# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

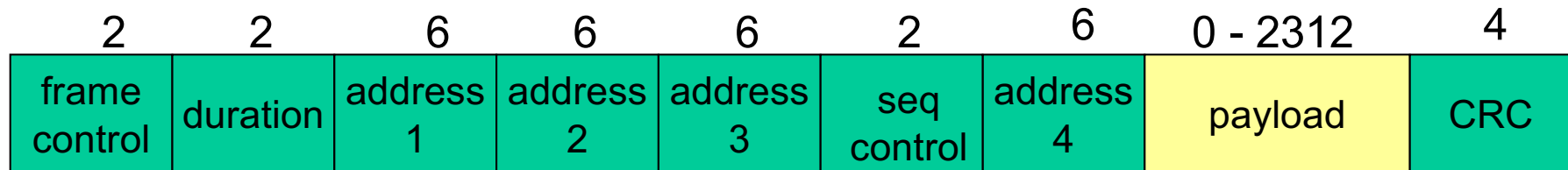6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request

# Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

| preamble | dest. address | source address | type | data (payload) | CRC |
|----------|--------------|----------------|------|----------------|-----|

# [802.11 frame structure: addressing]

| 2 | 2 | 6 | 6 | 6 | 2 | 6 | 0 - 2312 | 4 |
|---|---|---|---|---|---|---|---|---|
| frame control | duration | address 1 | address 2 | address 3 | seq control | address 4 | payload | CRC |

**Address 4:** used only in ad hoc mode

**Address 1:** MAC address of wireless host or AP to receive this frame (destination)

**Address 2:** MAC address of wireless host or AP transmitting this frame (source)

**Address 3:** MAC address of router interface to which AP is attached (first hop router; used for internetworking with wired LAN)
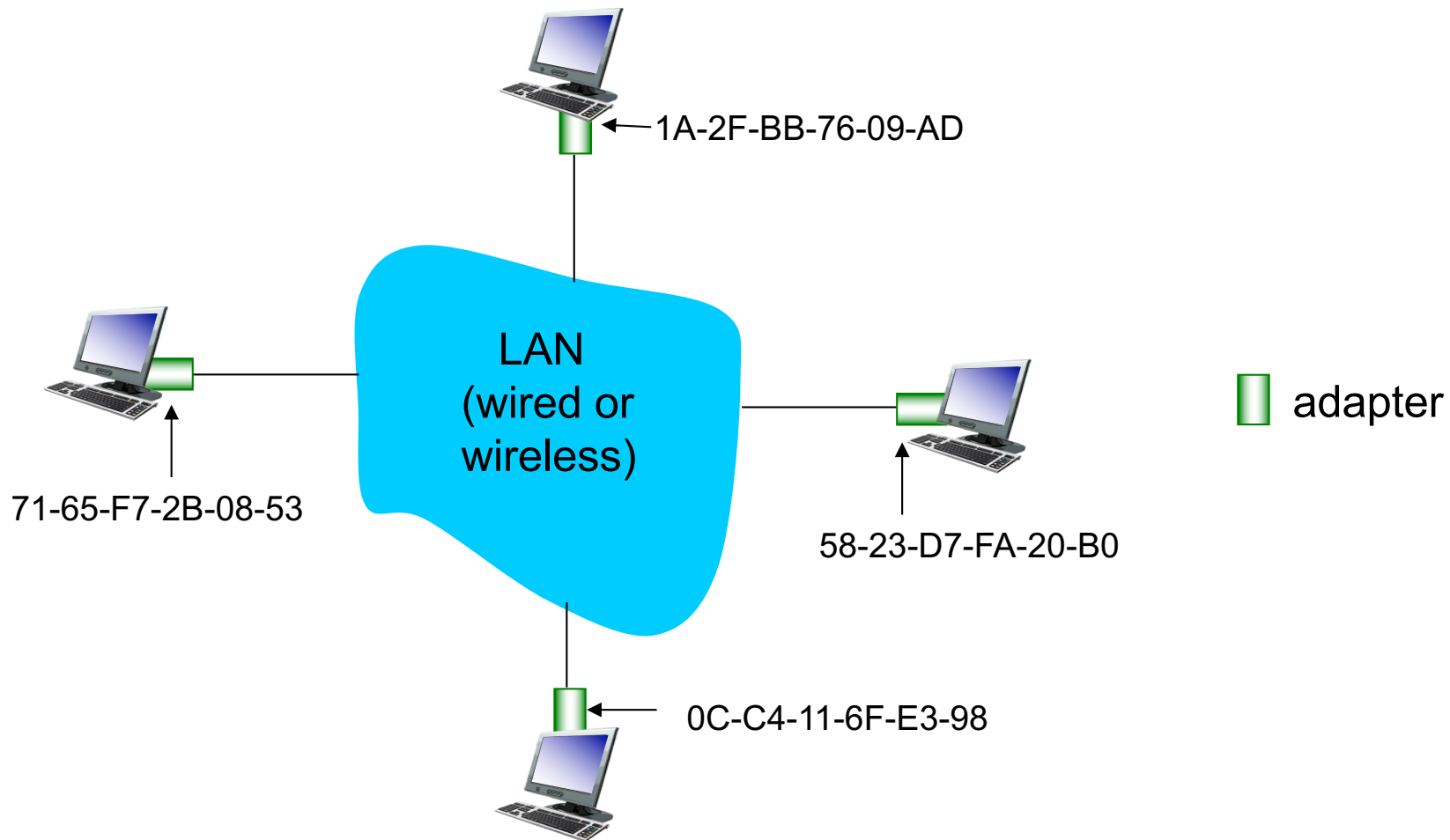
# MAC addresses and ARP

- ## 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- ## MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally'' to get frame from one interface to another physically-connected interface (same network, in IP-subnet/addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each "numeral" represents 4 bits)

# LAN addresses and ARP

each adapter on LAN has unique *LAN* address

1A-2F-BB-76-09-AD

LAN
(wired or
wireless)

adapter

71-65-F7-2B-08-53
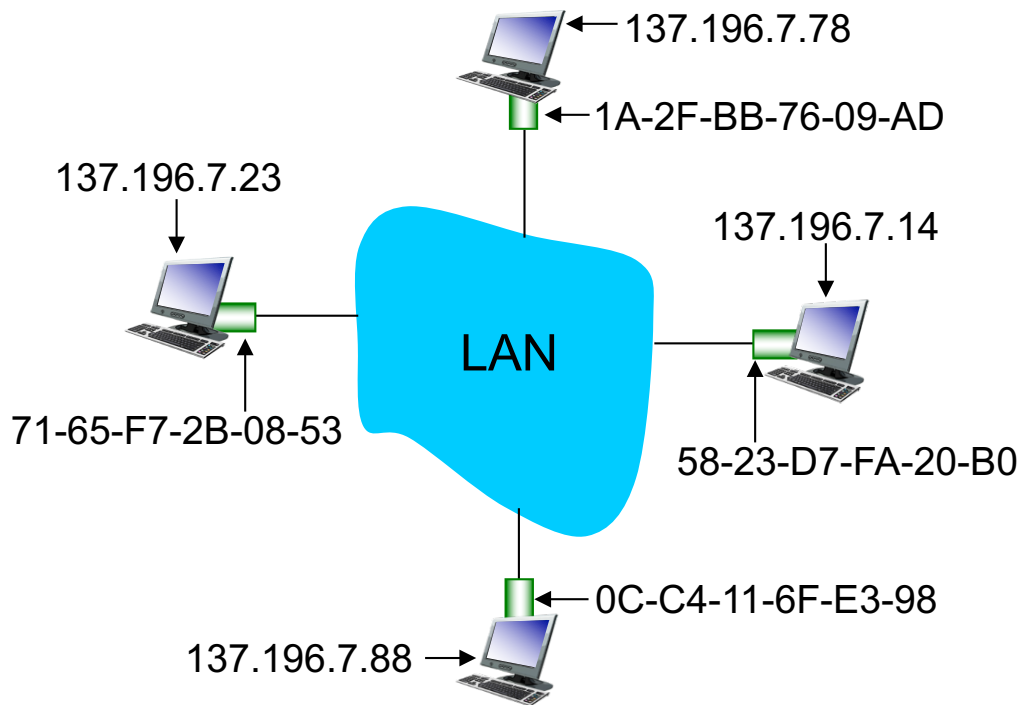
58-23-D7-FA-20-B0

0C-C4-11-6F-E3-98

# LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

# ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?

137.196.7.78

1A-2F-BB-76-09-AD

137.196.7.23

137.196.7.14

LAN

71-65-F7-2B-08-53

58-23-D7-FA-20-B0

0C-C4-11-6F-E3-98
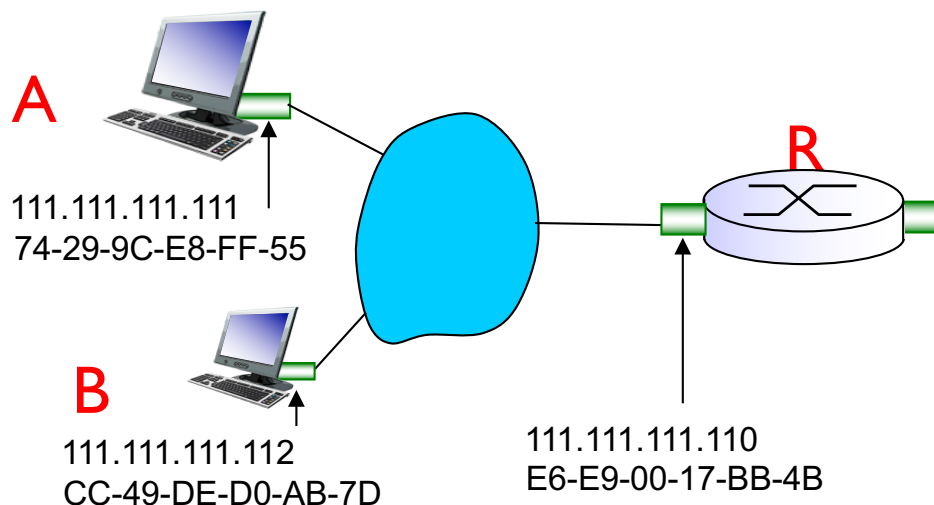
137.196.7.88

- **ARP table:** each IP node (host, router) **on same LAN** has table
  - IP/MAC address mappings for some LAN nodes:
    < IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

- Compare to DNS

# ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A **broadcasts** ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
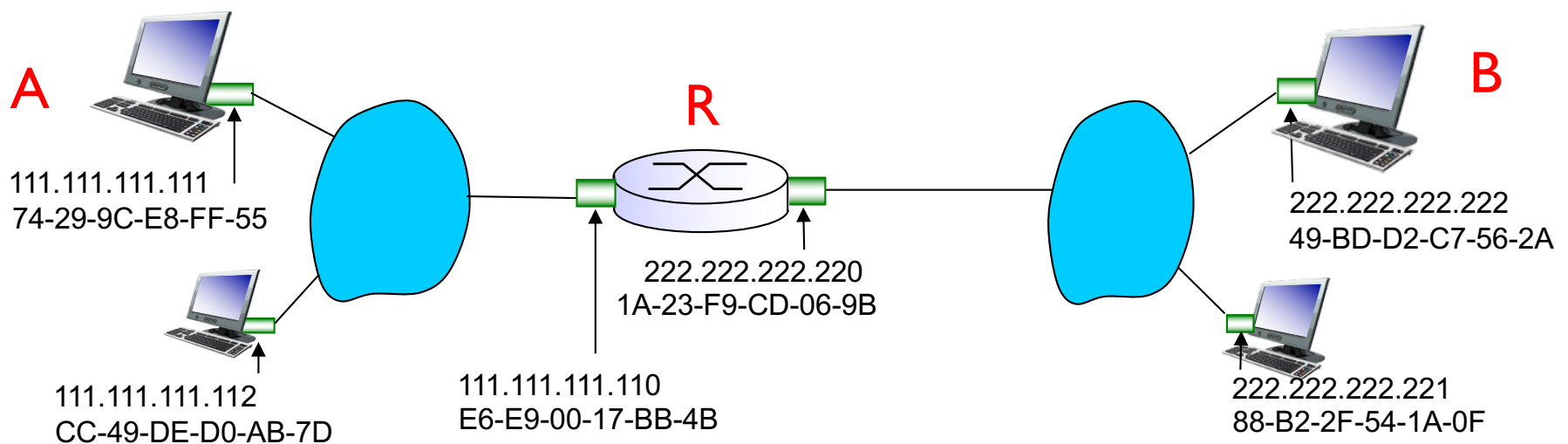  - frame sent to A's MAC address (unicast)

- A caches (saves) IP-to-MAC address pair in its ARP table until mapping times out
  - "soft state": information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*
- ARP is above Link Layer
  - ARP queries and repsonses encapsulated in Link Layer frames.

A
111.111.111.111
74-29-9C-E8-FF-55

B
111.111.111.112
CC-49-DE-D0-AB-7D

R

111.111.111.110
E6-E9-00-17-BB-4B
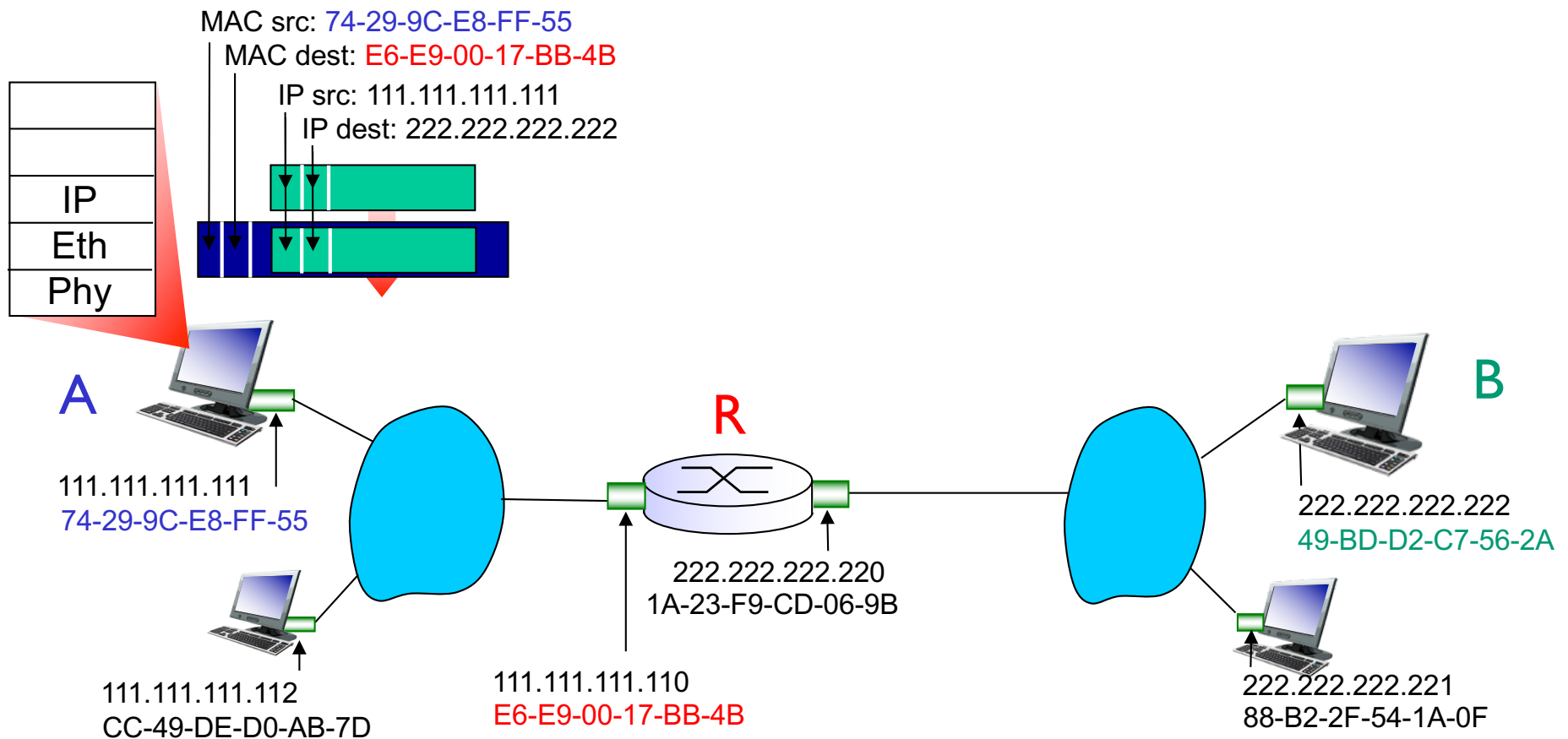
# Addressing: routing to another LAN

Send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- A doesn't know B's MAC; even if it did, it shouldn't use it!
- assume A knows IP address of first hop router, R (from DHCP)
- assume A knows R's MAC address (ARP)
- R runs two ARP modules, one per interface/LAN

A

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

R

222.222.222.220
1A-23-F9-CD-06-9B

111.111.111.110
E6-E9-00-17-BB-4B

B

222.222.222.222
49-BD-D2-C7-56-2A
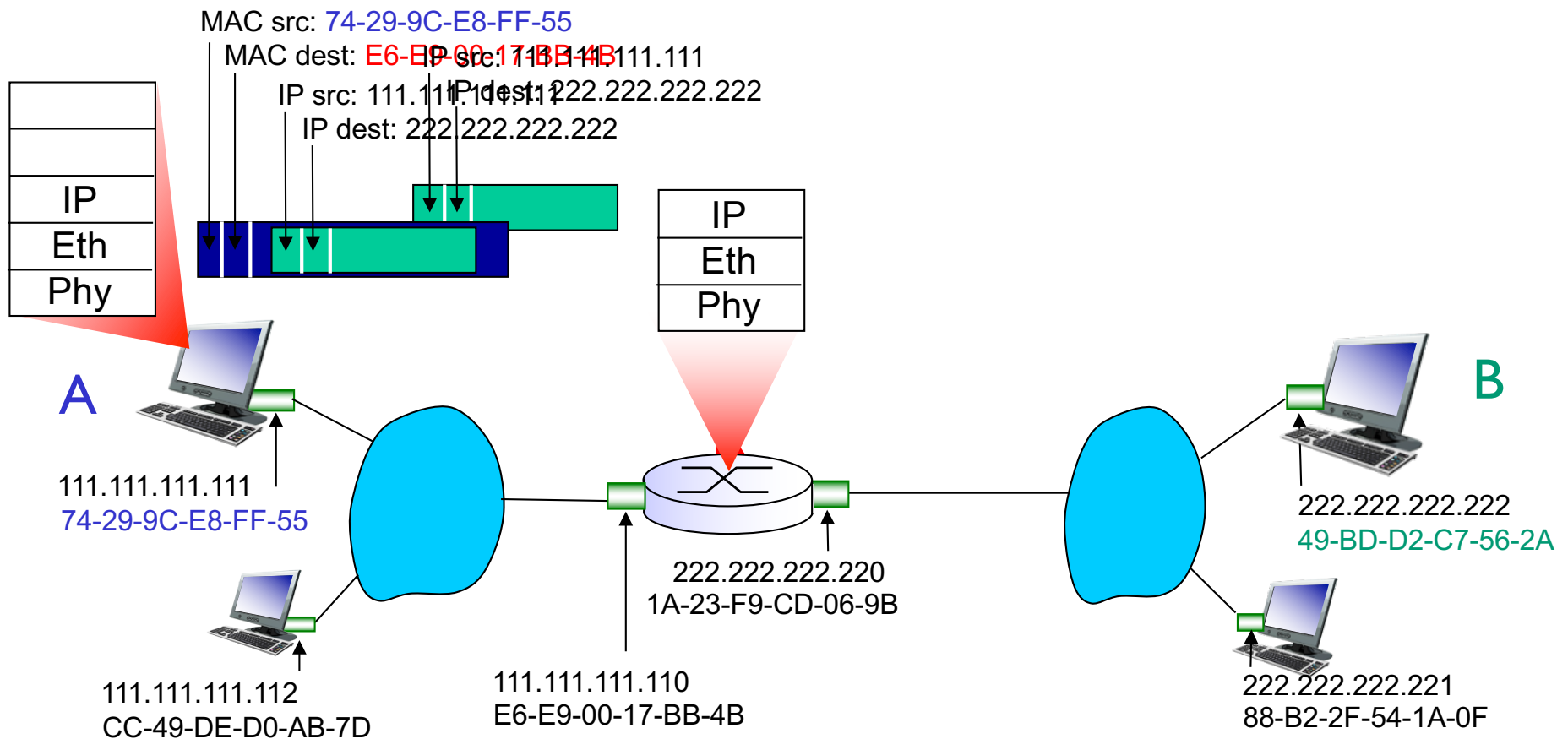
222.222.222.221
88-B2-2F-54-1A-0F

# Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B

- A creates link-layer frame with R's MAC address as destination address, frame contains A-to-B IP datagram
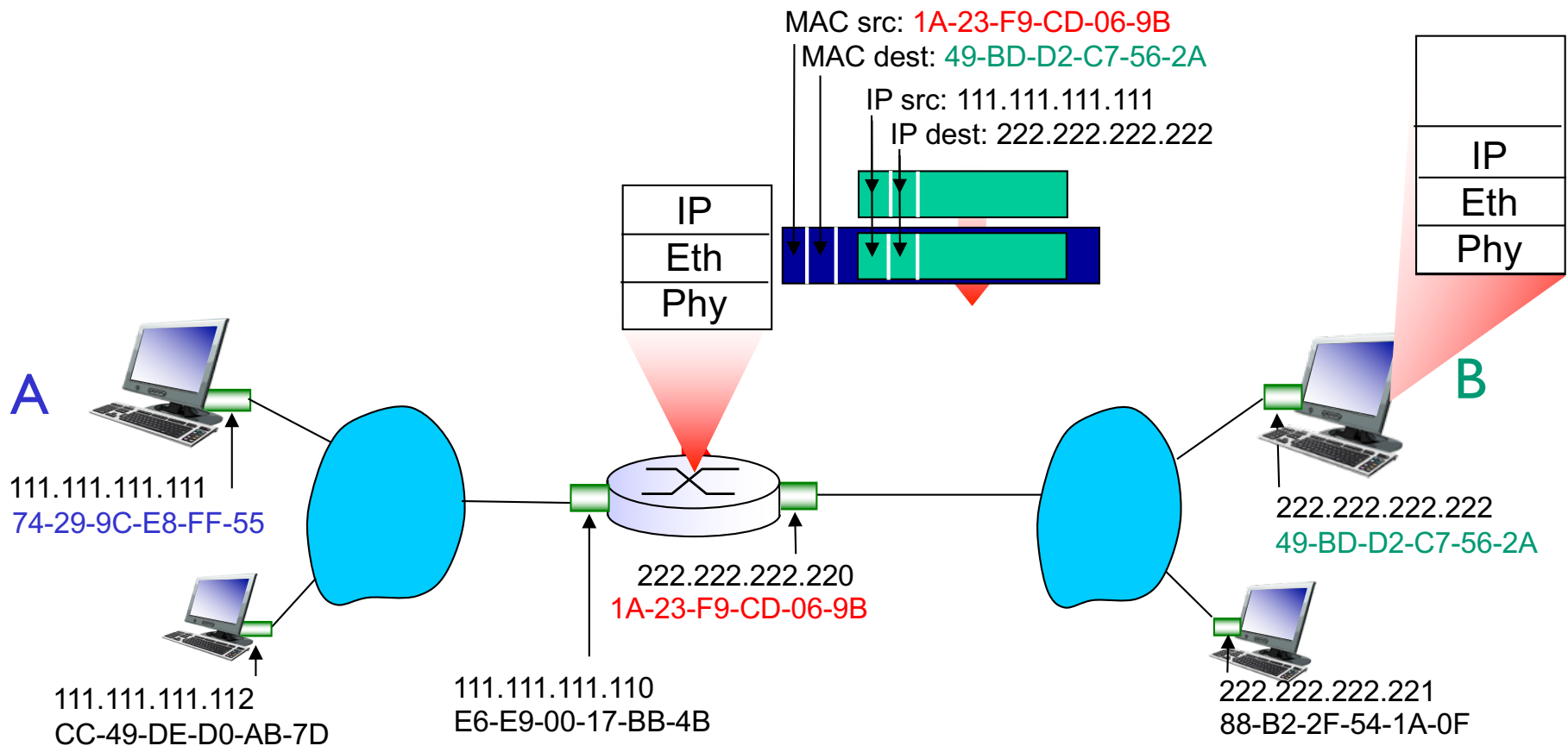
MAC src: 74-29-9C-E8-FF-55
MAC dest: E6-E9-00-17-BB-4B
IP src: 111.111.111.111
IP dest: 222.222.222.222

IP
Eth
Phy

A

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

R

222.222.222.220
1A-23-F9-CD-06-9B

111.111.111.110
E6-E9-00-17-BB-4B

B

222.222.222.222
49-BD-D2-C7-56-2A

222.222.222.221
88-B2-2F-54-1A-0F

# Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP

MAC src: 74-29-9C-E8-FF-55
MAC dest: E6-E9-00-17-BB-4B
IP src: 111.111.111.111
IP dest: 222.222.222.222

IP
Eth
Phy

IP
Eth
Phy

A

B

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

111.111.111.110
E6-E9-00-17-BB-4B

222.222.222.220
1A-23-F9-CD-06-9B

222.222.222.222
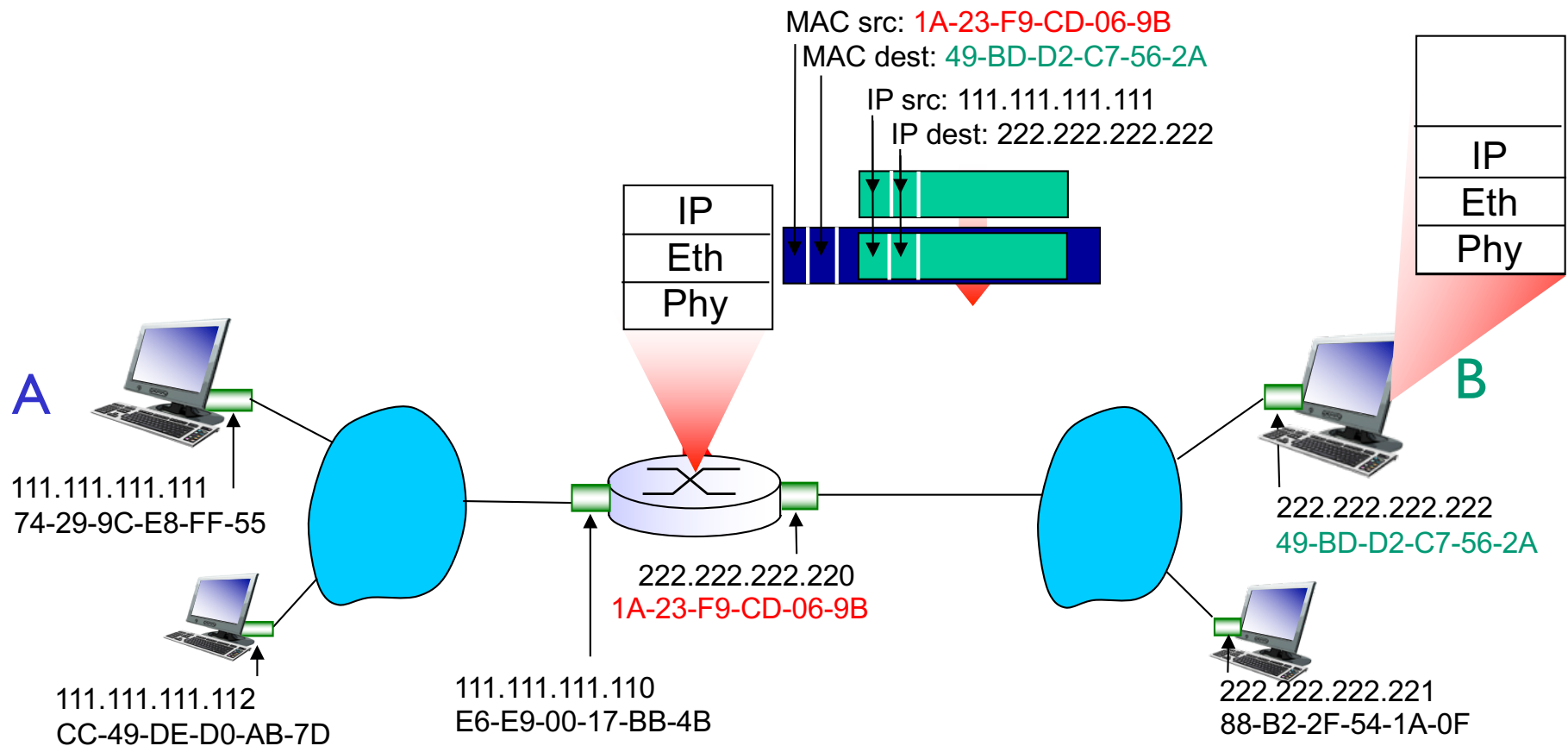49-BD-D2-C7-56-2A
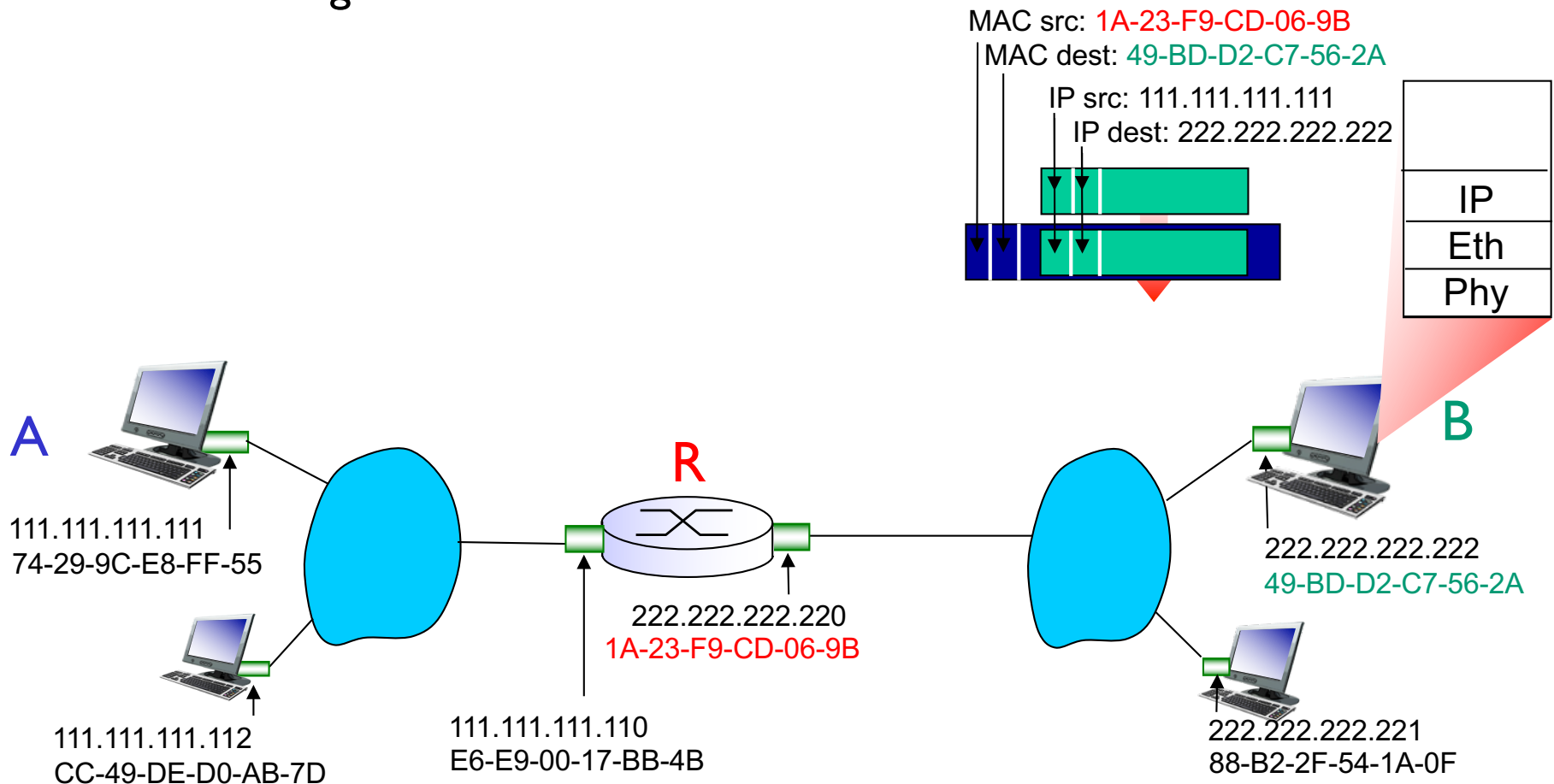
222.222.222.221
88-B2-2F-54-1A-0F

# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram

MAC src: 1A-23-F9-CD-06-9B
MAC dest: 49-BD-D2-C7-56-2A

IP src: 111.111.111.111
IP dest: 222.222.222.222

IP
Eth
Phy

IP
Eth
Phy

A

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

222.222.222.220
1A-23-F9-CD-06-9B

111.111.111.110
E6-E9-00-17-BB-4B

B

222.222.222.222
49-BD-D2-C7-56-2A

222.222.222.221
88-B2-2F-54-1A-0F

# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B

- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram

MAC src: 1A-23-F9-CD-06-9B
MAC dest: 49-BD-D2-C7-56-2A

IP src: 111.111.111.111
IP dest: 222.222.222.222

IP
Eth
Phy

IP
Eth
Phy

A

B

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

222.222.222.220
1A-23-F9-CD-06-9B

111.111.111.110
E6-E9-00-17-BB-4B

222.222.222.222
49-BD-D2-C7-56-2A

222.222.222.221
88-B2-2F-54-1A-0F

# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram

MAC src: 1A-23-F9-CD-06-9B
MAC dest: 49-BD-D2-C7-56-2A

IP src: 111.111.111.111
IP dest: 222.222.222.222

| IP |
| Eth |
| Phy |

A

111.111.111.111
74-29-9C-E8-FF-55

R

B

222.222.222.222
49-BD-D2-C7-56-2A

111.111.111.112
CC-49-DE-D0-AB-7D

222.222.222.220
1A-23-F9-CD-06-9B

111.111.111.110
E6-E9-00-17-BB-4B

222.222.222.221
88-B2-2F-54-1A-0F

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request
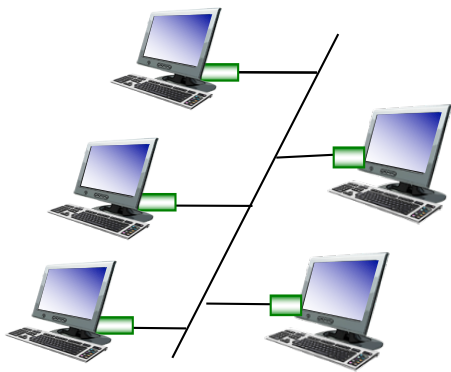
# Ethernet

"dominant" wired LAN technology:

- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
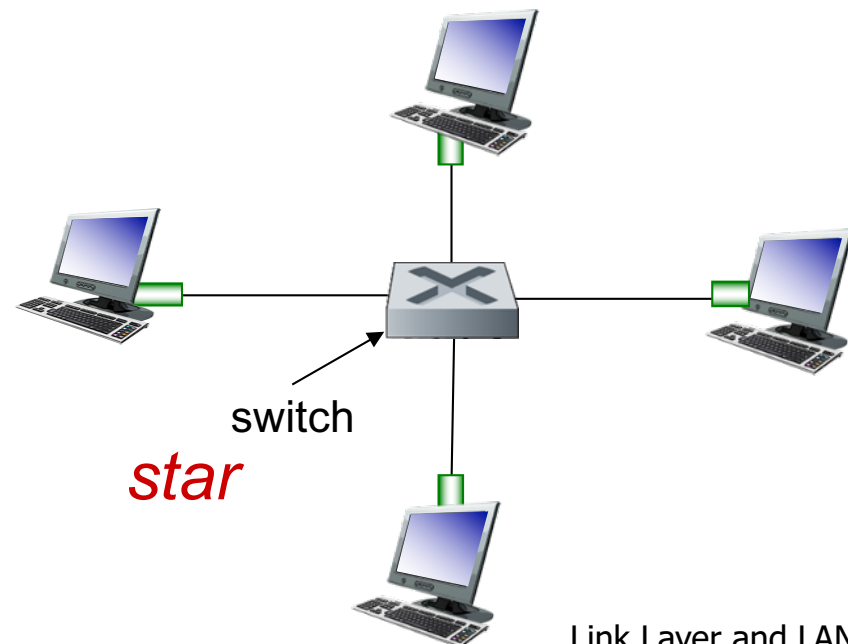- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps
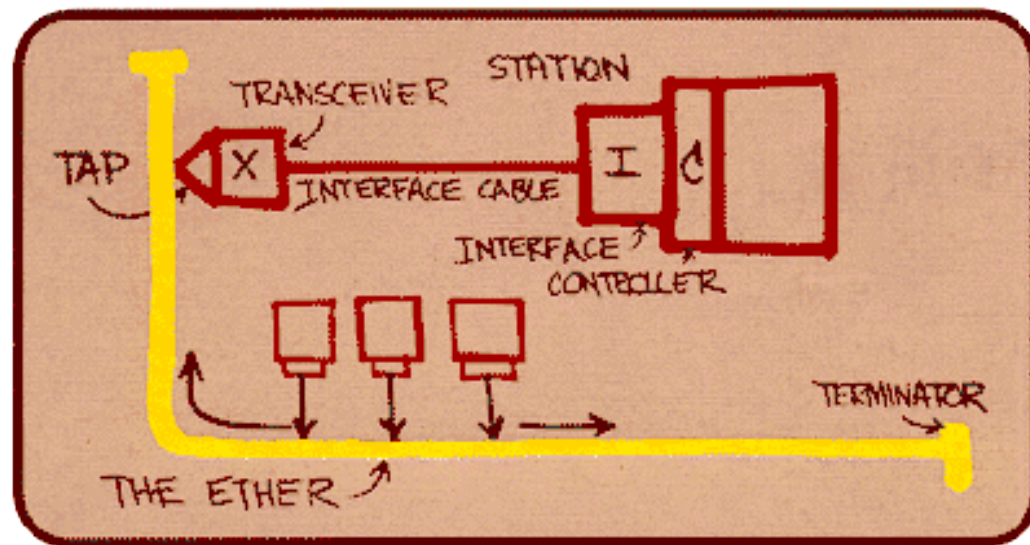


*Metcalfe's Ethernet sketch*

# Ethernet: physical topology

- *bus:* popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- *star:* prevails today
  - active *switch* in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)

*bus:* coaxial cable

switch

*star*

# [Ethernet frame structure]

sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

type

| preamble | dest. address | source address | | data (payload) | CRC |

*preamble:*

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

# [Ethernet frame structure (more)]

- *addresses:* 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- *type:* indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
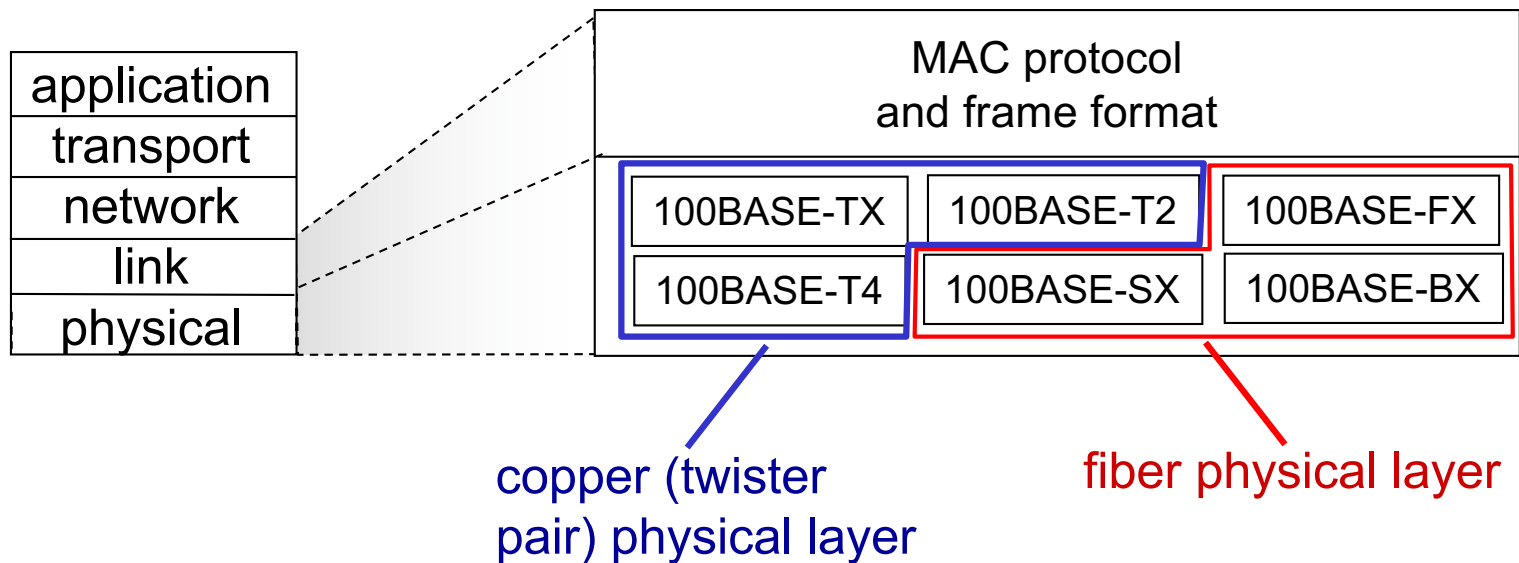- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped

type

| preamble | dest. address | source address | | data (payload) | CRC |

# [Ethernet: unreliable, connectionless]

- *connectionless:* no handshaking between sending and receiving NICs

- *unreliable:* receiving NIC doesn't send acks or nacks to sending NIC

  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost

- Ethernet's MAC protocol: unslotted *CSMA/CD with exponential backoff*

# 802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10 Gbps, 40 Gbps
  - different physical layer media: fiber, cable

| application |
|-------------|
| transport |
| network |
| link |
| physical |

| MAC protocol and frame format | | |
|---|---|---|
| 100BASE-TX | 100BASE-T2 | 100BASE-FX |
| 100BASE-T4 | 100BASE-SX | 100BASE-BX |

copper (twister pair) physical layer

fiber physical layer

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request

# Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

# Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch

- switches buffer packets

- Ethernet protocol used on *each* incoming link, but no collisions; full duplex

  - each link is its own collision domain

- *switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions
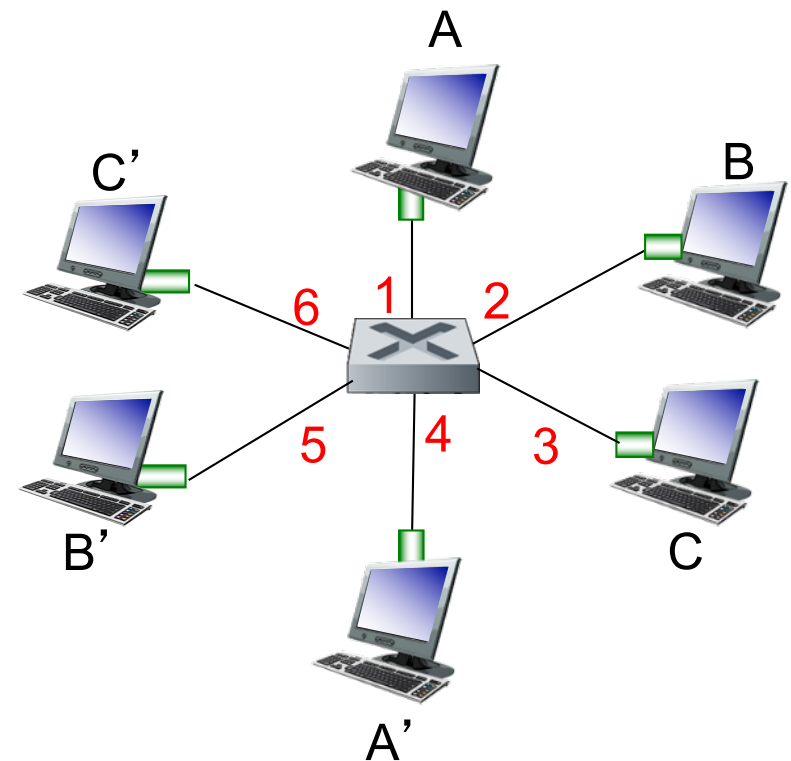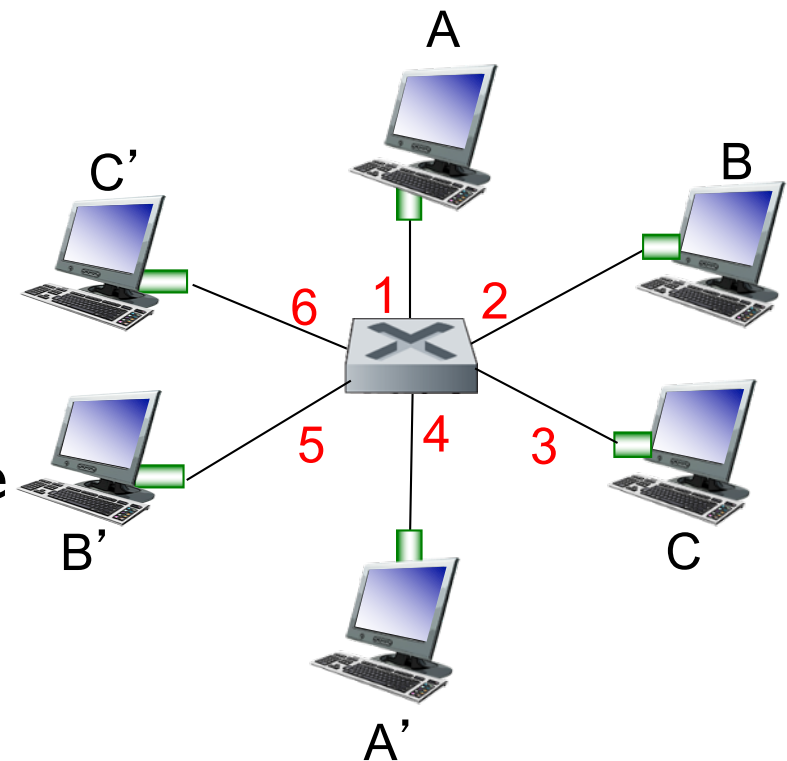
switch with six interfaces
(*1,2,3,4,5,6*)

# Switch forwarding table

*Q:* how does switch know A'
reachable via interface 4, B'
reachable via interface 5?

- *A:* each switch has a switch
  table, each entry:
  - (MAC address of host, interface
    to reach host, time stamp)
  - looks like a routing table!

*Q:* how are entries created,
maintained in switch table?
- Simpler than routing protocol?

*switch with six interfaces
(1,2,3,4,5,6)*

# Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
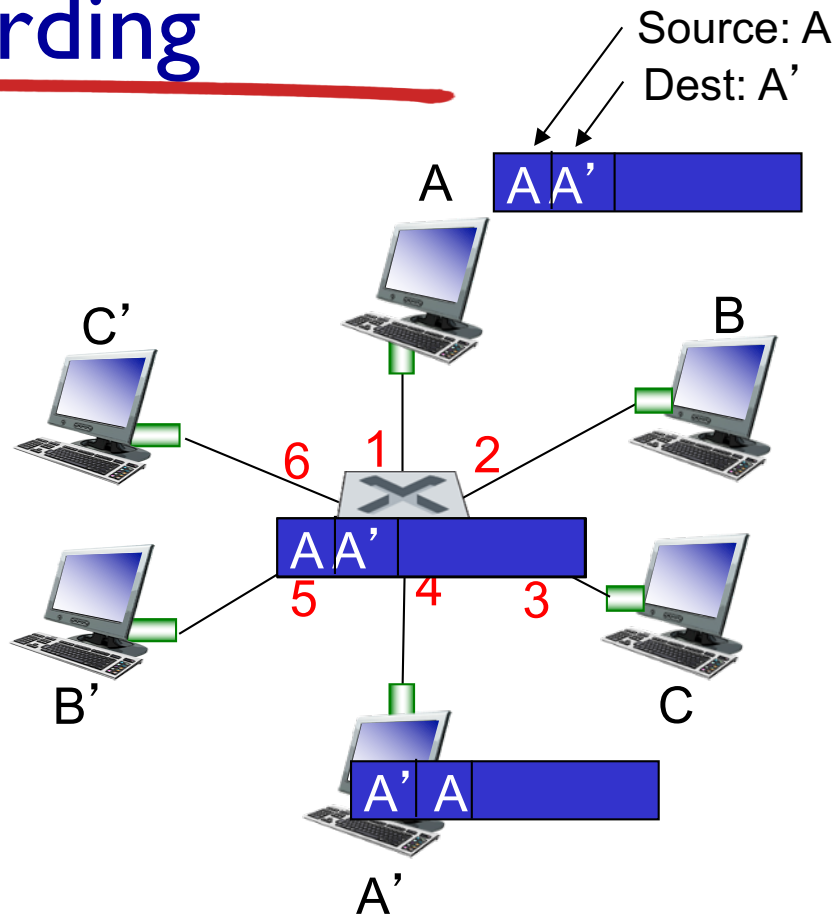  - records sender/location pair in switch table

Source: A
Dest: A'

A | A A'

A
B
C'
B'
C
A'

6  1  2
5  4  3
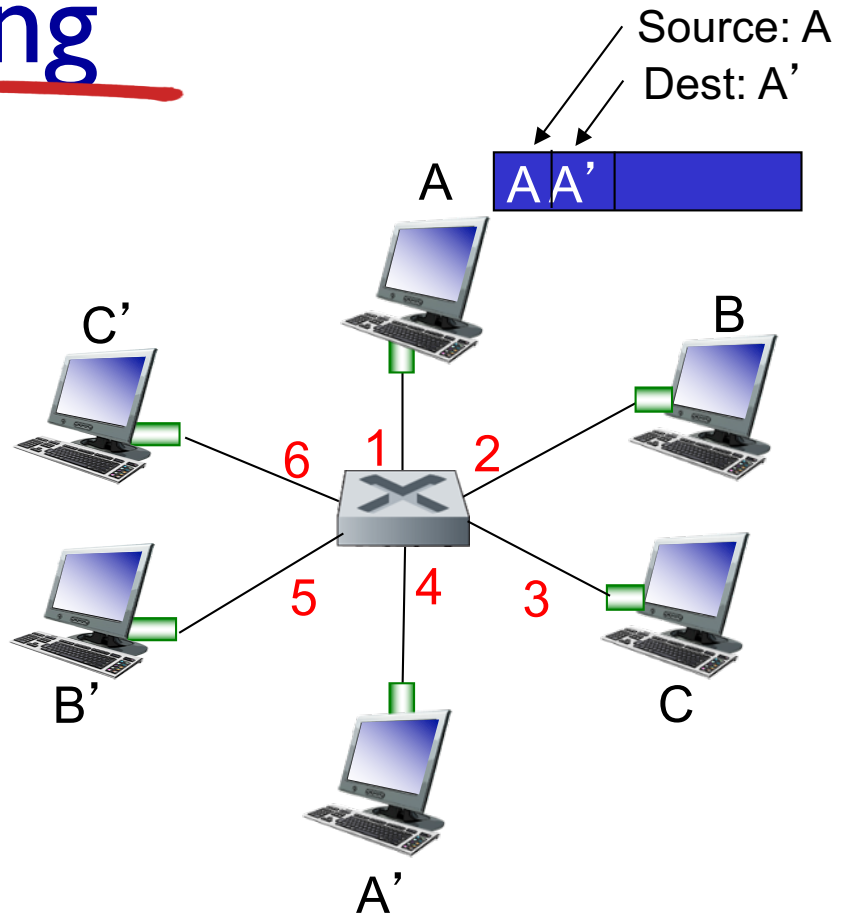
| MAC addr | interface | TTL |
|----------|-----------|-----|
| A | 1 | 60 |
| | | |

*Switch table (initially empty)*

# Switch: filtering, forwarding

Source: A
Dest: A'

- frame destination, A', location unknown: *flood*

- ❖ destination A location known:  selectively send on just one link

A  |A|A'| |

C'

B

6  1  2

|A|A'| |

5  4  3

B'

C

A' |A'|A| |

A'

| MAC addr | interface | TTL |
|----------|-----------|-----|
| A        | 1         | 60  |
| A'       | 4         | 60  |

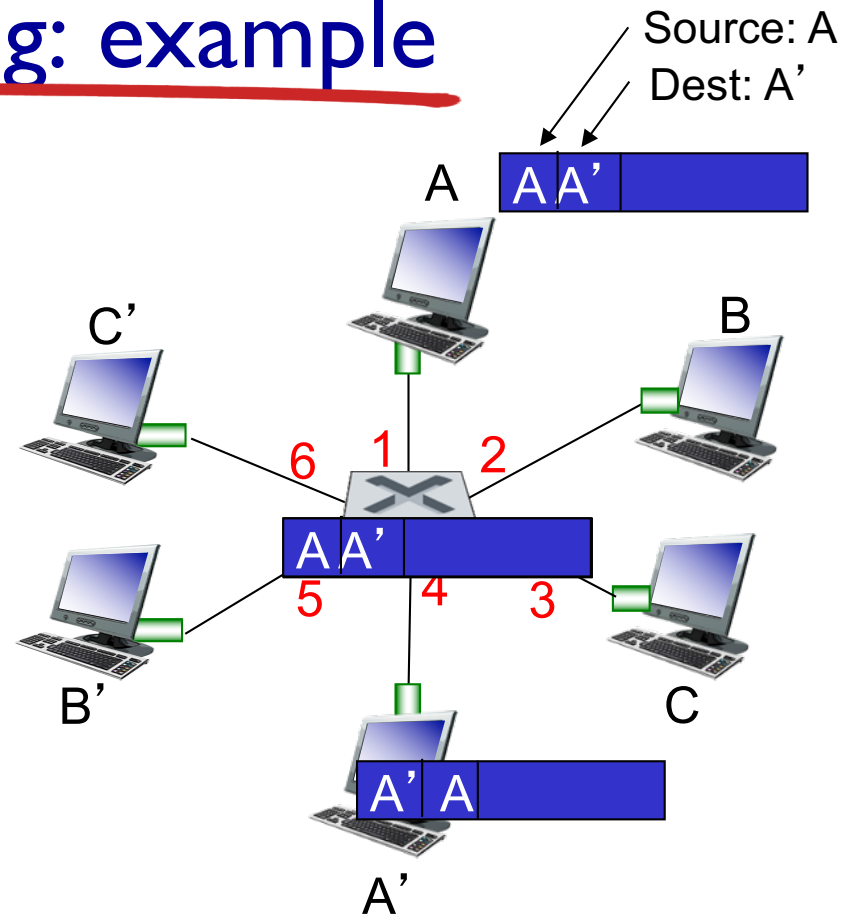*switch table (initially empty)*

# Switch: frame filtering/forwarding

when frame received at switch:

    1. record incoming link, MAC address of sending host
    2. index switch table using MAC destination address
    3. if entry found for destination
       then {
      if destination on segment from which frame arrived
         then drop frame
          else forward frame on interface indicated by entry
      }
      else flood  /* forward on all interfaces except arriving
                interface */

# Self-learning, forwarding: example

Source: A
Dest: A'

- frame destination, A', location unknown: *flood*

- destination A location known: selectively send on just one link

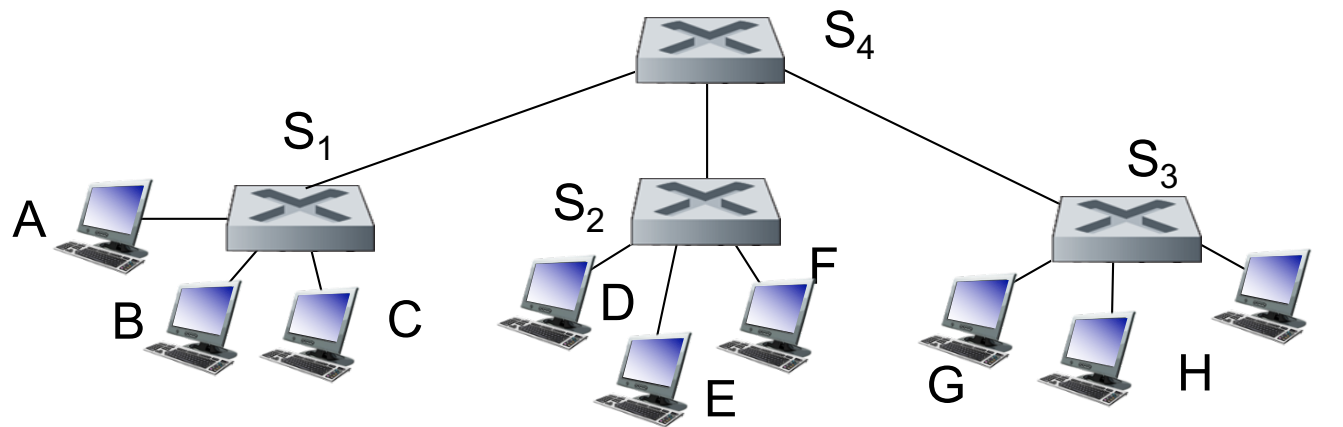| MAC addr | interface | TTL |
|----------|-----------|-----|
| A | 1 | 60 |
| A' | 4 | 60 |

*switch table (initially empty)*

# Interconnecting switches

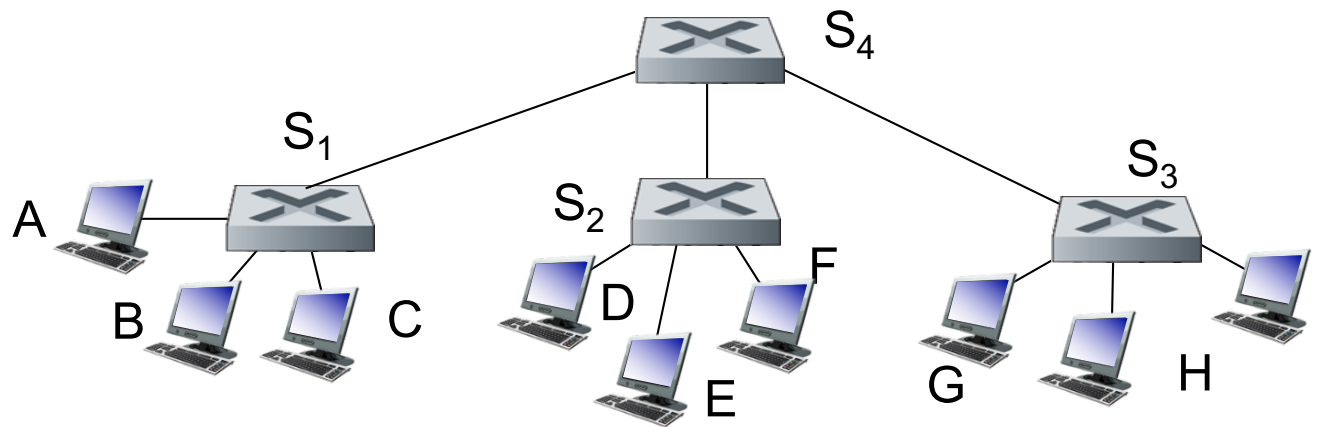self-learning switches can be connected together:



*Q:* sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

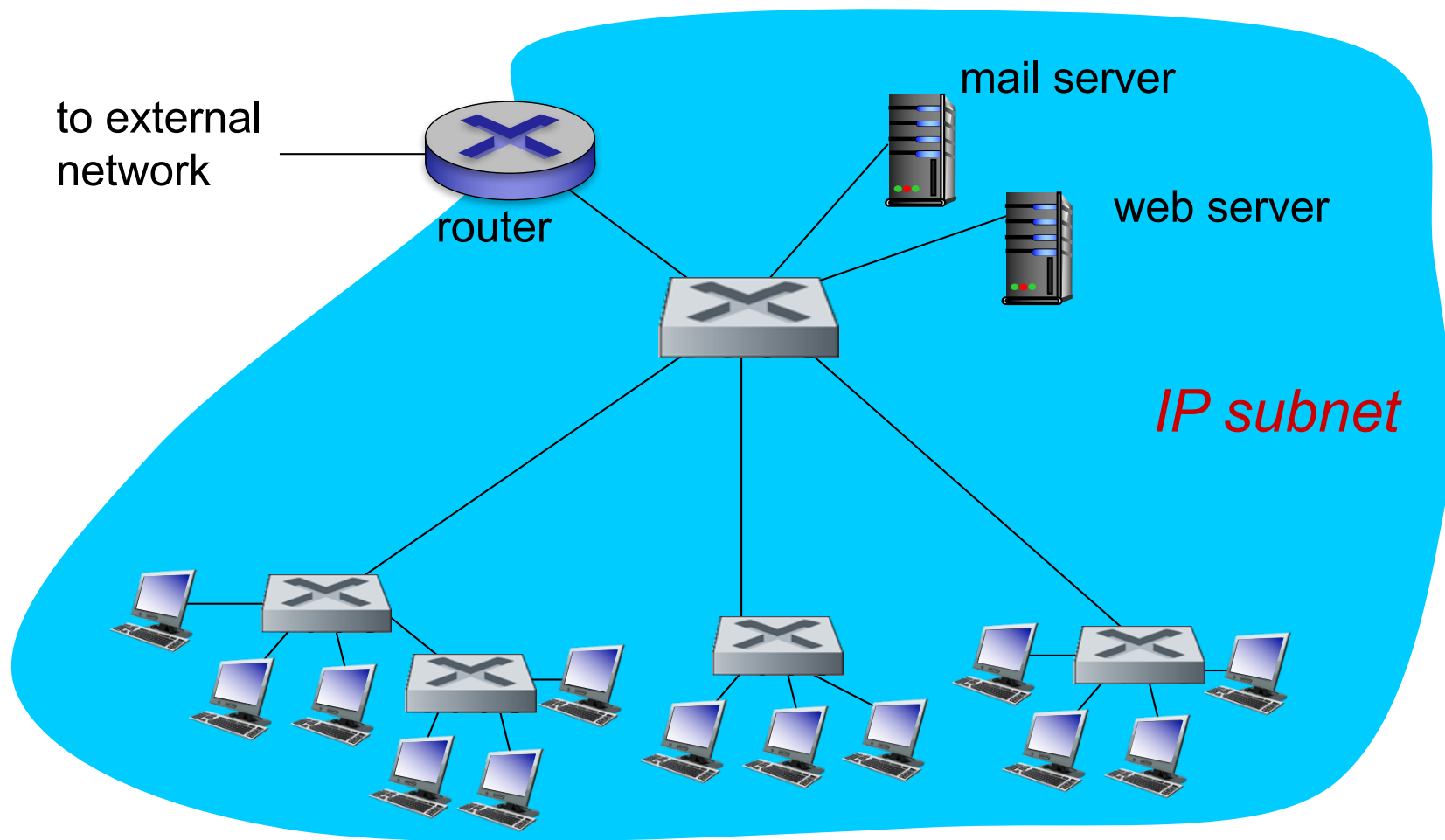- *A:* self learning! (works exactly the same as in single-switch case!)

# Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- Q: show switch tables and packet forwarding in $S_1$, $S_2$, $S_3$, $S_4$

# Institutional network



to external network

router

mail server

web server

*IP subnet*

Q: Why do we need hierarchy of switches?
- to handle large number of end hosts
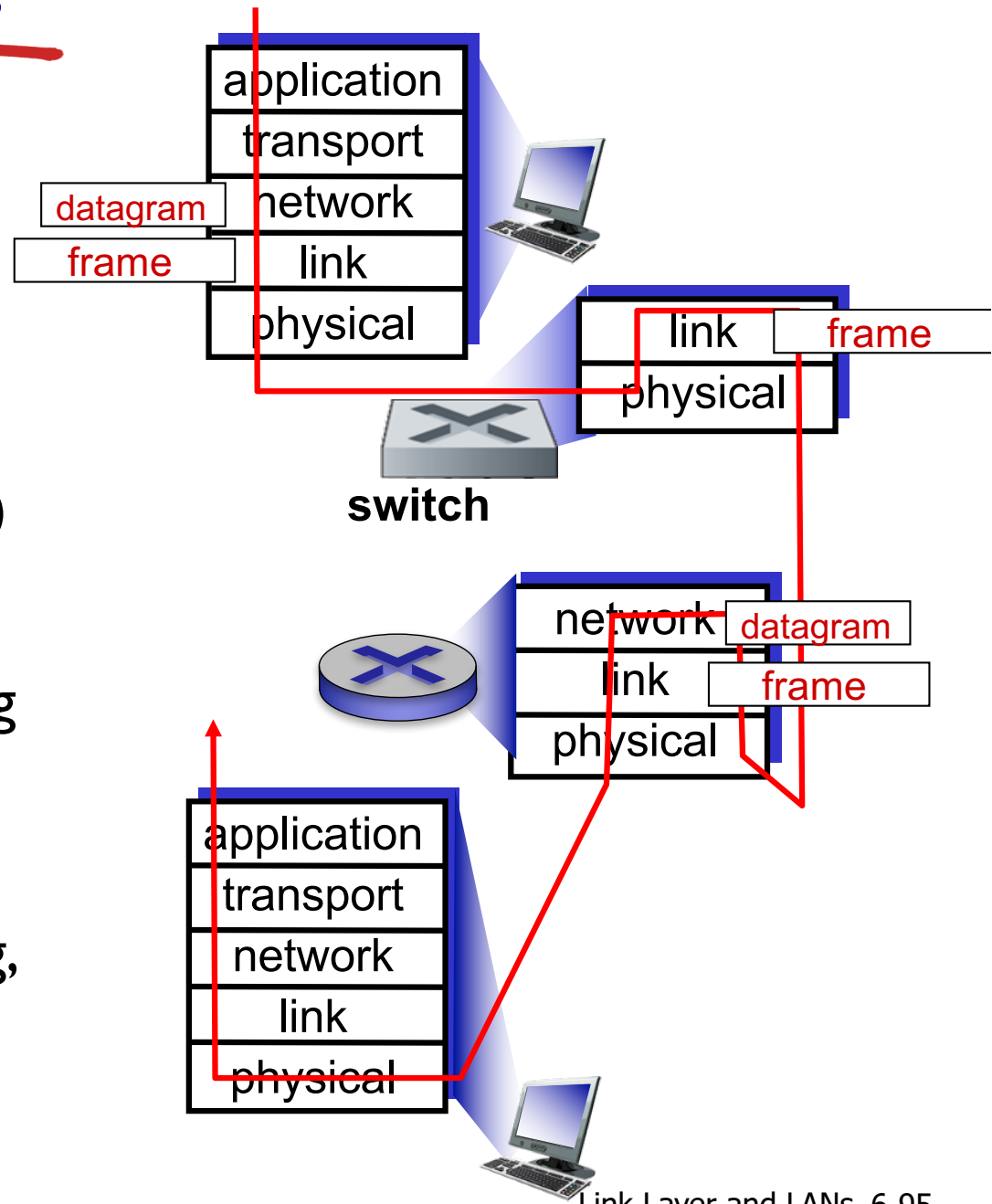- hierarchy of many small switches cheaper than vs. one big switch

# Switches vs. routers

**both are store-and-forward:**

- *routers:* network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

**both have forwarding tables:**

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses

application
transport
network
link
physical

datagram
frame

**switch**

link
physical

frame

network
link
physical

datagram
frame

application
transport
network
link
physical

# Switches vs. Routers Cont'd

- Switches
  - Switching table:
    - <dest MAC, outgoing link>
  - Self-learning
  - Layer-2
    - Fast processing in hw
  - Isolation via forwarding/filtering

  - Susceptible to broadcast storms (e.g. ARP)
    - Topology: spanning tree

  - # End-hosts: up to100s

- Routers
  - Forwarding Table
    - <dest IP, outgoing link>
  - Require configuration
  - Layer-3
    - Slower processing in sw
  - Isolation among subnets

  - No broadcast storms
    - Topology can be anything
    - Intelligent routing

  - #End-Hosts: 1000s+