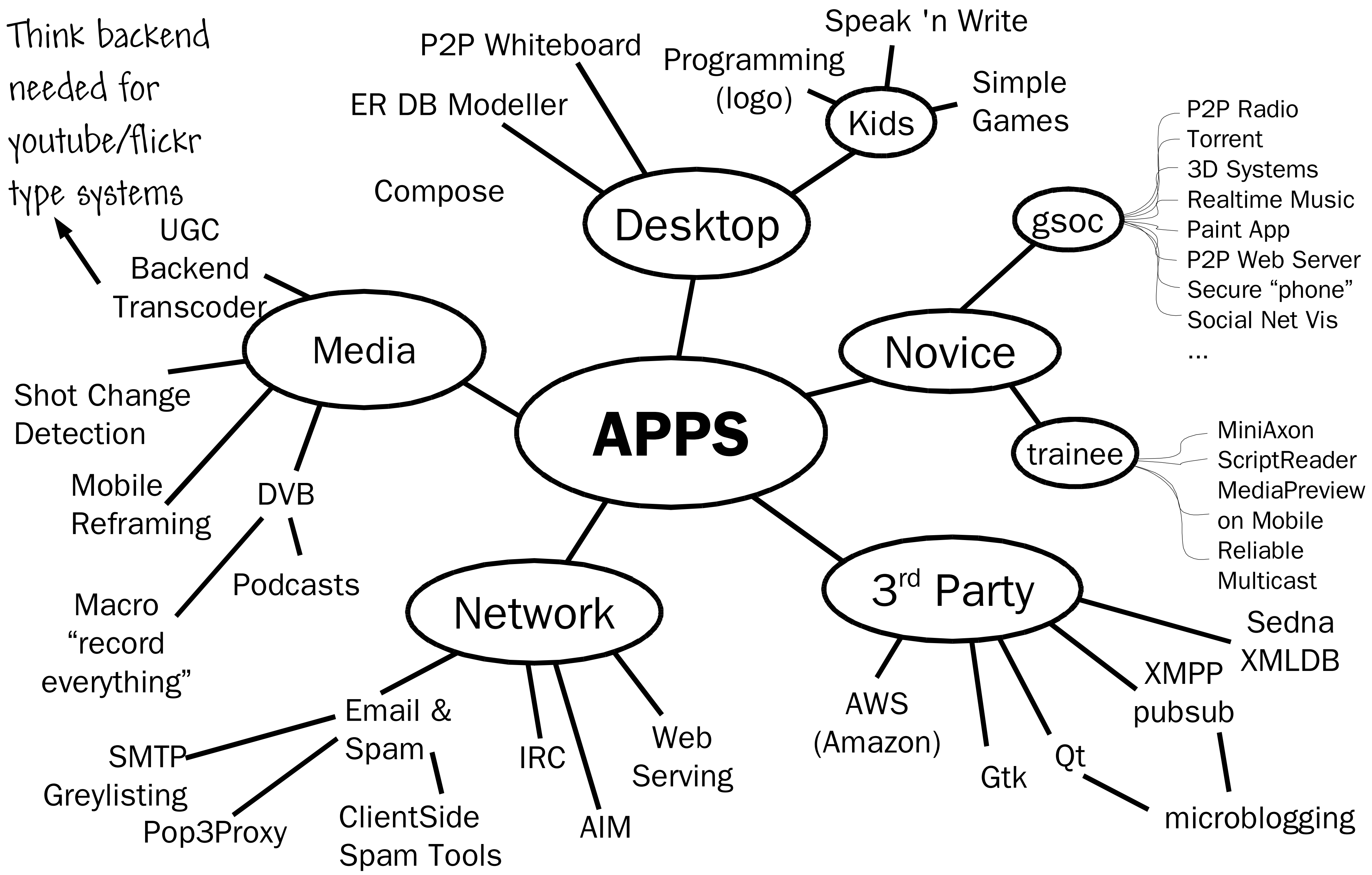
... in imperative languages **number greater than 3!**

Control Structure	Traditional Abstraction	Biggest Pain Points
Sequence	Function	Global Var
Selection	Function	Global Var
Iteration	Function	Global Var
Parallel	Thread	Shared Data

Usually Skipped

Lost or duplicate update are most common bugs


APPS
Desktop
Media
Novice
gsoc
trainee
Network
3rd Party



APPS

Desktop
- P2P Whiteboard
- ER DB Modeller
- Compose
- Kids
  - Speak 'n Write
  - Programming (logo)
  - Simple Games

Media
- UGC Backend Transcoder
- Shot Change Detection
- Mobile Reframing
- DVB
  - Macro "record everything"
  - Podcasts

Think backend needed for youtube/flickr type systems

Novice
- gsoc
  - P2P Radio
  - Torrent
  - 3D Systems
  - Realtime Music
  - Paint App
  - P2P Web Server
  - Secure "phone"
  - Social Net Vis
  - ...
- trainee
  - MiniAxon
  - ScriptReader
  - MediaPreview on Mobile
  - Reliable Multicast

Network
- Email & Spam
  - SMTP Greylisting
  - Pop3Proxy
  - ClientSide Spam Tools
- IRC
- AIM
- Web Serving

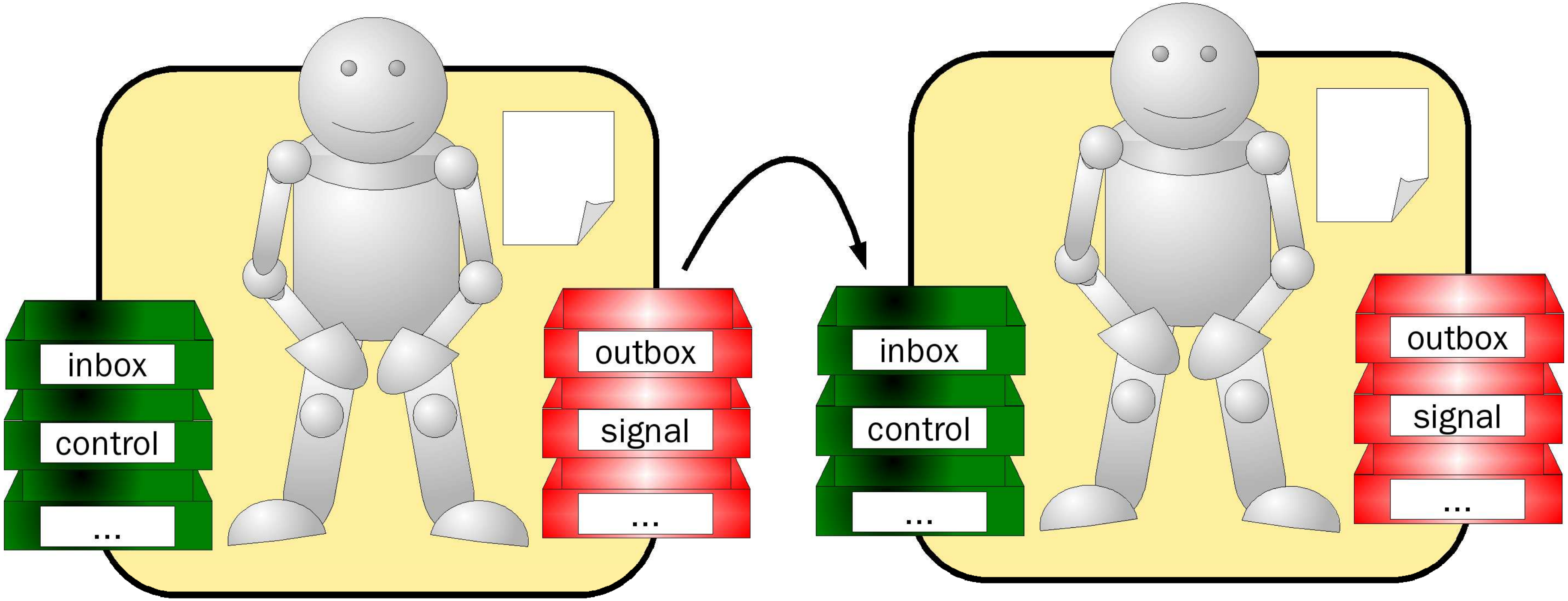
3rd Party
- AWS (Amazon)
- Gtk
- Qt
- XMPP pubsub
  - microblogging
- Sedna XMLDB

Core Approach:

Concurrent things with comms points

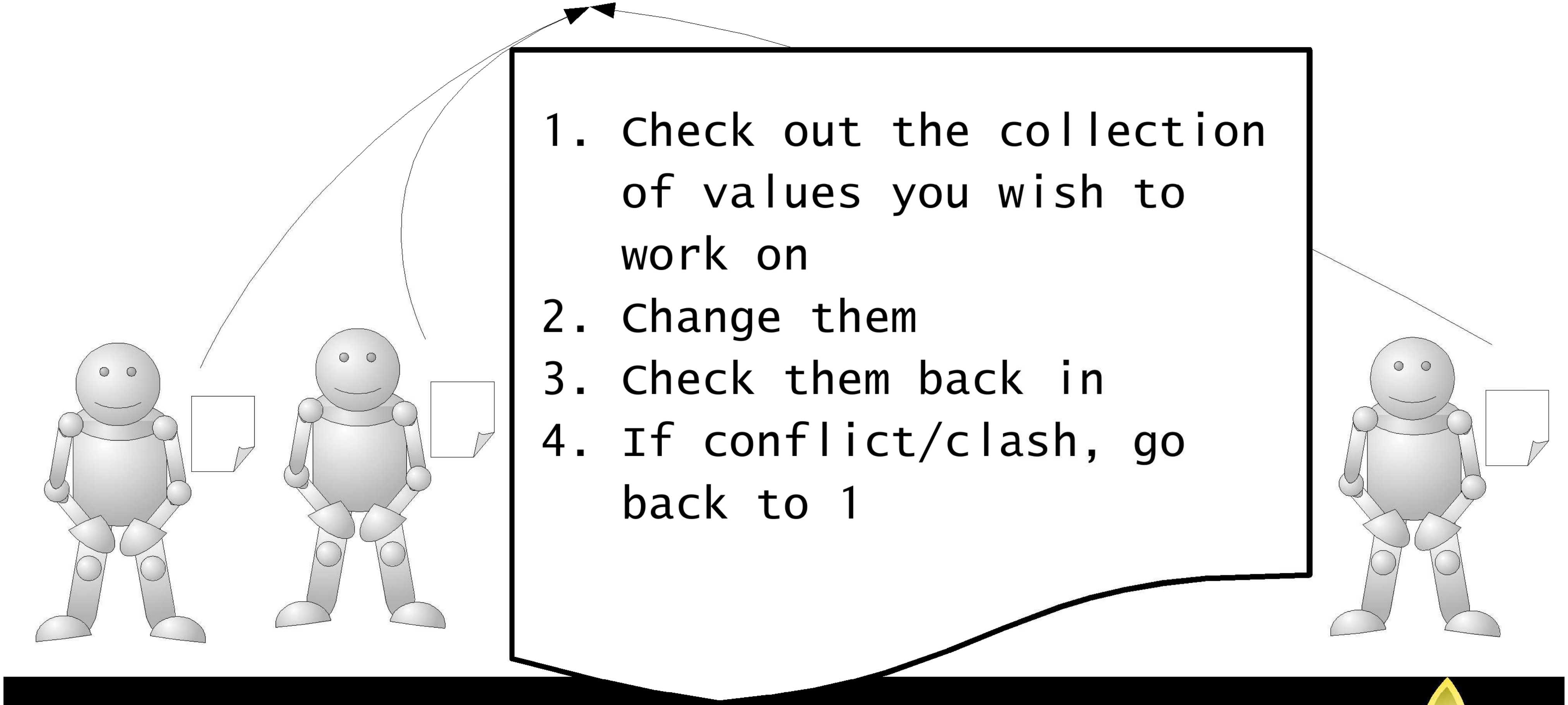
Generally send messages

Keep data private, don't share



But I must share data?

Use Software Transactional Memory
ie version control for variables.



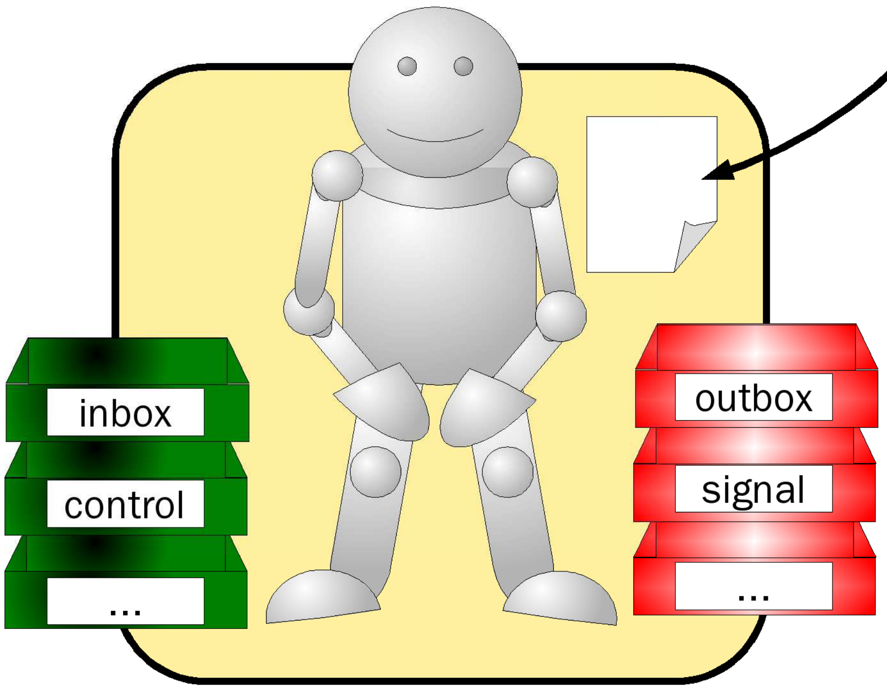
1. Check out the collection of values you wish to work on
2. Change them
3. Check them back in
4. If conflict/clash, go back to 1

Perspectives in APIs! (1/2)

1st, 2nd, 3rd Person

1st Person - **I** change my state

2nd Person – **YOU** want to me to do something (**you** send **me** a message)



3rd Person – **Bla** should do something (**I** send a message)

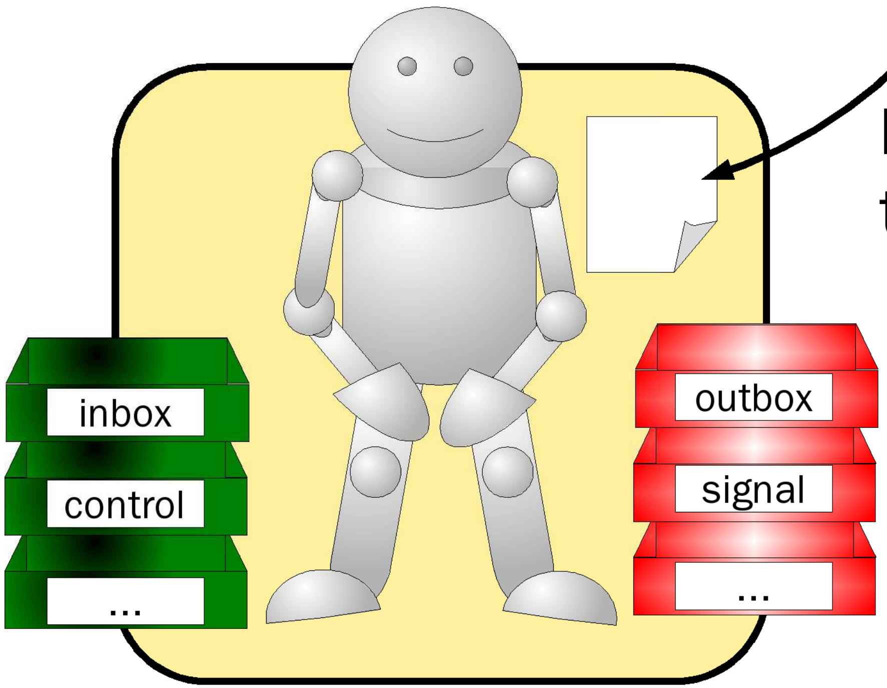
Michael Sparks
BBC R&D, http://www.kamaelia.org/Home

Perspectives in APIs! (2/2)

1st, 2nd, 3rd Person

private real methods

Messages from public inboxes



Messages sent to public outboxes

Also, think about stdin

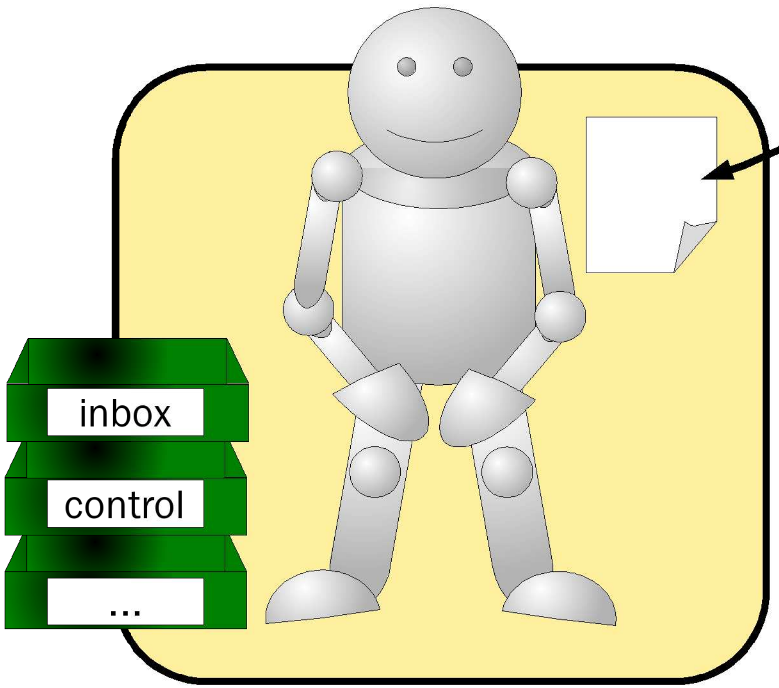
Also, think about stdout

Actor Systems

Distinction **can** be unclear, potential source of ambiguity*

private real methods

Messages from public inboxes



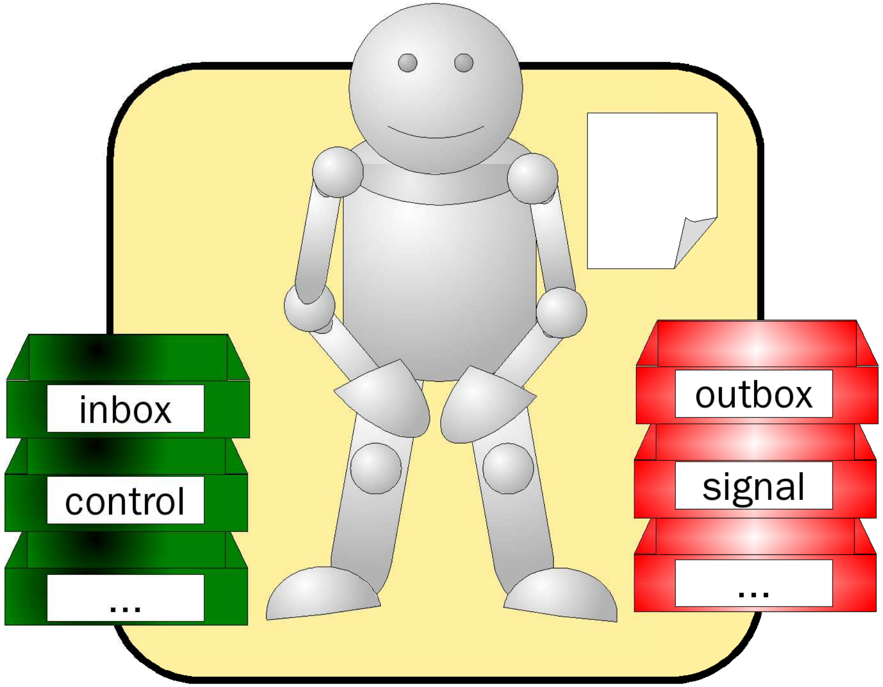
No outbox concept

Possible issues with rate limiting*

Hardcodes recipient in the sender

*system dependent issue

Advantages of outboxes



No hardcoding of recipient allows:

- Late Binding
- Dynamic rewiring

Concurrency Patterns as Reusable Code

... a concurrency DSL

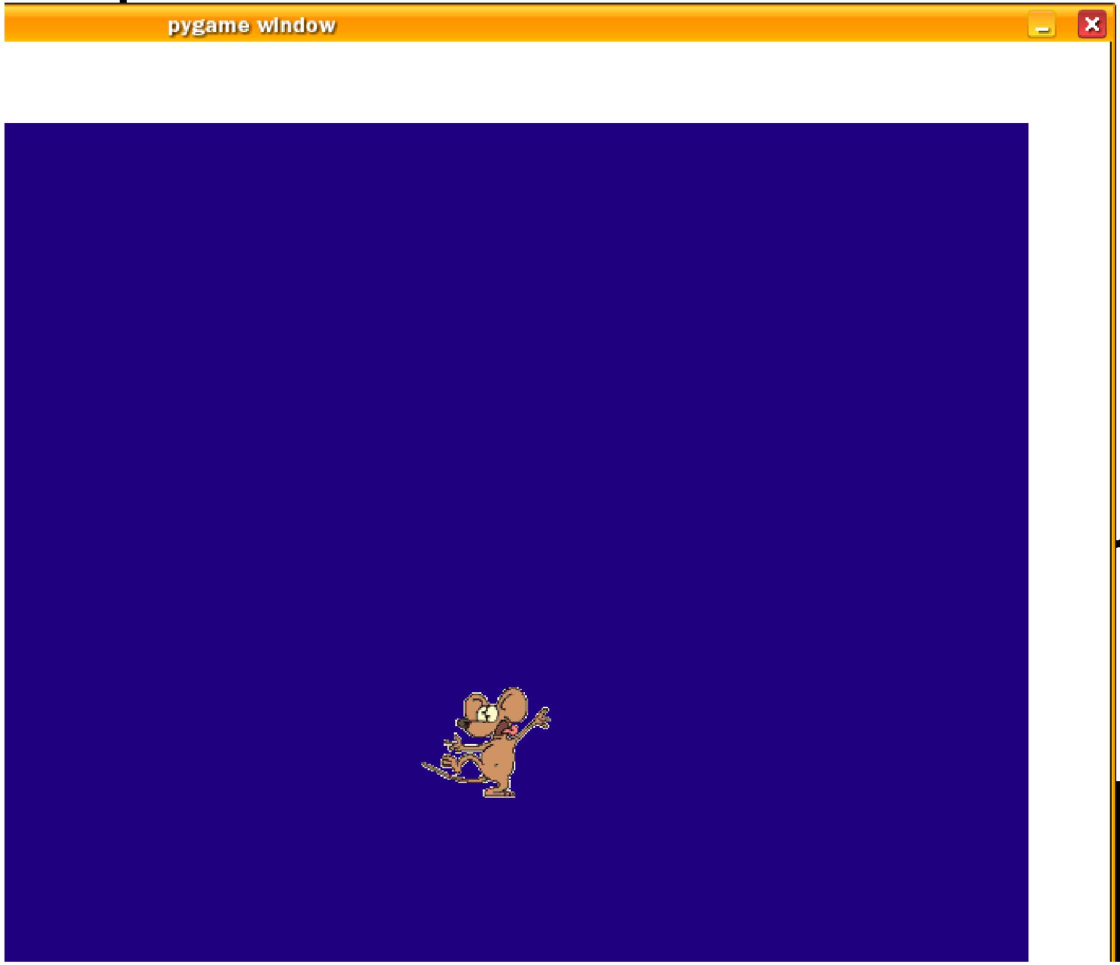
A Core Concurrency DSL

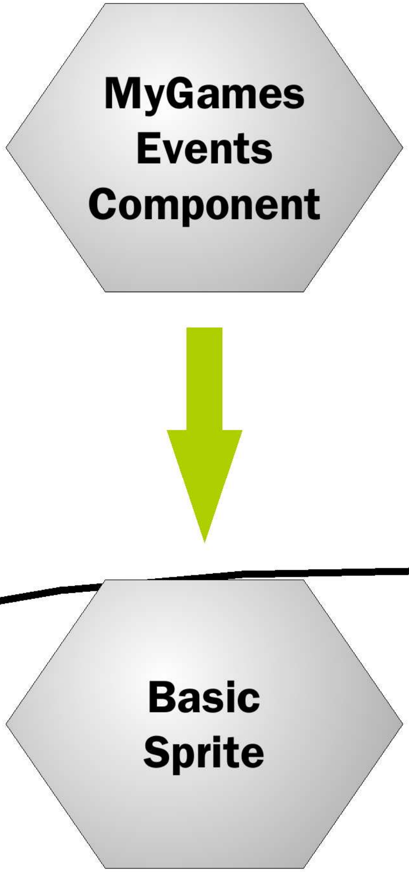
```
Pipeline(A,B,C)
Graphline(A=A,B=B, C=C, linkages = {})
Tpipe(cond, C)
Seq(A,B,C), PAR(), ALT()
Backplane("name"), PublishTo("name"), SubscribeTo("name")
Carousel(...)
PureTransformer(...)
StatefulTransformer(...)
PureServer(...)
MessageDemuxer(...)
Source(*messages)
NullSink
```

Some of these are *work in progress* – they've been identified as useful, but not implemented as chassis, yet

Pipeline Example

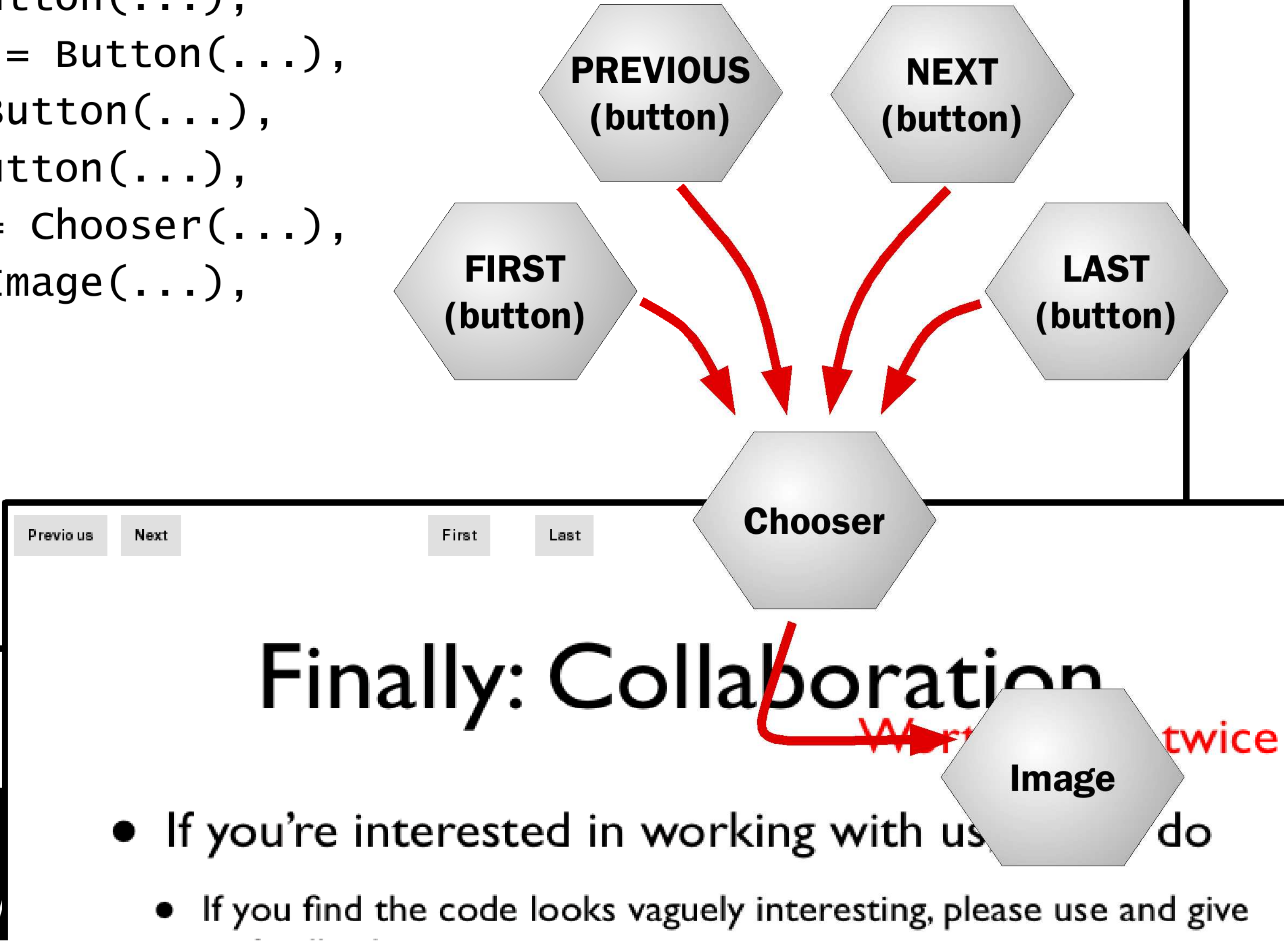
```
Pipeline(
    MyGamesEventsComponent(up="p", down="l", left="a", right="s"),
    BasicSprite("cat.png", name = "cat", border=40),
).activate()
```





Graphline Example

```
Graphline(
    NEXT = Button(...),
    PREVIOUS = Button(...),
    FIRST = Button(...),
    LAST = Button(...),
    CHOOSER = Chooser(...),
    IMAGE = Image(...),
    ...
).run()
```



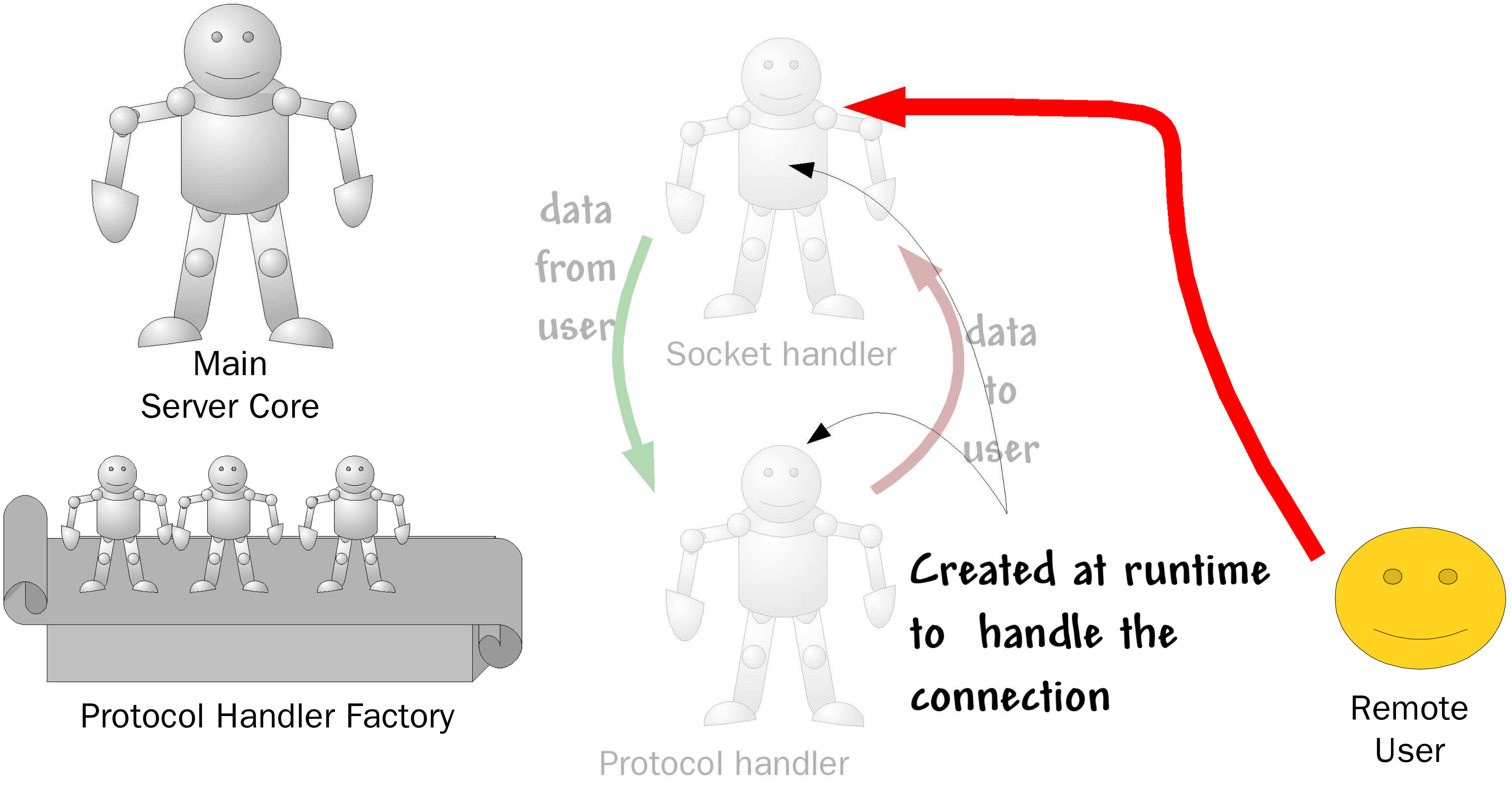
Finally: Collaborati

W rt twice

- If you're interested in working with us do
  - If you find the code looks vaguely interesting, please use and give

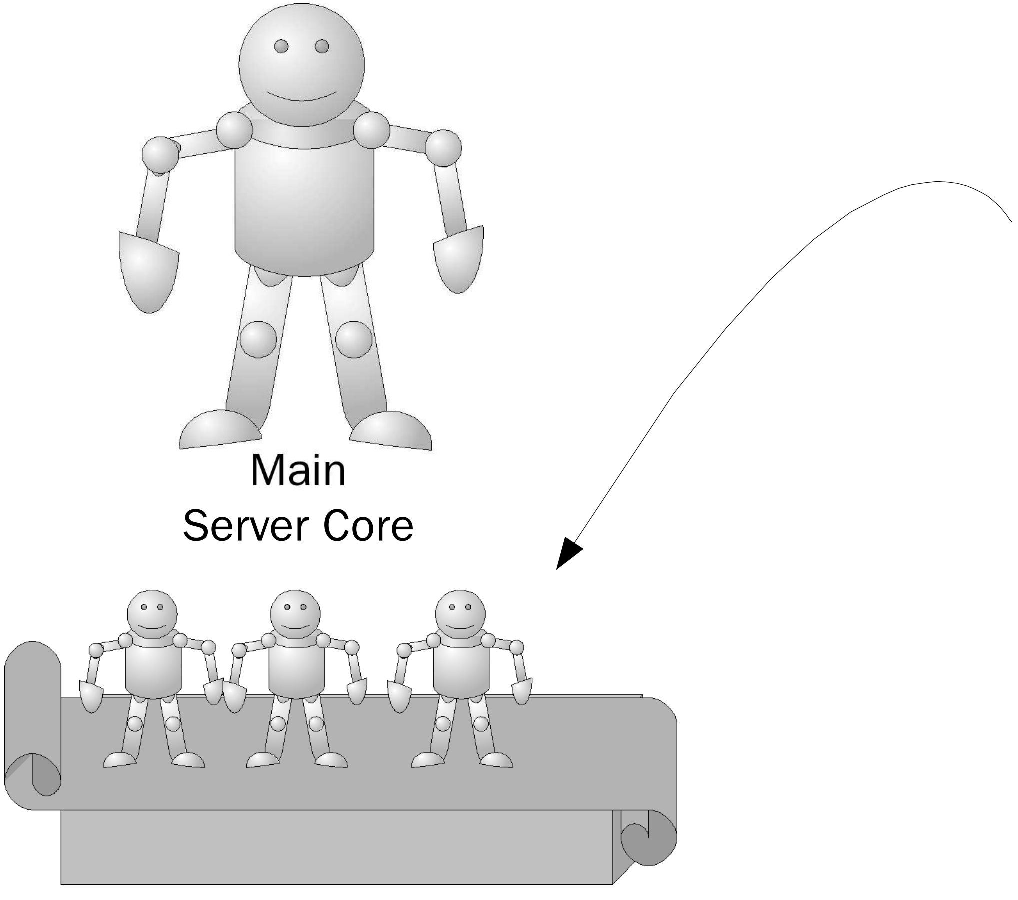
Michael Sparks
BBC R&D, http://wv

Server Example



Created at runtime to handle the connection

Server Example



You therefore need to provide this bit.

Server Example

```
from Kamaelia.Chassis.ConnectedServer import ServerCore
from Kamaelia.Util.PureTransformer import PureTransformer

def greeter(*argv, **argd):
    return PureTransformer(lambda x: "hello" +x)

class GreeterServer(ServerCore):
    protocol=greeter
    port=1601

GreeterServer().run()
```

Backplane Example

```
# Streaming Server for raw DVB of Radio 1
Backplane(“Radio”).activate()

Pipeline(
   DVB_Multiplex(850.16, [6210], feparams), # RADIO ONE
   PublishTo("RADIO"),
).activate()

def radio(*argv,**argd):
     return SubscribeTo(“RADIO”)

ServerCore(protocol=radio, port=1600).run()
```

Thank you for listening!

If you have questions, grab me later :-)

Michael Sparks
BBC R&D, http://www.kamaelia.org/Home