Flow processing and the rise of the middle.

Mark Handley, UCL

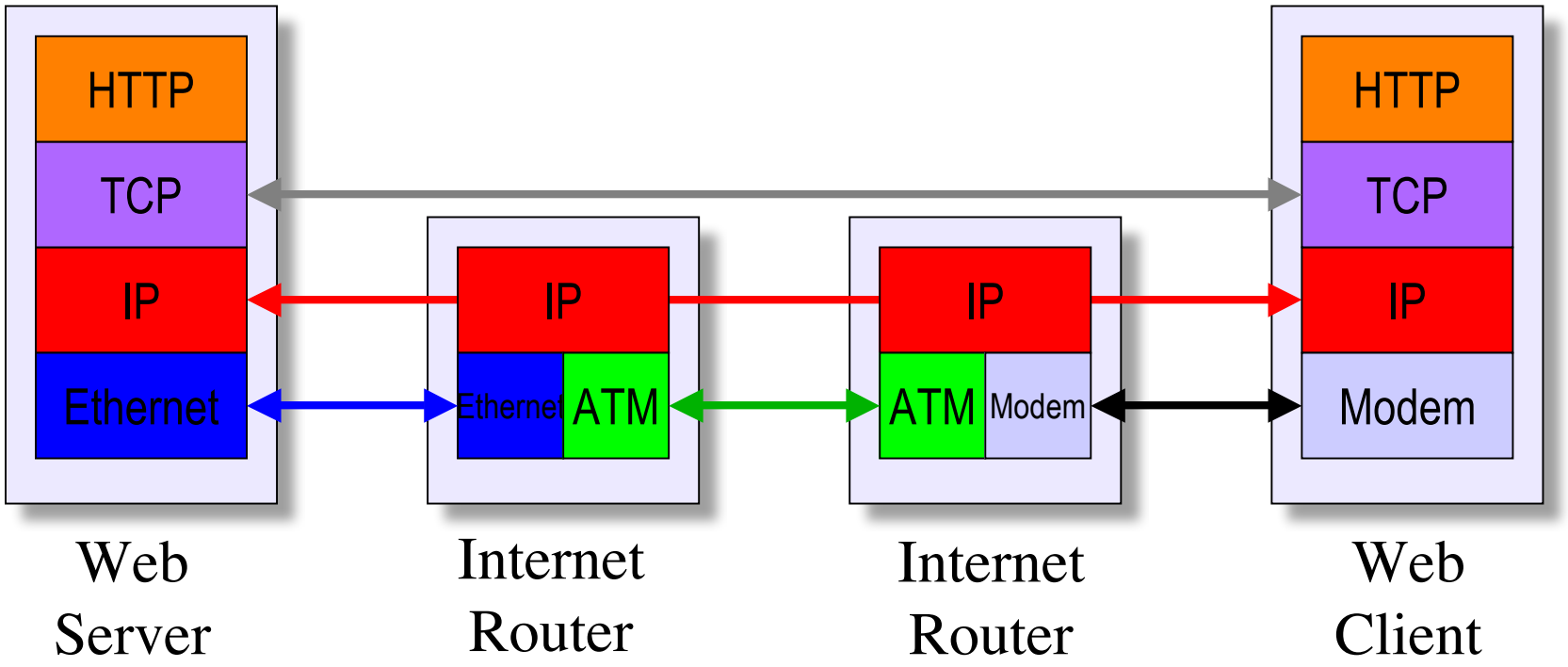
With acknowledgments to Michio Honda, Laurent Mathy, Costin Raiciu, Olivier Bonaventure, and Felipe Huici.

Part I

Today's Internet

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.



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 retransmiss , flow control, etc.
- A lication using OS' CP API to job.

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	80	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)
  - Presumably to improve initial sequence number randomization

- **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

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

What actually happens to TCP in the wild?

- **Rewrote sequence numbers**: 10% of paths (18% on port 80)
  - Presumably to improve initial sequence number randomization
- **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
- **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
  - Load balancers
  - Traffic scrubbers
  - Normalizers, etc

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

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?)

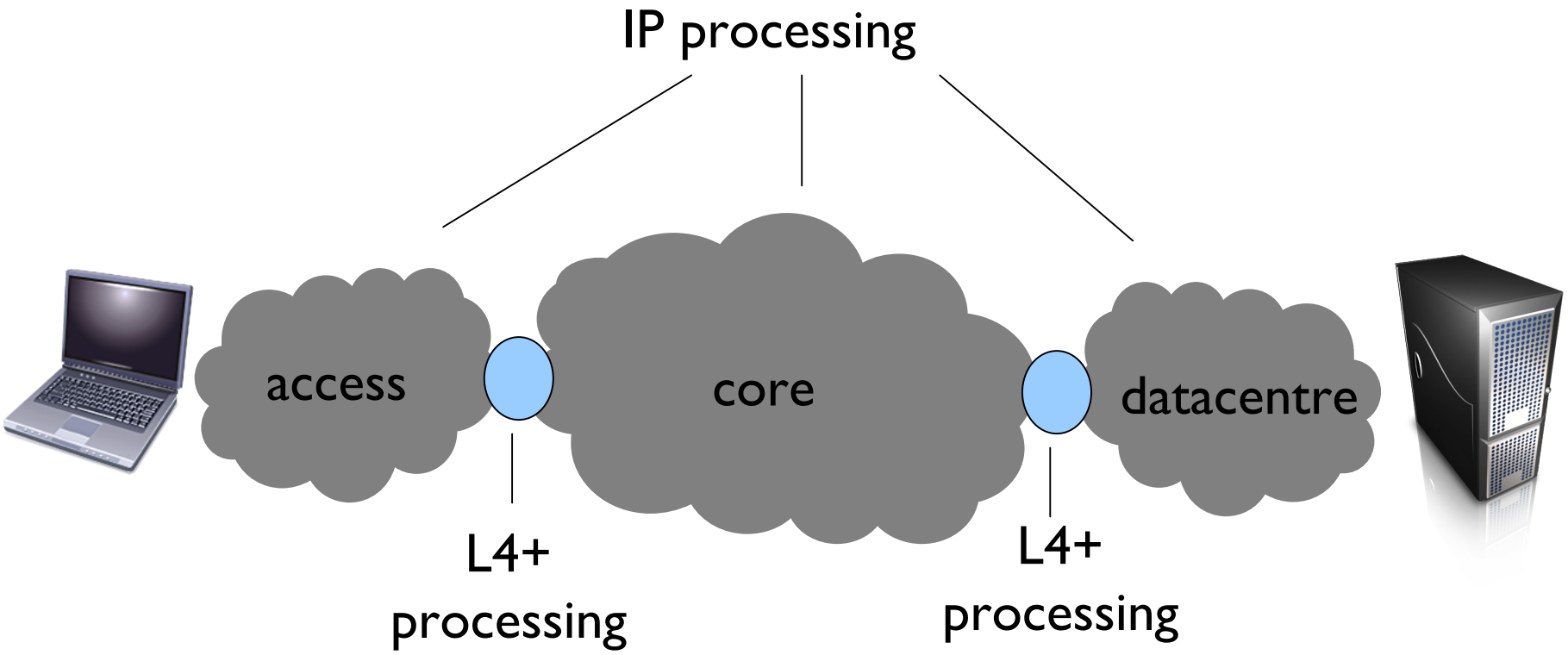
- Impossible to reason about the concatenation of middleboxes.
  - If you think STUN/TURN/ICE is hard to reason about, you've not seen anything yet,

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

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

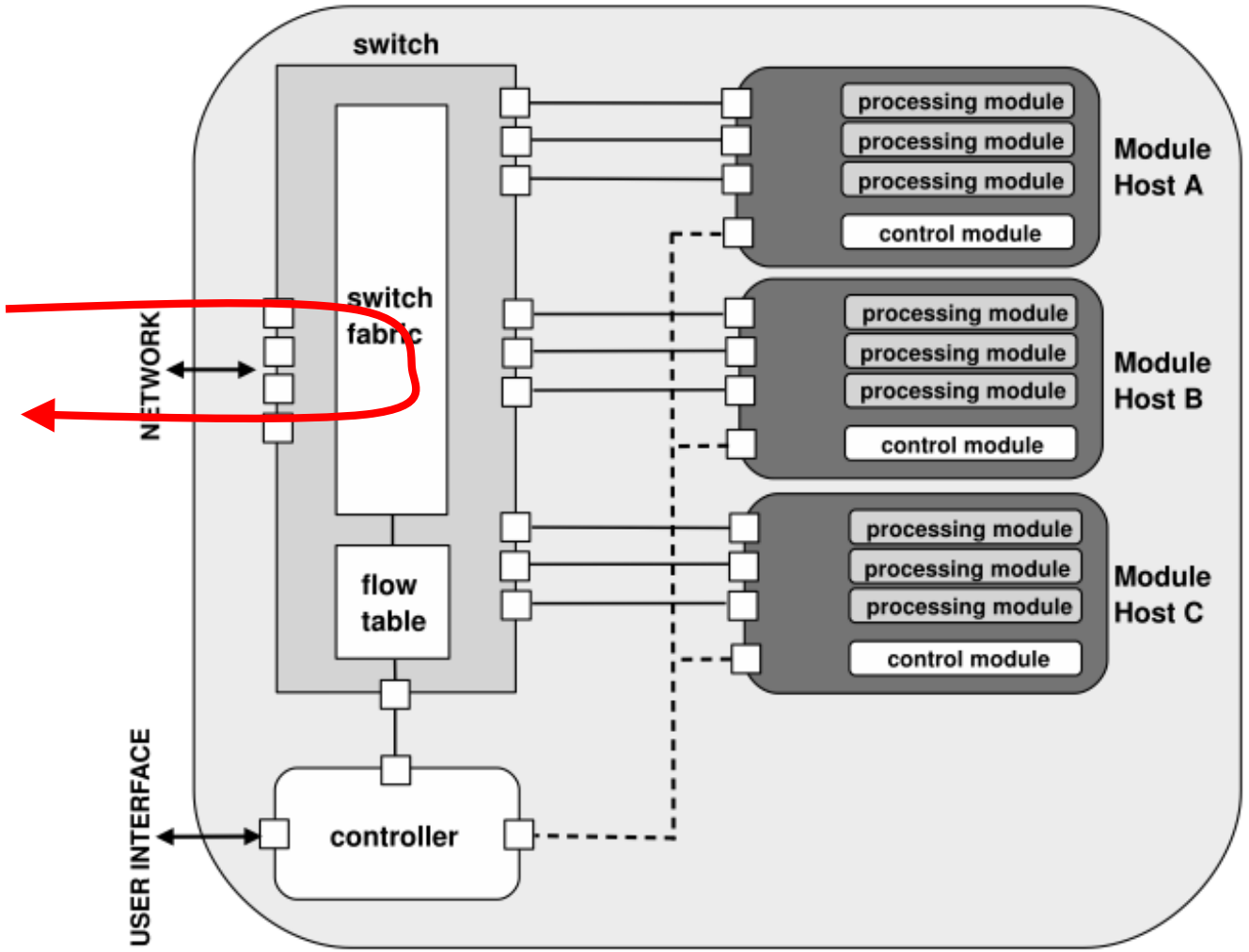


It already looks somewhat like this, but the L4+ processing is more distributed.

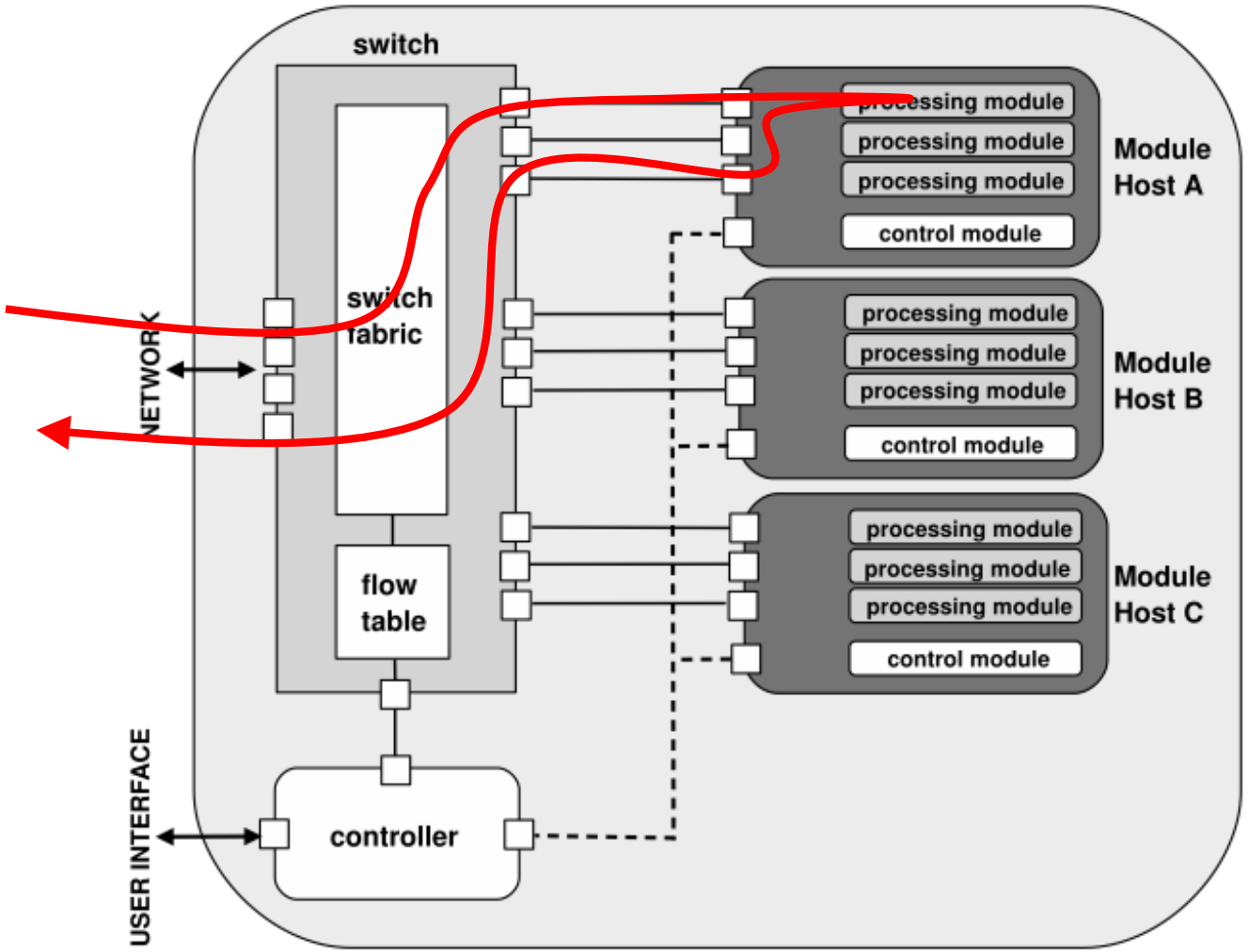
A platform for Change

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

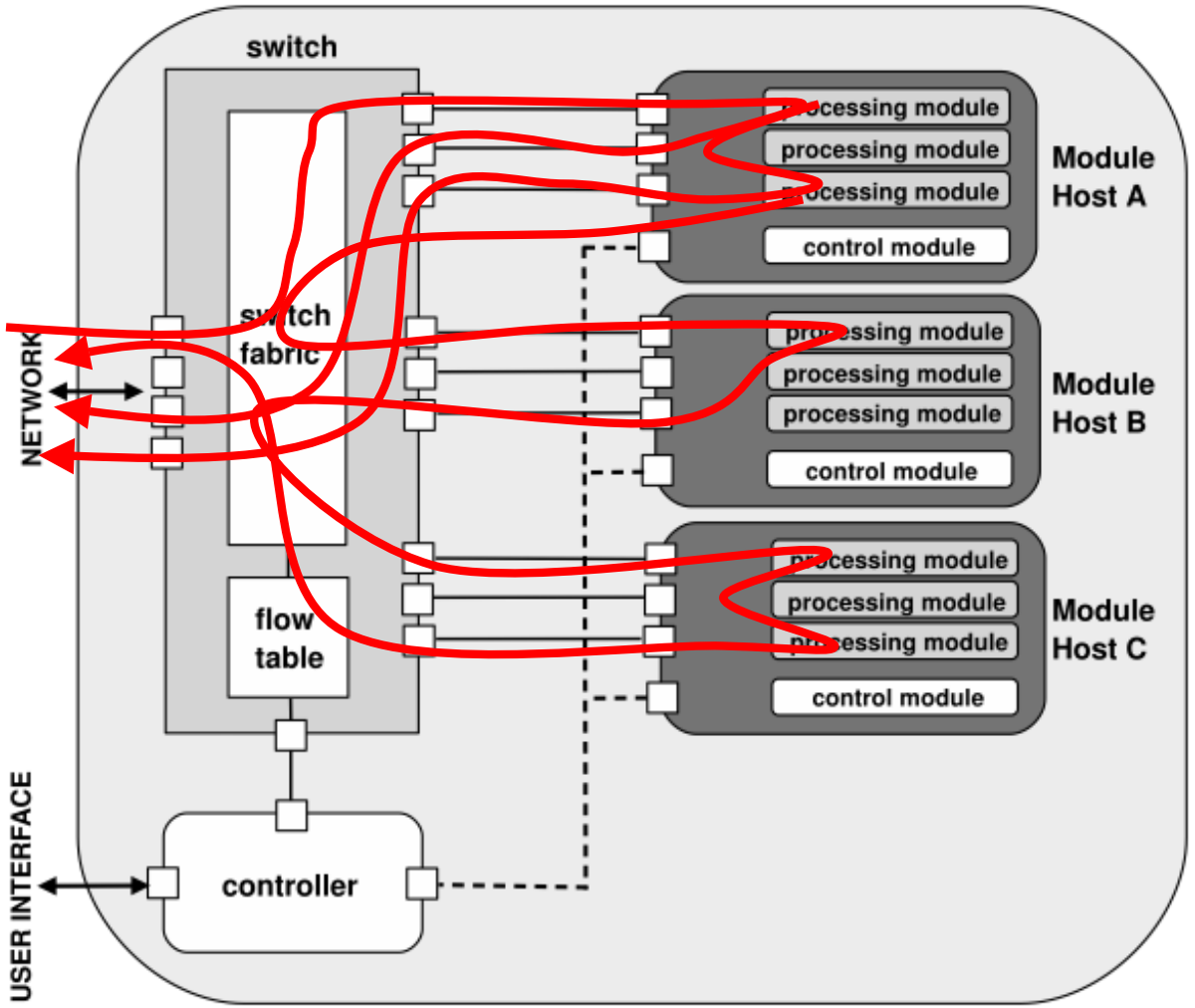
Flowstream


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

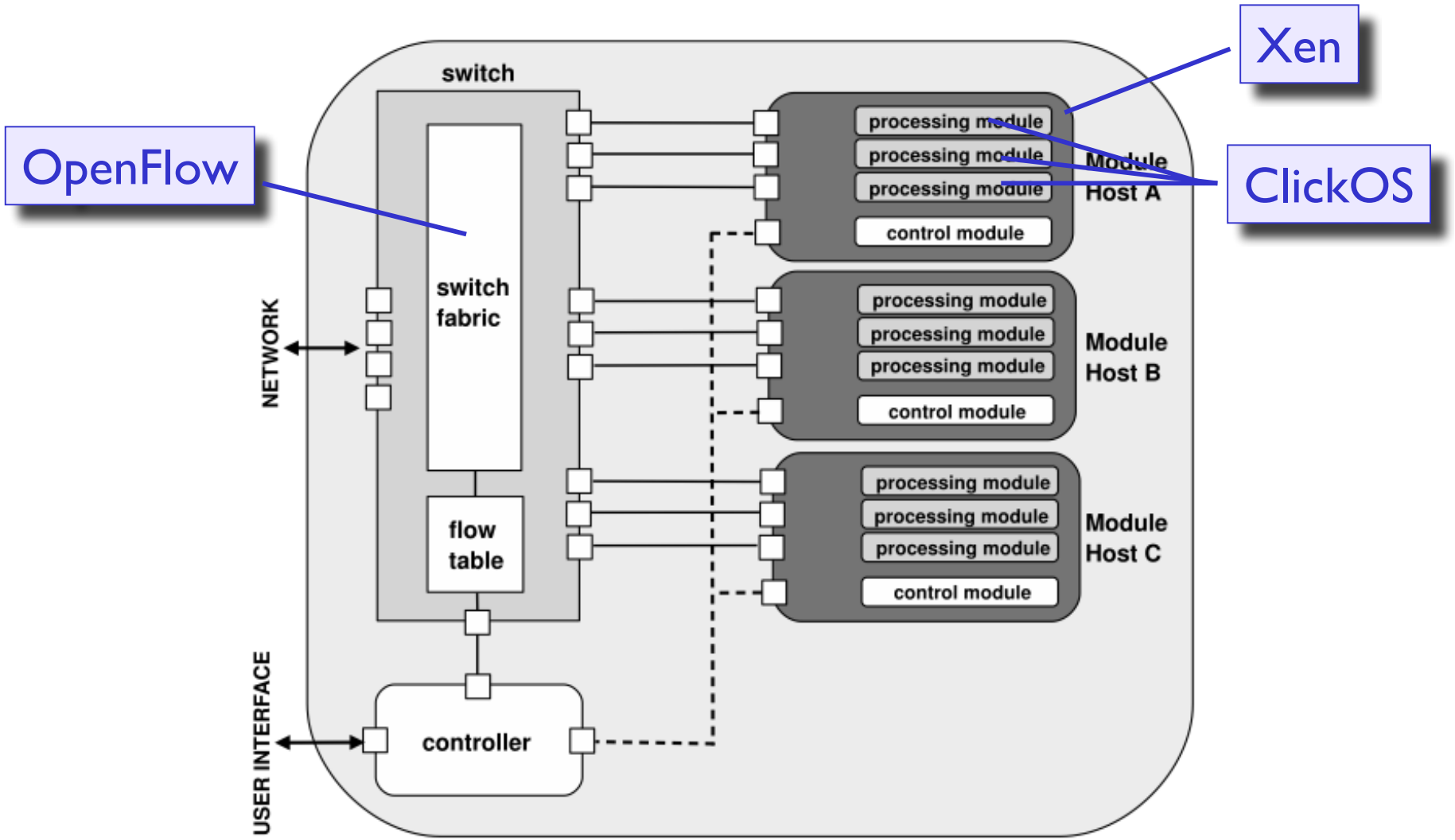
Flowstream


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

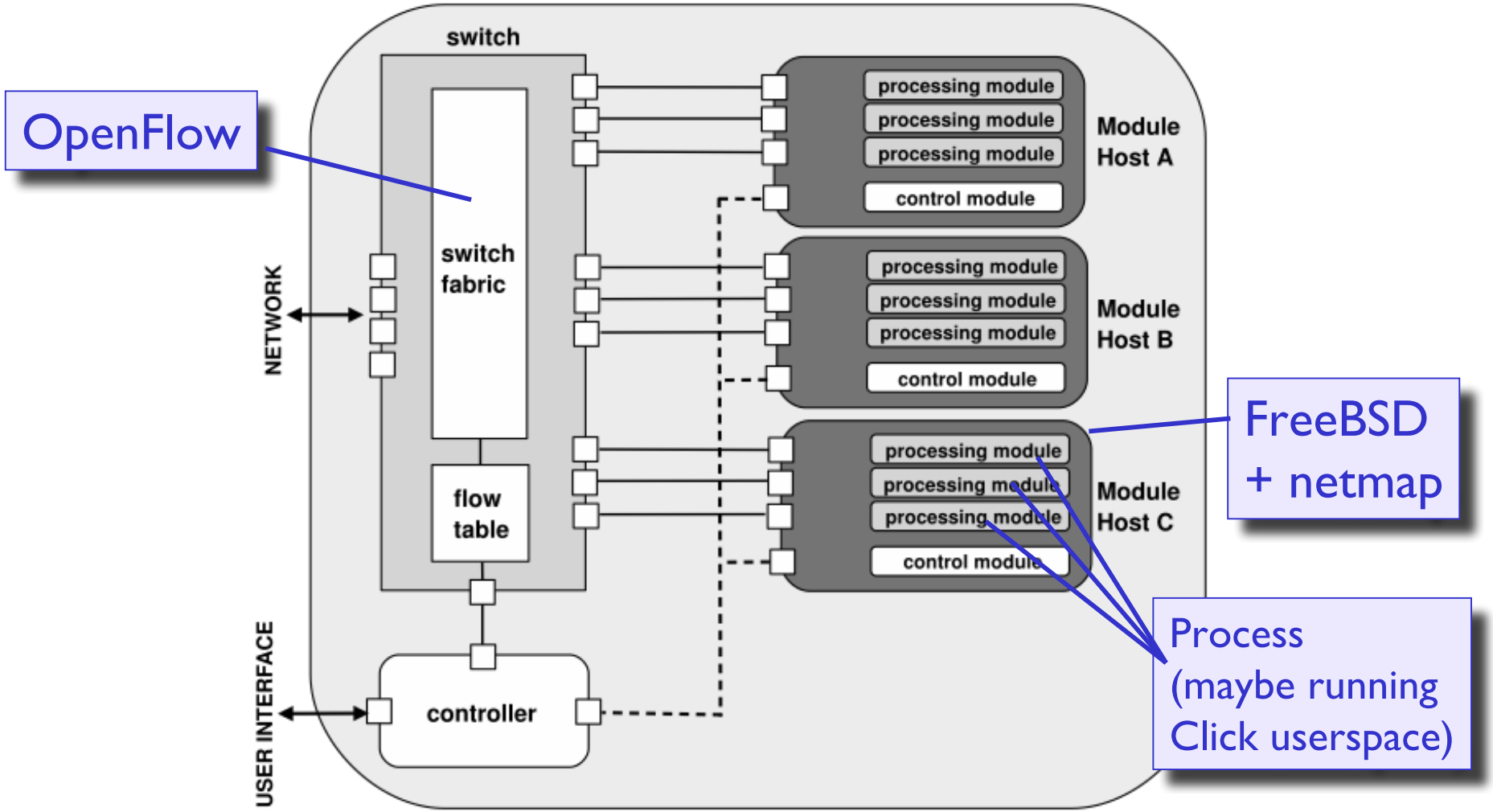
Flowstream


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

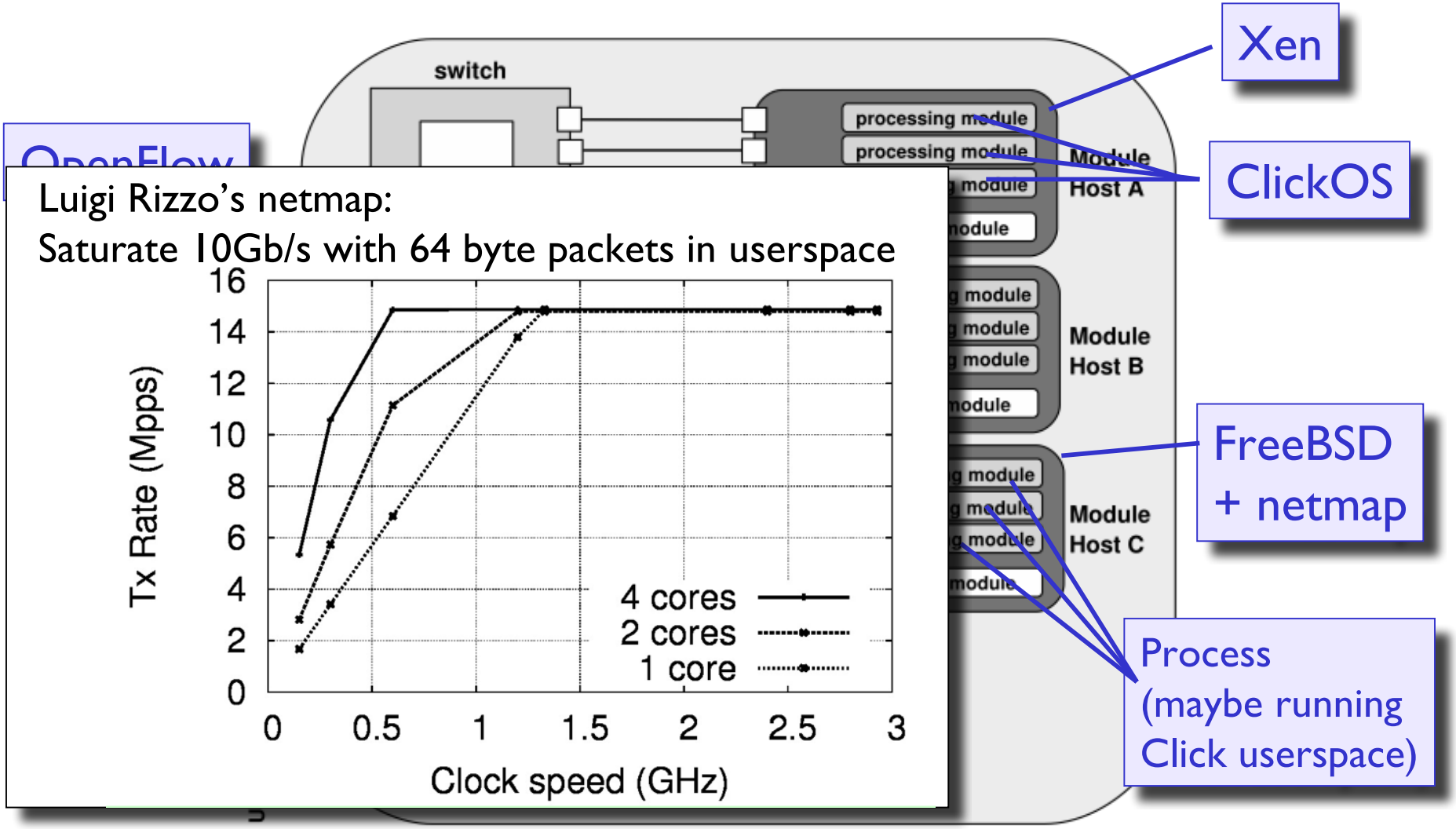
Flowstream


Xen
OpenFlow
ClickOS
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

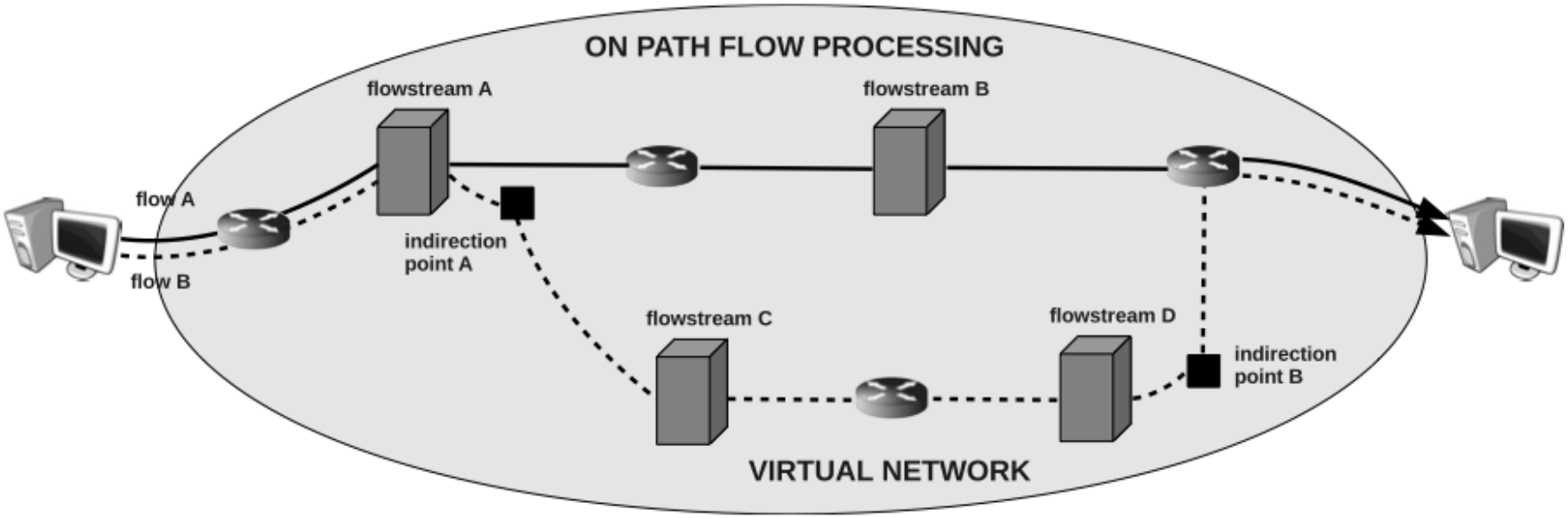
Flowstream


OpenFlow
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
FreeBSD + netmap
Process (maybe running Click userspace)

Flowstream


Xen
ClickOS
FreeBSD
+ netmap
Process
(maybe running
Click userspace)
switch
processing module
processing module
Module
Host A
Module
Host B
Module
Host C
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

Empowering the ends, not just the middle



Types of Processing

1. Monitoring/read-only
2. Drop/filter/rate-limit
3. Redirect (eg tunnel)
4. Tee
5. Rewrite

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?

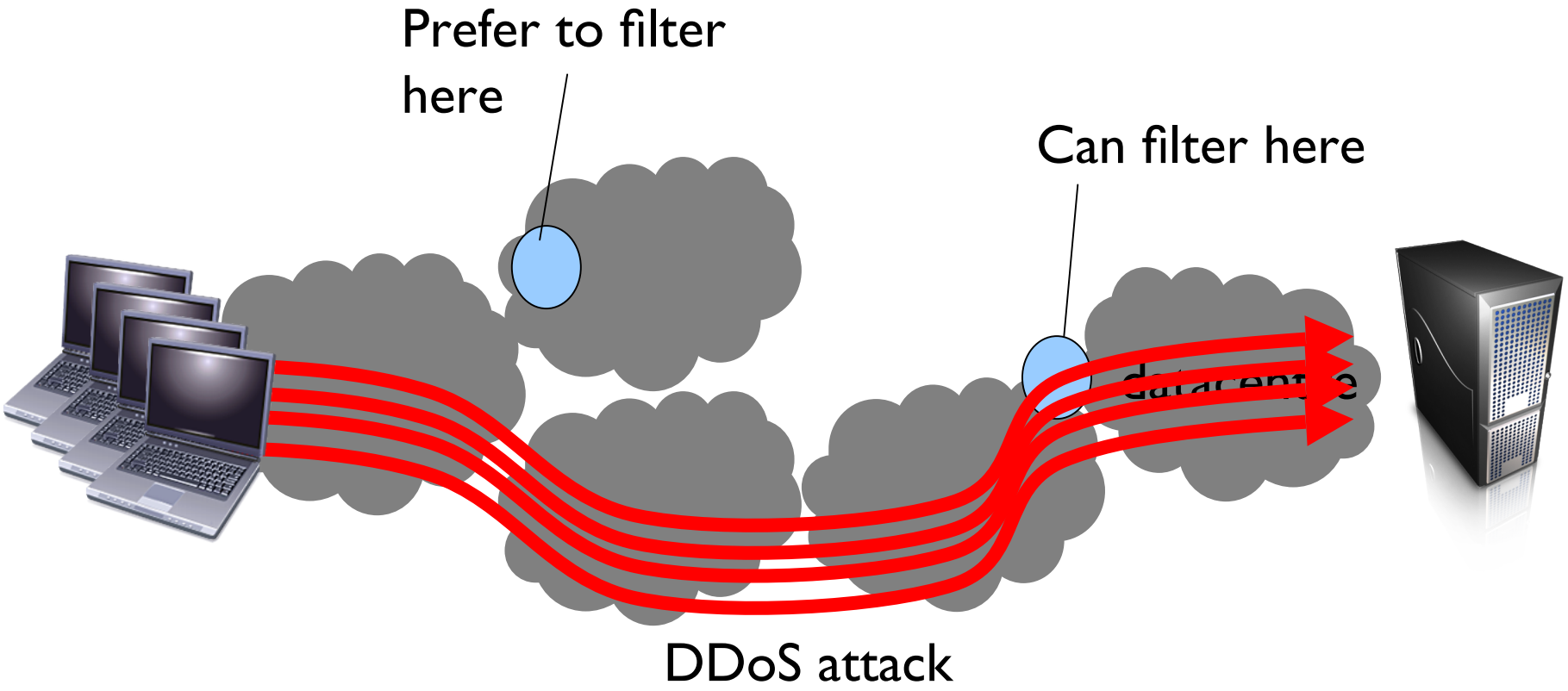
Authorization

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

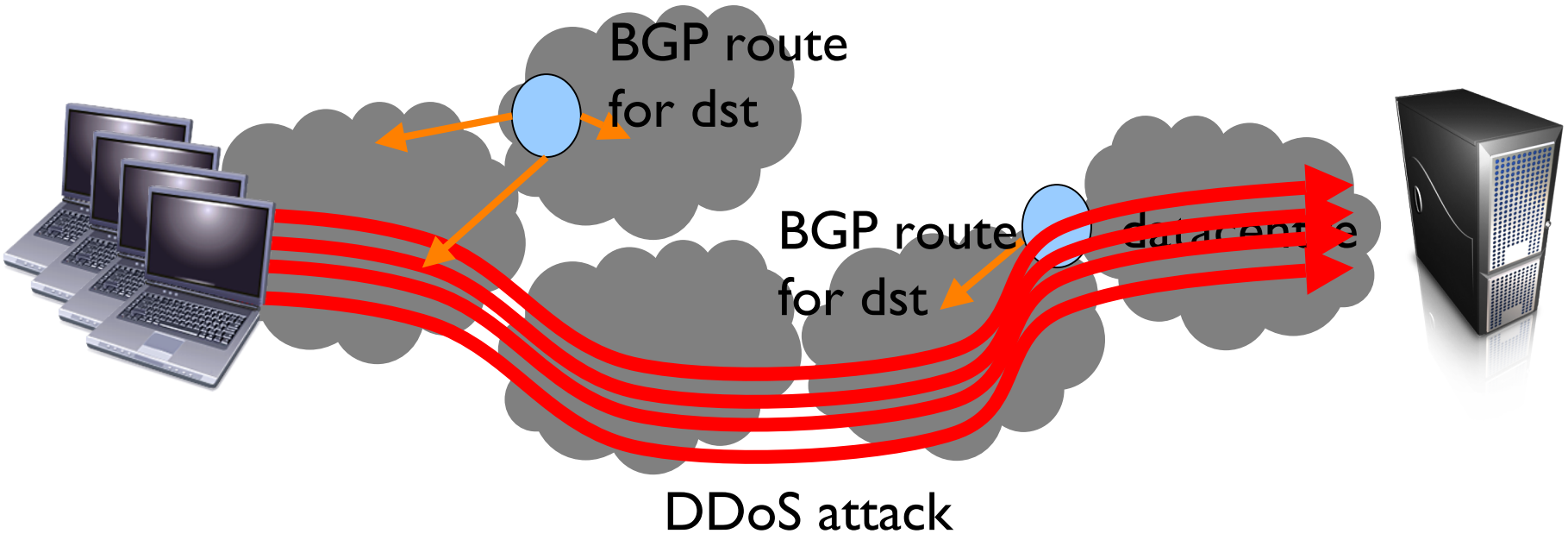
Authorization

- 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 (eg 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.

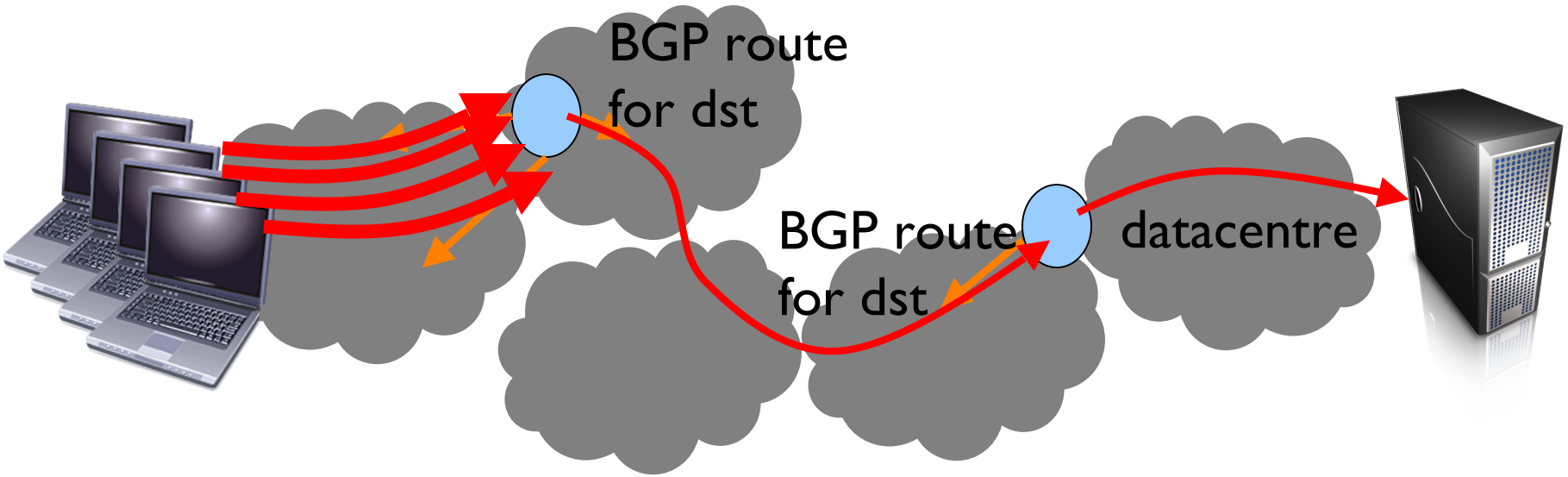
Becoming on-path



Becoming on-path



Becoming on-path



Destination ISP has dynamically extended the reach of its network

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.
- Protocols to communicate with flow processing platforms, so we can reason about the network.

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?**