CS5222 Computer Networks and Internets

Link Layer

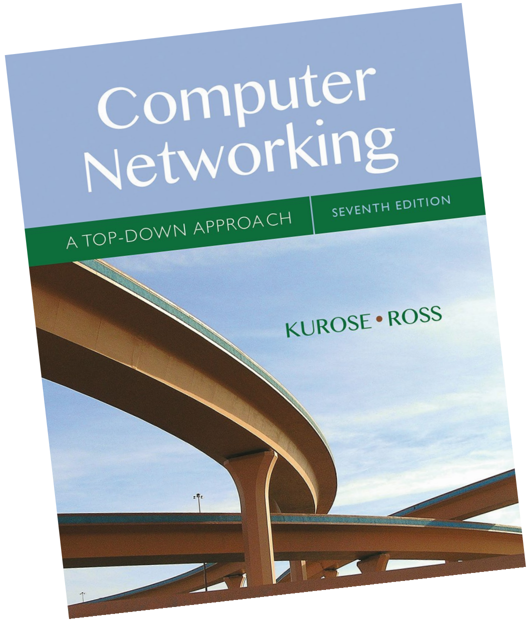
Prof Weifa Liang

Weifa.liang@cityu-dg.edu.cn

Slides based on book *Computer Networking: A Top-Down Approach.*

Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection and correction
- LANs
  - addressing, ARP, RARP
  - Ethernet
  - switches
- data center networking
- a day in the life of a web request



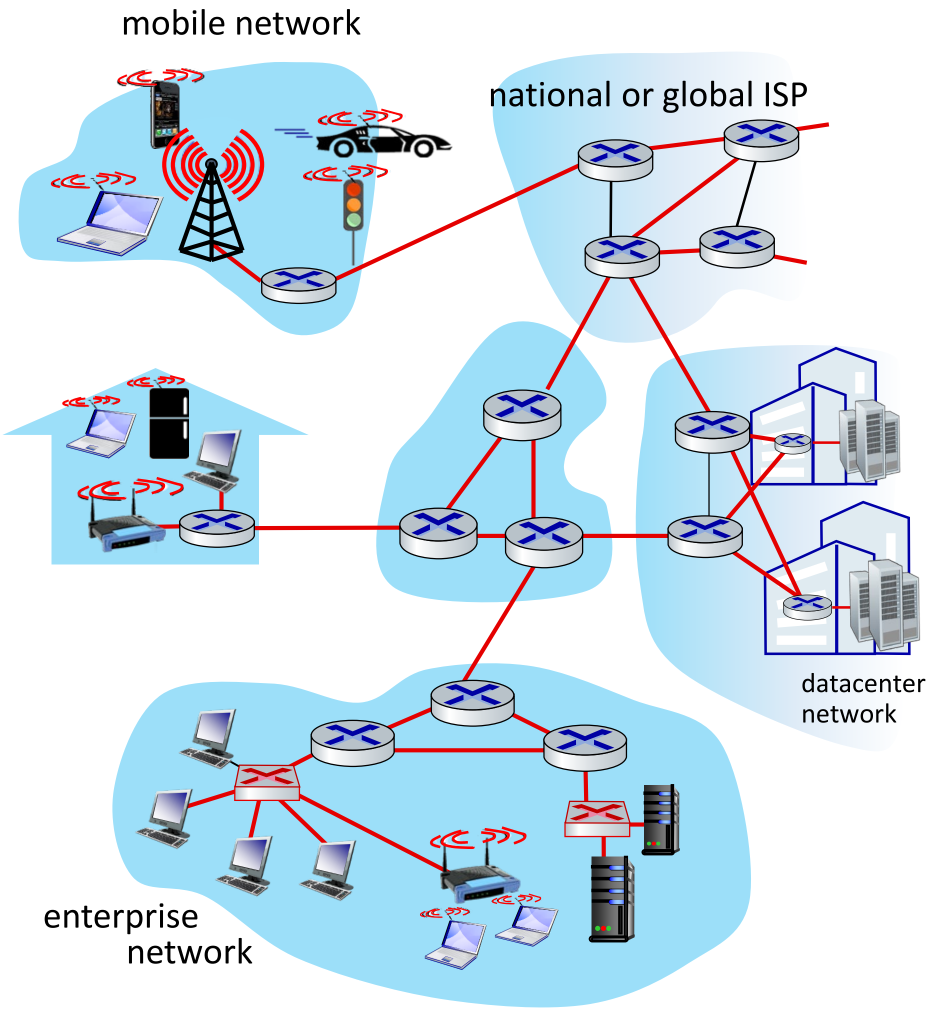
Chapter 6

Link layer: introduction

terminology:

- hosts, routers, switches, etc.: **nodes**
- communication channels that connect adjacent nodes along communication path: links
  - wired (cable, optical fiber)
  - wireless
  - LANs
- layer-2 "packet" (protocol data unit or PDU): *frame*, encapsulates datagram (packet)

link layer has the responsibility of reliably transferring datagrams from one node to a physically adjacent node over a link



Link layer: context

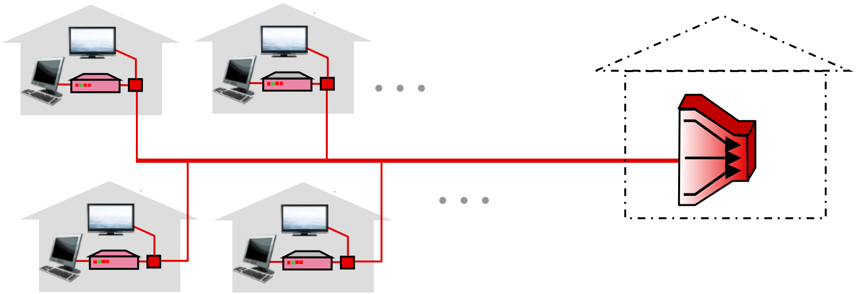
- datagram transferred by different link protocols over different links:
  - e.g., home network: Wi-Fi on first link, Ethernet on next link
- each link protocol provides different services
  - e.g., may or may not provide "reliable" data transfer over link

transportation analogy:

- you plan a trip from CityU (HK) to Disney (USA)
- A friend suggests the following way:
  - taxi: CityU to HKG
  - plane: HKG to SFO →MCO (Orlando)
  - At MCO, you ask for direction to Taxi stop.
  - Taxi: MCO to Disney
- you =
- transport segment =
- asking for direction =
- friend =
- transportation mode =

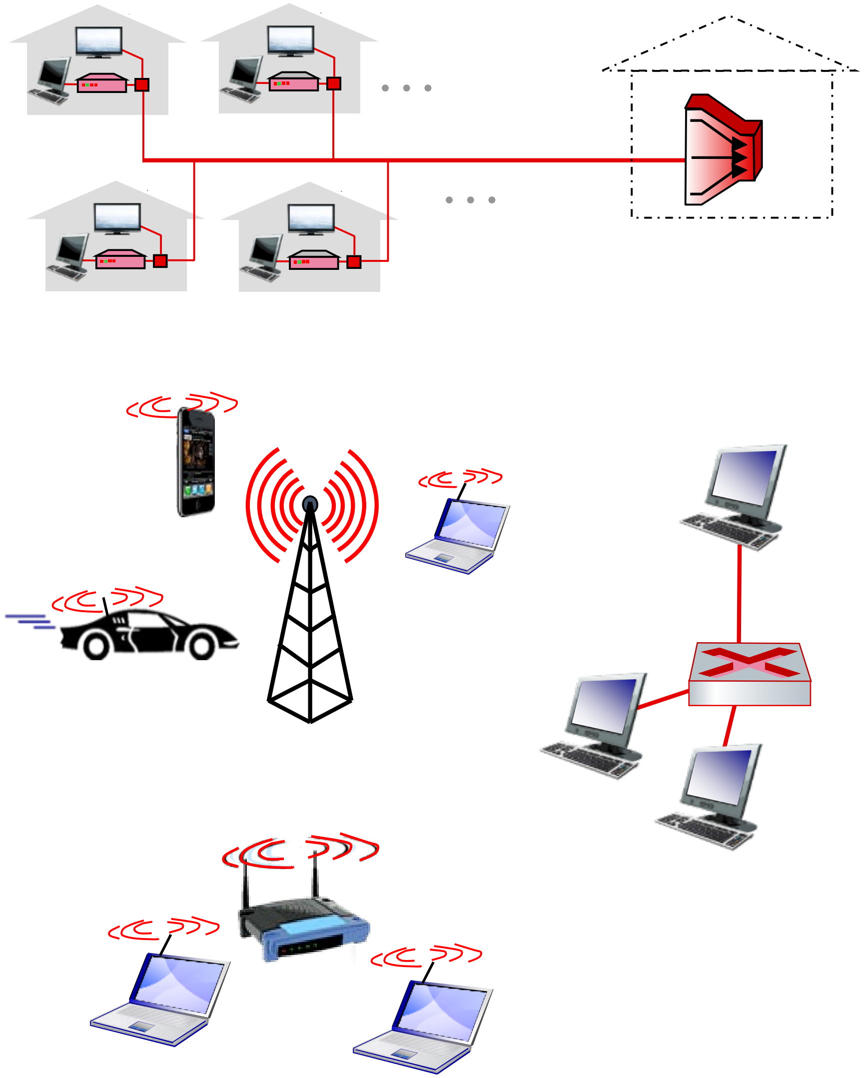
Link layer: possible services

- framing, link access:
  - encapsulate datagram into frame, adding header and trailer (control fields)
  - channel access if shared medium
  - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- reliable delivery between adjacent nodes
  - guarantees to move each datagram across link without error (“reliably”)
  - seldomly used on low bit-error links
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?



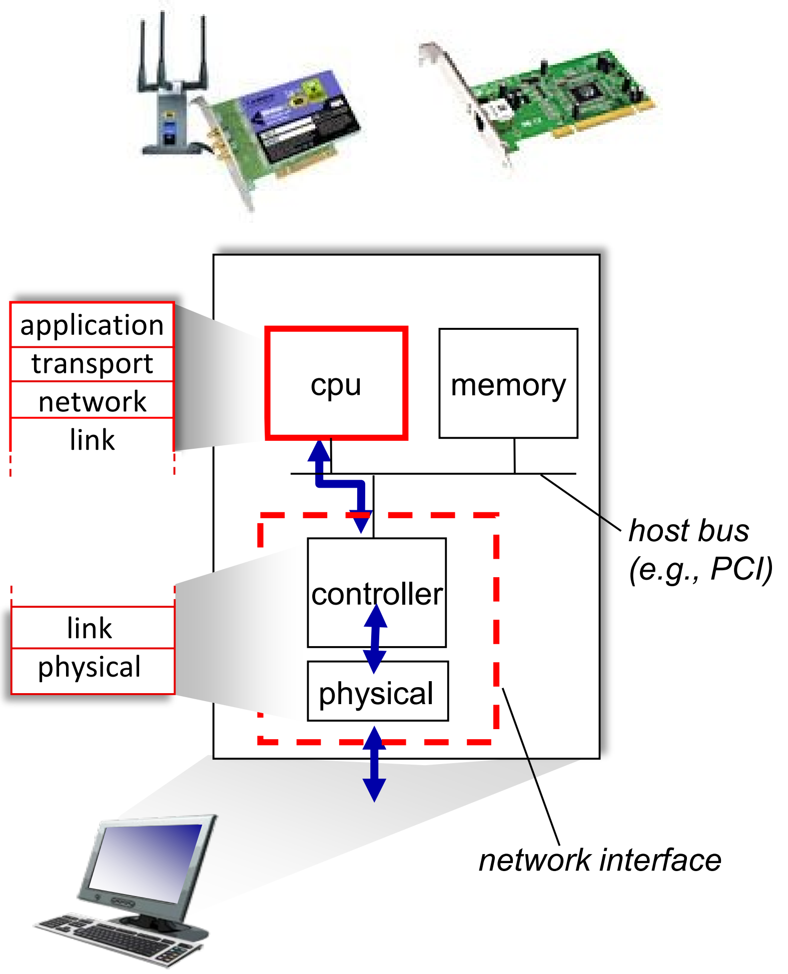
Link layer: services (more)

- error detection:
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- error correction:
  - receiver identifies *and corrects* bit error(s) without retransmission

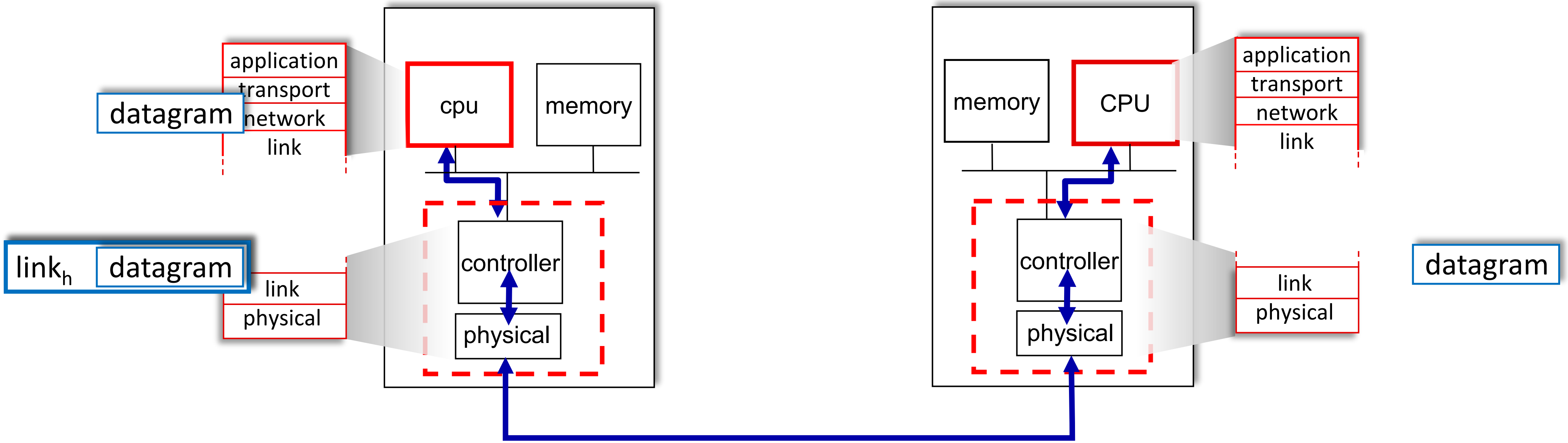


Where is the link layer implemented?

- in each-and-every node
- link layer implemented in *network interface card* (NIC) or chip
  - Ethernet, WiFi card or chip
  - implements link, physical layer
- attaches to host's system buses
- combination of hardware & software



Interfaces communicating



sending side:

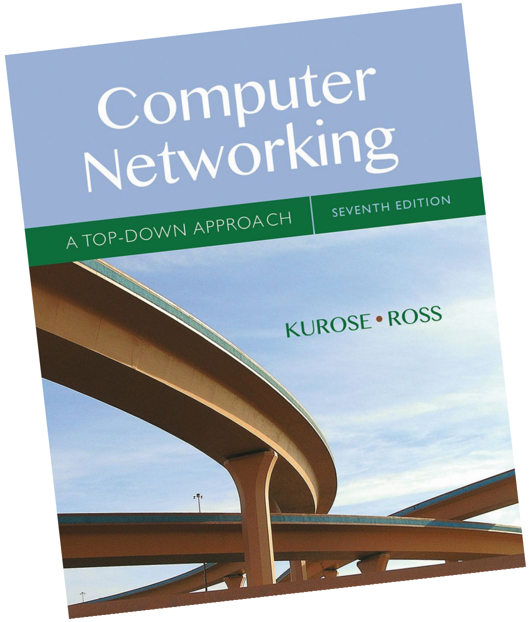
- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:

- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, if no error, passes to upper layer at receiving side

Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking
- a day in the life of a web request



Chapter 6

Multiple access links, protocols

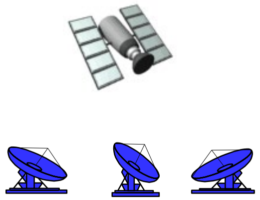
two types of "links":

- point-to-point
  - e.g., PPP for dial-up access
- broadcast (shared wire or medium)
  - Ethernet (wired bus)
  - 802.11 wireless LAN, Bluetooth, Zigbee, Lora, 4G/5G/6G, V2X, satellite

shared wire (e.g., cabled Ethernet)

shared radio: 4G/5G

shared radio: WiFi



shared radio: 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

- Coordination of the uses of a shared channel/medium
- distributed algorithm that determines how nodes share channel, i.e., determine when a node can transmit
- Communication about channel sharing must use the channel itself!
  - no out-of-band channel for coordination

An ideal multiple access protocol

Given: multiple access channel (MAC) of rate *R* bps

Properties:

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 (if fairly shared or centrally scheduled)*
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots, etc.
4. simple (i.e. easy to implement)

MAC protocols: taxonomy

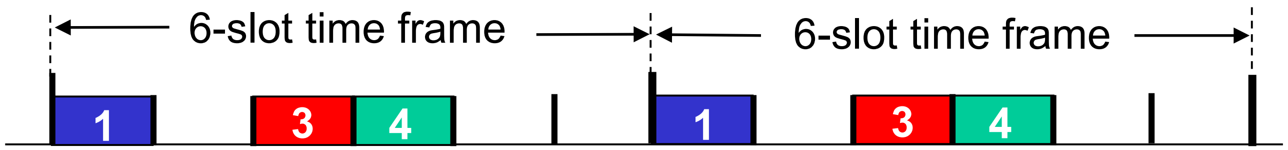
three classes:

- channel partitioning (deterministic access) or channelization
  - divide a "channel" into smaller "pieces" (time slots, frequency, code)
  - allocate a piece to a node for exclusive use
- *random access*
  - channel not divided, allow collisions
  - "recover" from collisions: collision resolution
- "take turns" (round-robin)
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

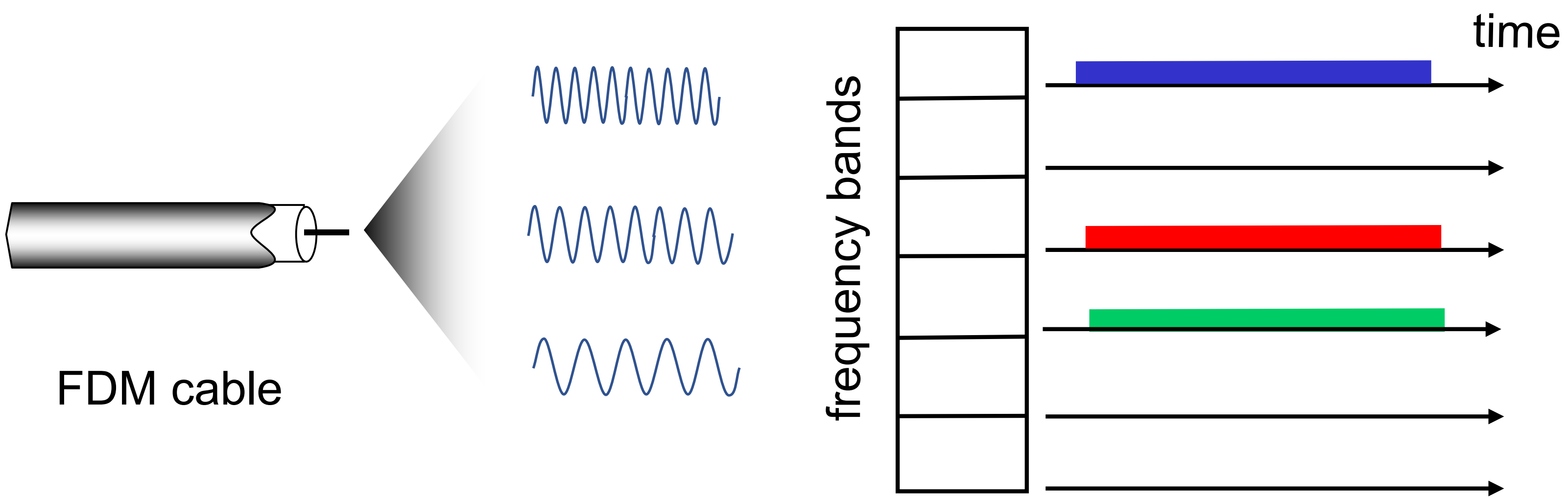
- access to channel in "rounds"
- each station gets a slot of fixed length (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum is divided into frequency bands
- each station is assigned to fixed frequency band
- unused transmission times in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



Channel partition MAC protocols: CDMA

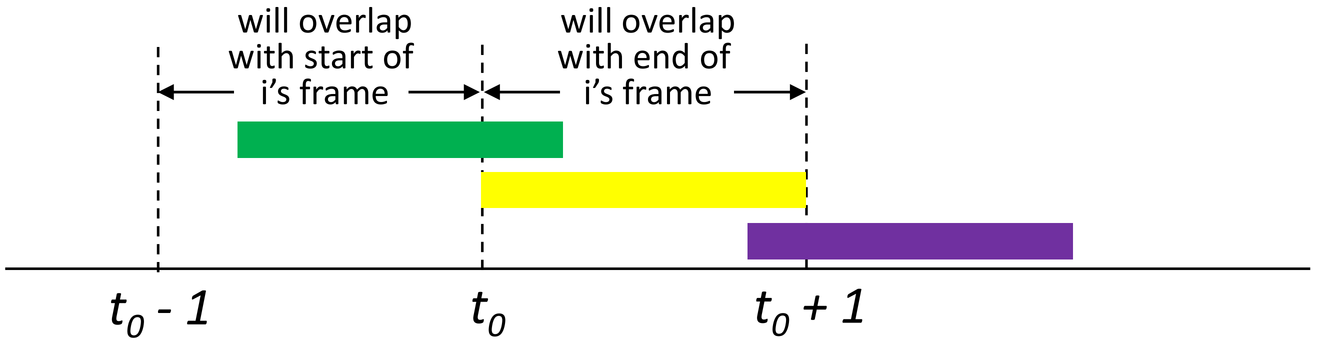
- CDMA: Code division multiple access
  - Time and frequency are not divided, but orthogonal codes are used
  - A user is assigned a code for transmission
    - Multiple users could transmit at the same time in the same band without interference
    - Similar to language usage: conversations with different languages will not interfere with each other
  - CDMA was popular in 2G/3G cellular systems
    - QualComm made its rich from CDMA patents
- OFDM: Orthogonal frequency division multiple access
  - Not for MAC, but popular with channel partitions
  - The most prevalent technologies in wireless industries

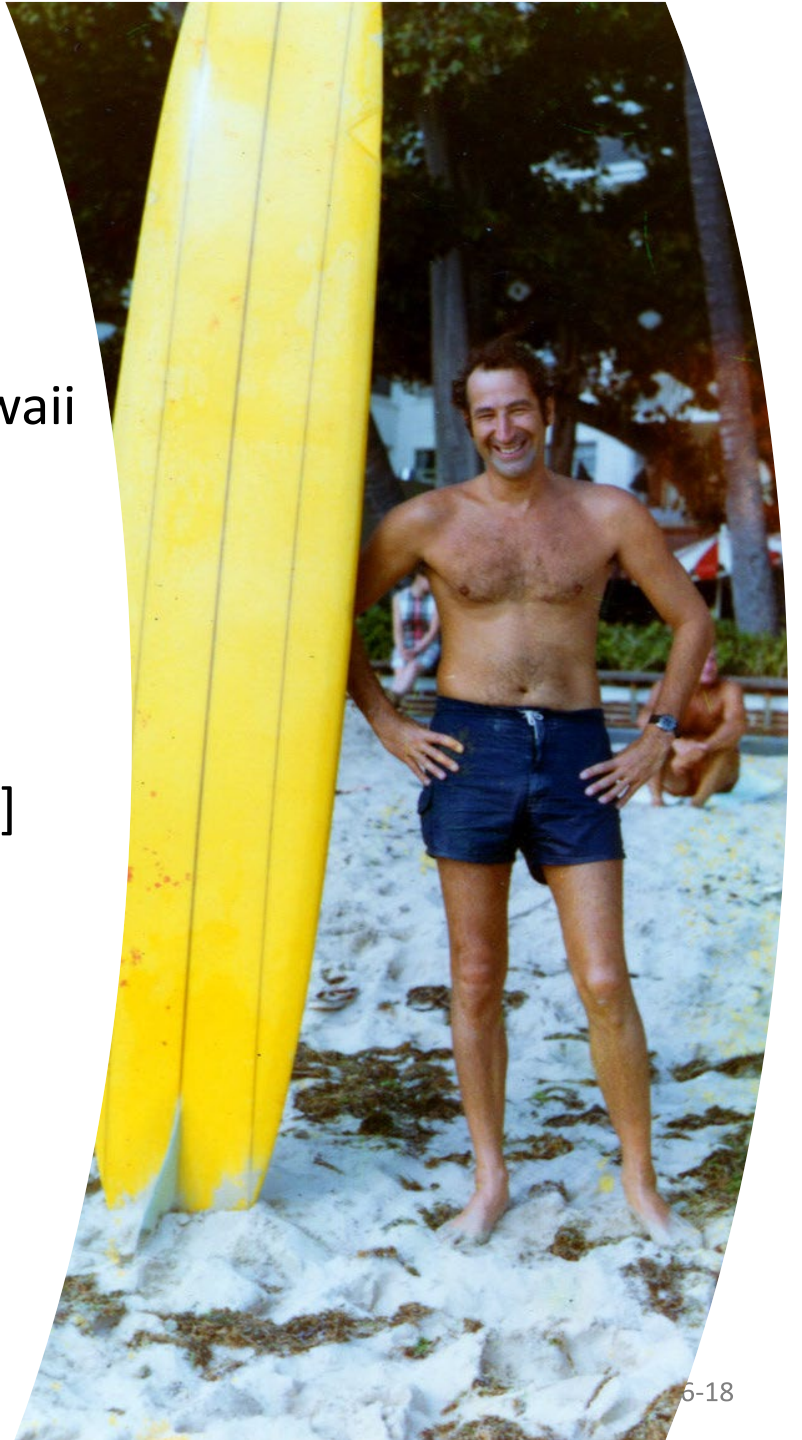
Random access protocols

- when a node has a packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes (some exceptions)
- two or more nodes transmit at the same time → "collision"
- random access MAC protocol specifies:
  - how to detect collisions: collision detection
  - how to recover from collisions (e.g., via delayed retransmissions): collision resolution
- examples of random access MAC protocols:
  - ALOHA, slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA
- information used in progression:
  - Aloha → slotted Aloha → CSMA →CSMA/CD →CSMA/CA

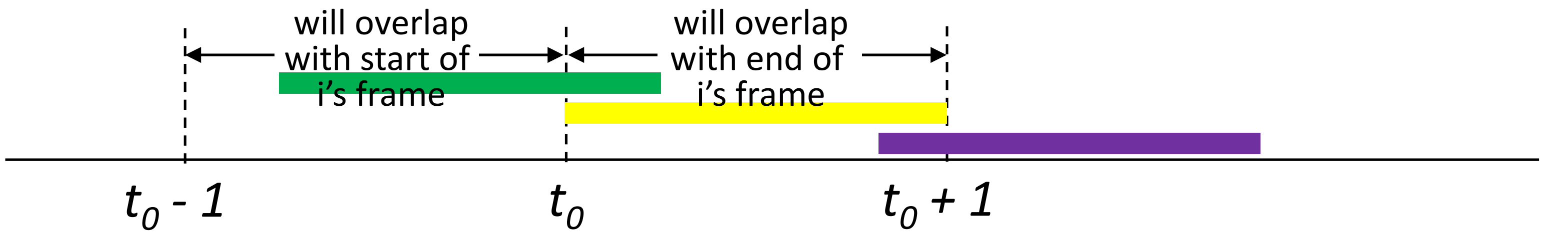
Pure ALOHA: no coordination!

- Aloha: Hawaiian language, meaning welcome and goodbye
- Motivated by linking multiple libraries of the University of Hawaii on different islands
- Developed by Norm Abramson in 1970s
- unslotted Aloha: simpler, no synchronization
  - Whenever a frame arrives, it is transmitted immediately
- collision probability increases with no synchronization:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$





Pure ALOHA: efficiency



- Performance: How to evaluate a coordination rule? A good engineer always seeks good design!
- Efficiency (throughput): long-run fraction of successful transmissions among all transmissions (many nodes, all with many frames to send)
- *Assumptions: N* nodes with many frames to send, each transmits with probability *p*
- Then, Pr(success by a given node) = Pr(one node transmits) ·

  Pr(no other node transmits in $[t_0\text{-}1, t_0]$ ·

  Pr(no other node transmits in $[t_0, t_{0+1}]$)

  $= p \cdot (1\text{-}p)^{N\text{-}1} \cdot (1\text{-}p)^{N\text{-}1}$

  $= p \cdot (1\text{-}p)^{2(N\text{-}1)}$
- Assume that each user is independently transmitting (what pure Aloha does!). Then,

  S = E(p)=*N* P(success by a given node) = $Np(1\text{-}p)^{2(N\text{-}1)}$

  → *Maximal efficiency S = 1/(2e)=.184 when N* → ∞

Pure ALOHA: efficiency or throughput

Let us derive the maximum efficiency under the assumption that there are a large number of nodes and each node has a large number of packets to send.

The efficiency of pure ALOHA is

$$\mathrm{E}(p) = Np(1-p)^{2(N-1)}$$

By derivation, we have

$$\begin{aligned}\mathrm{E}'(\mathrm{p}) &= \mathrm{N}(1-\mathrm{p})^{2(\mathrm{N}-1)} - Np2(N-1)(1-p)^{2\ N-3} \\ &= \mathrm{N}(1-\mathrm{p})^{2\ \mathrm{N}-3}\left((1-p) - p2(N-1)\right)\end{aligned}$$

Let $\mathrm{E}'(\mathrm{p}) = 0$, we have $\mathrm{p}^* = \frac{1}{2N-1}$, the corresponding efficiency is

$$\mathrm{E}(\mathrm{p}^*) = \frac{N}{2N-1}\left(1 - \frac{1}{2N-1}\right)^{2(N-1)}$$

Let n approach infinity,

$$\lim_{\mathrm{N}\to\infty} E(p^*) = \frac{1}{2}\cdot\frac{1}{e} = \frac{1}{2e}$$

Slotted ALOHA: a little coordination

Assumptions (for presentation or analysis):

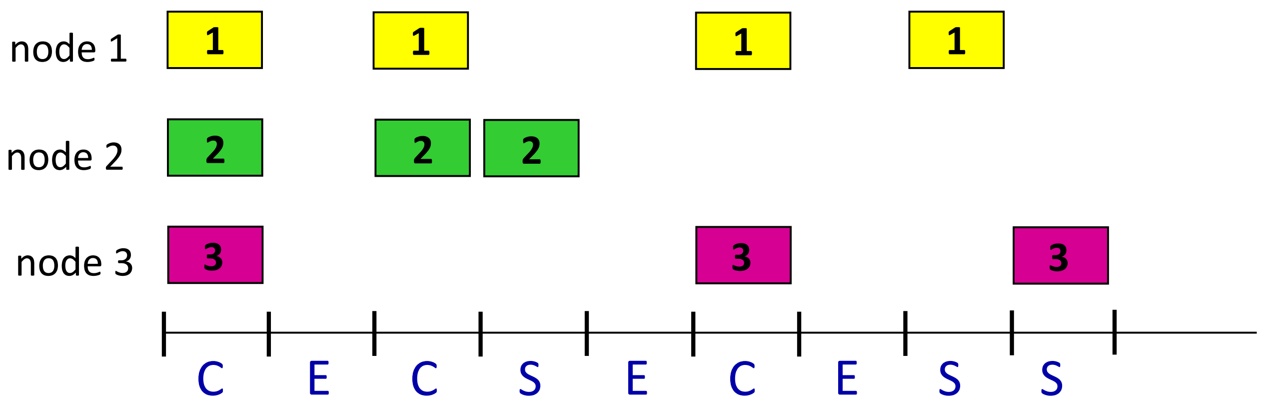
- all frames of the same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

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 probability *p* until success

must use randomization!

Slotted ALOHA



C: collision
S: success
E: empty

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

Slotted ALOHA: efficiency or throughput

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *Assumptions: N* nodes with many frames to send, each transmits in slot with probability *p*
  - prob that a given node has success in a slot $= p(1-p)^{N-1}$
  - prob that *any* node has a success $= Np(1-p)^{N-1}$
  - **maximum efficiency**: find *p* that maximizes $Np(1-p)^{N-1}$
  - $\rightarrow p = 1/N$
  - for many nodes, take limit of $Np(1-p)^{N-1}$ as *N* goes to infinity, gives:

  max efficiency = 1/e = .368

- *at best:* channel used for useful transmissions 36.8% of time!

Slotted ALOHA: efficiency or throughput

Efficiency $E(p)=Np(1-p)^{N-1}$

Let us derive the maximum efficiency under the assumption that there are a large number of nodes and each node has a large number of packets to send.

By derivation, we have

$$\mathrm{E}'(\mathrm{p}) = \mathrm{N}(1-\mathrm{p})^{\mathrm{N}-1} - Np(N-1)(1-p)^{N-2} = N(1-p)^{N-2}((1-p) - p(N-1))$$

Let

$$\mathrm{E}'(\mathrm{p}) = 0$$

Thus, when

$$\mathrm{p}^* = \frac{1}{N}$$

We have the maximum efficiency.

Slotted Aloha: efficiency or throughput

When p equals to $\frac{1}{\mathrm{N}}$, the efficiency of slotted ALOHA is

$$\mathrm{E(p^*)} = N \cdot \frac{1}{N}\left(1 - \frac{1}{N}\right)^{N-1} = \left(1 - \frac{1}{N}\right)^{N-1} = \frac{\left(1 - \frac{1}{N}\right)^{N}}{1 - \frac{1}{N}}$$

$$\lim_{\mathrm{N}\to\infty}\left(1 - \frac{1}{N}\right) = 1$$

, then

$$\lim_{\mathrm{N}\to\infty}\left(1 - \frac{1}{N}\right)^{N} = \frac{1}{e}$$

. Thus when n approaching infinity, the efficiency is

$$\lim_{\mathrm{N}\to\infty} E(p^*) = \frac{1}{e}$$

at best: channel used for useful transmissions 36.8% of time!

CSMA (carrier sense multiple access): listen before talk

- Coordination rule: sense before transmit, or listen before you speak
- How to sense: detect whether there is any transmission activity in the channel
  - Energy or voltage or current…
- Many operating modes
  - non-persistent
  - 1-persistent
  - p-persistent

Non-persistent CSMA

- A station senses the channel
- If the channel is busy, it will not continue sensing the channel, instead, it delays a random time period (backoff), and come back and listen (sense) again
- If the channel is idle, it transmits
- If a collision occurs, it will delay a random period of time (backoff), starts all over again

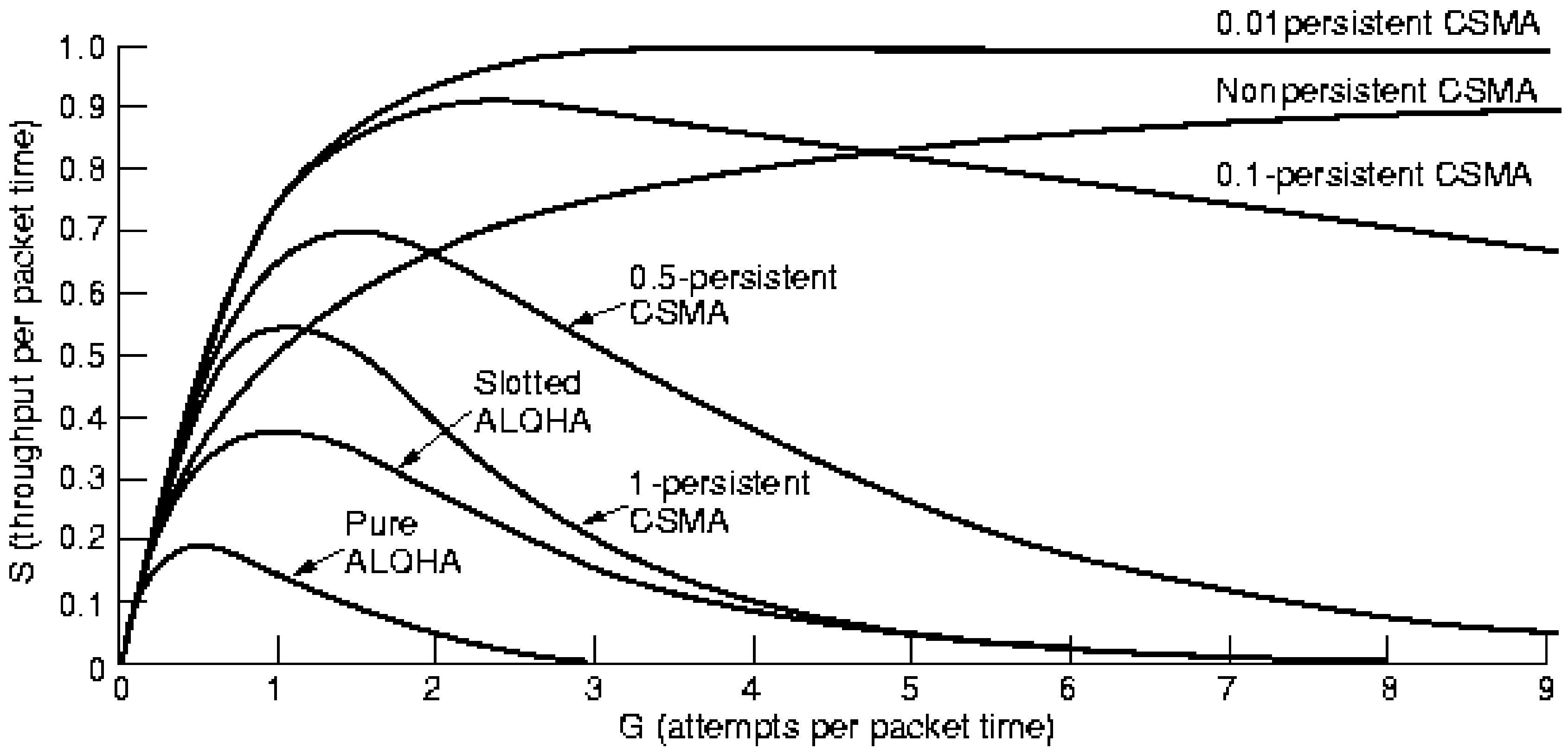
1-persistent CSMA

- A station (a user, a transmitter) senses the channel
- If the channel is busy, it will wait until the channel is idle (sensing all the time)
- Whenever it senses the channel is idle, it will transmit
- If a collision occurs, it will delay a random period of time (backoff), and start it all over again

p-persistent CSMA

- Observation: if you sense idle, others too, and collision surely occurs
- p-persist CSMA applies to slotted channels
- A station senses the channel
- Slotted version: If it senses idle channel, it transmits with probability p, or defers to next slot with probability q=1-p and then repeats the same procedure
- If it senses busy channel, it delays a random number of slots (backoff), starts all over again

Comparison



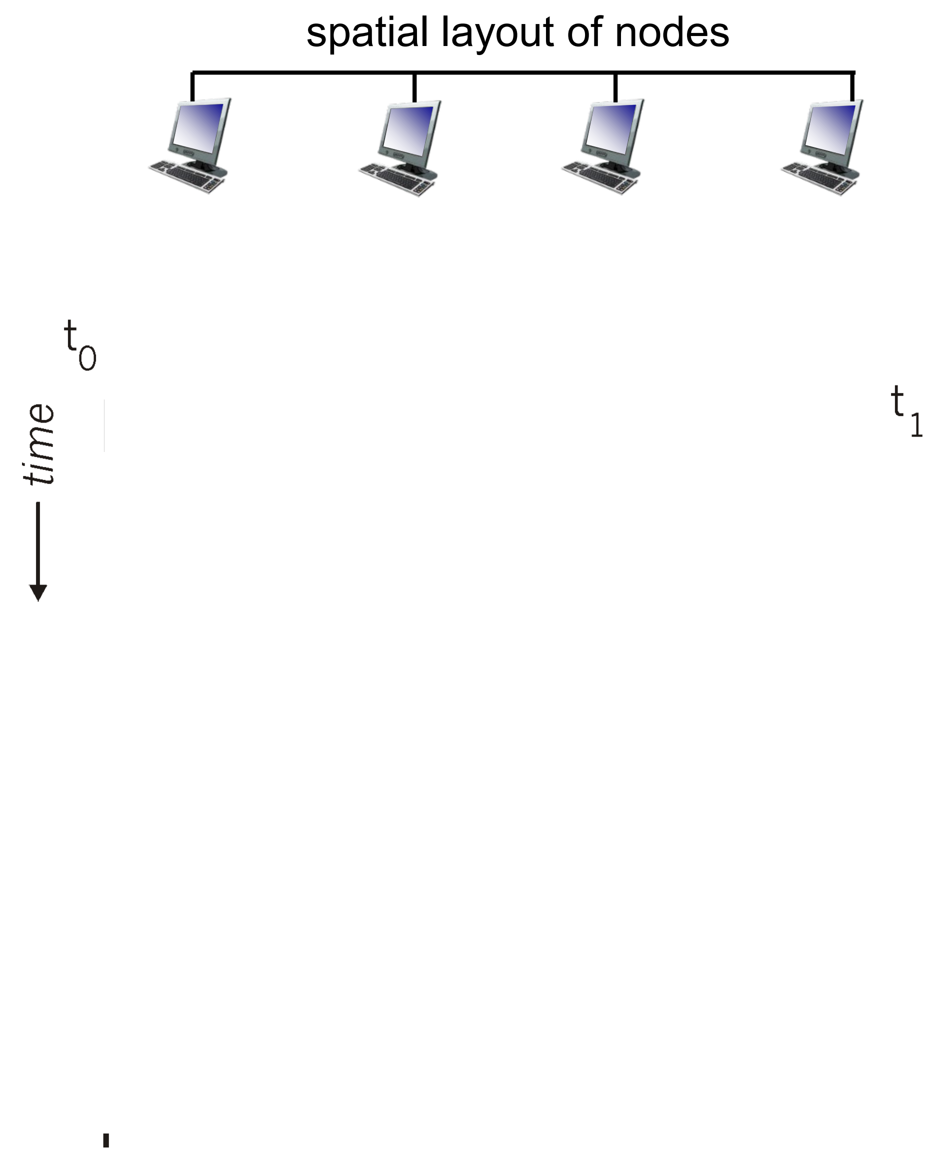
0.01persistent CSMA
Nonpersistent CSMA
0.1-persistent CSMA
0.5-persistent CSMA
Slotted ALOHA
1-persistent CSMA
Pure ALOHA
S (throughput per packet time)
G (attempts per packet time)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8
9

CSMA/CD: early stoppage

- If you know your transmission is messed up, stop!
- CSMA/CD: CSMA with *collision detection*
  - collisions *detected* within a shorter time
  - colliding transmissions aborted, reducing channel wastage
  - collision detection easy in wired, difficult in wireless
  - human analogy: the polite conversationalist
- CSMA/CD, widely used in LAN (IEEE 802.3)
  - Robert Metcalfe’s PhD thesis at MIT

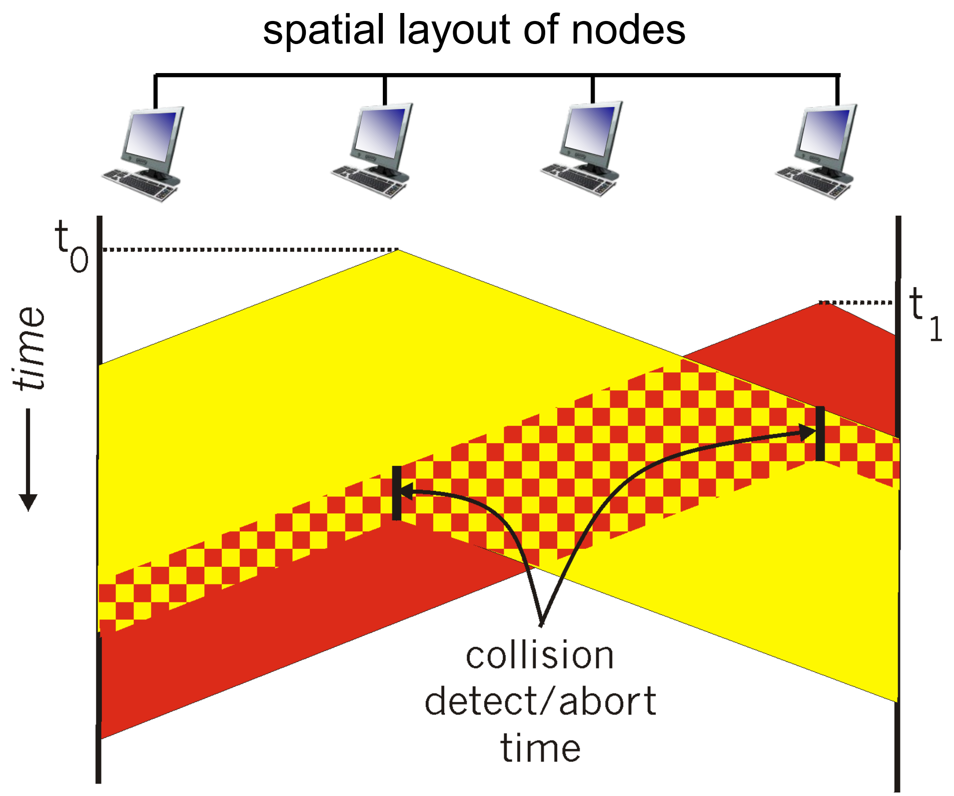
CSMA: collisions

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



CSMA/CD:

- CSMA/CD reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



Ethernet CSMA/CD algorithm

1. NIC receives datagram (packet) from network layer, creates frame
2. If NIC senses channel:
   if idle: start frame transmission.
   if busy: wait until channel idle, then transmit (p-persistent CSMA may be adopted)
3. If NIC transmits entire frame without collision, NIC is done with frame.
4. If NIC detects another transmission while transmitting: abort
5. After aborting, NIC enters *binary (exponential) backoff:*
   - after the *m*-th collision, NIC chooses *K* at random from $\{0,1,2, ..., 2^m-1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
   - more collisions: longer backoff interval

Backoff interval is **reset for new frame**!

CSMA/CD: efficiency or throughput

- t_{prop} = max prop delay between 2 nodes in LAN (two furthest nodes)
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- better performance than ALOHA: still simple, decentralized!

CSMA/CA: wireless

- CSMA/CD does not work well in a wireless channel
  - difficult to detect collisions in a radio environment
  - difficult to control the wireless channel
  - hidden and exposed terminal problems
- CA: Collision Avoidance
- CSMA/CA is used in wireless environments
  - sense before transmit (CSMA)
  - gain the access right via time-spacing and/or contention (slotted Aloha)
  - wait for channel grant of usage (let the receiver tell the status)
  - busy-tone may be used
  - IEEE 802.11x standard
    - There are many variants in wireless protocols

“Taking turns” MAC protocols: time-sharing

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

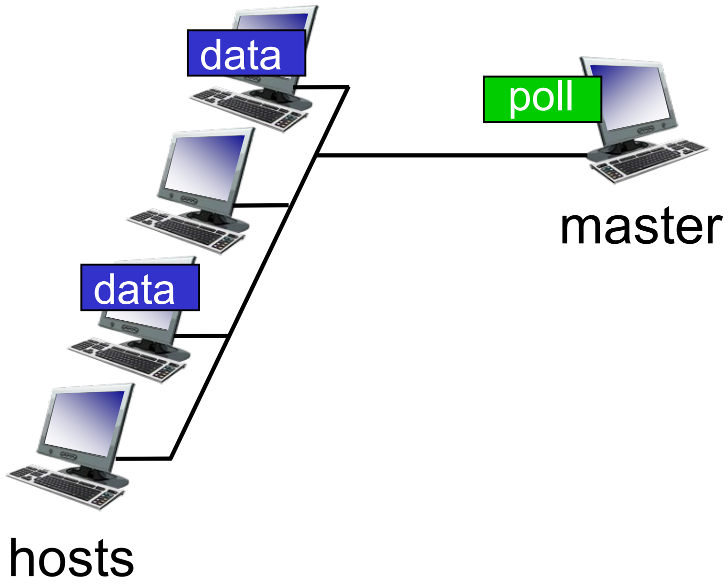
“taking turns” protocols

- look for the best of both worlds!

“Taking turns” MAC protocols

polling:

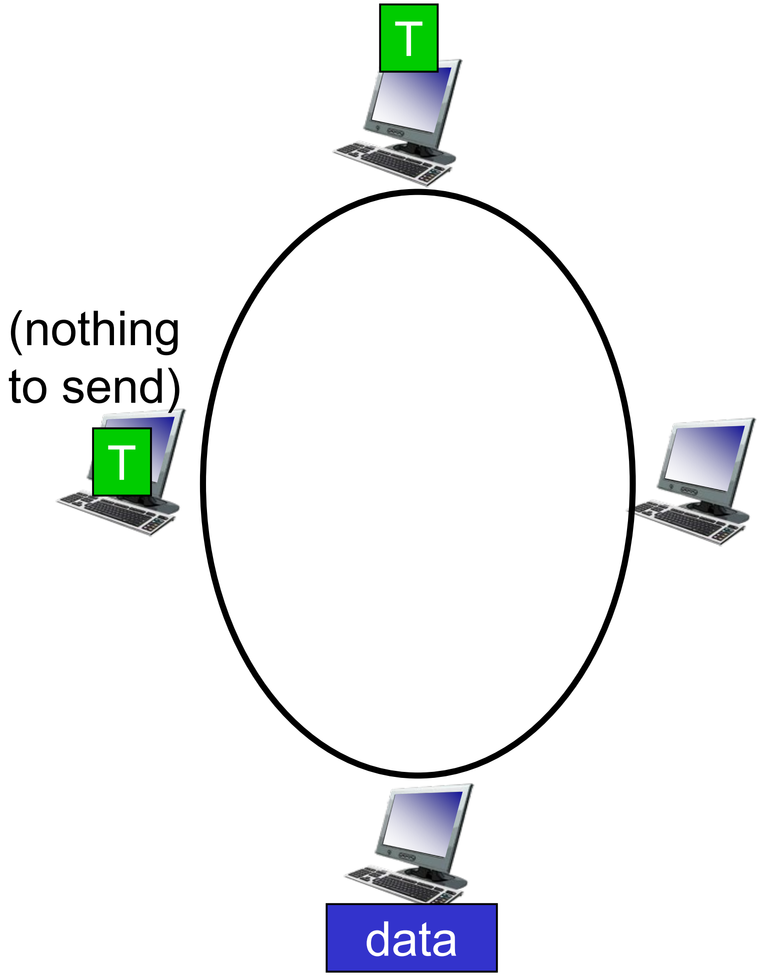
- access is structured into rounds.
- each node gets its turn during a round.
- master node “invites” nodes to transmit (one after the other)
- typically used with “simple” devices
- concerns:
  - polling delay (time d_{poll})
  - Single-point failure (master)



"Taking turns" MAC protocols

token passing:

- control *token* passed from one node to next sequentially.
  - token message (short packet)
  - token is the access right
- concerns:
  - token overhead
  - latency
  - Single-point failure (token)

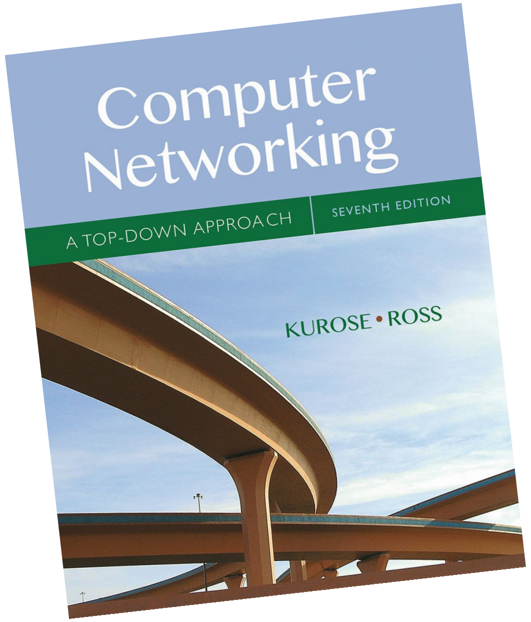


Summary of MAC protocols

- channel partitioning, by time, frequency or code
  - Time Division, Frequency Division, code 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 (TDMA)

Link layer, LANs: roadmap

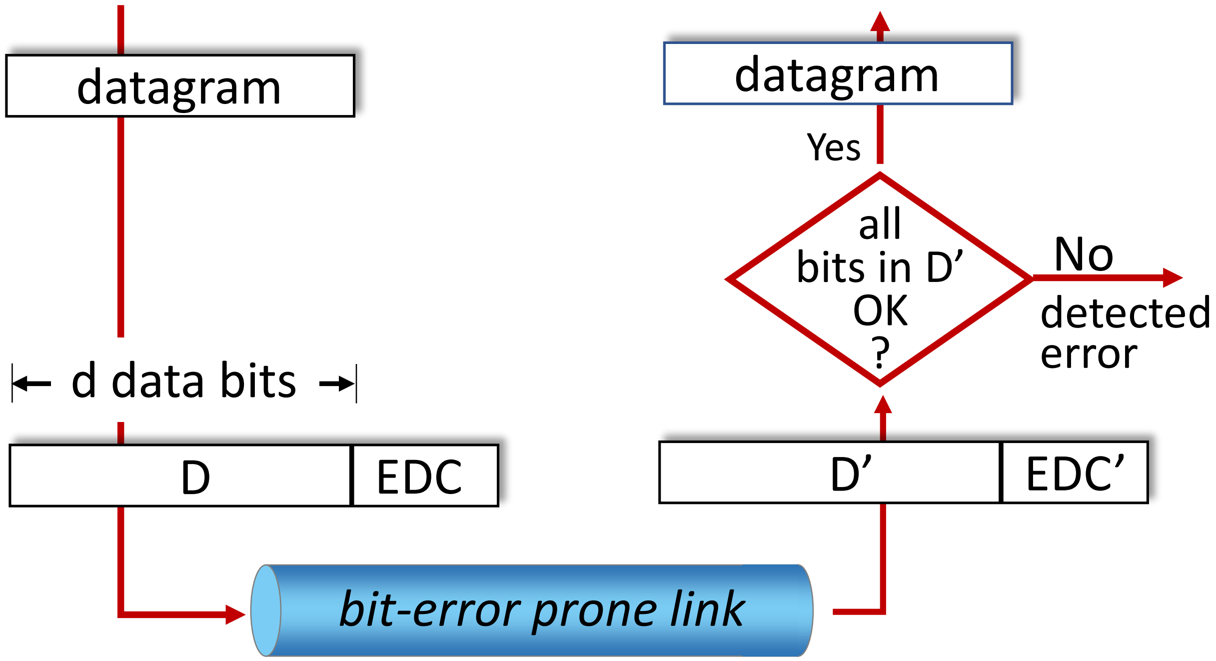
- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking
- a day in the life of a web request



Chapter 6

Error detection

EDC: error detection and correction bits (e.g., redundancy)
D: data protected by error checking, may include header fields



Error detection not 100% reliable!

- protocol may miss some errors (rarely)
- larger EDC field yields better detection and correction, but more overhead, a tradeoff!

Fundamental idea for EDC

- Link/channel condition is not ideal, transmission error does occur, and needs to tell right or wrong!
- Basic idea: frame is so constructed that a certain ***relationship*** among all bits in the frame is embedded. If the received frame does not have that relationship, then something must be wrong with the received frame!
- Redundancy bits or the check bits ("relationship") are added by the transmitter for error detection, and the receiver checks it!

Parity checking

single bit parity:

- detect single bit errors

0111000110101011	1
⟵ d data bits ⟶	parity bit

Even parity: set parity bit so there is an even number of 1’s

- count the number of 1s: set to 1 when it is odd and 0 otherwise
- Modulo-2 sum of all bits

two-dimensional bit parity:

- detect *and correct* single bit errors

row parity ⟶

column parity ↓

$$
\begin{array}{cccc|c}
d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\
d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\
\cdots & \cdots & \cdots & & \cdots \\
d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\
\hline
d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1}
\end{array}
$$

no errors:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
0 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

detected and **correctable(!)** single-bit error:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
0 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

parity error

parity error

Internet checksum (review)

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment

sender:

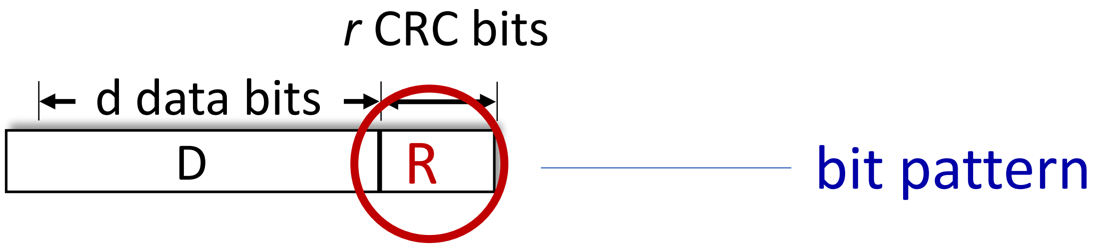
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP frame checksum field (FCS)

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - not equal - error detected
  - equal - no error detected. *But maybe errors nonetheless?* More later ….

Cyclic Redundancy Check (CRC)

- more powerful error-detection coding
- D: data bits (given, think of these as a binary number)
- G: bit pattern (generator), of *r+1* bits (given)



$<D,R> = D * 2^r$ XOR R — formula for bit pattern

goal: choose *r* CRC bits, R, such that <D,R> exactly divisible by G (mod 2)

- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors of up to r bits
- widely used in practice (Ethernet, 802.11 WiFi)

Cyclic Redundancy Check (CRC): example

We want:

$D \cdot 2^r \;\; XOR \;\; R = nG$

or equivalently:

$D \cdot 2^r = nG \;\; XOR \;\; R$

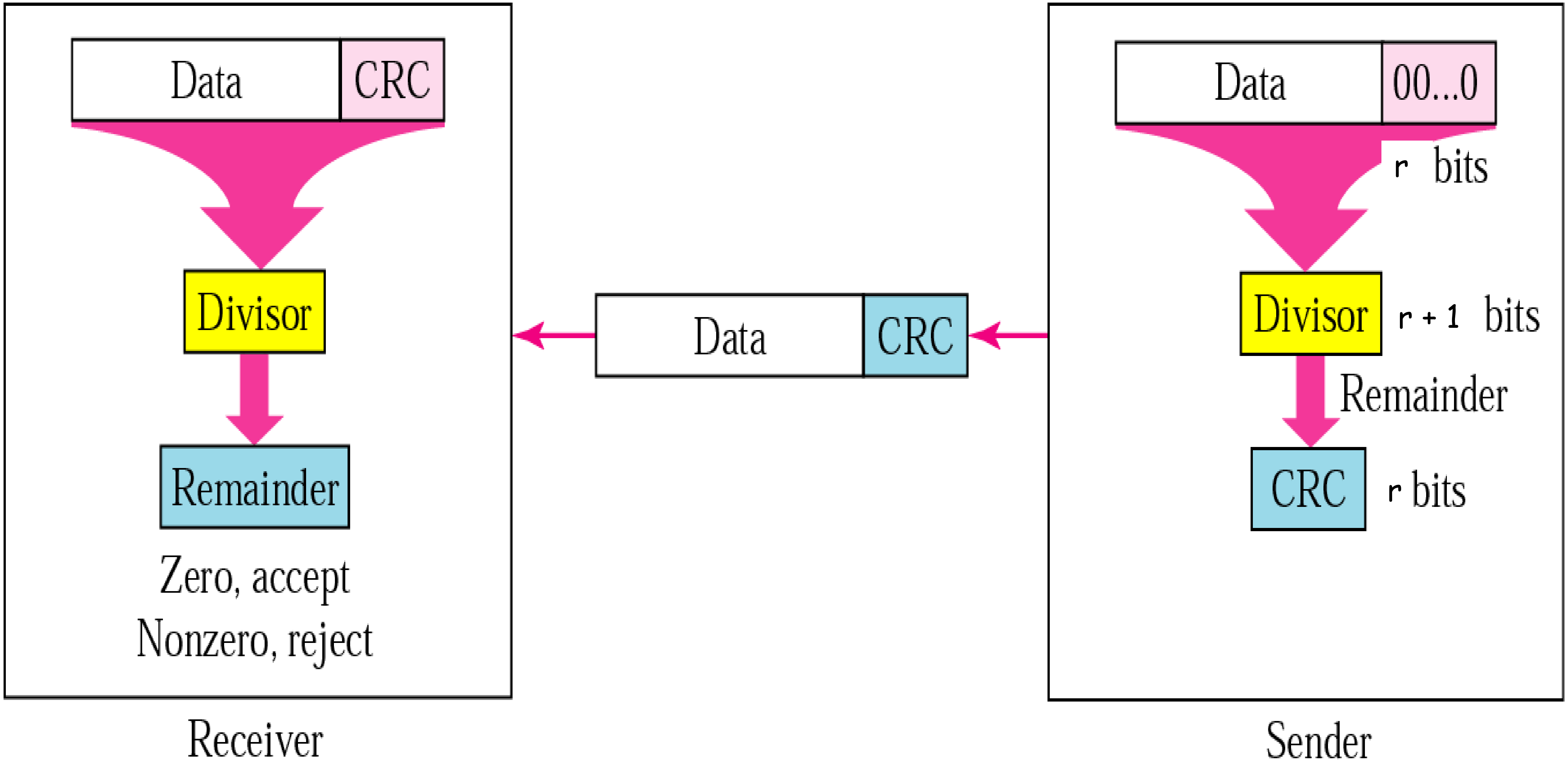
or equivalently:

if we divide $D \cdot 2^r$ by G, want R to satisfy:

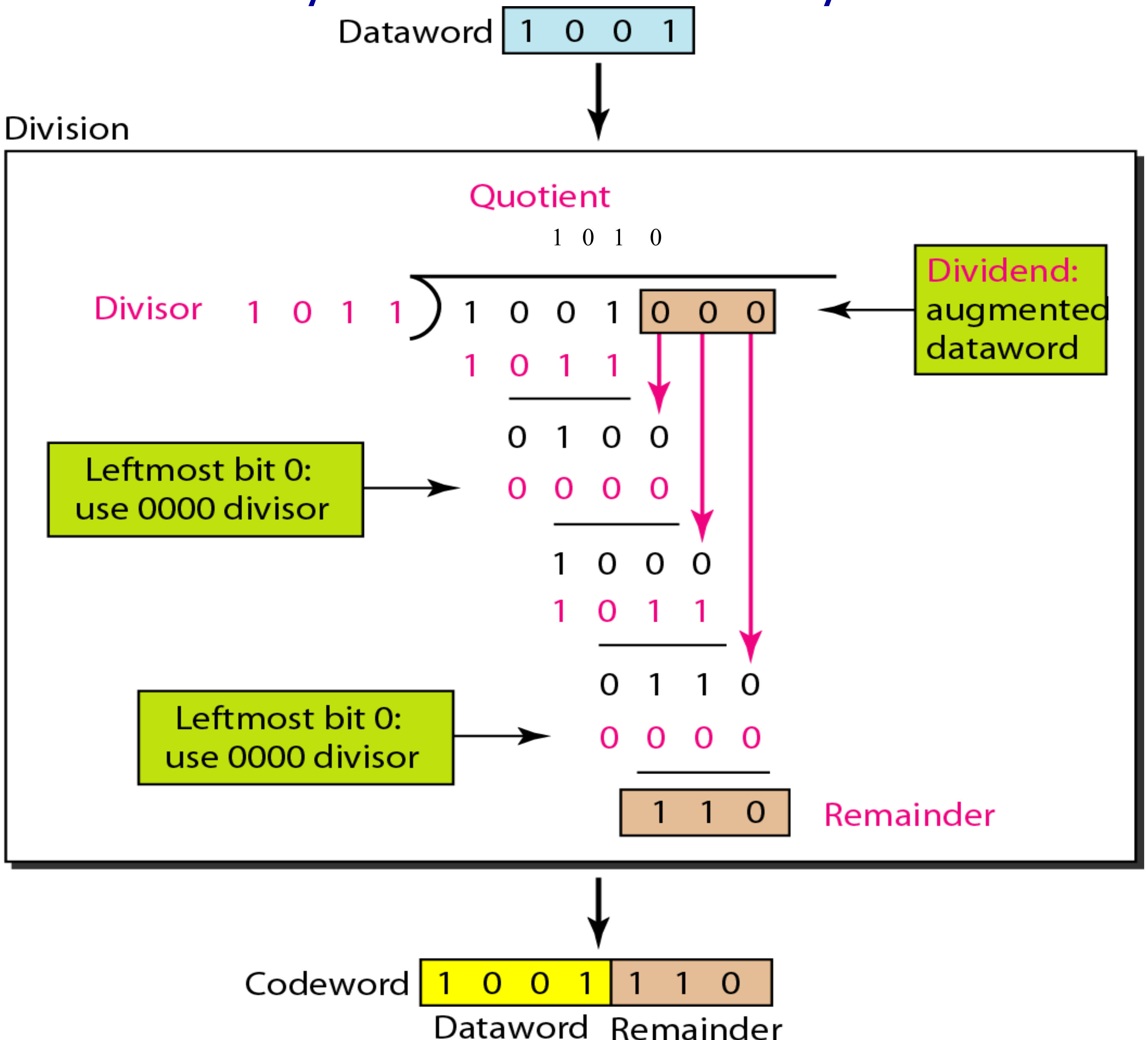
$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$

$$\overbrace{1\;0\;0\;1}^{G}\;\big)\;\underbrace{\boxed{1\;0\;1\;1\;1\;0\;0\;0\;0}}_{D*2^r}$$

Cyclic redundancy check



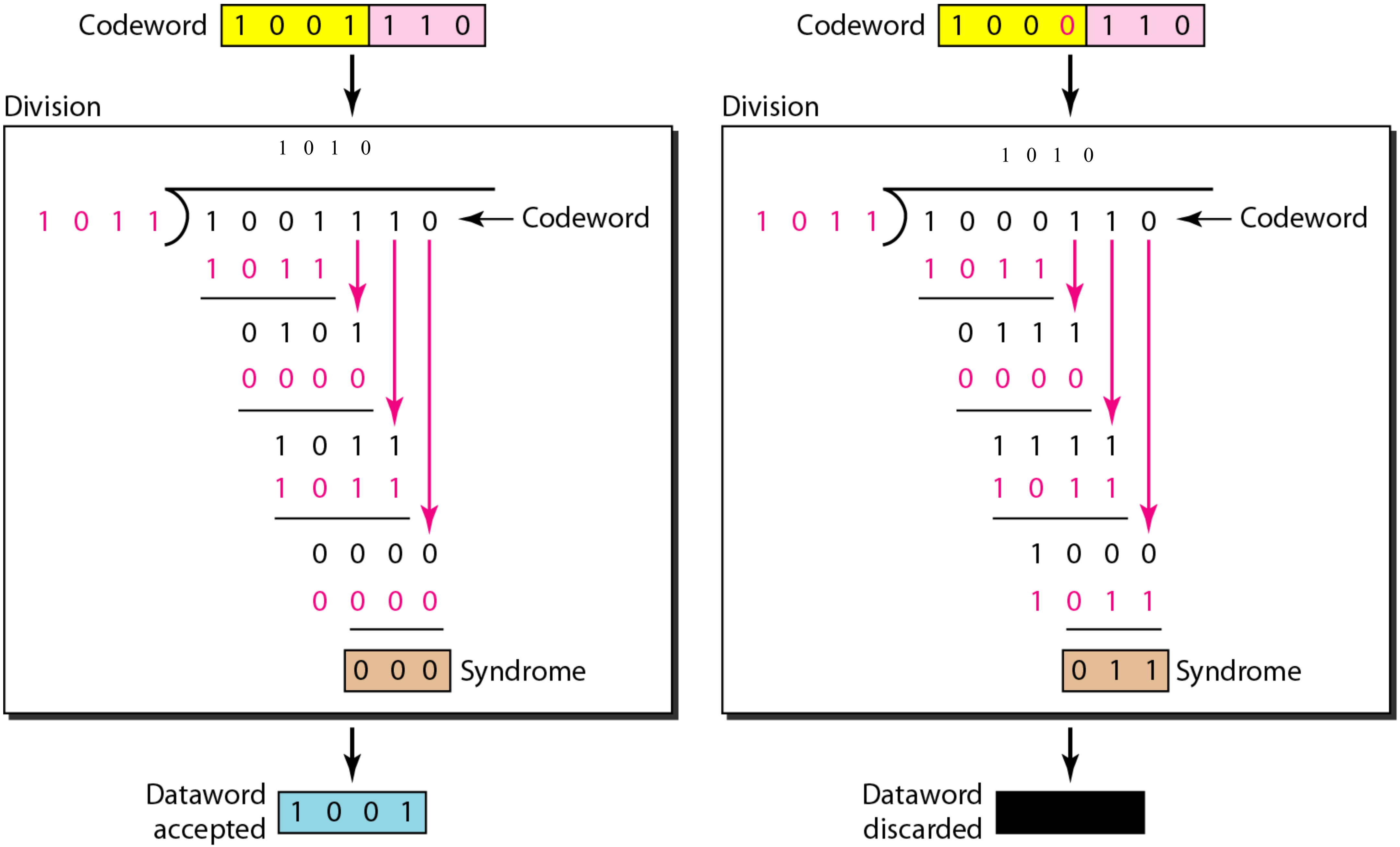
Cyclic redundancy check



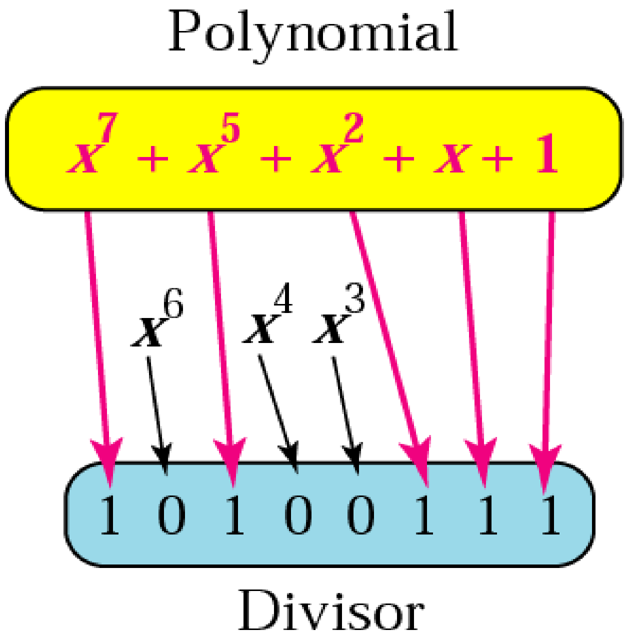
In each step: check the **leading most significant bit**

- If it's 0: place a 0 in the quotient and XOR the current bits with 000.
- If it's 1: place a 1 in the quotient and XOR the current bits with the divisor

Two Cases at the Receiver Side



Clarification: CRC Error Detection
A polynomial representing a divisor (generator)



Standard polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
ITU-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Cyclic Redundancy Check (CRC): example

We want:

$$D \cdot 2^r \ \ XOR \ \ R = nG$$

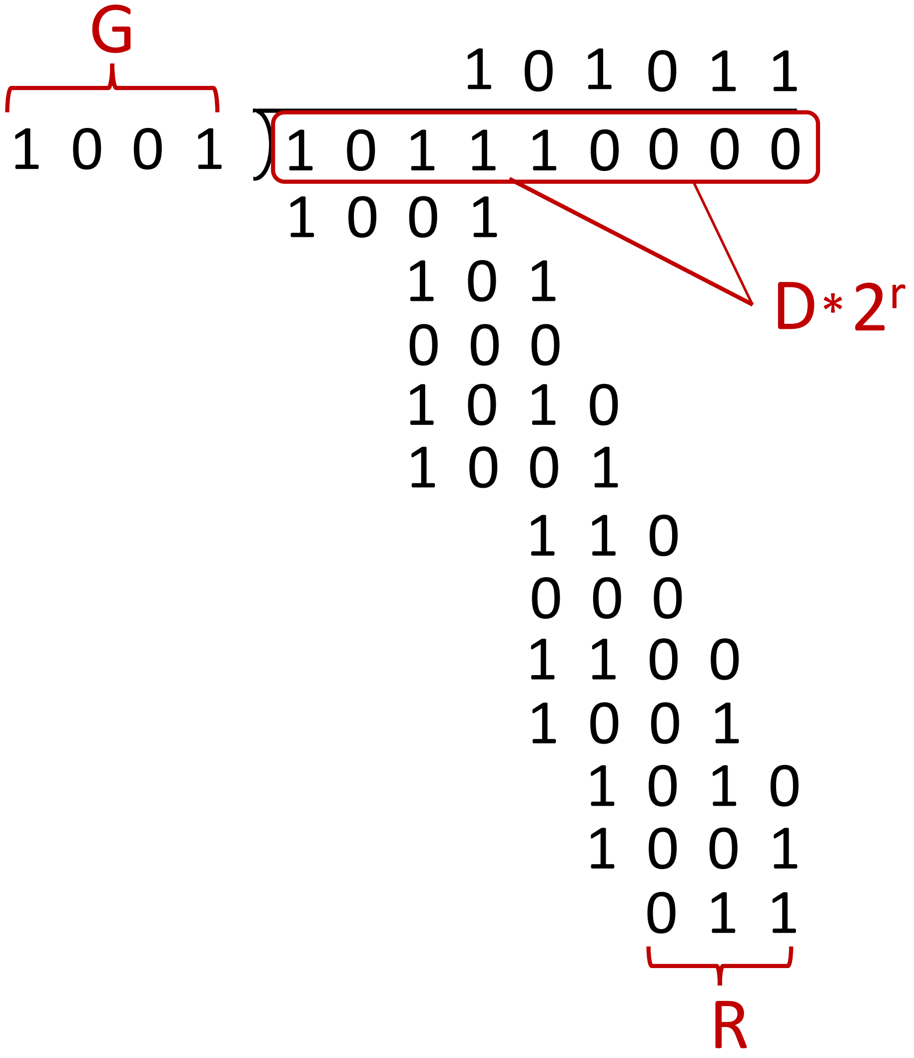
or equivalently:

$$D \cdot 2^r = nG \ \ XOR \ \ R$$

or equivalently:

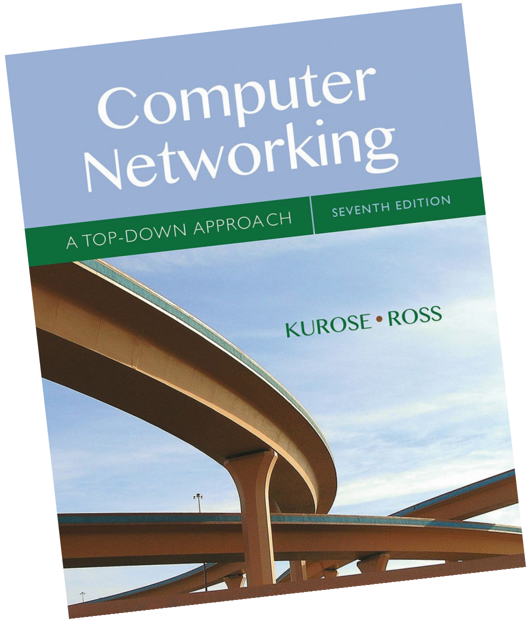
if we divide $D \cdot 2^r$ by G, want R to satisfy:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking
- a day in the life of a web request



Chapter 6

MAC addresses

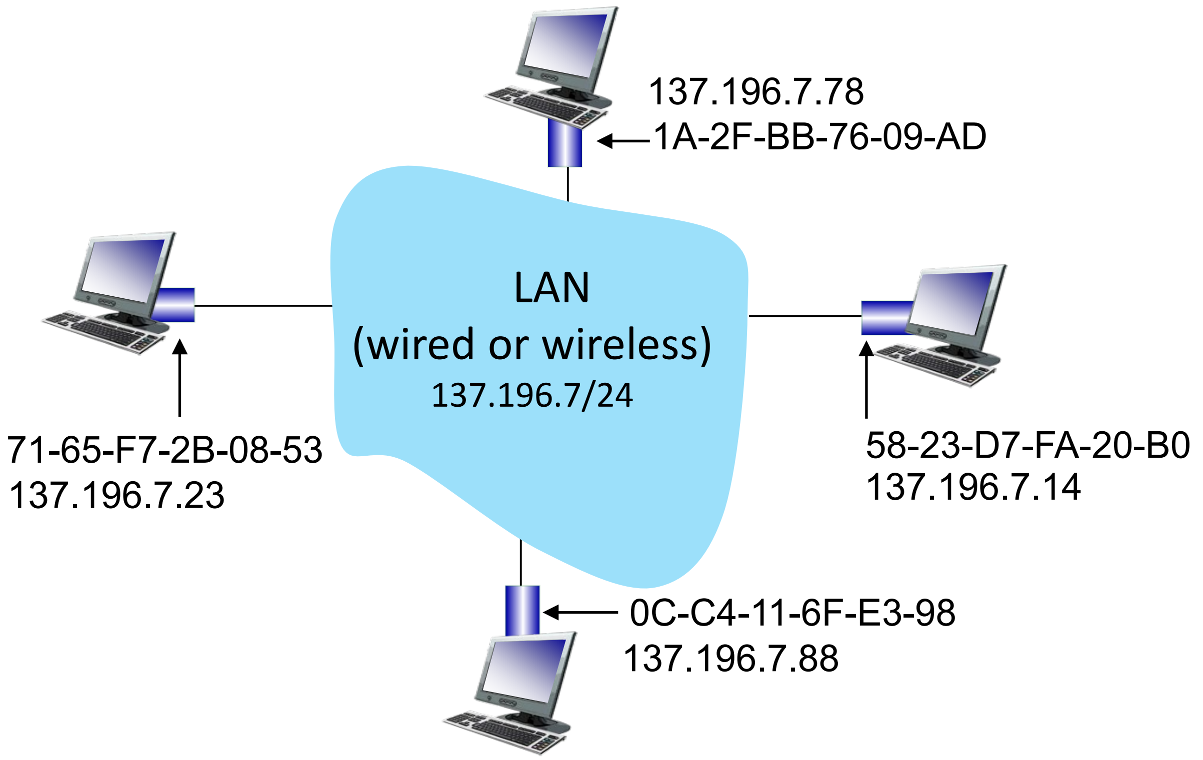
- 32-bit IP address to identify a user in Internet:
  - *Network-layer* address for an interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address to identify a device:
  - used "locally" to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
  - 48-bit MAC address (for most LANs) in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

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

MAC addresses

each interface on LAN

- has a (globally) unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)

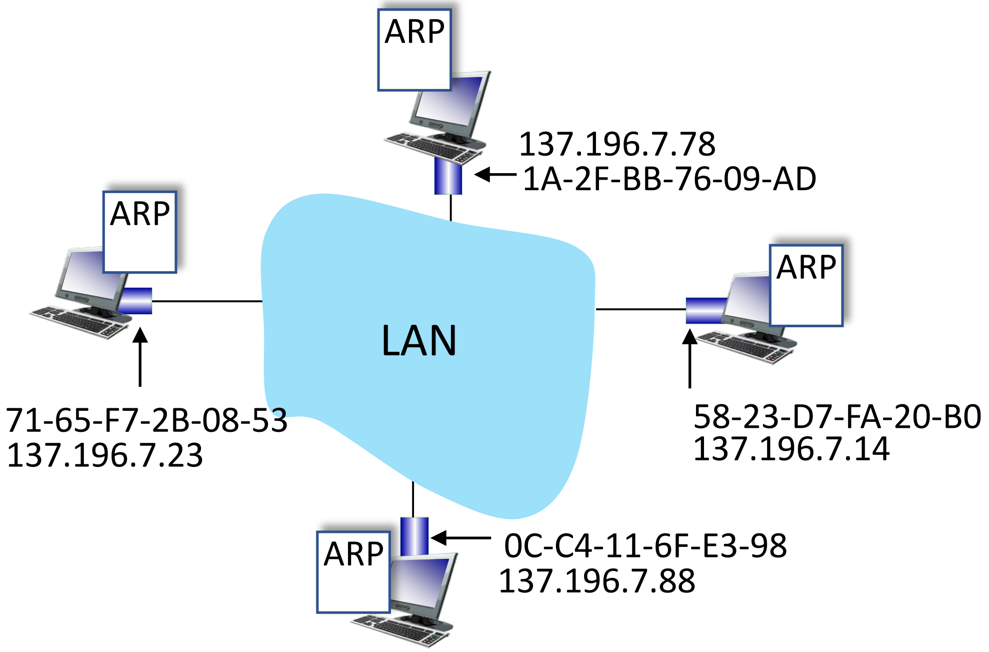


MAC addresses

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like HKID, Social Security Number (USA)
  - IP address: like postal address or student ID
- Flat address space: portability
  - can move interface from one LAN to another, MAC addr fixed
  - recall IP address *not* portable: depends on IP subnet to which node is attached

ARP: address resolution protocol

Question: how to determine interface's MAC address, given the IP address of the interface? (another question: how to find IP address from MAC address?)



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

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

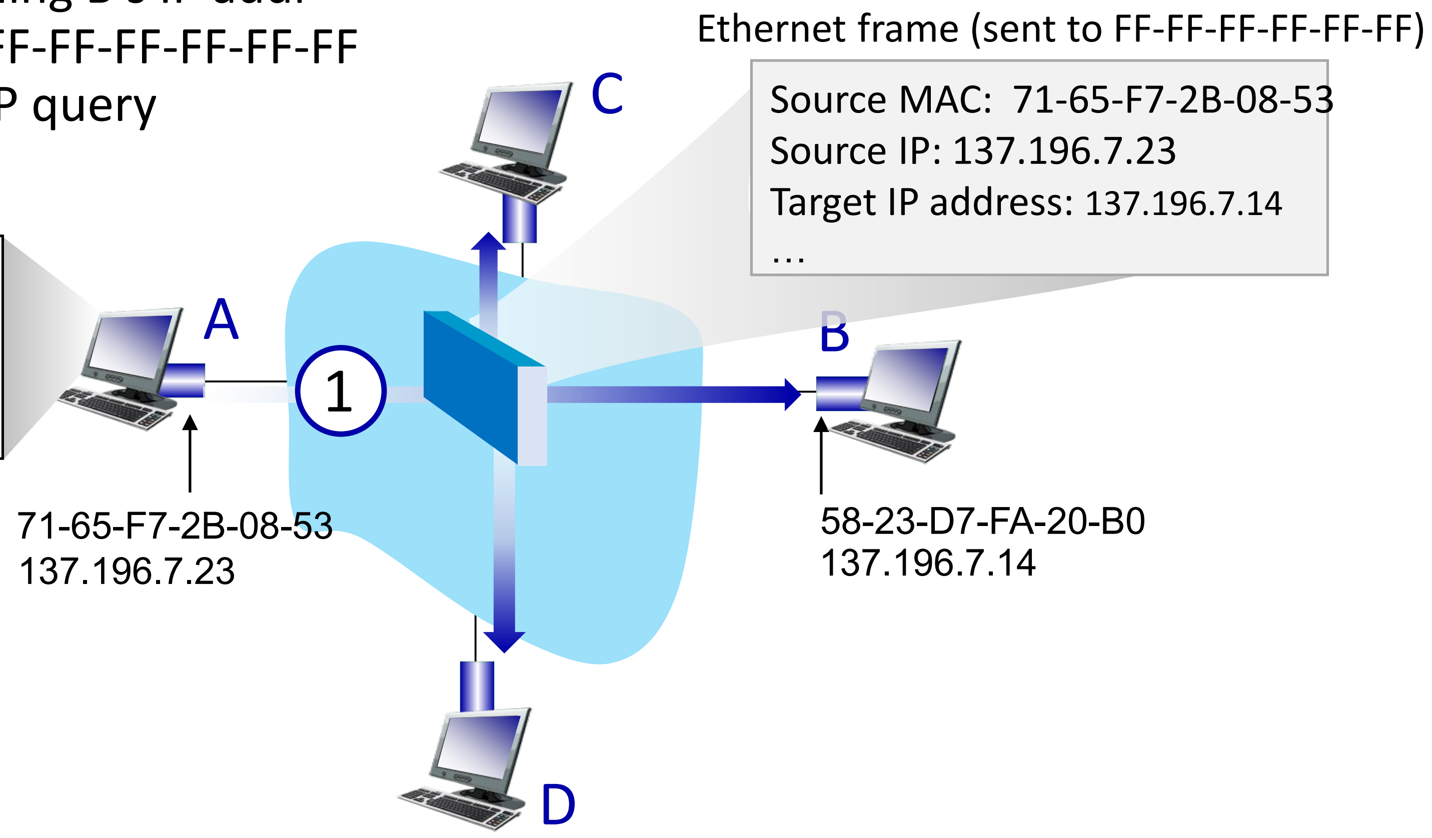
A broadcasts ARP query, containing B's IP addr

1

- destination MAC address = FF-FF-FF-FF-FF-FF
- all nodes on LAN receive ARP query

ARP table in A

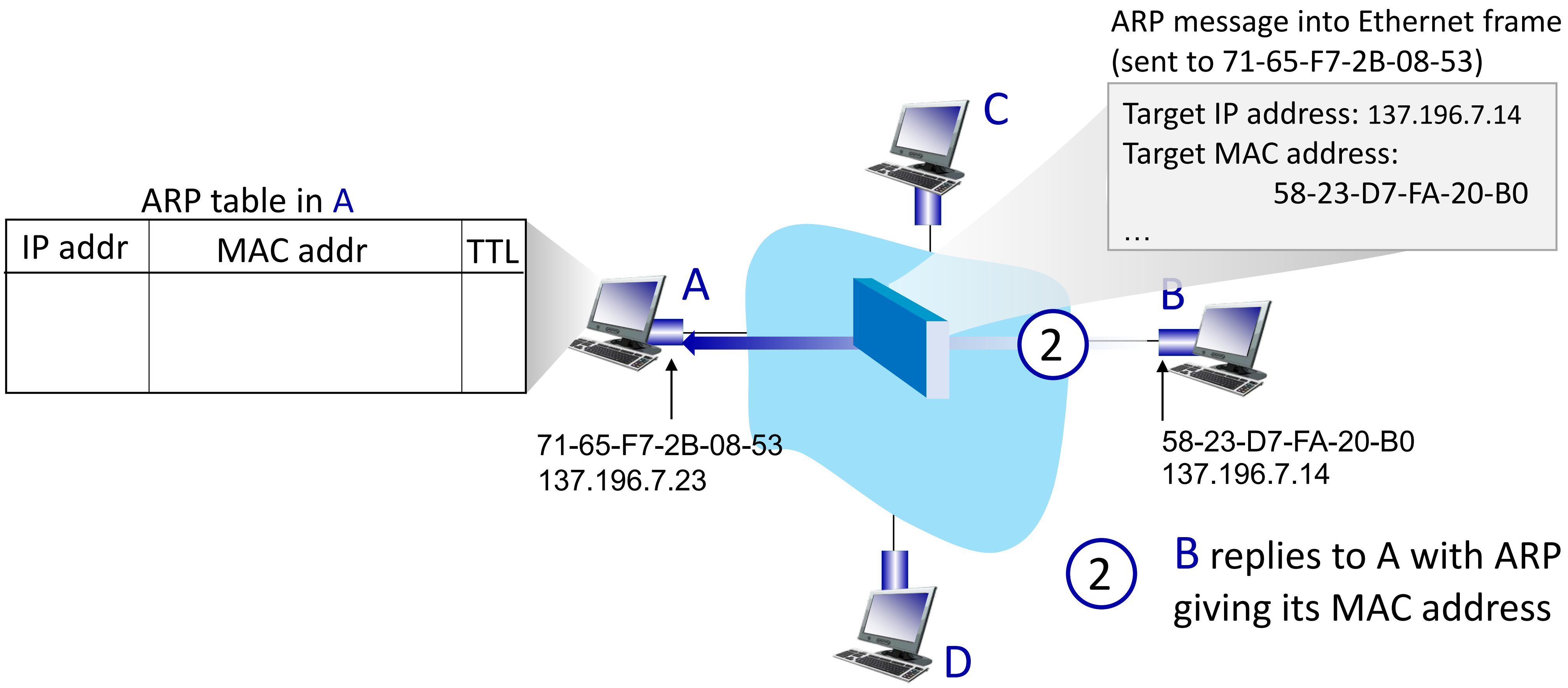
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



2 B replies to A with ARP response, giving its MAC address

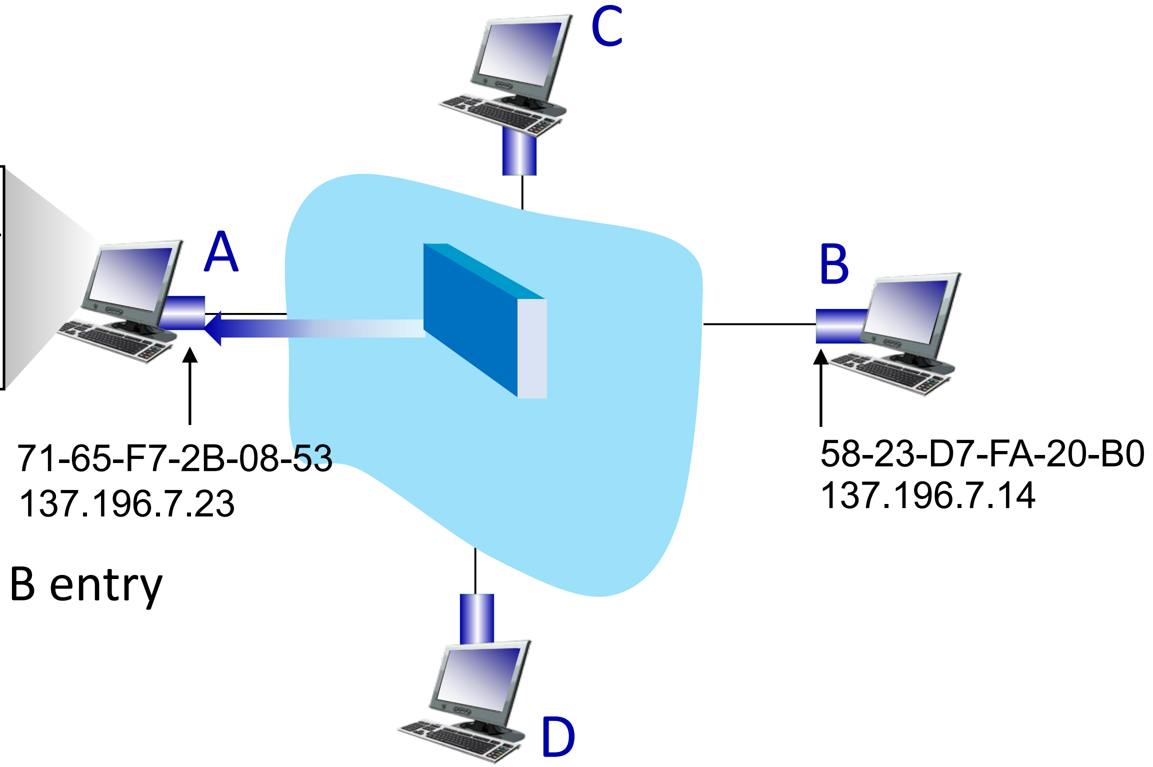
ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

ARP table in A

IP addr	MAC addr	TTL
137.196.7.14	58-23-D7-FA-20-B0	500

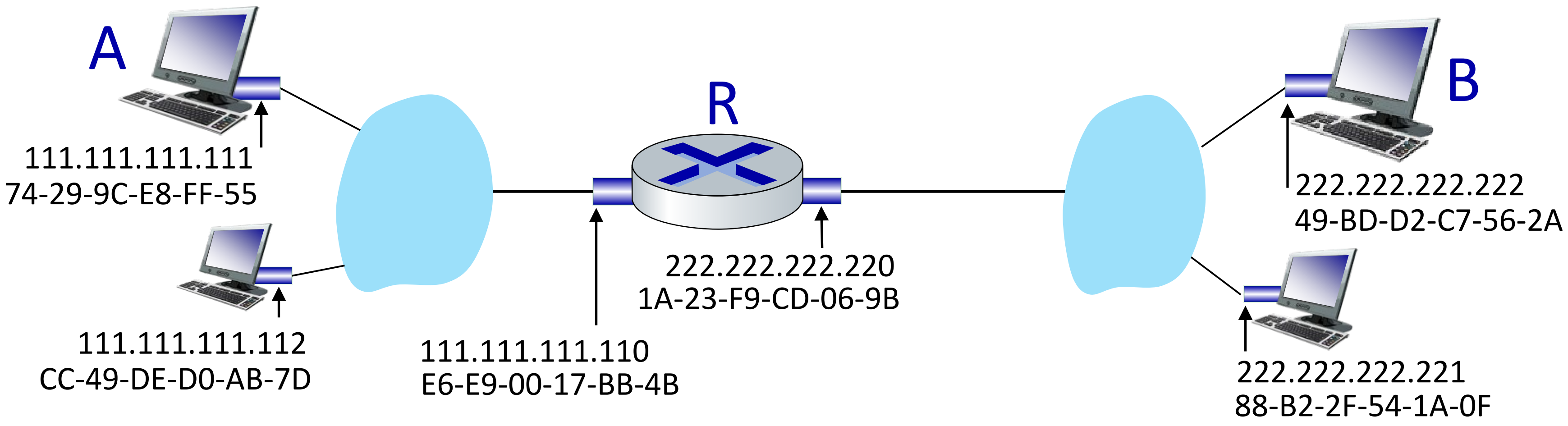


3 A receives B's reply, adds B entry into its local ARP table

Routing to another subnet: addressing

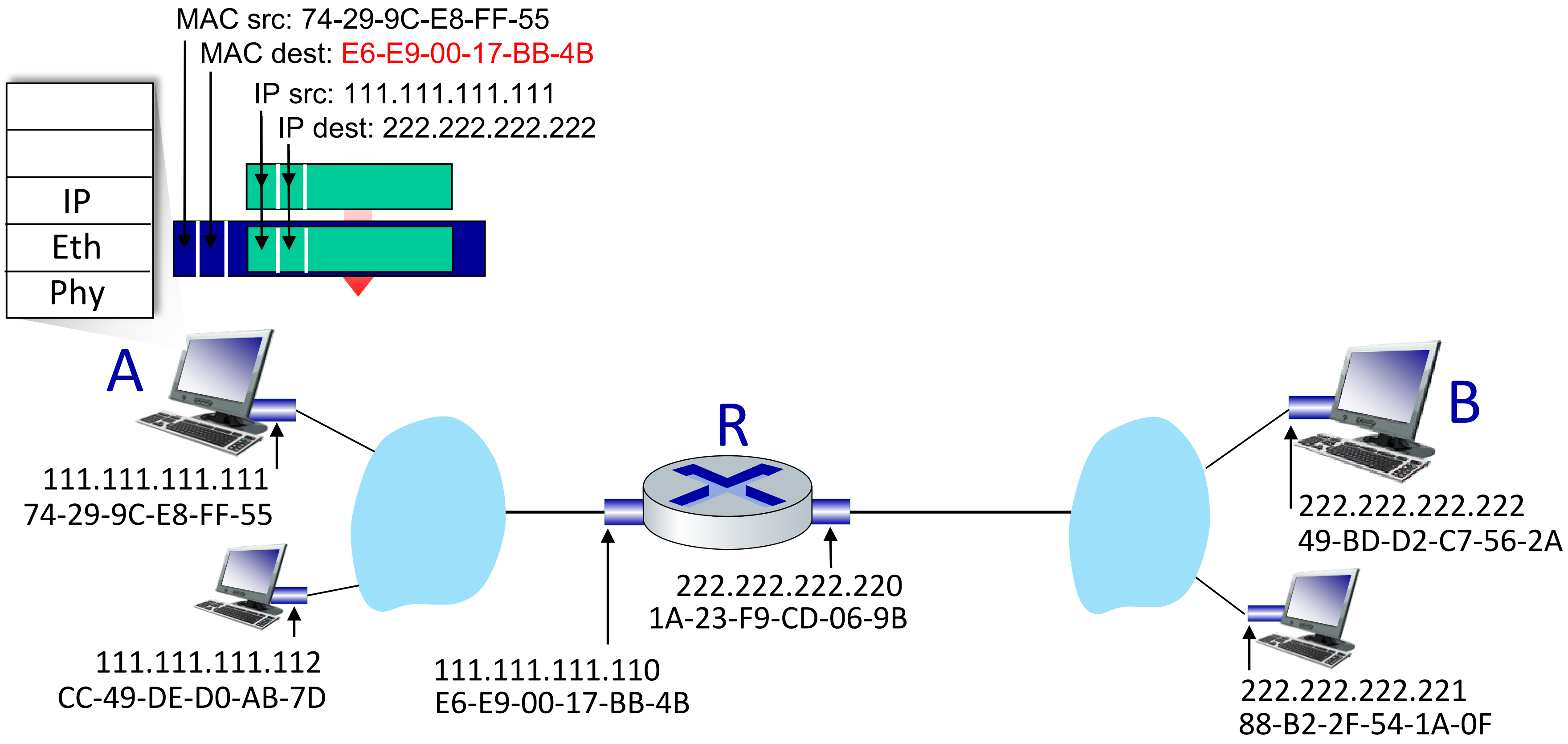
walkthrough: sending a datagram from *A* to *B* via *R*

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



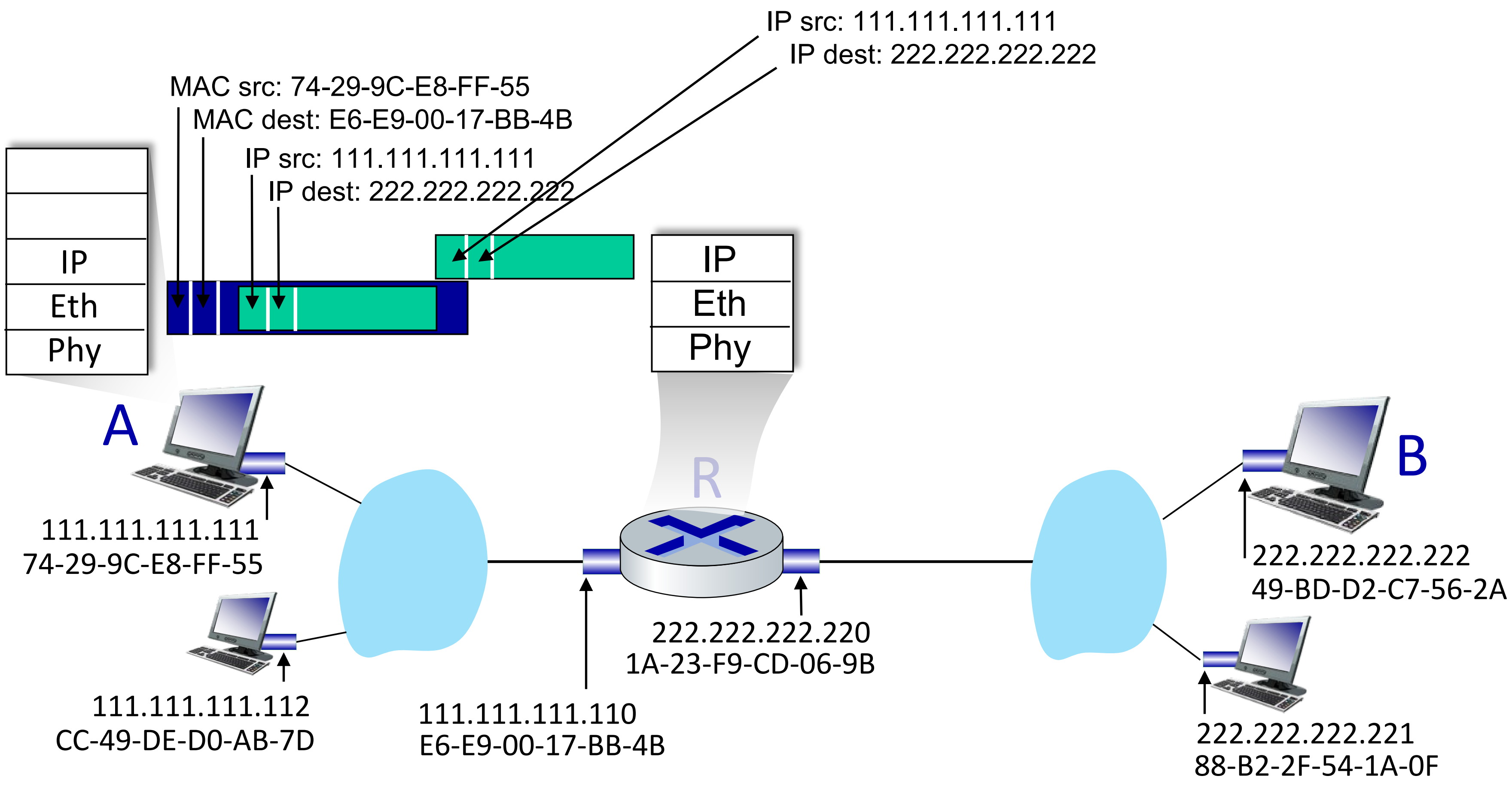
Routing to another subnet: addressing

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame containing A-to-B IP datagram
    - R's MAC address is frame's destination



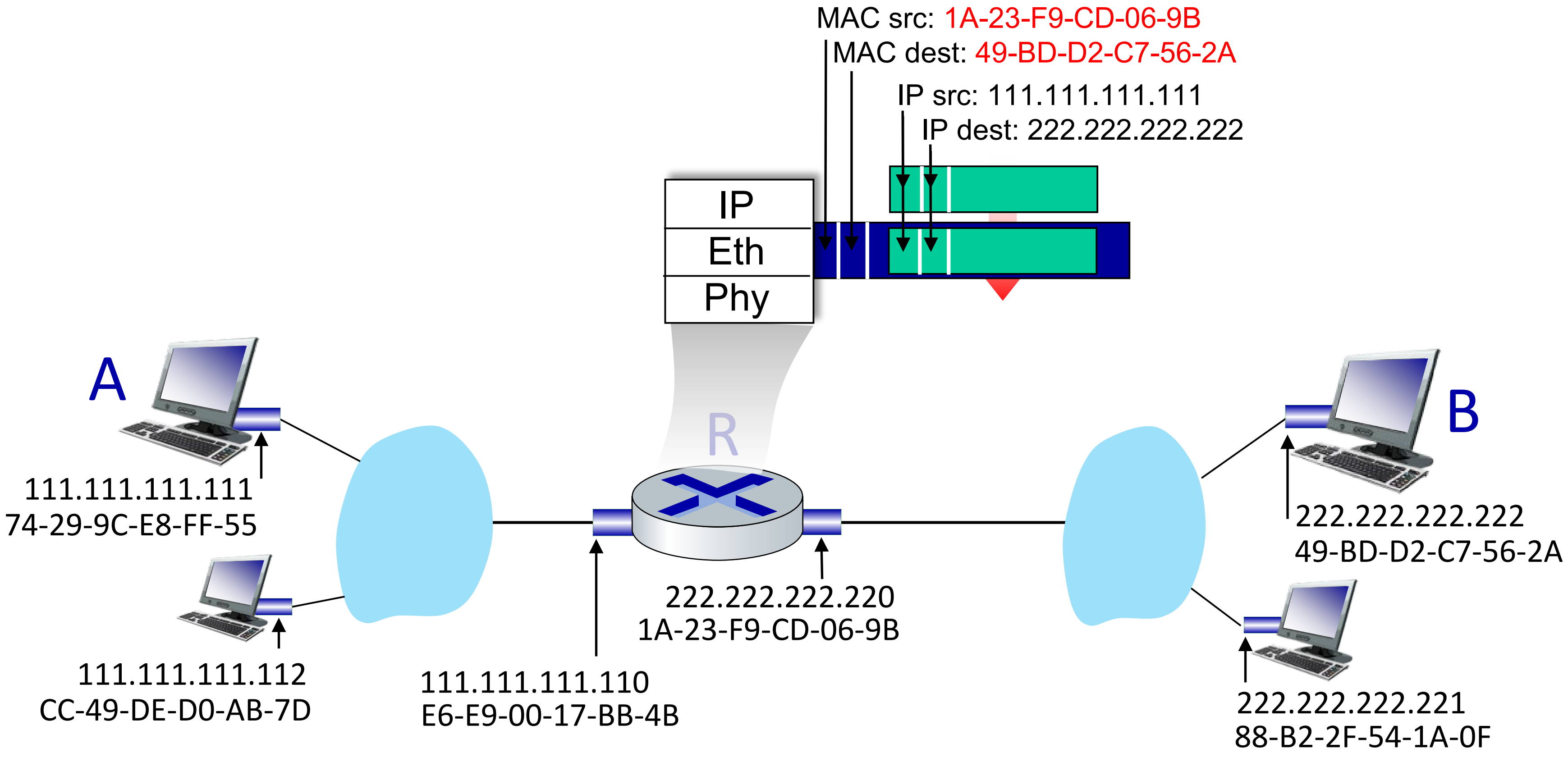
Routing to another subnet: addressing

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



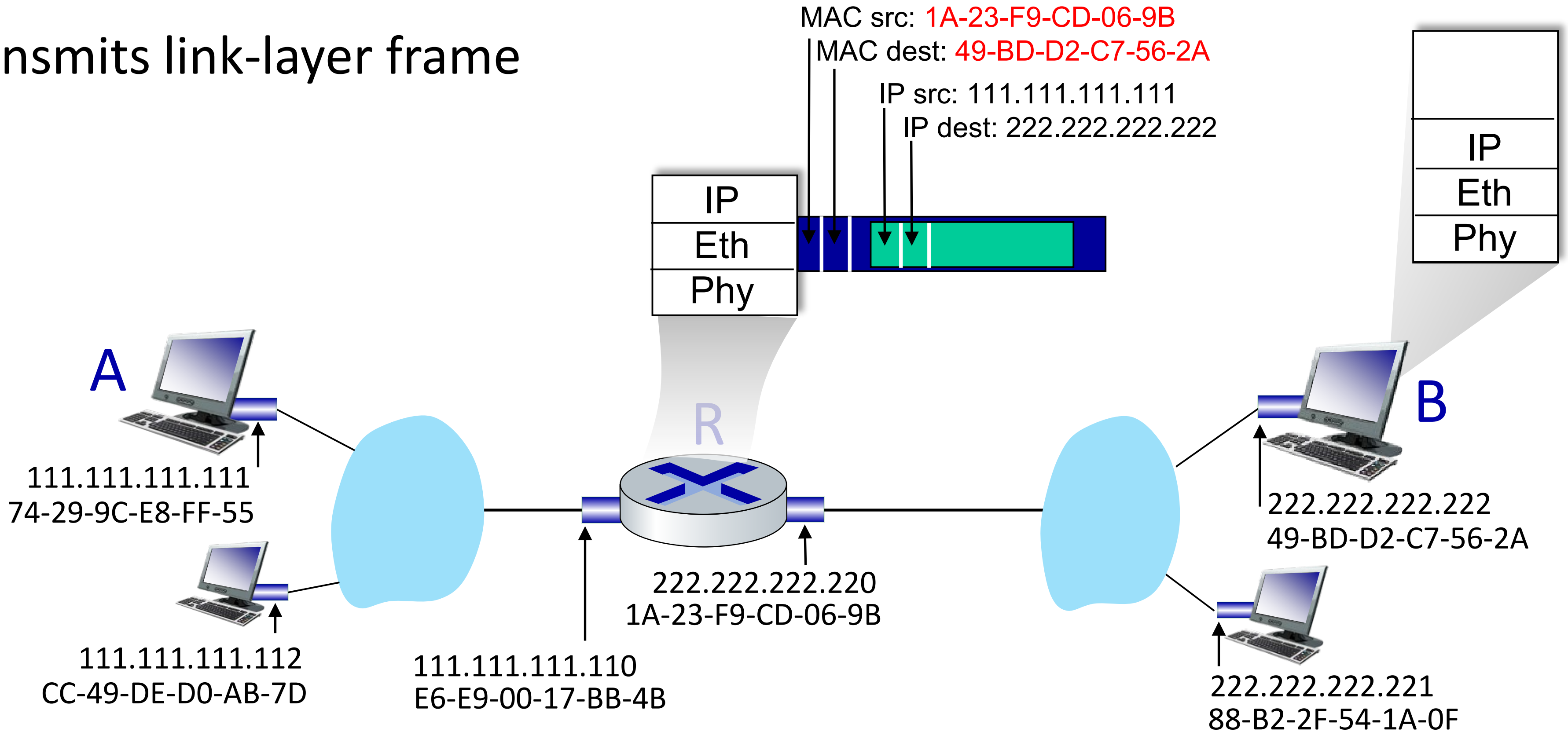
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address



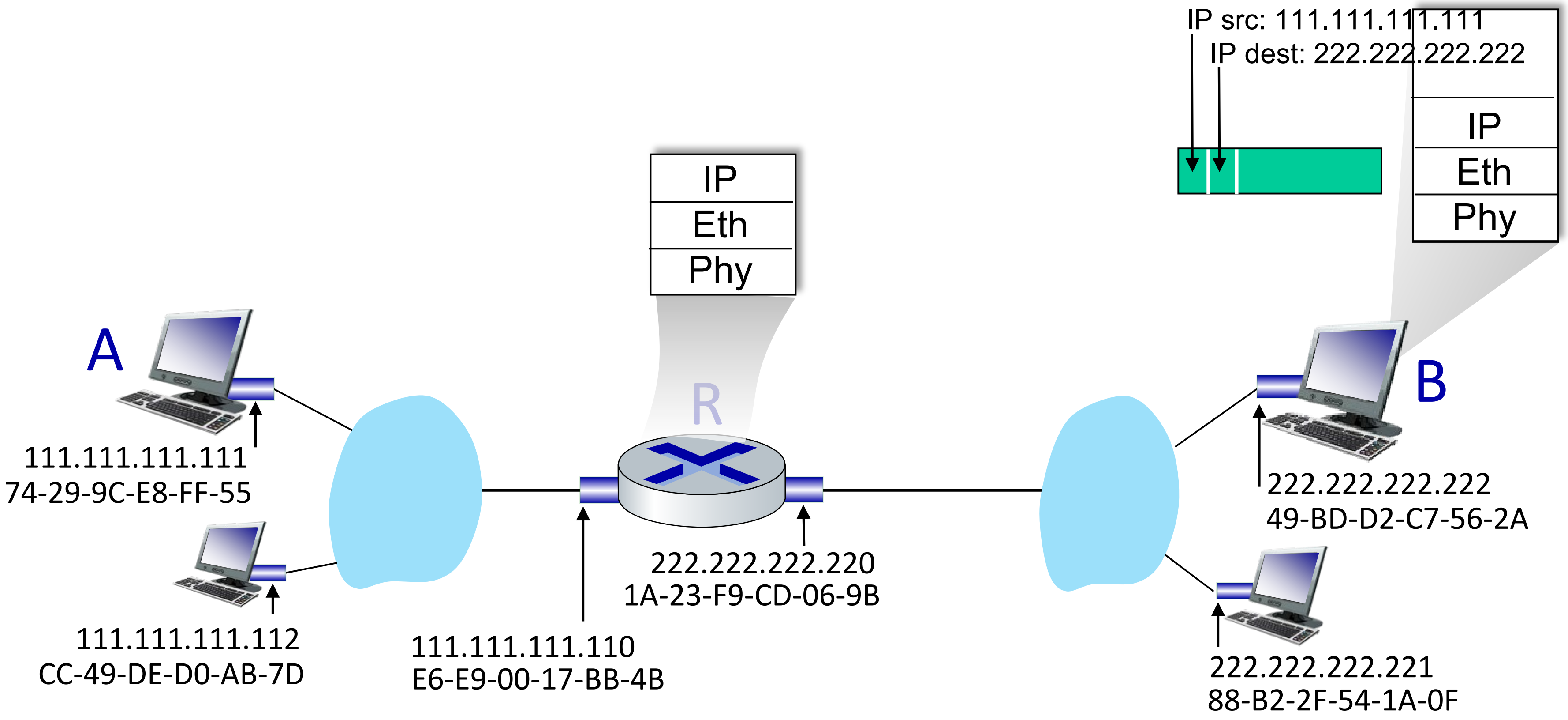
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- transmits link-layer frame



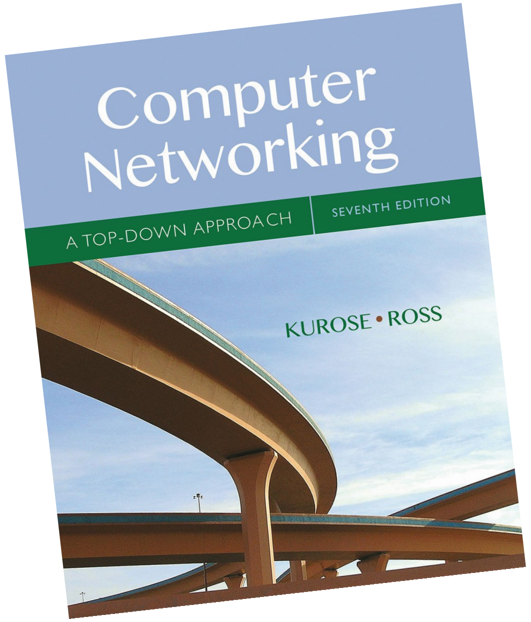
Routing to another subnet: addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking
- a day in the life of a web request



Chapter 6

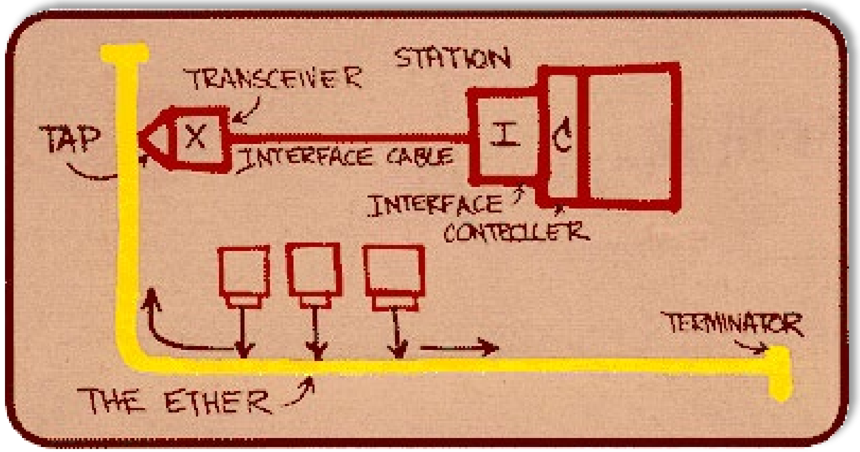
Ethernet switch

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

Ethernet

"dominant" wired LAN technology:

- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 400 Gbps
- single chip, multiple speeds (e.g., Broadcom BCM5761)



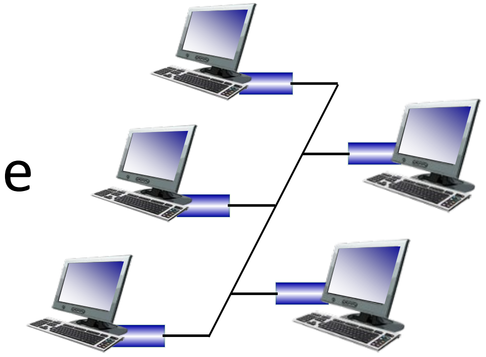
Metcalfe's Ethernet sketch

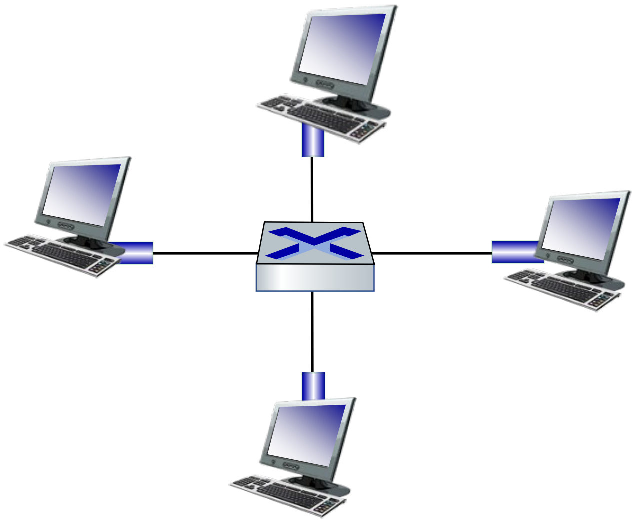
https://www.uspto.gov/learning-and-resources/journeys-innovation/audio-stories/defying-doubters

Ethernet: physical topology

- bus: popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- switched: prevails today
  - active link-layer 2 *switch* in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)

bus: coaxial cable





switched

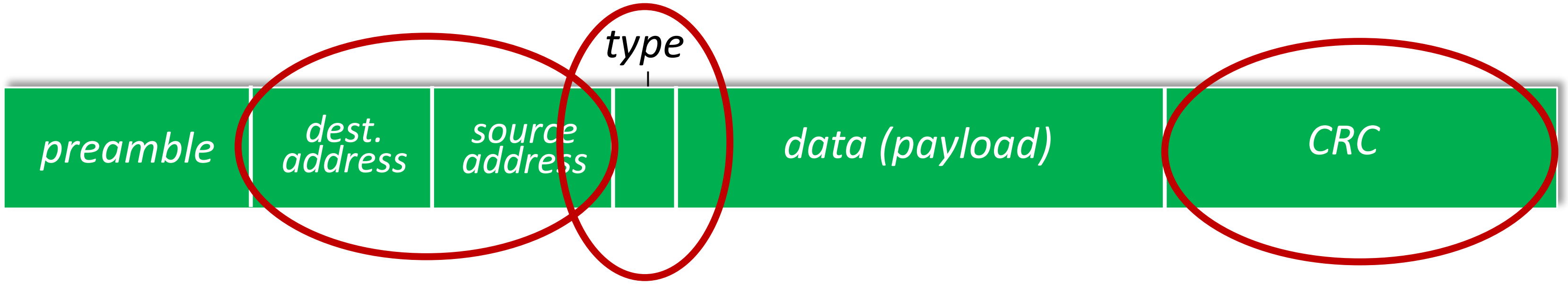
Ethernet frame structure

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

preamble:

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

Ethernet frame structure (more)



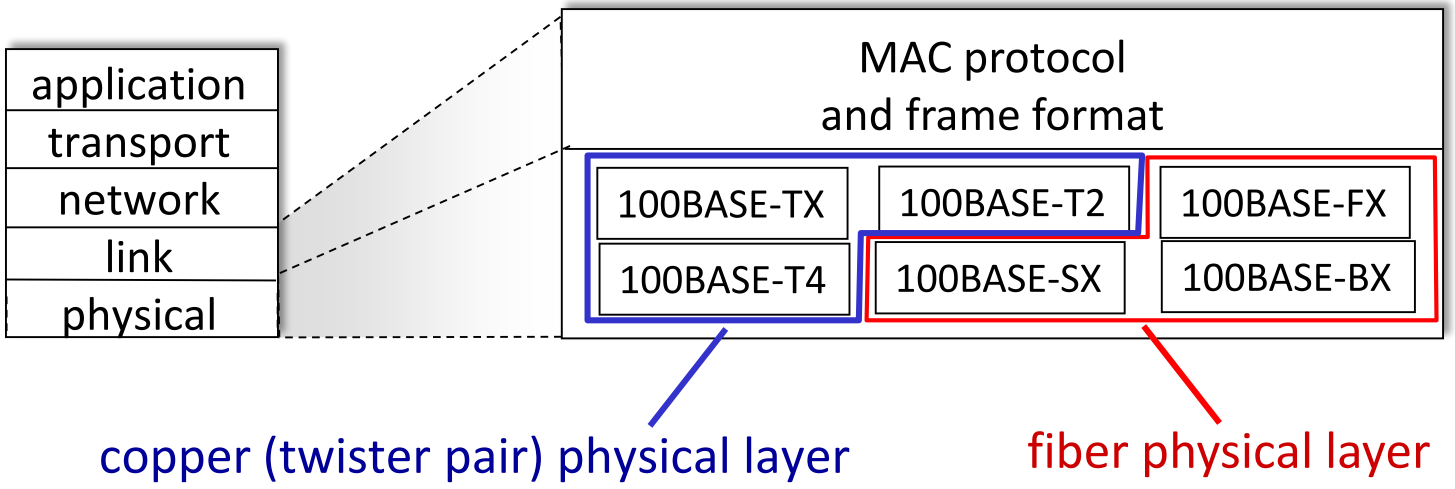
- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- type: indicates higher layer protocol
  - mostly IP but others possible, e.g., Novell IPX, AppleTalk
  - used to demultiplex up at receiver
- 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 does not send ACKs or NAKs 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

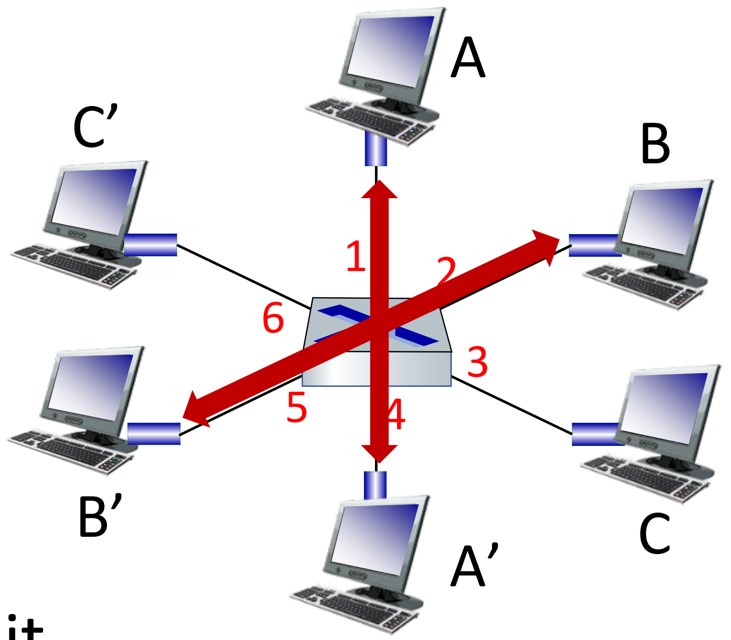
802.3 Ethernet standards: link & physical layers

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



Switch: multiple simultaneous transmissions

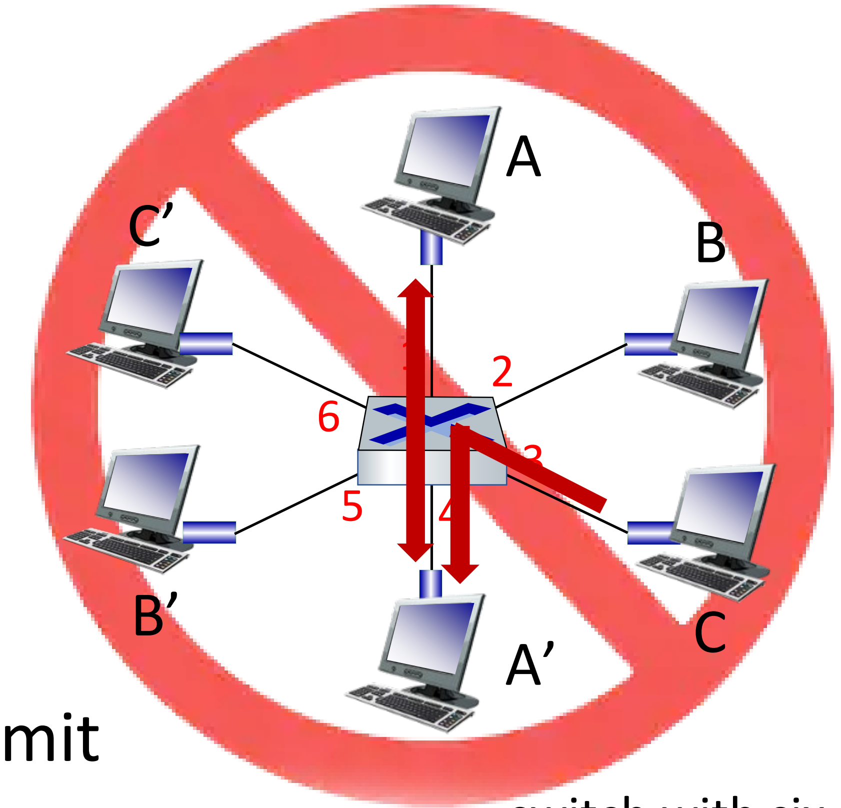
- Switched Ethernet
- hosts have dedicated, direct connections to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A’ and B-to-B’ can transmit simultaneously, without collisions



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

Switch: multiple simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions
  - but A-to-A' and C to A' cannot transmit simultaneously



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

Switch table

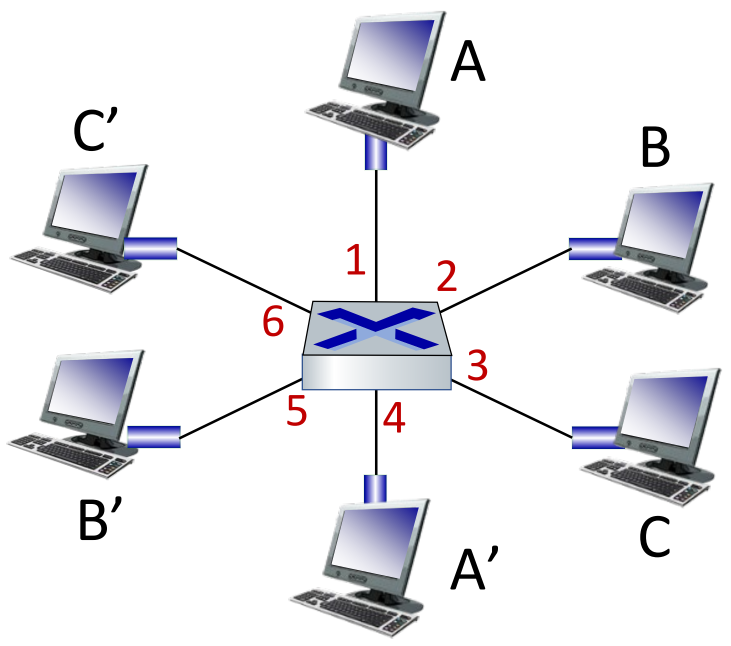
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

A: a switch has a switch table:

- each entry: (MAC address of host, interface to reach host, time stamp)
- looks like a forwarding table!

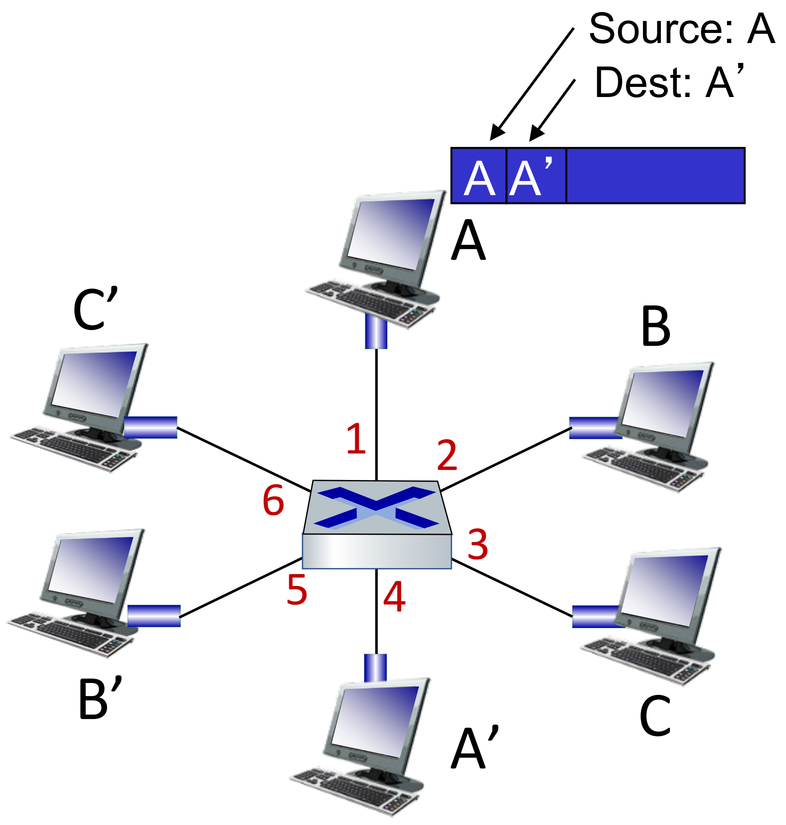
Q: how are entries created, maintained in switch table?

- something like a routing protocol?



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)

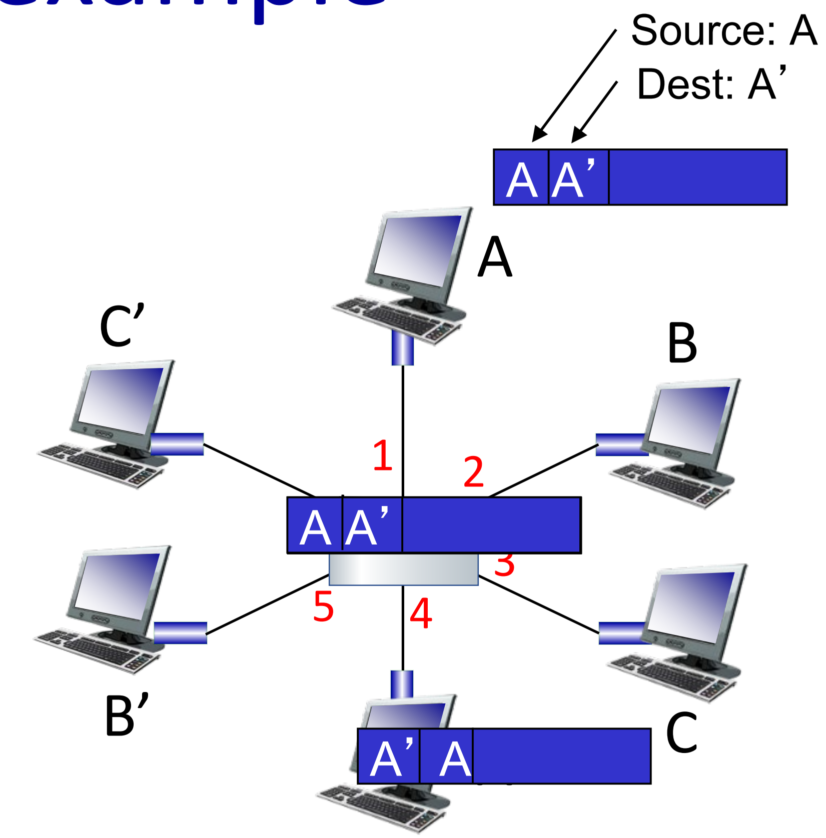
Switch: frame filtering/forwarding

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

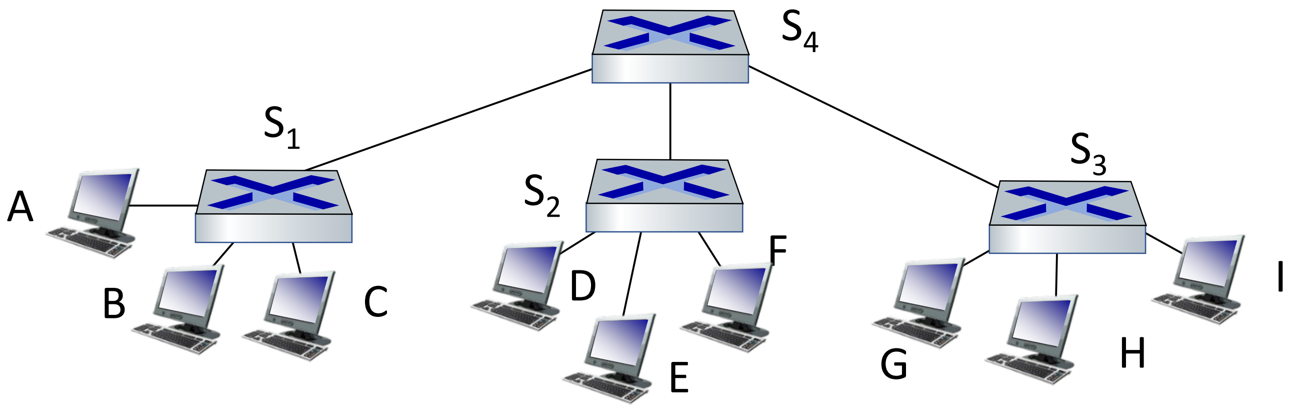


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

self-learning switches can be connected together:

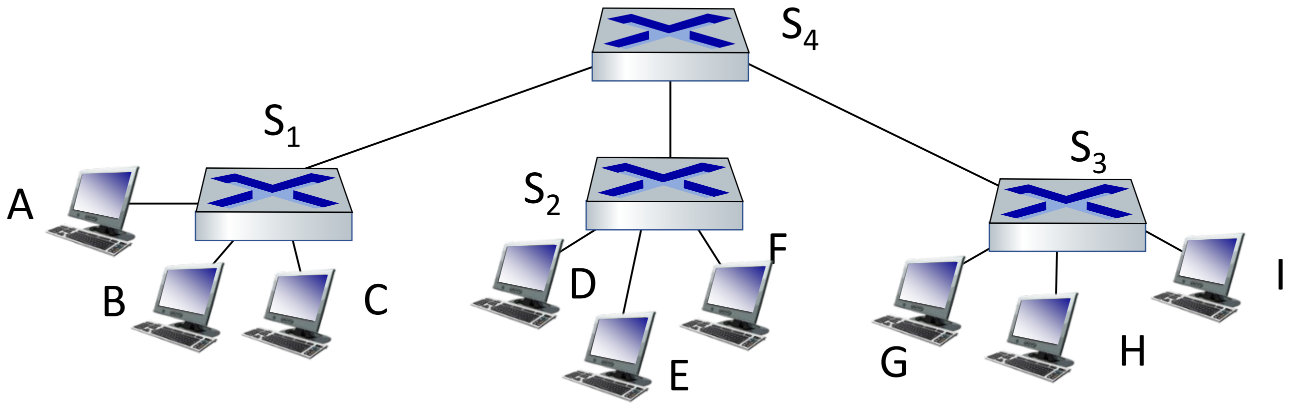


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

- *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

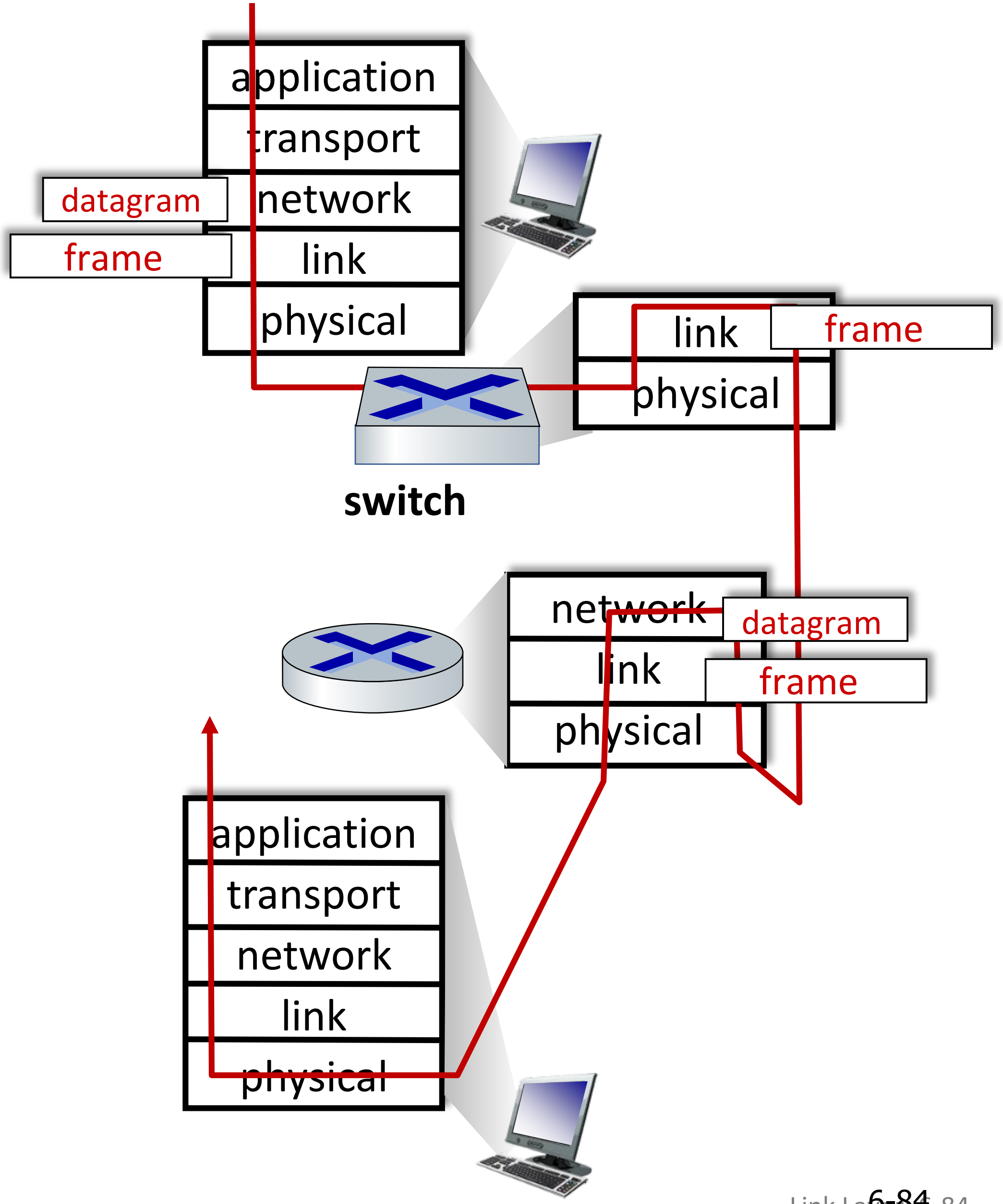
Switches vs. routers

both are store-and-forward:

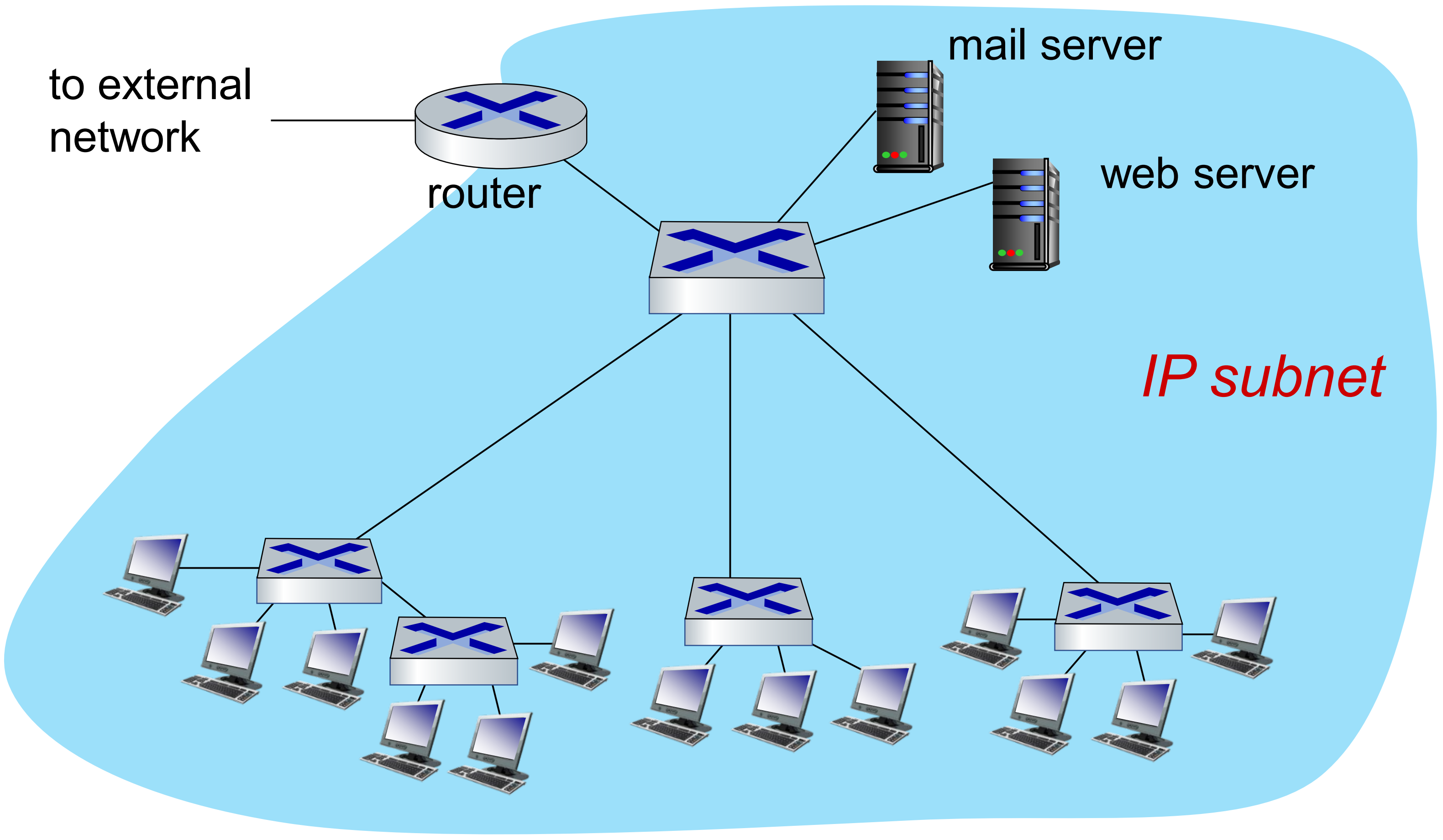
- *routers*: network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

both have forwarding tables:

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

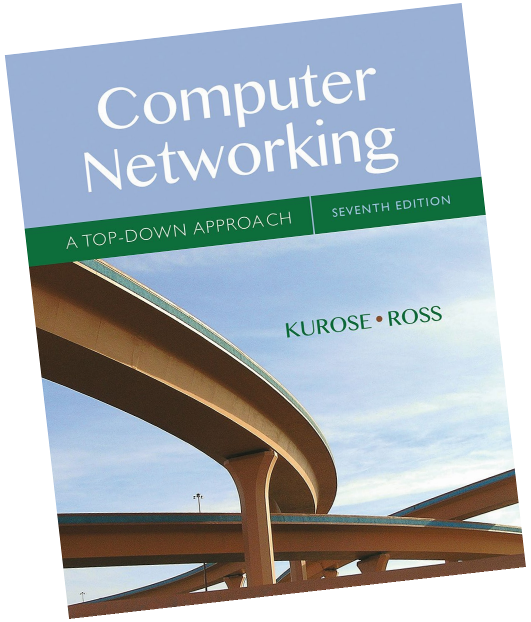


Small institutional network



Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - switches
- data center networking
- a day in the life of a web request



Chapter 6

Datacenter networks

10's to 100's of thousands of hosts, often closely coupled, in close proximity:

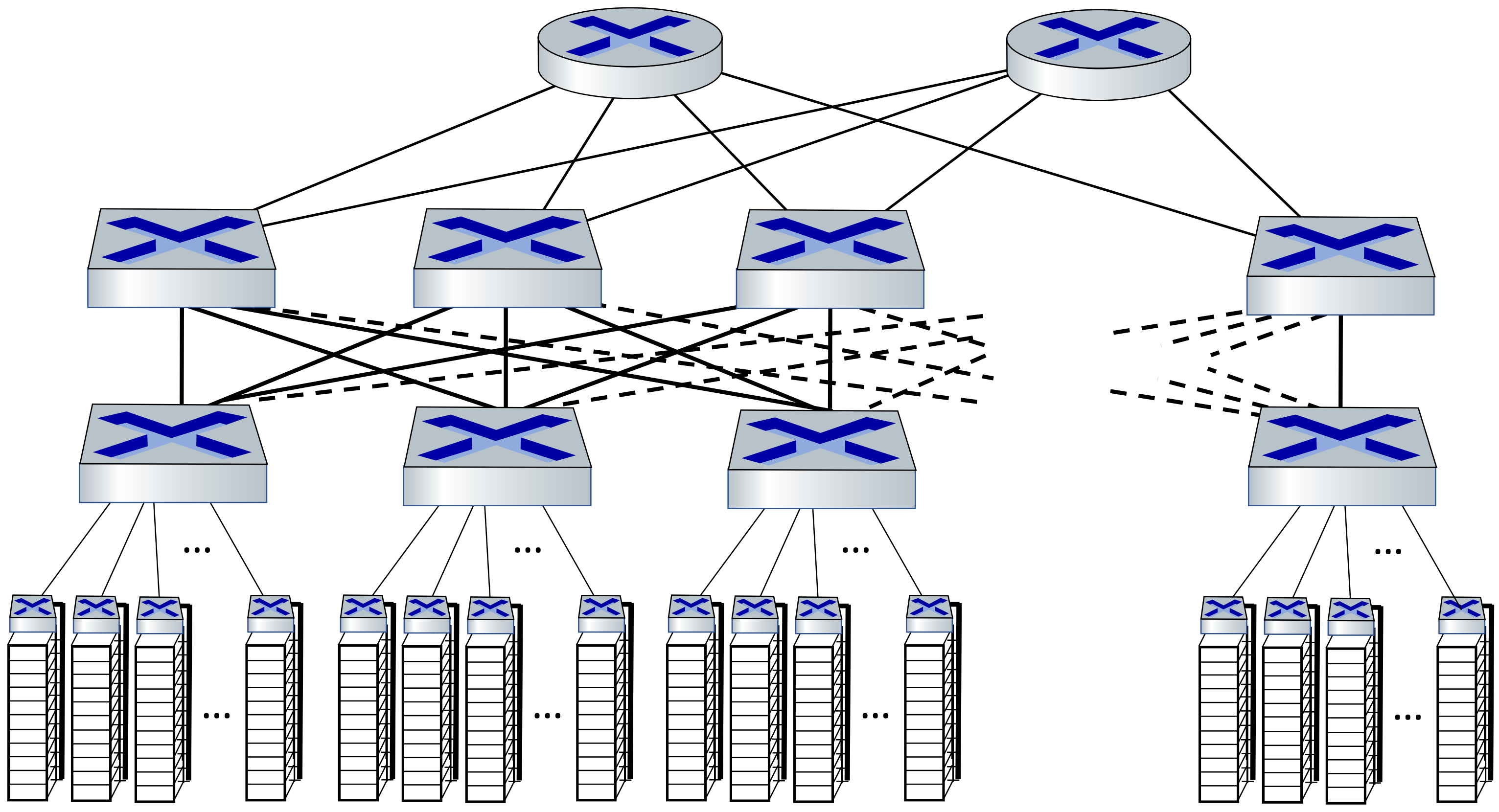
- e-business (e.g., Amazon)
- content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
- search engines, data mining (e.g., Google)

challenges:

- multiple applications, each serving massive numbers of clients/users
- reliability
- managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Datacenter networks: network elements



Border routers
- connections outside datacenter

Tier-1 switches
- connecting to ~16 T-2s below

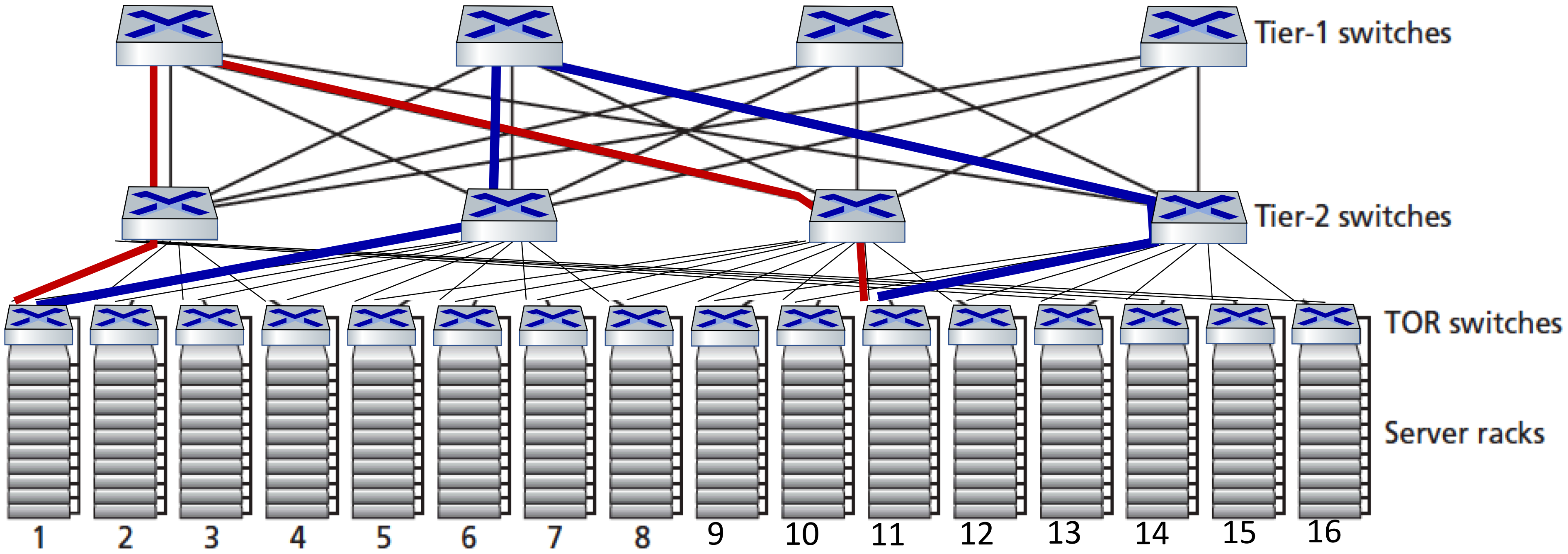
Tier-2 switches
- connecting to ~16 TORs below

Top of Rack (TOR) switch
- one per rack
- 40-100Gbps Ethernet to blades

Server racks
- 20- 40 server blades: hosts

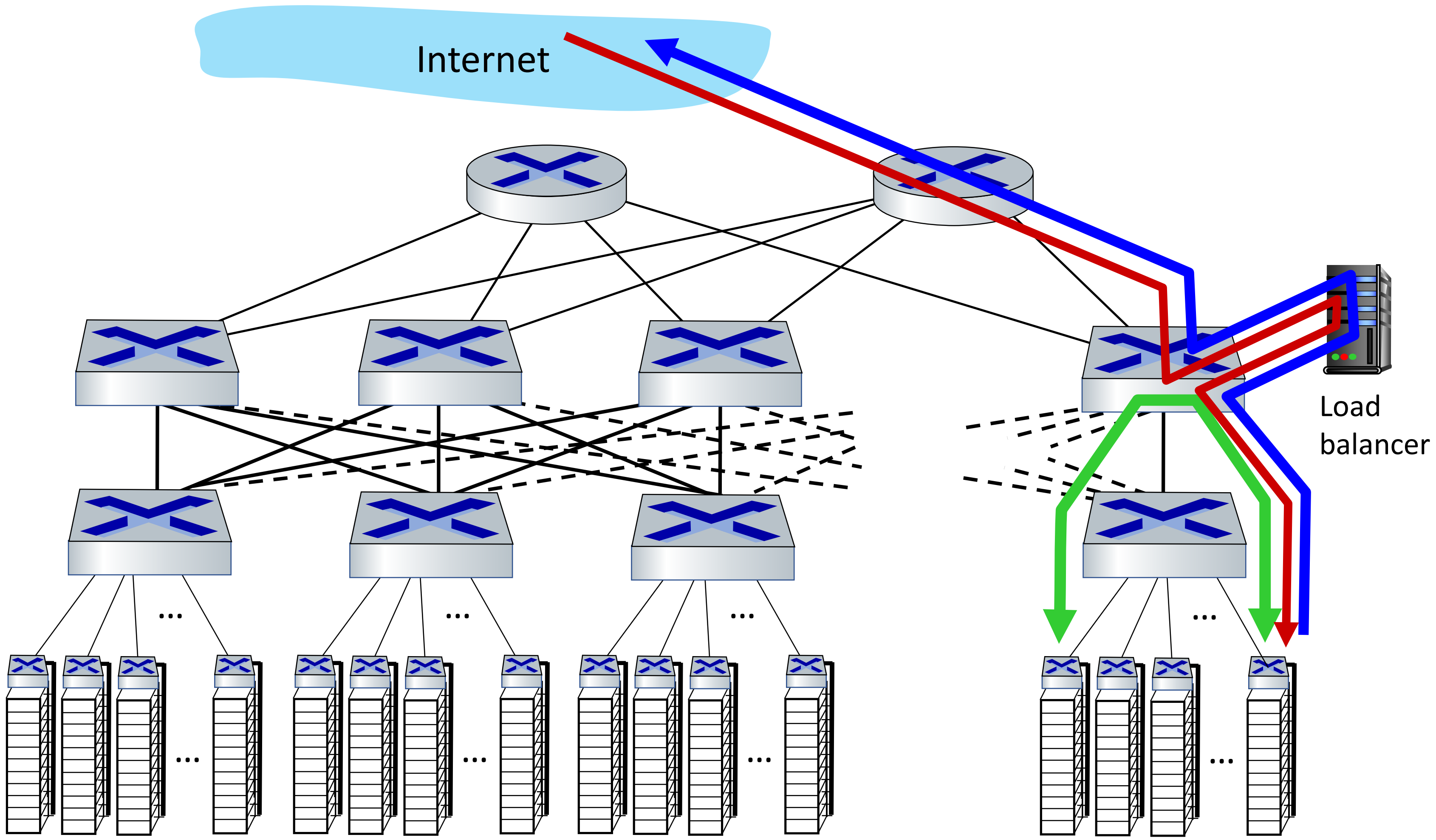
Datacenter networks: multipath

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



two disjoint paths highlighted between racks 1 and 11 (red and blue)

Datacenter networks: application-layer routing

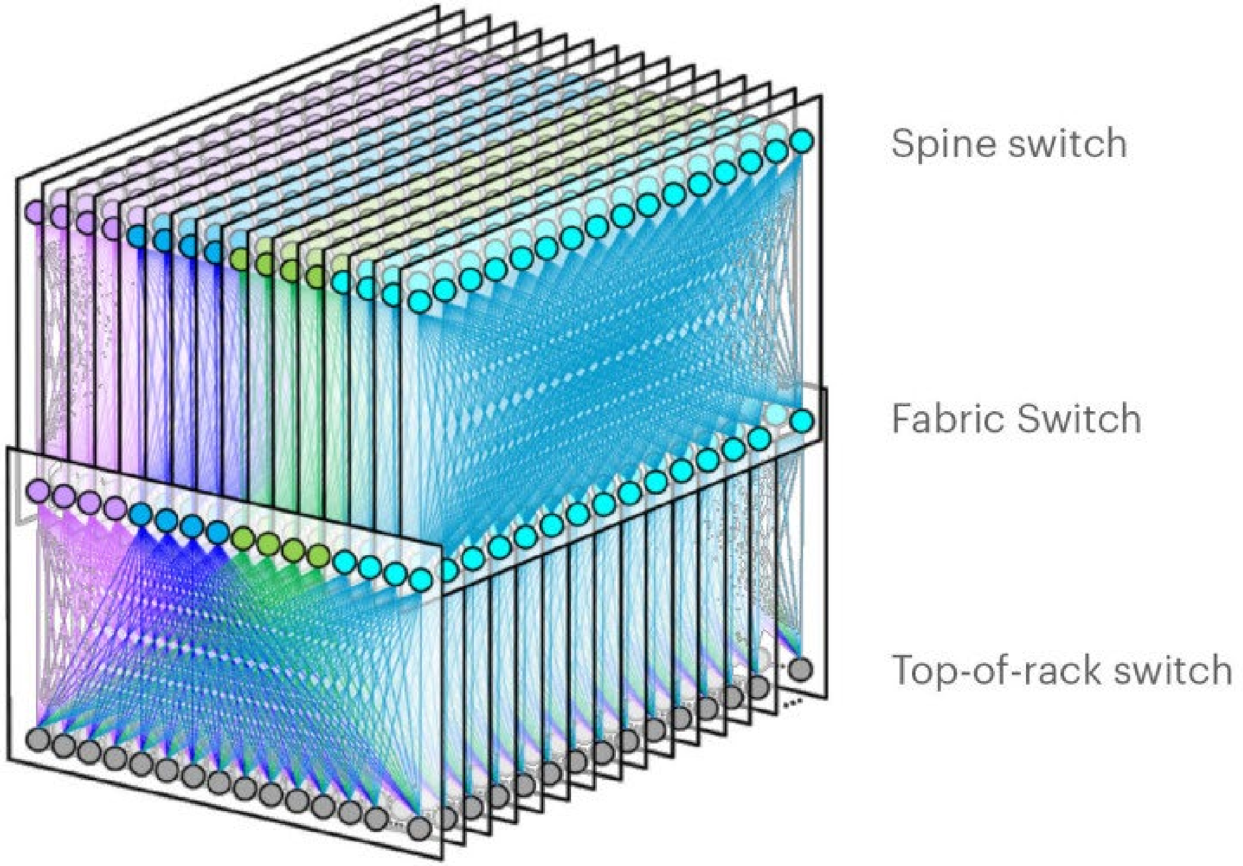


load balancer: application-layer routing

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

Datacenter networks: network elements

Facebook F16 data center network topology:



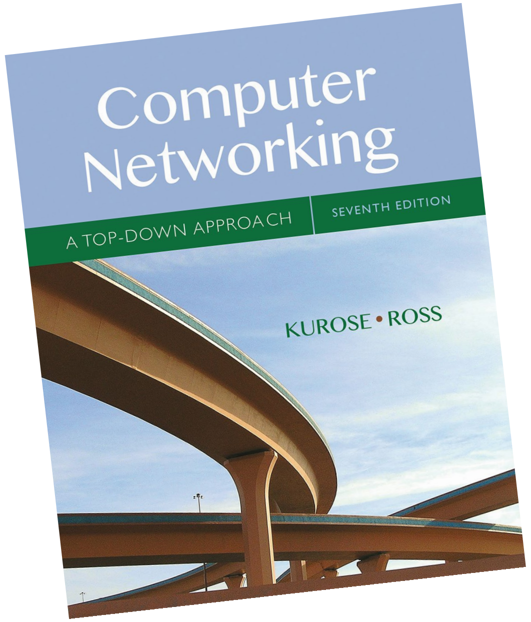
https://engineering.fb.com/data-center-engineering/f16-minipack/ (posted 3/2019)

Datacenter networks: protocol innovations

- link layer:
  - RoCE: remote DMA (RDMA) over Converged Ethernet
- transport layer:
  - ECN (explicit congestion notification) used in transport-layer congestion control (DCTCP, DCQCN)
  - experimentation with hop-by-hop (backpressure) congestion control
- routing, management:
  - SDN widely used within/among organizations' datacenters
  - place related services, data as close as possible (e.g., in same rack or nearby rack) to minimize tier-2, tier-1 communication

Link layer, LANs: roadmap

- introduction
- multiple access protocols
- error detection, correction
- LANs
  - addressing, ARP
  - switches
- data center networking
- a day in the life of a web request

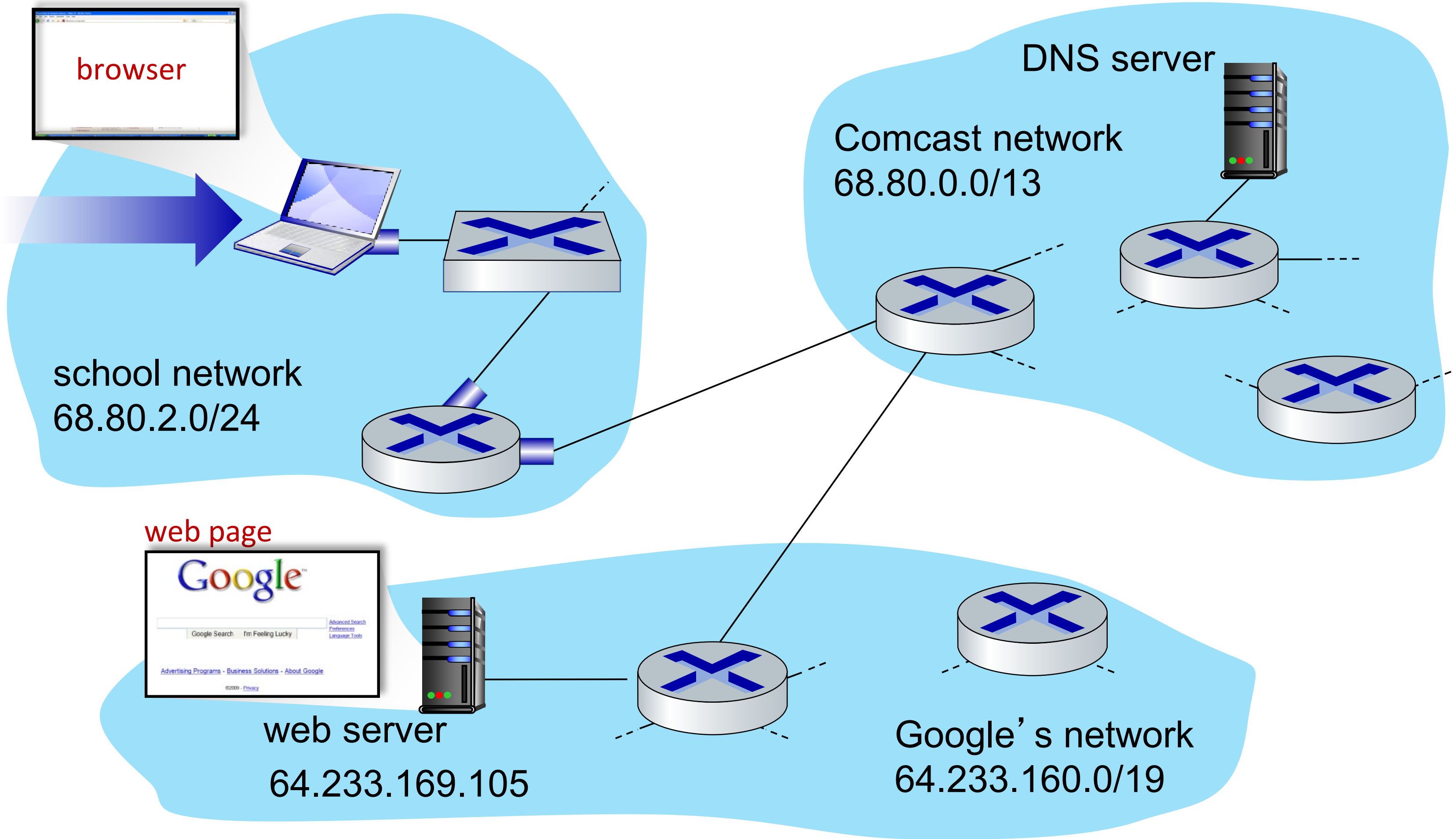


Chapter 6

Synthesis: a day in the life of a web request

- our journey down the protocol stack is now 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

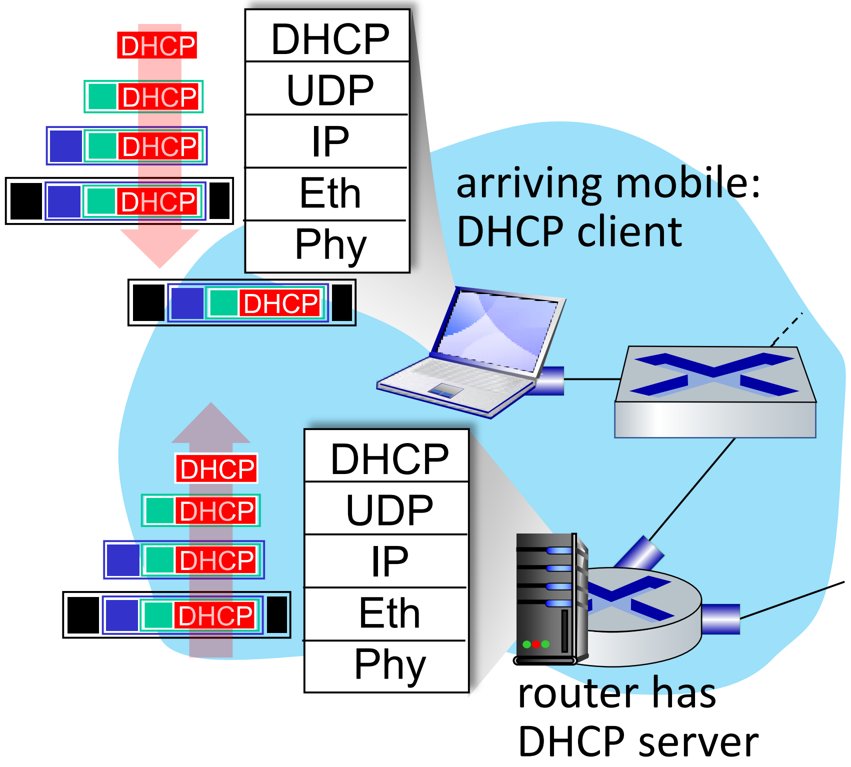


scenario:

- arriving mobile client attaches to network …
- requests web page: www.google.com

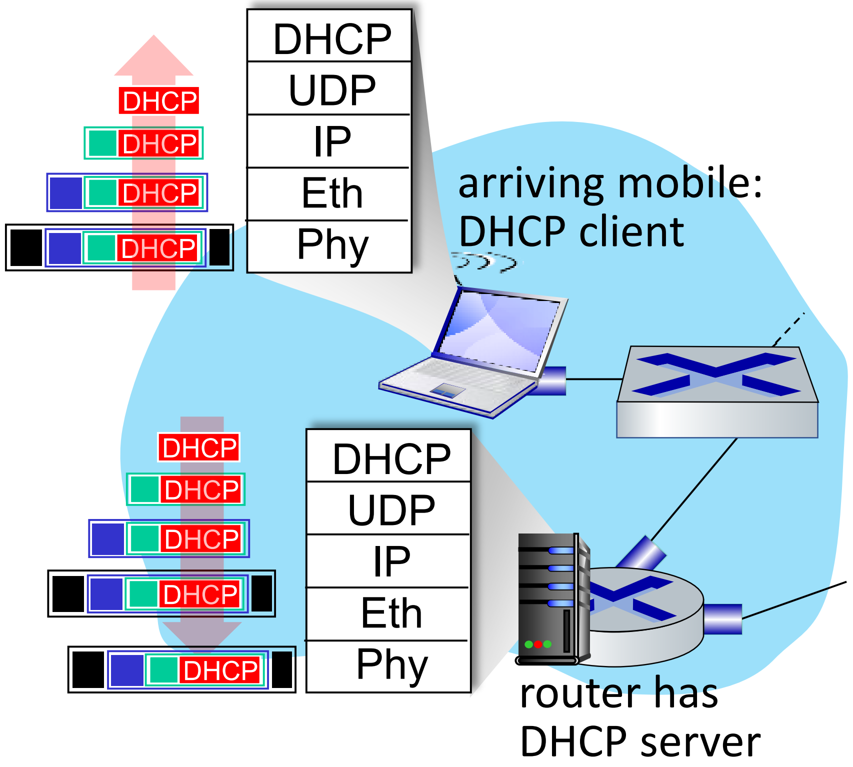
Sounds simple!

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: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

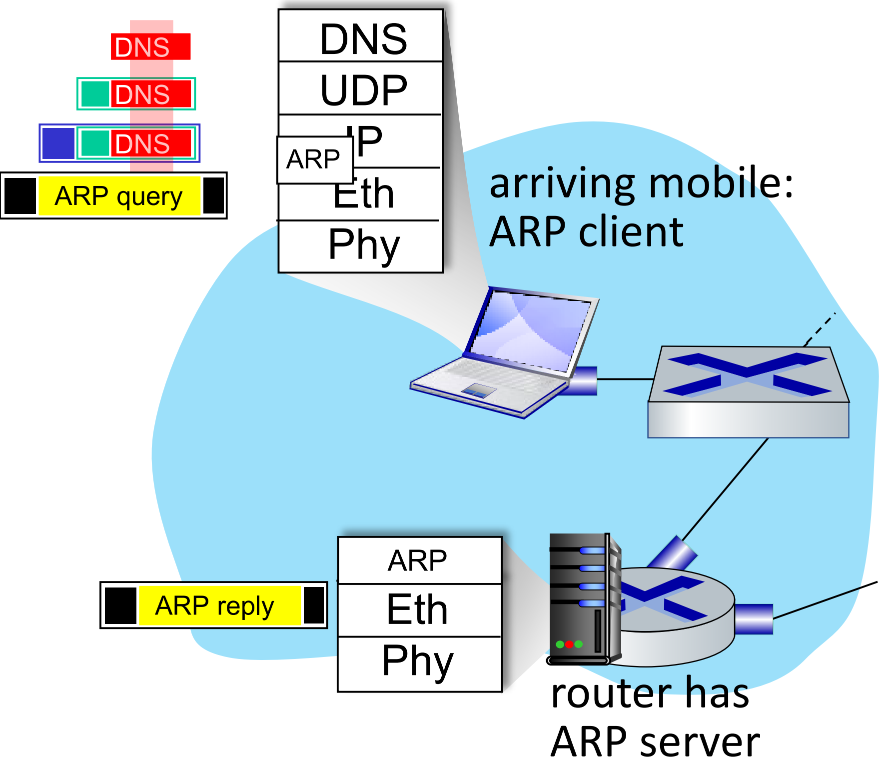
A day in the life: connecting to the Internet



- 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

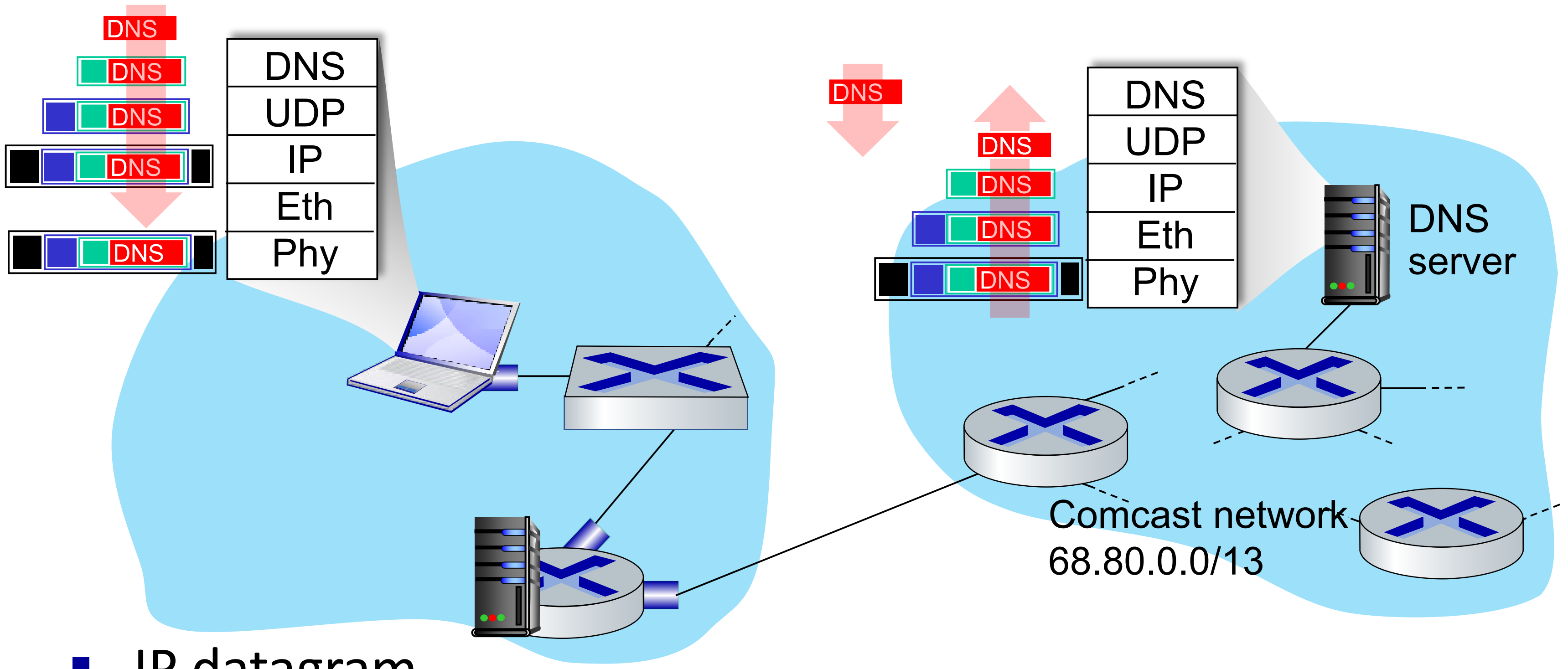
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

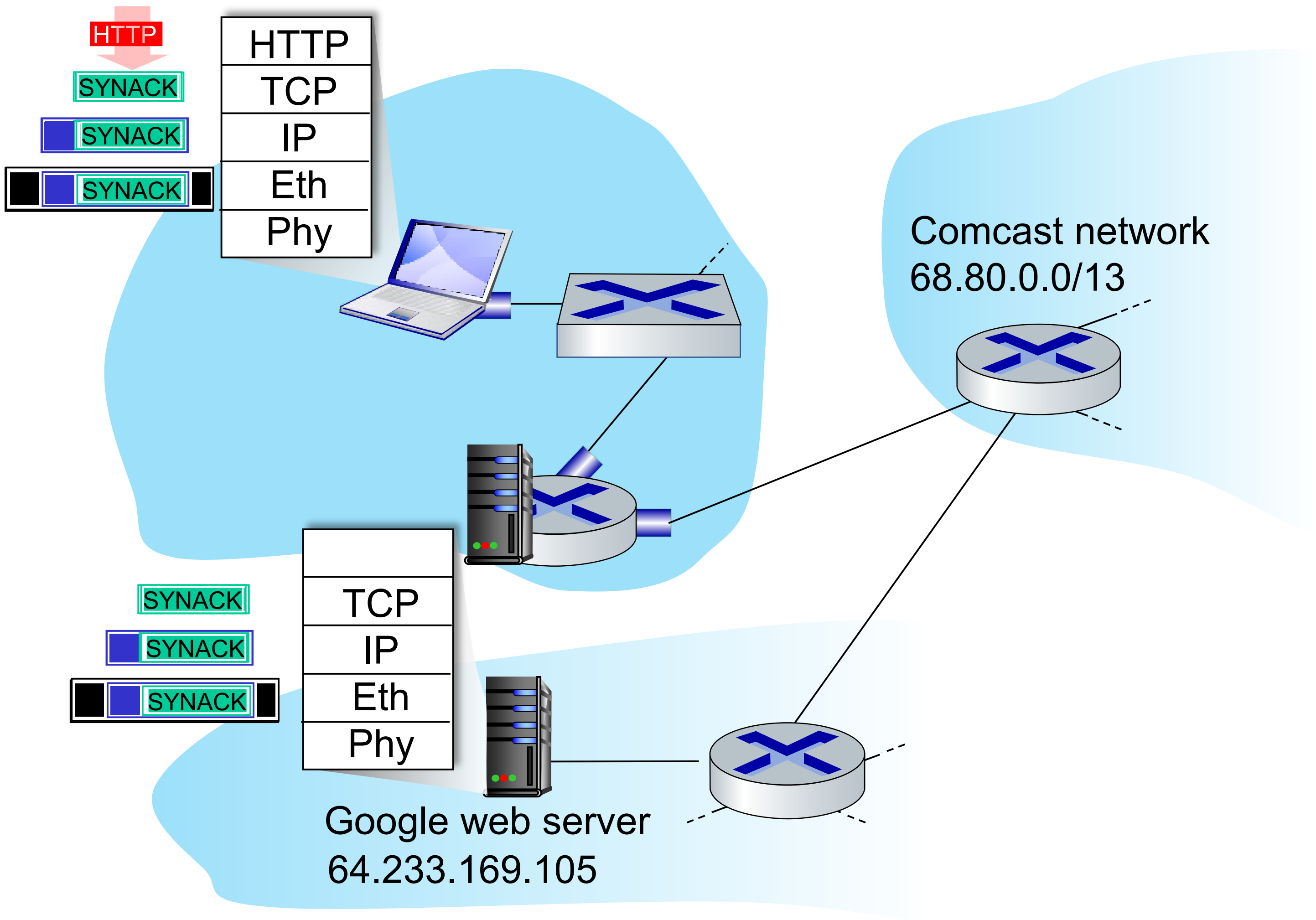


- demuxed to DNS
- DNS replies to client with IP address of www.google.com

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

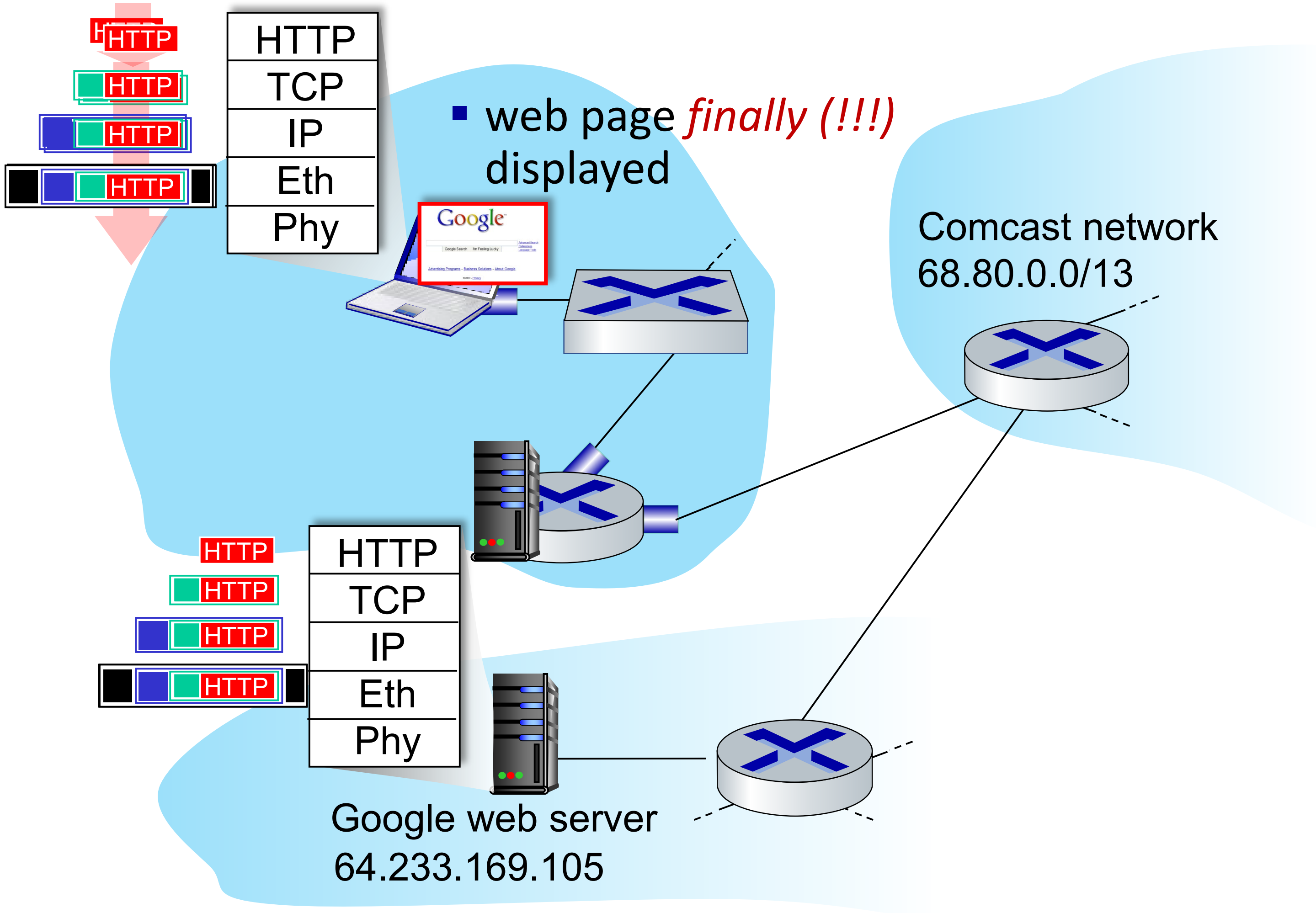
- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

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



- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in TCP 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in TCP 3-way handshake)
- TCP connection established!

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



- HTTP request sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client