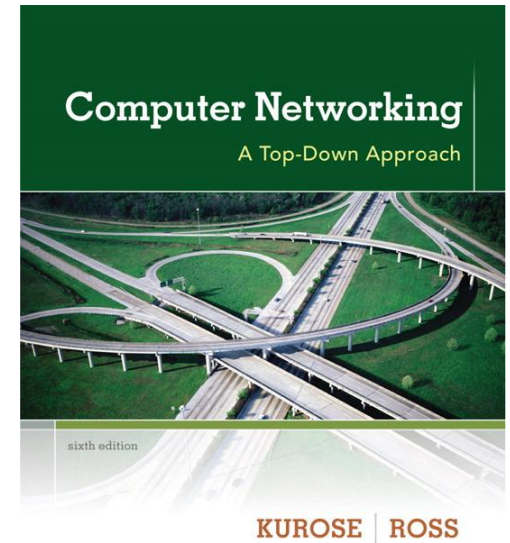


Chapter 5

Link Layer



*Computer
Networking: A Top
Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012*

© All material copyright 1996-2012
J.F Kurose and K.W. Ross, All Rights Reserved

Chapter 5: Link layer

our goals:

- ❖ understand principles behind link layer services:
 - error detection, correction
 - sharing a broadcast channel: multiple access
 - link layer addressing
 - local area networks: Ethernet, VLANs
- ❖ instantiation, implementation of various link layer technologies

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

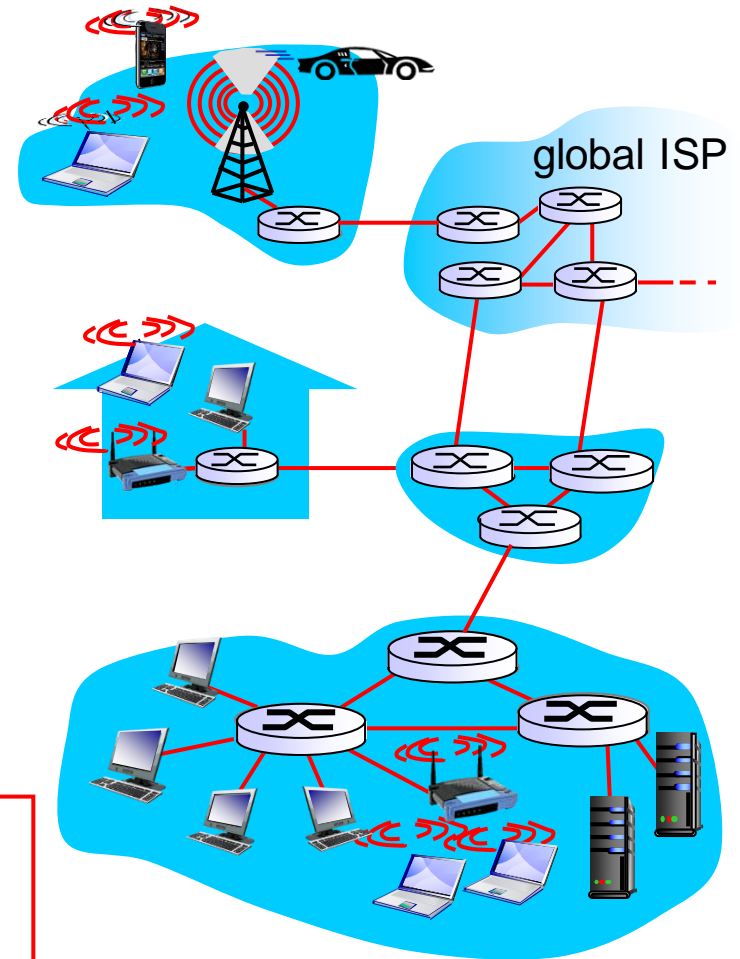
5.7 a day in the life of a
web request

Link layer: introduction

terminology:

- ❖ hosts and routers: **nodes**
- ❖ communication channels that connect adjacent nodes along communication path: **links**
 - wired links
 - wireless links
 - LANs
- ❖ layer-2 packet: **frame**, encapsulates datagram

data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link



Exercise : การเดินทาง

- ❖ จากบ้านมาที่สถาบันฯ ต้องเดินทางมาช่องทางใดบ้าง
 - รถเมล์
 - รถไฟ
 - เรือ
 - เครื่องบิน
 - ฯลฯ

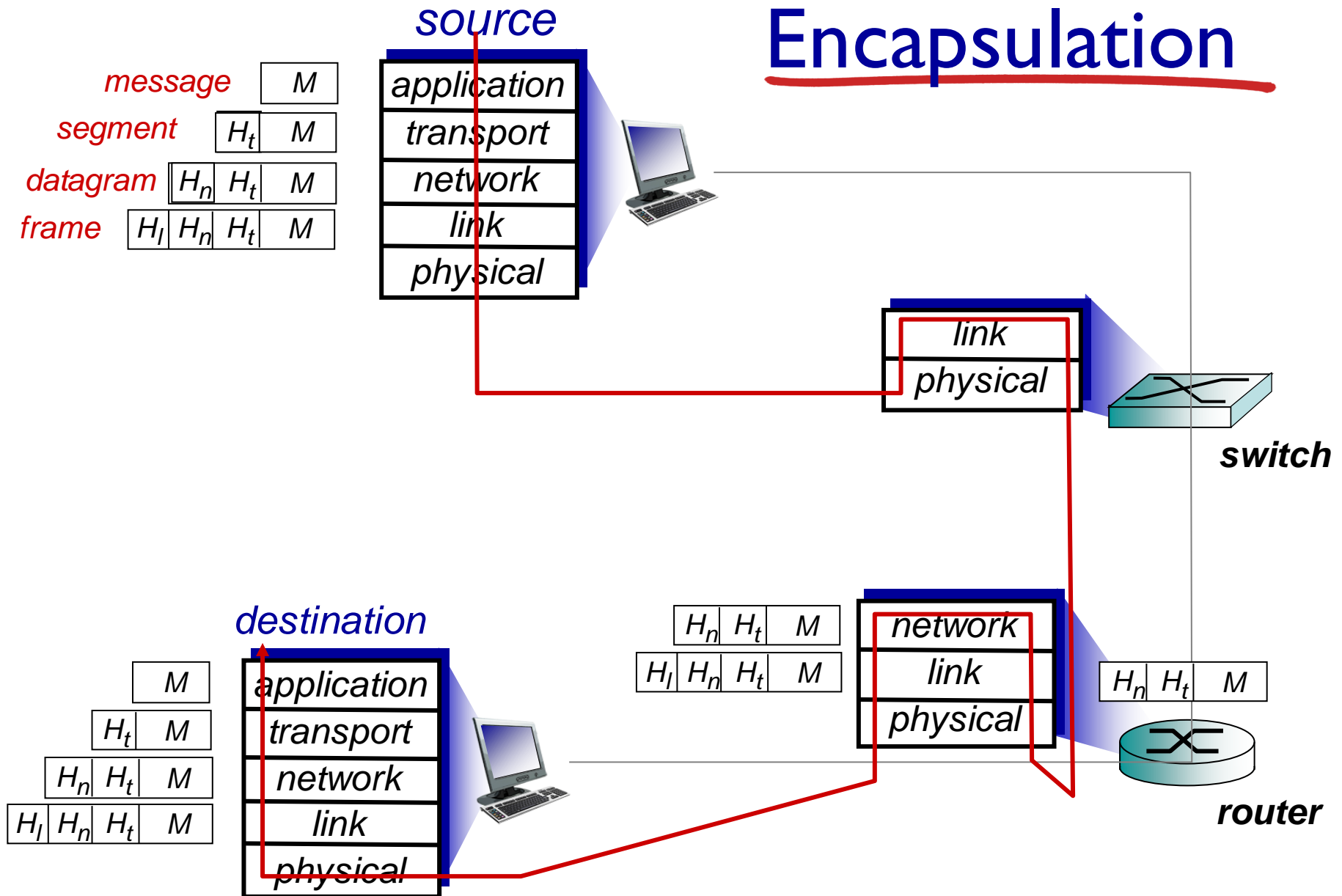
Link layer: context

- ❖ datagram transferred by different link protocols over different links:
 - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- ❖ each link protocol provides different services
 - e.g., may or may not provide rdt over link

transportation analogy:

- ❖ trip from Princeton to Lausanne
 - limo: Princeton to JFK
 - plane: JFK to Geneva
 - train: Geneva to Lausanne
- ❖ tourist = **datagram**
- ❖ transport segment = **communication link**
- ❖ transportation mode = **link layer protocol**
- ❖ travel agent = **routing algorithm**

Encapsulation



Link layer services

❖ *framing, link access:*

- encapsulate datagram into frame, adding header, trailer
- channel access if shared medium
- “MAC” addresses used in frame headers to identify source, dest
 - different from IP address!

❖ *reliable delivery between adjacent nodes*

- we learned how to do this already (chapter 3)!
- seldom used on low bit-error link (fiber, some twisted pair)
- wireless links: high error rates
 - *Q*: why both link-level and end-end reliability?

Link layer services (more)

❖ *flow control:*

- pacing between adjacent sending and receiving nodes

❖ *error detection:*

- errors caused by signal attenuation, noise.
- receiver detects presence of errors:
 - signals sender for retransmission or drops frame

❖ *error correction:*

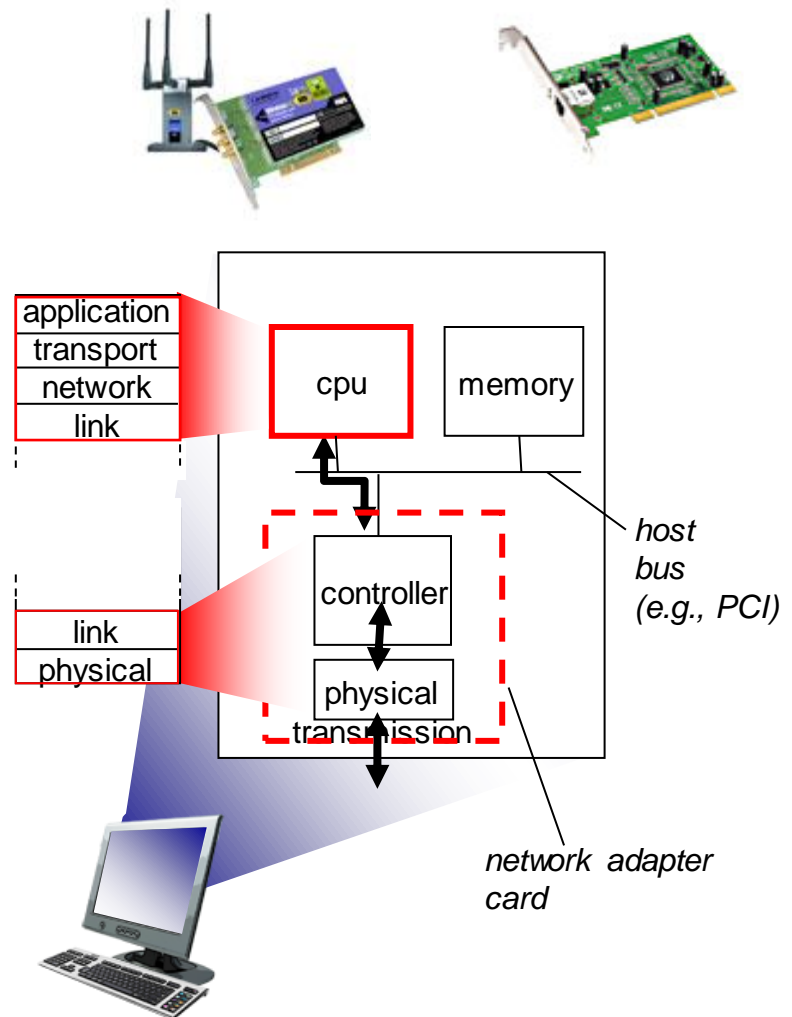
- receiver identifies *and corrects* bit error(s) without resorting to retransmission

❖ *half-duplex and full-duplex*

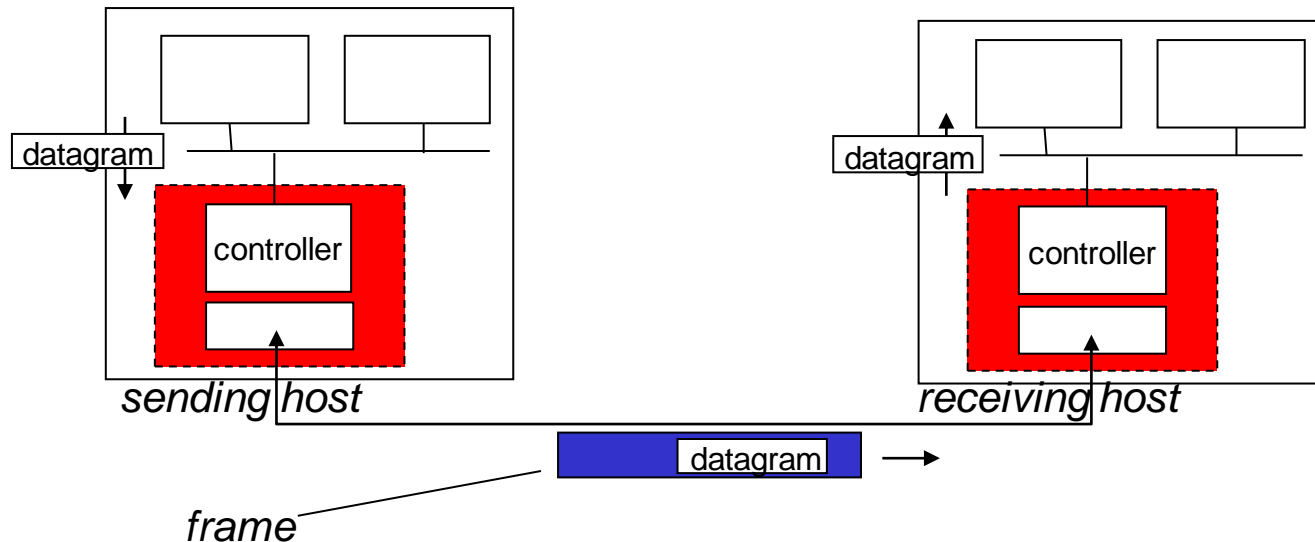
- with half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- ❖ in each and every host
- ❖ link layer implemented in “adaptor” (aka *network interface card* NIC) or on a chip
 - Ethernet card, 802.11 card; Ethernet chipset
 - implements link, physical layer
- ❖ attaches into host's system buses
- ❖ combination of hardware, software, firmware



Adaptors communicating



❖ sending side:

- encapsulates datagram in frame
- adds error checking bits, rdt, flow control, etc.

❖ receiving side

- looks for errors, rdt, flow control, etc
- extracts datagram, passes to upper layer at receiving side

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

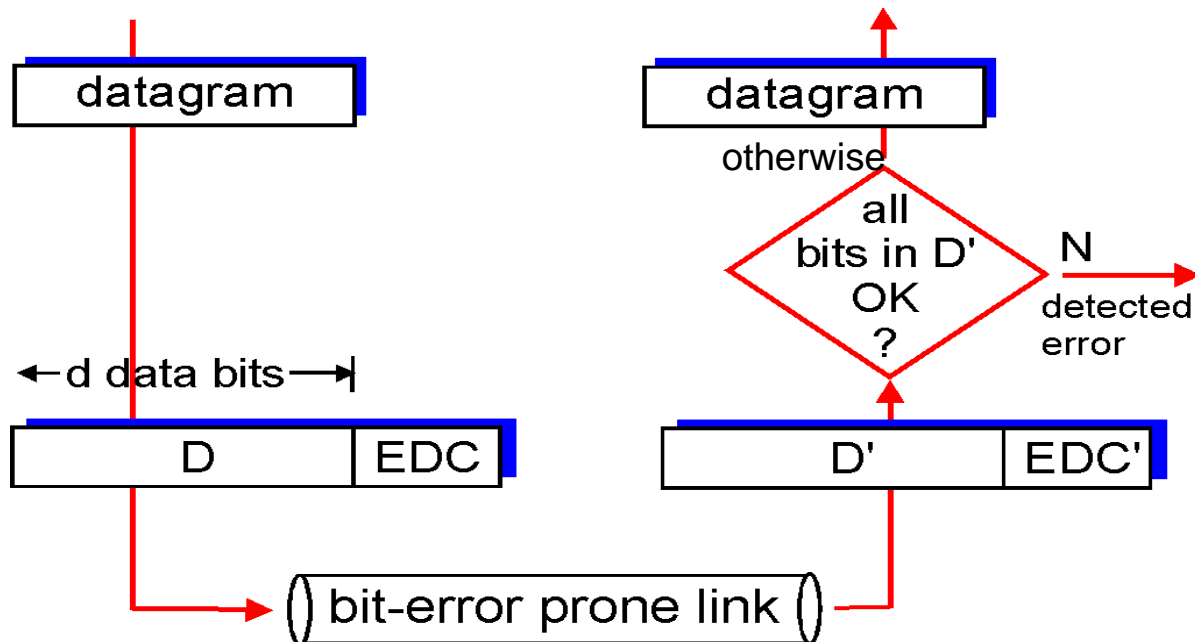
5.7 a day in the life of a
web request

Error detection

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

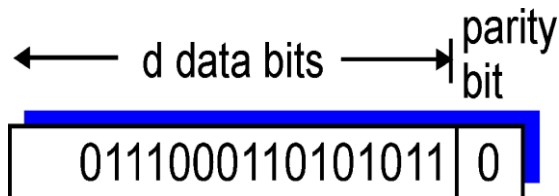
- Error detection not 100% reliable!
 - protocol may miss some errors, but rarely
 - larger EDC field yields better detection and correction



Parity checking

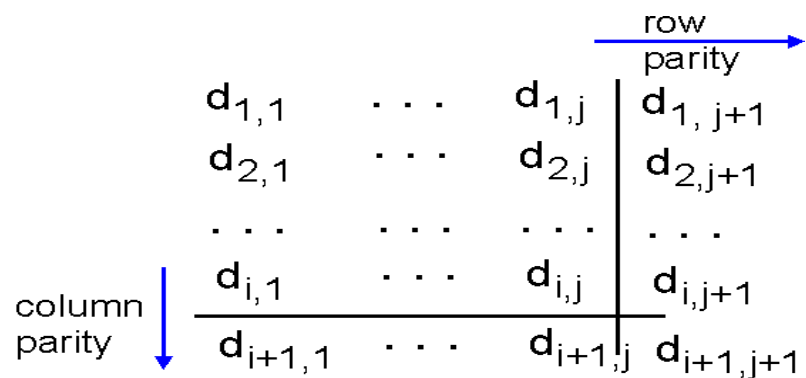
single bit parity:

- ❖ detect single bit errors



two-dimensional bit parity:

- ❖ detect and correct single bit errors



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

no errors

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

parity error

*correctable
single bit error*

Example : Parity Check

Figure 10.11 Two-dimensional parity-check code

1	1	0	0	1	1	1	1	Row parities
1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1	1
Column parities								1
0	1	0	1	0	1	0	1	1

a. Design of row and column parities

Figure 10.11 Two-dimensional parity-check code

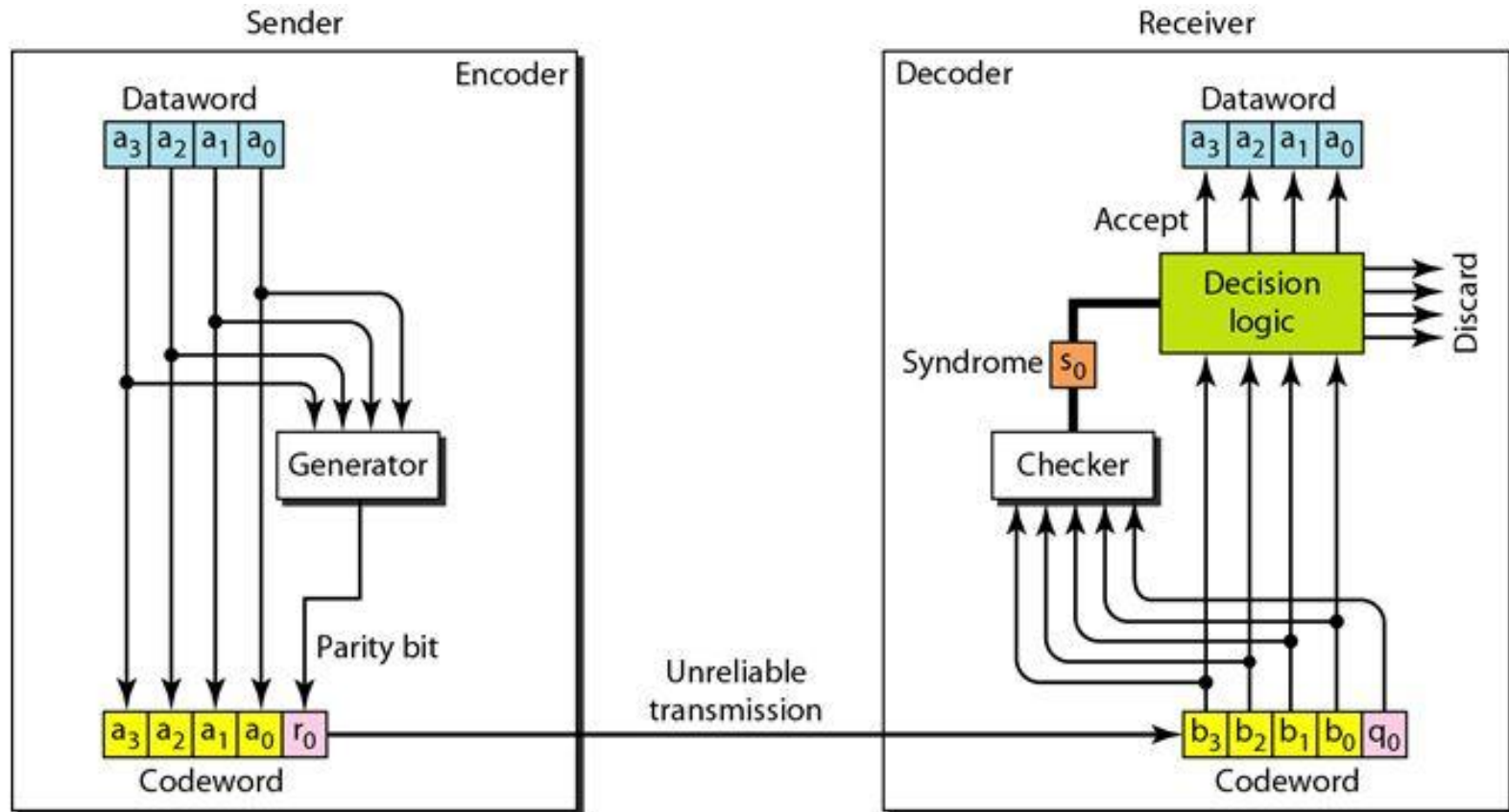
1	1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1	1
								1

b. One error affects two parities

1	1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1	1
								1

c. Two errors affect two parities

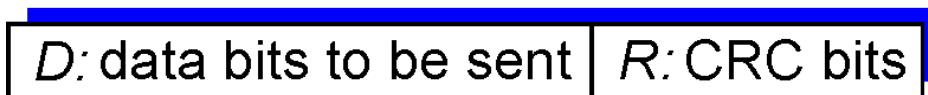
Exercise : Parity Check



Cyclic redundancy check

- ❖ more powerful error-detection coding
- ❖ view data bits, **D**, as a binary number
- ❖ choose $r+1$ bit pattern (generator), **G**
- ❖ goal: choose r CRC bits, **R**, such that
 - $\langle D, R \rangle$ exactly divisible by G (modulo 2)
 - receiver knows G , divides $\langle D, R \rangle$ by G . If non-zero remainder: error detected!
 - can detect all burst errors less than $r+1$ bits
- ❖ widely used in practice (Ethernet, 802.11 WiFi, ATM)

← d bits → ← r bits →

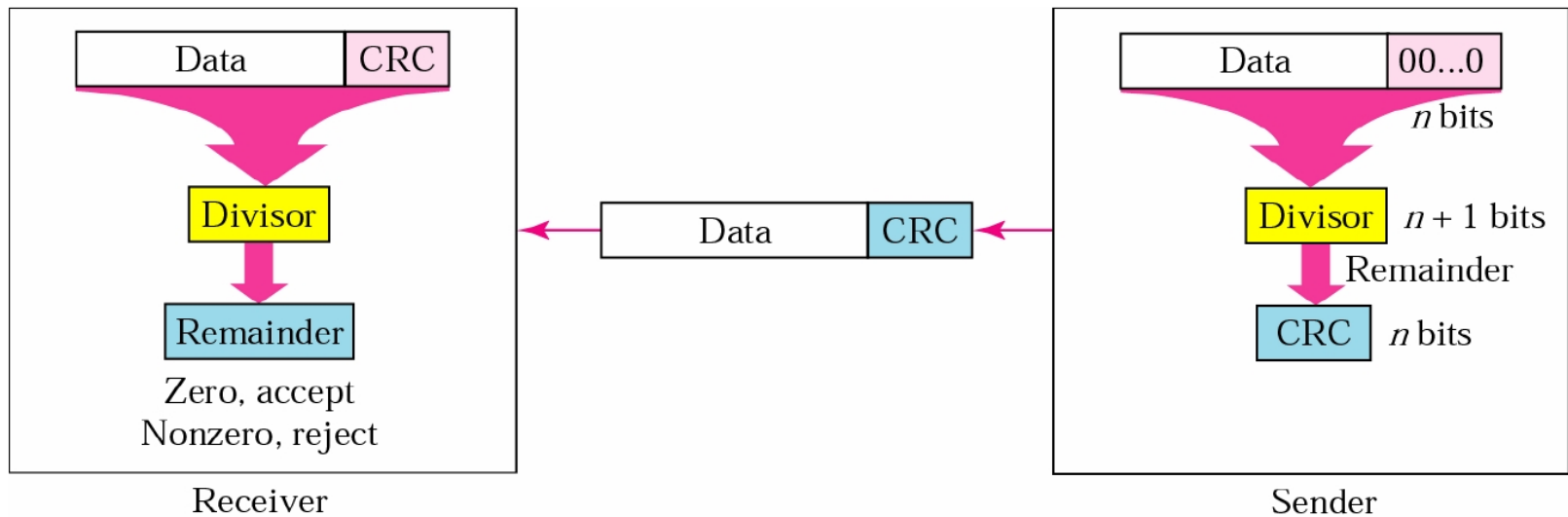


*bit
pattern*

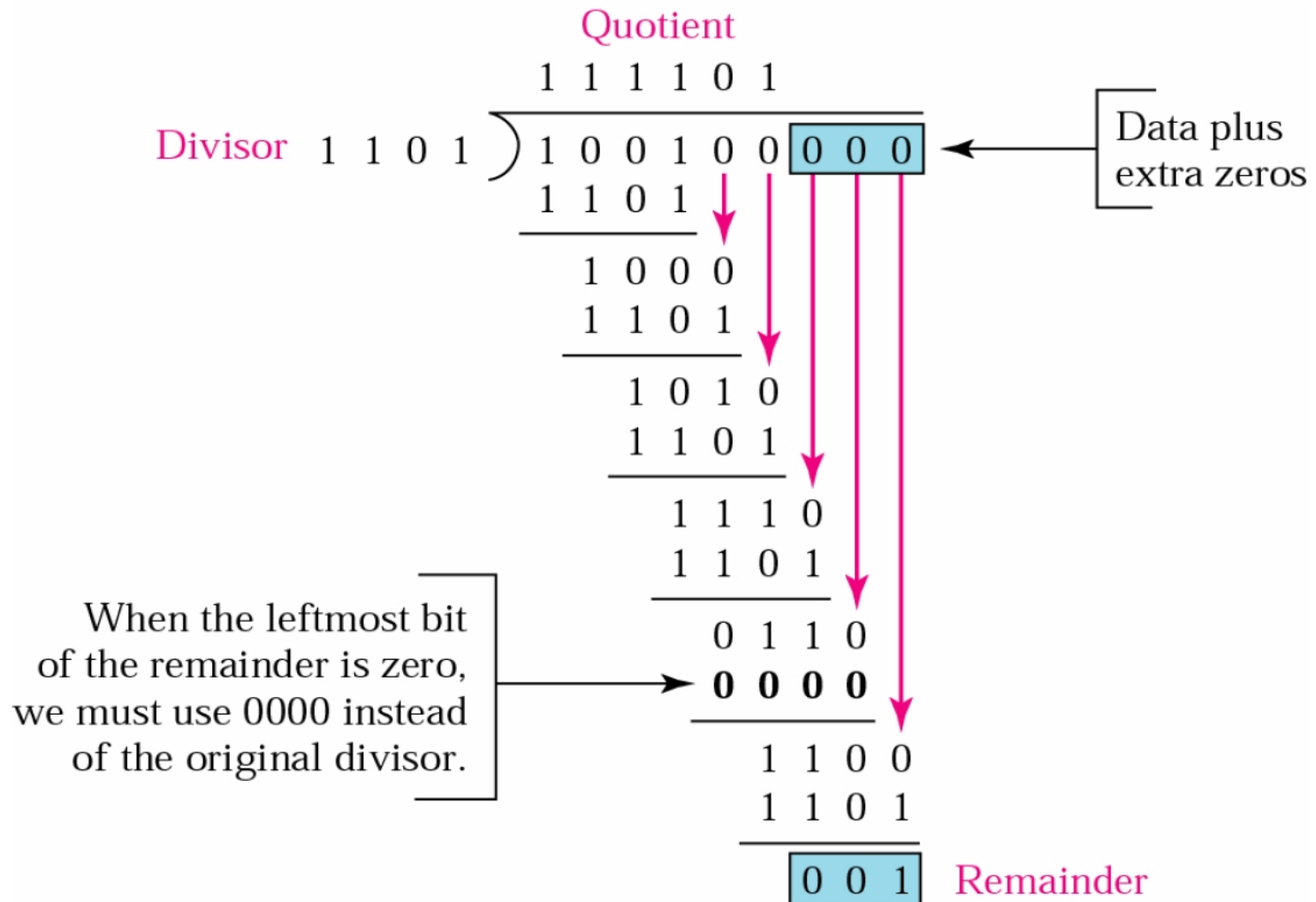
$$D * 2^r \text{ XOR } R$$

*mathematical
formula*

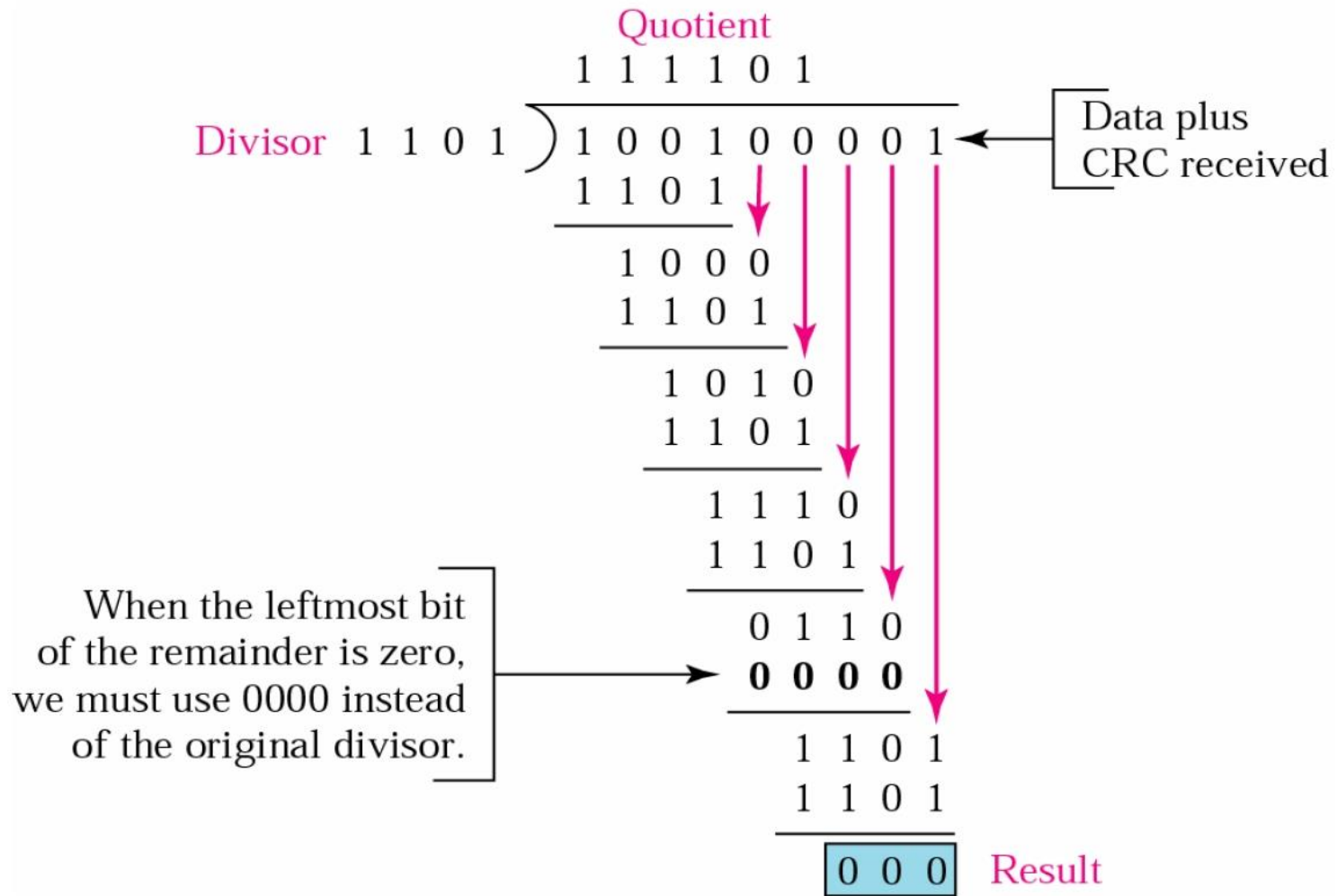
Cyclic redundancy check



CRC example



CRC example



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

Multiple access links, protocols

two types of “links”:

- ❖ point-to-point

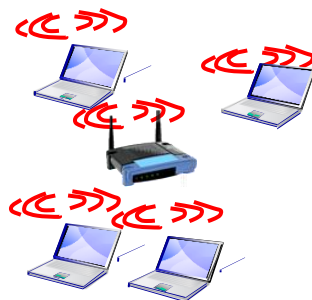
- PPP for dial-up access
- point-to-point link between Ethernet switch, host

- ❖ *broadcast (shared wire or medium)*

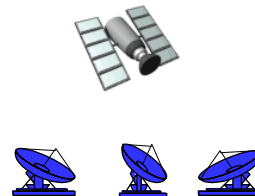
- old-fashioned Ethernet
- upstream HFC
- 802.11 wireless LAN



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)



humans at a
cocktail party
(shared air, acoustical)

Multiple access protocols

- ❖ single shared broadcast channel
- ❖ two or more simultaneous transmissions by nodes:
interference
 - *collision* if node receives two or more signals at the same time

multiple access protocol

- ❖ distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- ❖ communication about channel sharing must use channel itself!
 - no out-of-band channel for coordination

Exercise : Multiple Access

- ❖ ให้นักศึกษาเสนอวิธีพูดคุยกันในกลุ่ม โดยไม่มี 2 คนที่พูดในเวลาเดียวกัน

An ideal multiple access protocol

given: broadcast channel of rate R bps

desiderata:

1. when one node wants to transmit, it can send at rate R .
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
 - no special node to coordinate transmissions
 - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

three broad classes:

- ❖ *channel partitioning*

- divide channel into smaller “pieces” (time slots, frequency, code)
- allocate piece to node for exclusive use

- ❖ *random access*

- channel not divided, allow collisions
- “recover” from collisions

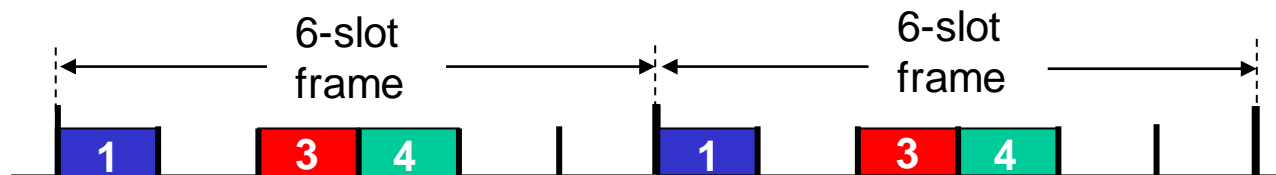
- ❖ *“taking turns”*

- nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

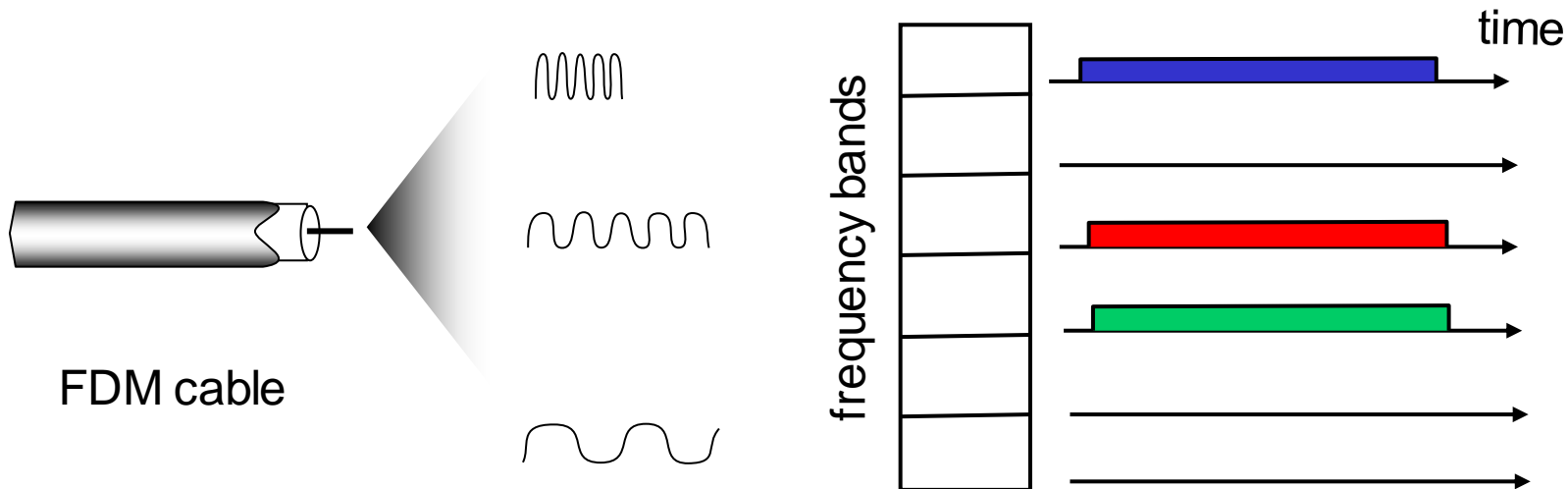
- ❖ access to channel in "rounds"
- ❖ each station gets fixed length slot (length = pkt trans time) in each round
- ❖ unused slots go idle
- ❖ example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- ❖ channel spectrum divided into frequency bands
- ❖ each station assigned fixed frequency band
- ❖ unused transmission time in frequency bands go idle
- ❖ example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random access protocols

- ❖ when node has packet to send
 - transmit at full channel data rate R .
 - no *a priori* coordination among nodes
- ❖ two or more transmitting nodes → “collision”,
- ❖ **random access MAC protocol** specifies:
 - how to detect collisions
 - how to recover from collisions (e.g., via delayed retransmissions)
- ❖ examples of random access MAC protocols:
 - slotted ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

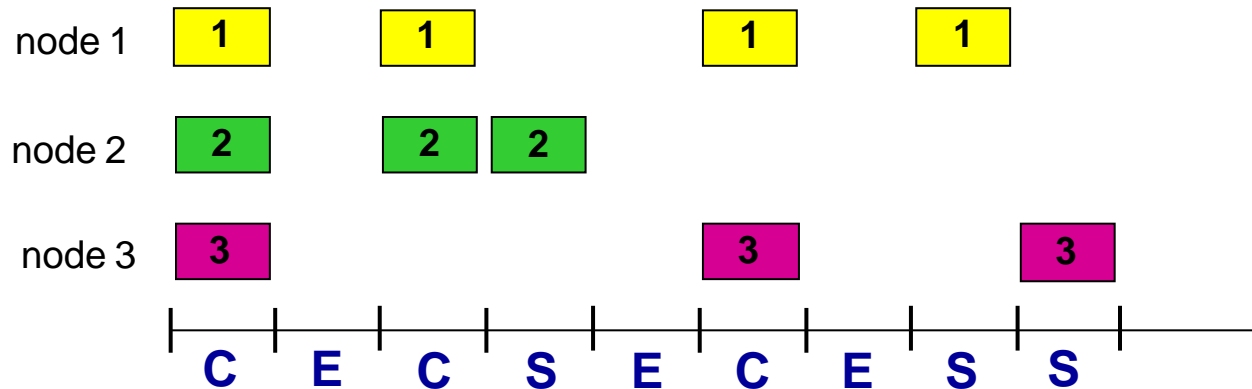
assumptions:

- ❖ all frames same size
- ❖ time divided into equal size slots (time to transmit 1 frame)
- ❖ nodes start to transmit only slot beginning
- ❖ nodes are synchronized
- ❖ if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- ❖ when node obtains fresh frame, transmits in next slot
 - *if no collision:* node can send new frame in next slot
 - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros:

- ❖ single active node can continuously transmit at full rate of channel
- ❖ highly decentralized: only slots in nodes need to be in sync
- ❖ simple

Cons:

- ❖ collisions, wasting slots
- ❖ idle slots
- ❖ nodes may be able to detect collision in less than time to transmit packet
- ❖ clock synchronization

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

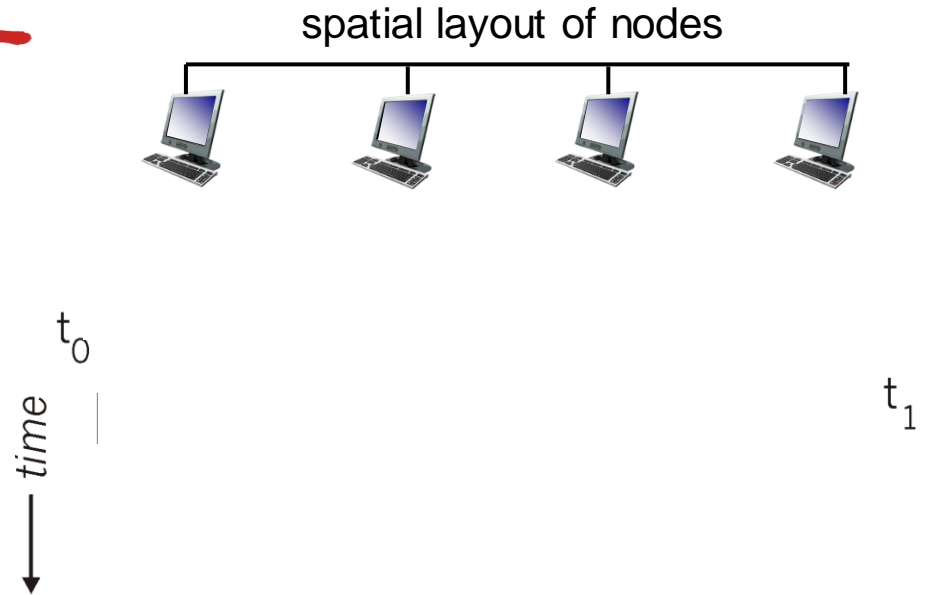
if channel sensed idle: transmit entire frame

- ❖ if channel sensed busy, defer transmission

- ❖ human analogy: don't interrupt others!

CSMA collisions

- ❖ collisions *can* still occur:
propagation delay means
two nodes may not hear
each other's
transmission
- ❖ collision: entire packet
transmission time
wasted
 - distance & propagation
delay play role in in
determining collision
probability

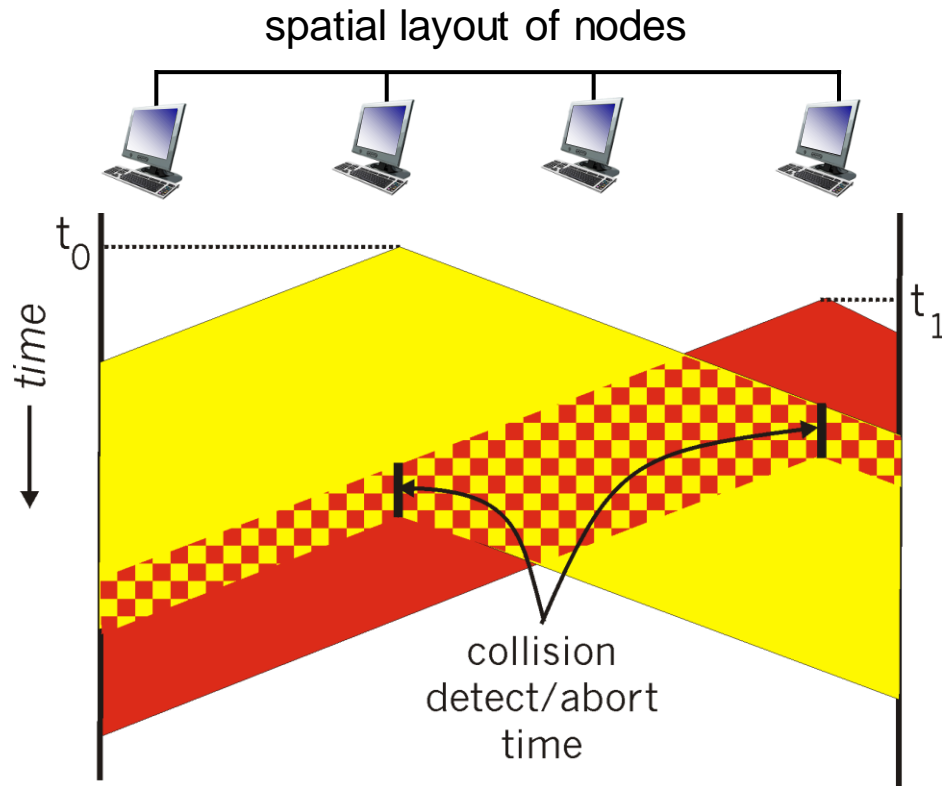


CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- ❖ collision detection:
 - easy in wired LANs: measure signal strengths, compare transmitted, received signals
 - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- ❖ human analogy: the polite conversationalist

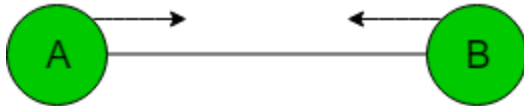
CSMA/CD (collision detection)



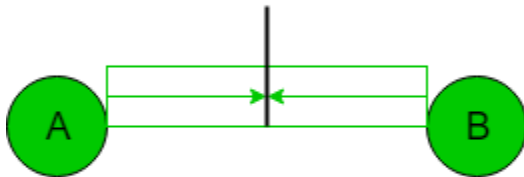
Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters *binary (exponential) backoff*:
 - after m th collision, NIC chooses K at random from $\{0, 1, 2, \dots, 2^m - 1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
 - longer backoff interval with more collisions

Ethernet CSMA/CD algorithm



At $t = 0$, both A and B start transmission

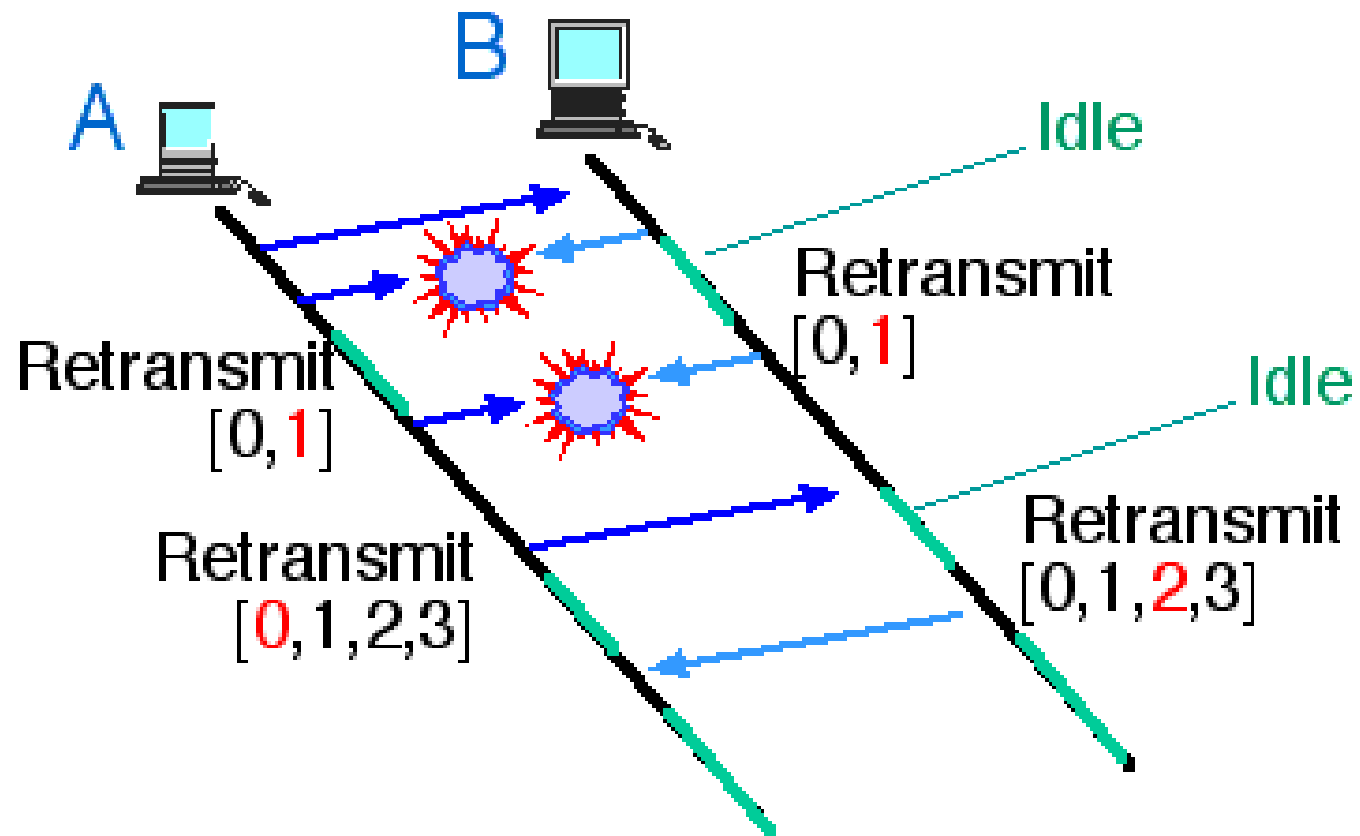


Packets of both A and B collide



Both stations A and B detect collision

Ethernet CSMA/CD algorithm



Summary of MAC protocols

- ❖ *channel partitioning*, by time, frequency or code
 - Time Division, Frequency Division
- ❖ *random access* (dynamic),
 - ALOHA, S-ALOHA, CSMA, CSMA/CD
 - carrier sensing: easy in some technologies (wire), hard in others (wireless)
 - CSMA/CD used in Ethernet
 - CSMA/CA used in 802.11
- ❖ *taking turns*
 - polling from central site, token passing
 - bluetooth, FDDI, token ring

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

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)

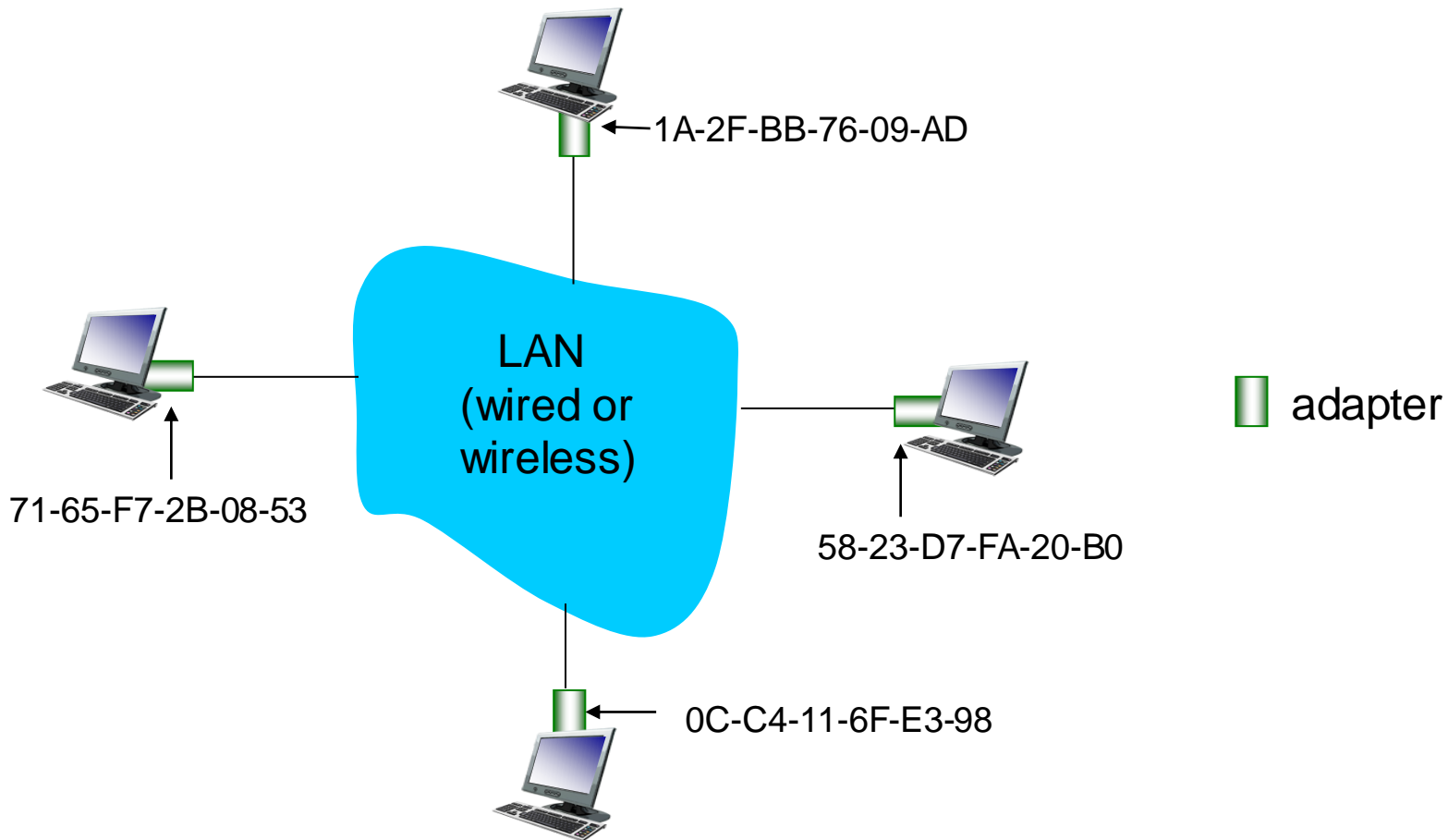
Exercise :View MAC Address

- ❖ MAC Address บางครั้งเรียก Ethernet Address, Hardware Address, Physical Address
- ❖ ใช้คำสั่ง ipconfig /all

```
Connection-specific DNS Suffix . :  
Description . . . . . : Realtek PCIe GBE Family Controller  
Physical Address. . . . . : 18-66-DA-02-EB-60  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . : Yes  
IPv6 Address. . . . . : 2403:6200:88a0:d78:55e6:effb:b759:a285(Preferred)  
Temporary IPv6 Address. . . . . : 2403:6200:88a0:d78:e481:6f76:dcb4:2431(Preferred)  
Link-local IPv6 Address . . . . . : fe80::55e6:effb:b759:a285%25(Preferred)  
IPv4 Address. . . . . : 192.168.1.4(Preferred)  
Subnet Mask . . . . . : 255.255.255.0
```

LAN addresses and ARP

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

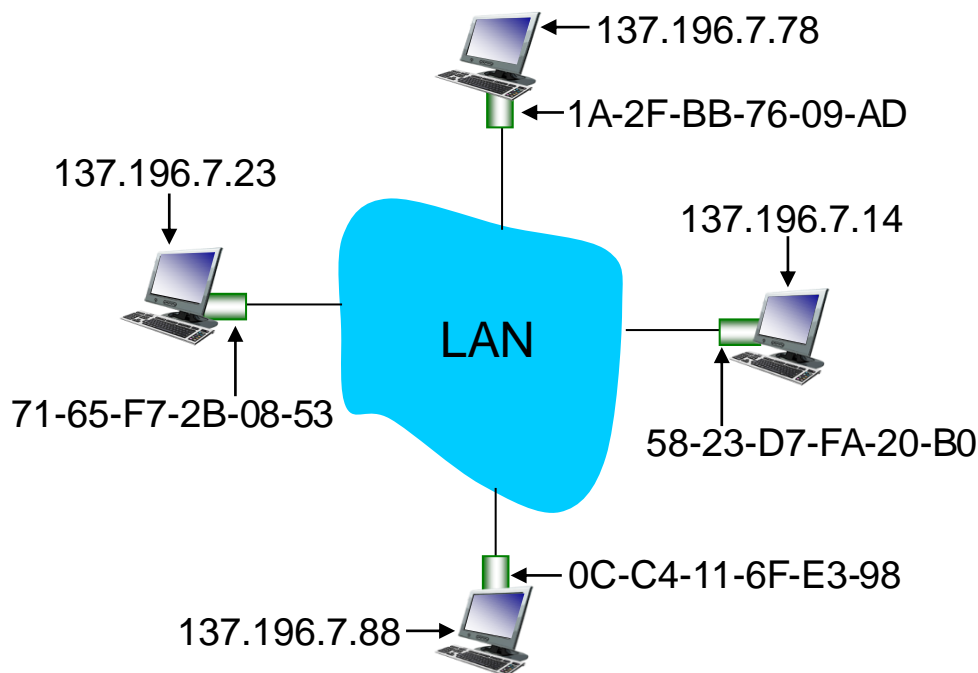


LAN addresses (more)

- ❖ MAC address allocation administered by IEEE
- ❖ manufacturer buys portion of MAC address space (to assure uniqueness)
- ❖ analogy:
 - MAC address: like Citizen ID 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?



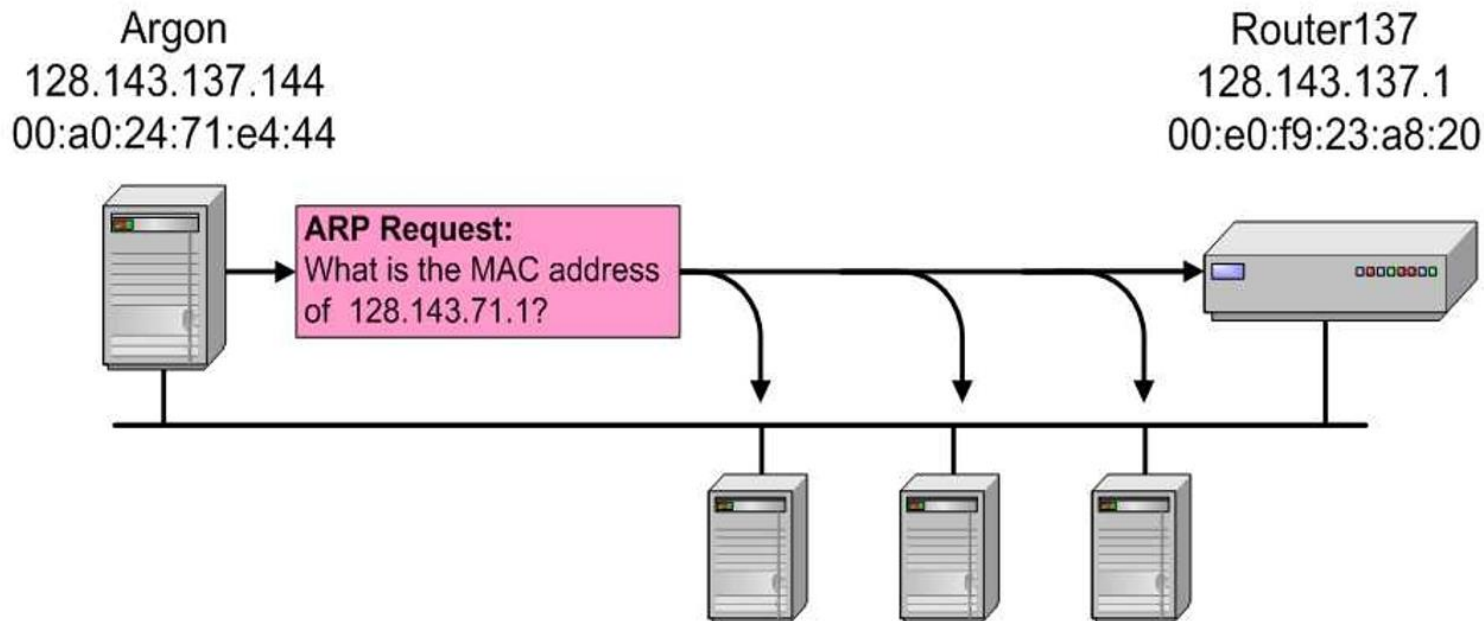
ARP table: each IP node (host, router) on LAN has table

- 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: address resolution protocol

ARP Request:

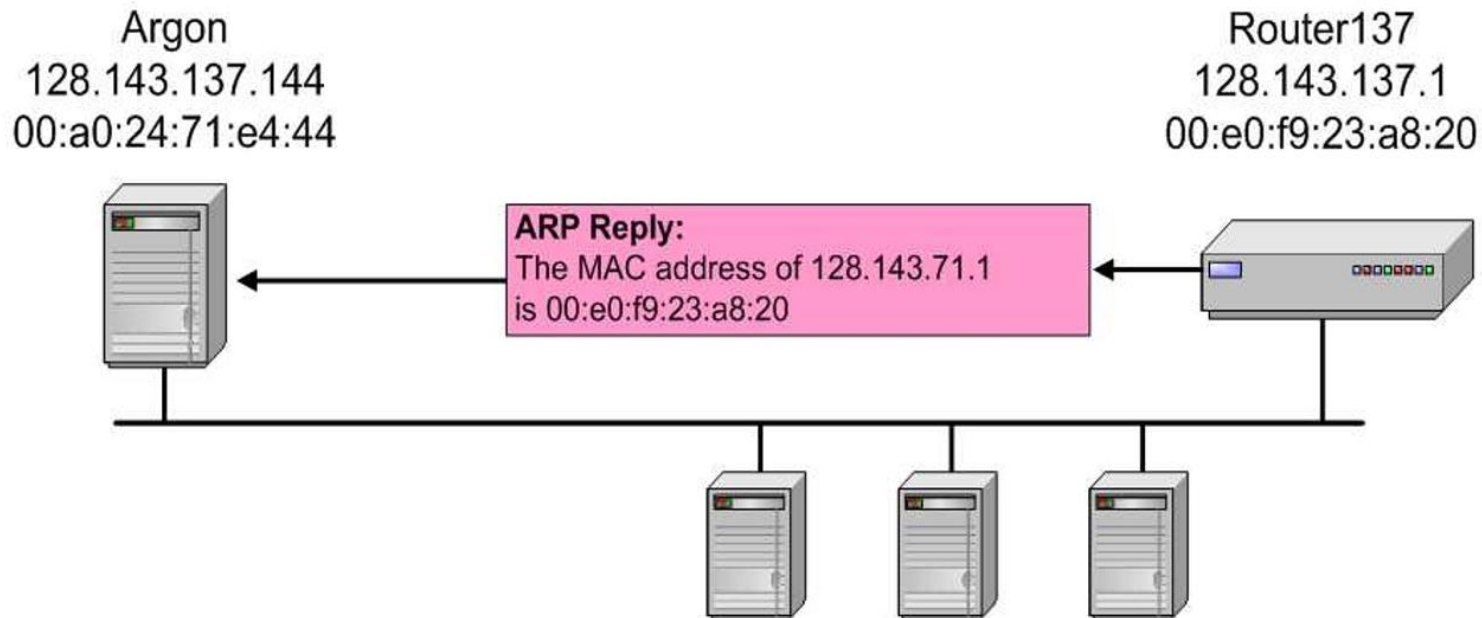
Argon broadcasts an ARP request to all stations on the network: **“What is the hardware address of Router137?”**



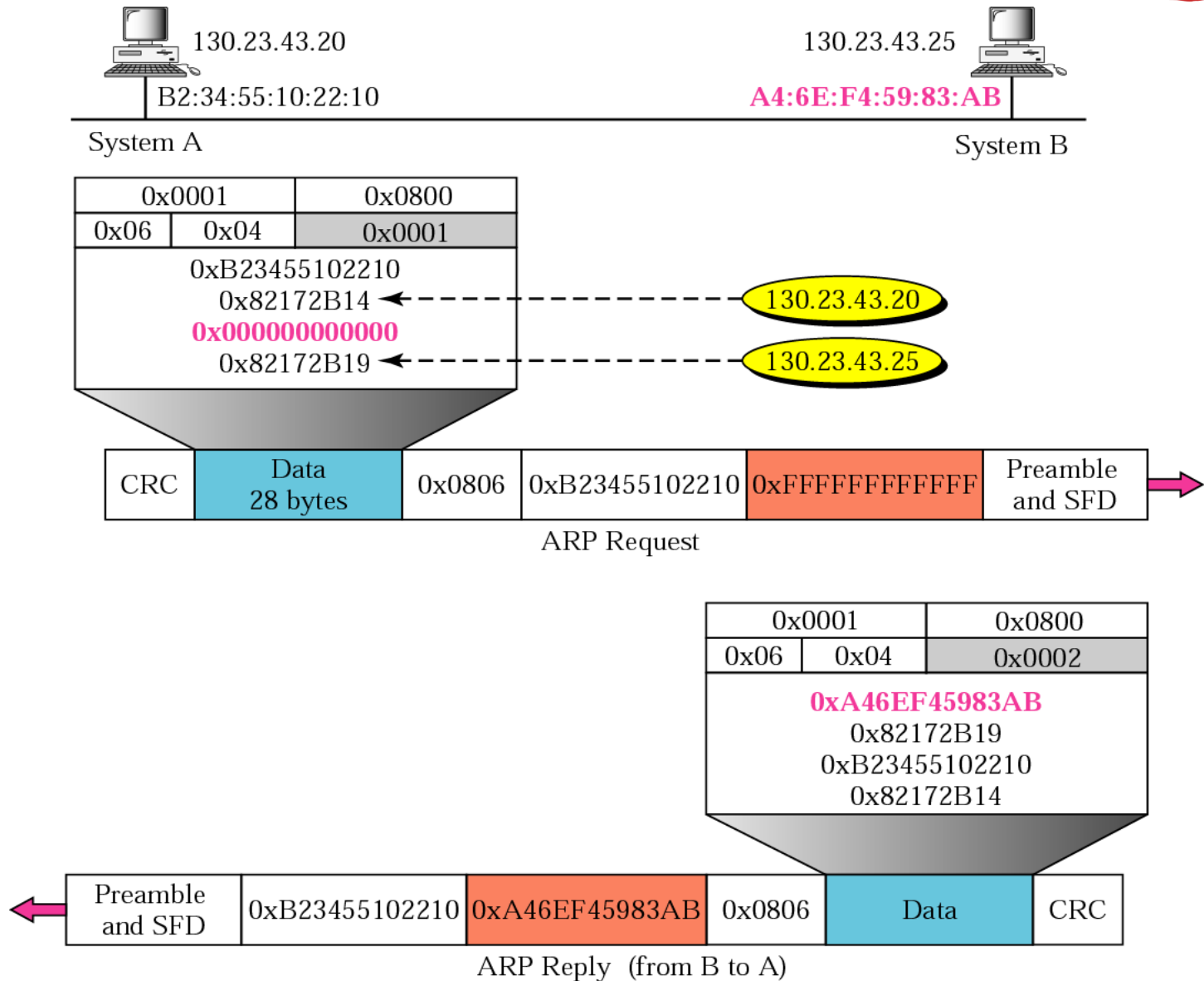
ARP: address resolution protocol

ARP Reply:

Router 137 responds with an ARP Reply which contains the hardware address



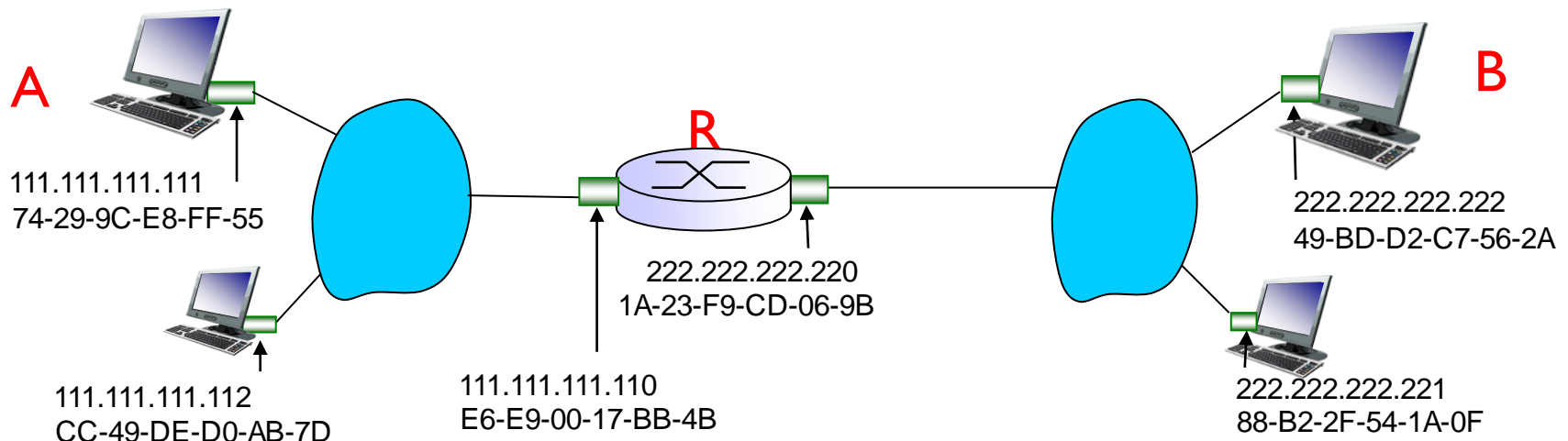
ARP Example



Addressing: routing to another LAN

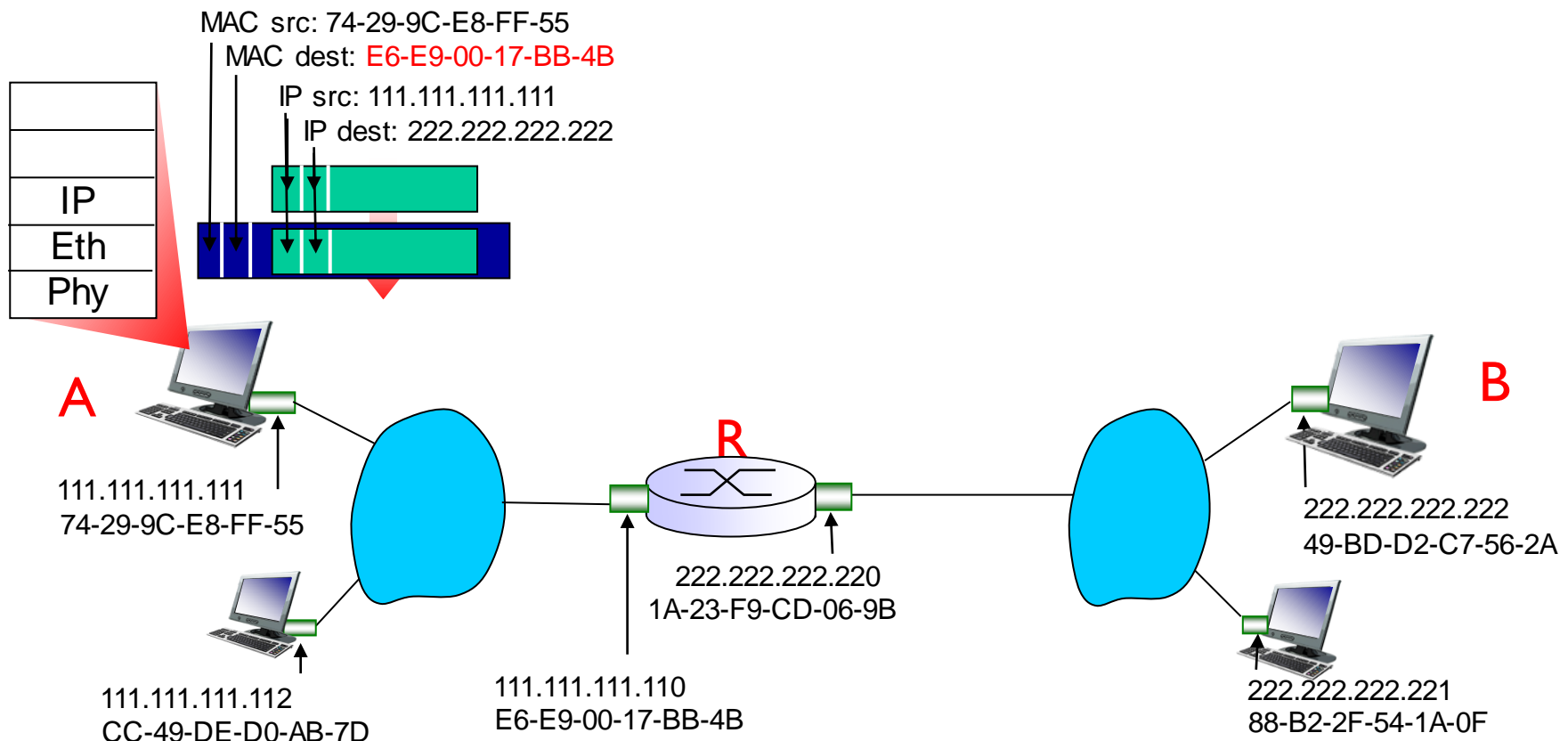
walkthrough: **send datagram from A to B via R**

- 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?)



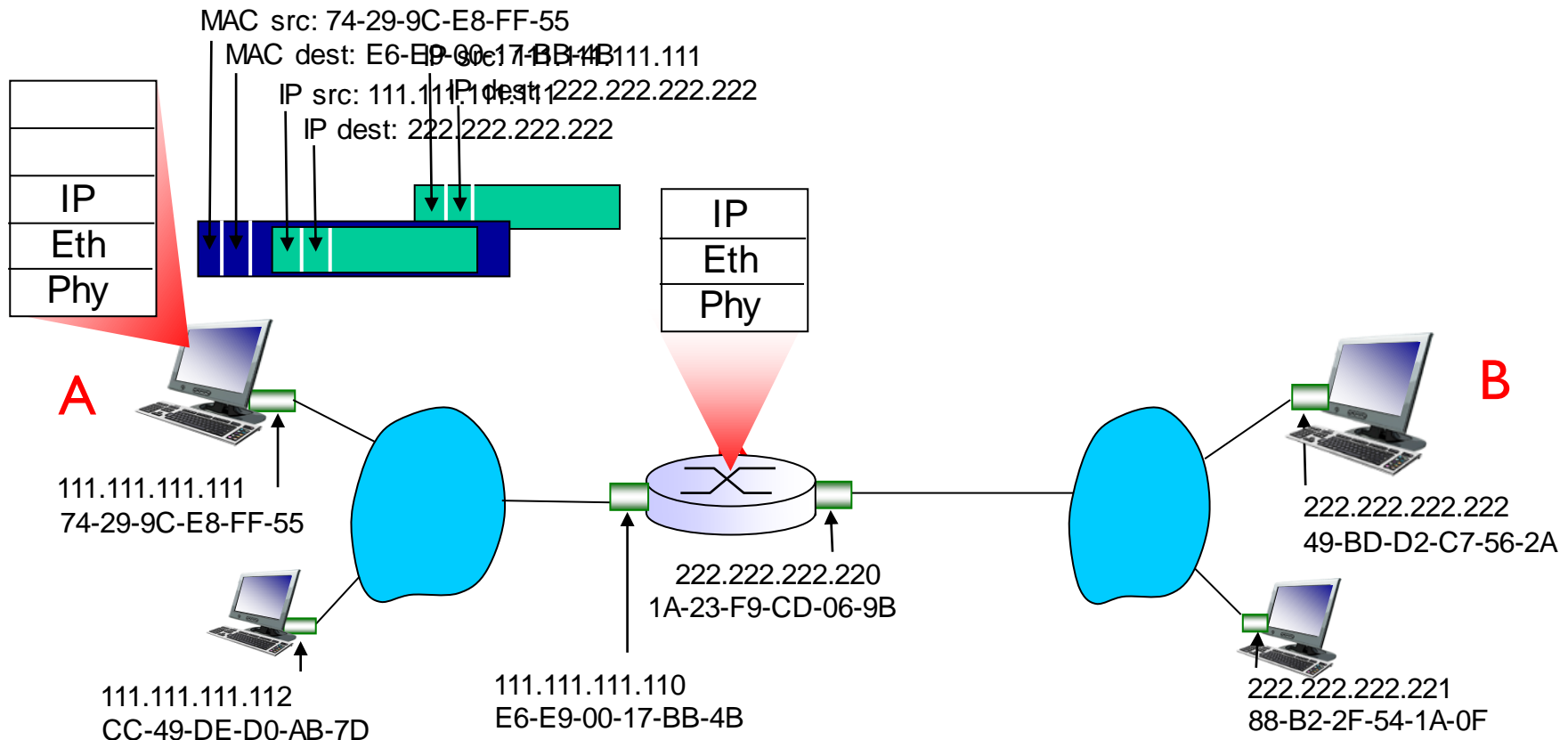
Addressing: routing to another LAN

- ❖ 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



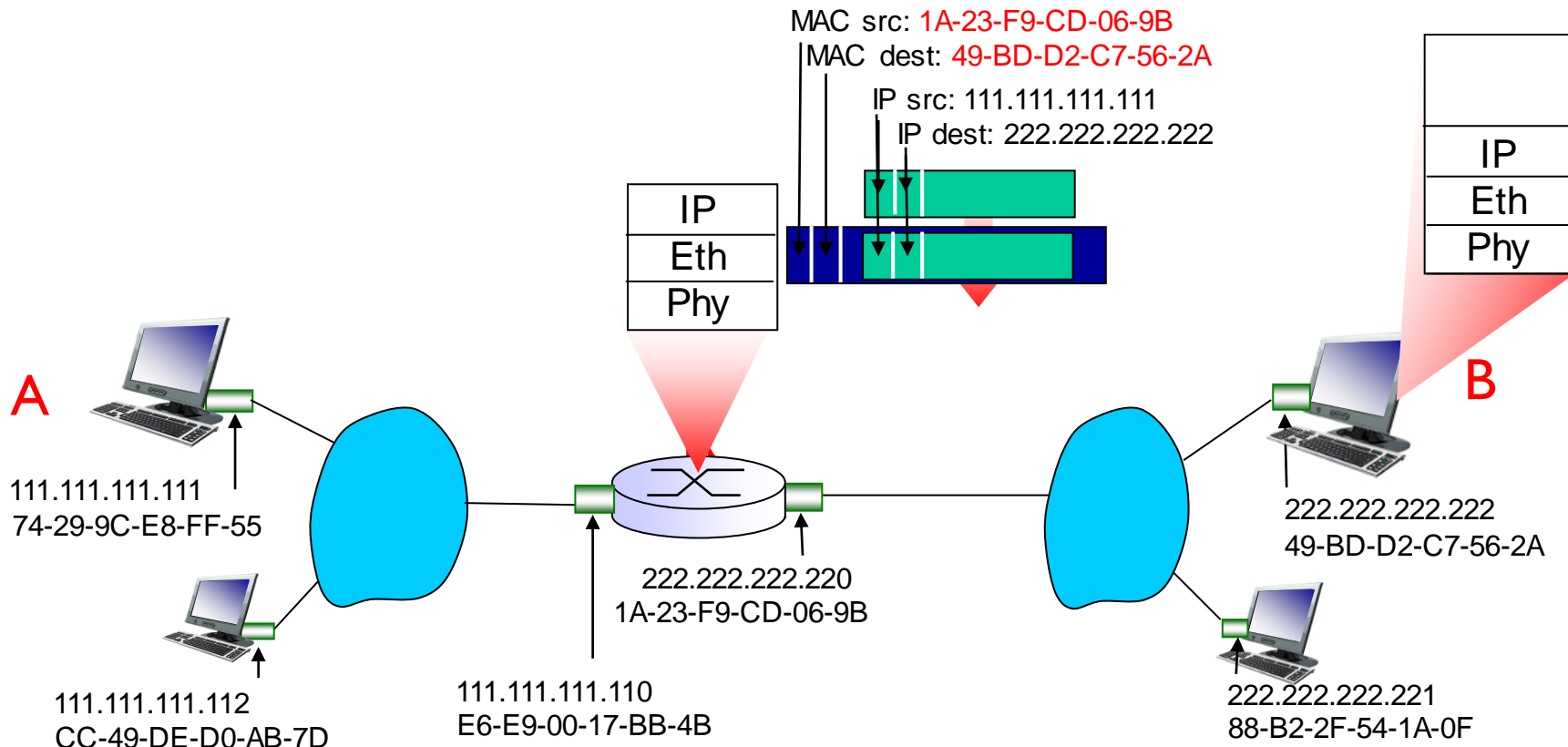
Addressing: routing to another LAN

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



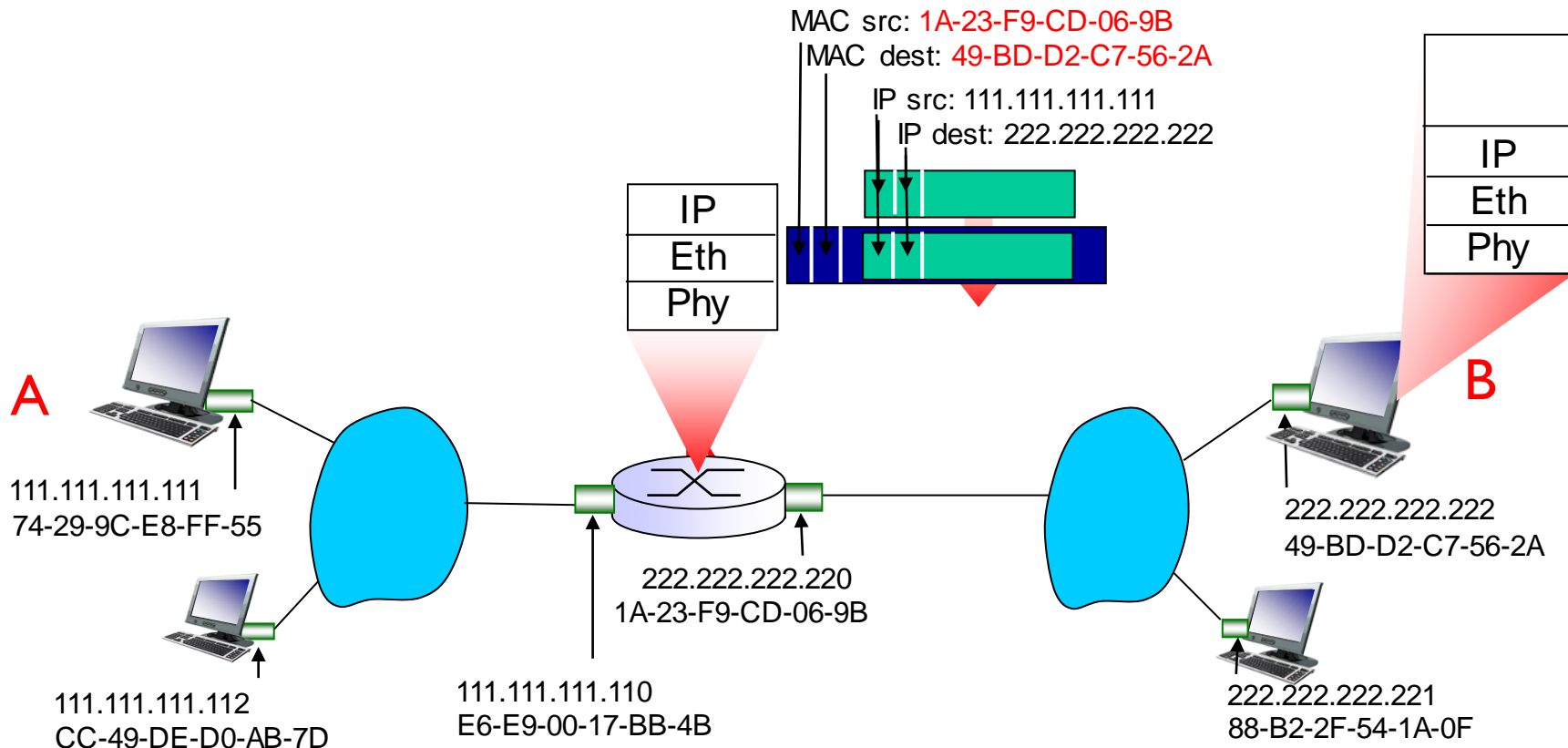
Addressing: routing to another LAN

- ❖ 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



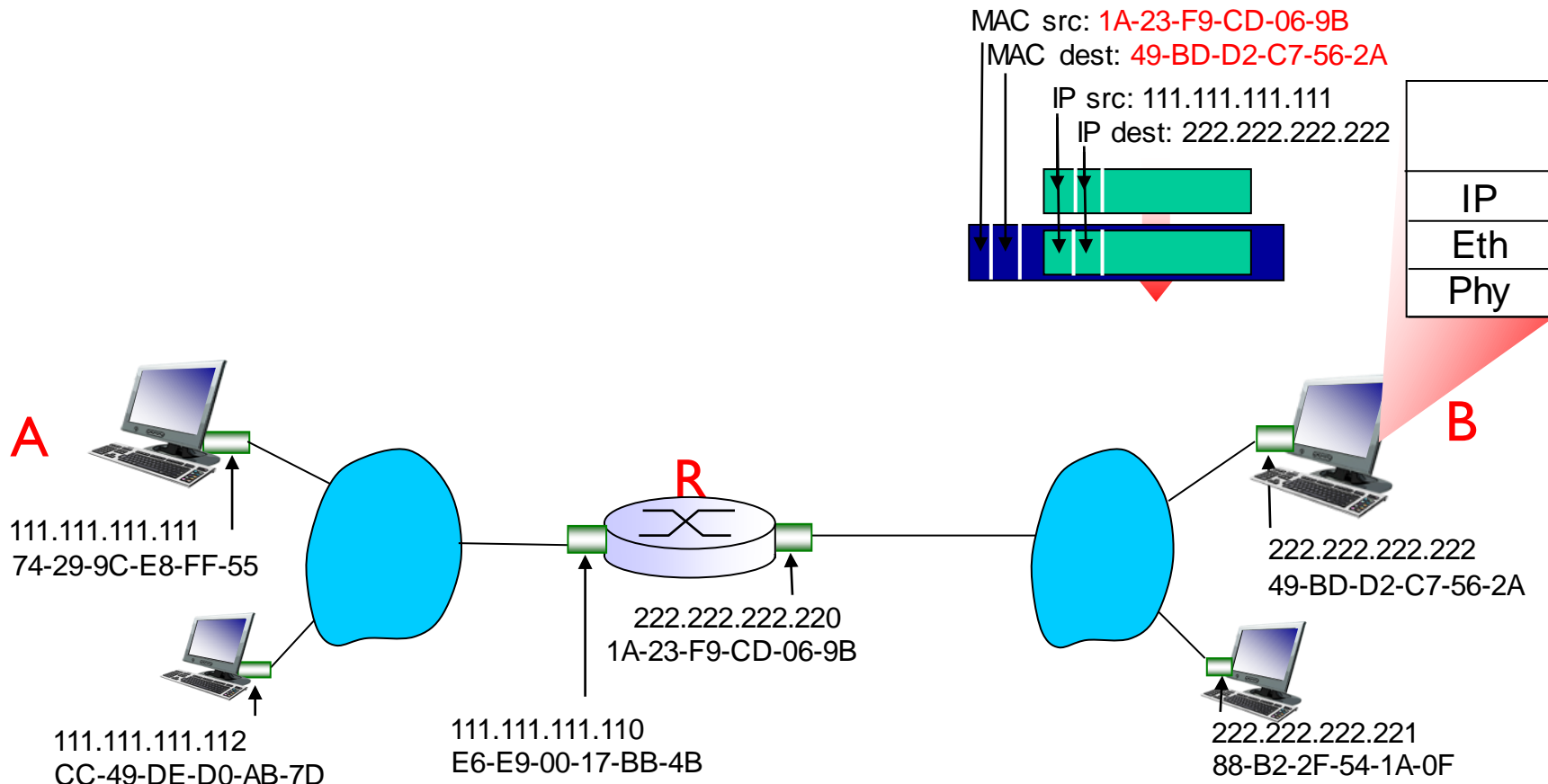
Addressing: routing to another LAN

- ❖ 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



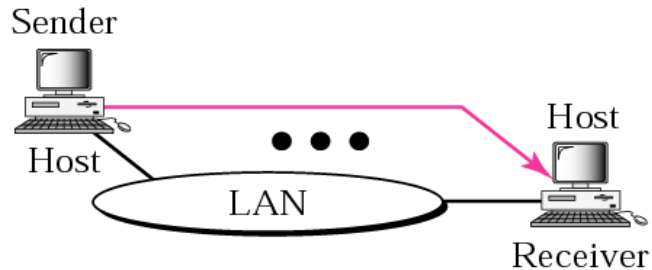
Addressing: routing to another LAN

- ❖ 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



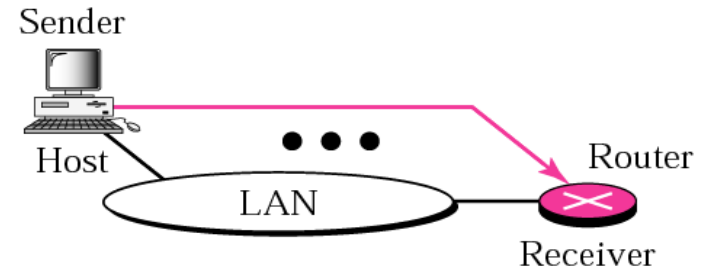
4 Case using ARP

Target IP address:
Destination address in the IP datagram



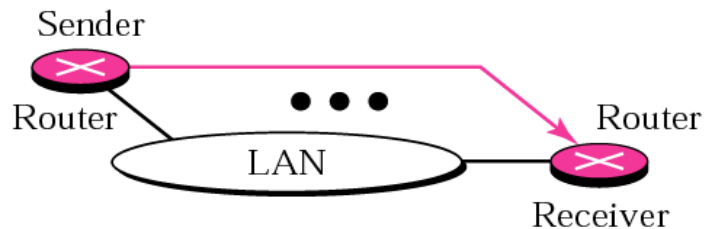
Case 1. A host has a packet to send to another host on the same network.

Target IP address:
IP address of a router



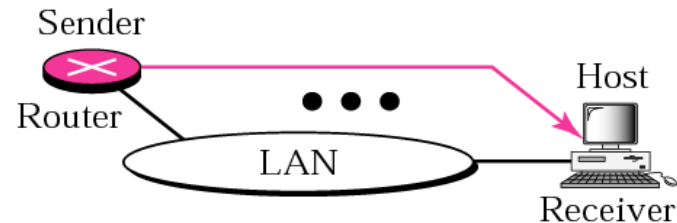
Case 2. A host wants to send a packet to another host on another network.
It must first be delivered to a router.

Target IP address:
IP address of the appropriate router
found in the routing table



Case 3. A router receives a packet to be sent to a host on another network.
It must first be delivered to the appropriate router.

Target IP address:
Destination address in the IP datagram



Case 4. A router receives a packet to be sent to a host on the same network.

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

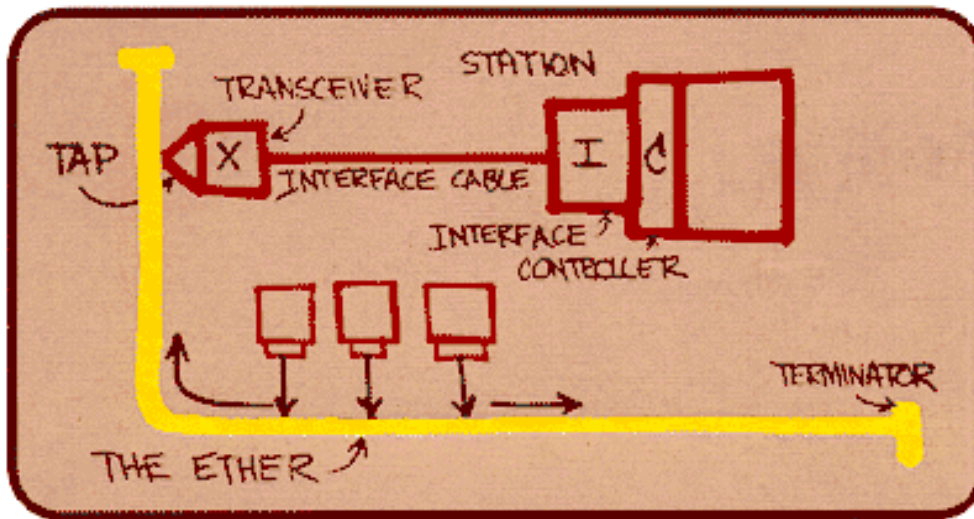
5.6 data center
networking

5.7 a day in the life of a
web request

Ethernet

“dominant” wired LAN technology:

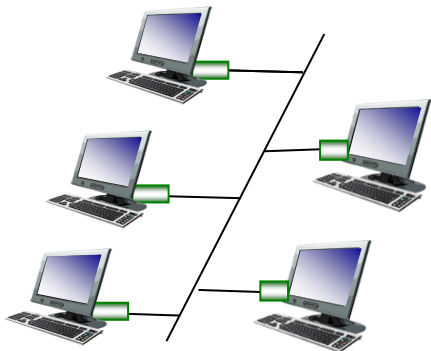
- ❖ cheap \$20 for NIC
- ❖ first widely used LAN technology
- ❖ simpler, cheaper than token LANs and ATM
- ❖ kept up with speed race: 10 Mbps – 10 Gbps



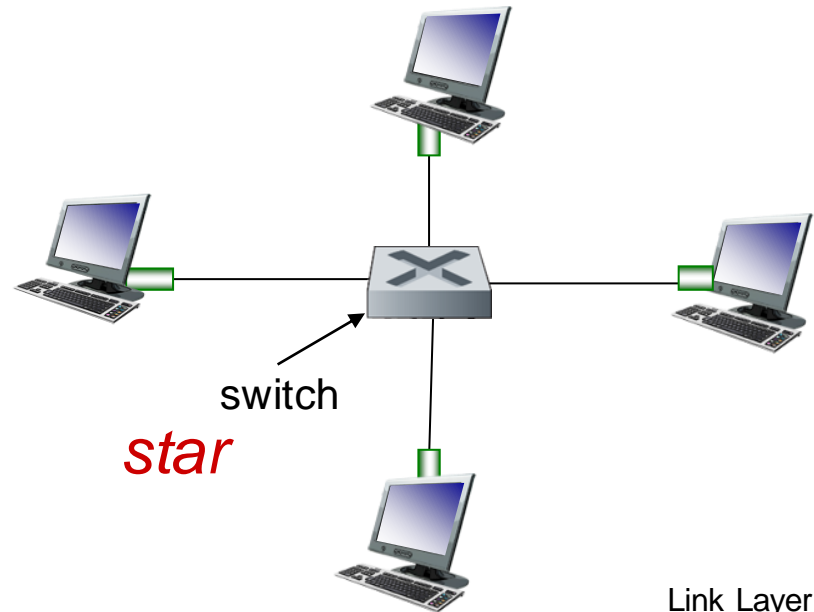
Metcalfe's Ethernet sketch

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

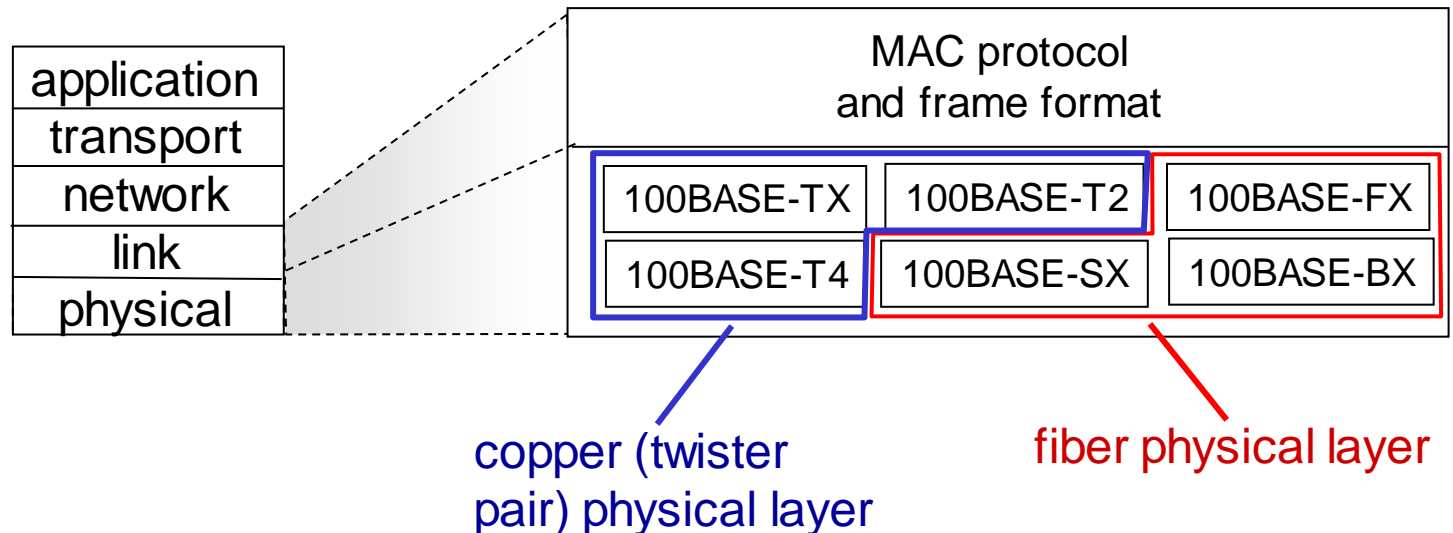


Ethernet: unreliable, connectionless

- ❖ *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

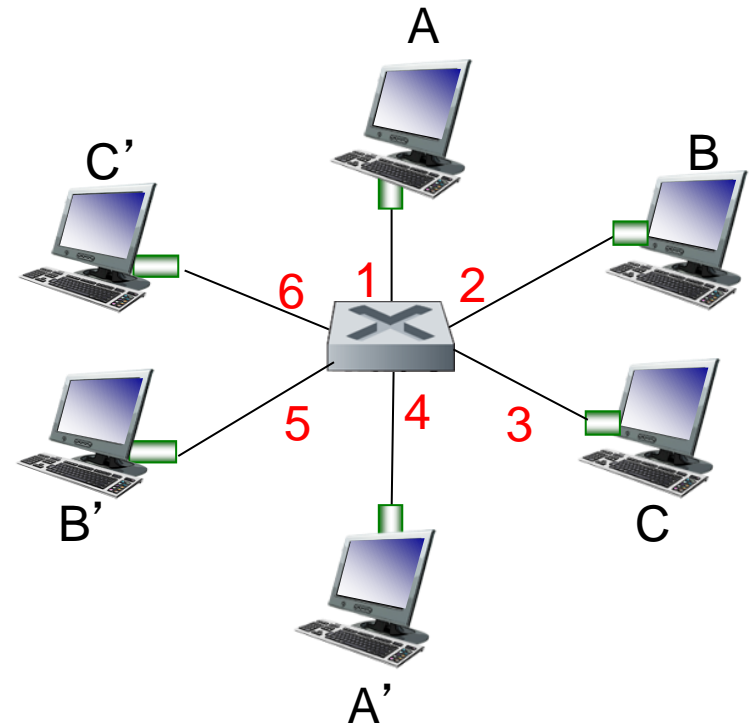
5.7 a day in the life of a
web request

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

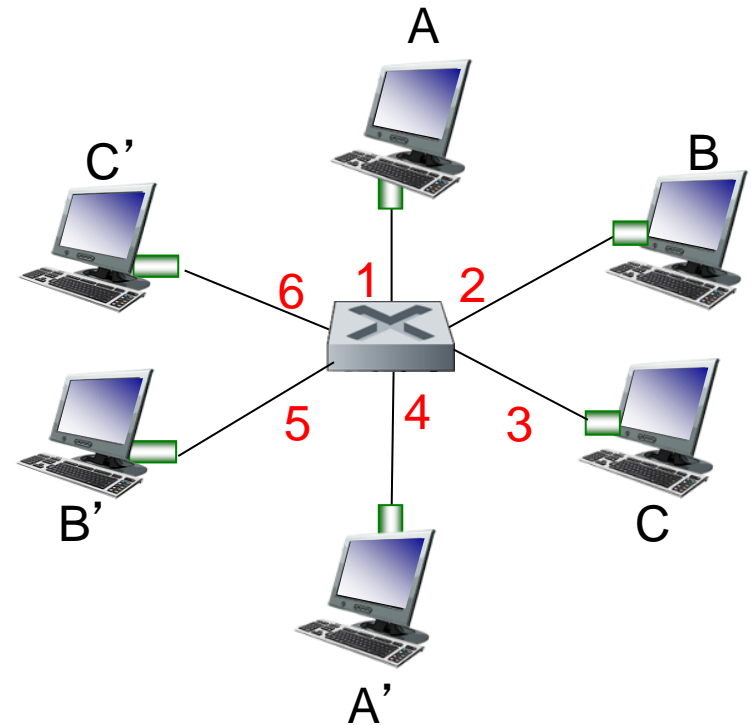
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

❖ **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?

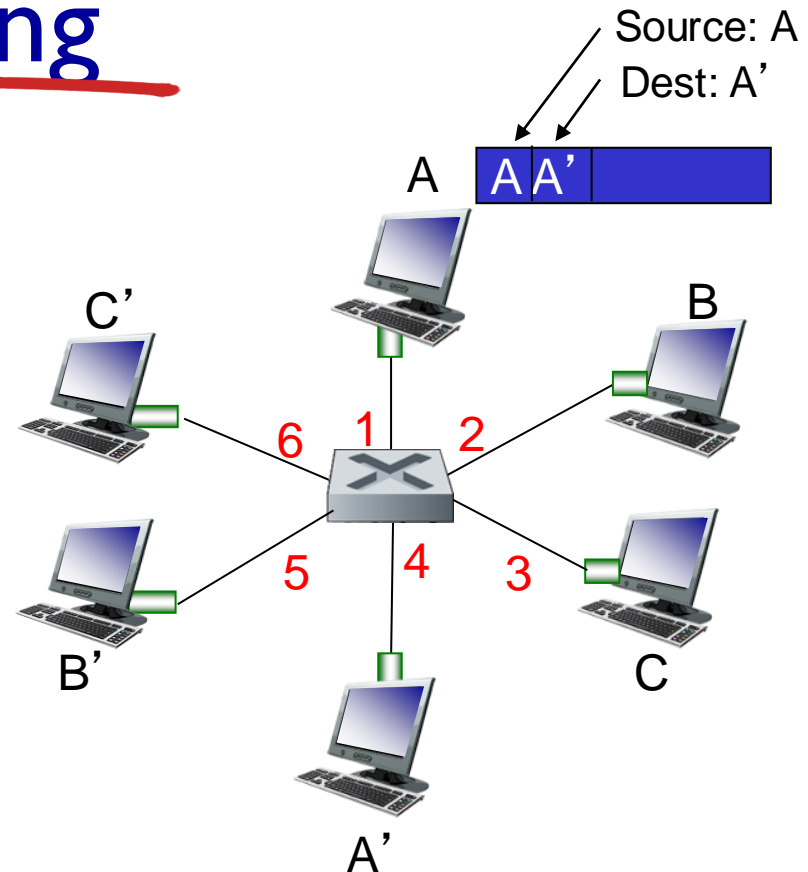
- something like a routing protocol?



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)*

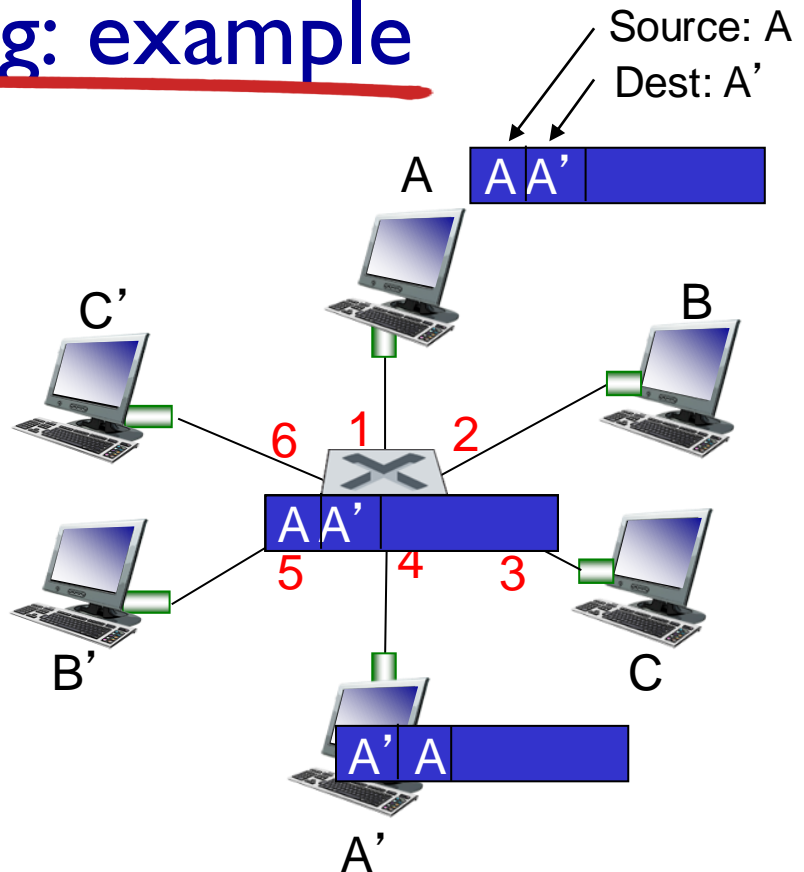
Switch: frame filtering/forwarding

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*

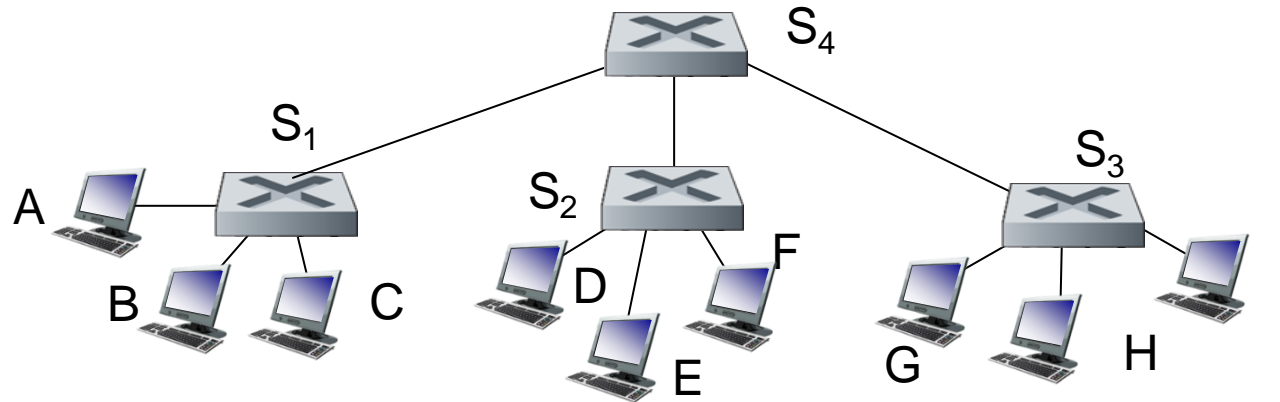


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

Interconnecting switches

- ❖ switches can be connected together

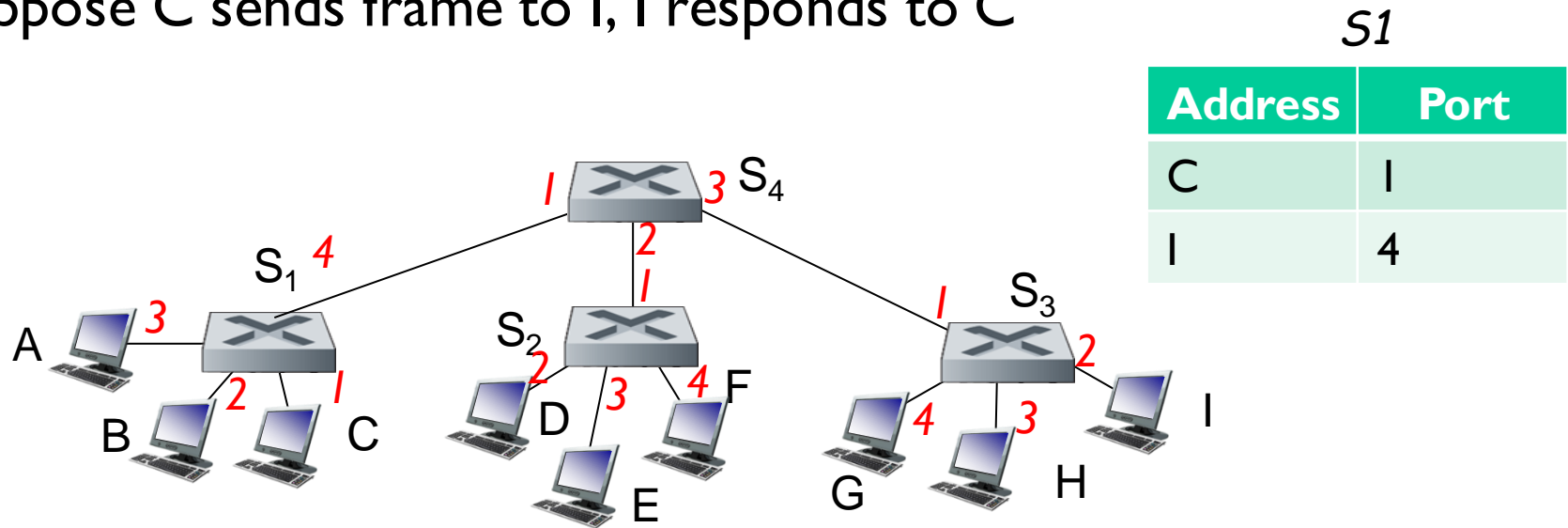


Q: sending from A to G - how does S₁ know to forward frame destined to F via S₄ and S₃?

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

Example : Self-learning multi-switch

Suppose C sends frame to I, I responds to C

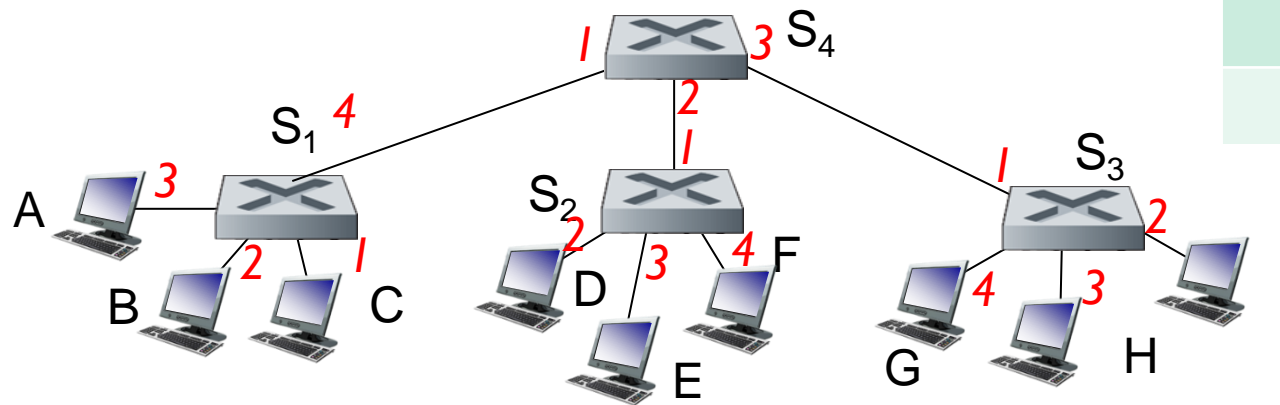


❖ Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

<i>S2</i>		<i>S3</i>		<i>S4</i>	
Address	Port	Address	Port	Address	Port
C	I	C	I	C	I
		I	2	I	3

Exercise : Self-learning multi-switch

A sends frame to E, E sends frame to H



S1

Address	Port

❖ Q: show switch tables and packet forwarding in S₁, S₂, S₃, S₄

S2

Address	Port

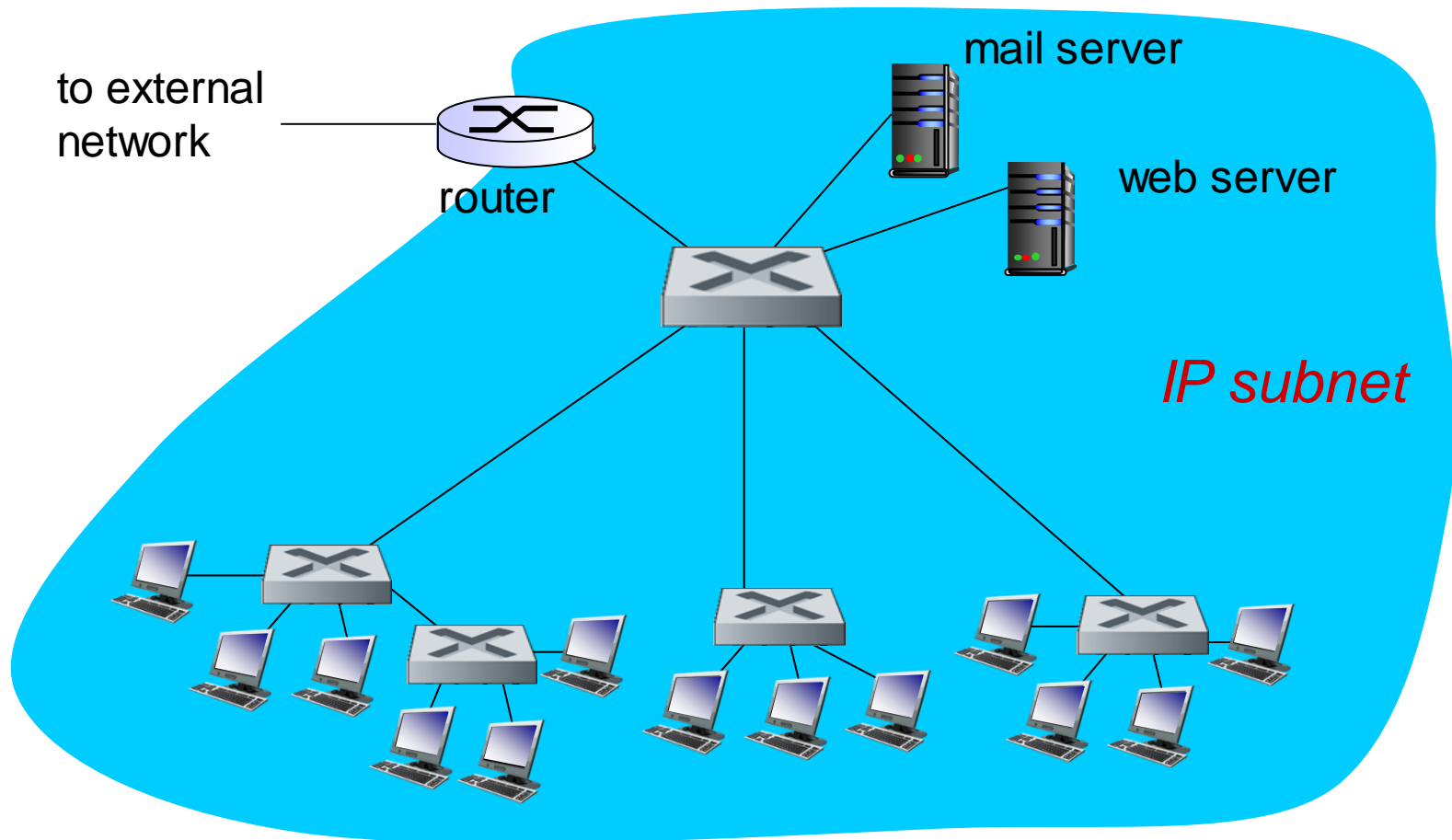
S3

Address	Port

S4

Address	Port

Institutional network



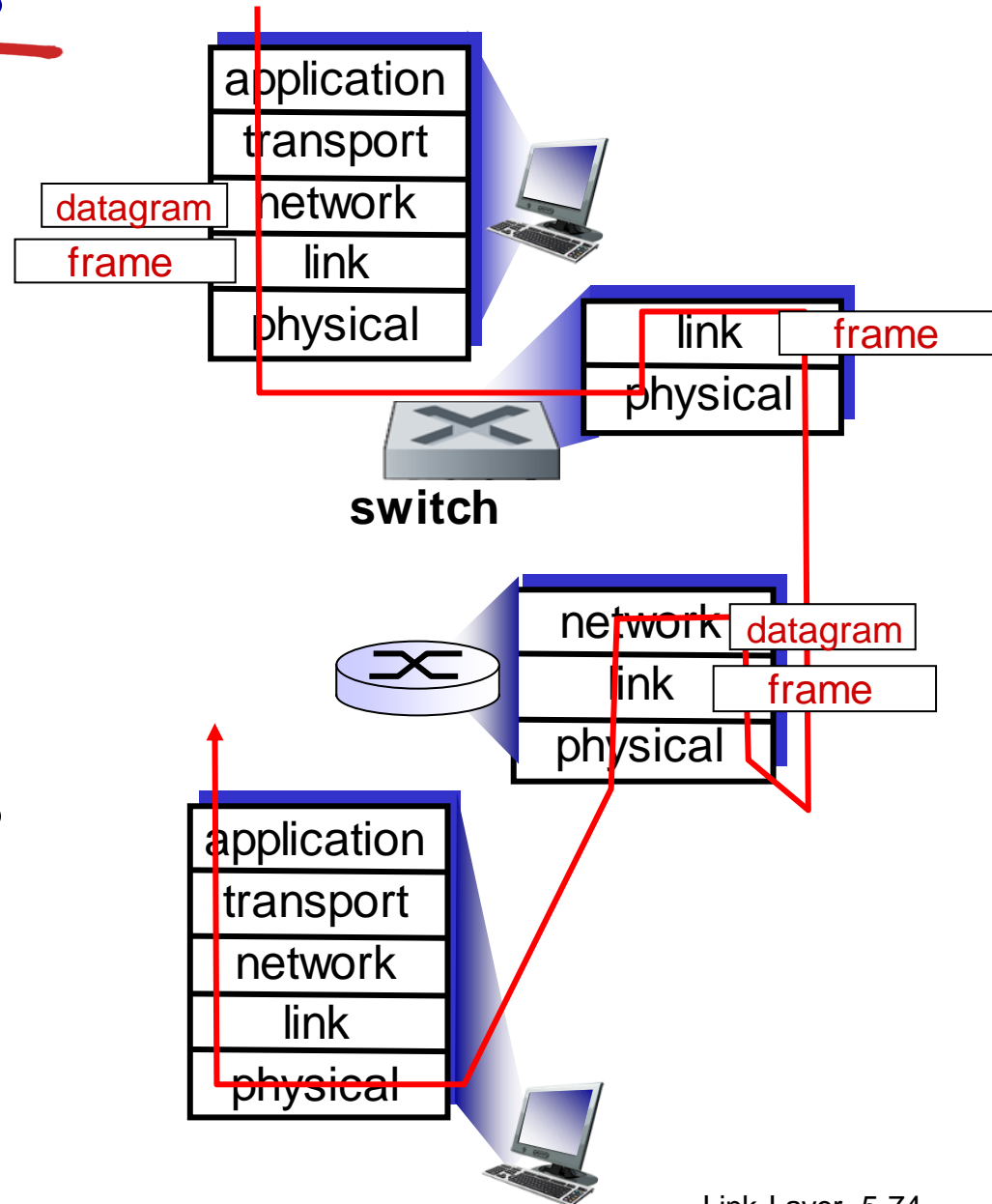
Switches vs. routers

both are store-and-forward:

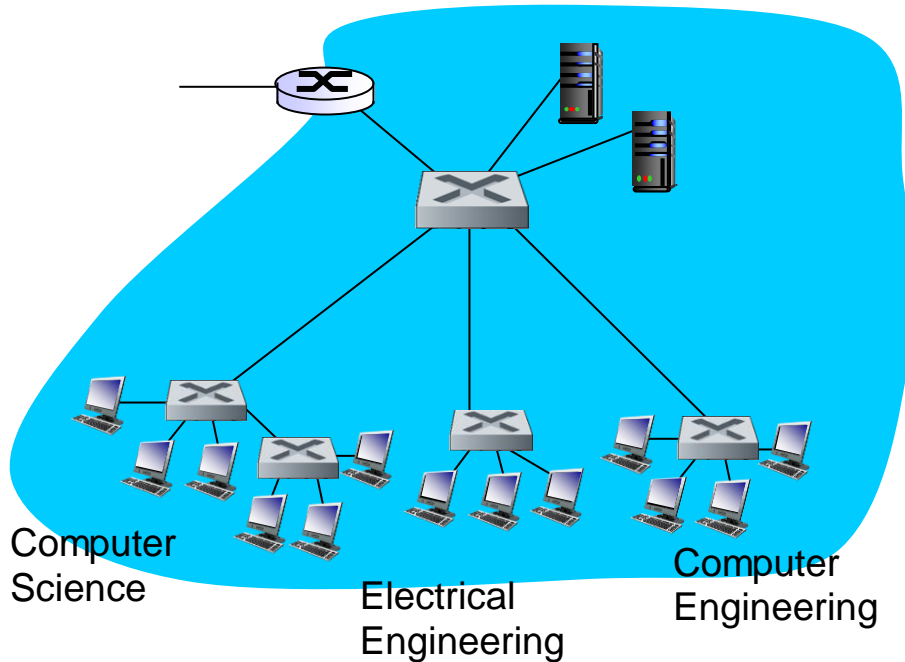
- **routers:** network-layer devices (examine network-layer headers)
- **switches:** link-layer devices (examine link-layer headers)

both have forwarding tables:

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



VLANs: motivation



consider:

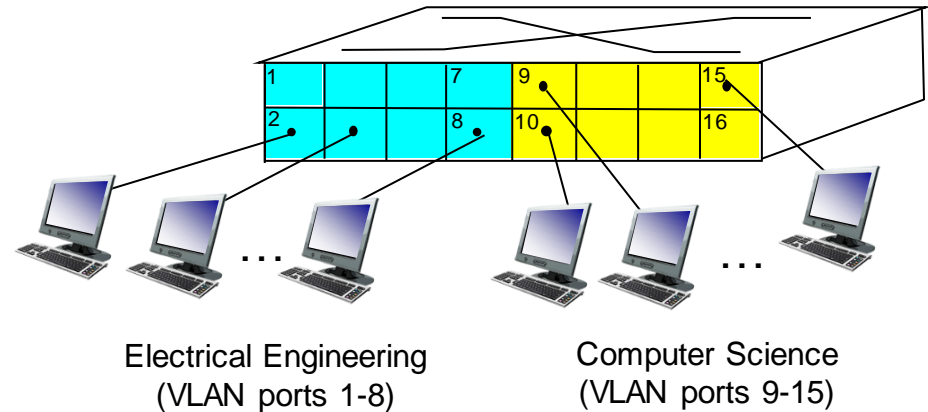
- ❖ CS user moves office to EE, but wants connect to CS switch?
- ❖ single broadcast domain:
 - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
 - security/privacy, efficiency issues

VLANs

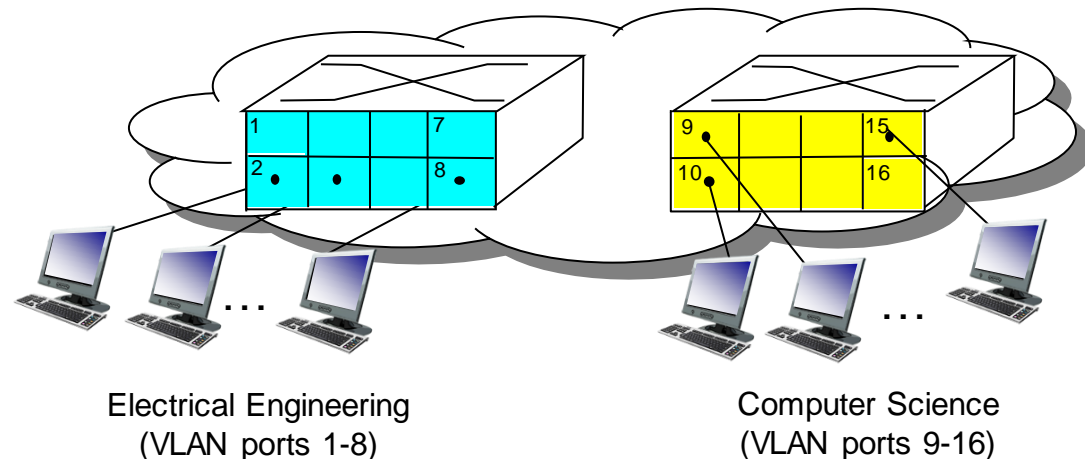
Virtual Local Area Network

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

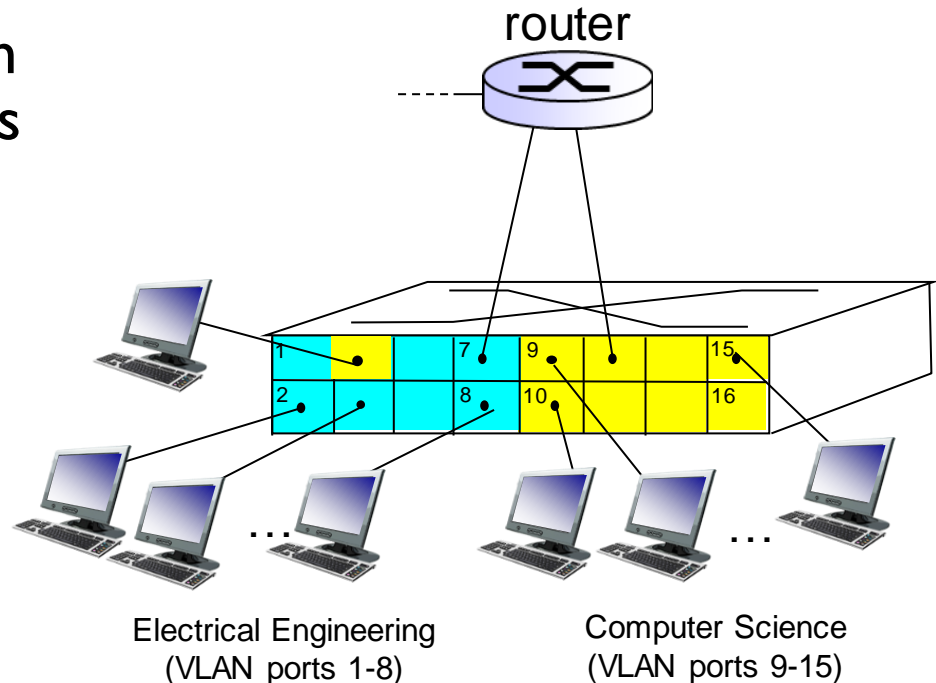


... operates as *multiple* virtual switches

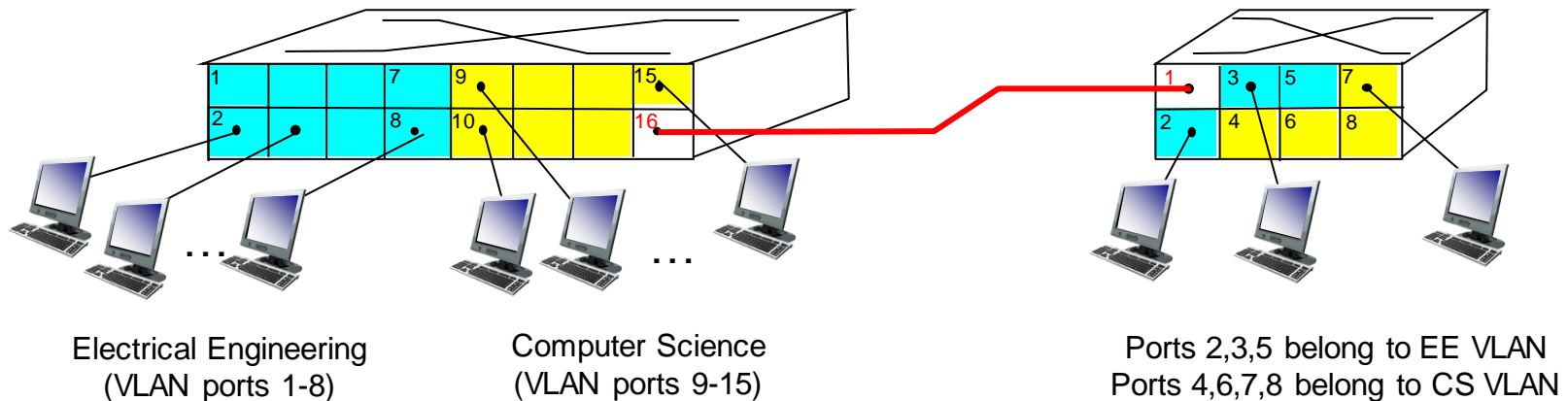


Port-based VLAN

- ❖ **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

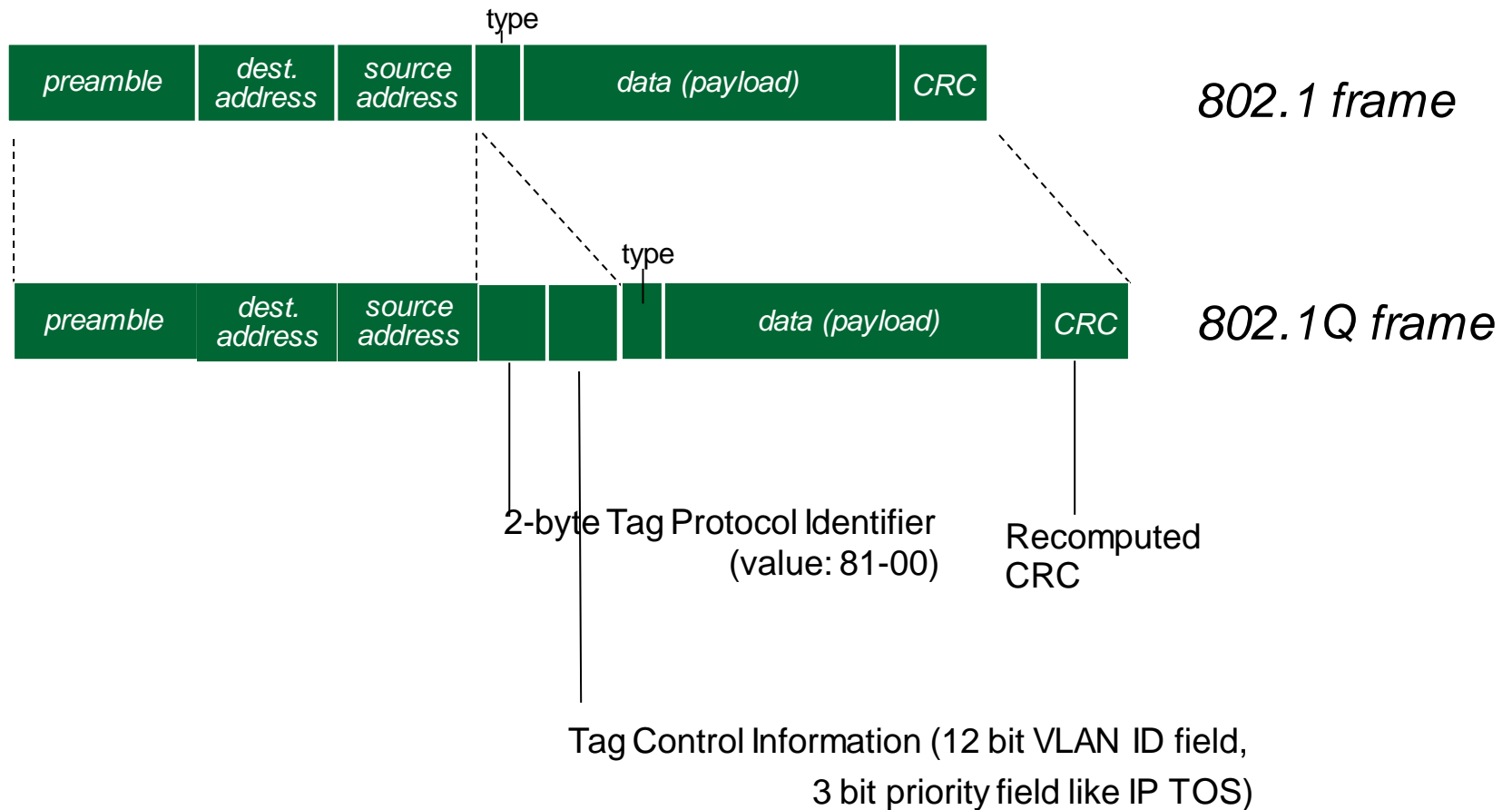


VLANs spanning multiple switches



- ❖ **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



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

Data center networks

- ❖ 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
 - e-business (e.g. Amazon)
 - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
 - search engines, data mining (e.g., Google)
- ❖ challenges:
 - multiple applications, each serving massive numbers of clients
 - managing/balancing load, avoiding processing, networking, data bottlenecks

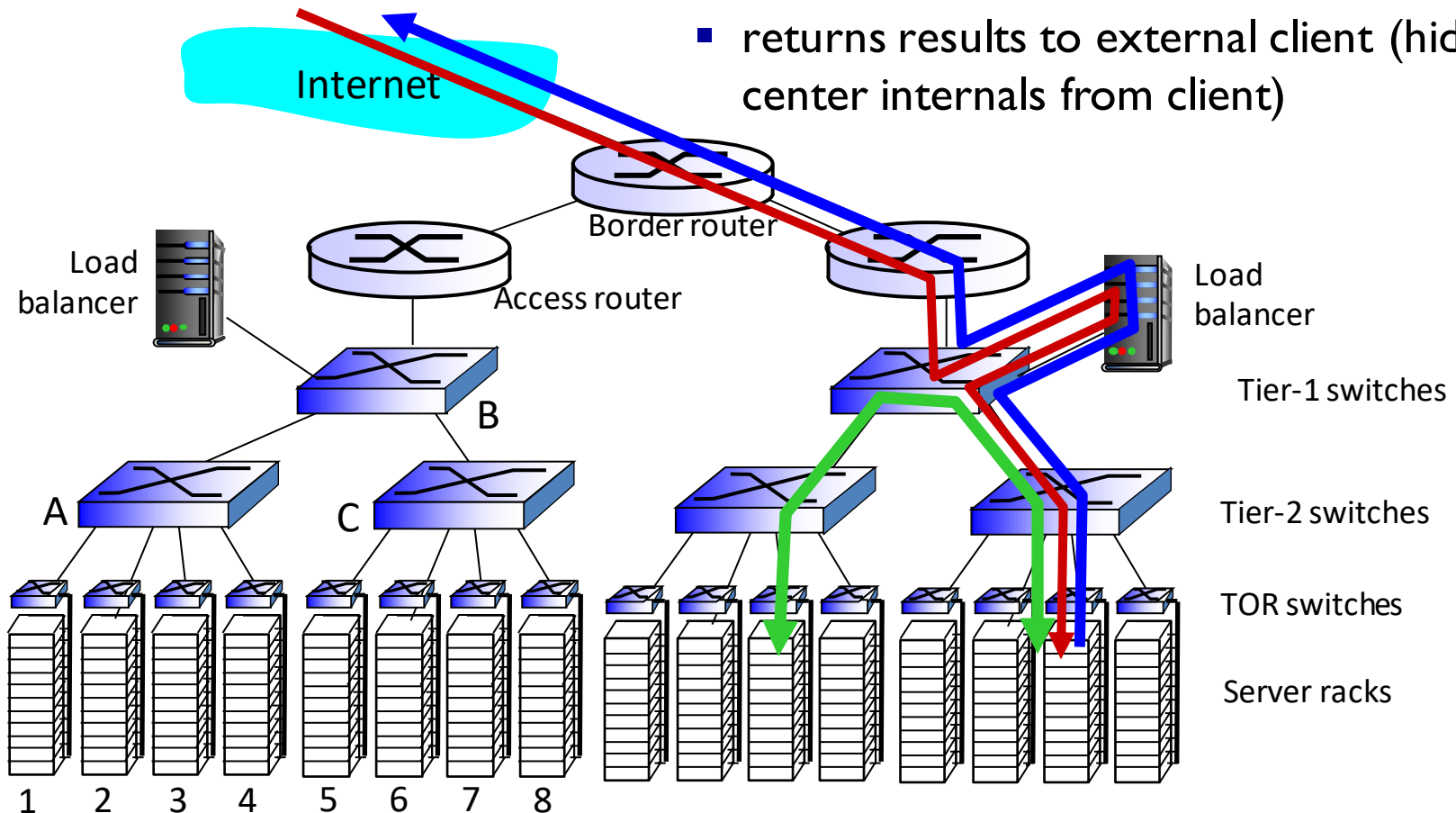


Inside a 40-ft Microsoft container,
Chicago data center

Data center networks

load balancer: application-layer routing

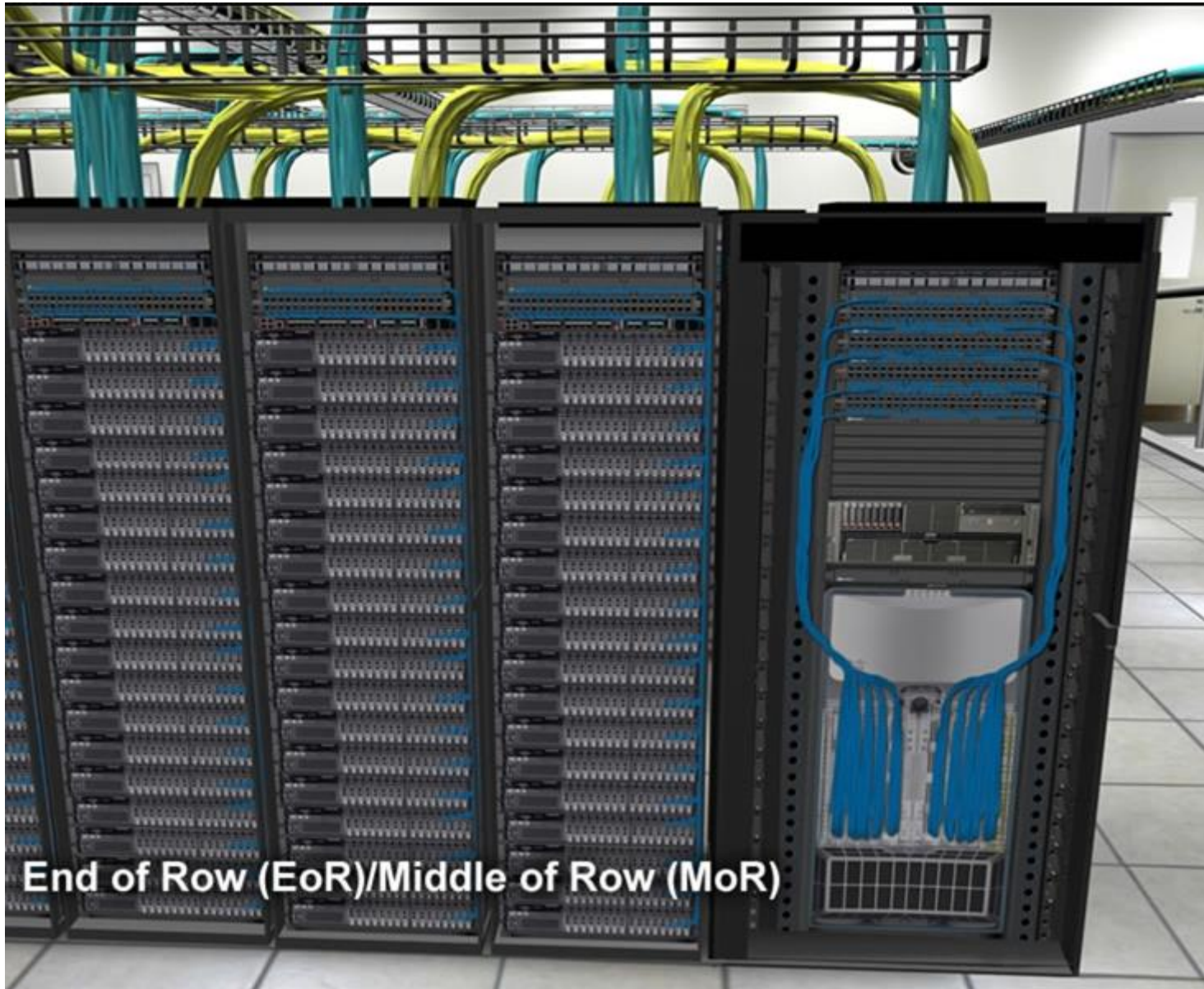
- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Top of rack

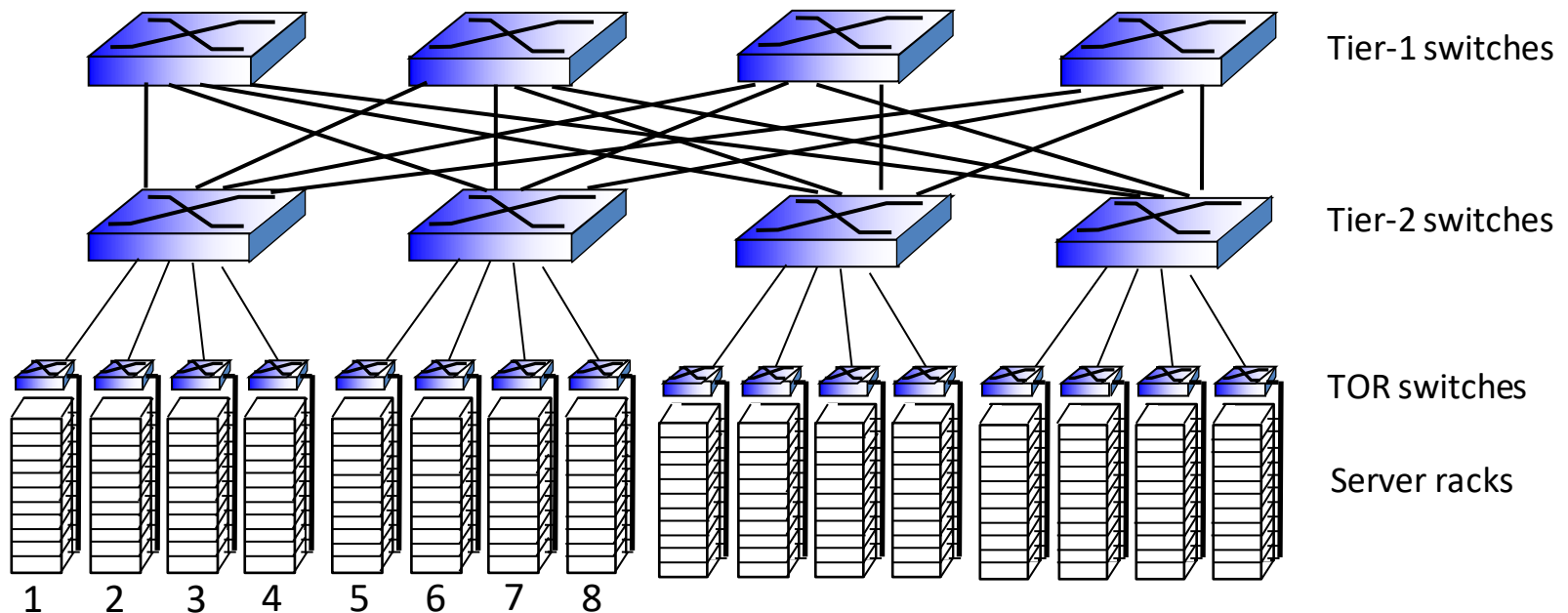


End of Row



Data center networks

- ❖ rich interconnection among switches, racks:
 - increased throughput between racks (multiple routing paths possible)
 - increased reliability via redundancy



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

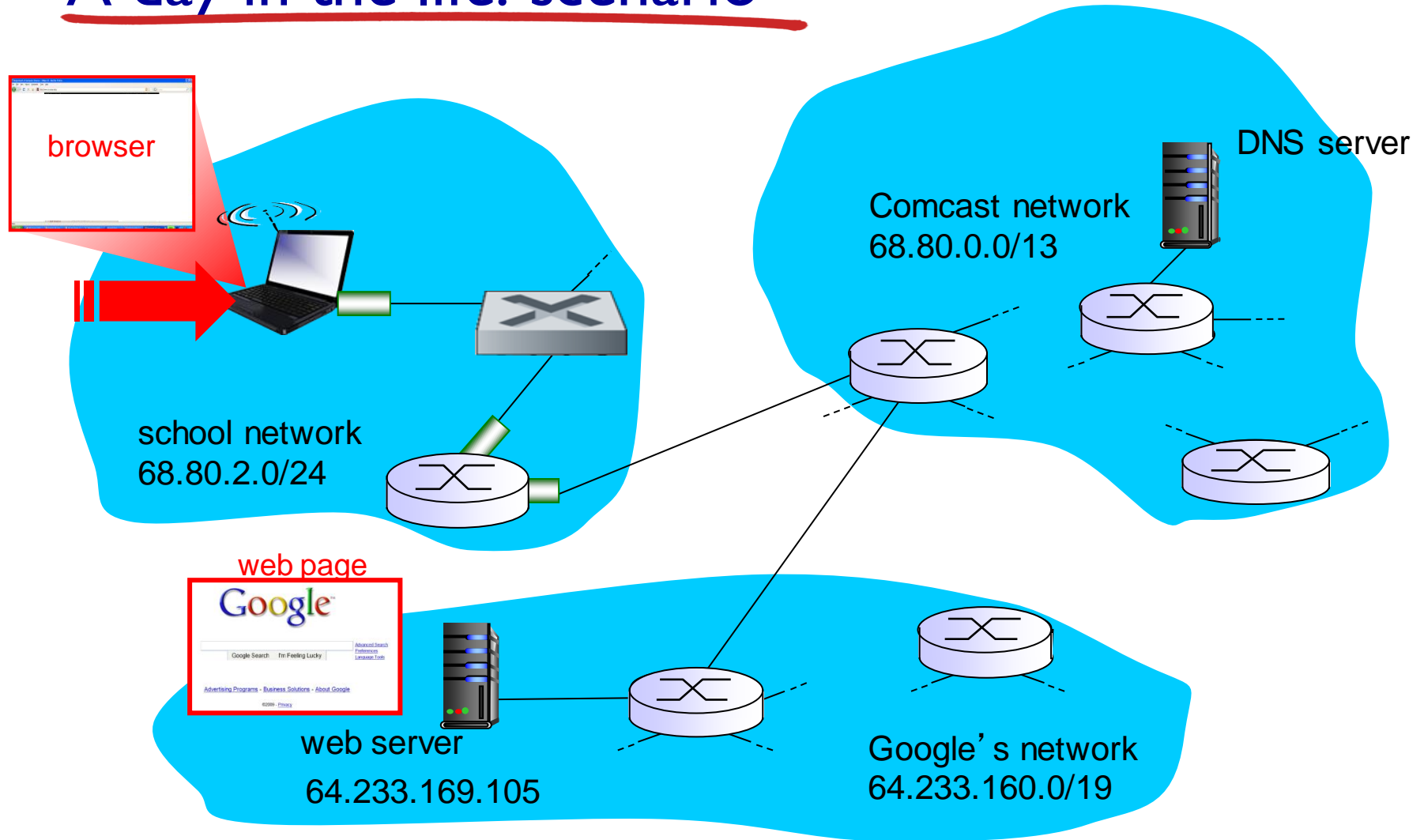
5.6 data center
networking

5.7 a day in the life of a
web request

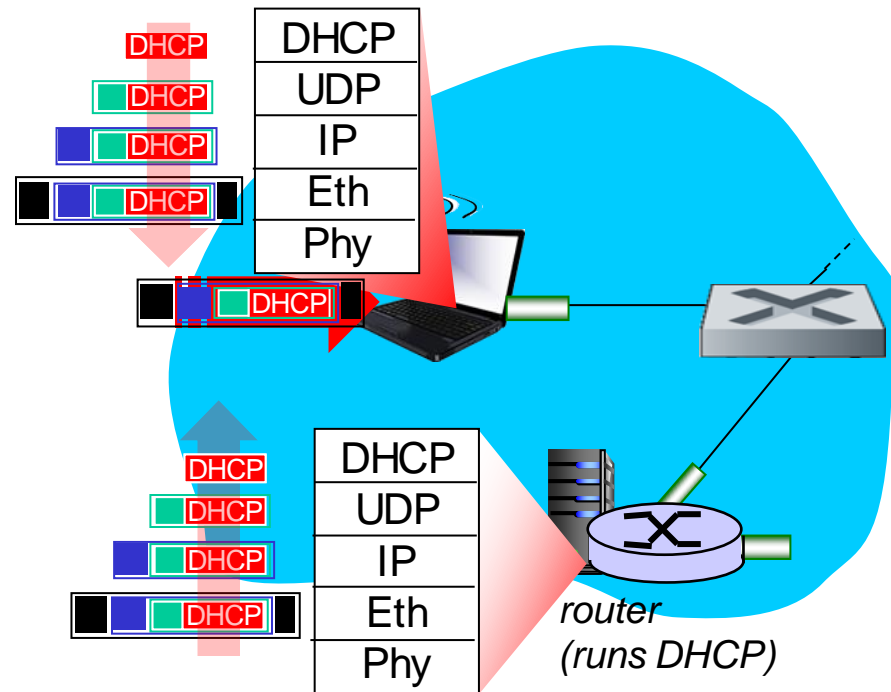
Synthesis: a day in the life of a web request

- ❖ journey down protocol stack complete!
 - application, transport, network, link
- ❖ putting-it-all-together: synthesis!
 - *goal*: identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
 - *scenario*: student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

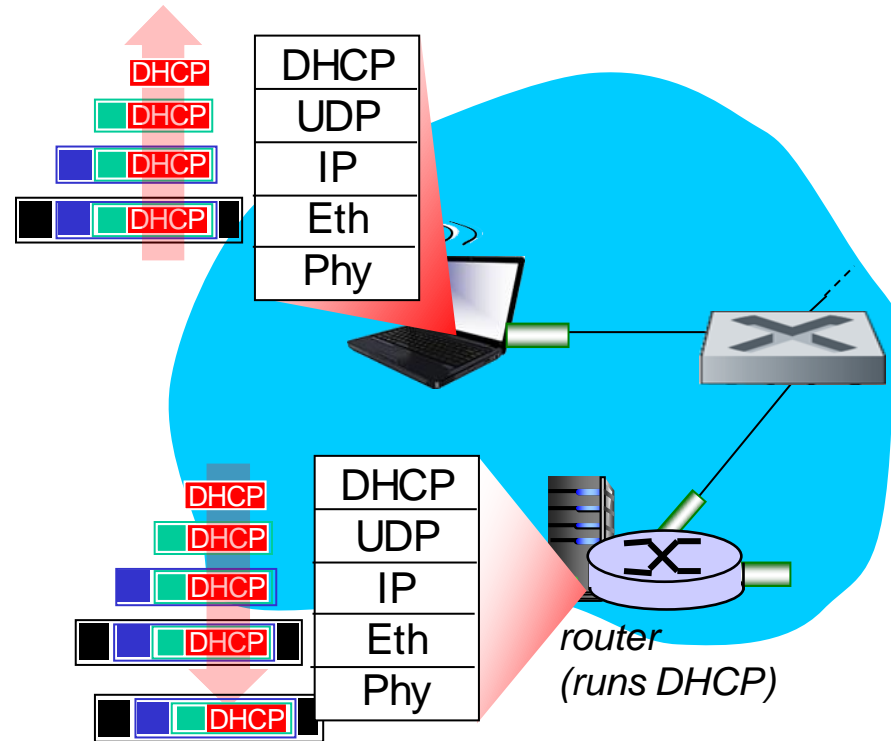


A day in the life... connecting to the Internet



- ❖ connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*
- ❖ DHCP request *encapsulated* in *UDP*, encapsulated in *IP*, encapsulated in *802.3* Ethernet
- ❖ Ethernet frame *broadcast* (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- ❖ Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

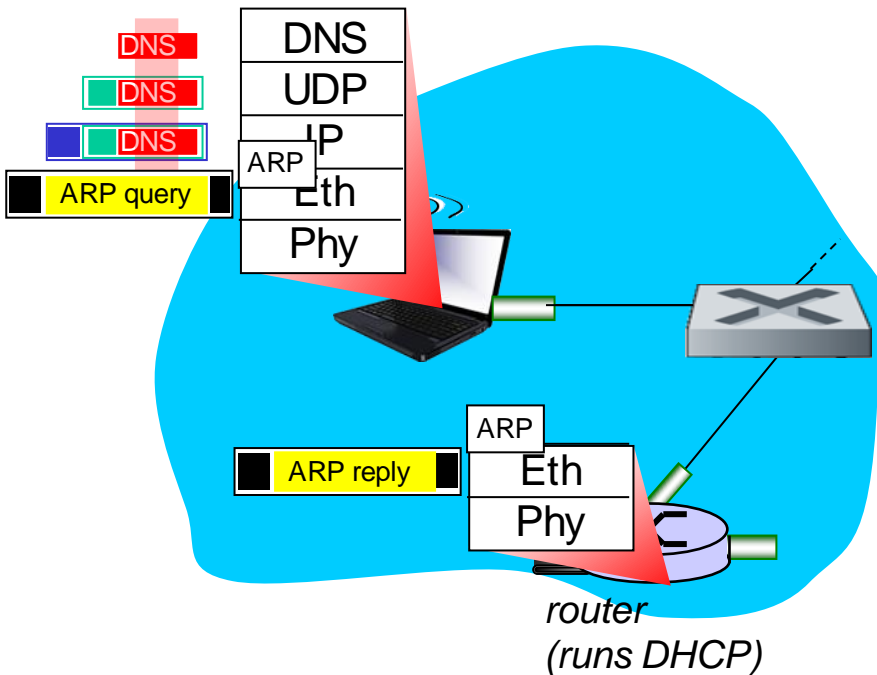
A day in the life... connecting to the Internet



- ❖ DHCP server formulates **DHCP ACK** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- ❖ encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- ❖ DHCP client receives DHCP ACK reply

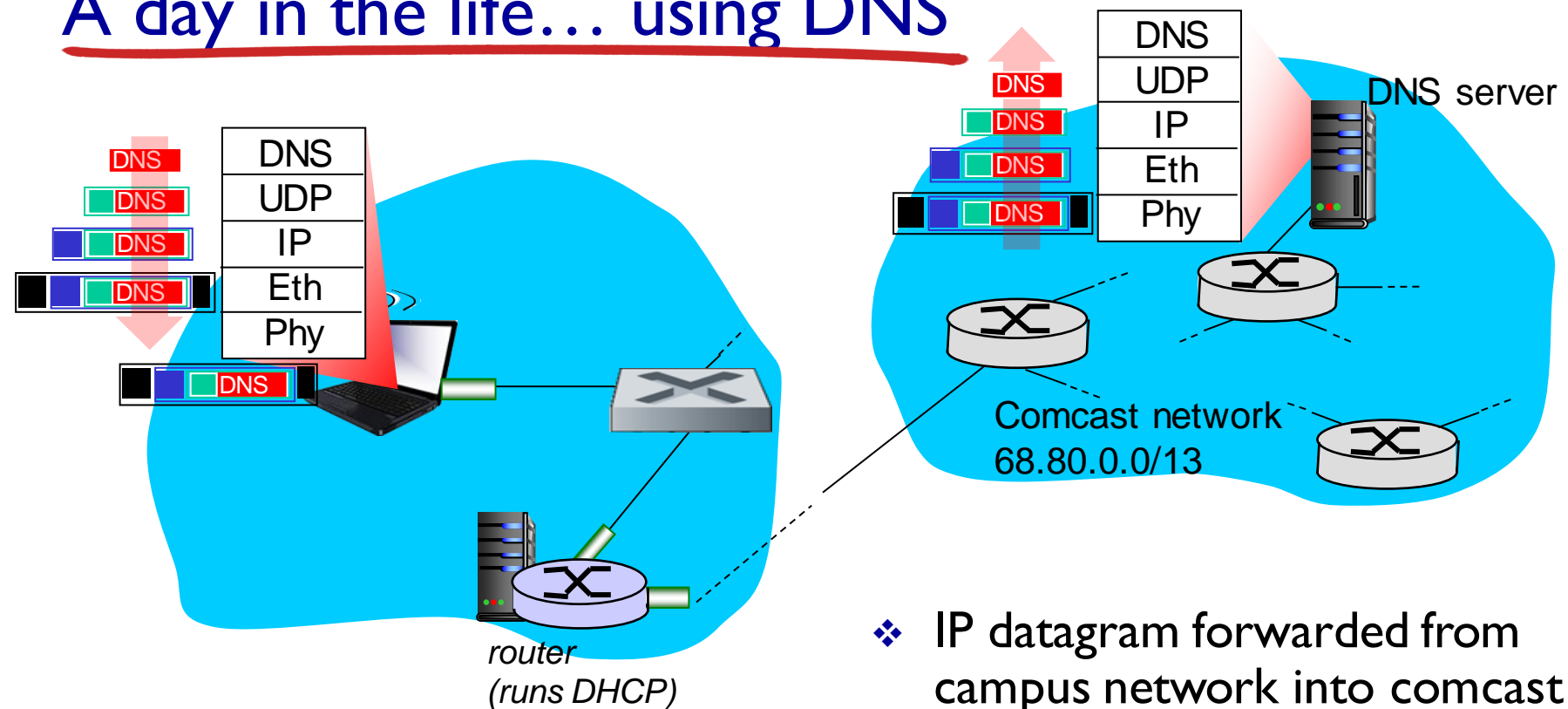
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life... ARP (before DNS, before HTTP)



- ❖ before sending *HTTP* request, need IP address of `www.google.com`:
DNS
- ❖ DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: *ARP*
- ❖ *ARP query* broadcast, received by router, which replies with *ARP reply* giving MAC address of router interface
- ❖ client now knows MAC address of first hop router, so can now send frame containing DNS query

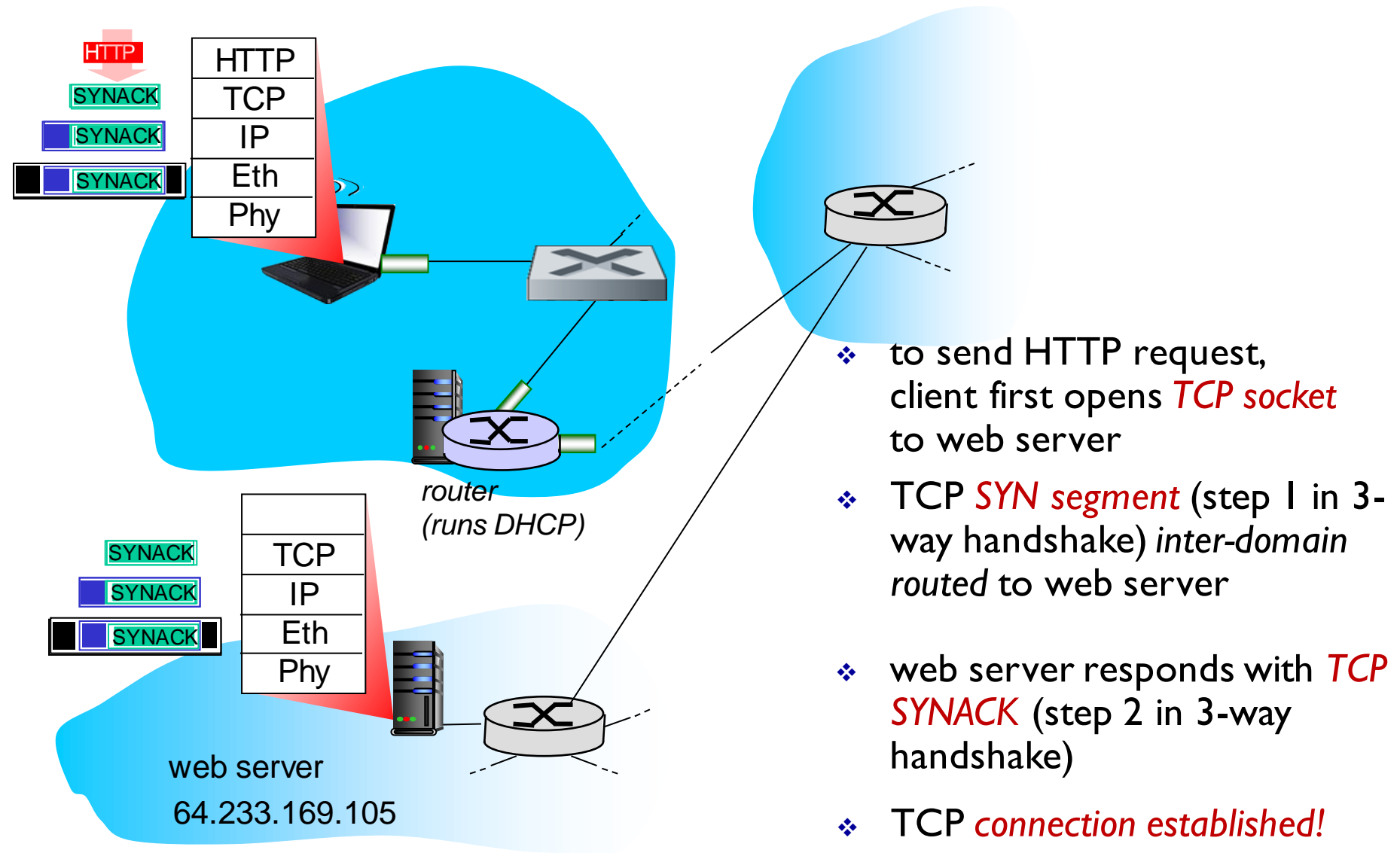
A day in the life... using DNS



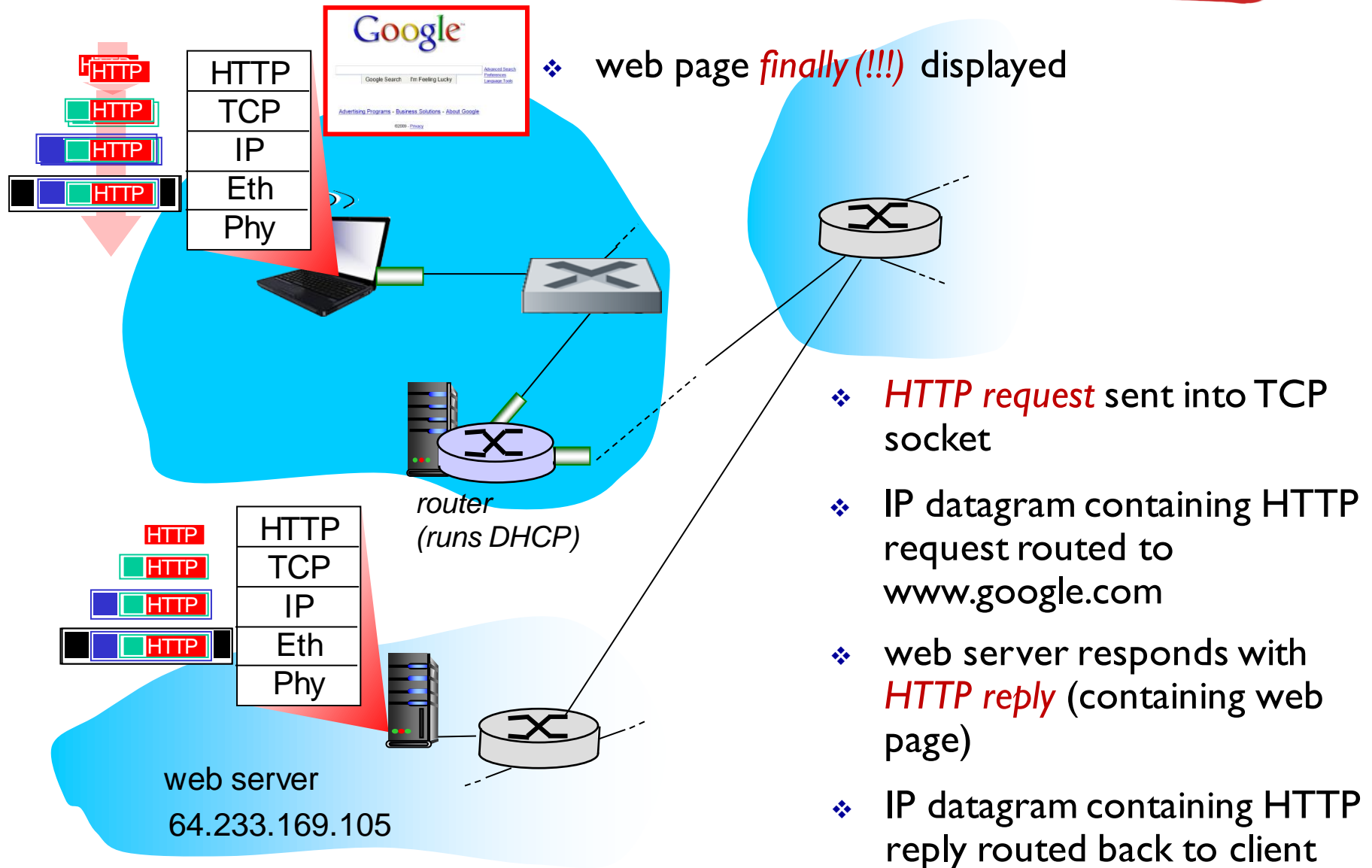
- ❖ IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

- ❖ IP datagram forwarded from campus network into comcast network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- ❖ demux'ed to DNS server
- ❖ DNS server replies to client with IP address of www.google.com

A day in the life...TCP connection carrying HTTP



A day in the life... HTTP request/reply



Chapter 5: Summary

- ❖ principles behind data link layer services:
 - error detection, correction
 - sharing a broadcast channel: multiple access
 - link layer addressing
- ❖ instantiation and implementation of various link layer technologies
 - Ethernet
 - switched LANS, VLANs
 - virtualized networks as a link layer: MPLS
- ❖ synthesis: a day in the life of a web request

Chapter 5: let's take a breath

- ❖ journey down protocol stack *complete* (except PHY)
- ❖ solid understanding of networking principles, practice
- ❖ could stop here but *lots* of interesting topics!
 - wireless
 - multimedia
 - security
 - network management