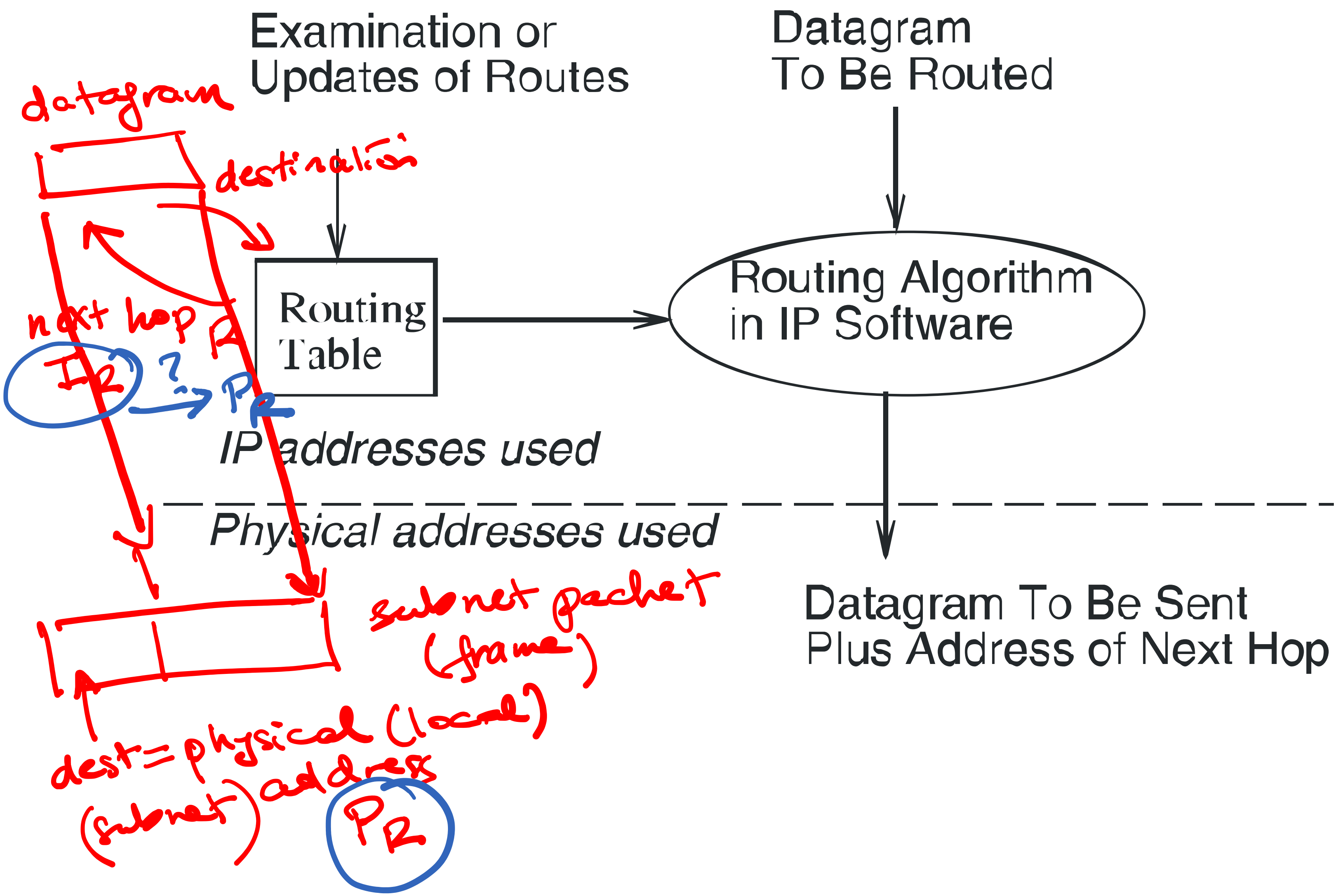
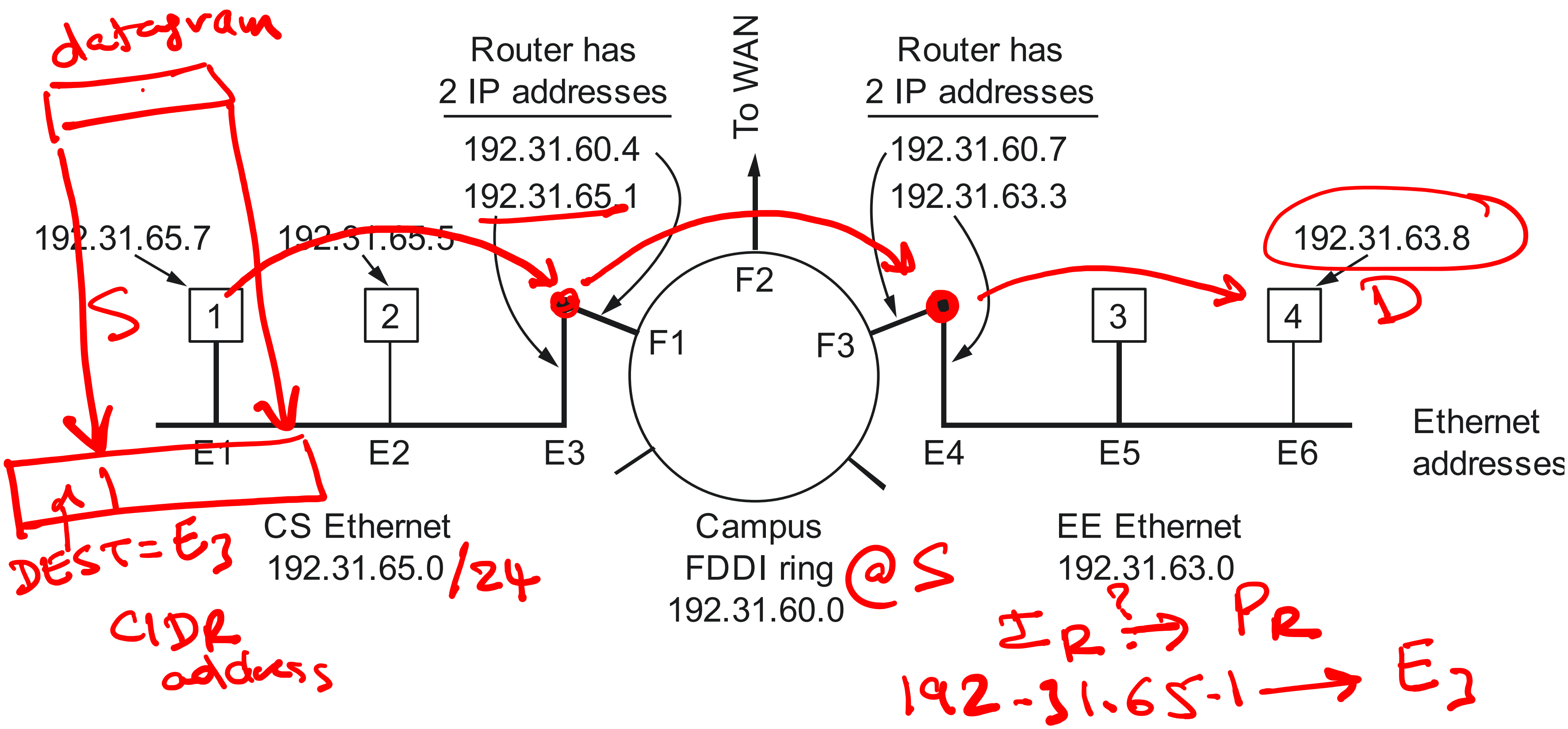
Routing


Examination or
Updates of Routes
Datagram
To Be Routed
Routing
Table
Routing Algorithm
in IP Software
IP addresses used
Physical addresses used
Datagram To Be Sent
Plus Address of Next Hop
datagram
destination
next hop
IR
?
PR
subnet packet
(frame)
dest = physical (local)
(subnet) address
PR

Address Translation



- Map IP (high-level) addresses into physical (low-level) addresses
  - destination host
  - next hop router

Address Translation (cont'd)

① Direct mapping

IP address of next hop

netid	hostid

= physical address

ETH 48 bits

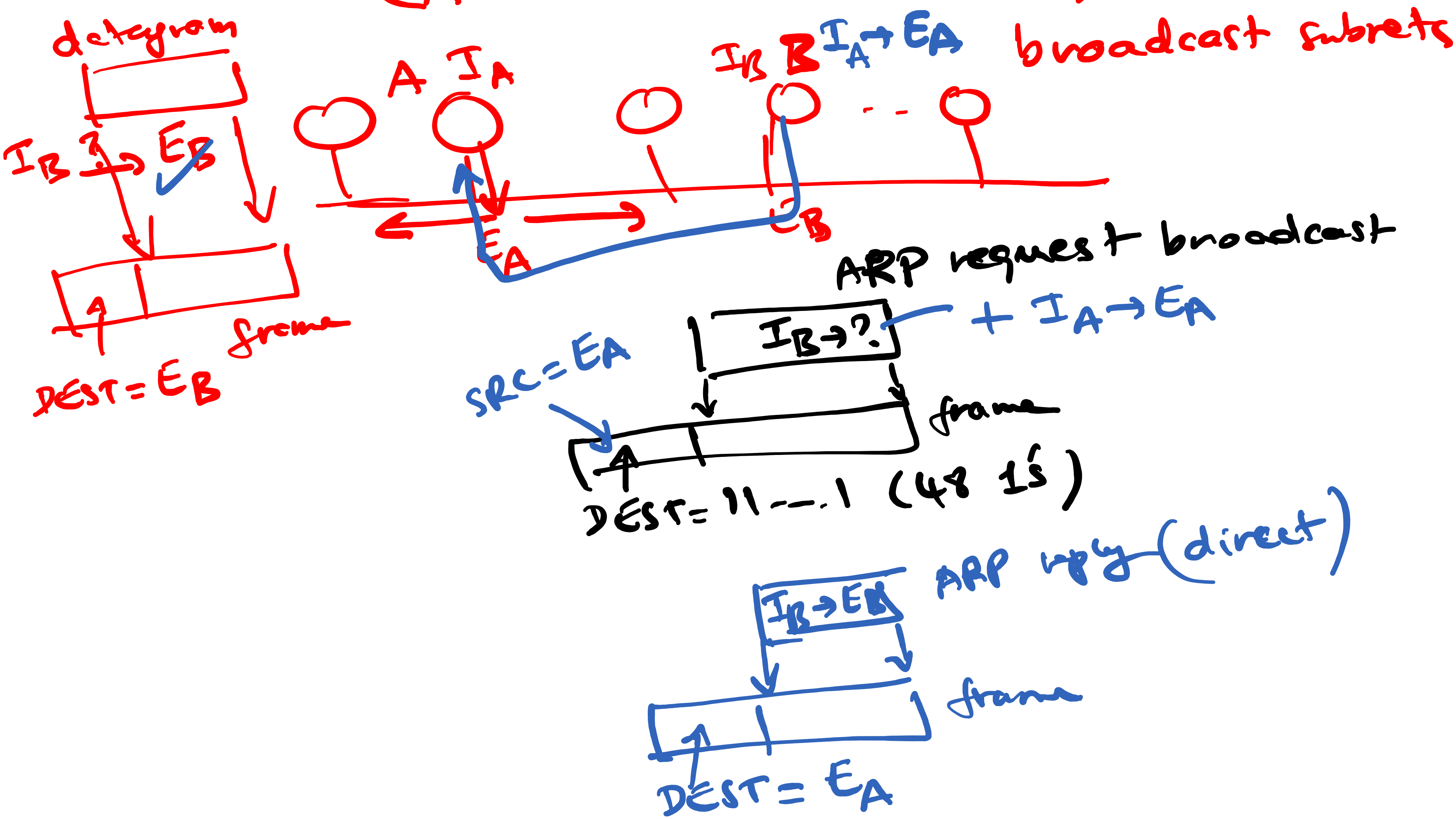
v4

← 32 bits →

v6 128 bits ✓

② Table_based dynamically maintained

ARP (Address Resolution Protocol)



Address Translation (cont'd)

- Techniques
  - encode physical address in host part of IP address
  - table-based
- ARP (Address Resolution Protocol)
  - table of IP to physical address bindings
  - broadcast request if IP address not in table
  - target machine responds with its physical address
  - table entries are discarded if not refreshed

ARP

- ARP messages are encapsulated in physical frames
  - HardwareType: type of physical network (e.g., Ethernet)
  - ProtocolType: type of higher layer protocol (e.g., IP)
  - HLEN & PLEN: length of physical and protocol addresses
  - Operation: request or response
  - Source/Target Physical/Protocol addresses

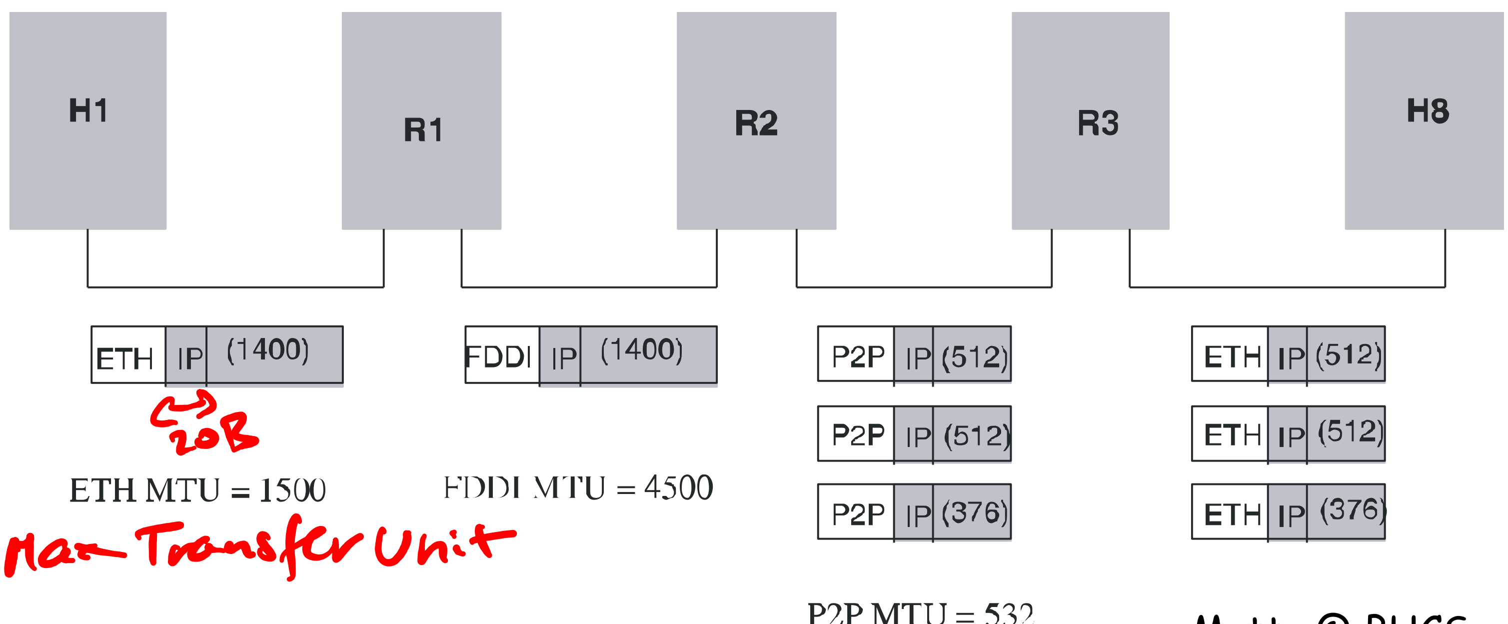
<table>
<tr><td colspan="2">HardwareType=1</td><td>ProtocolType=0x0800</td></tr>
<tr><td>HLEN=48</td><td>PLEN=32</td><td>Operation</td></tr>
<tr><td colspan="3">SourceHardwareAddr</td></tr>
<tr><td colspan="2">SourceHardwareAddr</td><td>SourceProtocolAddr</td></tr>
<tr><td colspan="2">SourceProtocolAddr</td><td>TargetHardwareAddr</td></tr>
<tr><td colspan="3">TargetHardwareAddr</td></tr>
<tr><td colspan="3">TargetProtocolAddr</td></tr>
</table>

- Notes
  - table entries timeout in about 10-20 minutes
  - update table with source when you are the target
  - refresh entry if already have an entry for source
  - do not refresh table entries upon reference

Fragmentation and Reassembly

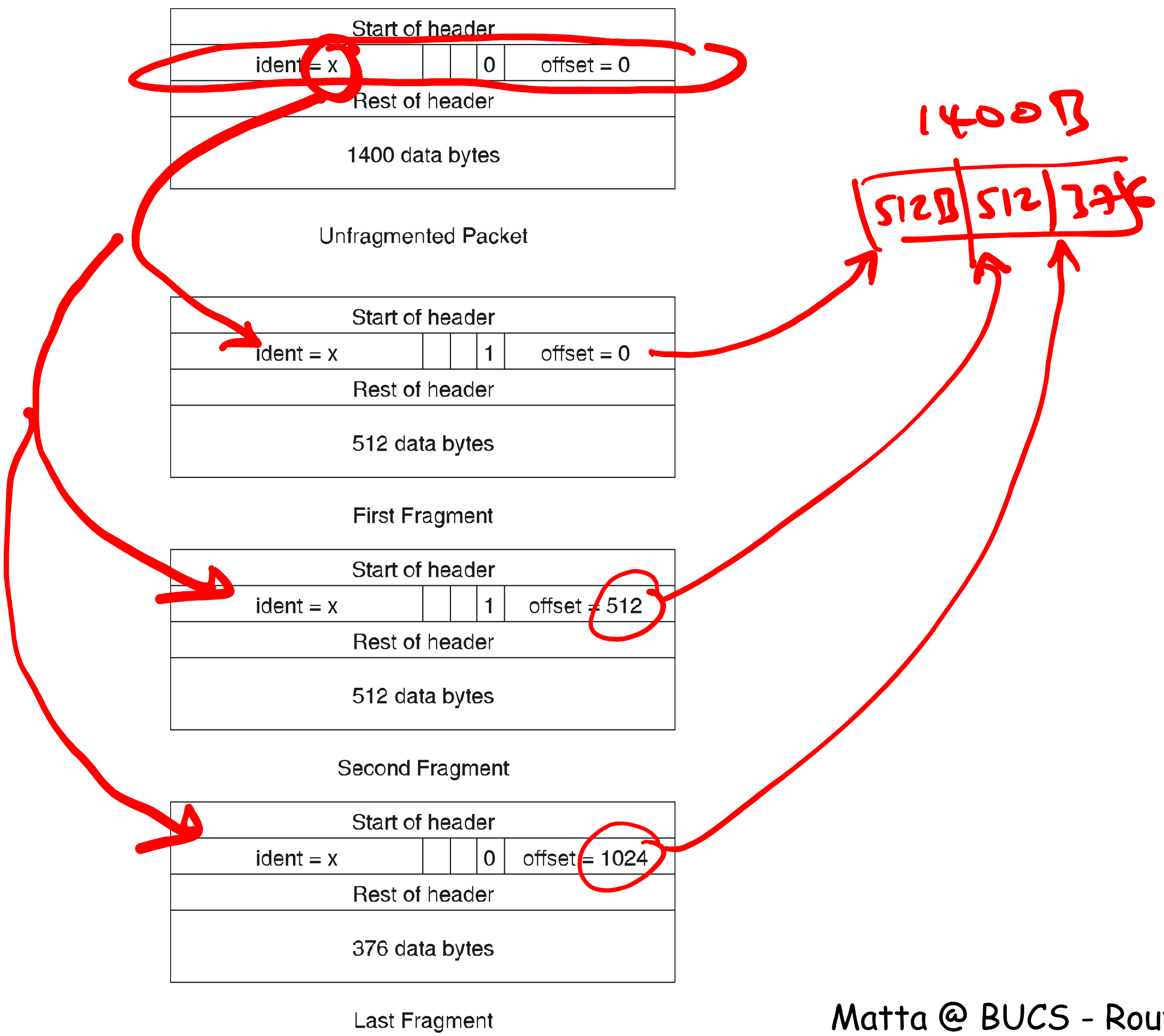
- Each network has some MTU
- Strategy
  - fragment when necessary (MTU < Datagram Size) $= 1420B$
  - try to avoid fragmentation at source host
  - re-fragmentation is possible
  - fragments are self-contained datagrams
  - delay reassembly until destination host ←
  - do not recover from lost fragments

$$\left\lceil \frac{1400B}{512B} \right\rceil = 3 \text{ fragments}$$



Max Transfer Unit

Fragmentation Example



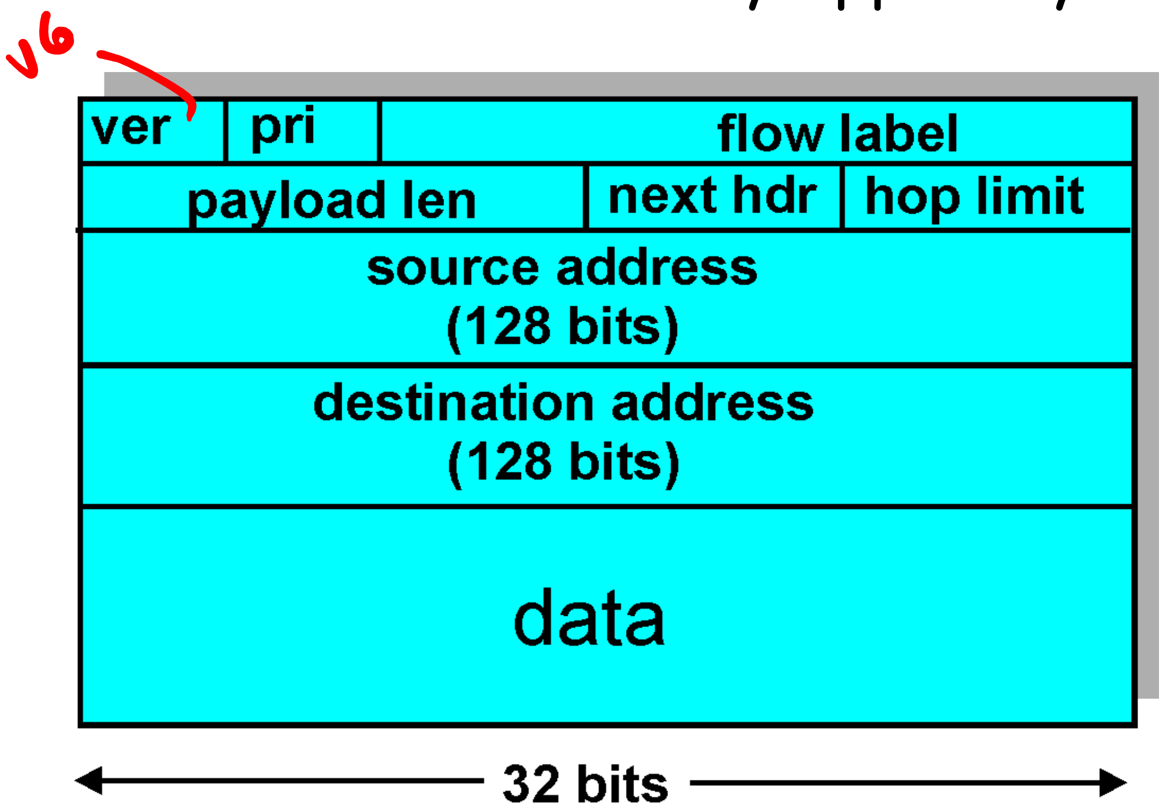
IPv6

- Initial motivation: 32-bit address space completely allocated
- 128-bit IPv6 address length gives us more than 10^{28} times as many IPv4 addresses
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

  IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header (Cont)

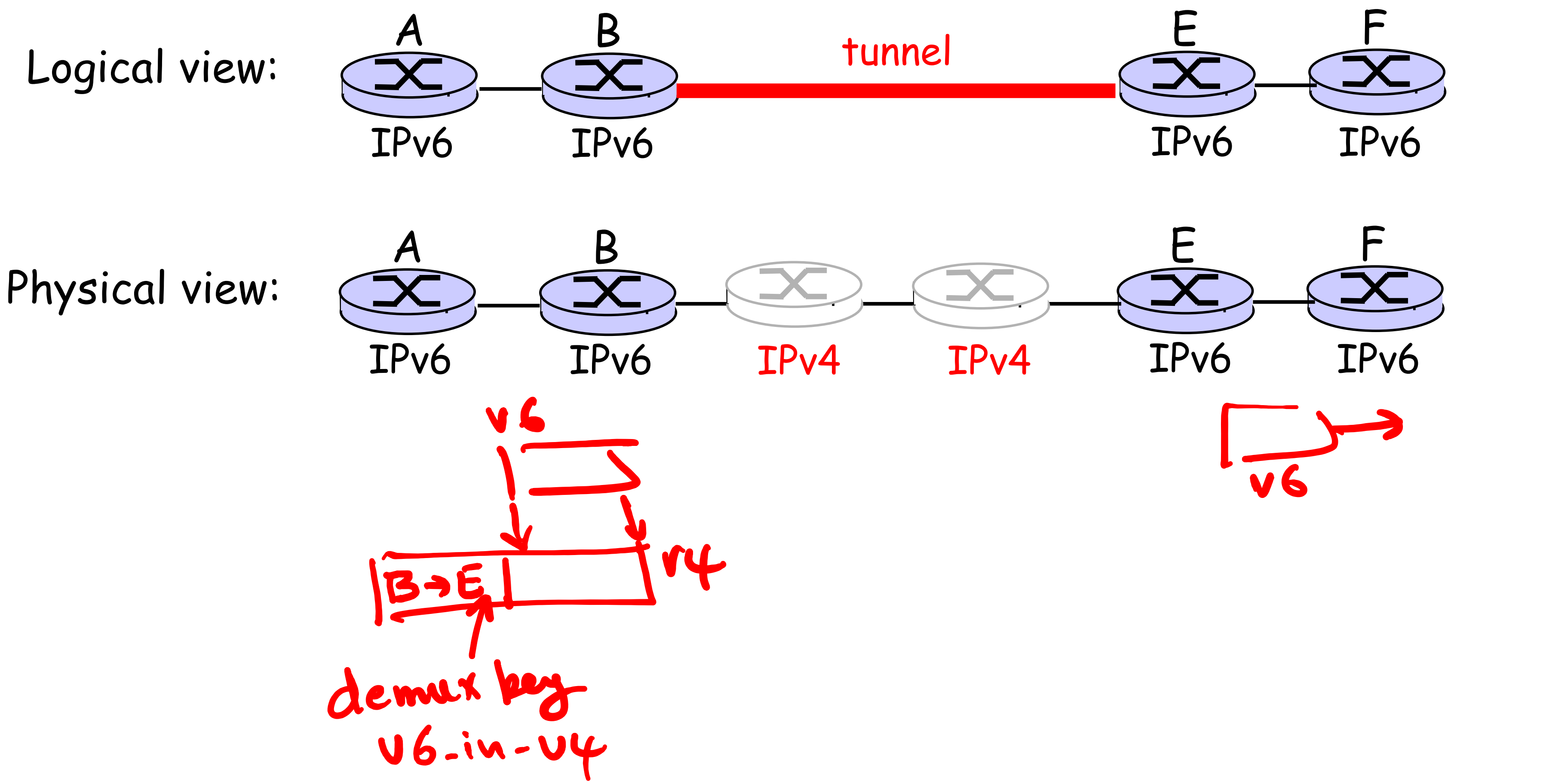
Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in same "flow"
(concept of "flow" not well defined)
Next header: identify upper layer protocol for data



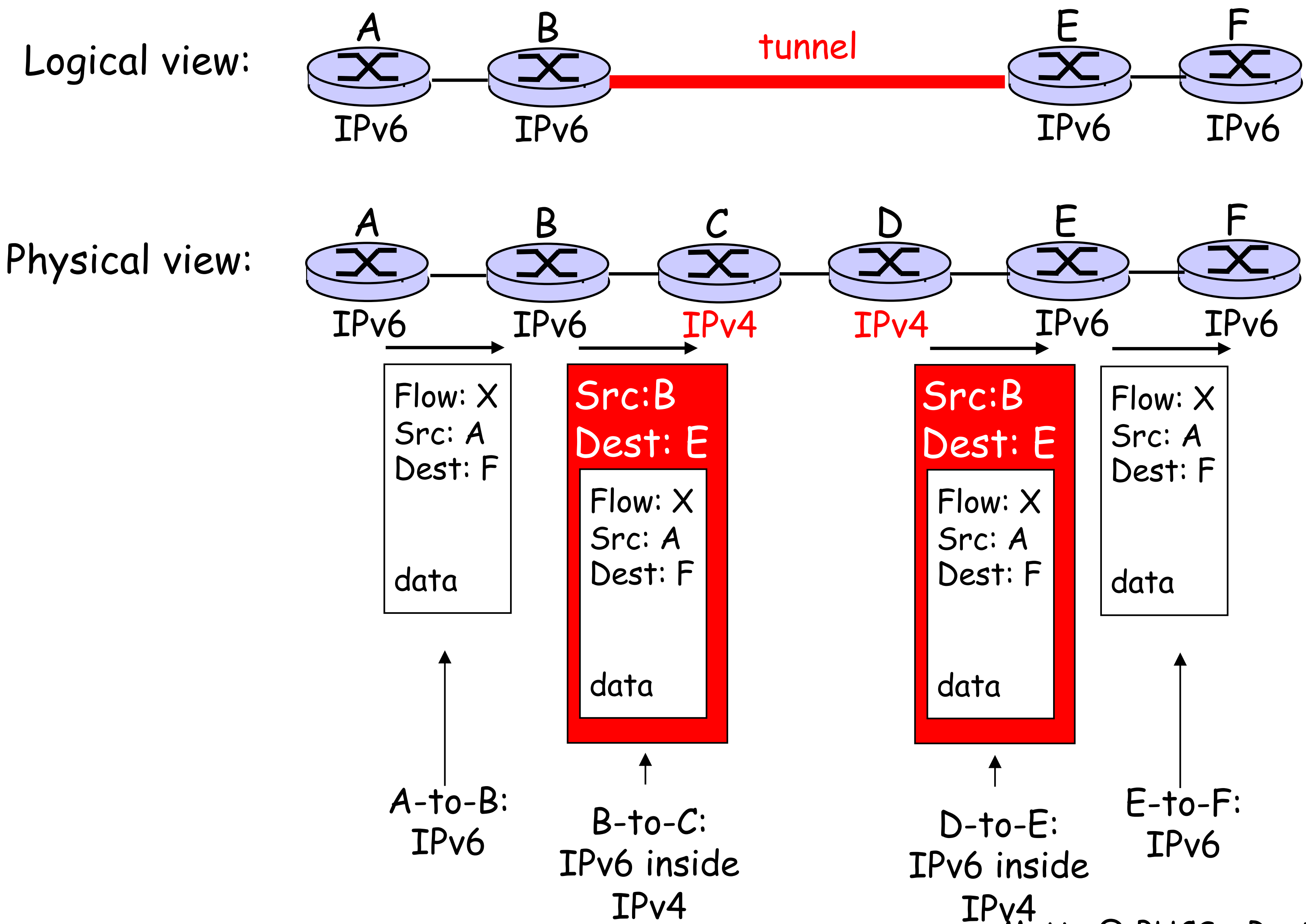
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no "flag day"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

Tunneling

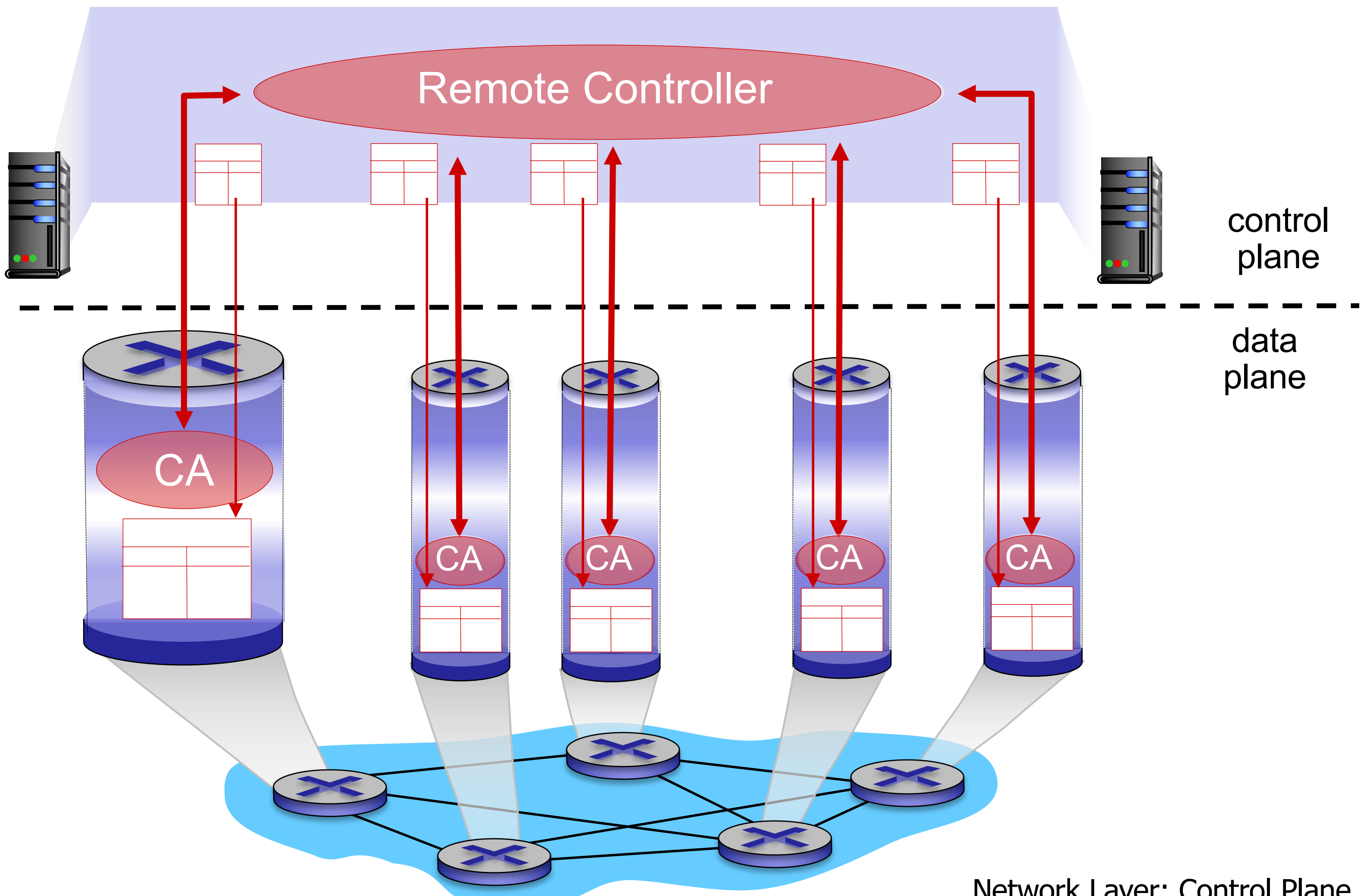

Logical view:
A
B
E
F
tunnel
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
v6
v4
B→E
demux key
v6-in-v4
v6

Tunneling

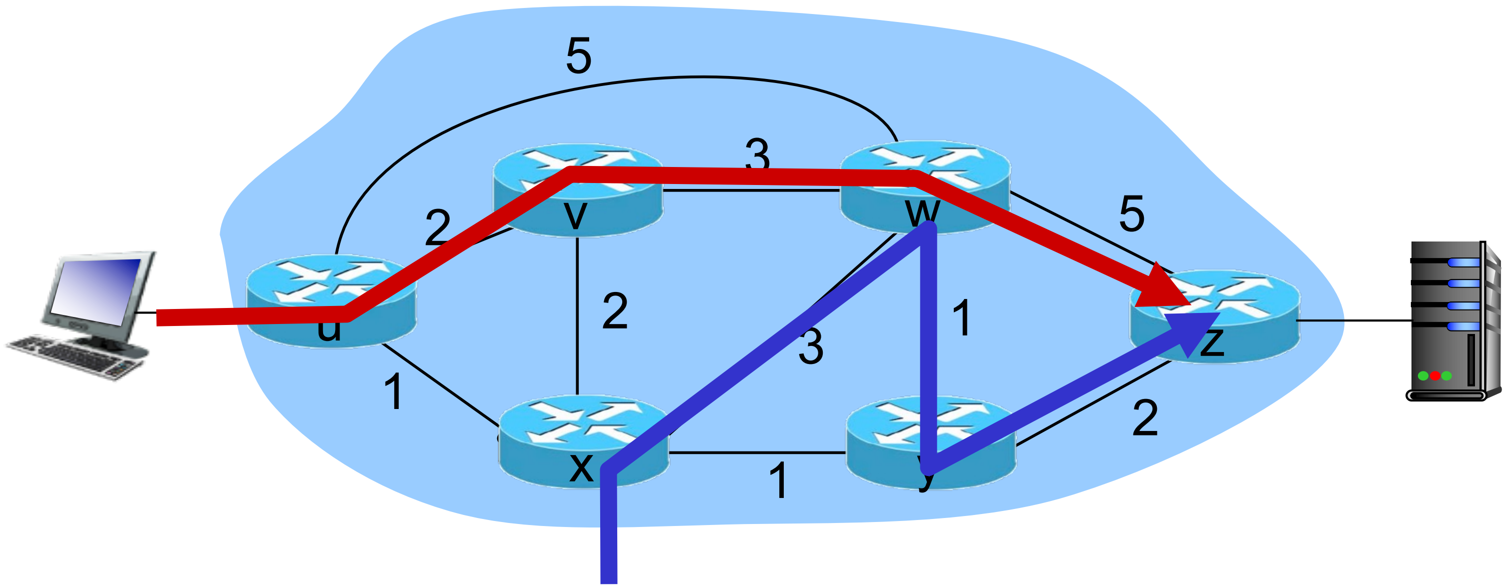

Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
D-to-E:
IPv6 inside
IPv4
E-to-F:
IPv6

Generalized Forwarding and SDN

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



Traffic engineering: difficult



Q: what if w wants to route blue and red traffic differently?

A: can't do it (with destination based forwarding, and LS, DV routing)

OpenFlow: Flow Table Entries

Rule	Action	Stats

Packet + byte counters

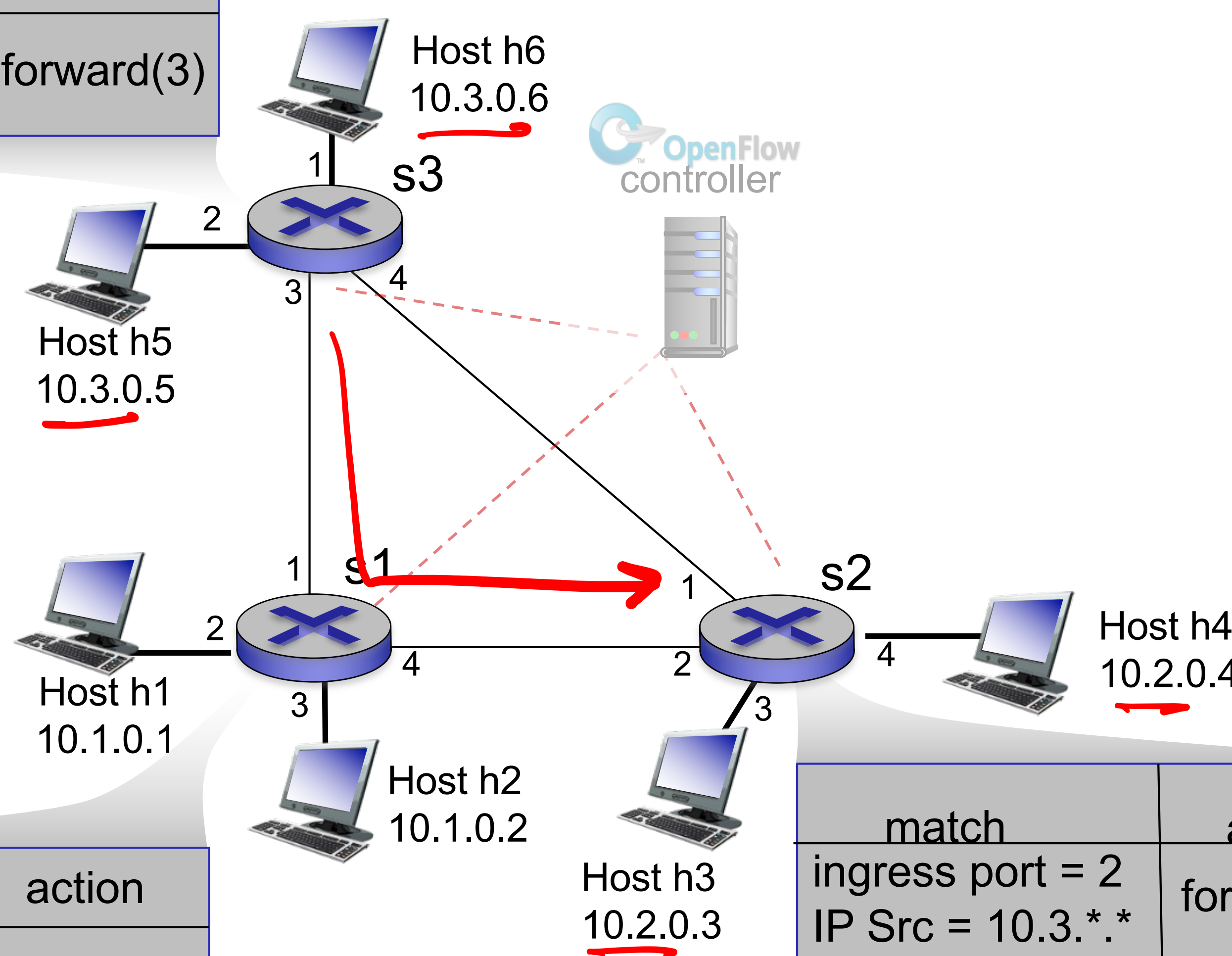
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
	Link layer				Network layer			Transport layer	

OpenFlow example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Src = 10.3.*.* IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Src = 10.3.*.* IP Dst = 10.2.0.4	forward(4)