Not your father's Internet.

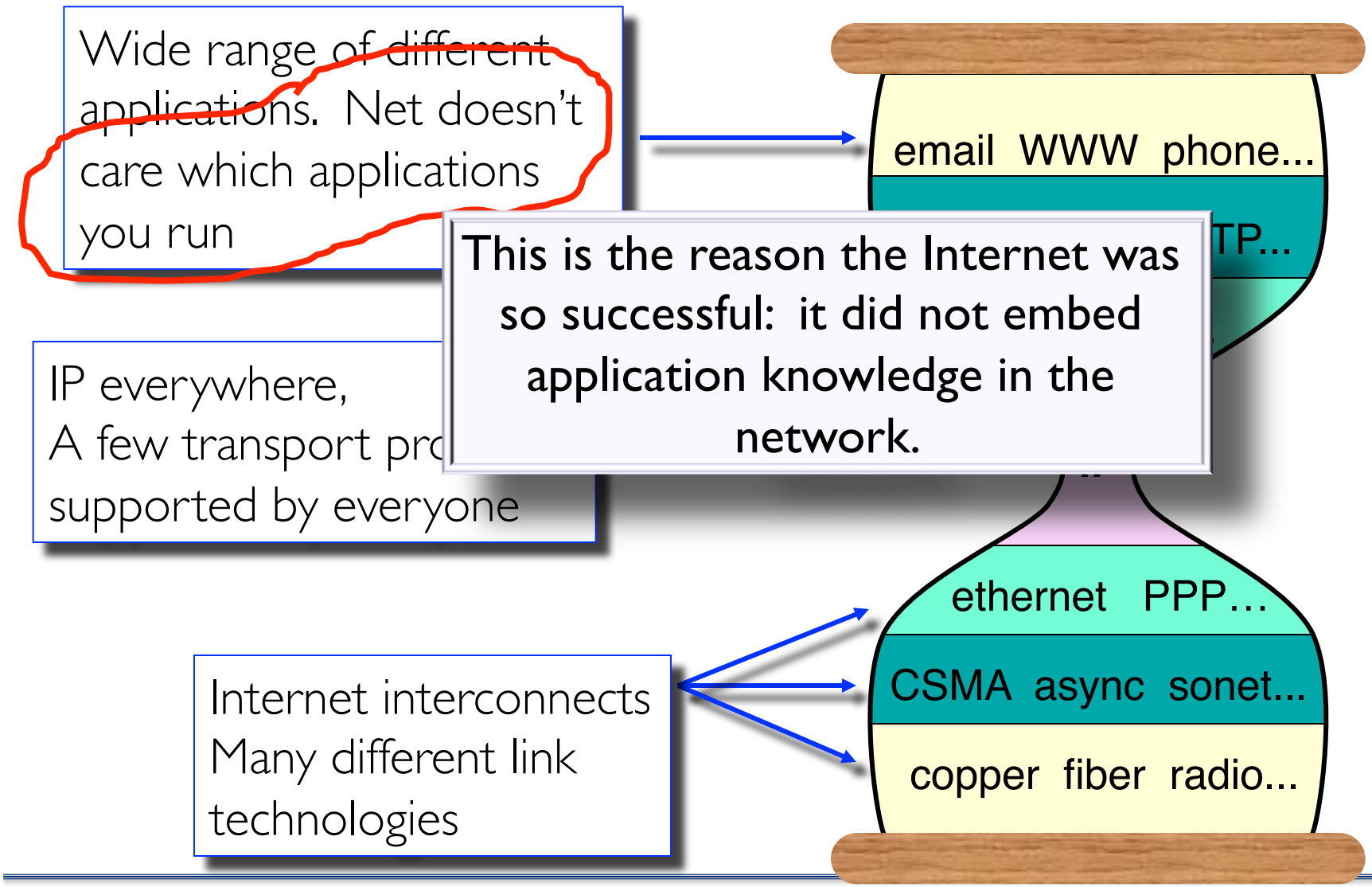
Or, where did the Internet architecture go?

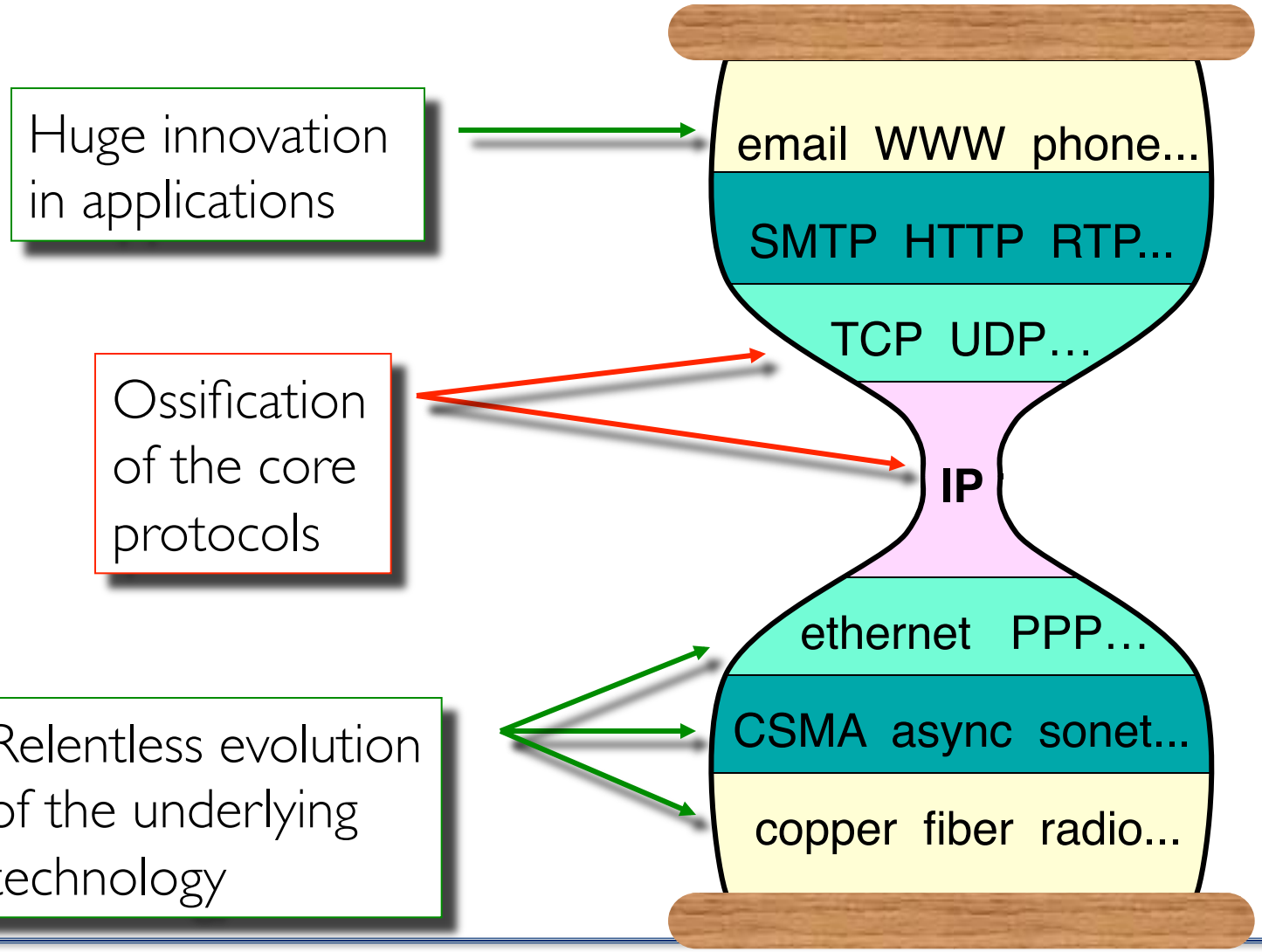
[And what can we do about it]

Mark Handley, UCL

Part I

Today's Internet


Wide range of different applications. Net doesn't care which applications you run
email WWW phone...
TP...
This is the reason the Internet was so successful: it did not embed application knowledge in the network.
IP everywhere,
A few transport pro
supported by everyone
ethernet PPP…
CSMA async sonet...
copper fiber radio...
Internet interconnects
Many different link technologies


Huge innovation in applications
Ossification of the core protocols
Relentless evolution of the underlying technology
email WWW phone...
SMTP HTTP RTP...
TCP UDP…
IP
ethernet PPP…
CSMA async sonet...
copper fiber radio...

Ossification of the Core Protocols?

- IP has a defined extension mechanism:
  - **IP Options**.

- Problem:
  - Fast router implementations are often **hardware**.
  - IP options get punted to the control processor for **software** processing -> slow!
- Result:
  - No-one uses IP options anymore.

Ossification of the Core Protocols?

- TCP and IP were standardized in 1981.
  - How many other transport protocols are commonly deployed today?

NONE

Ossification of the Core Protocols?

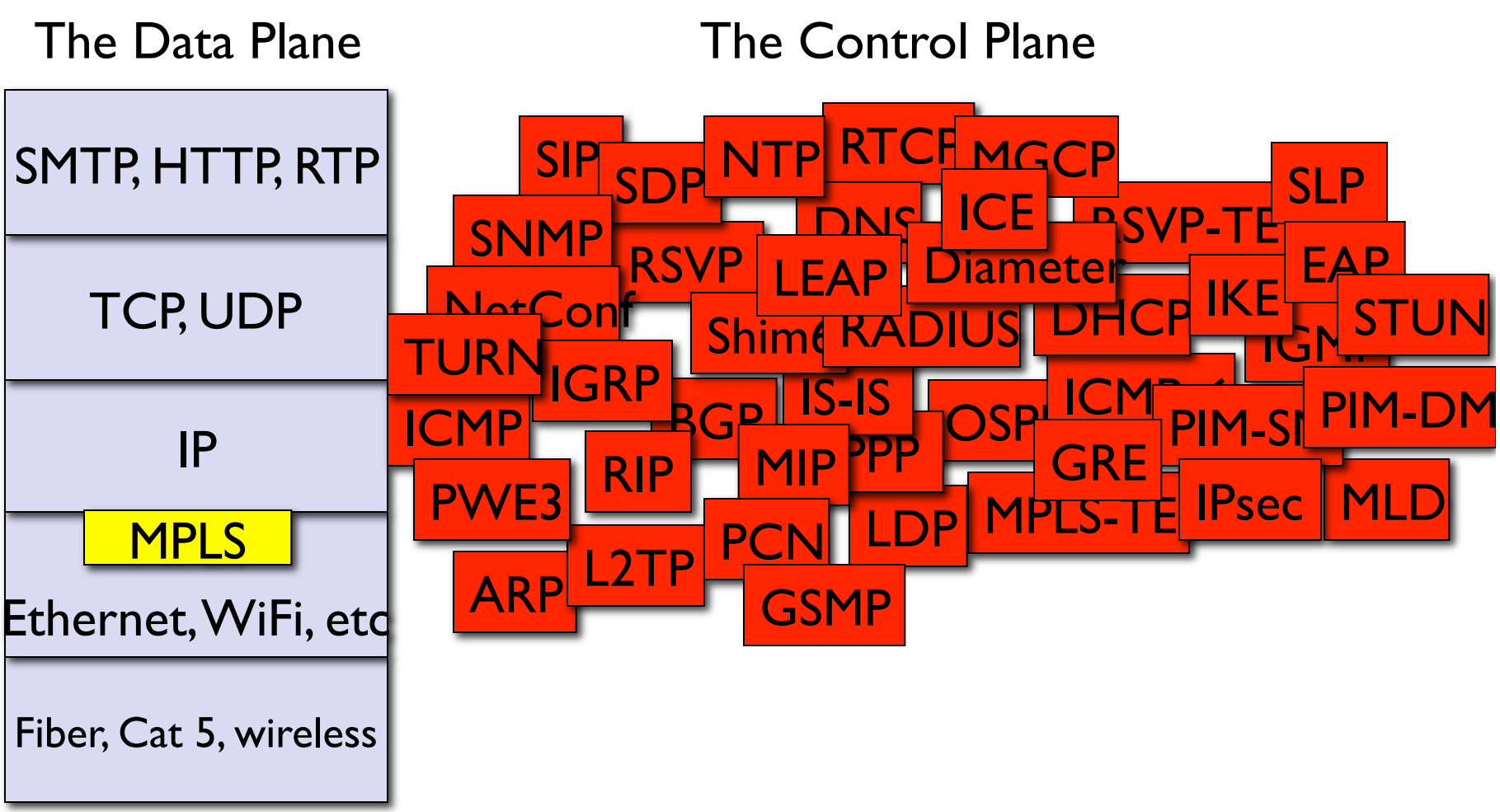
- TCP and IP were standardized in 1981.
  - How many other transport protocols are standardized and still actively worked on?

SCTP: Signalling Control Transport Protocol

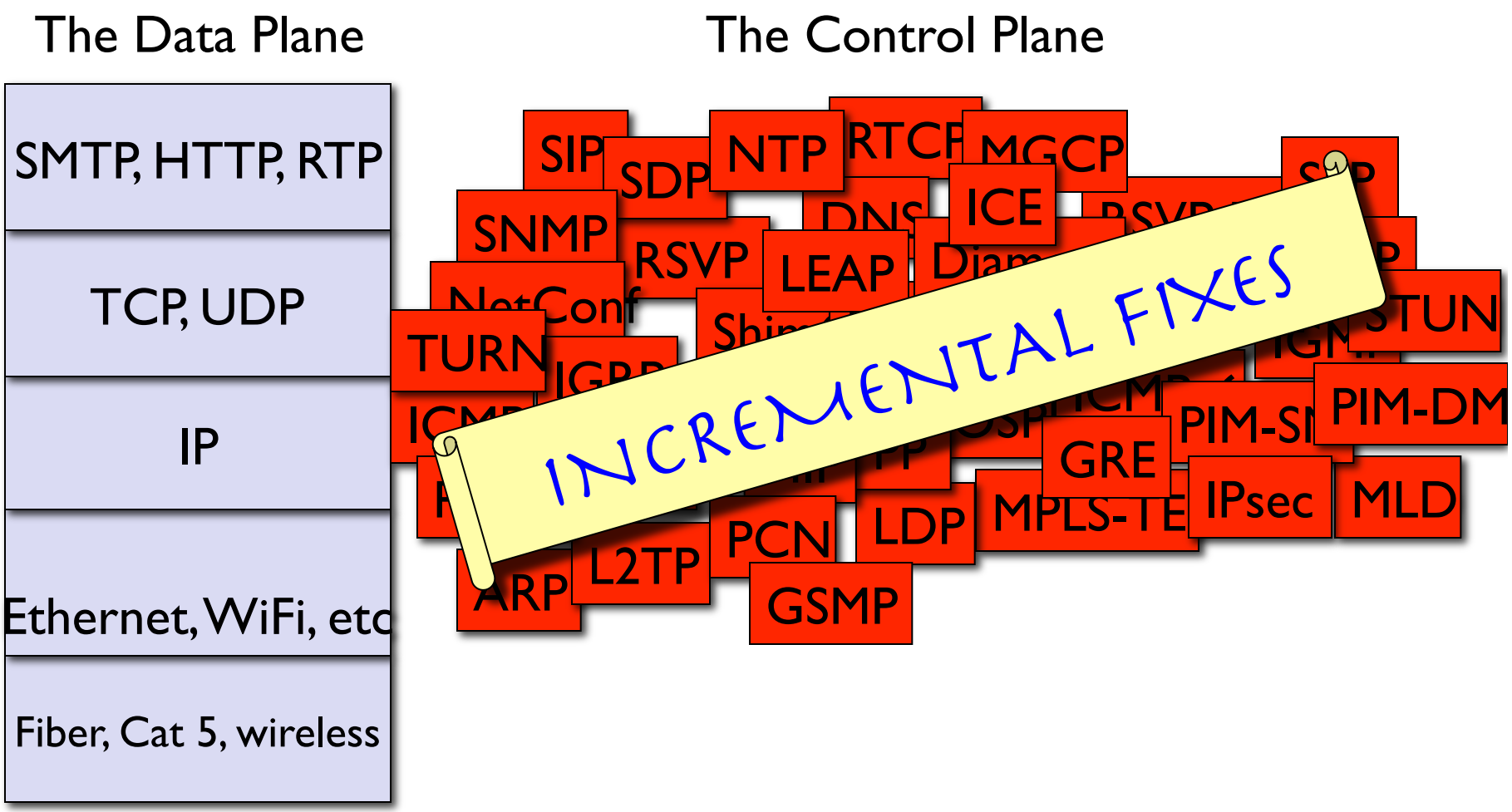
DCCP: Datagram Congestion Control Protocol

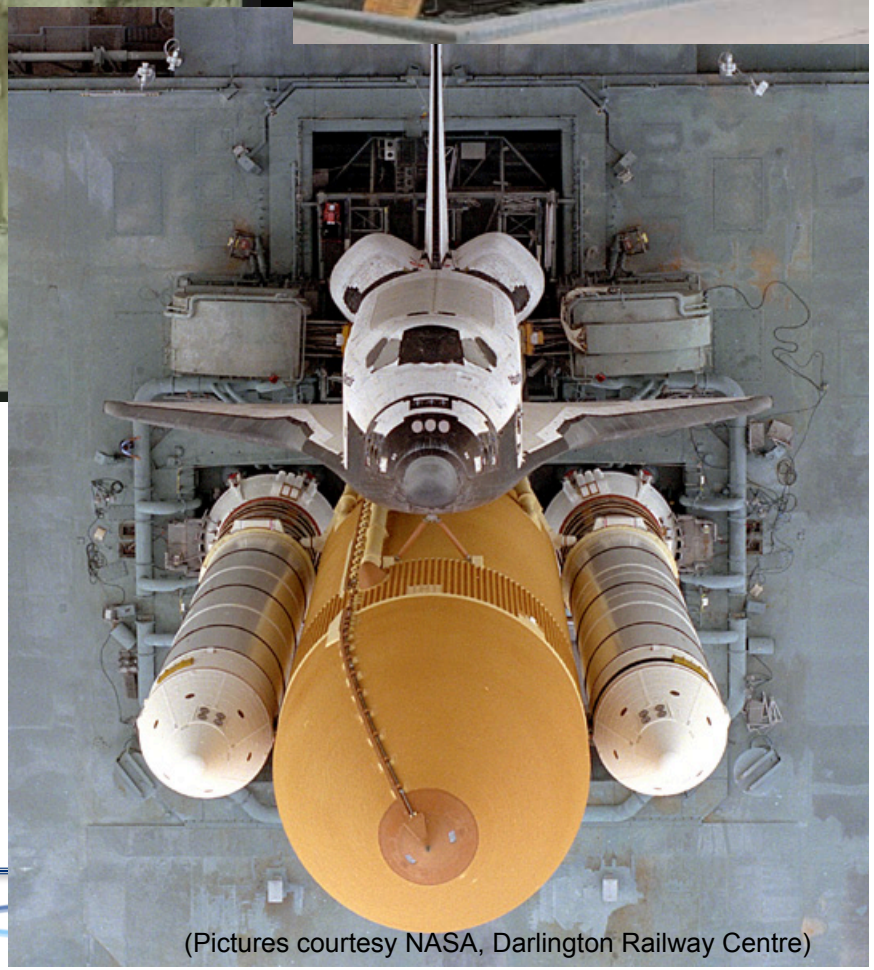
- *No apps because not commonly available.*
- *Not commonly available because no apps.*
- *Won't work end-to-end anyway – no firewall support.*

The Internet Stack



The Internet Stack





The Power of Legacy

(Pictures courtesy NASA, Darlington Railway Centre)

CHANGE

Brunel's 7-foot gauge.

More stable, faster, more spacious.

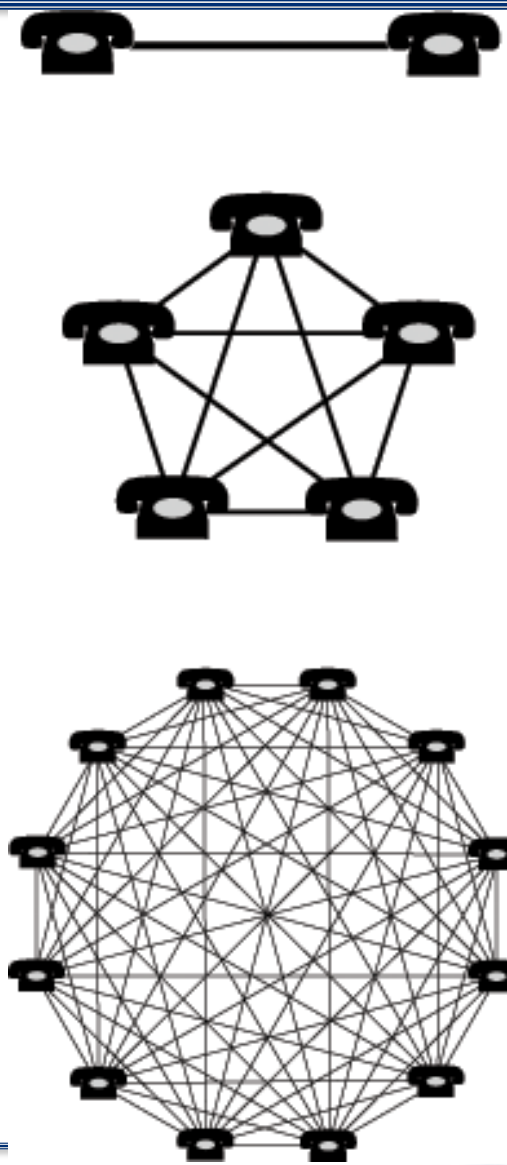
But, better technology often fails to win.

Network effect:

- Unloading cargo to transfer between railways was too expensive.

Metcalfe's Law:

- The utility of a telecommunications network grows with the square of the number of users.



Picture by Derrick Coetzee

The Internet Stack

The Data Plane

SMTP, HTTP, RTP

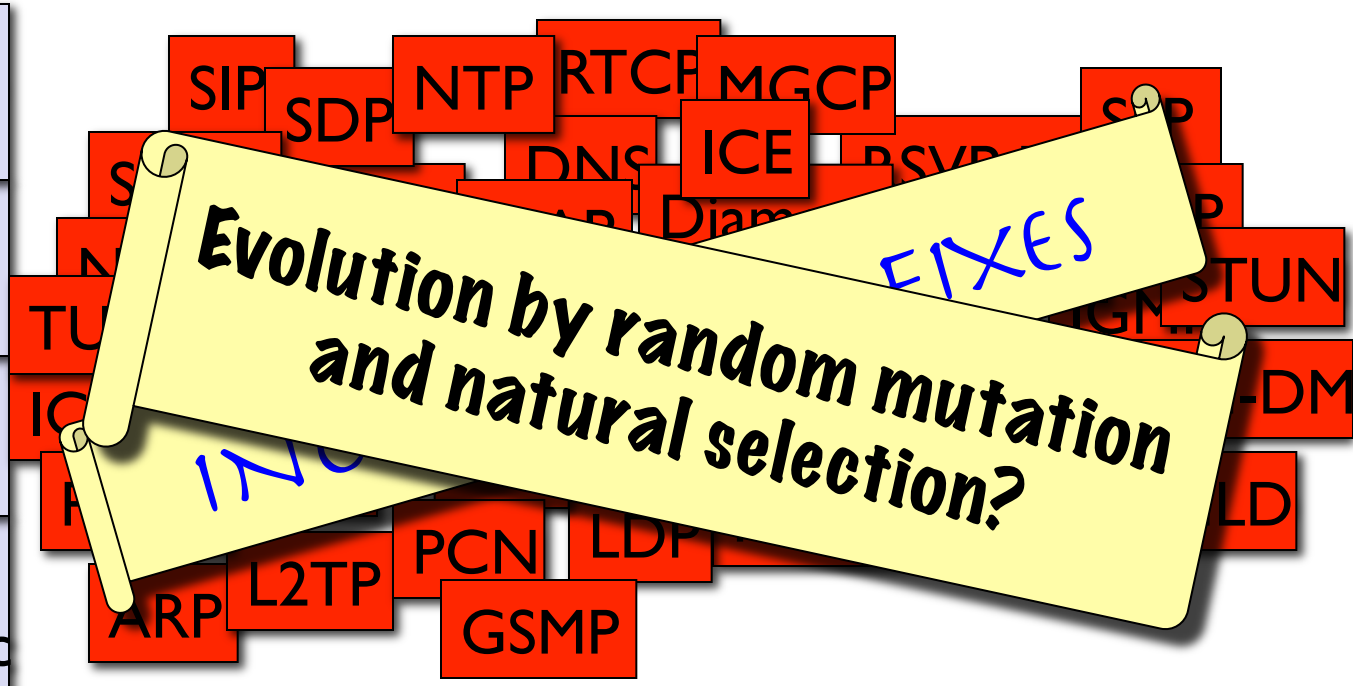
TCP, UDP

IP

Ethernet, WiFi, etc

Fiber, Cat 5, wireless

The Control Plane



Architecture Lesson #1:

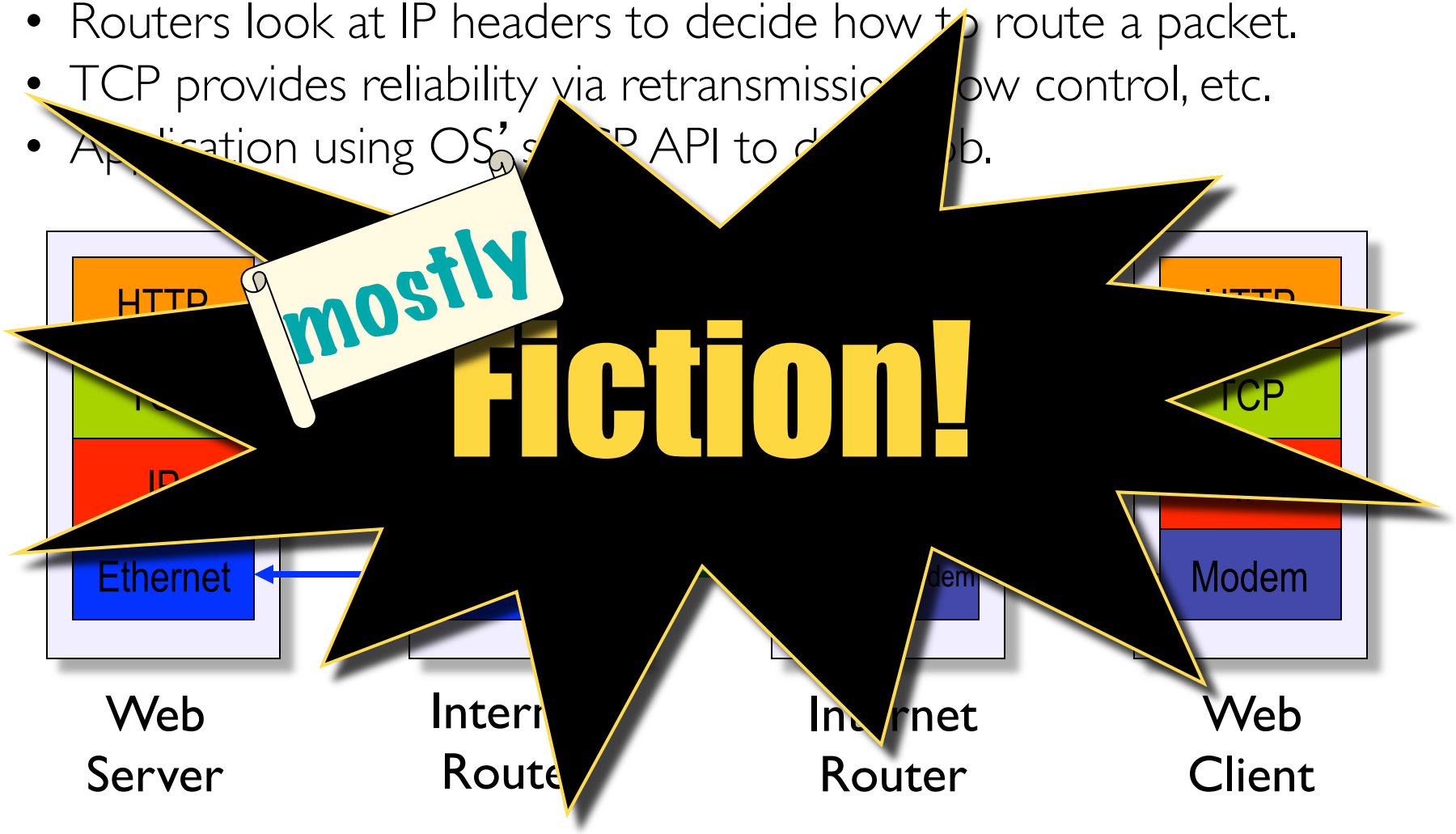
If you need to change the network, but can't change IP itself, you can always add another control protocol to modify IP's behaviour.

Architecture Lesson #1:

If you need to change the network, but can't change IP itself, you can always

Protocol Layering

- Link layers (eg Ethernet) are local to a particular link
- Routers look at IP headers to decide how to route a packet.
- TCP provides reliability via retransmissio[illegible]ow control, etc.
- Ap[illegible]ation using OS's [illegible]P API to d[illegible]b.



The usual suspects

- NATs are ubiquitous
  - We've become pretty good at working around them.

- Firewalls are ubiquitous
  - Ability to communicate using one port does not imply that communication is possible on any other port.

An Aside: **Multipath transport**

WiFi

3G

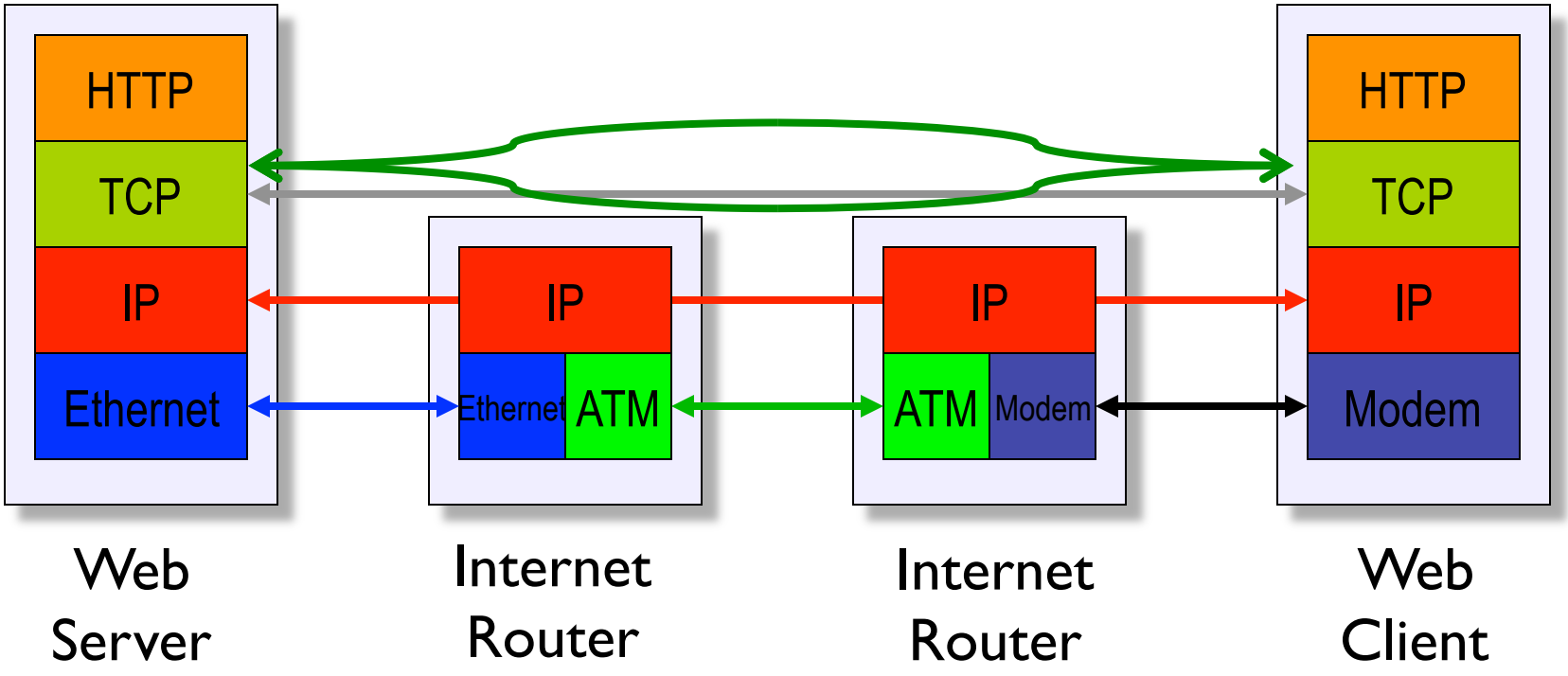
Bluetooth

Why does my phone have to choose between connections?

What if it could use several connections at the same time?

Protocol Layering

- Link layers (eg Ethernet) are local to a particular link
- Routers look at IP headers to decide how to route a packet.
- TCP provides reliability via retransmission, flow control, etc.
- Application using OS's TCP API to do its job.

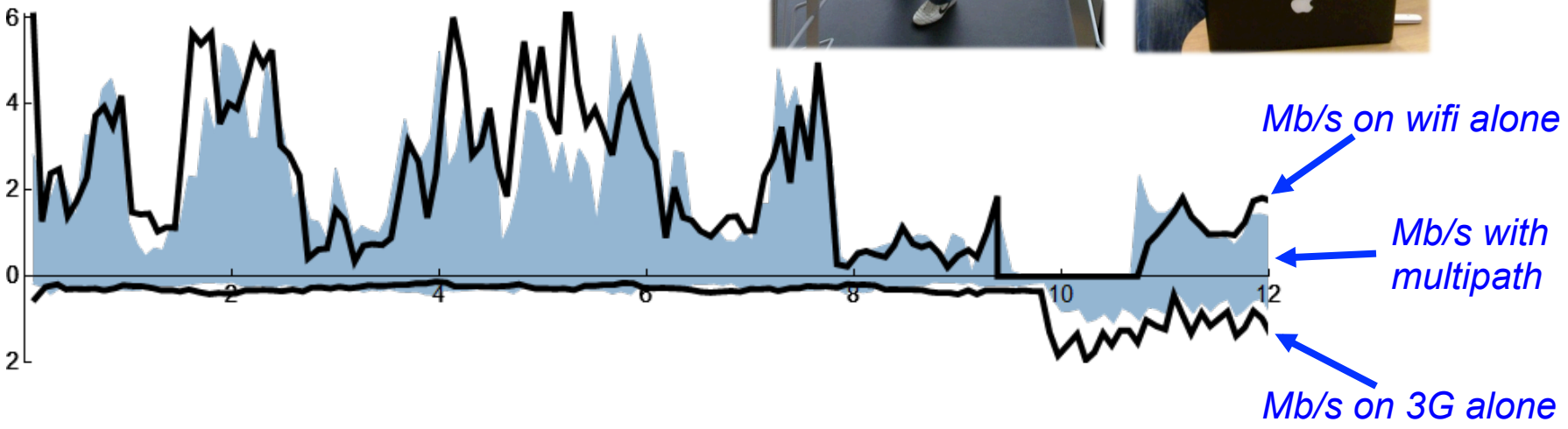


We've been standardizing multipath extensions for the TCP protocol.

At my desk. Good wifi reception, poor 3G.

Go downstairs to make coffee. Wifi fails. 3G is good.

In the kitchen. Wifi is OK, 3G is good.



Multipath TCP: protocol issues

- Need to negotiate MPTCP using options in TCP SYN.
- Need to send some sequence numbers one way, some another way.
  - One sequence number: each path only sees some paths.
  - Two sequence numbers: need to embed mapping from one to the other in packets.
- Need to send retransmissions on a different path from where the original went.

What actually happens to TCP in the wild?

- We studied 142 access networks in 24 countries.
- Ran tests to measure what actually happened to TCP.
  - Are new options actually permitted?
  - Does re-segmentation occur in the network?
  - Are sequence numbers modified?
  - Do middleboxes proactively ack?

Middleboxes and new TCP Options in SYN

Observed Behavior	TCP Port 34343	TCP Port 80	TCP Port 443
Passed	129 (96%)	122 (86%)	133(94%)
Removed	6 (4%)	20 (14%)	9 (6%)
Changed	0 (0%)	0 (0%)	0 (0%)
Error	0 (0%)	0 (0%)	0 (0%)
Total	135 (100%)	142 (100%)	142 (100%)

- Middleboxes that remove unknown options are not so rare, especially on port 80

What actually happens to TCP in the wild?

- **Rewrote sequence numbers**:
  - 10% of paths (18% on port 80)
  - Two probable causes:
    - » TCP-level proxy behaviour
    - » Firewalls trying to improve initial sequence number randomization

What actually happens to TCP in the wild?

- Testing for TCP-level proxies:
  - **Resegmented data**: 3% of paths (13% on port 80)
  - **Proxy Ack**: 3% of paths (7% on port 80)

  - Note: all of these paths also removed new options from the SYN

What actually happens to TCP in the wild?

- **Ack data not sent:**
  - 26% of paths (33% on port 80) do strange things if you send an ack for data not yet sent.
    - » Drop the ack
    - » "correct" it.

What actually happens to TCP in the wild?

- **Rewrote sequence numbers**:
  - 10% of paths (18% on port 80)
- **Resegmented data**:
  - 3% of paths (13% on port 80)
- **Proxy Ack**:
  - 3% of paths (7% on port 80)
- **Ack data not sent:**
  - 26% of paths (33% on port 80) do strange things if you send an ack for data not yet sent.

Not to mention…

- NAT
  - Pretty nearly ubiquitous, but comparatively benign
- DPI-driven rate limiters
- Lawful intercept equipment
- Application optimizers
- Anything at the server end:
  - Firewalls
  - Reverse proxies
  - Server load balancers
  - Traffic scrubbers
  - Normalizers, etc

Our methodology will not detect most of these, but we're pretty sure they're out there too.

Architecture Lesson #2

If you need better control of your network, you can always add another middlebox

Architecture Lesson #2

If you need better control of your network, you can always add another middlebox

Step 1: make sure you understand how it will interact with all the other undocumented middleboxes.

What does this mean for MP-TCP?

- Most of the protocol mechanisms in MP-TCP are dedicated to being robust to undefined behaviour of boxes in the middle of the network.
- Probably 75% of the protocol spec is dedicated to this.

- Basic strategy: fall back to regular TCP behaviour when unrecoverable events occur.
  - Not all protocols have this luxury.

IPv6 will save us!

- No.

Part 2:
Tomorrow's Internet

Option 1: Extrapolate the current Internet

- Plenty of box vendors will sell you a solution.
  - Whatever you think your problem is.

- Current apps get optimized and set in silicon.
- Future apps tunnelled over HTTP
  - (but what do all those port 80 specialized middleboxes do?)

- Impossible to reason about the concatenation of middleboxes.
  - If you think STUN/TURN/ICE is hard to reason about, you've not seen anything yet,

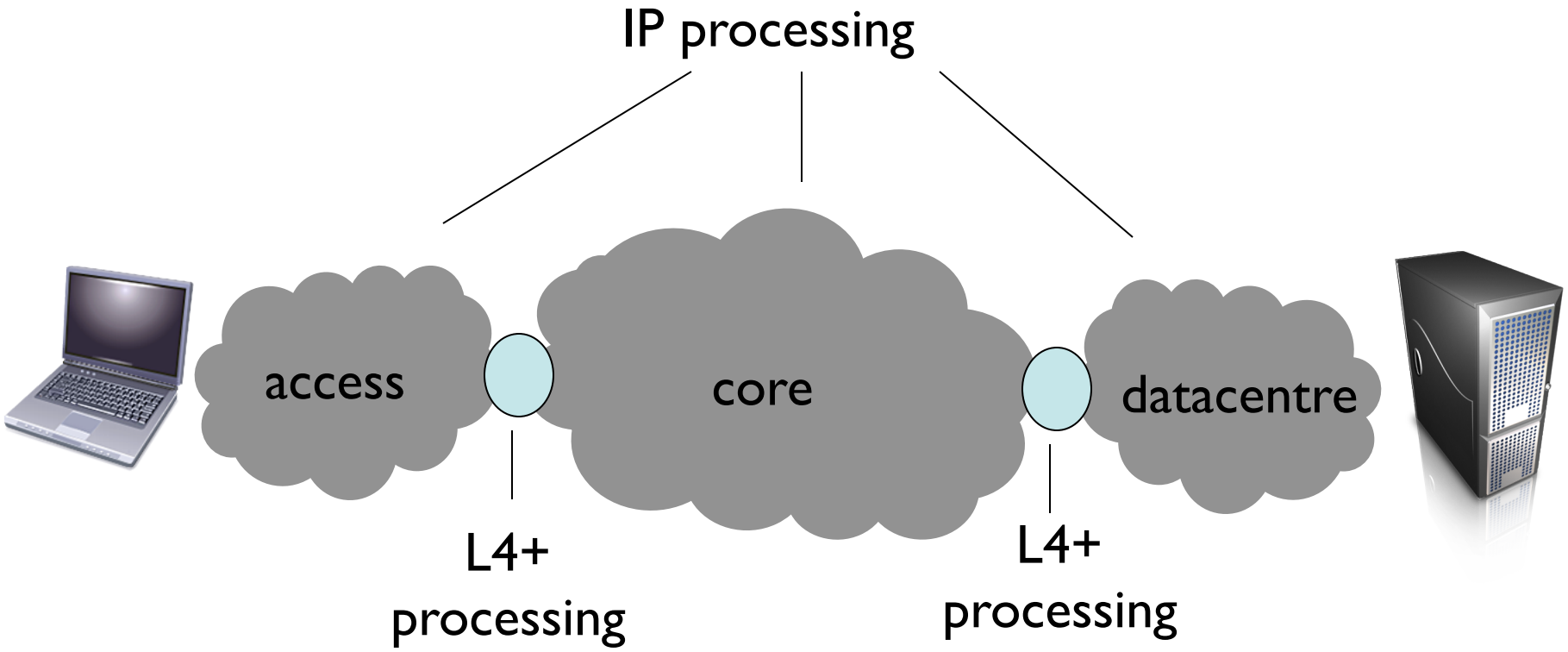
Option 2: Devise a wonderful new Internet architecture that everyone will love and deploy.

- Any change we make will need to be:
  - Incrementally deployable.
  - Pretty radical.

Option 3: Reverse engineer a new Internet architecture from the current mess.

- Observation: The Internet is becoming a concatenation of IP networks interconnected by L4+ functionality.

A segmented Internet

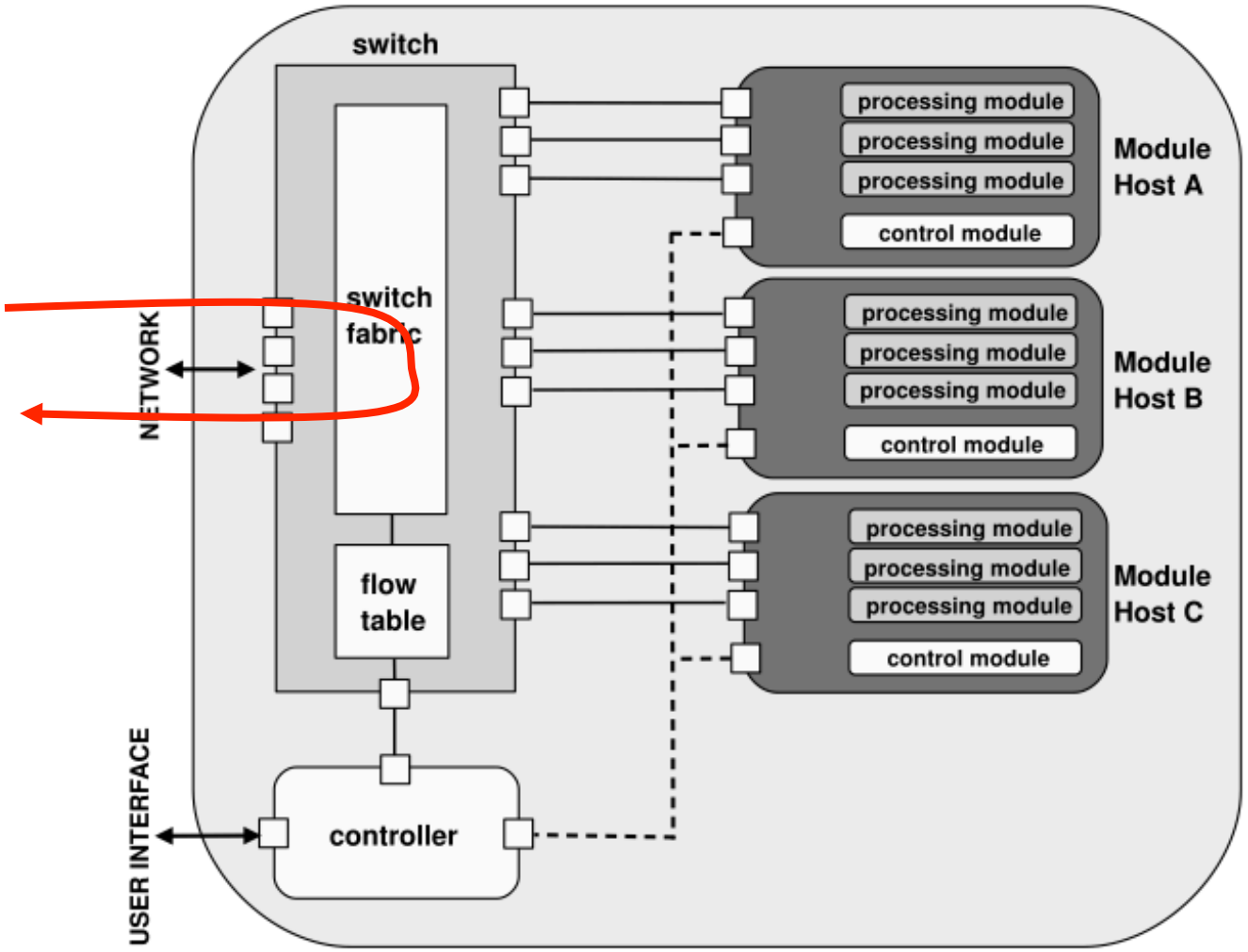


It already looks somewhat like this, but the L4+ processing is more distributed.

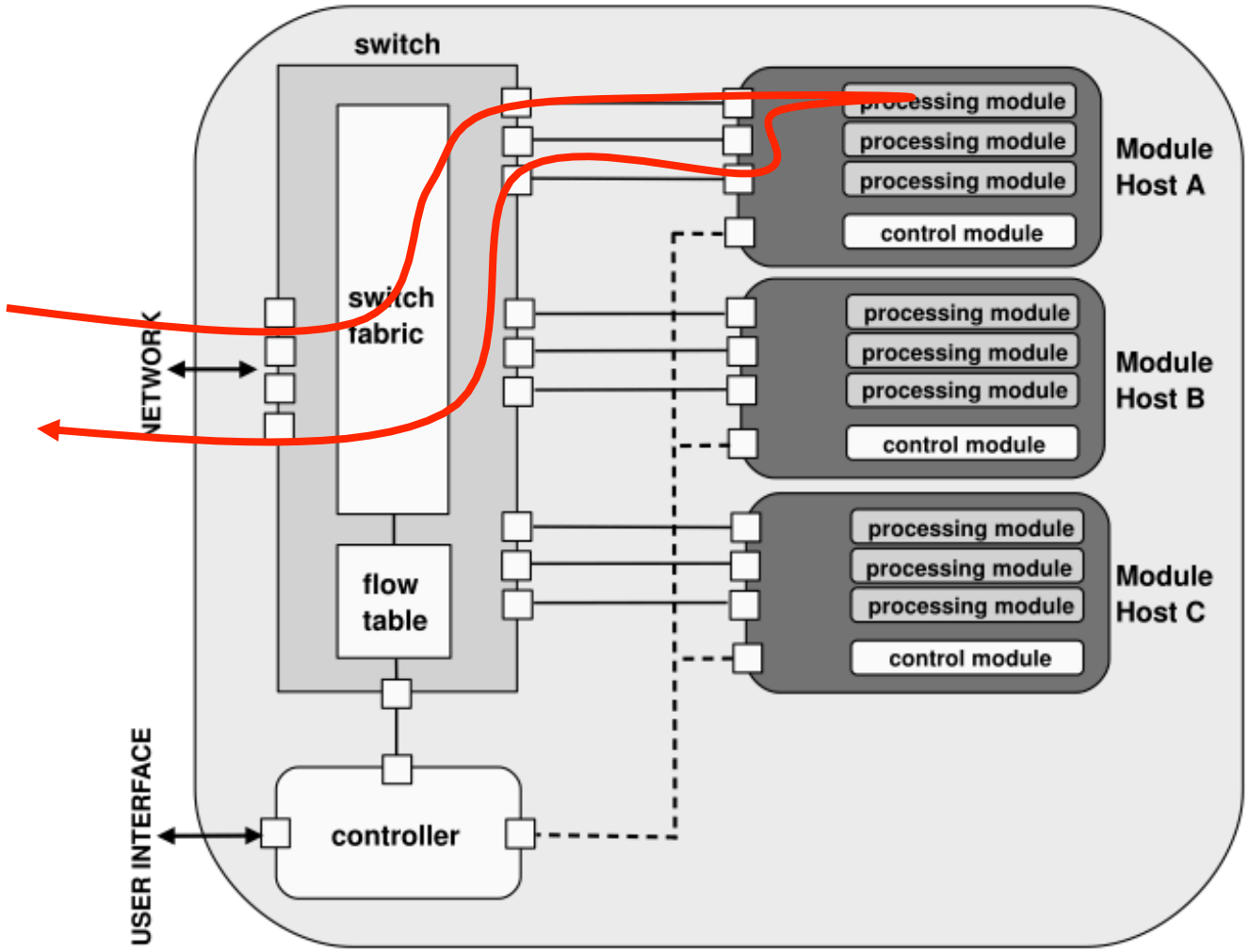
A platform for CHANGE

- Those L4+ platforms need to be more general than today's middleboxes.
  - More open.
  - More upgradable, as new apps arrive.
  - Aggregate functionality, so it is manageable.
  - Identifiable, so we can reason about them
  - Cheap and scalable.

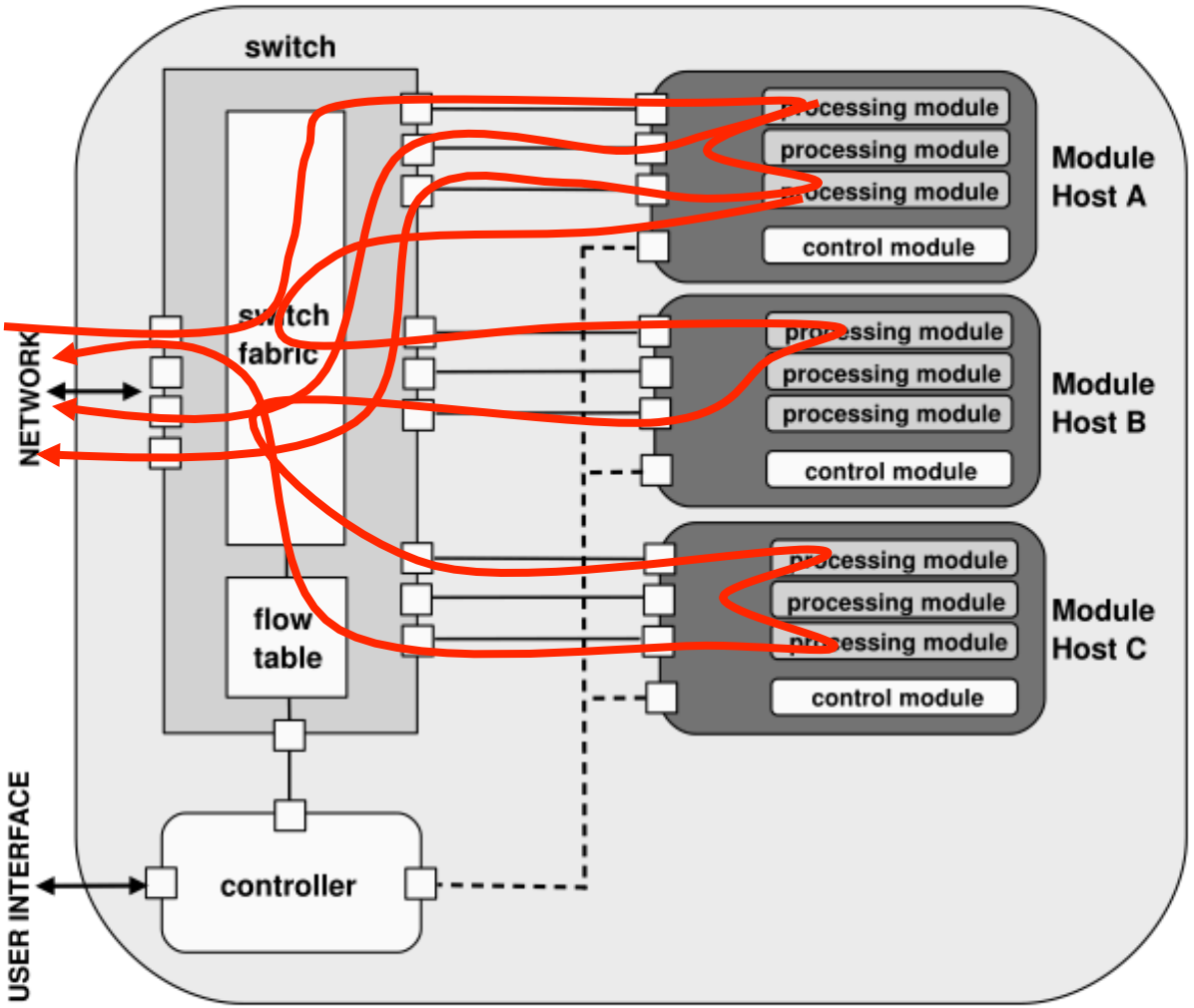
Flowstream


switch
switch fabric
flow table
controller
NETWORK
USER INTERFACE
processing module
processing module
processing module
control module
Module Host A
processing module
processing module
processing module
control module
Module Host B
processing module
processing module
processing module
control module
Module Host C

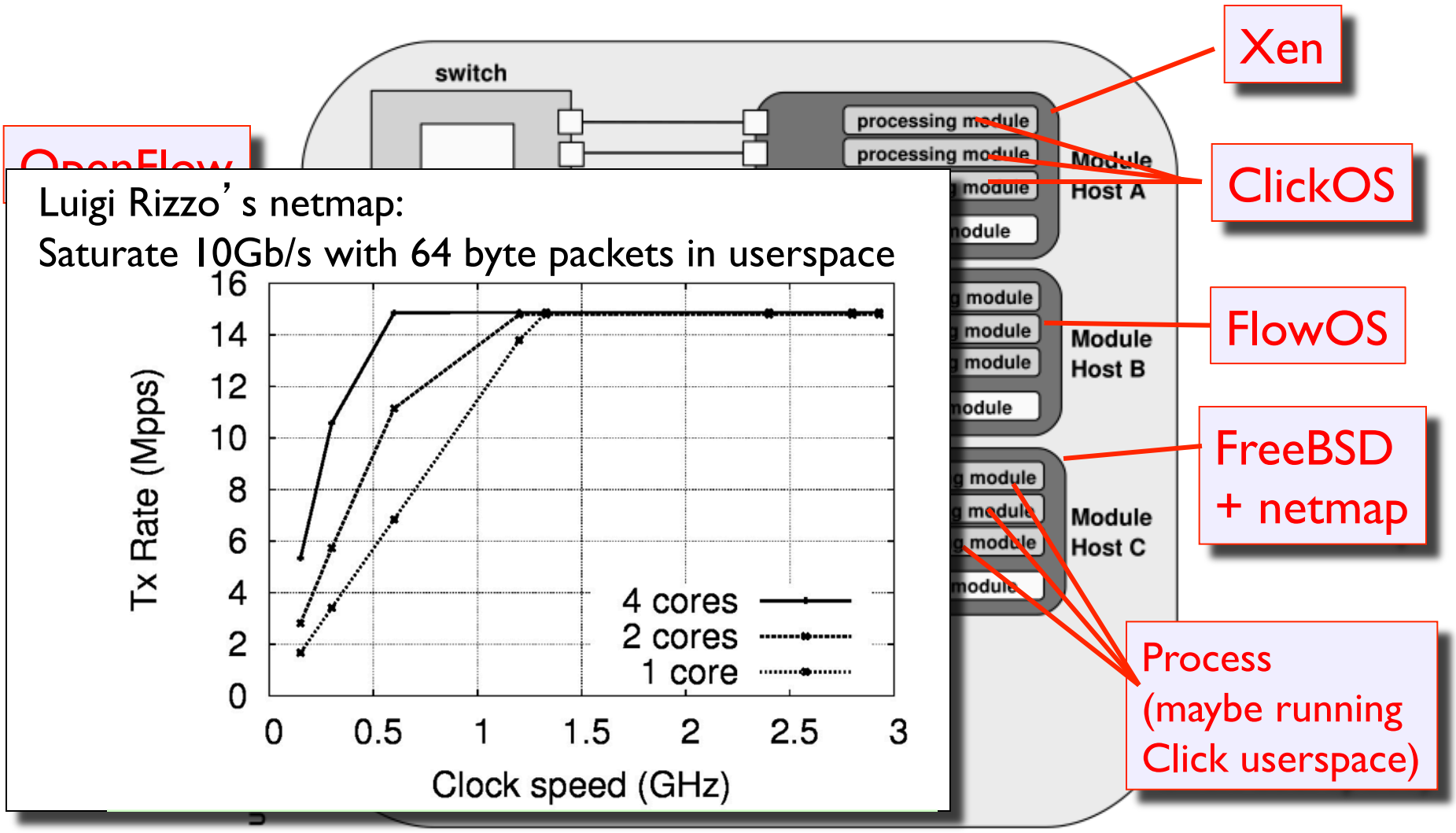
Flowstream


switch
switch fabric
flow table
controller
NETWORK
USER INTERFACE
processing module
processing module
processing module
control module
Module Host A
processing module
processing module
processing module
control module
Module Host B
processing module
processing module
processing module
control module
Module Host C

Flowstream


switch
switch fabric
flow table
controller
NETWORK
USER INTERFACE
processing module
processing module
processing module
control module
Module Host A
processing module
processing module
processing module
control module
Module Host B
processing module
processing module
processing module
control module
Module Host C

Flowstream


switch
processing module
processing module
Module
Host A
module
module
module
module
Module
Host B
module
module
module
module
Module
Host C
module
Xen
ClickOS
FlowOS
FreeBSD
+ netmap
Process
(maybe running
Click userspace)
Luigi Rizzo's netmap:
Saturate 10Gb/s with 64 byte packets in userspace
Tx Rate (Mpps)
Clock speed (GHz)
4 cores
2 cores
1 core
0
2
4
6
8
10
12
14
16
0
0.5
1
1.5
2
2.5
3

A better mousetrap

- Change is not primarily about building a better middlebox.
  - Though much of the effort goes on this.

- The observation is that the Internet has already embraced **flow processing**, albeit implicitly.
  - We believe we need to make flow processing a **first-class citizen** within the Internet architecture.

What's a flow?

Very vague:

- Packets that have something in common.

Vague:

- An aggregation of packets that requires processing in a way other than regular IP forwarding.

State-centric:

- Packets that are processed differently because of state in the network.

User-centric:

- Whatever a user requests processing on.

Operator-centric:

- Packets for which we will violate network neutrality.

Pragmatic:

- Something you can specify an openflow filter for.

Application middleboxes

- Many applications are already built around middleboxes:
  - Skype supernodes
  - SMTP servers and IMAP servers for email.
  - CDNs for video streaming.

- Unlike ISP-imposed middleboxes, these are:
  - Application specific
  - Directly addressable

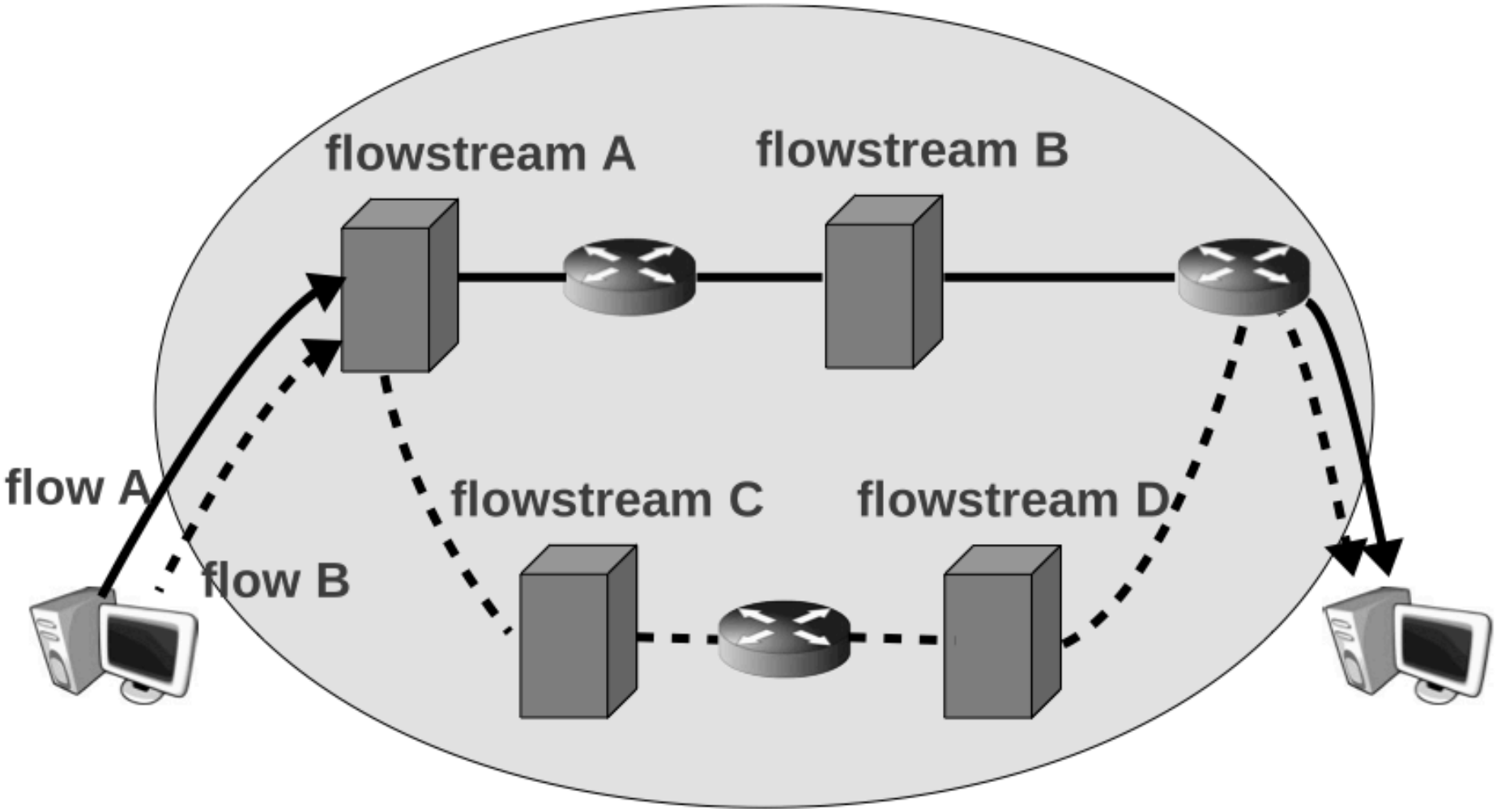
Unified Goal

- *Application middleboxes*
- *ISP-imposed middleboxes*

Our goal is to provide a unifying framework that can perform both middlebox roles.

- Network operators can manage their net effectively
- App developers can enhance their applications.

Empowering both the ends and the middle

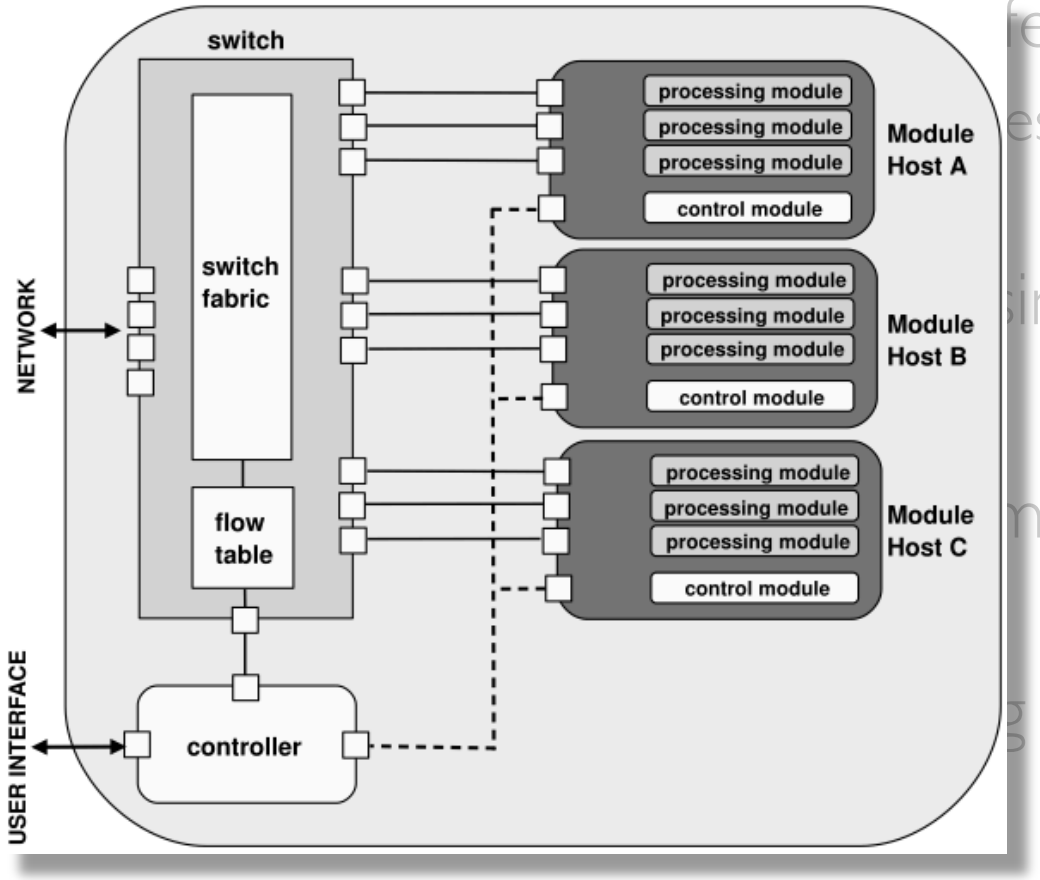


Architectural components

- A scalable general purpose flow processing platform.
- A categorization of flow processing into a few classes.
  - Allows reasoning about concatenation of processing without needing to know the details.
- A way to identify who can request processing.
- A way to name flows to be processed.
- A way for end-systems to discover platforms which they can enlist to perform processing.
- A way to attract flows to a flow processing platform.

Architectural components

- A scalable general purpose flow processing platform.
- A categ… few classes.
  - Allows … essing without needir…
- A way t… sing.
- A way t…
- A way f… ms which they can…
- A way t… g platform.



Architectural components

- A scalable general purpose flow processing platform.
- A categorization of flow processing into a few classes.
  - Allows reasoning about concatenation of processing without needing to know the details.
- A way to identify who can request processing.
- A way to name flows to be processed.
- A way for end-systems to discover platforms which they can enlist to perform processing.
- A way to attract flows to a flow processing platform.

Classes of Flow Processing

Read-only (RO):	Read the contents of packets to perform some action. Eg monitor.
Filter (F)	Drop some or all packets, or rate limit. Does not affect flow behaviour.
ReRoute (RR)	Change the path, but otherwise leave unchanged.
Redirect (RD)	Change the destination.
Modify (M)	Change the contents in a way that changes the E2E semantics
Originate (O)	Originate new packets on behalf of another host.

ReRoute examples: TCP flow

- Client can tunnel flow to a flow processing platform to reroute the **forward path**.
- Middlebox can route the **reverse path** via itself by NATing the forward path. [As can a TCP proxy]
- VPN can reroute **both paths** (and pin contents too)

  - *Observation: platform that needs to see bidirectional flows needs to reroute to pin both directions.*
  - *After reroute, flow continues to its original destination.*

Modify can do anything.

Pretty hard to reason about in general case.

Invariants.

- Processing module can specify invariants it assumes on the rest of the path.
- So long as the invariants are satisfied, composition is safe.

Examples:

- *"TCP bytestream contents must be invariant"*
- *"Packet boundaries must be invariant"*

Apps and flow processing classes

Type	RO	F	RR	RD	M	O
DPI	X					
NAT	X		X			
Rate limiter	X	X				
Firewall	X	X				
IDS	X					
IPS	X	X				X
Transcoder	X				X	
Multimedia mixer	X	X				X
Implicit proxy	X		X			
Explicit proxy	X	X			X	
Scrubber	X	X			X	
Tunnel	X		X			
Multicast	X			X		X

Authorization

- On-path providers can instantiate flow-processing functionality.
  - Can't stop them anyway.

- Source and destination also share ownership of a flow.
  - Can we allow them to set up flow processing?

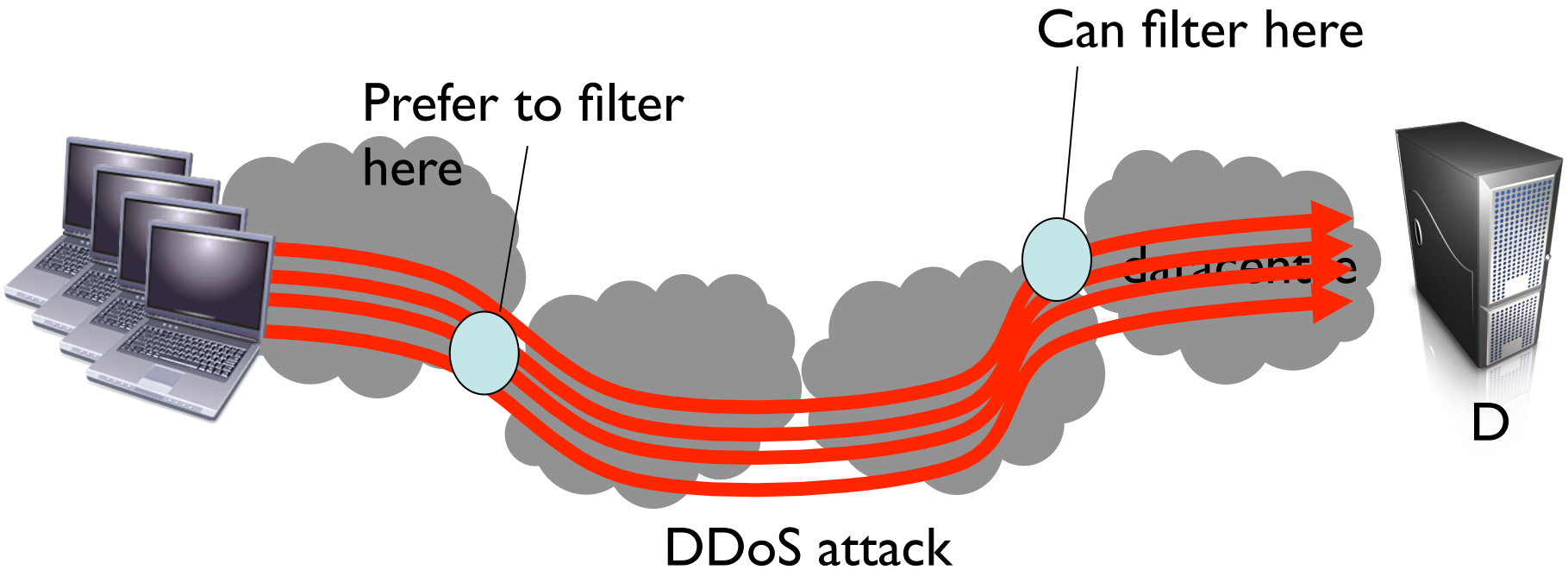
Authorization

- Source or destination-initiated processing:
  - Need some way to pay.
  - Need authorization framework to avoid hijacking.

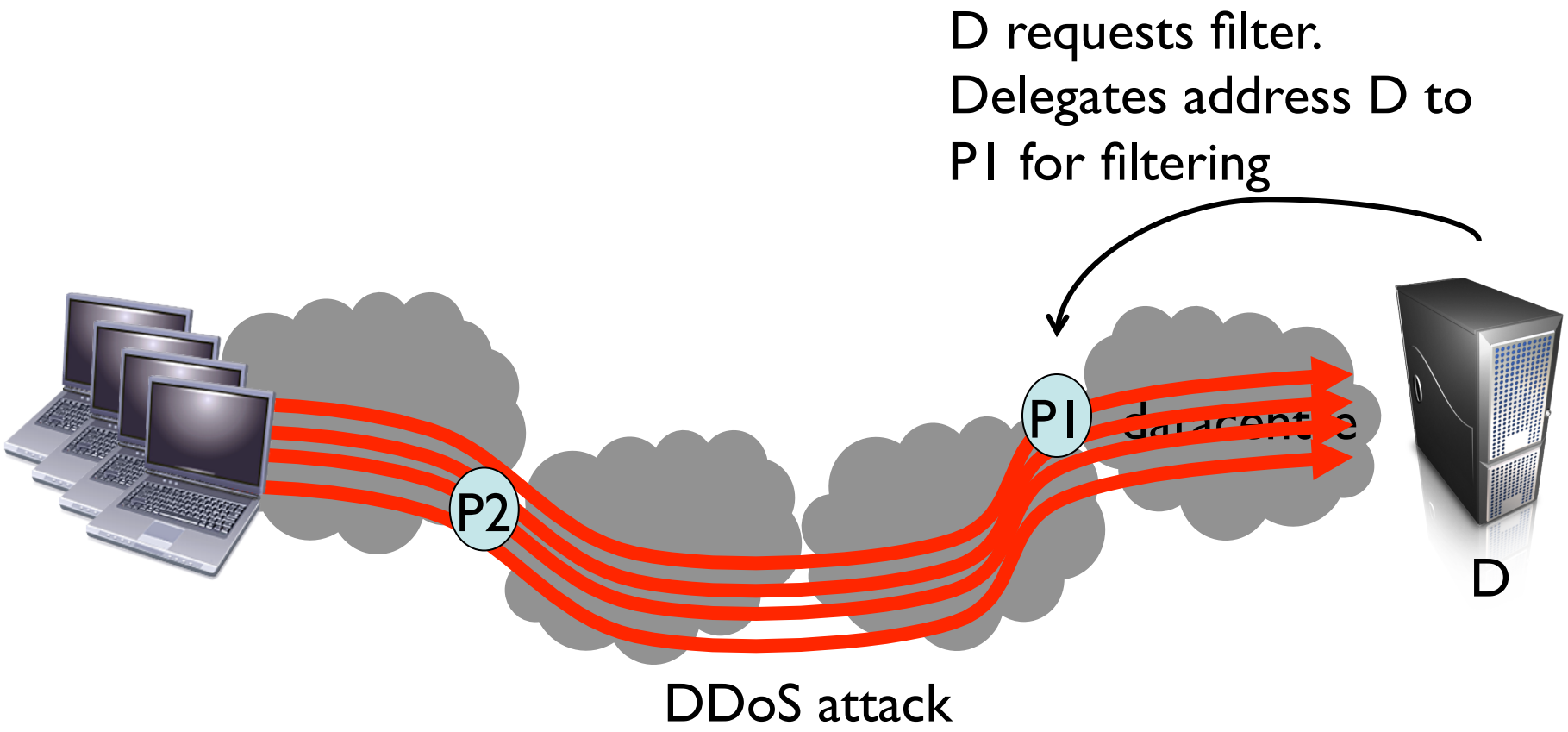
Authorization

- Request from destination is simple(ish) to authenticate.
  - Simple nonce exchange proves requester is downstream. May be sufficient for monitoring, etc.
  - Otherwise need to prove address ownership
    - » E.g. via RPKI
- Request from source is harder. Anyone upstream can NAT traffic to claim ownership.
  - Address proof (even using RPKI) only proves requester is on path upstream.

Direct authorization is insufficient

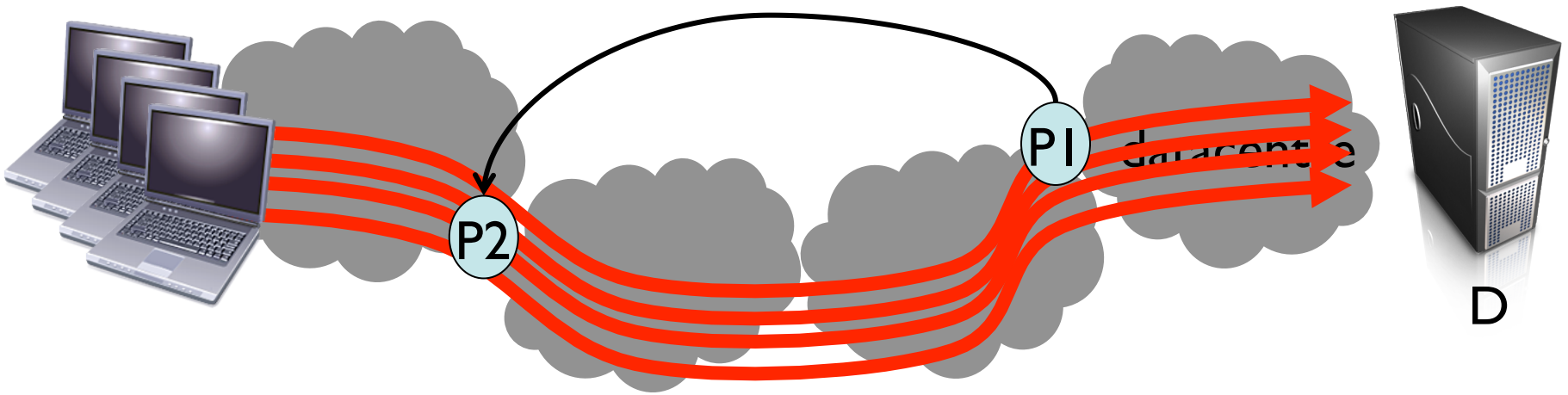


Delegated authorization.



Delegated authorization.

P1 can request filtering at P2 for traffic to deligated address D



DDoS attack

Low level security rules

Changing source address

- Only permitted if new address has been delegated to the requester. Can be address of platform.

Changing destination address

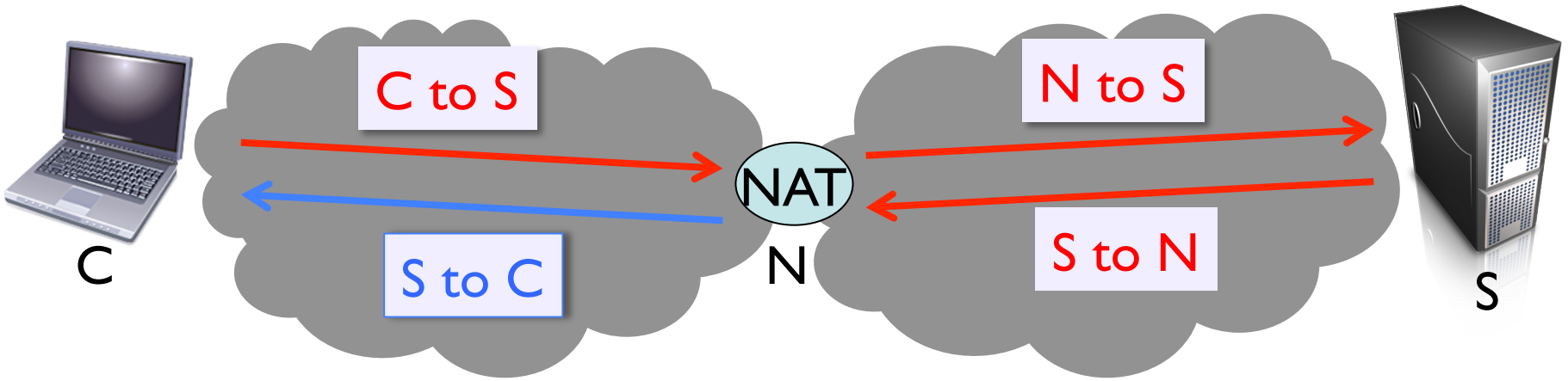
- Only permitted if new address has been delegated for use by the requester.
- Default-off: new destination must agree.

Implicit authorization

- Explicit authorization is not always necessary

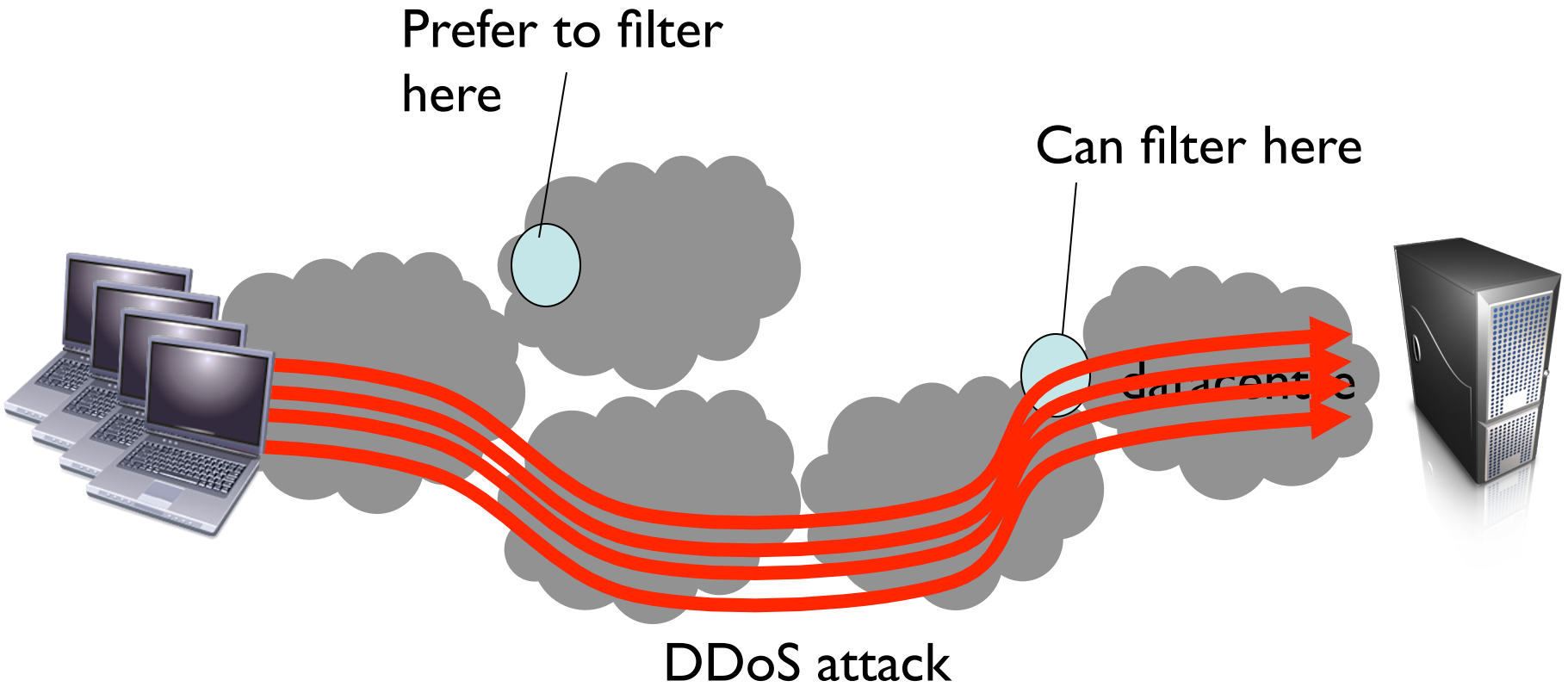
Implicit Authentication

- Consider a NAT

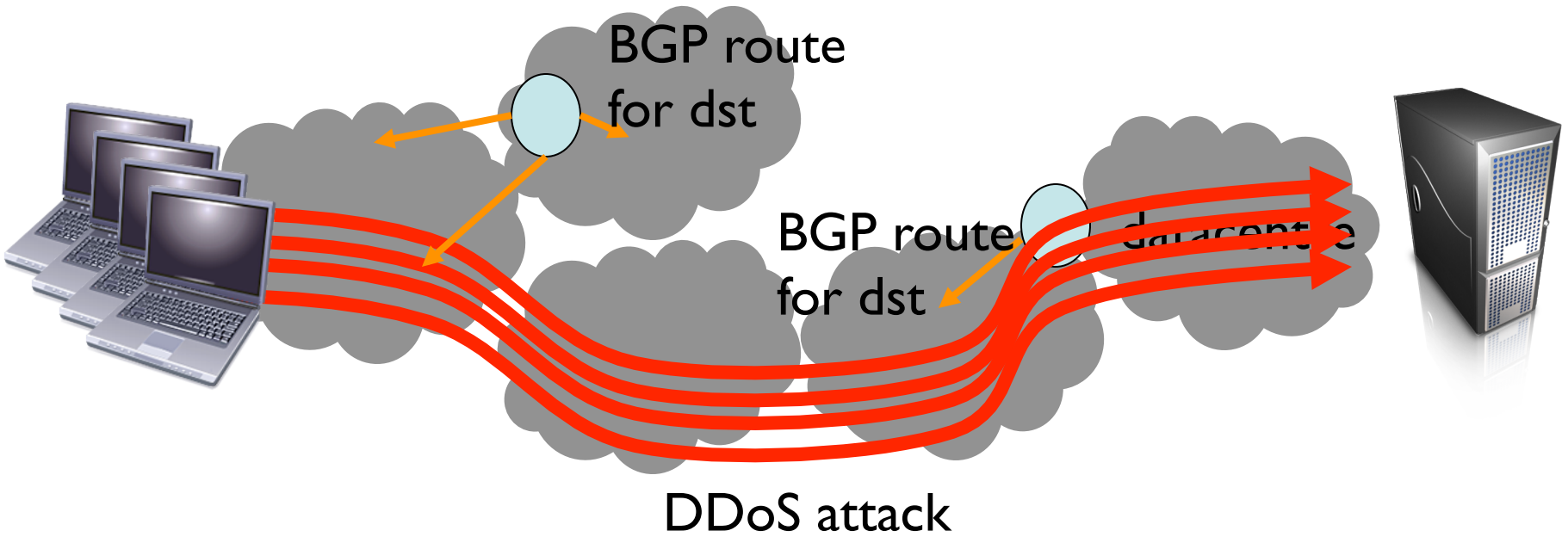


- NAT ReDirect's response packet to C
- Normally this would require C to request the redirection.
- In such cases we say C has **implicitly authenticated** N to reply to C when C initiated the connection.

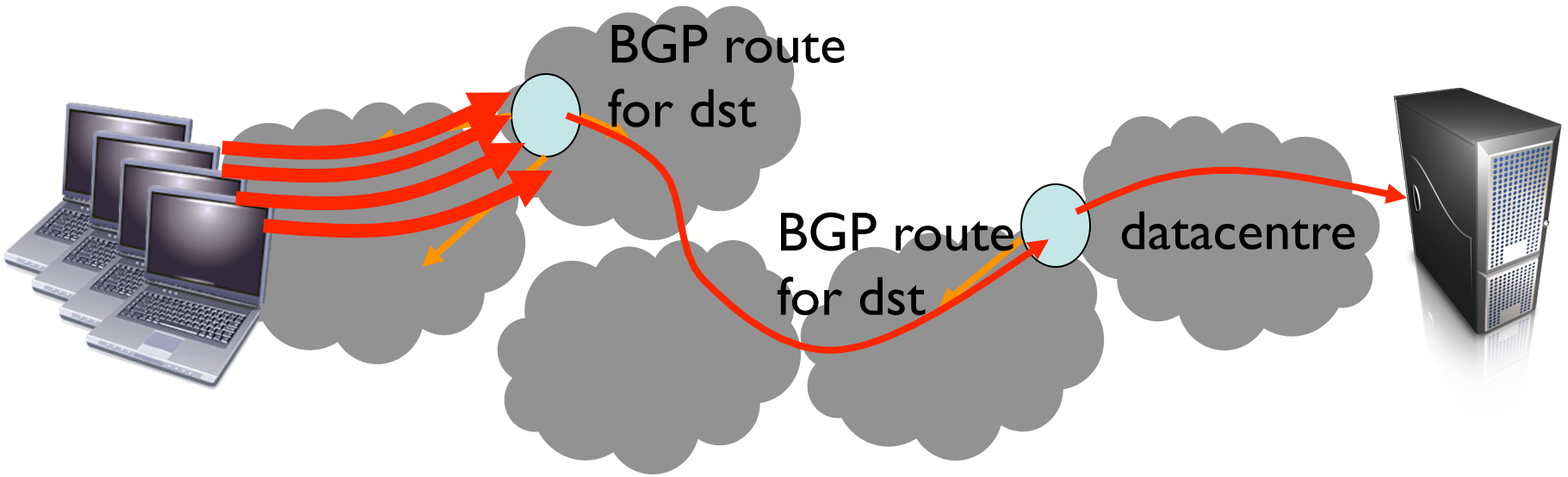
Becoming on-path



Becoming on-path



Becoming on-path



Destination ISP has dynamically extended the reach of its network

ReRouting using BGP

- Rerouting using BGP is not trivial.
  - Still need to be able to reach real destination.
  - Need to avoid looping.
  - Anycast BGP not ideal for TCP - route change can cause platform switch.
  - Difficult to scope announcements without causing blackholes or loops.

- For most purposes, higher layer reroute is simpler.
- Only tool available for uncooperative clients though.

http://www.change-project.eu/

- Flow processing as a first class primitive
- Scalable extensible software platform to enable it.
- Mechanisms to remotely authorize instantiation of processing and protocols to communicate with flow processing platforms.
- Architectural framework to reason about the emergent behaviour of the network.

The End-to-End Principle

- Don't put application functionality in the network!

- Application specific functions should reside in the end-hosts of a network rather than the intermediary nodes, provided they can be implemented "completely and correctly" in the end hosts.

The End-to-End Principle

- Application specific functions should reside in the end-hosts of a network rather than the intermediary nodes, provided they can be implemented "completely and correctly" in the end hosts.

- Essentially this is a recipe for enabling application innovation.
  - But it only works if the network operator really doesn't care about which applications are running.
  - Security, performance, legal requirements are some reasons they do in practice care.

The End-to-Middle-to-End Principle

- When application-specific functions are placed in the intermediary nodes, it must be possible to reason about the emergent behaviour.

Going with the flow…

- Currently flow processing in middleboxes serves to inhibit new applications.
  - Optimization of the present
  - Inextensible inflexible network security

- **Key question: is it possible to re-claim the middlebox as a force for enabling end-to-end innovation?**