

# *Chapter 5*

## *Link Layer*

---

*A S M Touhidul Hasan, Ph.D.*  
*Assistant Professor,*  
*Department of CSE, UAP*

# Link layer, LANs: outline

## 5.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

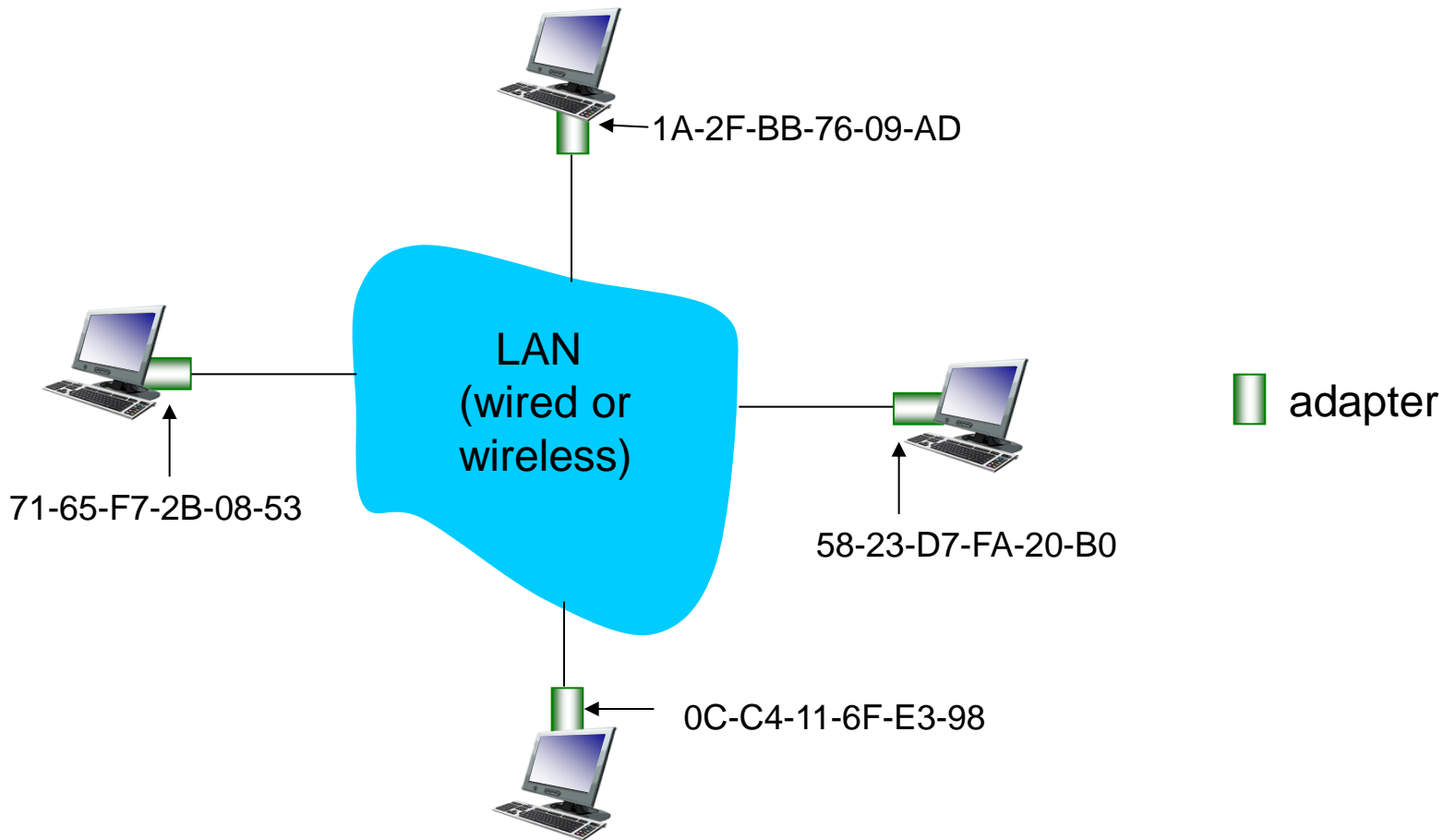
# MAC addresses and ARP

- ❖ 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-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 “number” represents 4 bits)

# LAN addresses and ARP

each adapter on LAN has unique **LAN** address



# 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

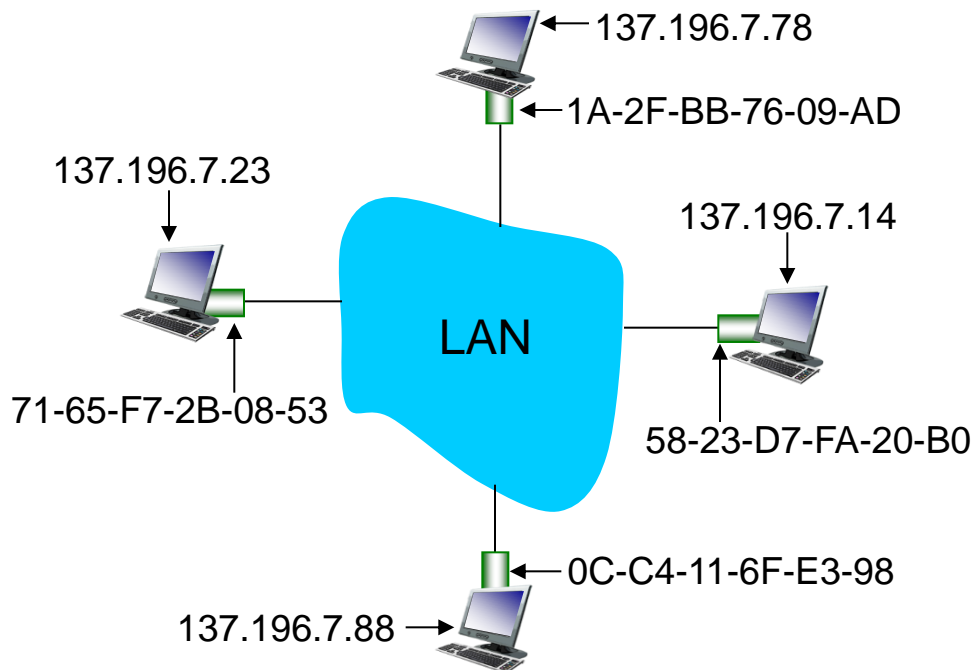
**Question:** how to determine interface's MAC address, knowing its IP address?

**ARP table:** each IP node (host, router) on 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)



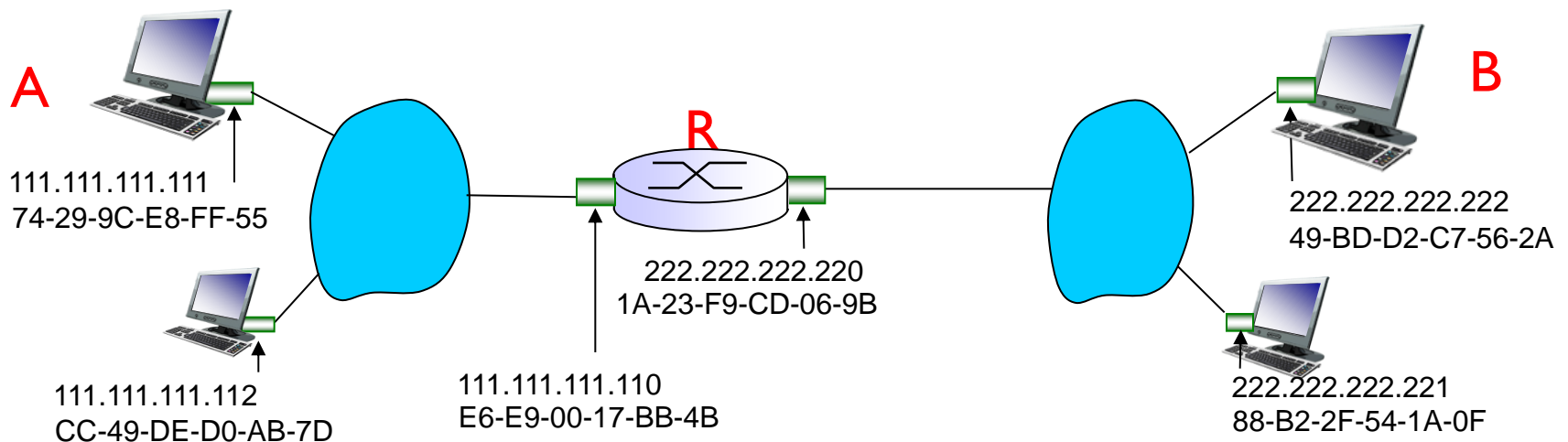
# ARP protocol: same LAN

- ❖ 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 information becomes old (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*

# Addressing: routing to another LAN

walkthrough: **send datagram from A to B via R**

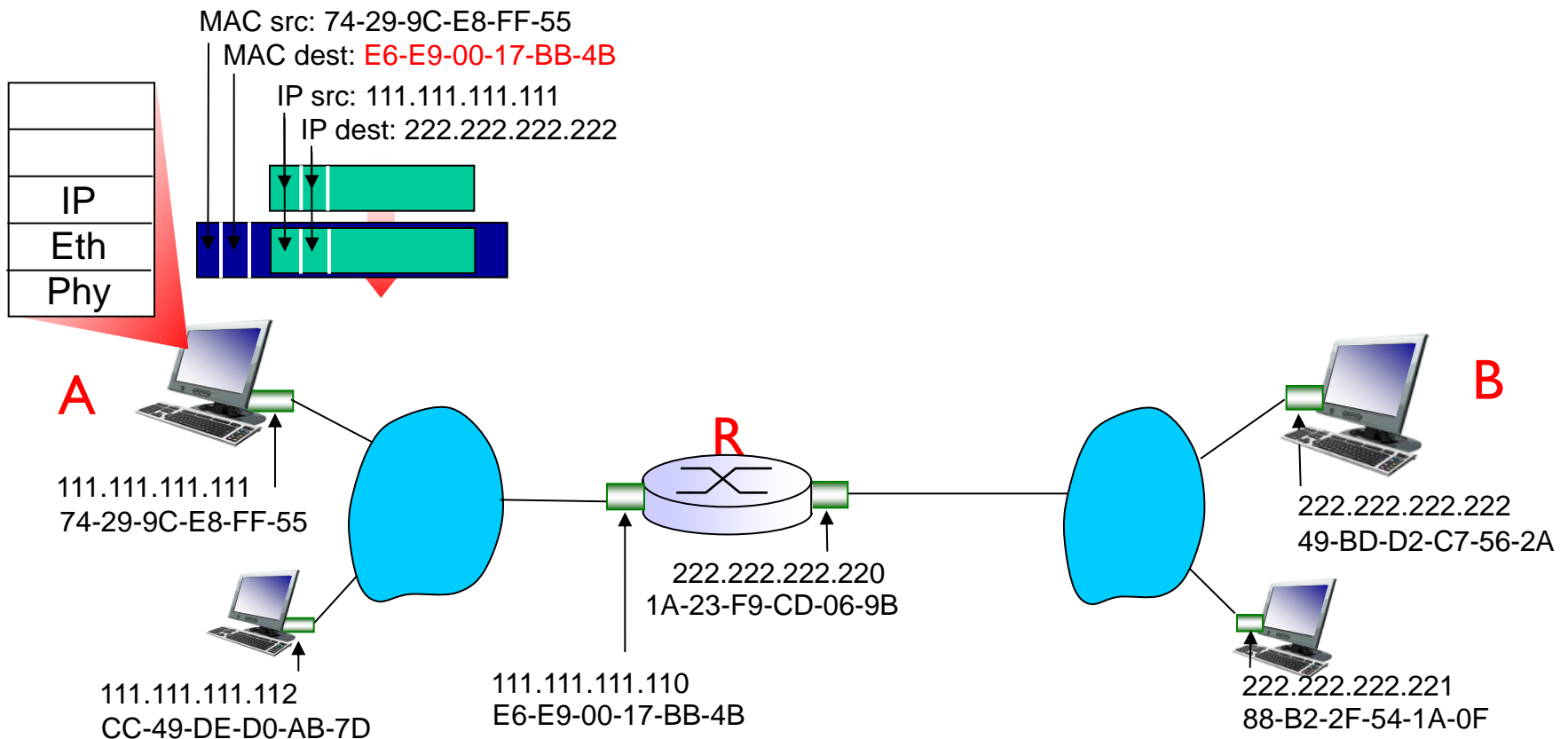
- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)





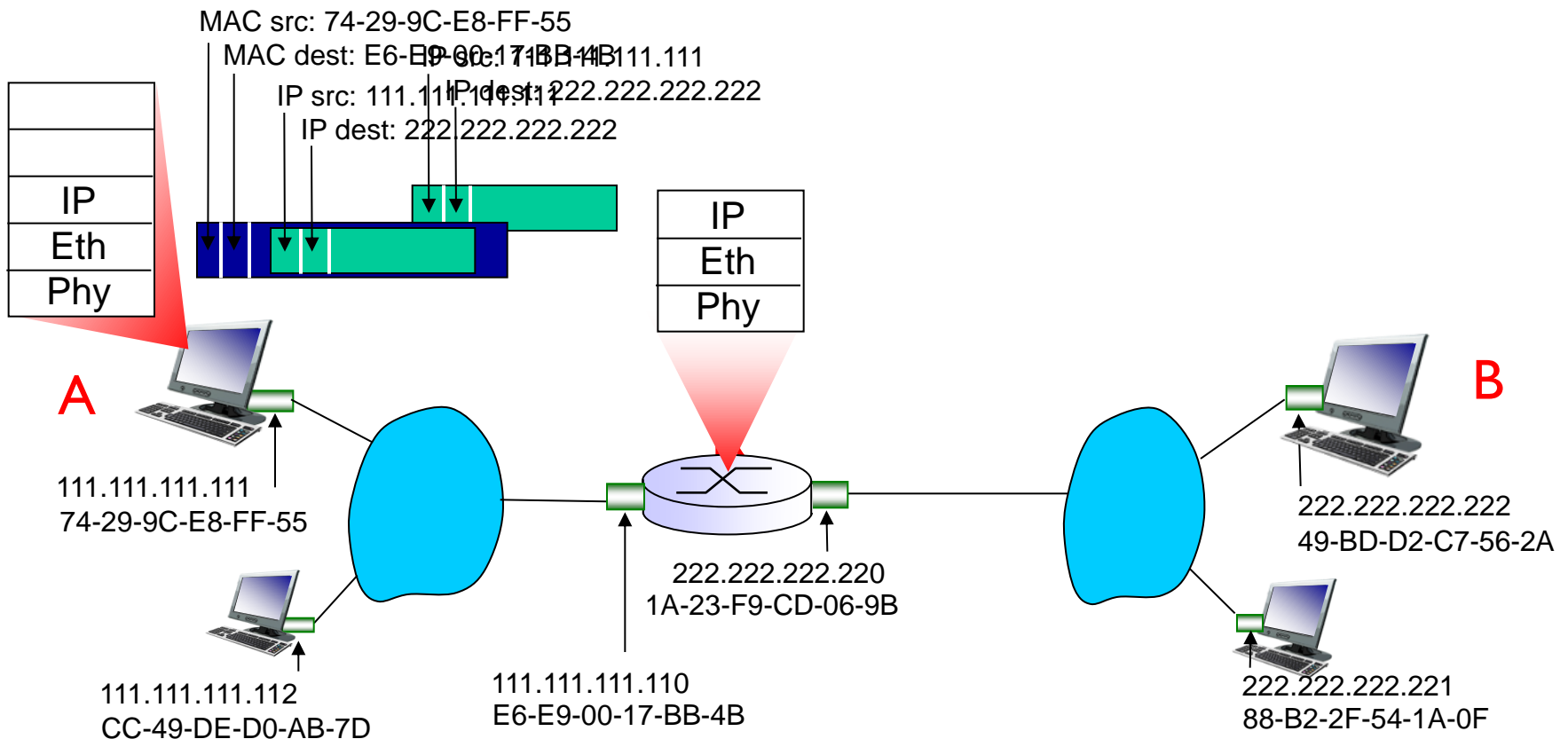
# Addressing: routing to another LAN

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



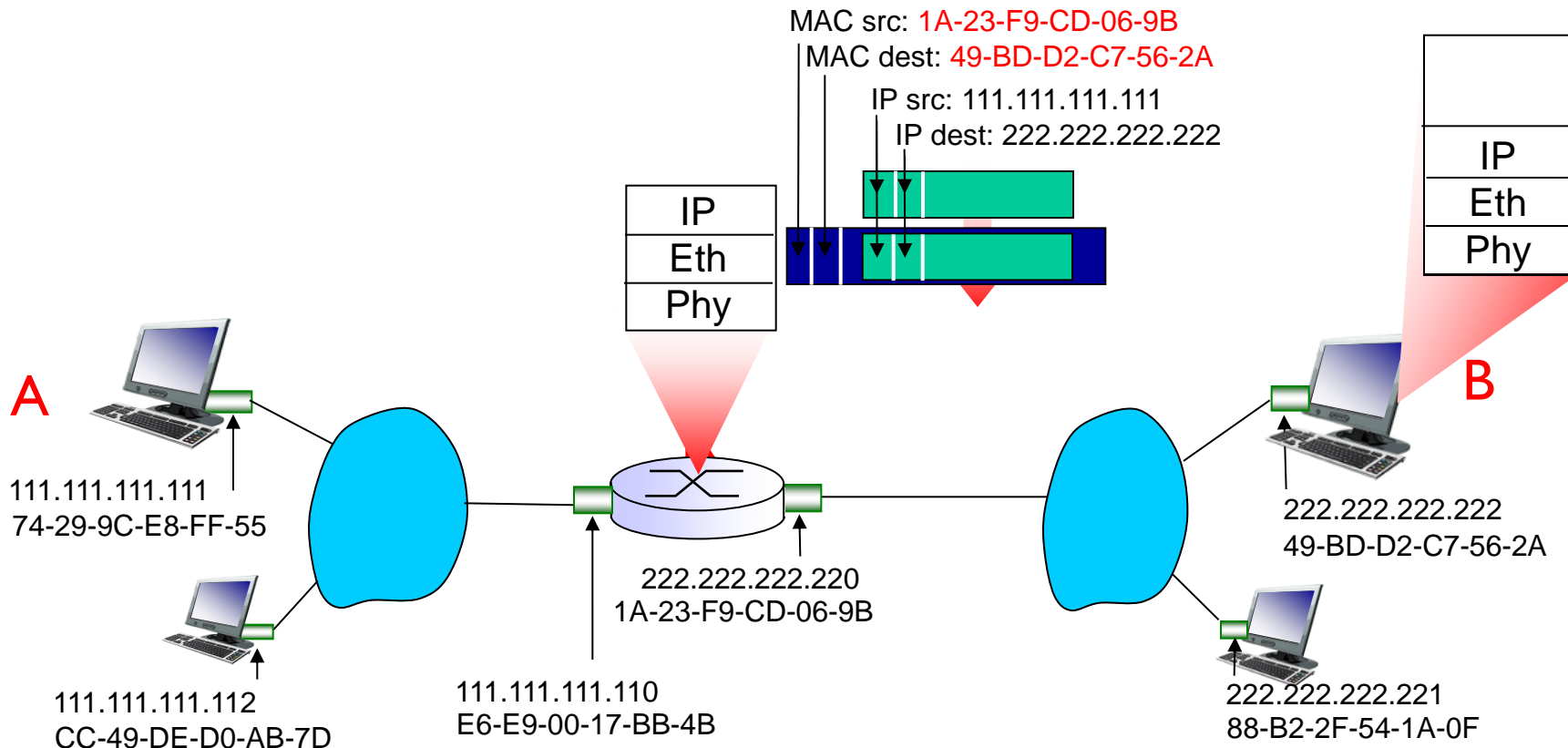
# Addressing: routing to another LAN

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



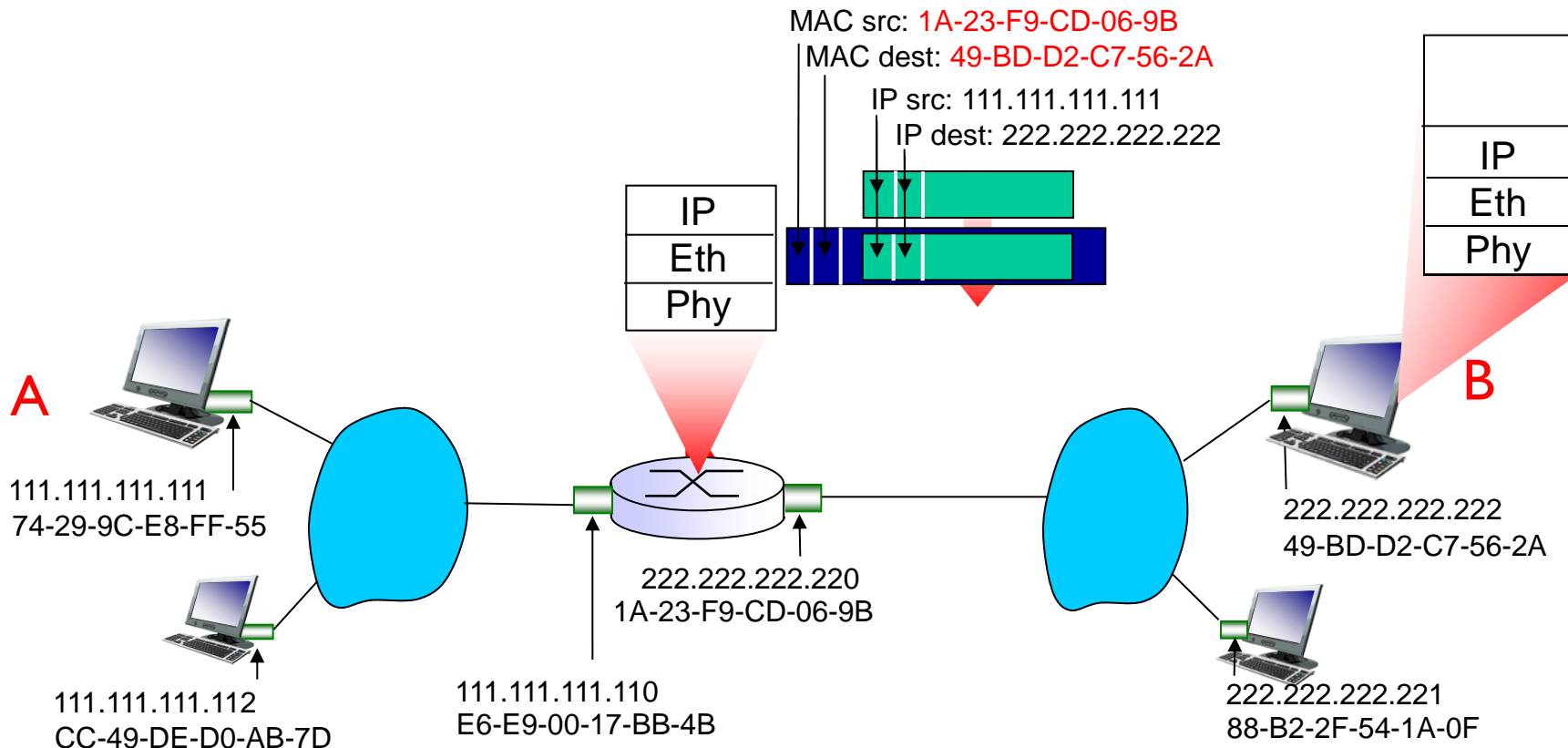
# Addressing: routing to another LAN

- ❖ 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



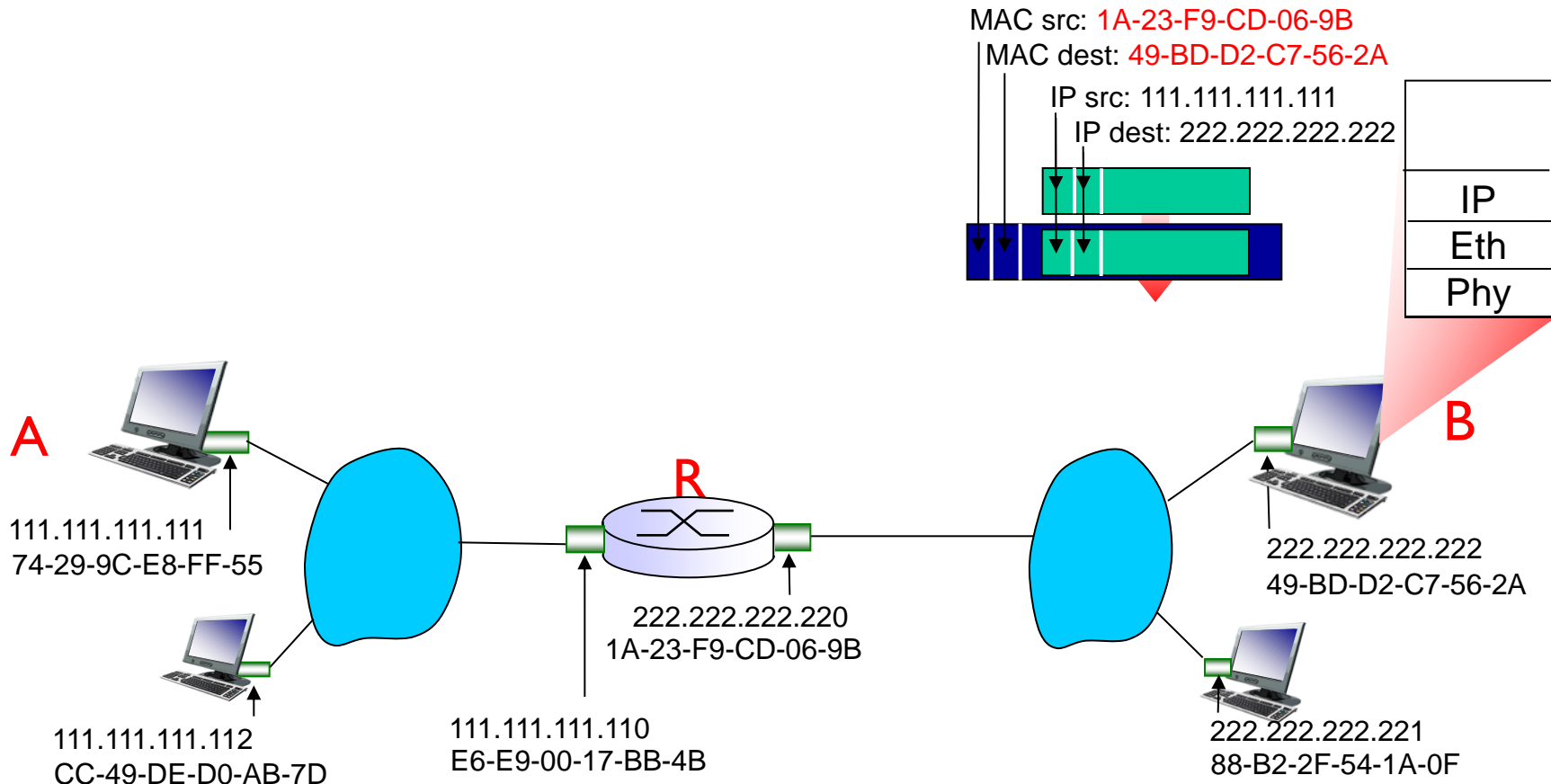
# Addressing: routing to another LAN

- ❖ 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



# 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



# Link layer, LANs: outline

5.1 introduction, services

5.2 error detection,  
correction

5.3 multiple access  
protocols

## 5.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

5.5 link virtualization:  
MPLS

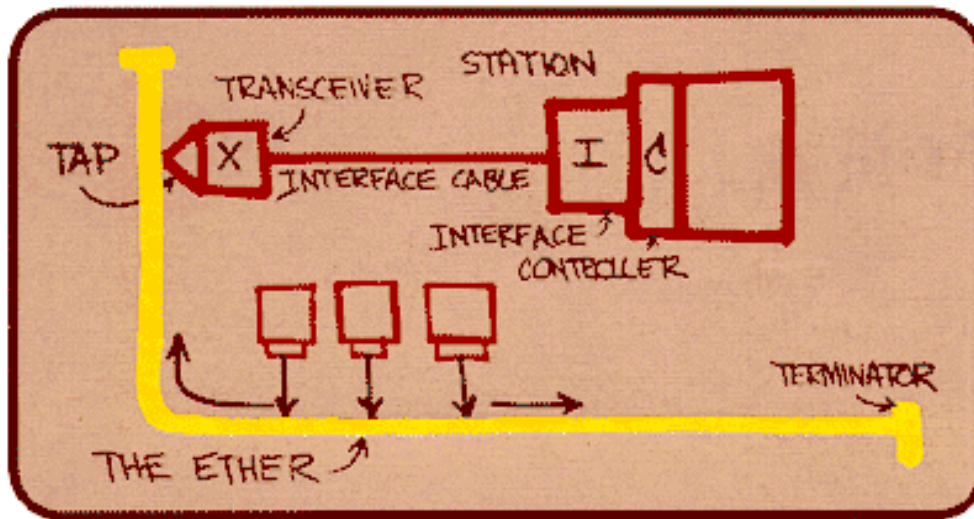
5.6 data center  
networking

5.7 a day in the life of a  
web request

# Ethernet

“dominant” wired LAN technology:

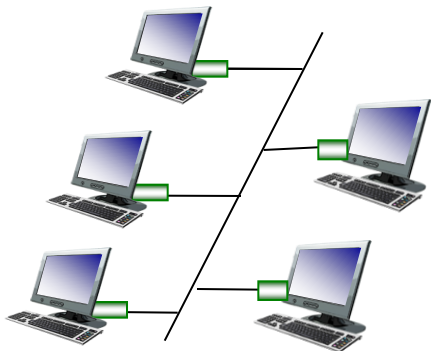
- ❖ cheap \$20 for NIC
- ❖ first widely used LAN technology
- ❖ simpler, cheaper than token LANs and ATM
- ❖ kept up with speed race: 10 Mbps – 10 Gbps



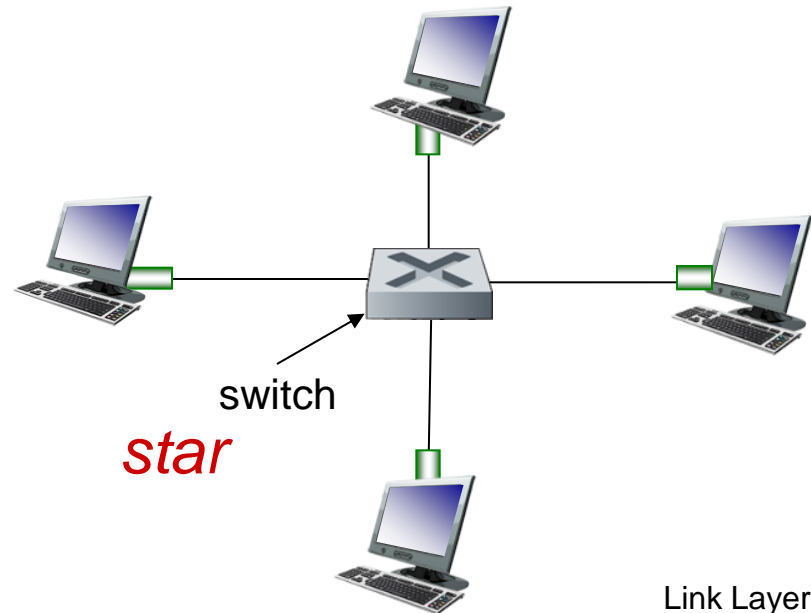
*Metcalfe's Ethernet sketch*

# 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





# Ethernet frame structure

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



## *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



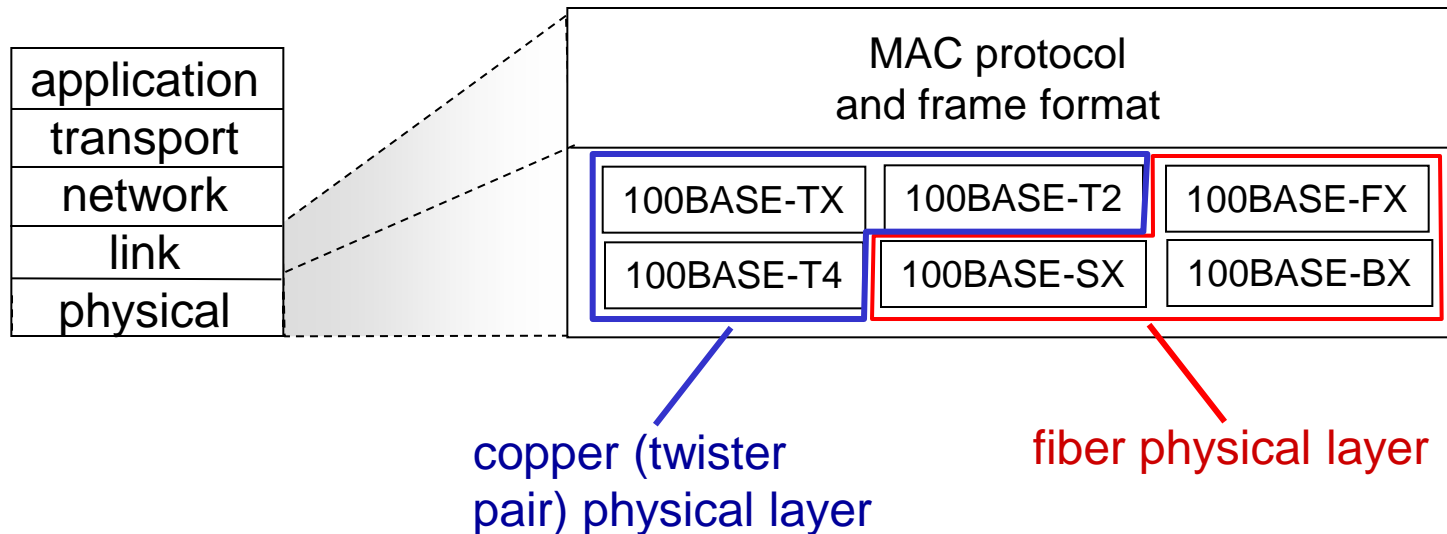
# Ethernet: unreliable, connectionless

- ❖ *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 binary 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, 1 Gbps, 10G bps
- different physical layer media: fiber, cable



# Link layer, LANs: outline

5.1 introduction, services

5.2 error detection,  
correction

5.3 multiple access  
protocols

5.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

5.5 link virtualization:  
MPLS

5.6 data center  
networking

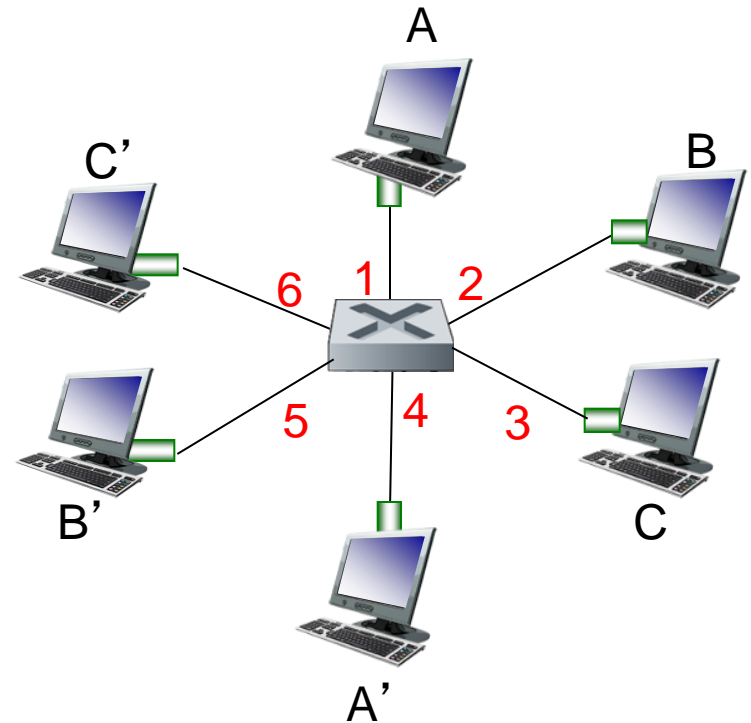
5.7 a day in the life of a  
web request

# 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

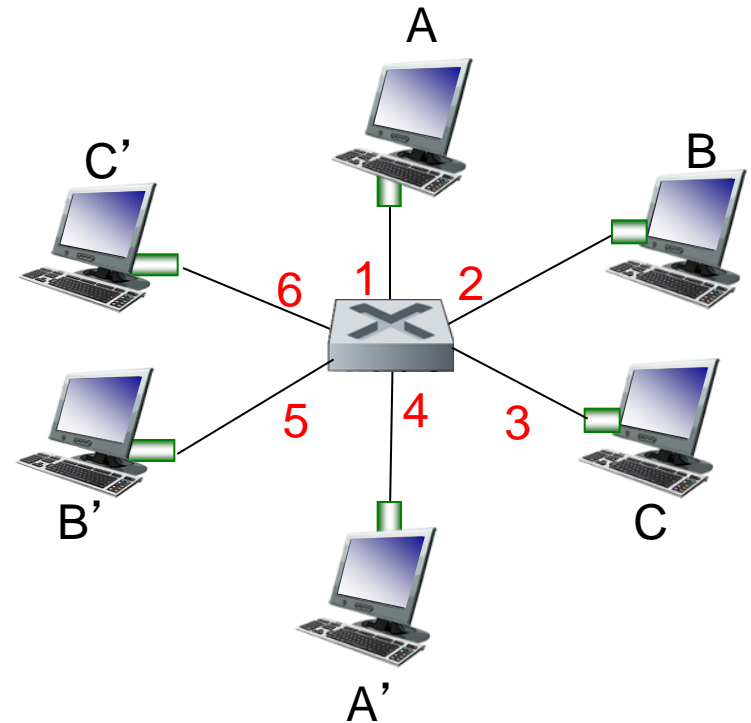
**Q:** how does switch know A' reachable via interface 4, B' reachable via interface 5?

❖ **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?

- something like a routing protocol?

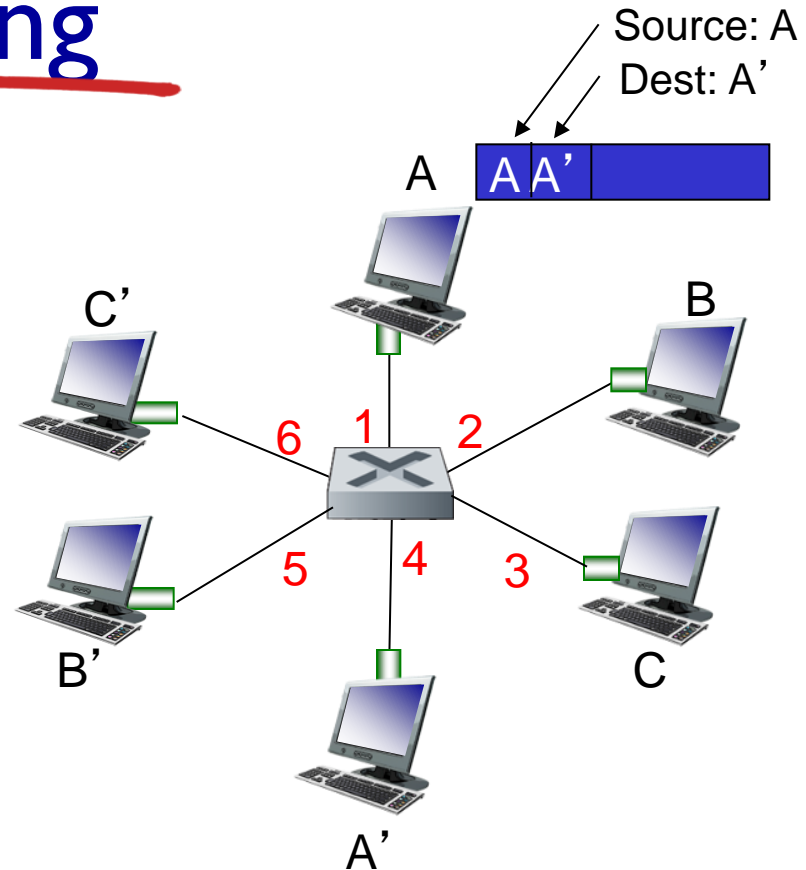


switch with six interfaces  
(1,2,3,4,5,6)



# 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



MAC addr	interface	TTL
A	1	60

*Switch table  
(initially empty)*

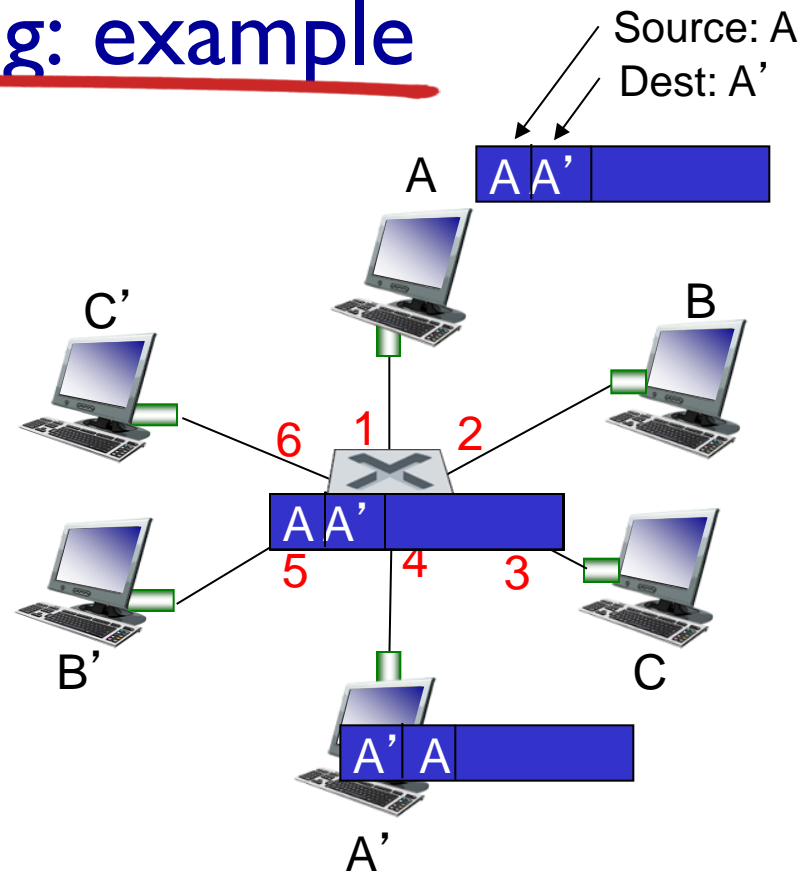
# Switch: frame filtering/forwarding

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

- ❖ frame destination, A', location unknown: *flood*
- ❖ destination A location known: *selectively send on just one link*

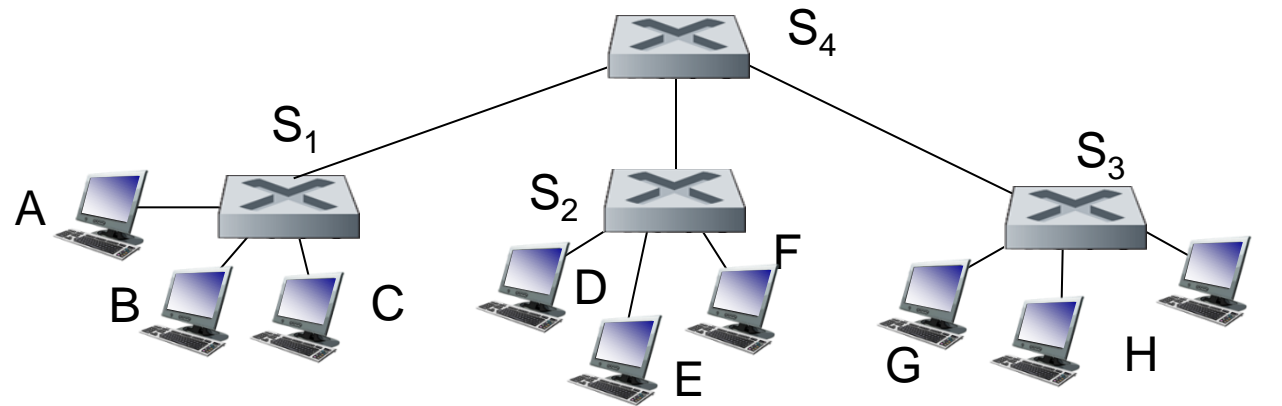


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table  
(initially empty)*

# Interconnecting switches

- ❖ switches can be connected together

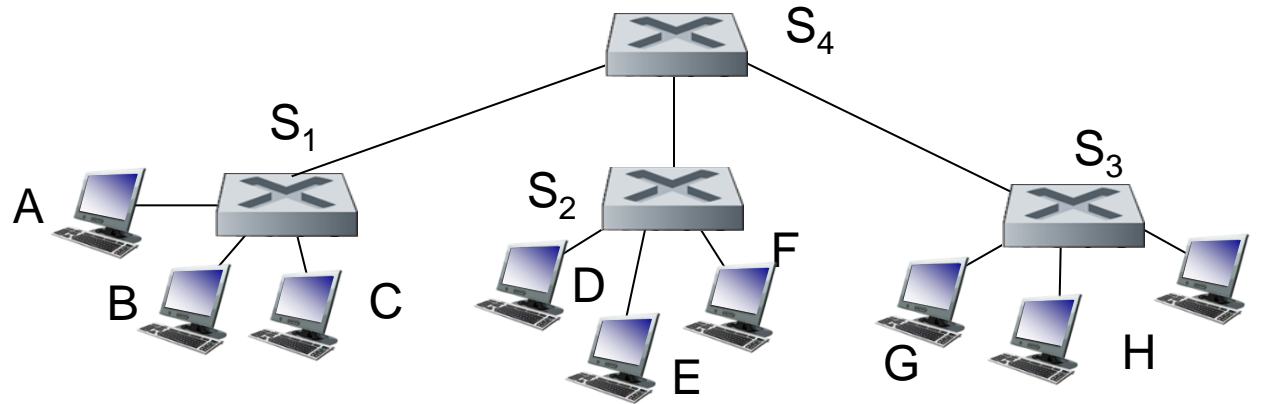


**Q:** sending from A to G - how does  $S_1$  know to forward frame destined to F via  $S_4$  and  $S_3$ ?

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

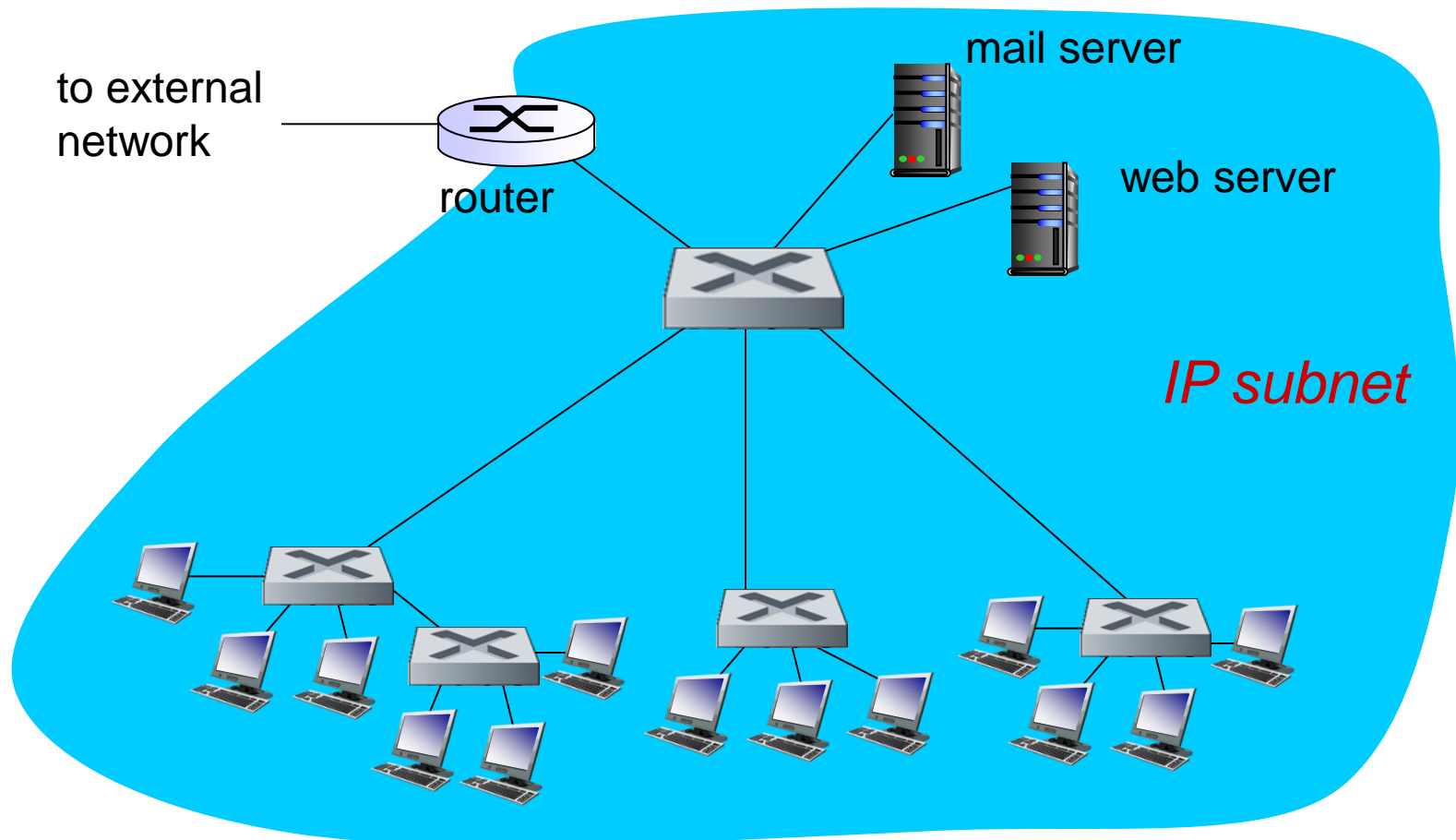
# Self-learning multi-switch example

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



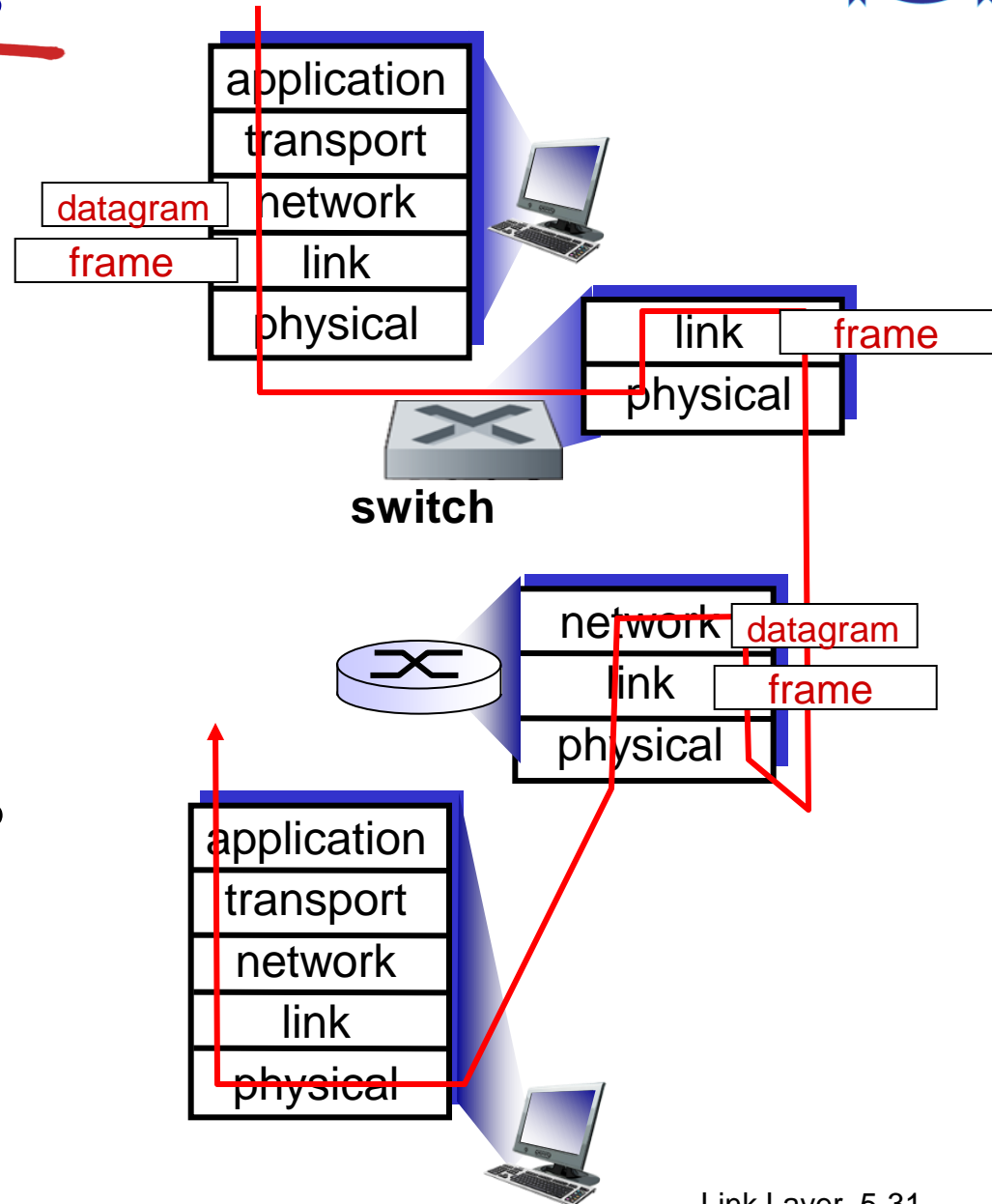
# Switches vs. routers

both are store-and-forward:

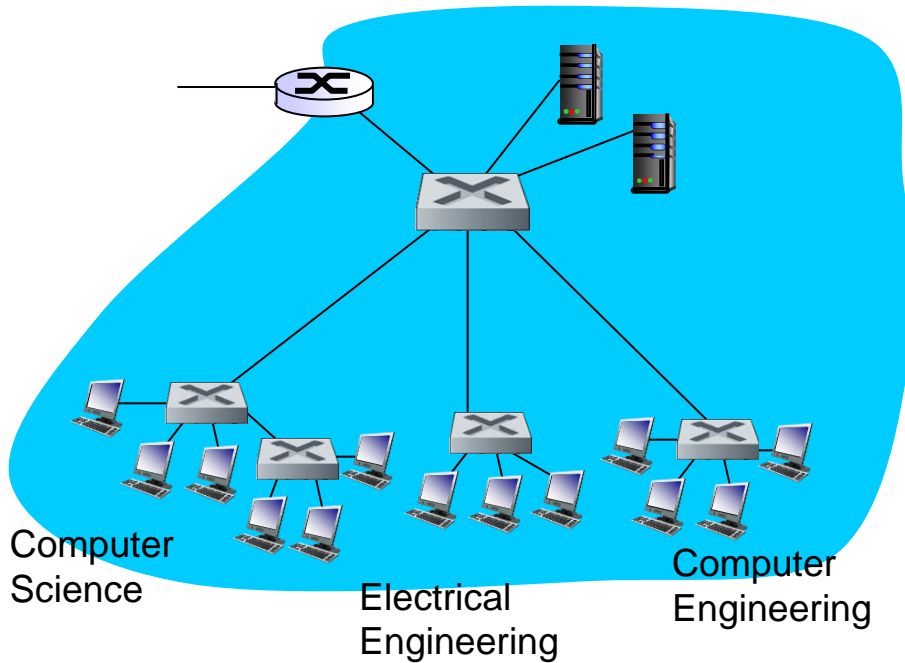
- **routers:** network-layer devices (examine network-layer headers)
- **switches:** link-layer devices (examine link-layer headers)

both have forwarding tables:

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



# VLANs: motivation



*consider:*

- ❖ CS user moves office to EE, but wants connect to CS switch?
- ❖ single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

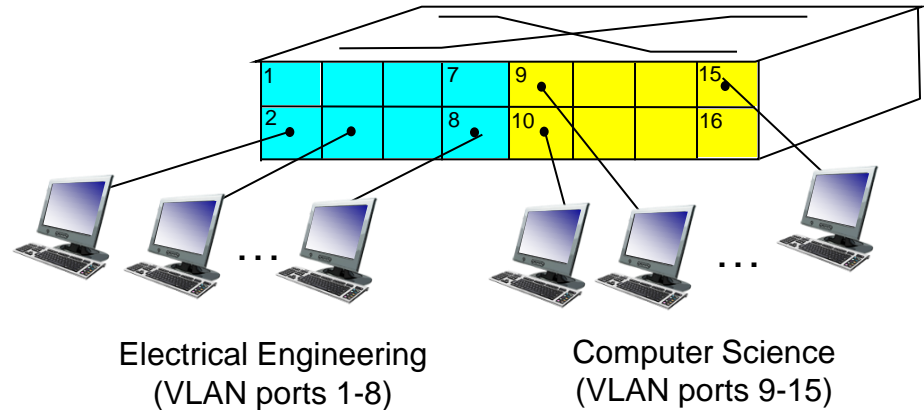


# VLANs

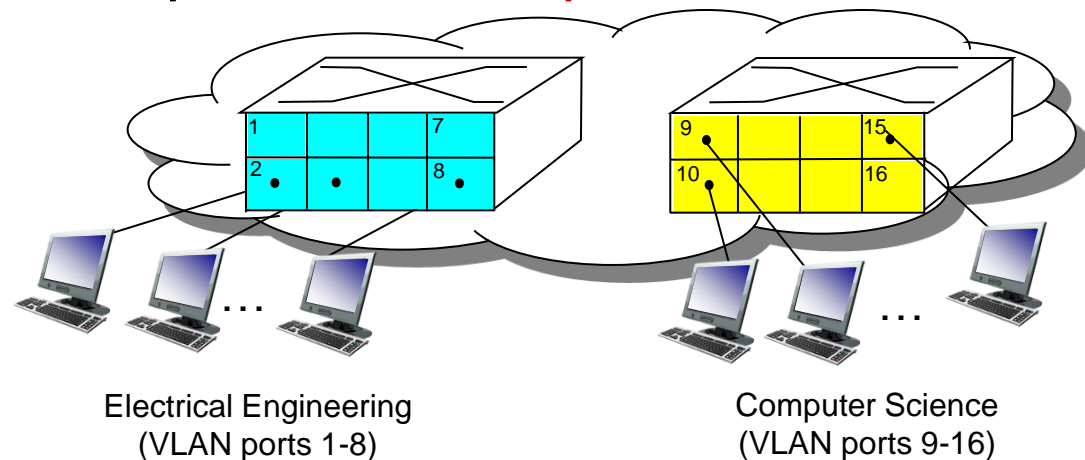
## *Virtual Local Area Network*

switch(es) supporting VLAN capabilities can be configured to define multiple *virtual* LANS over single physical LAN infrastructure.

**port-based VLAN:** switch ports grouped (by switch management software) so that *single* physical switch .....

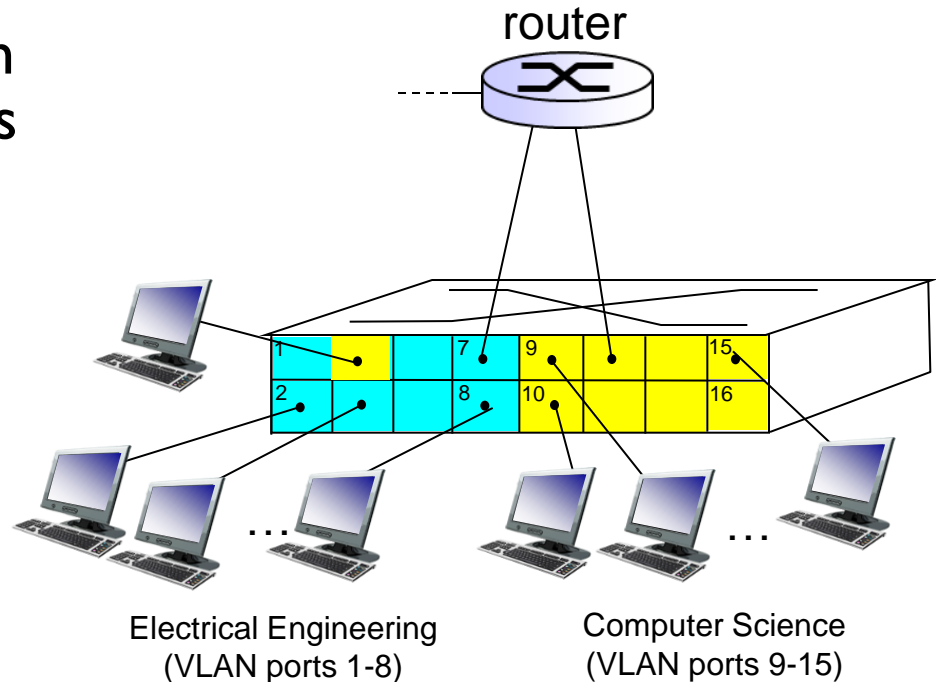


... operates as *multiple* virtual switches

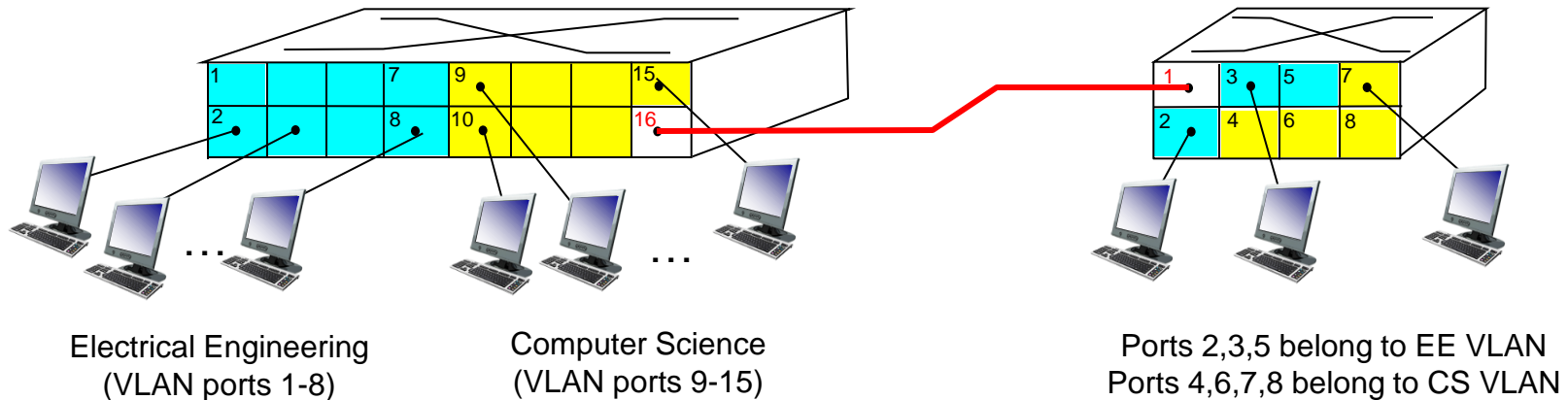


# Port-based VLAN

- ❖ *traffic isolation*: frames to/from ports 1-8 can only reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- ❖ *dynamic membership*: ports can be dynamically assigned among VLANs
- ❖ *forwarding between VLANs*: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



# VLANs spanning multiple switches



- ❖ **trunk port:** carries frames between VLANs defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

# 802.1Q VLAN frame format

