# Chapter 6
# The Link Layer and LANs

# Link layer and LANs: our goals

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet
- datacenter networks

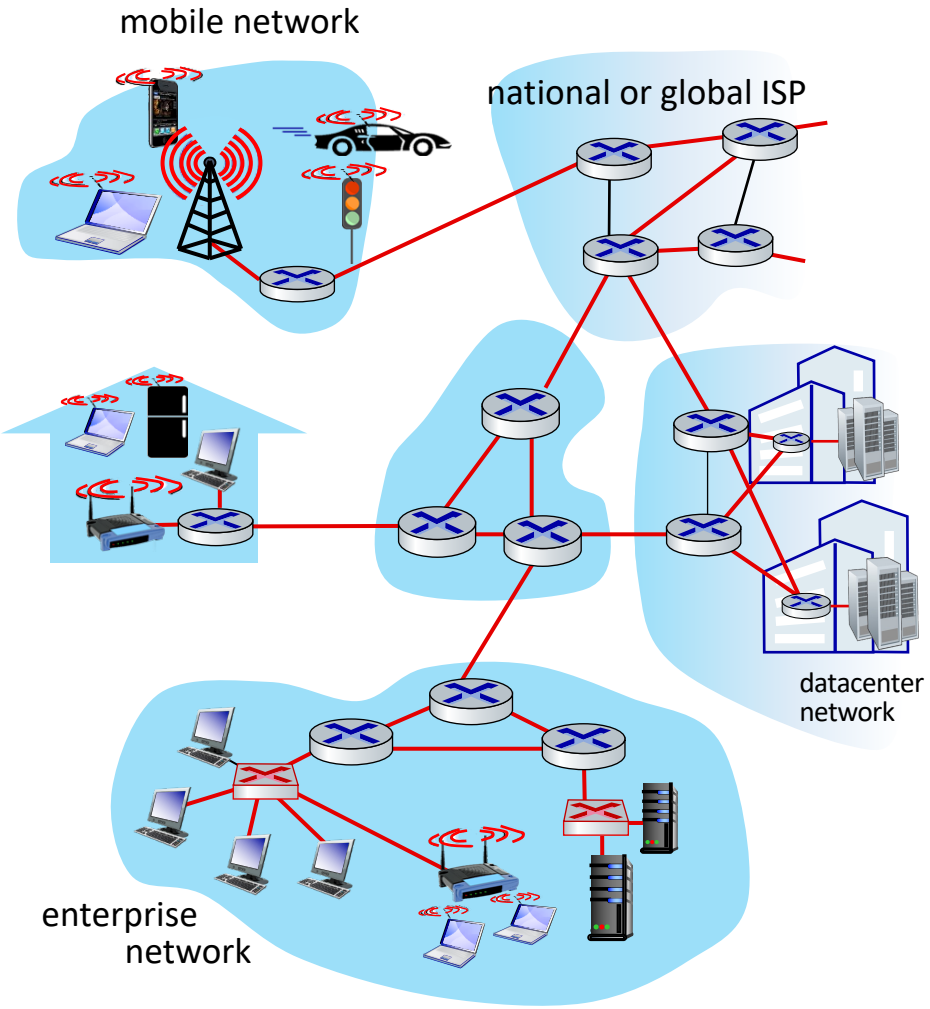
# Link layer: introduction

terminology:

- hosts and routers: *nodes*
- communication channels that connect adjacent nodes along a communication path: *links*
  - wired
  - wireless

- layer-2 packet: *frame*, encapsulates datagram

*link layer has responsibility of transferring datagrams between physically adjacent nodes over a* link



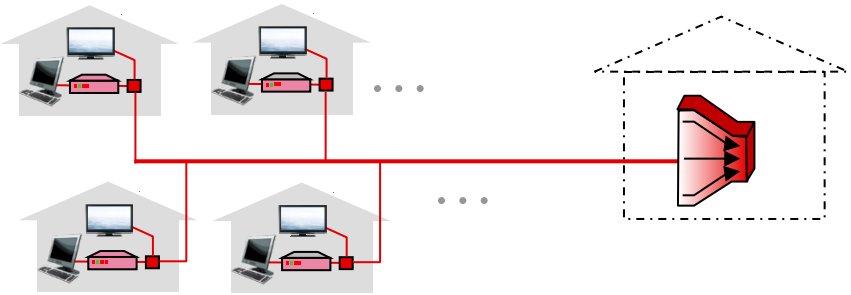
# Link layer: context

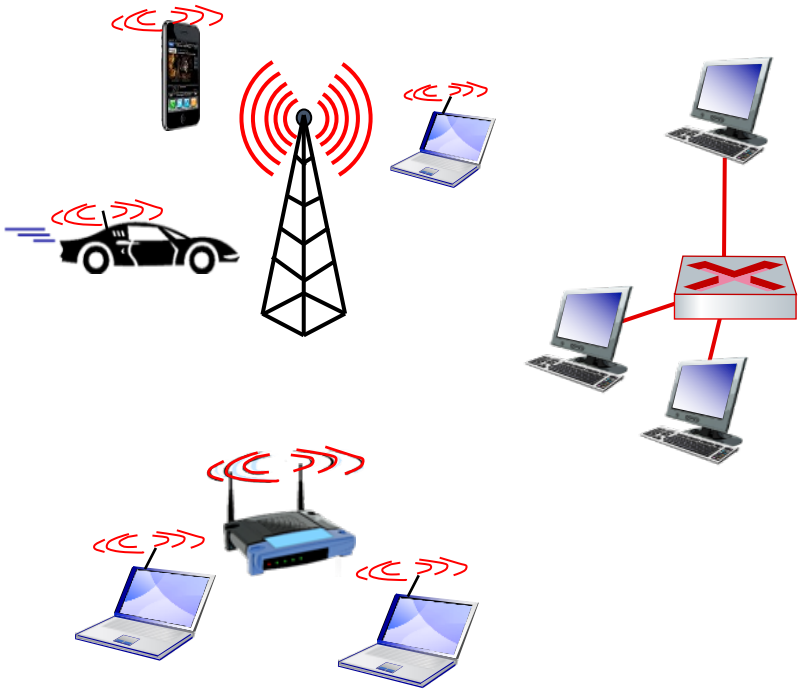
- datagram transferred by different link protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link
- each  link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

# Link layer: services

- framing:
  - encapsulate datagram into frame, adding header, trailer
- link access
  - channel access if shared medium
  - "MAC" addresses in frame headers identify source, destination (different from IP address!)

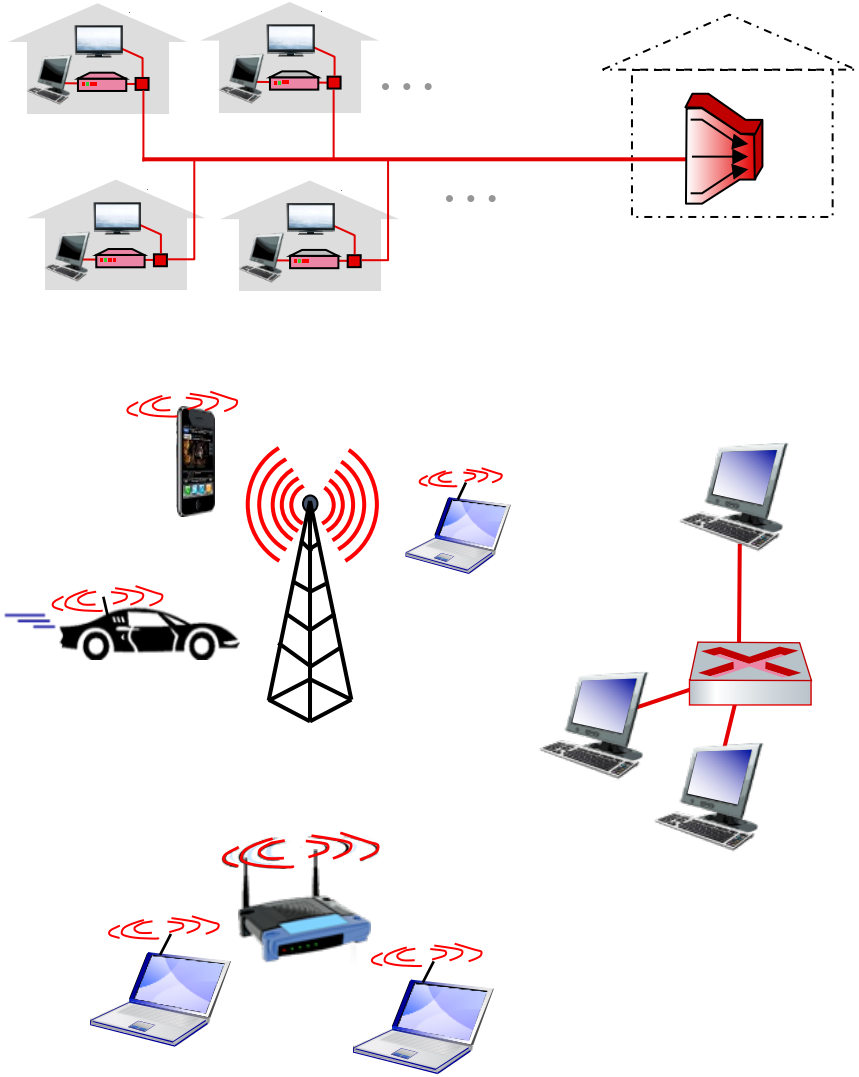
  *MAC= Medium Access Control*
- reliable delivery between adjacent nodes
  - we already know how to do this!
  - seldom used on low bit-error links
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?
    - *A:* Local corrections rather than end-to-end retransmission + more sophisticated hardware implementation





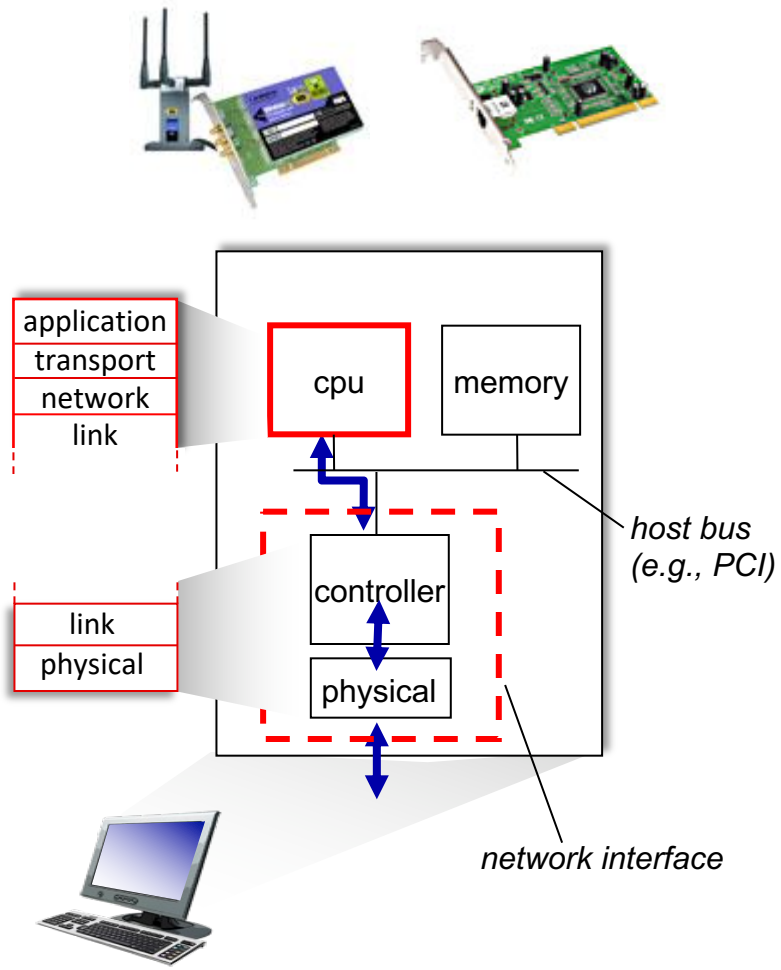
# Link layer: services (more)

- flow control:
  - pacing between adjacent sending and receiving nodes
- error detection:
  - errors are caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- error correction:
  - receiver identifies *and corrects* bit error(s) without retransmission
- half-duplex and full-duplex:
  - with half duplex, nodes at both ends of link can transmit, but not at same time

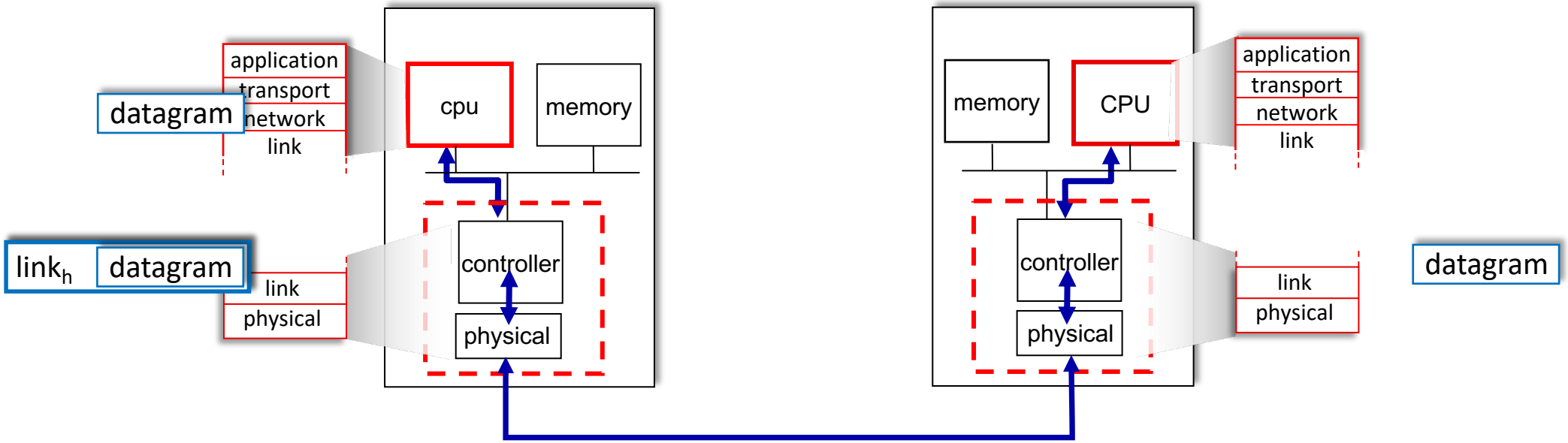


# Where is the link layer implemented?

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



# 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, passes to upper layer at receiving side

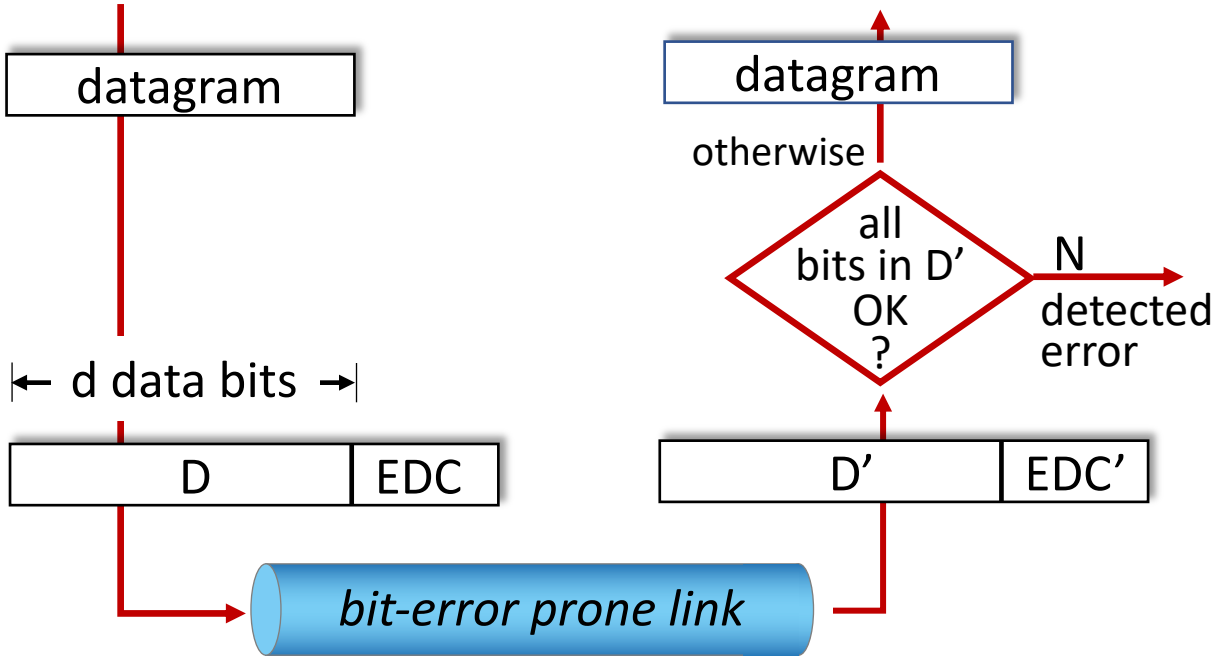
# Link layer, LANs: roadmap

- introduction
- **error detection, correction**
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

# Error detection

EDC: error detection and correction bits (e.g., redundancy)

D: data protected by error checking, may include header fields



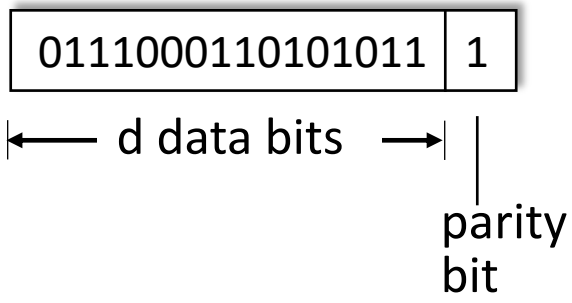

Error detection not 100% reliable!

- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction

# Parity checking

## single bit parity:

- detect single bit errors

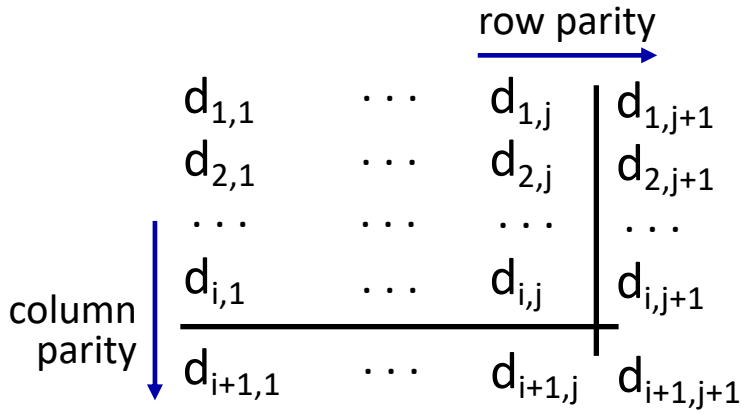



Even parity: set parity bit so there is an even number of 1's

*It actually detects any odd-number bit error in the frame*
*What if even number of errors?*

## two-dimensional bit parity:

- detect *and correct* single bit errors




Under which error scenario does this method fail?

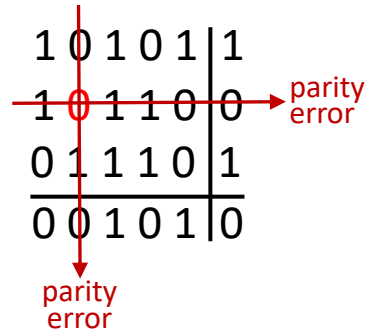
Can it detect multiple simultaneous errors?

Can it correct them?

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 |

detected and correctable single-bit 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 checksum field

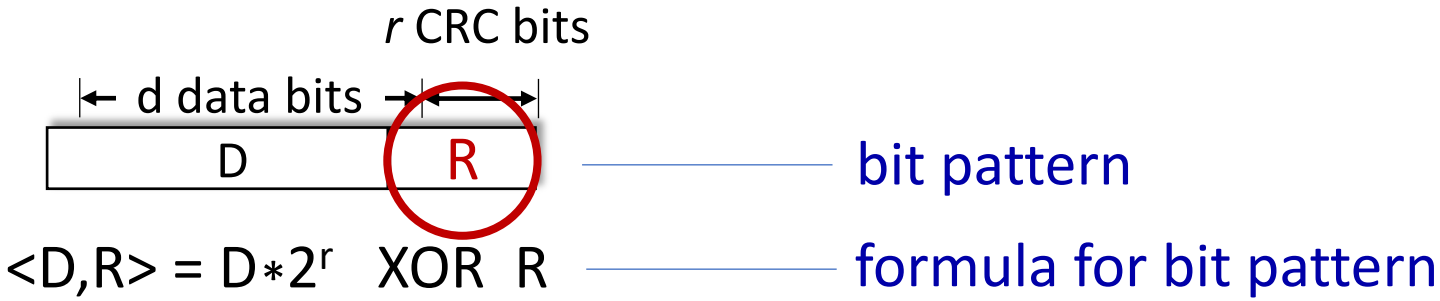
## receiver:

- 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)

  *sender and receiver must agree on the bit pattern G*




*goal:* sender chooses *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 whose length is less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi)

# …recall that

Modulo 2 arithmetic – no carries in addition, nor borrows in subtraction

1011 - 0101 = 1011 + 0101 = 1110 = 1011 XOR 0101

1001 - 1101 = 0100 = 1001 XOR 1101

Addition and subtraction are identical

# Cyclic Redundancy Check (CRC): example

We want:

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

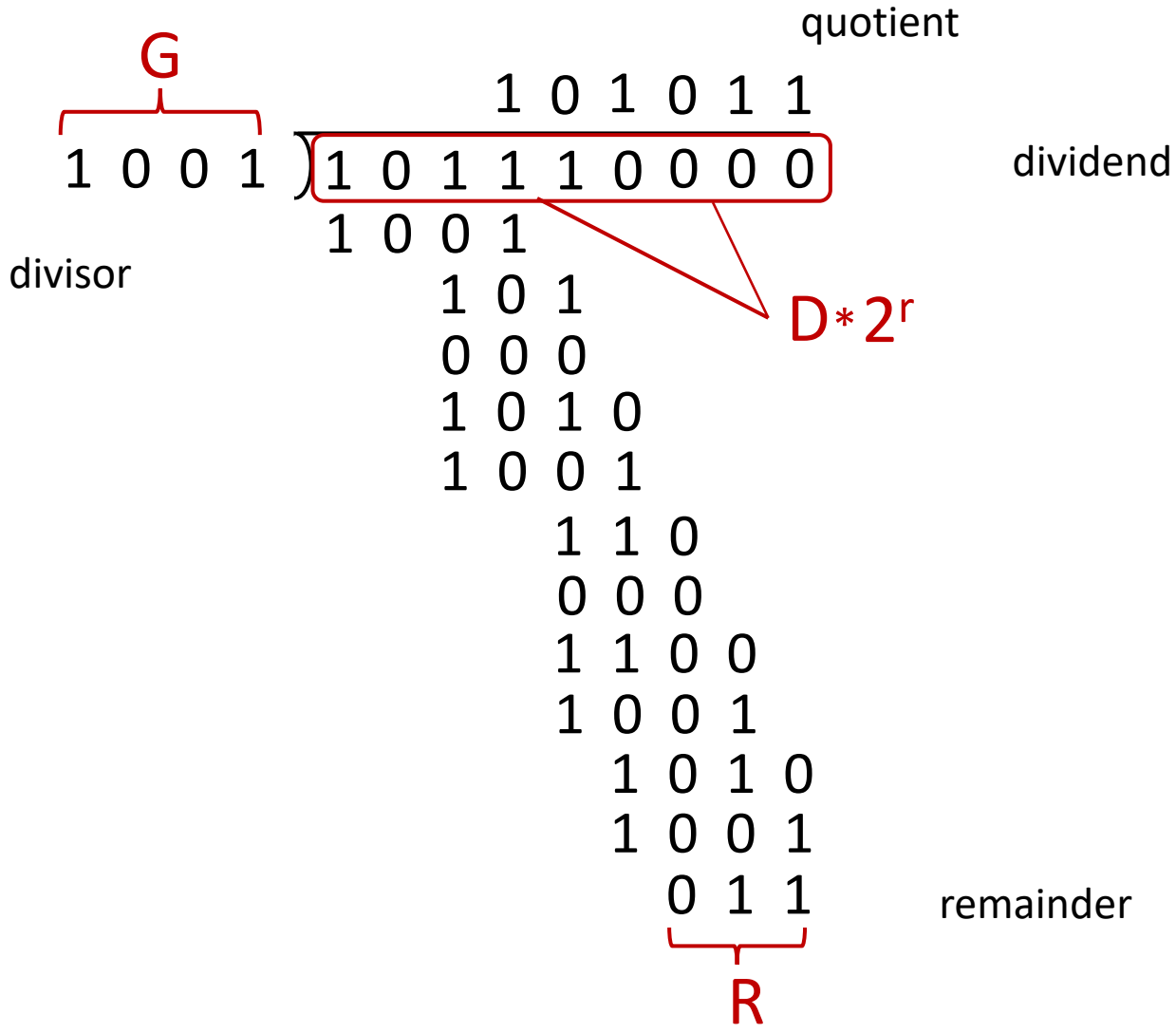
or equivalently (xor R both sides):

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

or equivalently:

if we divide $D \cdot 2^r$ by G, we want the remainder to be R (*which is why we want G to start with a 1*):

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



# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

- a day in the life of a web request

# Multiple access links, protocols

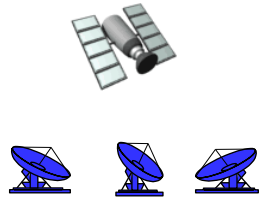
two types of “links”:

- point-to-point
  - point-to-point link between an Ethernet switch and a host
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/4G. 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**

- 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

# An ideal multiple access protocol

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

*ideally:*

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

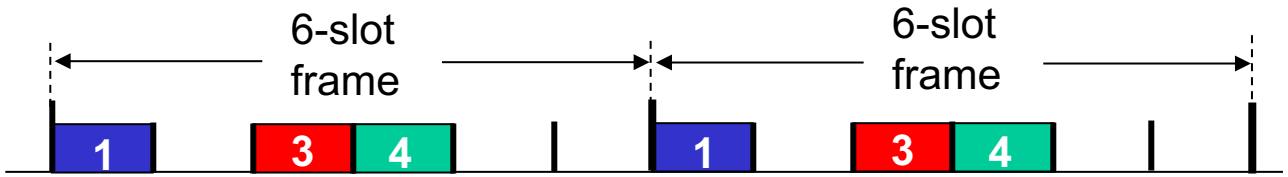
three broad classes:

- 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

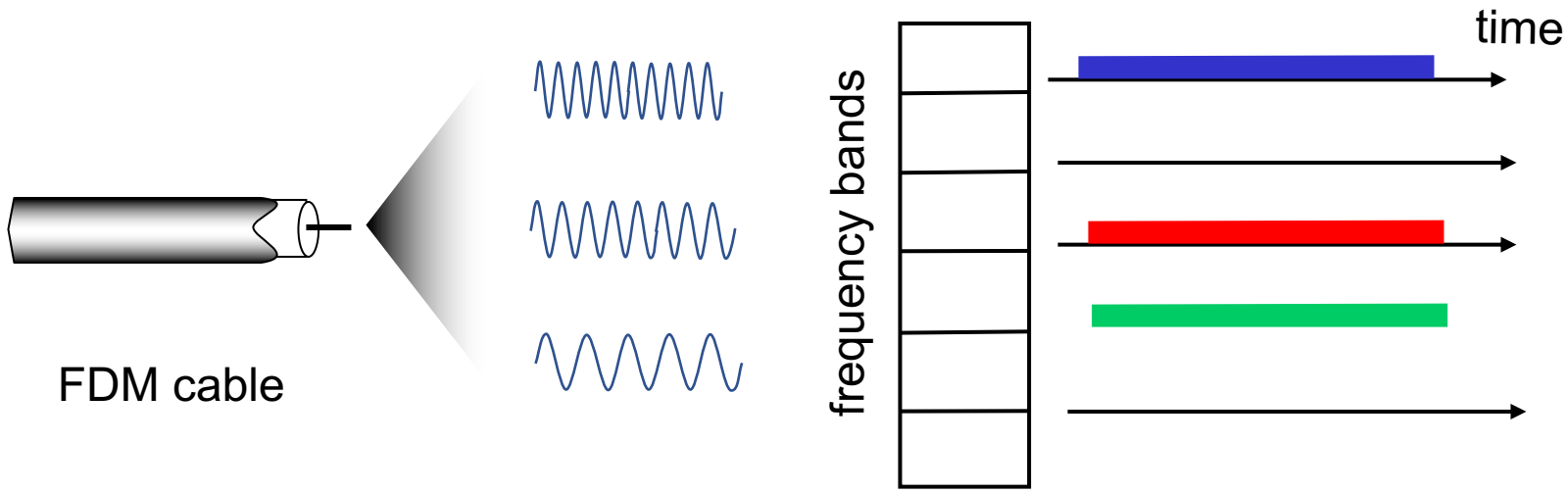
- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# Channel partitioning MAC protocols: FDMA

## FDMA: frequency division multiple access

- channel spectrum 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 packet to send, 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
- If there are 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:
  - ALOHA, slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

# Slotted ALOHA

## assumptions:

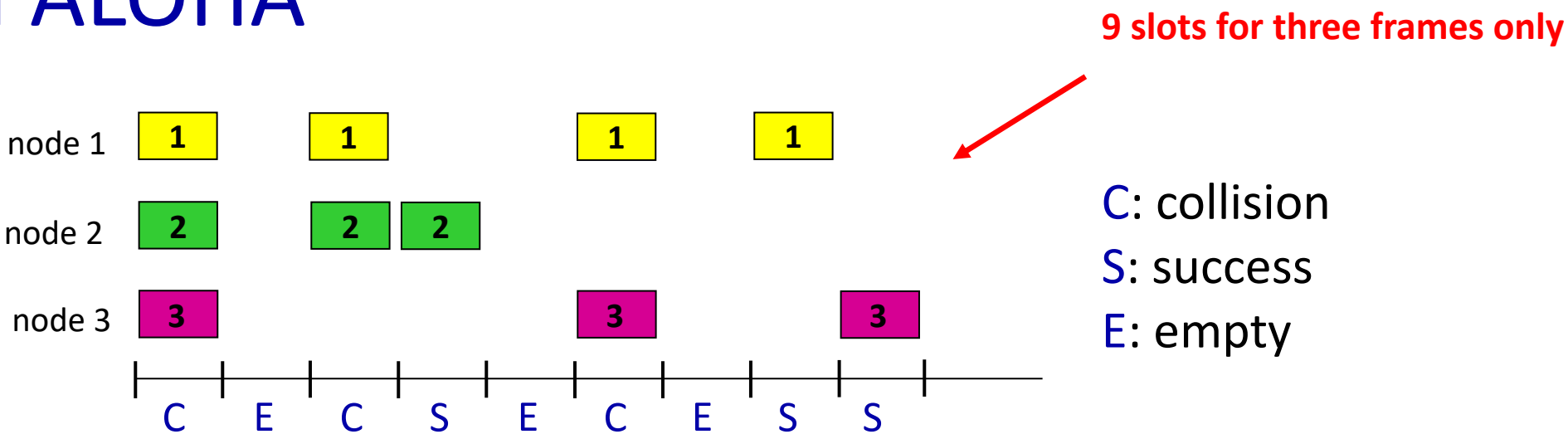
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at the beginning of a slot
- nodes are synchronized
- if 2 or more nodes transmit in the same slot, all nodes detect collision before the end of the slot

## 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

randomization – *why*?

# Slotted ALOHA




## Pros:

- If there is a single active node, it can continuously transmit at full channel rate
- 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

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

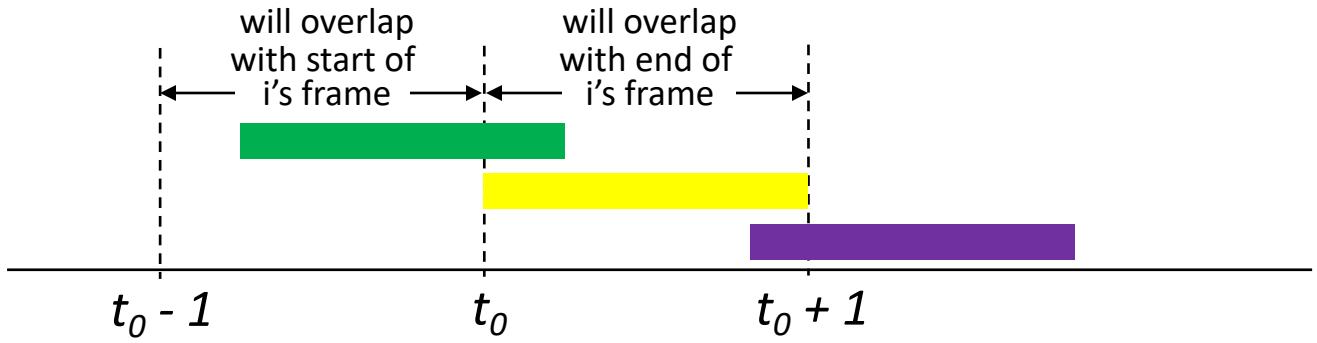
- *suppose: N* nodes with many frames to send, each transmits in slot with probability *p*
  - prob that given node has success in a slot = $p(1-p)^{N-1}$
  - prob that *any* node has a success = $Np(1-p)^{N-1}$
  - max efficiency: find $p^*$ that maximizes $Np(1-p)^{N-1}$
  - for many nodes, take limit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

  *max efficiency = 1/e = .37*

- *at best:* channel used for useful transmissions 37% of time!

# Pure ALOHA

- unslotted Aloha: simpler, no synchronization
  - when frame first arrives: transmit 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

P(success by given node) = P(node transmits) *

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

P(no other node transmits in $[t_0,t_0+1]$

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

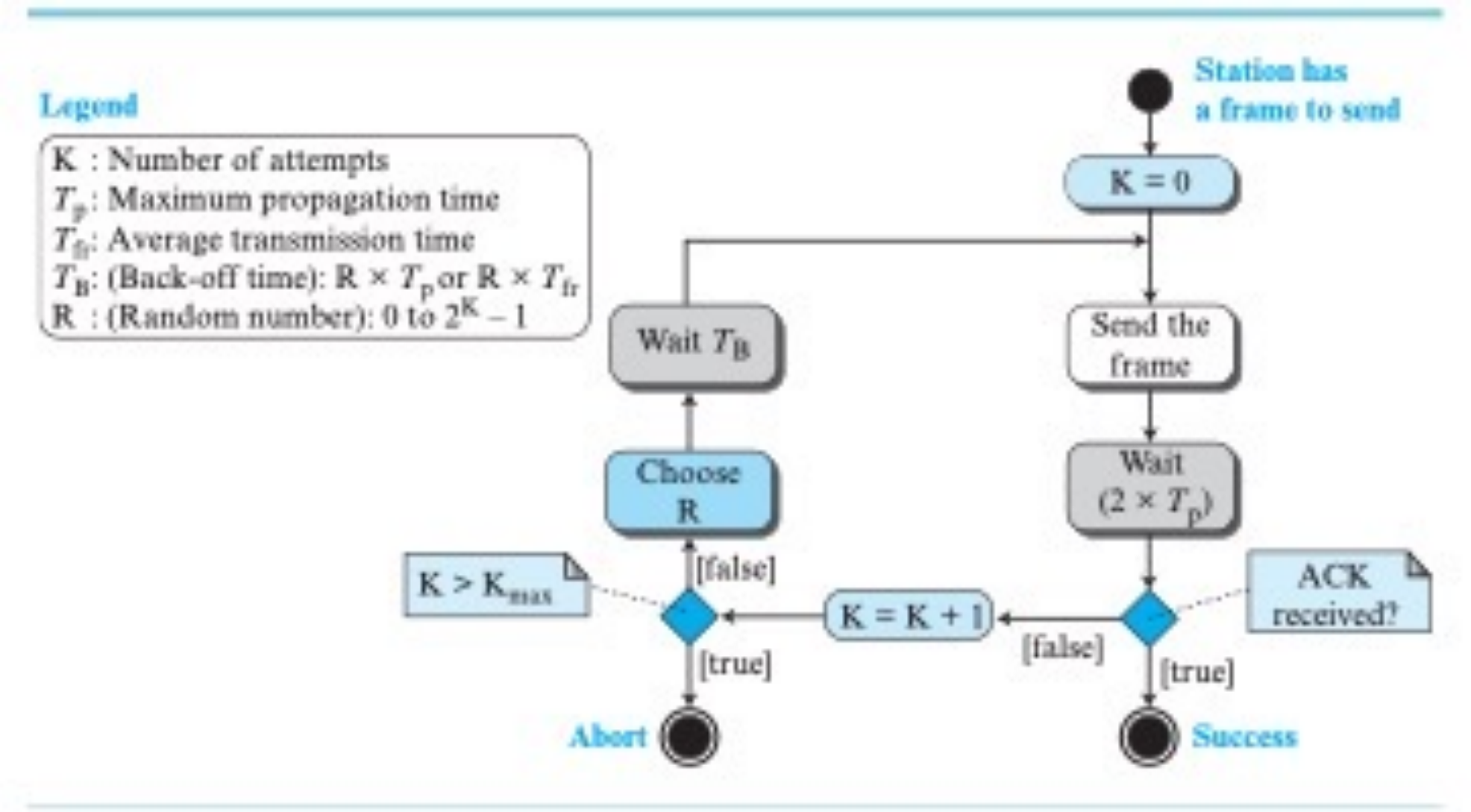
$= p \cdot (1\text{-}p)^{2(N-1)}$

… choosing optimum p and then letting *n*

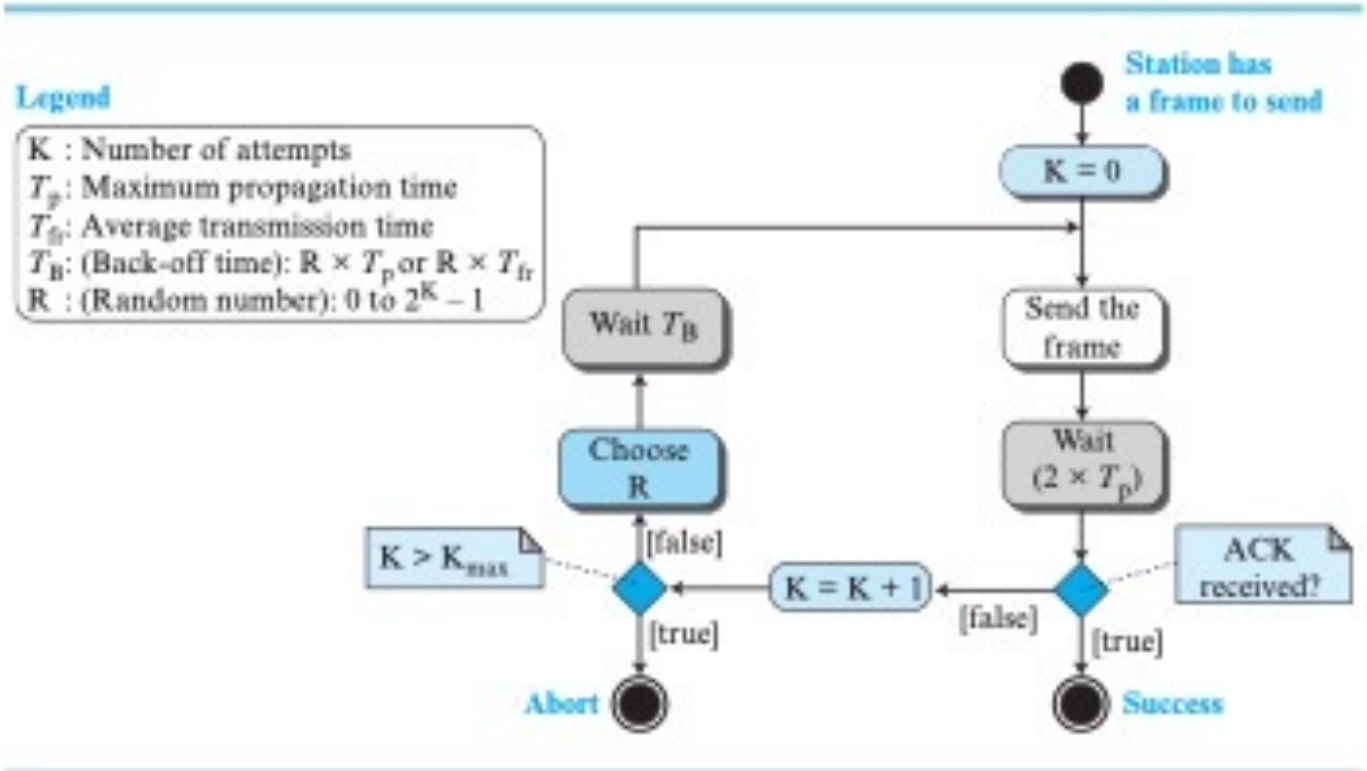
$= 1/(2e) = .18 \rightarrow \infty$

even worse than slotted Aloha!

# Pure ALOHA protocol with backoff

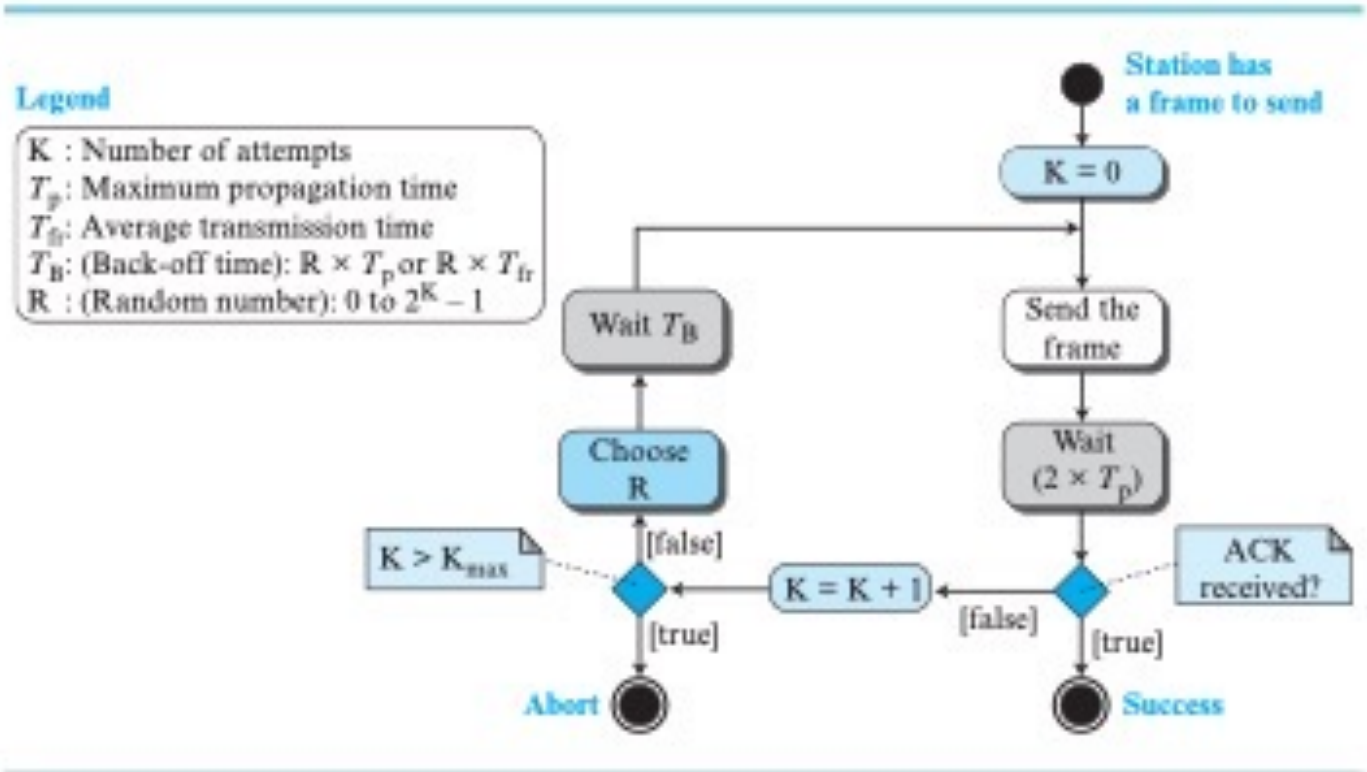


# Pure ALOHA protocol




Consider two stations at a distance of 600 km.
Consider a propagation speed of $3x10^8$ m/s.
Consider K=2, how long is the backoff time $T_B$ ?

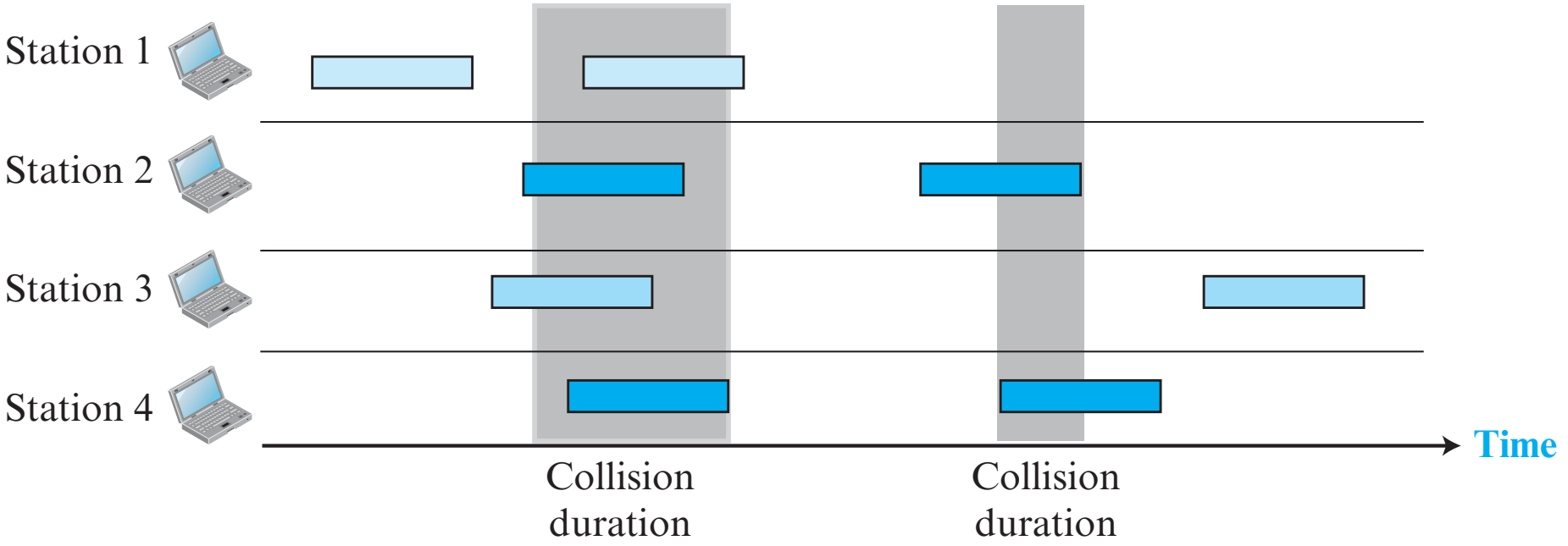
# Pure ALOHA protocol




Consider two stations at a distance of 600 km.
Consider a propagation speed of $3x10^8$ m/s.
Consider K=2, how long is the backoff time $T_B$ ?

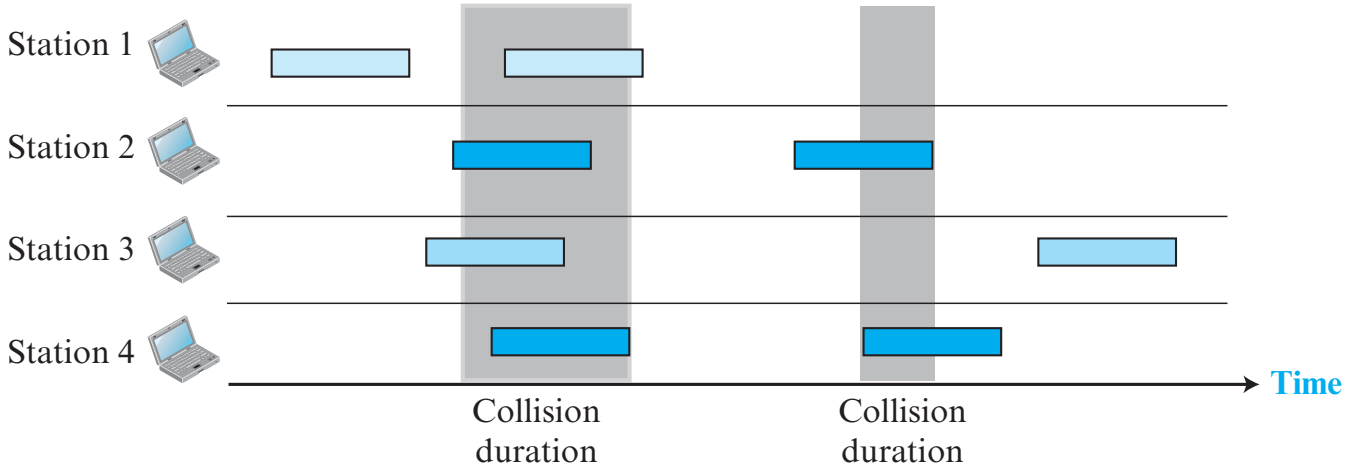
We must find $T_P=(600 \times 10^3)/(3 \times 10^8) = 2ms$.
If K=2, R={0, 1, 2, 3}
$T_B$ can be 0, 2, 4, or 6 ms.

# Collision duration of pure ALOHA frames

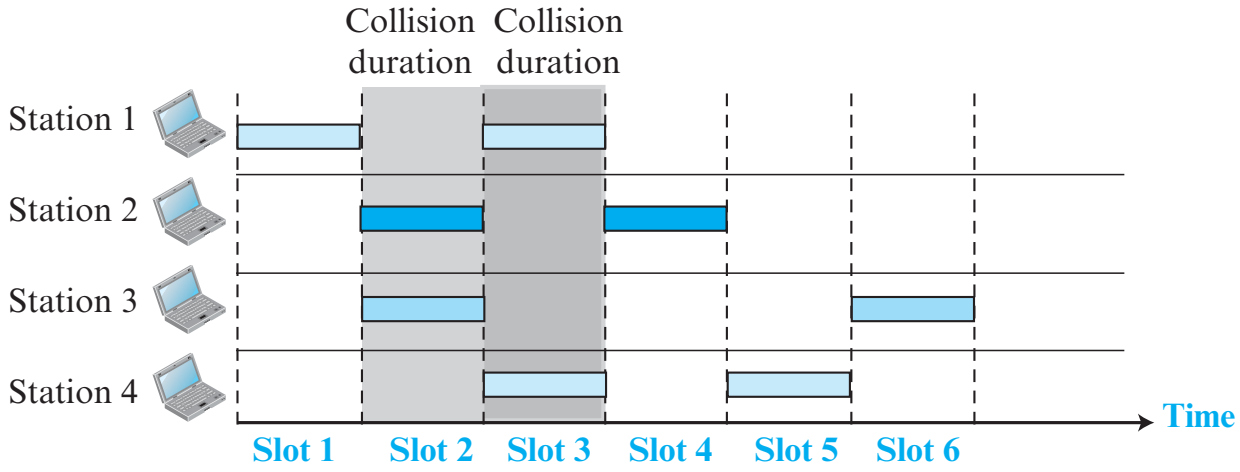



Vulnerability time: time from start to end of transmission of two colliding frames

# Comparison of vulnerability times in pure- and slotted-ALOHA

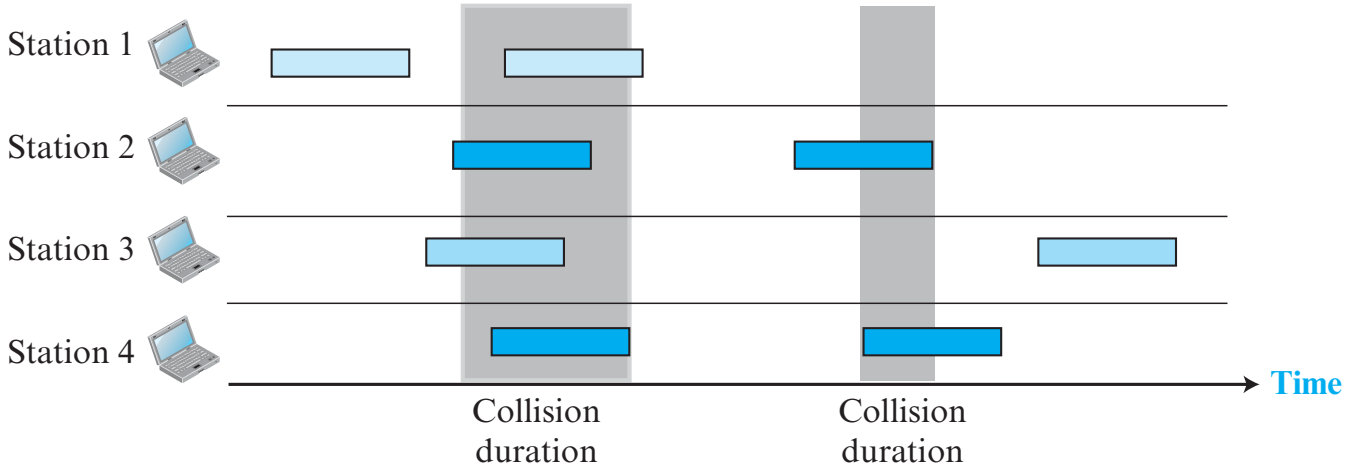



Pure ALOHA



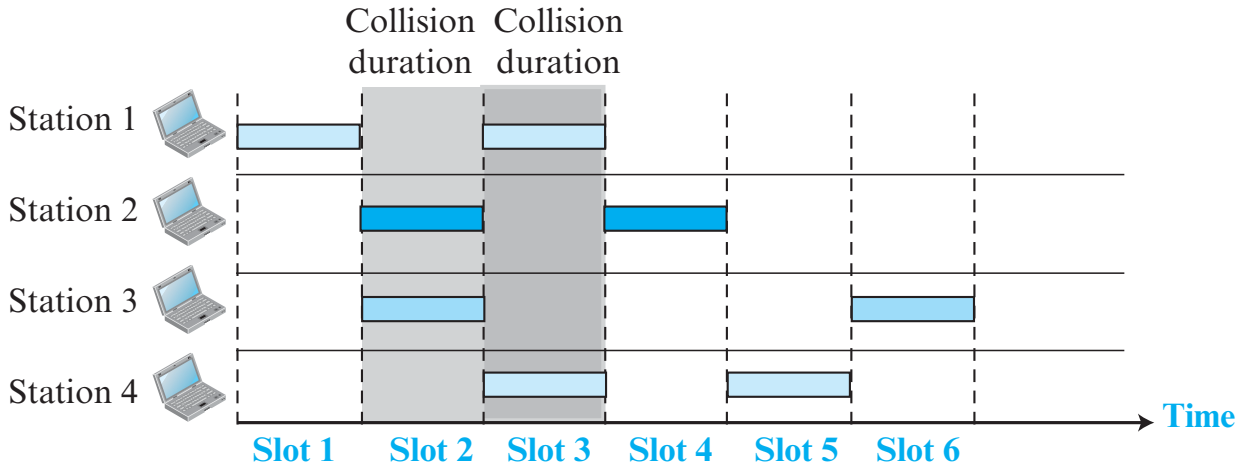

Slotted ALOHA

# Comparison of vulnerability times in pure- and slotted-ALOHA




Pure ALOHA

Vulnerability time: $2xT_{fr}$




Slotted ALOHA

Vulnerability time: $T_{fr}$

# CSMA (carrier sense multiple access)

simple CSMA: listen before transmit:

- if channel sensed idle: transmit entire frame
- if channel sensed busy: defer transmission

- human analogy: don't interrupt others! Listen before you speak.

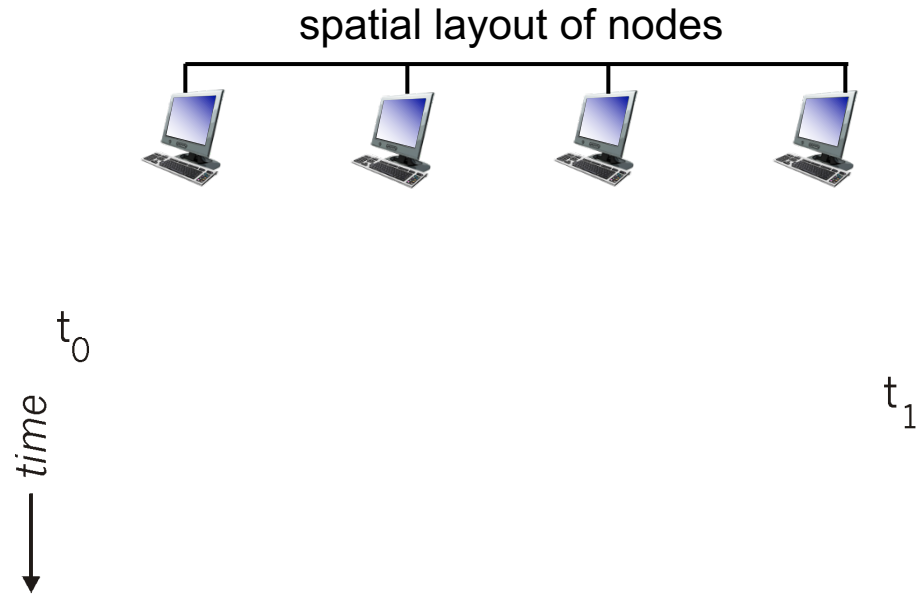
CSMA/CD: CSMA with *collision detection*

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection easy in wired, difficult with wireless

- human analogy: the polite conversation list

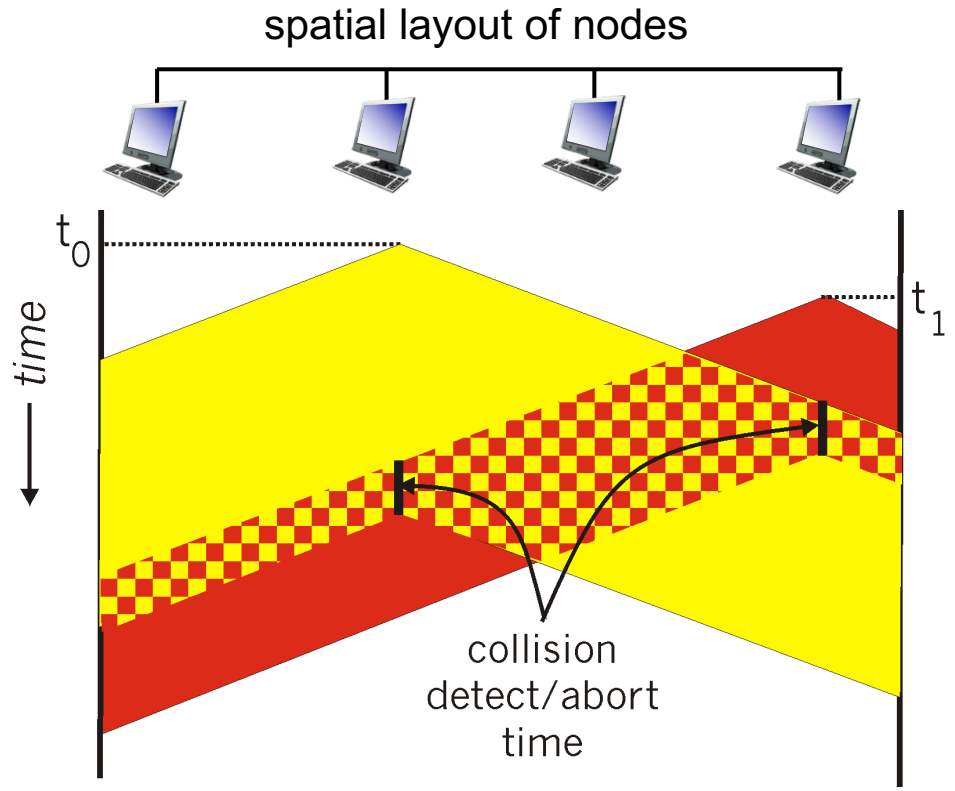
# CSMA: collisions

- 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 in determining collision probability

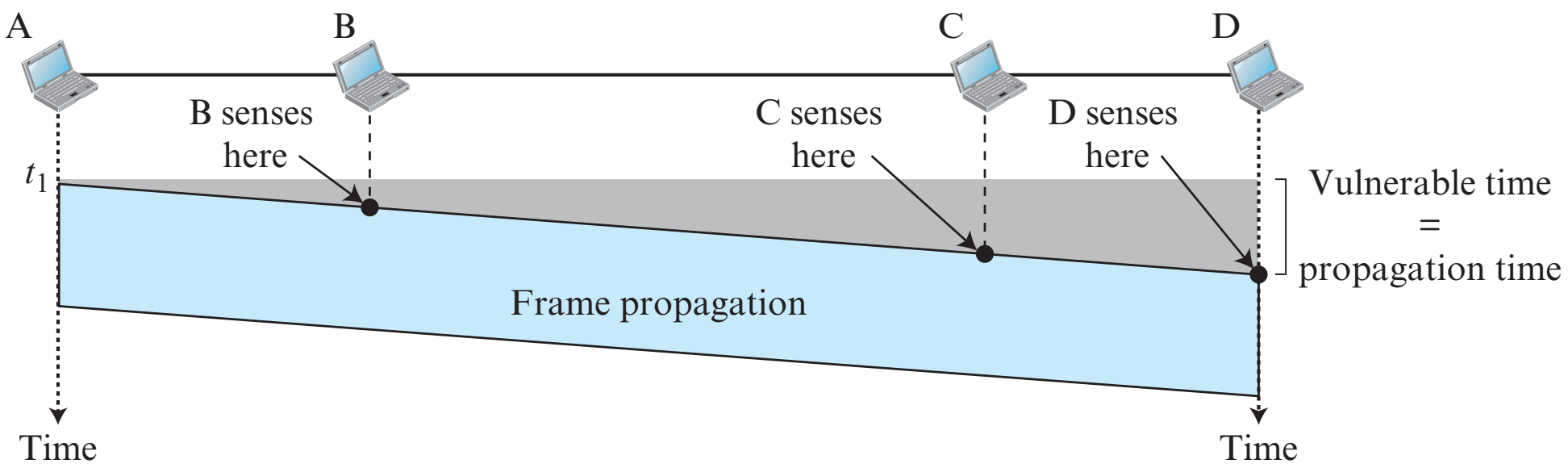


# CSMA/CD:

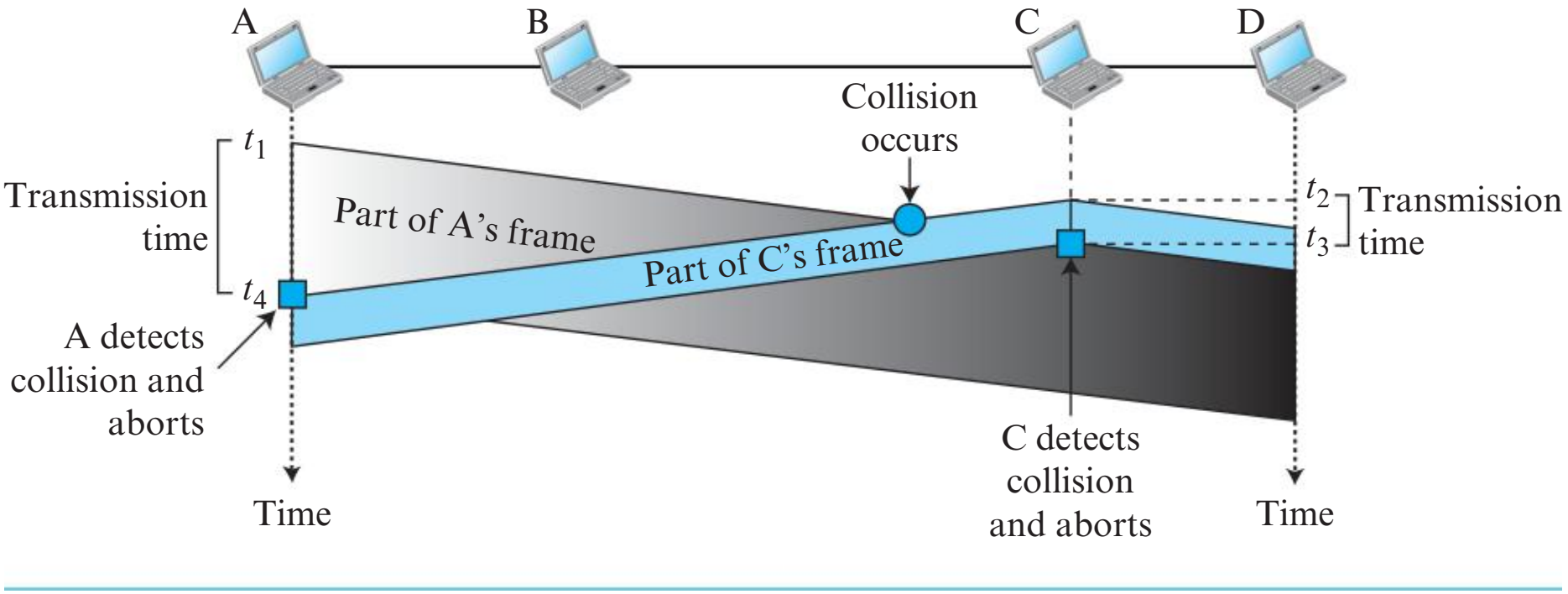
- CSMA/CD reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



# Vulnerability time in CSMA/CD

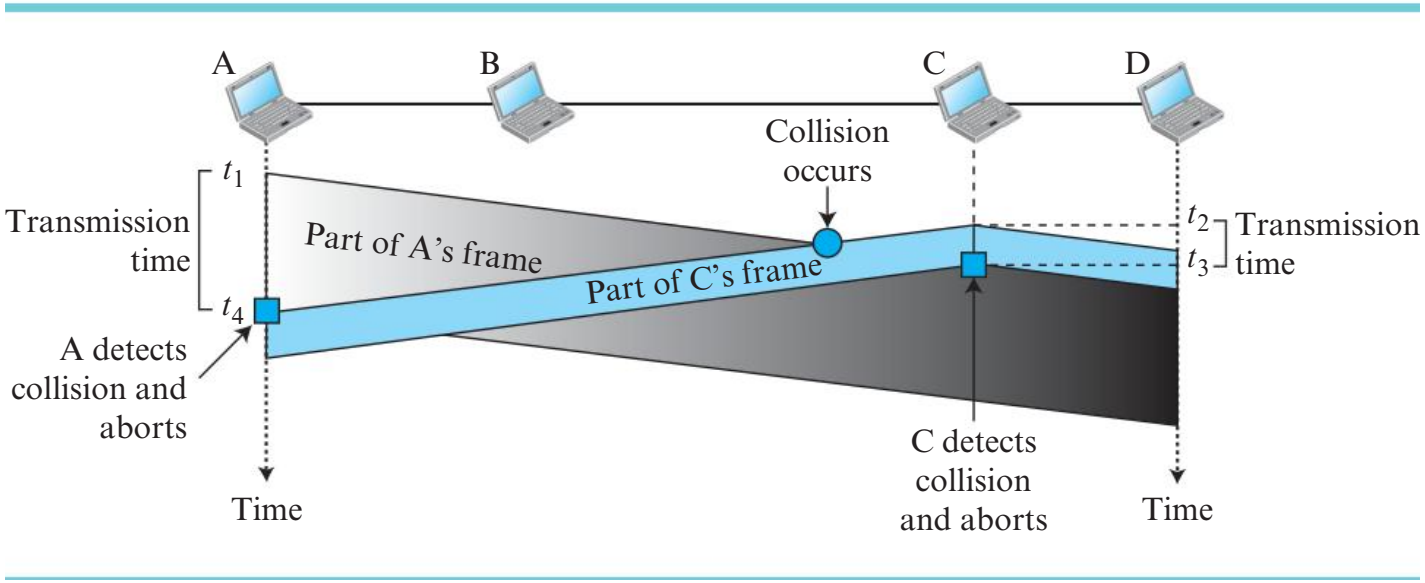


# Collision and abortion in CSMA/CD




Upon collision detection, a station sends a *jamming signal*

# Minimum frame size in CSMA/CD



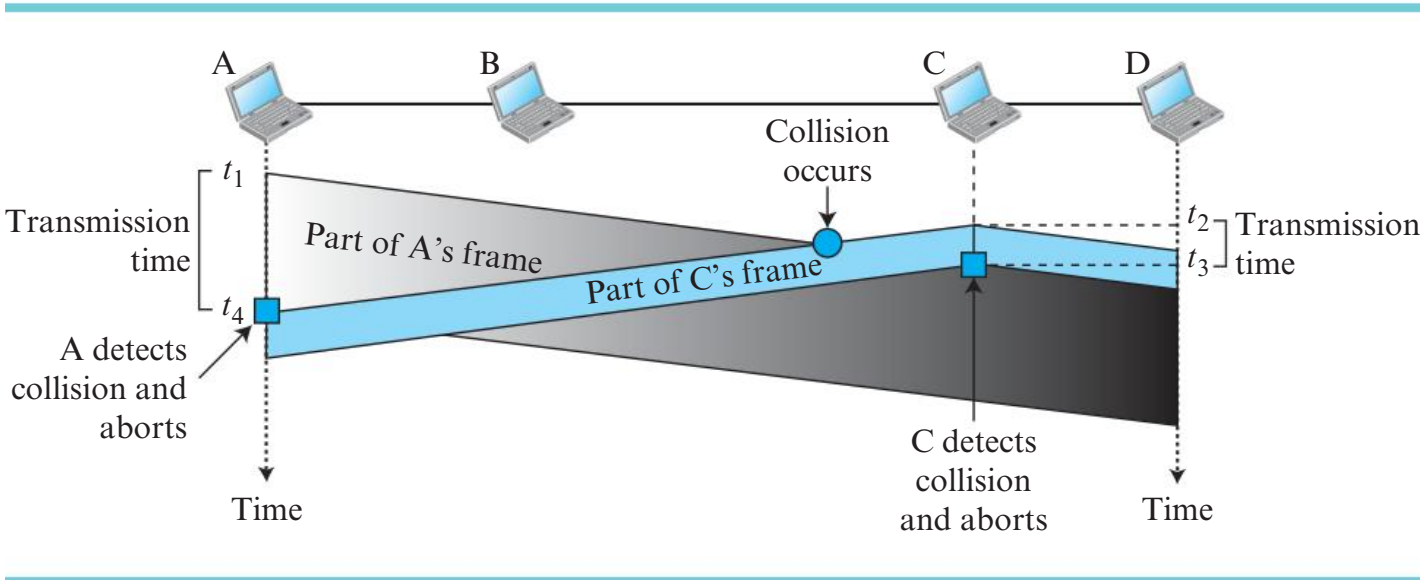

We note that

**if a station does NOT detect a collision within the frame transmission time:**

1) **It does not store a copy of the frames it sent**
2) **It does not check for existing collisions**

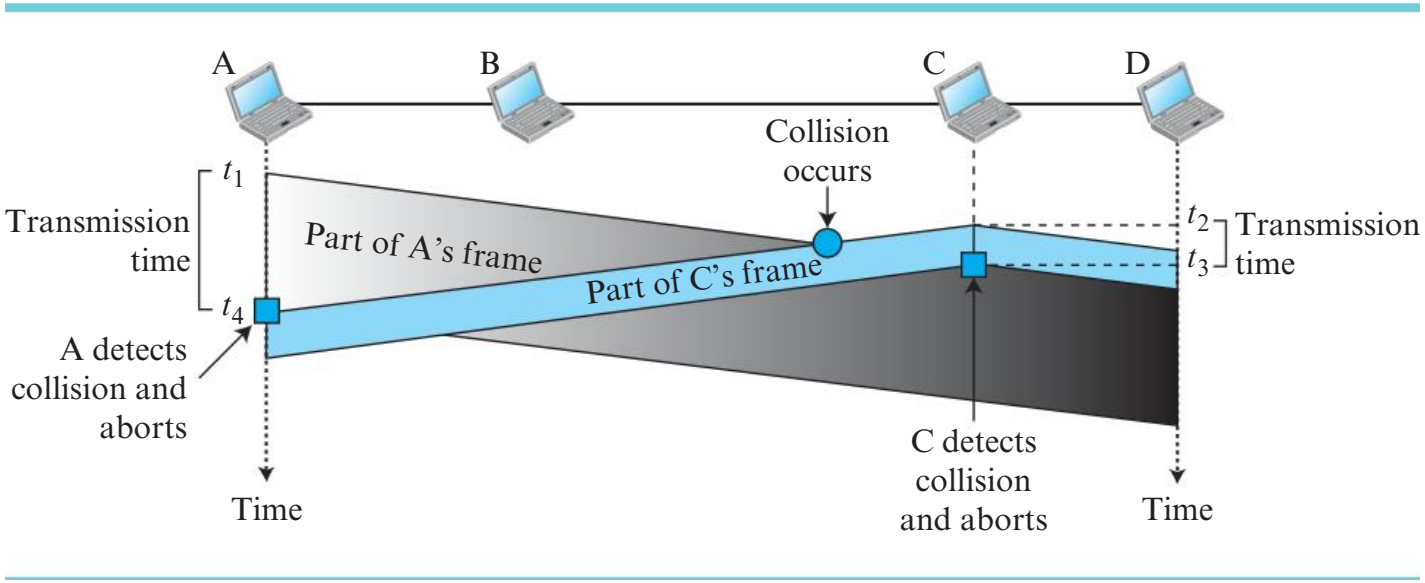
hence the size of the frame must be long enough to allow collision detection at any sender station (worst case)

# Minimum frame size in CSMA/CD




How long is the minimum frame size?

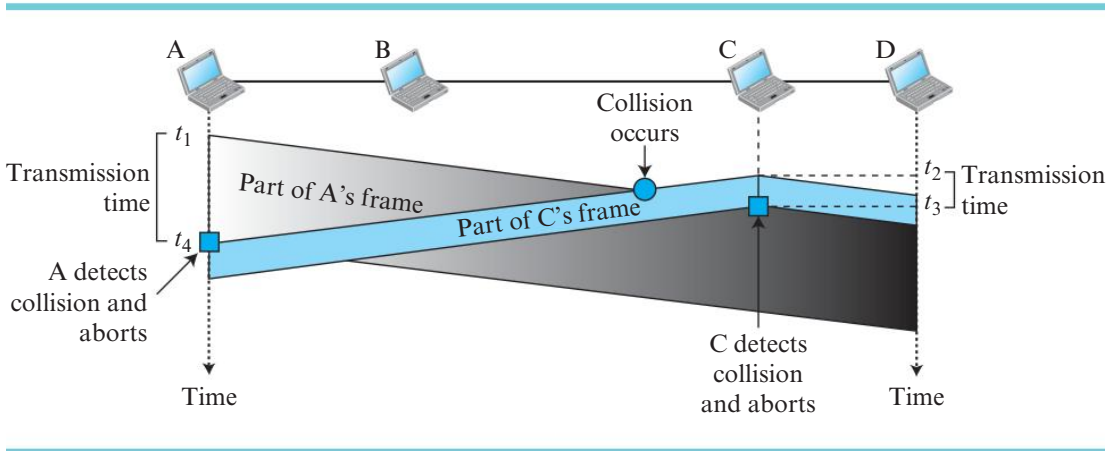
# Minimum frame size in CSMA/CD




How long is the minimum frame size?

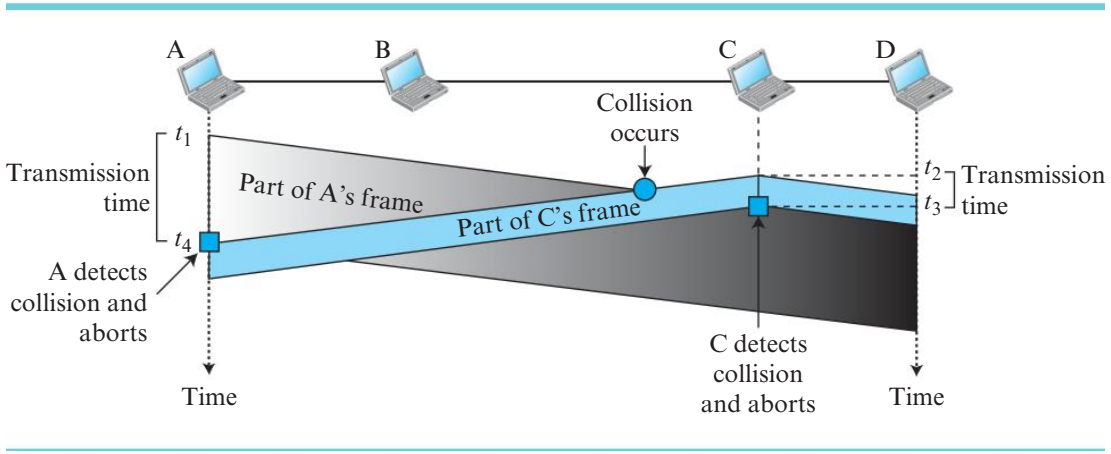
It is the amount of bits you can transmit in twice the propagation time $2 \times T_p$.

# Exercize on min frame size in CSMA/CD




A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices) is 25.6 μs, what is the minimum size of the frame?

# Exercize on min frame size in CSMA/CD




A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices) is 25.6 μs, what is the minimum size of the frame?

The minimum frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet.

# Ethernet CSMA/CD algorithm

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

# “Taking turns” MAC protocols

## 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

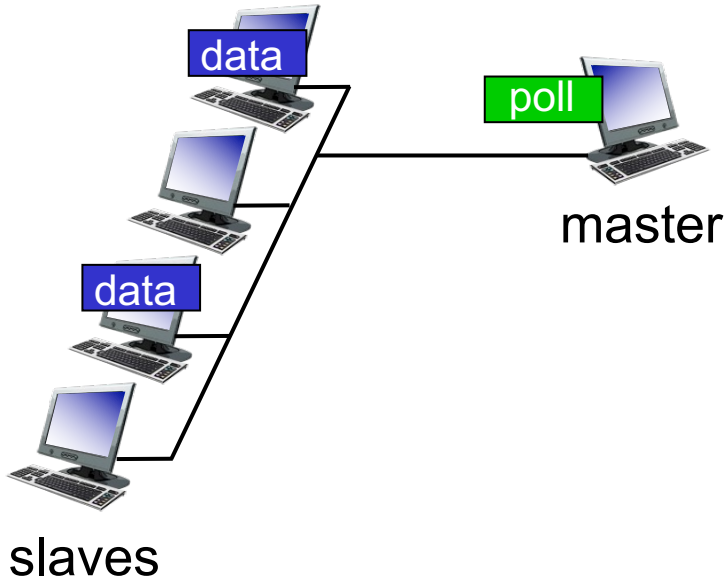
## “taking turns” protocols

- look for best of both worlds!

# “Taking turns” MAC protocols

## polling:

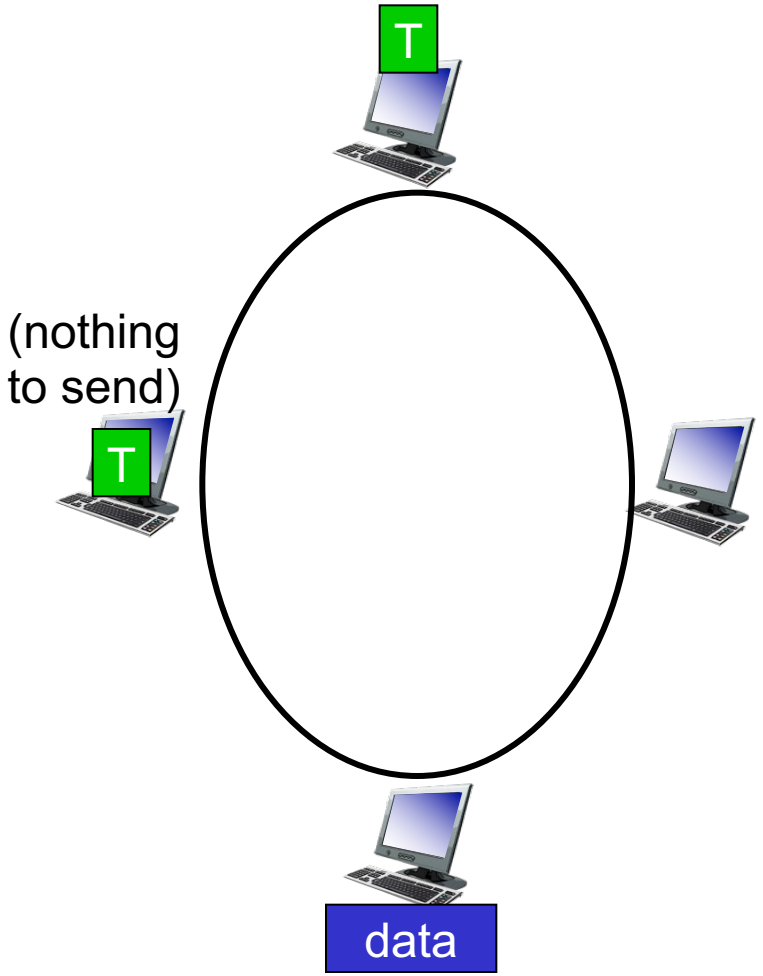
- master node “invites” other nodes to transmit in turn
- typically used with “dumb” devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



# “Taking turns” MAC protocols

token passing:

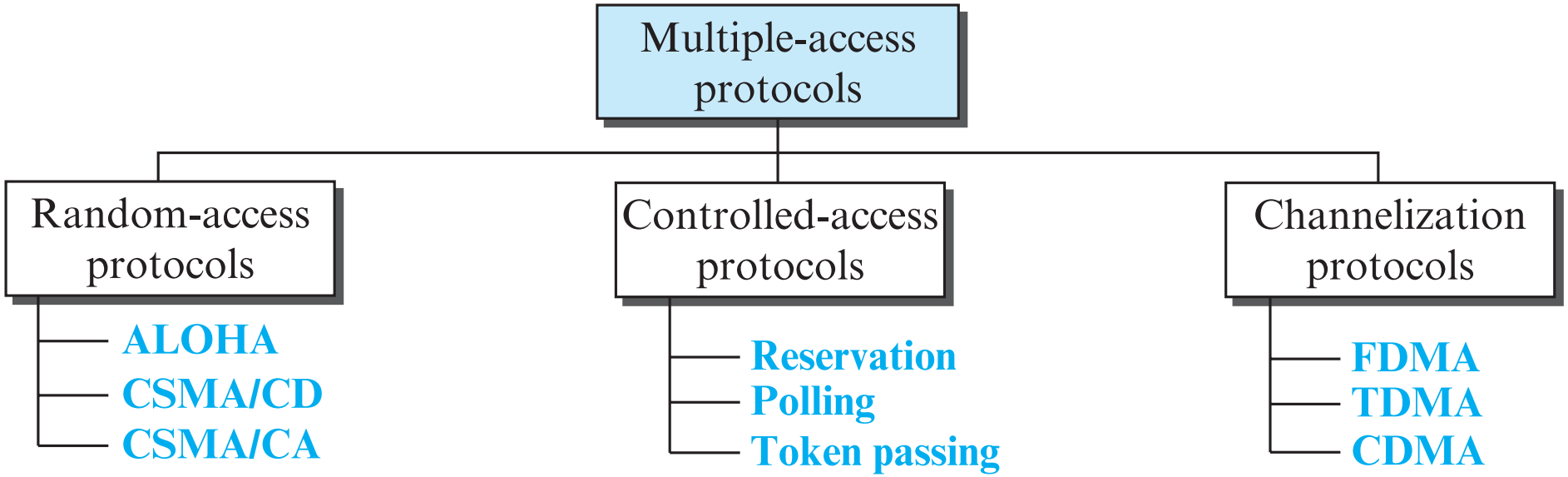
- control *token* passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



# 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,  token ring (e.g., Fiber Distributed Data Interface, FDDI)

# Summary of MAC protocols



# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

- a day in the life of a web request

# MAC addresses

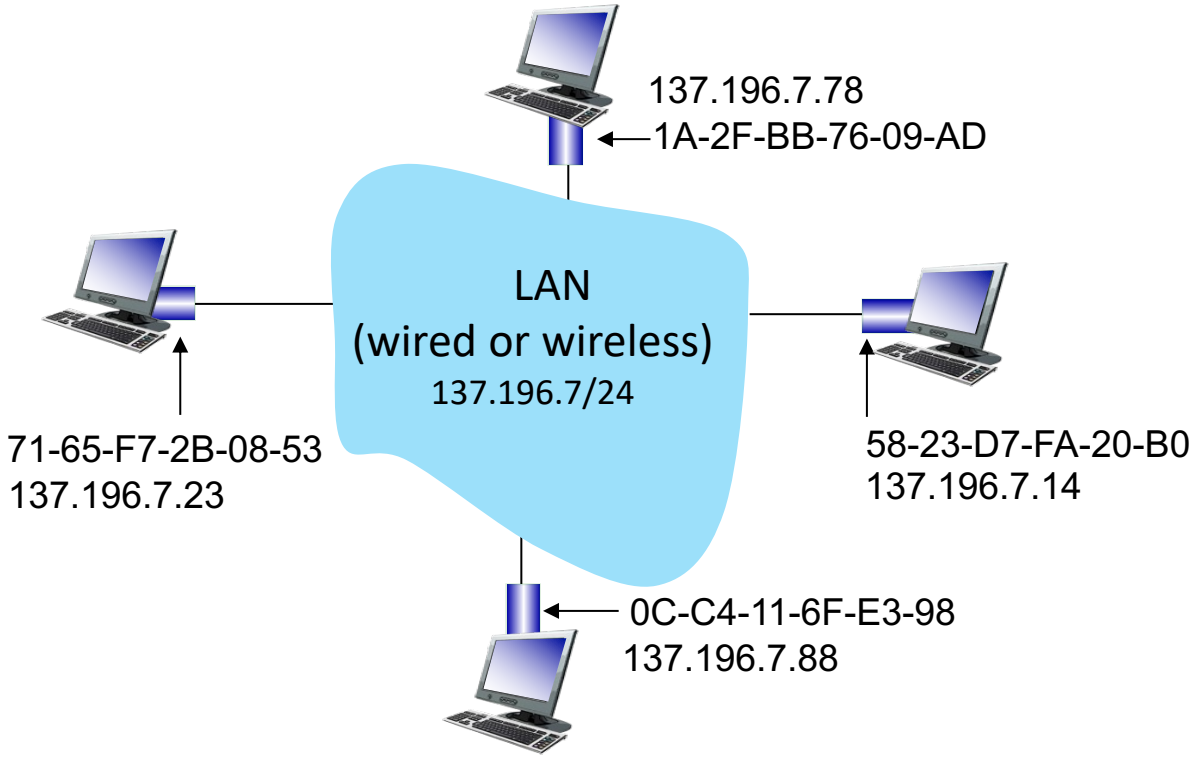
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface (same subnet,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

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

# MAC addresses

each interface on LAN

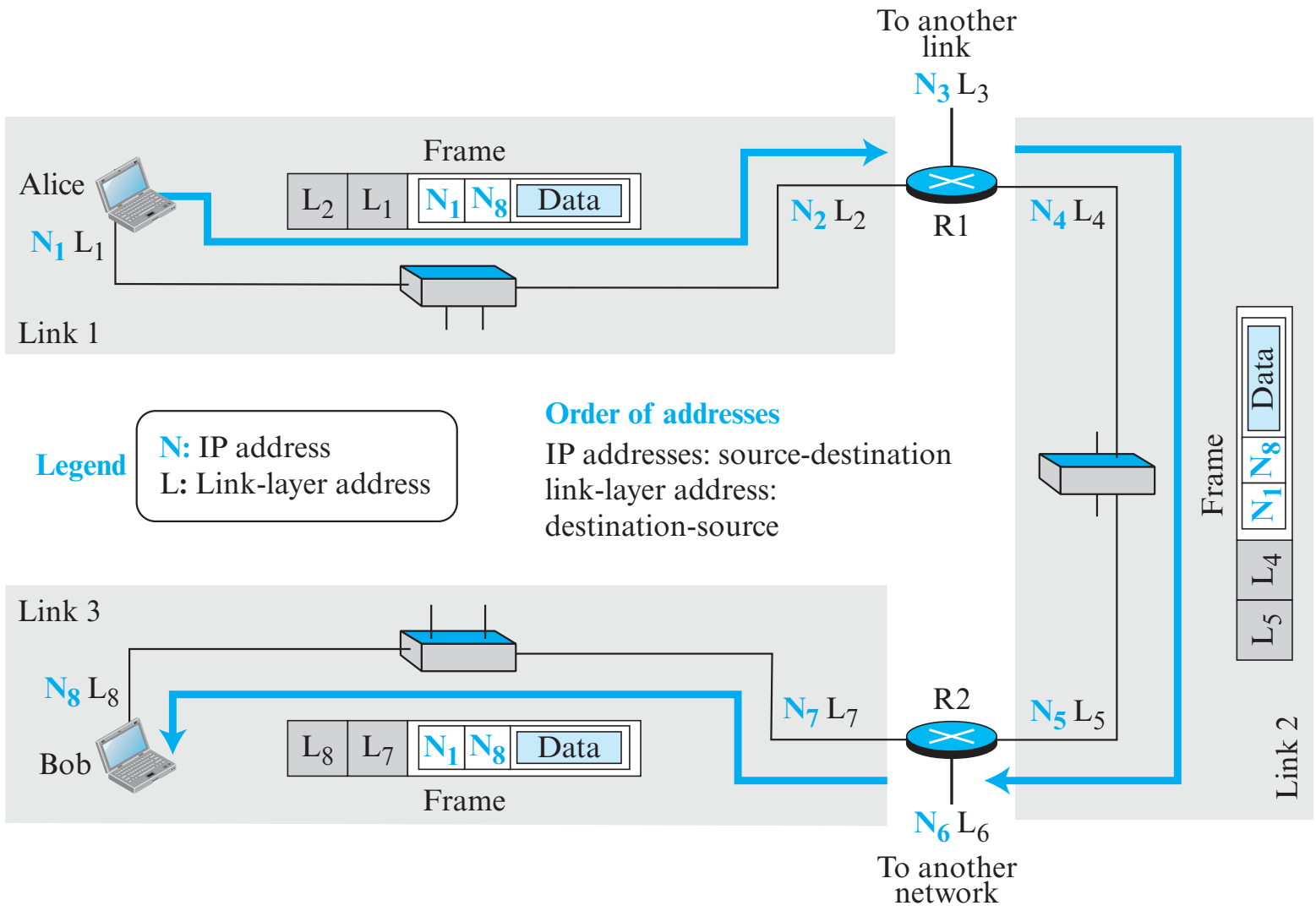
- has unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)



# MAC addresses

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

# IP and Link layer (MAC) addresses



# Question 1

- If the IP address of a router does not appear in any datagram sent from a source to a destination, why do we need to assign IP addresses to routers?

# Question 1

- If the IP address of a router does not appear in any datagram sent from a source to a destination, why do we need to assign IP addresses to routers?
- The answer is that in some protocols a router may act as a sender or receiver of a datagram. For example, in routing protocols a router is a sender or a receiver of a message. The communications in these protocols are between routers.

# Question 2

- Why do we need more than one IP address in a router, one for each interface?

# Question 2

- Why do we need more than one IP address in a router, one for each interface?
- The answer is that an interface is a connection of a router to a link. We said that an IP address defines a point in the Internet at which a device is connected. A router with n interfaces is connected to the Internet at n points.

  *This is the situation of a house at the corner of a street with two gates; each gate has the address related to the corresponding street.*

# Question 3

- How are the source and destination IP addresses in a packet determined?

# Question 3

- How are the source and destination IP addresses in a packet determined?
- The answer is that the host should know its own IP address, which becomes the source IP address in the packet. The application layer uses the services of DNS to find the destination address of the packet and passes it to the network layer to be inserted in the packet.

# Question 4

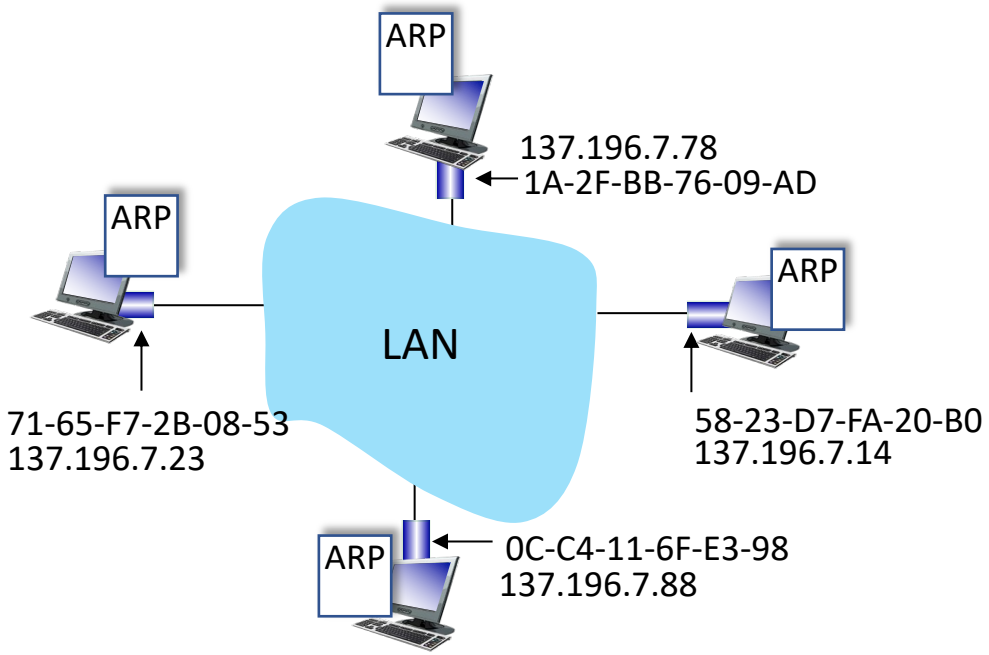
- How are the source and destination link-layer addresses determined for each link?

# Question 4

- How are the source and destination link-layer addresses determined for each link?
- Each hop (router or host) should know its own link-layer address. The destination link-layer is determined by using the Address Resolution Protocol (ARP).

# 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 an ARP 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

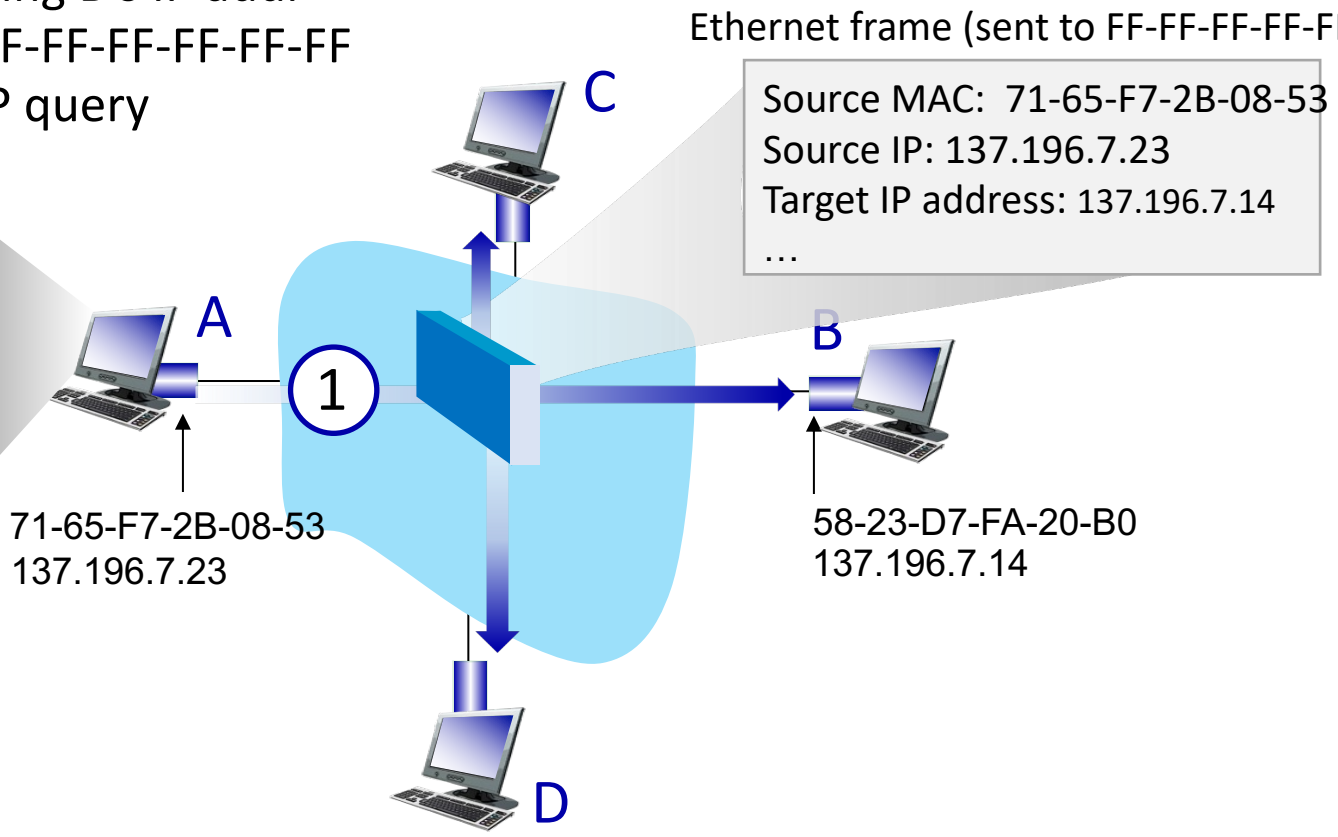
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

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

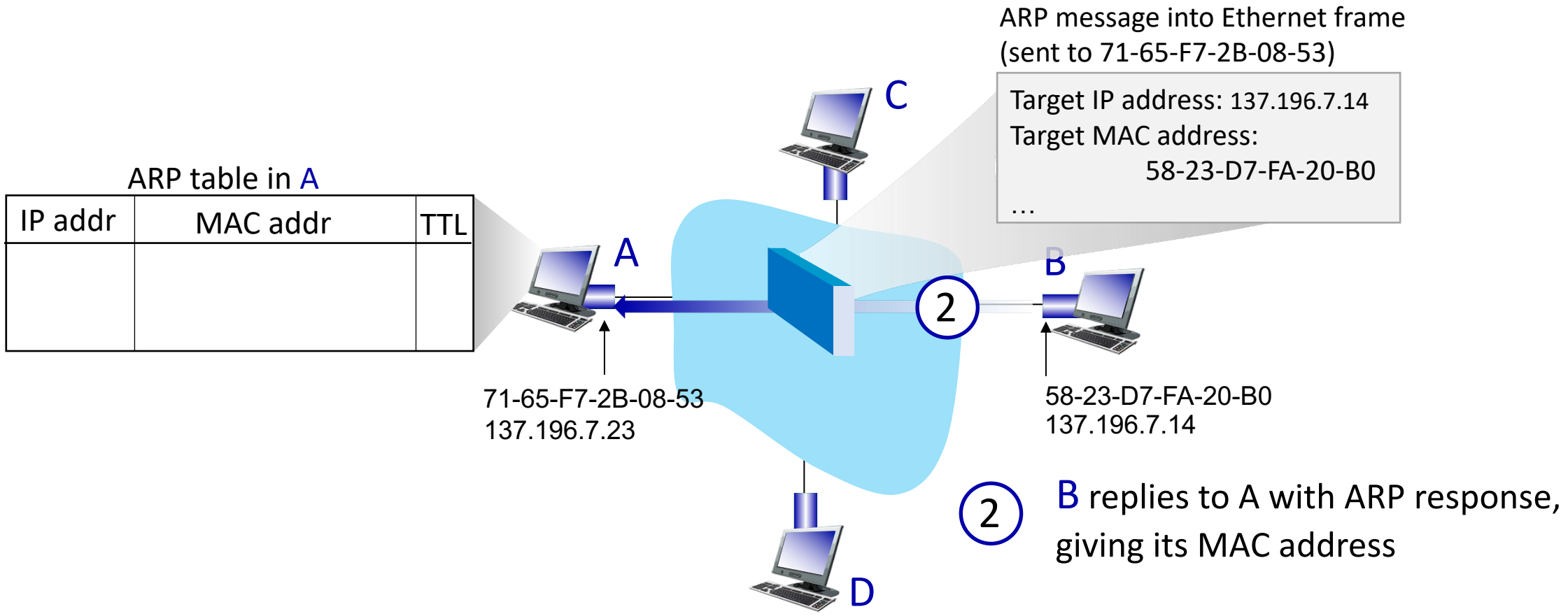
| IP addr | MAC addr | TTL |
|---|---|---|
| | | |



# 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



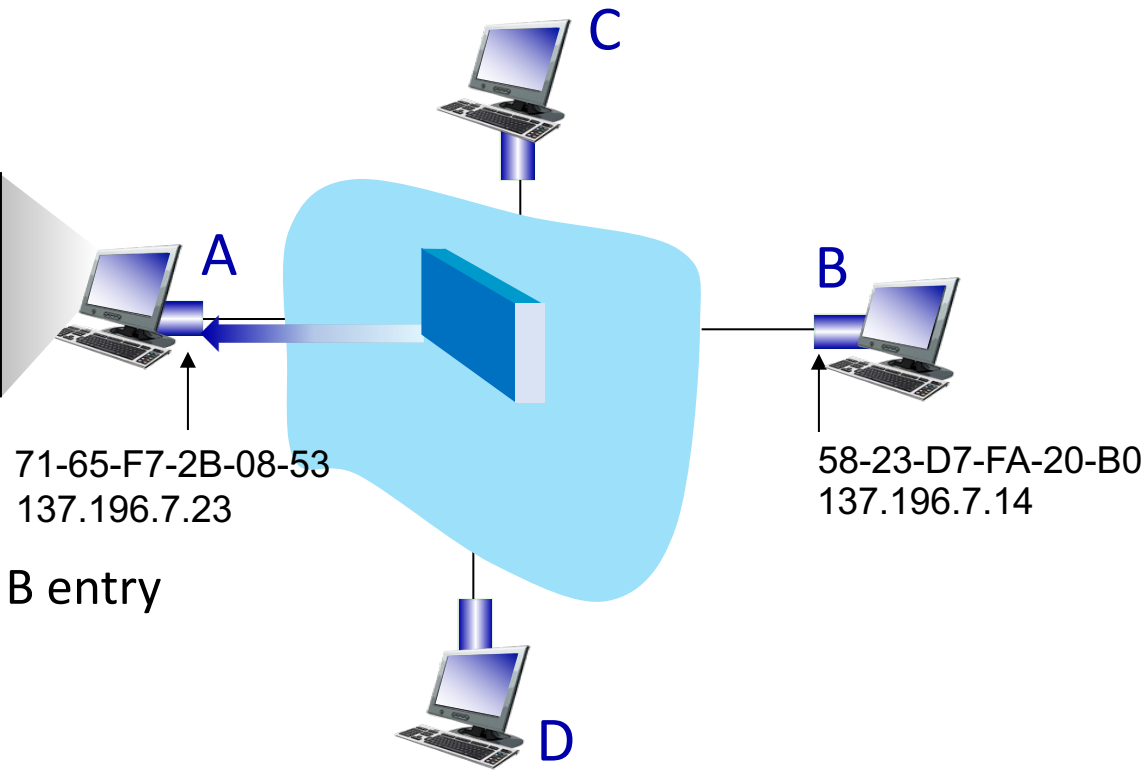
# ARP protocol in action

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 |

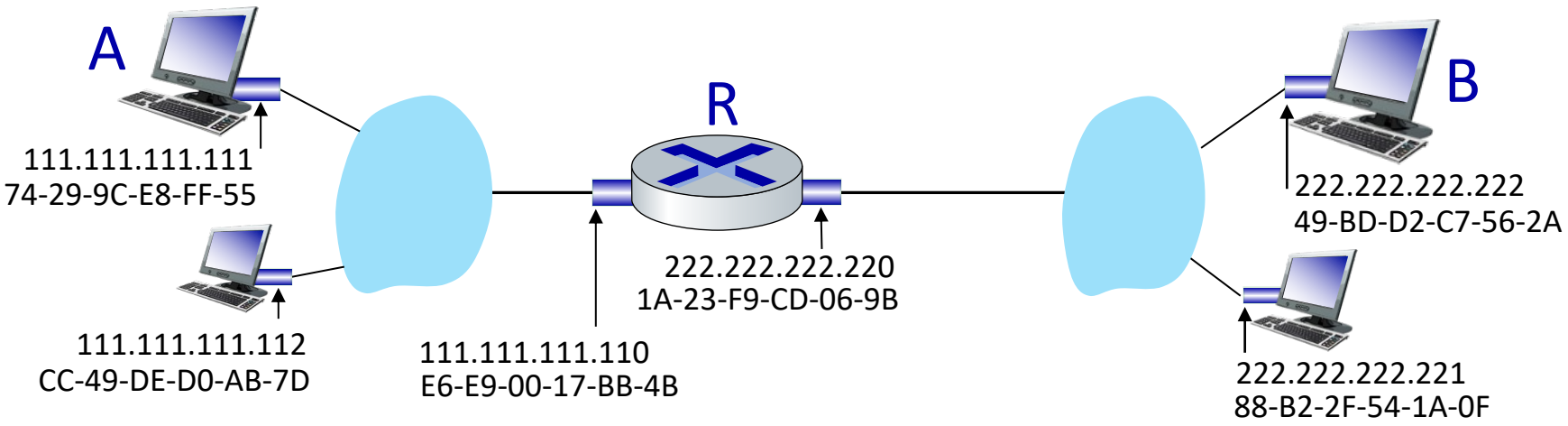



③ A receives B's reply, adds B entry into its local ARP table

# Routing to another subnet: addressing

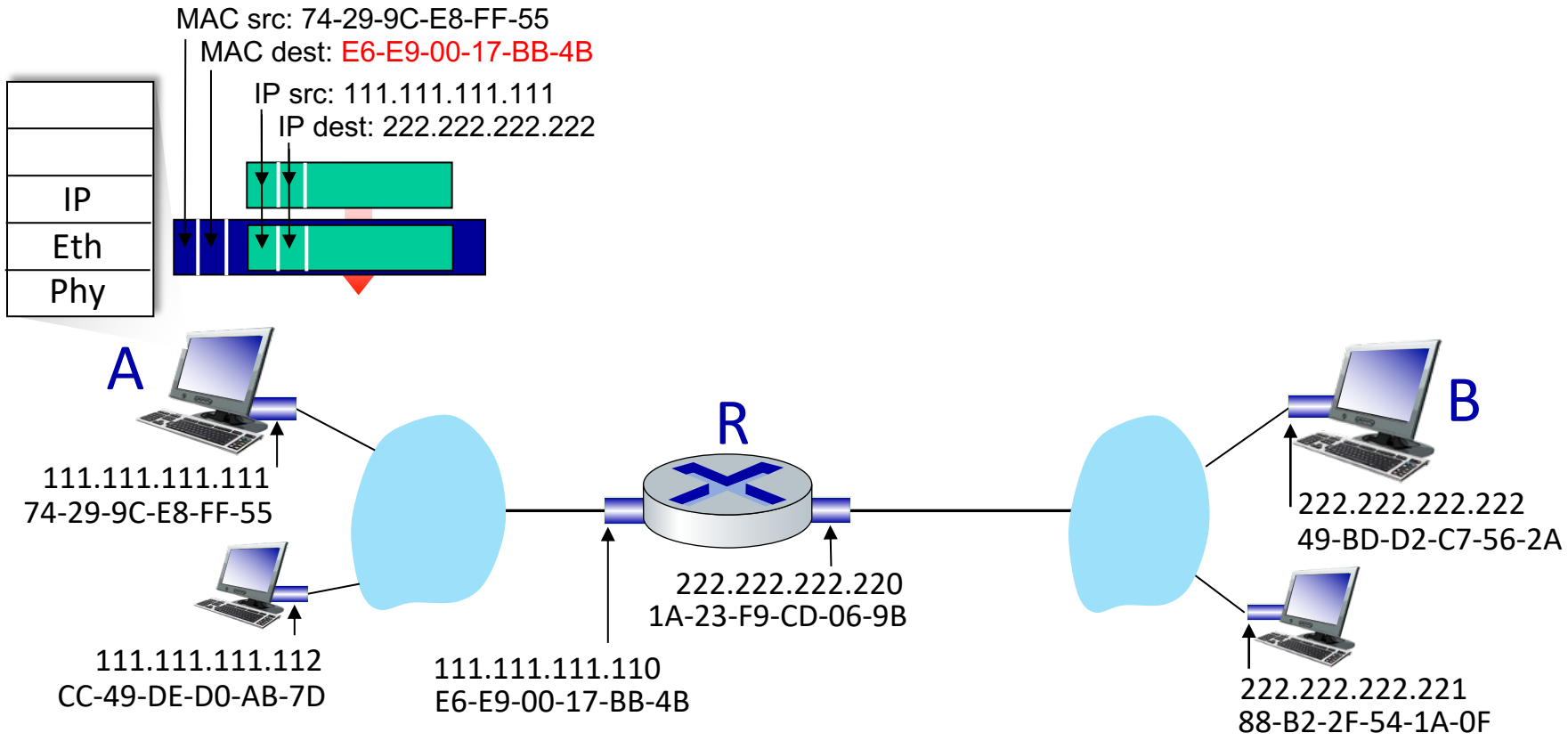
walkthrough: sending a datagram from *A* to *B* via *R*

- 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?)



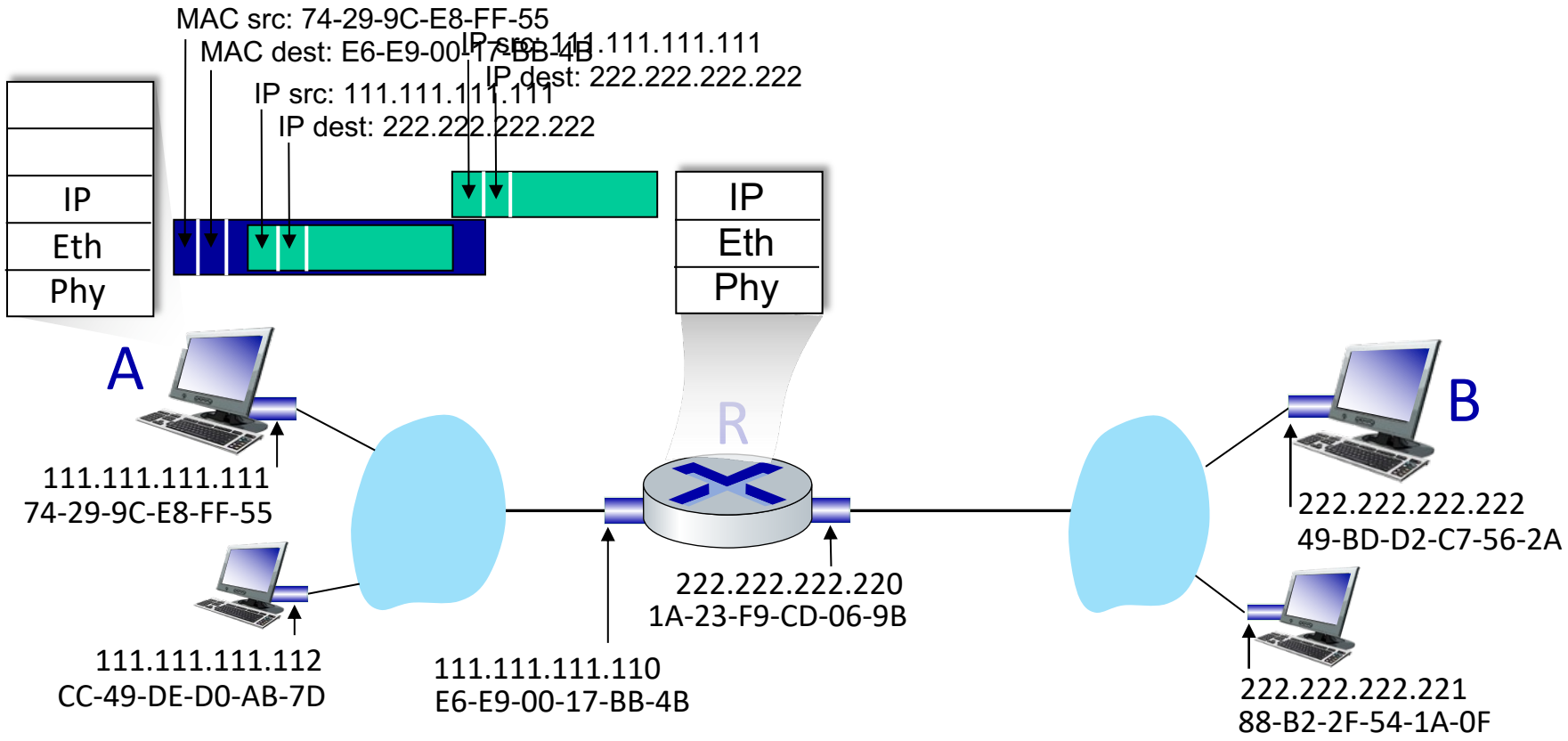
# Routing to another subnet: addressing

- 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



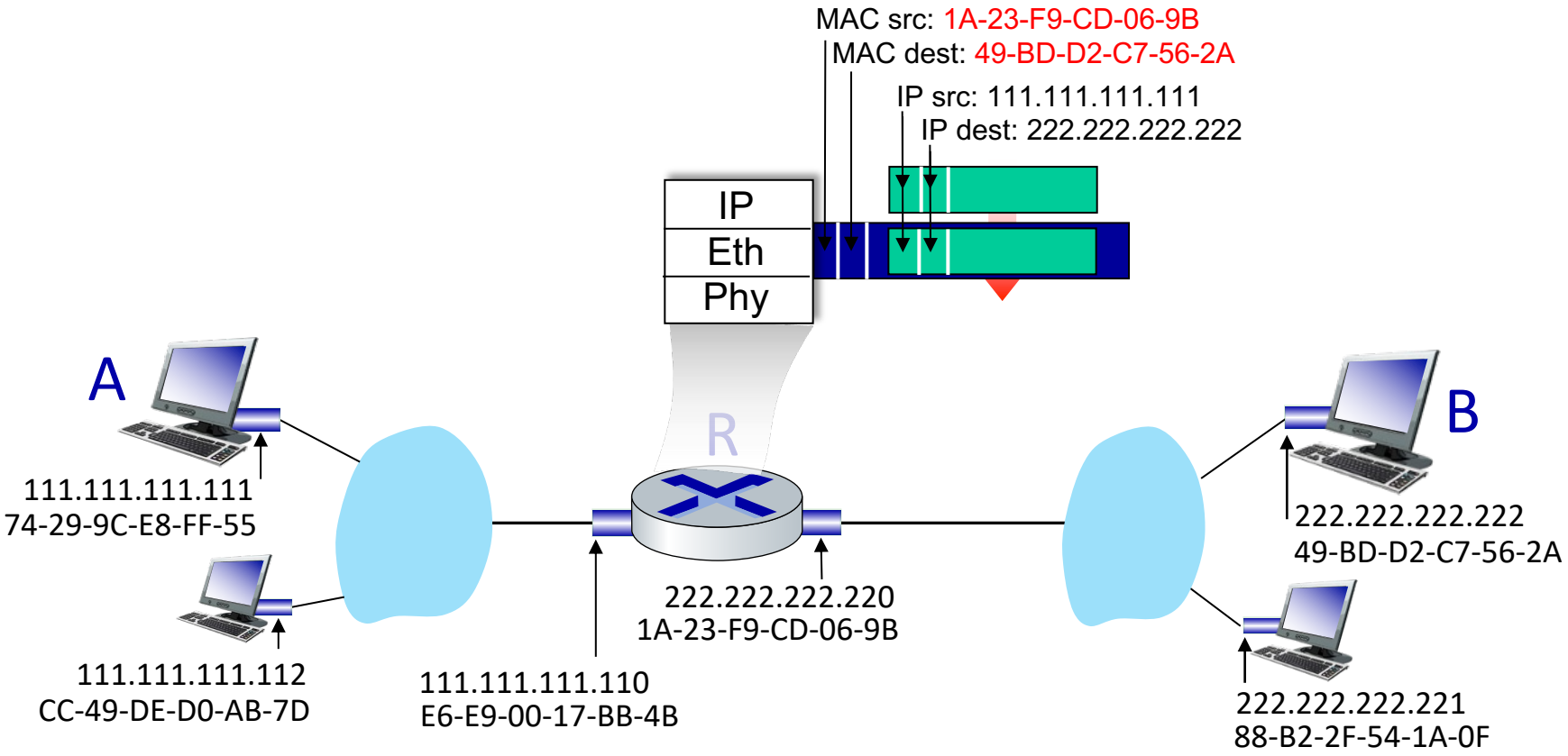
# Routing to another subnet: addressing

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



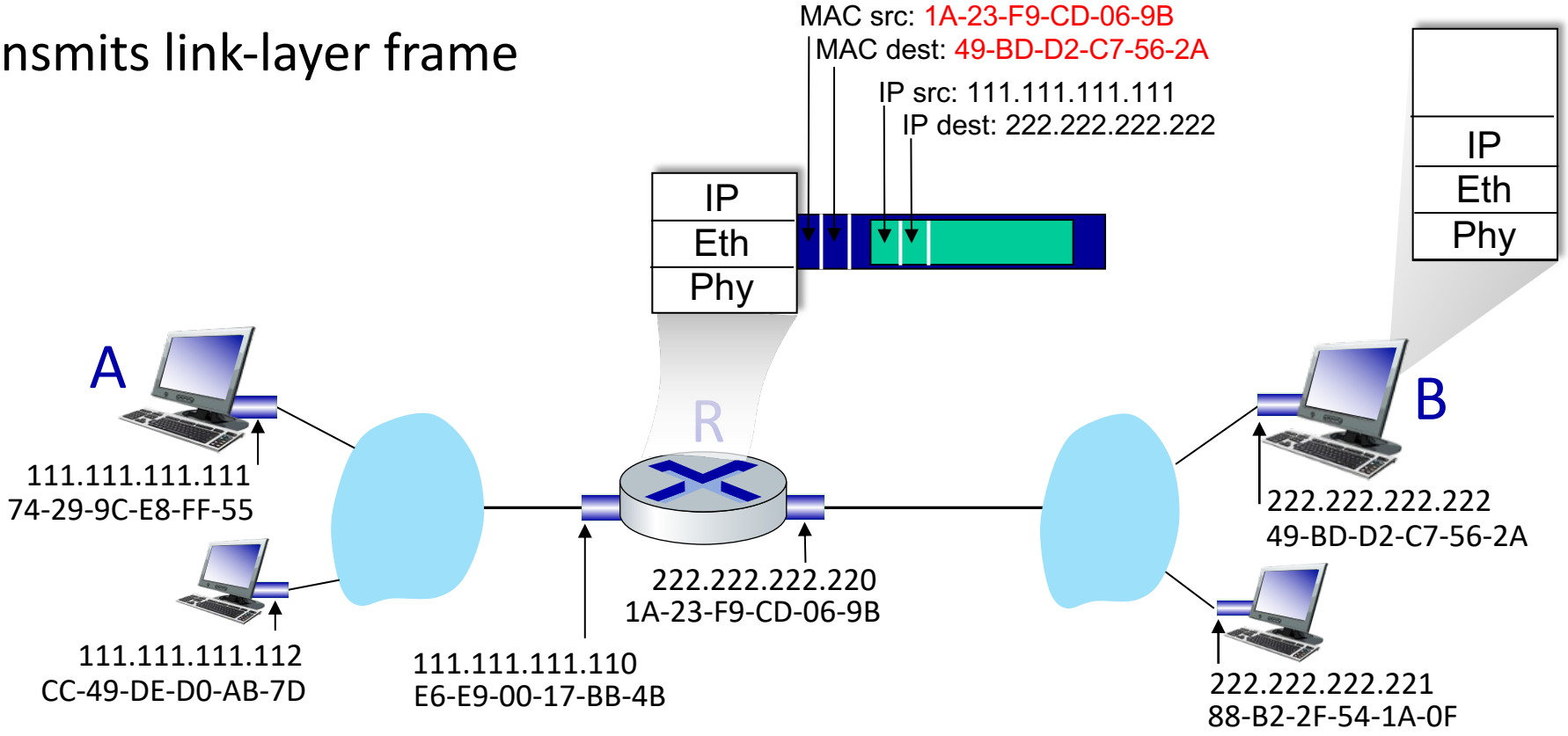
# Routing to another subnet: addressing

- 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



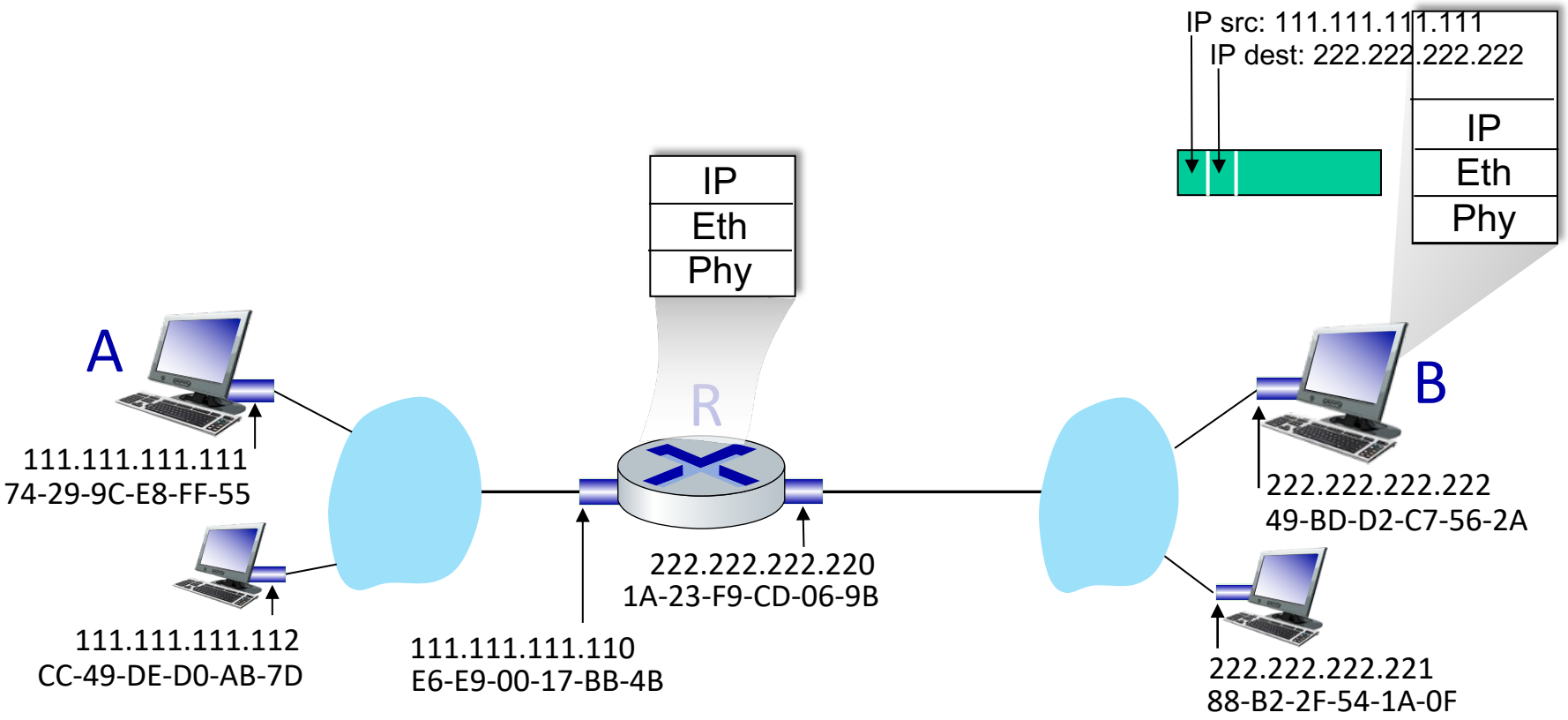
# Routing to another subnet: addressing

- 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

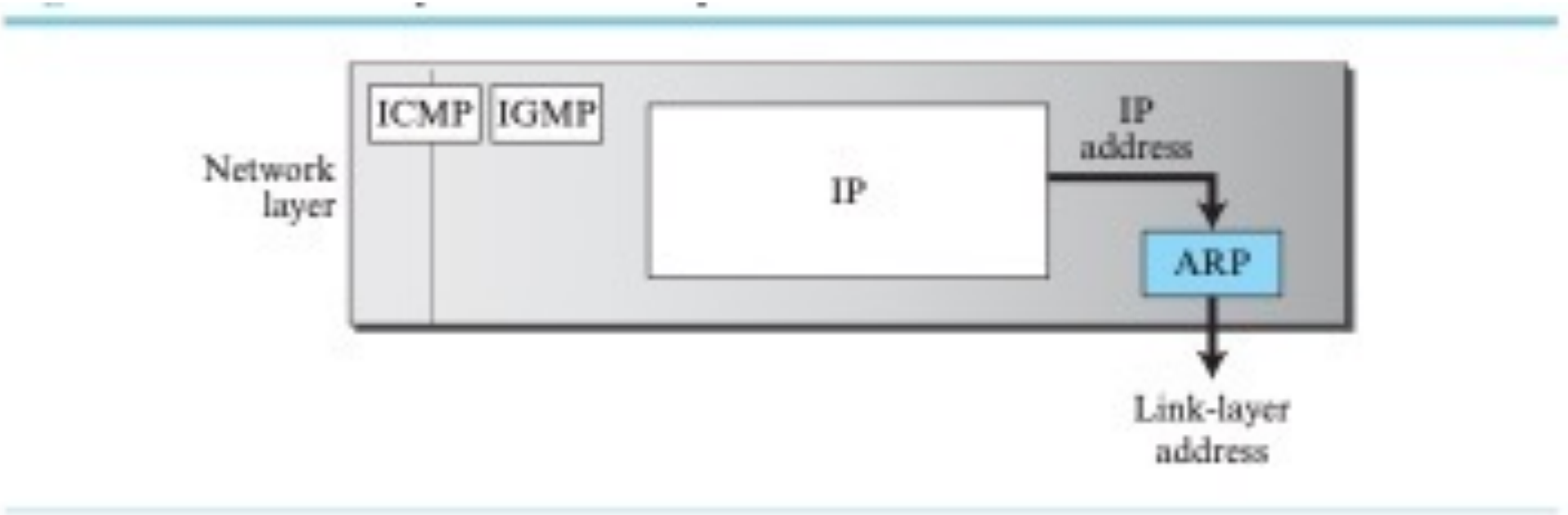


# Routing to another subnet: addressing

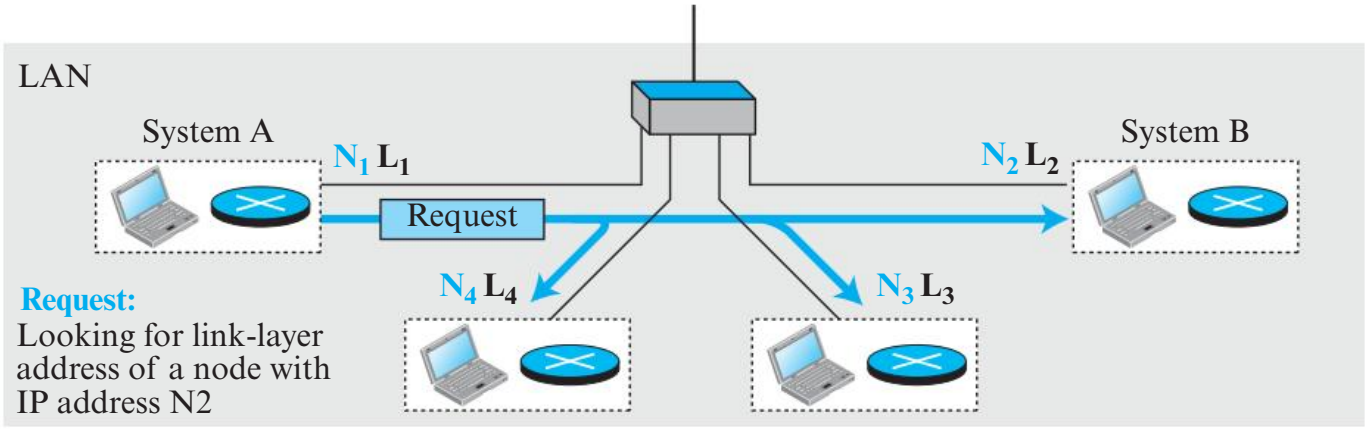
- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



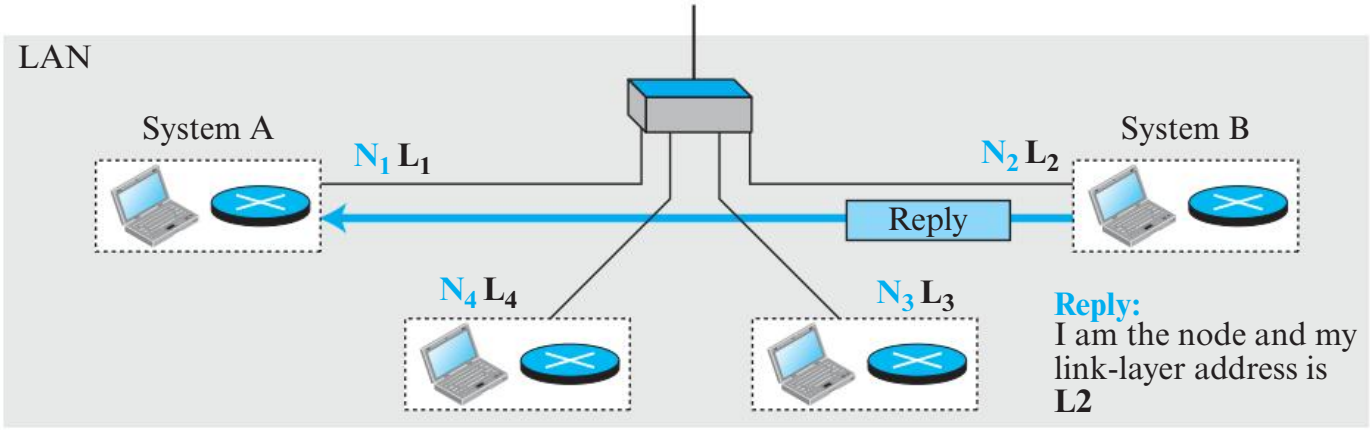
# Position of ARP in the TCP-IP protocol suite



# ARP operation

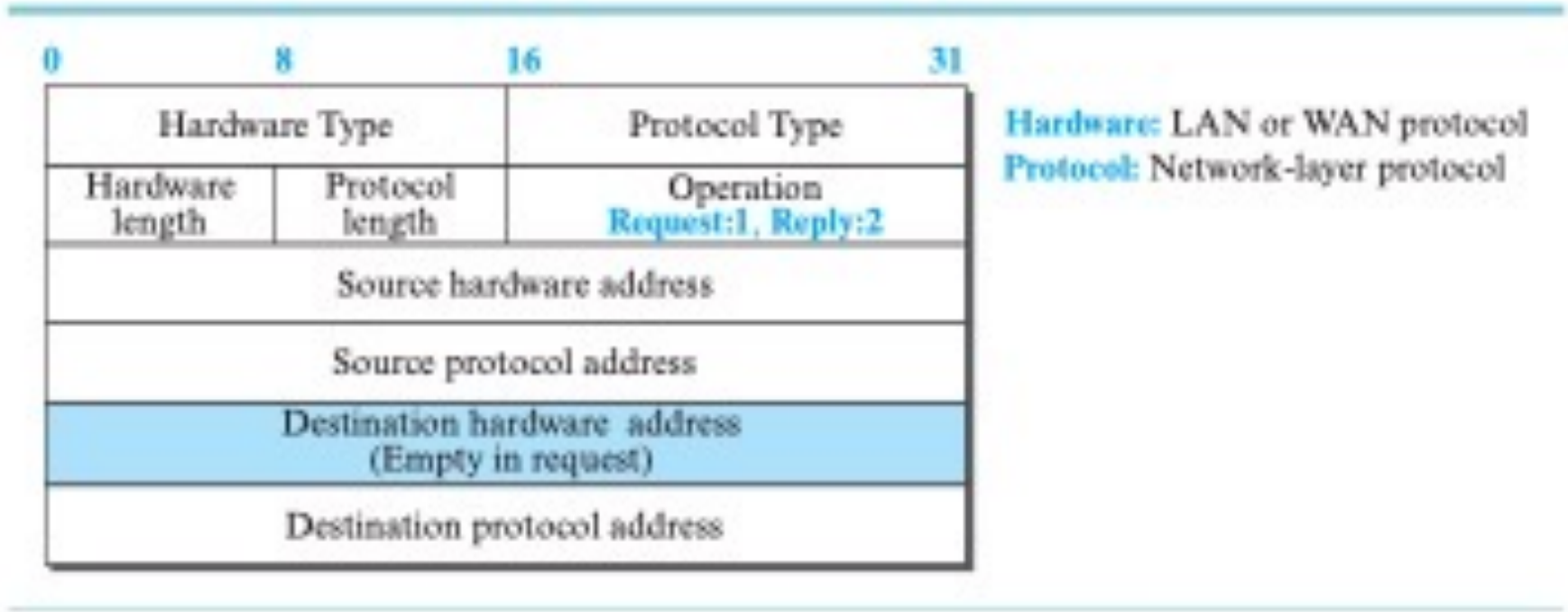



a. ARP request is broadcast

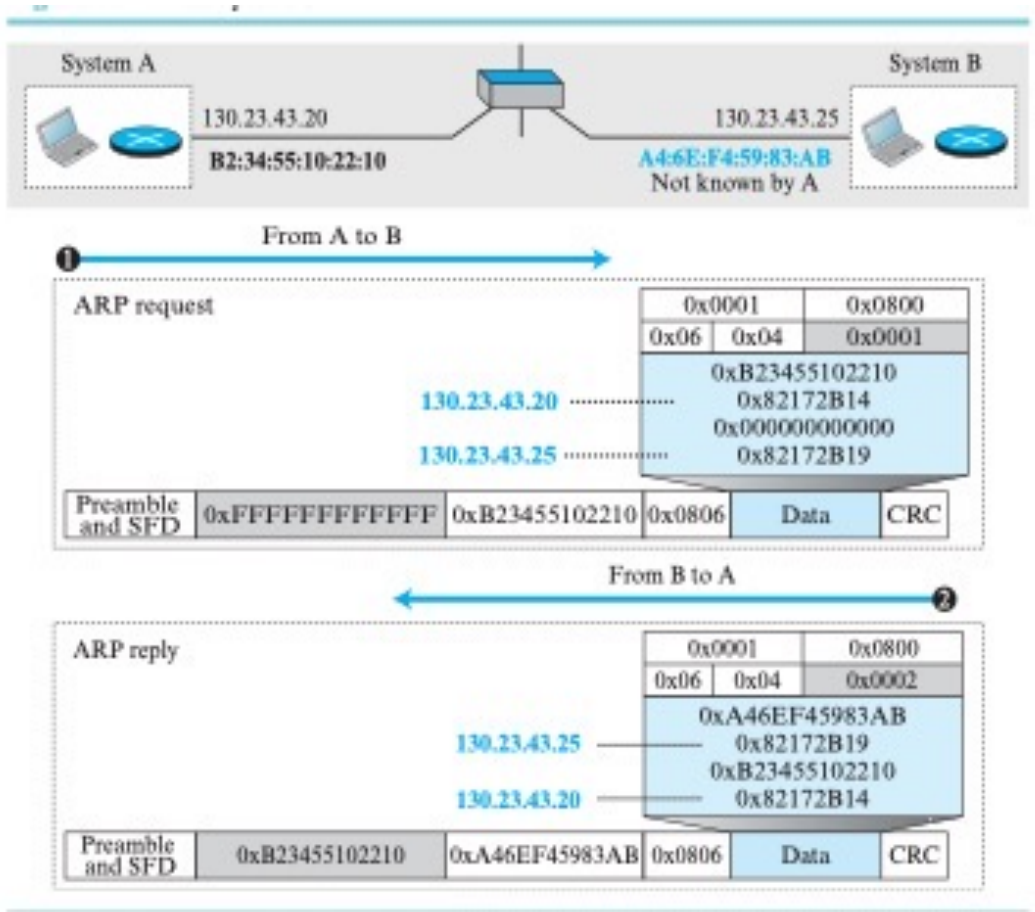



b. ARP reply is unicast

# ARP packet format

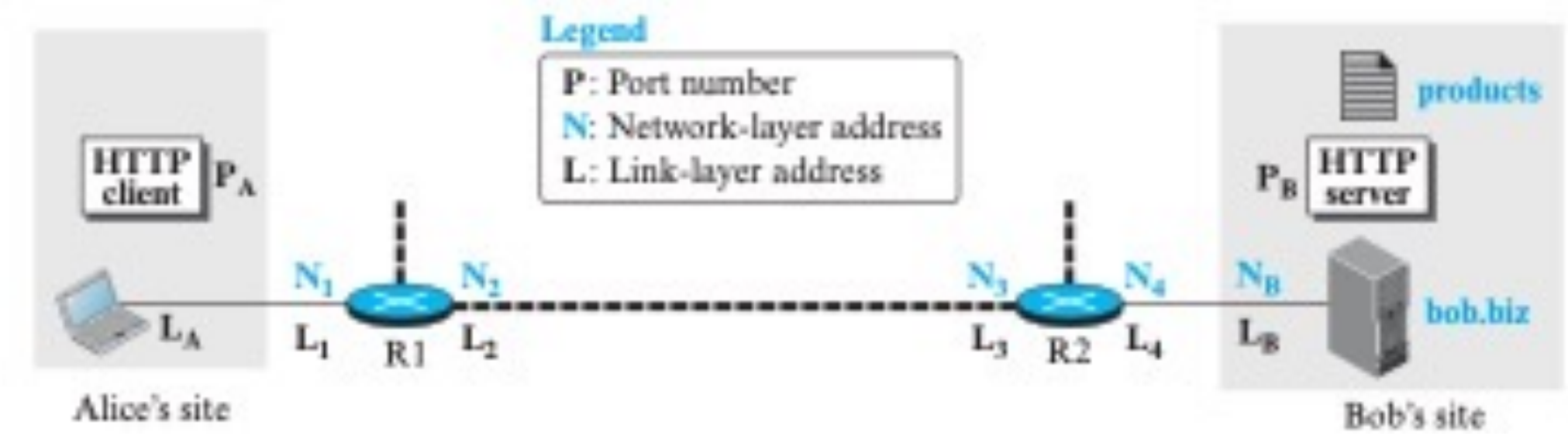


# Example

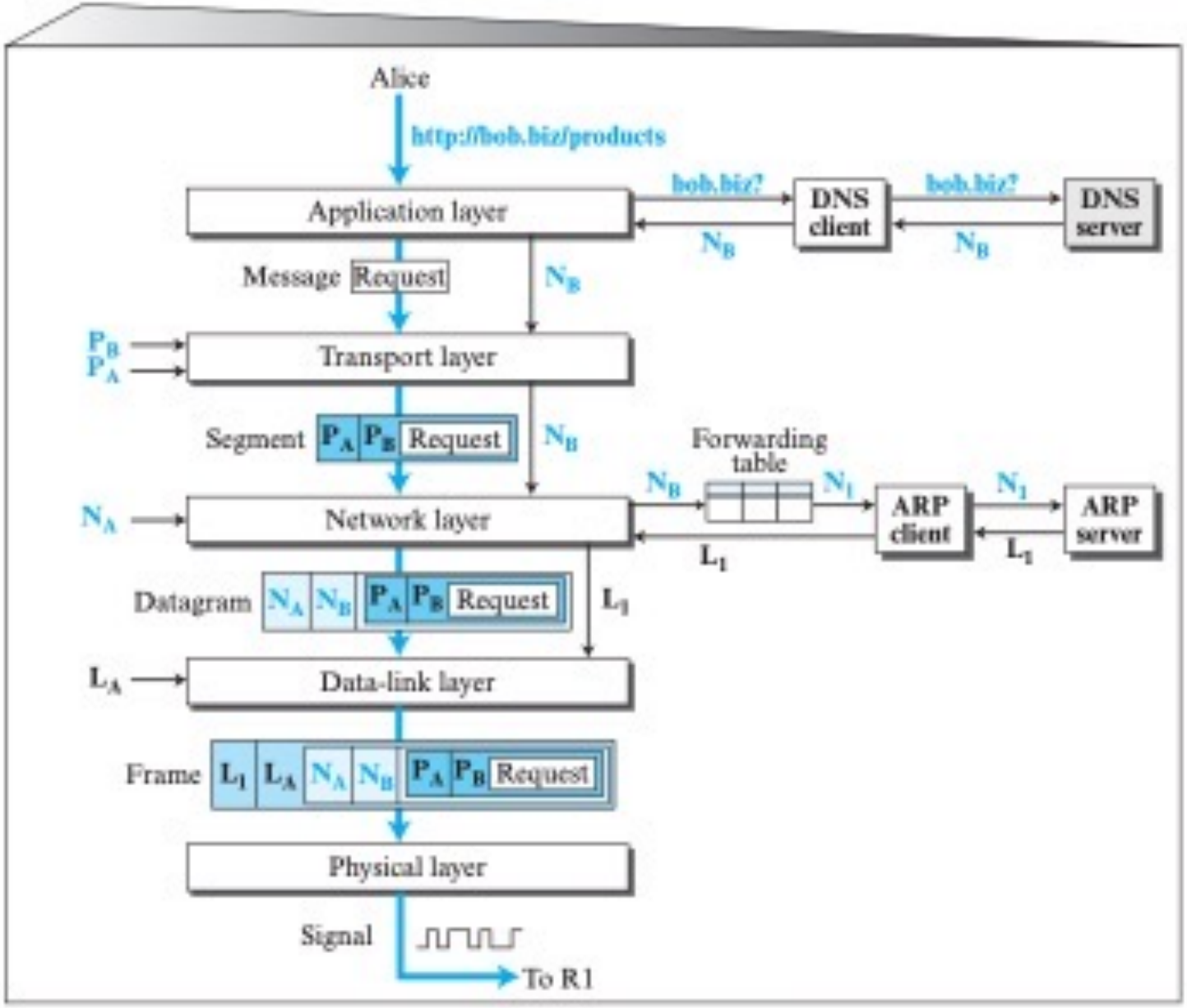



A host with IP address N1 and MAC address **L1** has a packet to send to another host with IP address N2 and physical address **L2** (which is unknown to the first host). The two hosts are on the same network. Show the ARP request and reply packets encapsulated in Ethernet frames.

# Example of address management in an HTTP interaction (1/4)

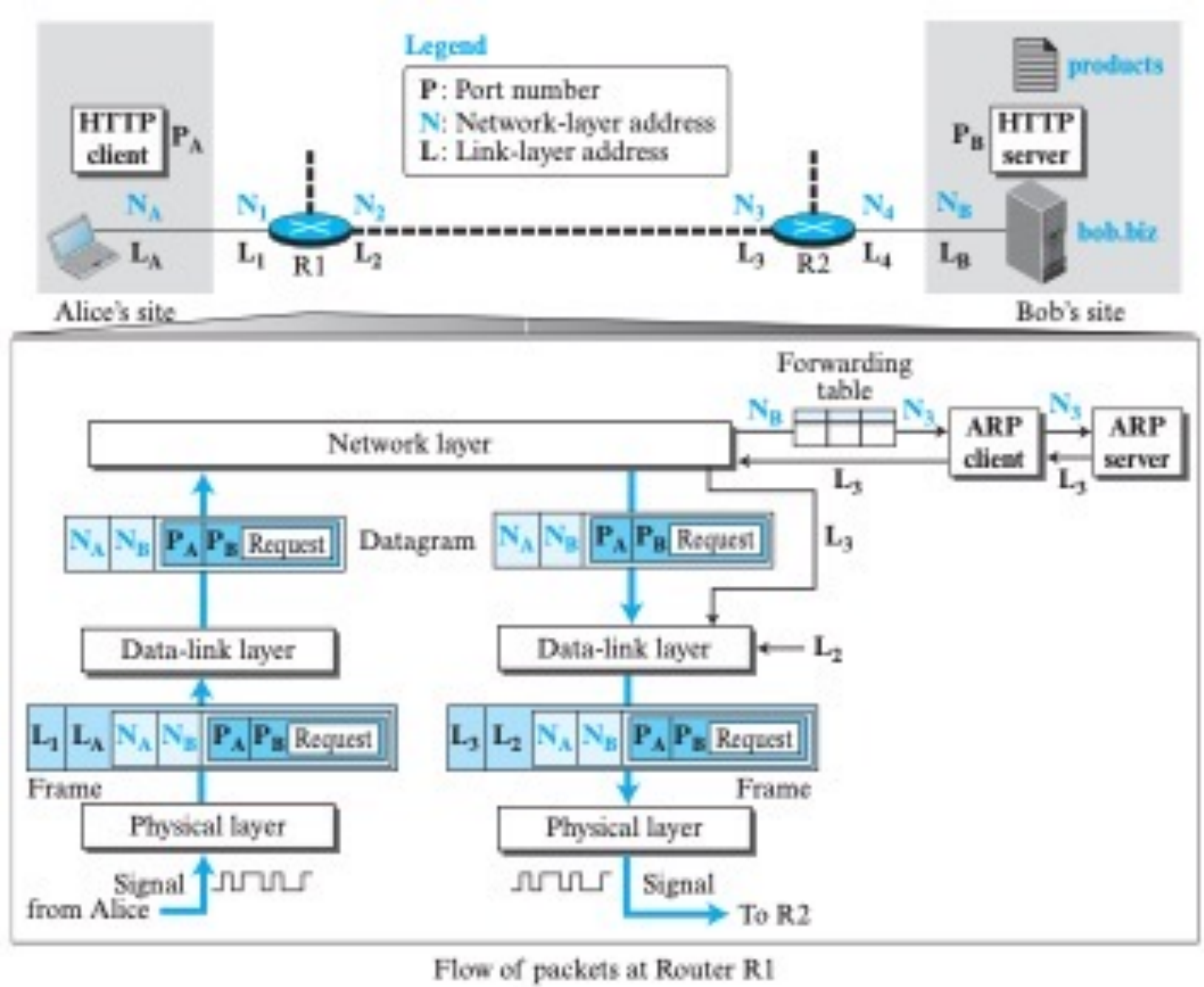


# … example of HTTP interaction (2/4)



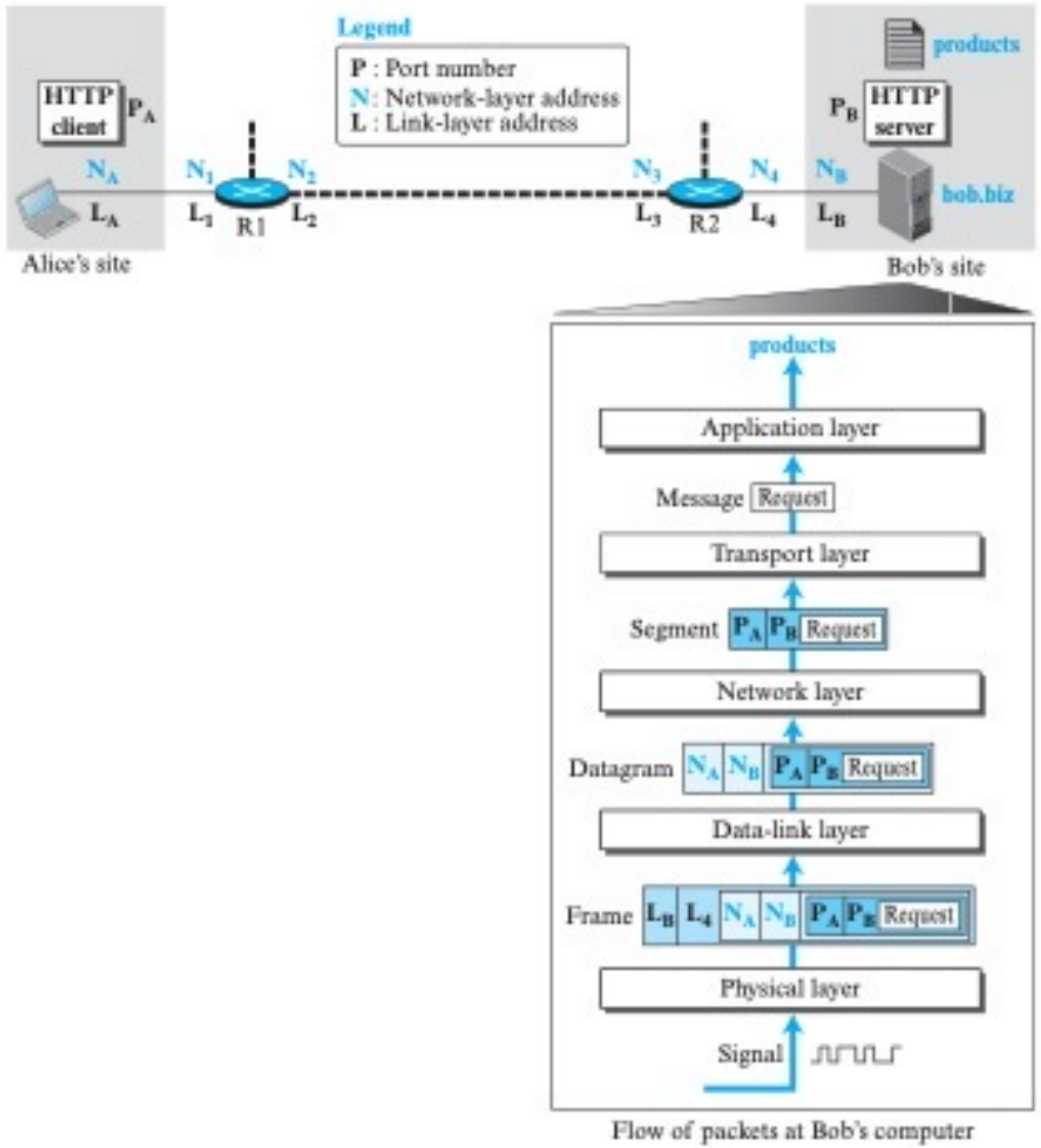

Flow of packets at Alice's computer

# …example of HTTP interaction (3/4)




Flow of packets at Router R1

# …example of HTTP interaction (4/4)

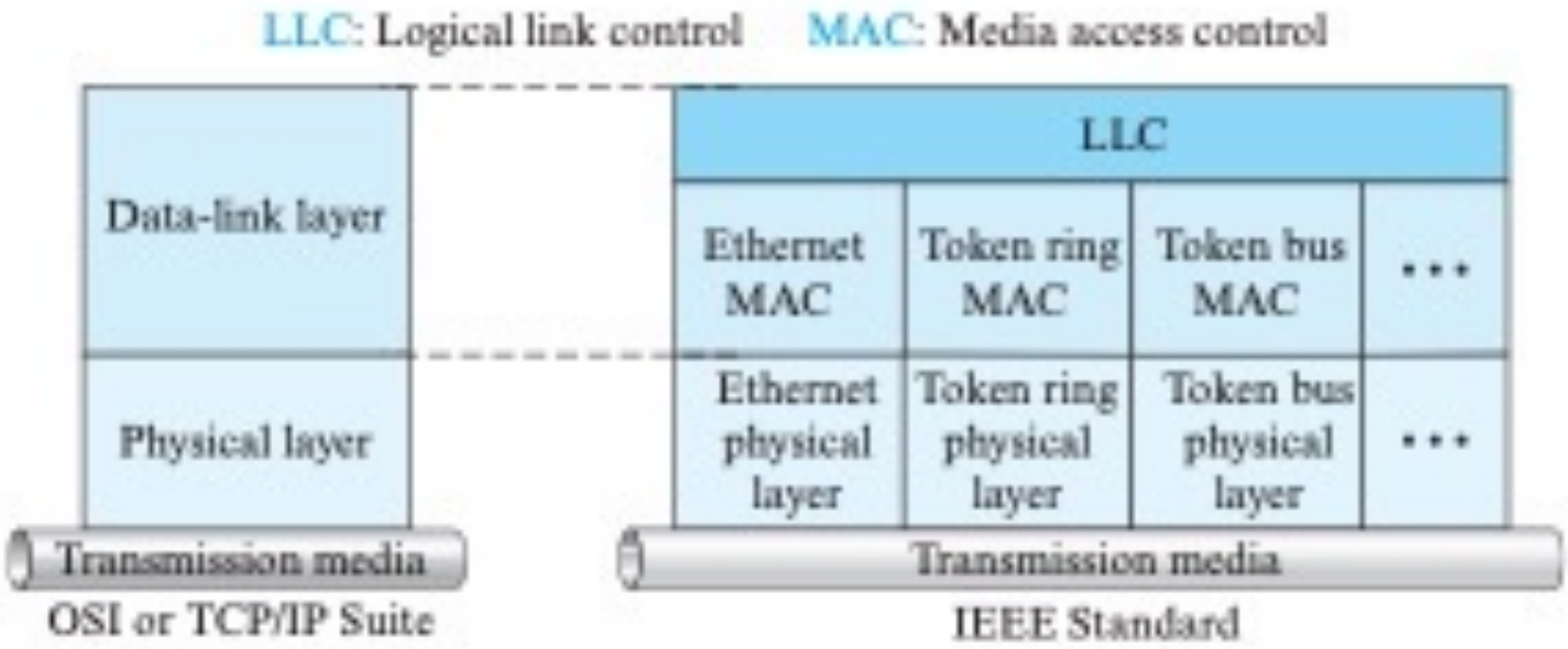


# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

- a day in the life of a web request

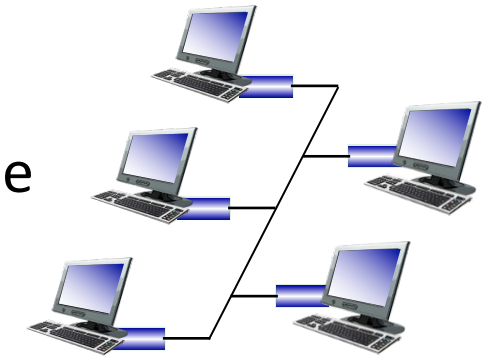
# IEEE 802 project – standards for LANs

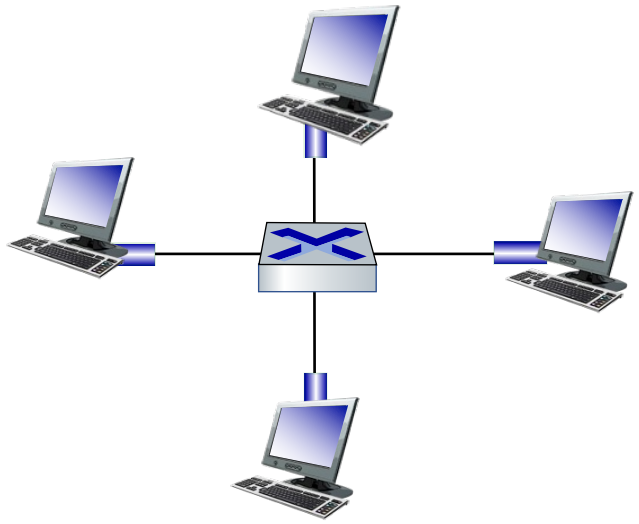


# Ethernet: physical topology

- 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

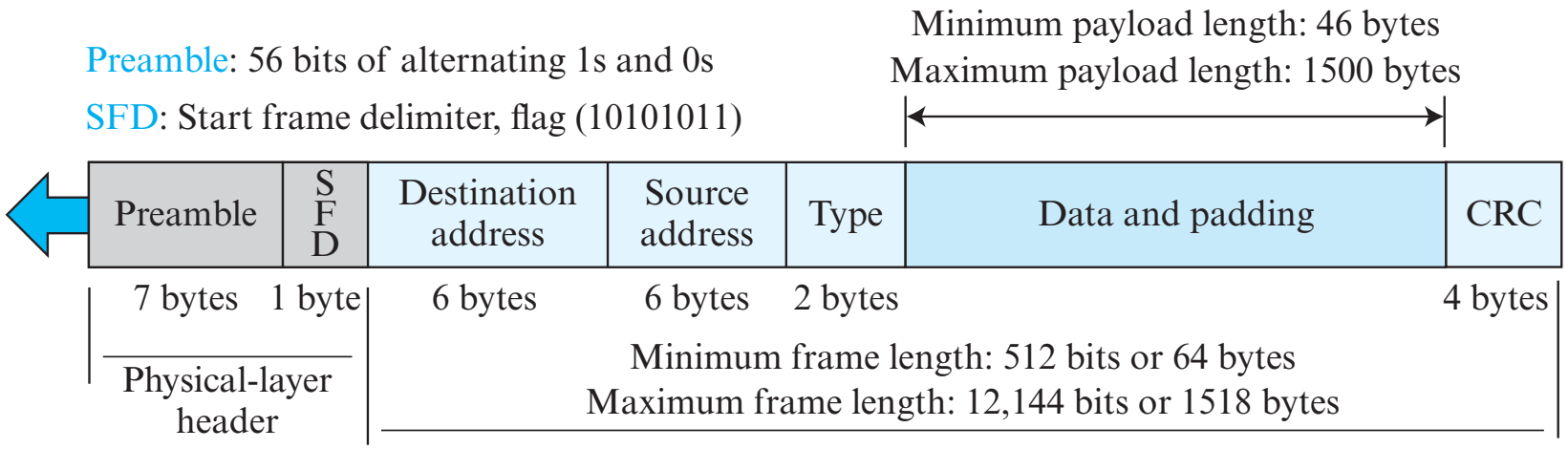




switched

# Ethernet frame structure

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

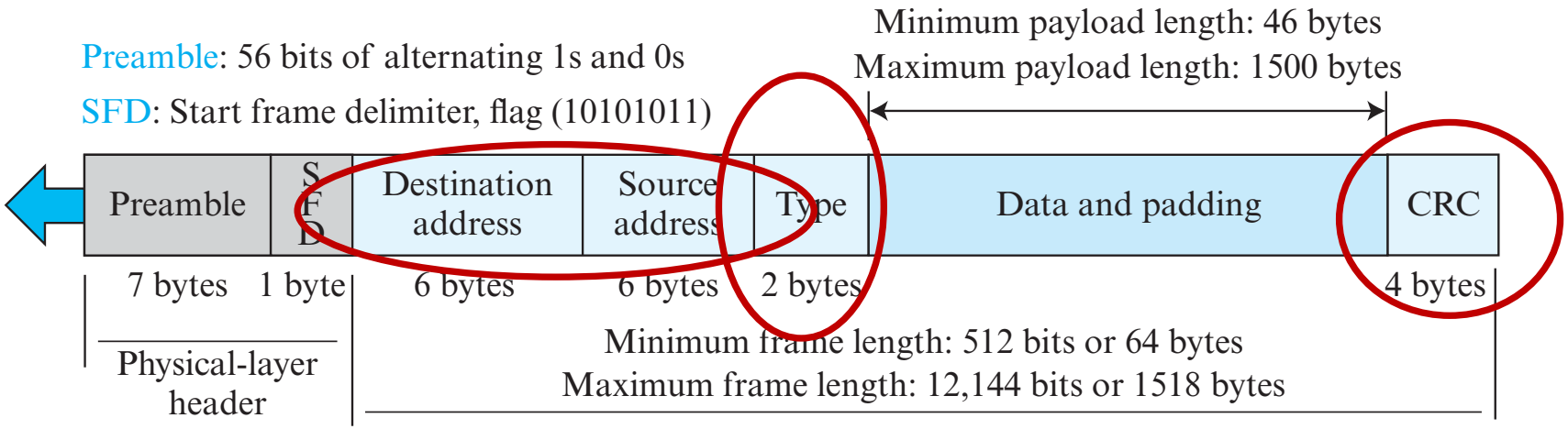



*preamble:*

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

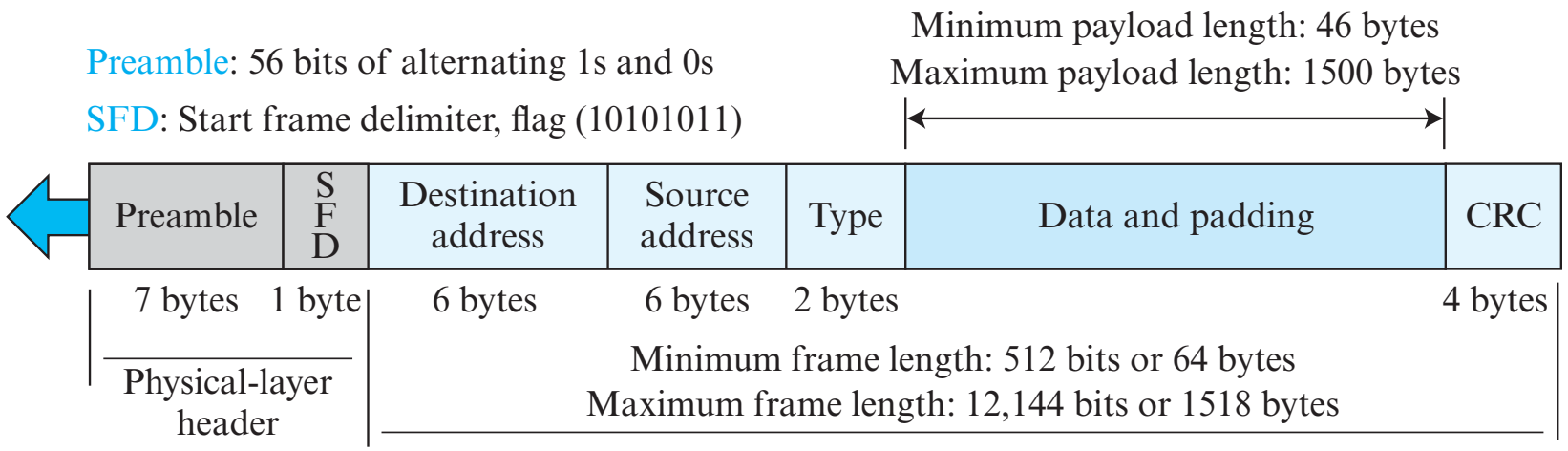
Added by physical layer

# Ethernet frame structure (more)




- addresses: 6 byte source and 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., ARP, OSPF
  - used to demultiplex up at receiver
- CRC: cyclic redundancy check at receiver
  - error detected: frame is dropped

# Ethernet frame structure – data and padding



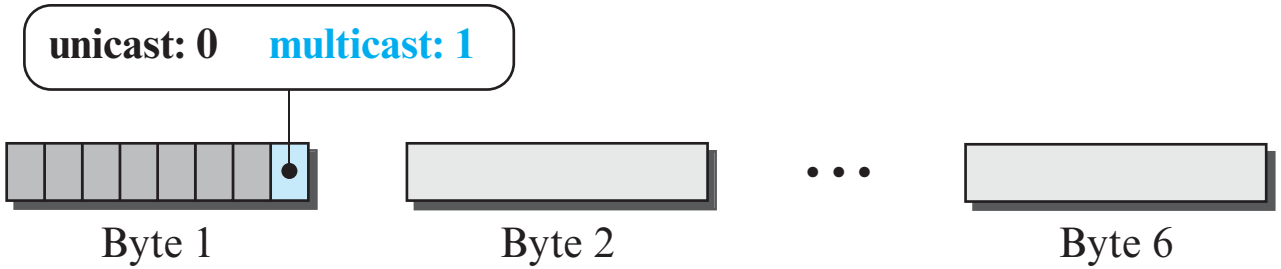

**Minimum frame length: 64 bytes**
**Maximum frame length: 1518 bytes**

**Minimum data length: 46 bytes**
**Maximum data length: 1500 bytes**

Max data length defined to limit buffer size and to avoid excessive amount of data transferred by a unique device (no channel monopolization)

Min data length obtained by adding padding bits upon necessity (collision detection)

# Transmission of MAC address in reverse bit order




| Hexadecimal | 47 | 20 | 1B | 2E | 08 | EE |
|---|---|---|---|---|---|---|
| Binarys | 01000111 | 00100000 | 00011011 | 00101110 | 00001000 | 11101110 |
| Transmitted ← | 11100010 | 00000100 | 11011000 | 01110100 | 00010000 | 01110111 |

In this manner the first bit identifies a multicast or a broadcast (all 1s)

# Unicast, multicast, broadcast

- In a unicast transmission, all stations will receive the frame, the intended recipient keeps and handles the frame; the rest discard it.
- In a multicast transmission, all stations will receive the frame, the stations that are members of the group keep and handle it; the rest discard it.
- In a broadcast transmission, all stations (except the sender) will receive the frame and all stations (except the sender) keep and handle it.

# Exercize

Define the type of the following destination addresses:

a. **4A:30:10:21:10:1A**
b. **47:20:1B:2E:08:EE**
c. **FF:FF:FF:FF:FF:FF**

# Exercize

Define the type of the following destination addresses:

a. **4A:30:10:21:10:1A**
b. **47:20:1B:2E:08:EE**
c. **FF:FF:FF:FF:FF:FF**

To find the type of the address, we need to look at the second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are Fs, the address is broadcast. Therefore, we have the following:

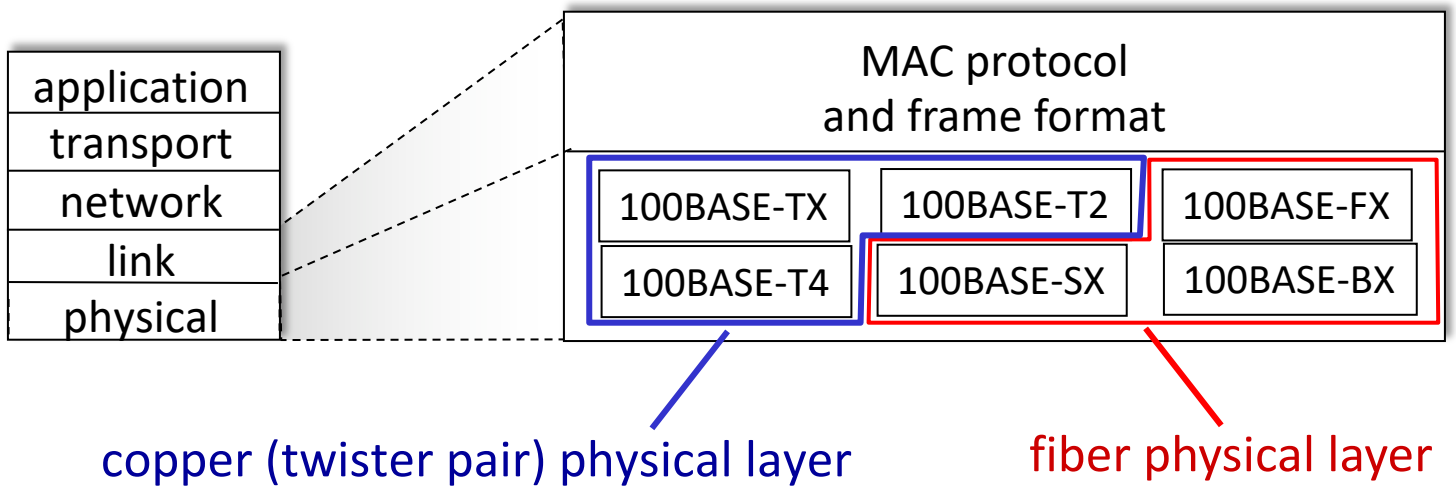
a. This is a unicast address because A in binary is 1010 (even).
b. This is a multicast address because 7 in binary is 0111 (odd).
c. This is a broadcast address because all digits are Fs in hexadecimal.

# Ethernet: unreliable, connectionless

- connectionless: no handshaking between sending and receiving NICs
- unreliable: receiving NIC doesn't 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

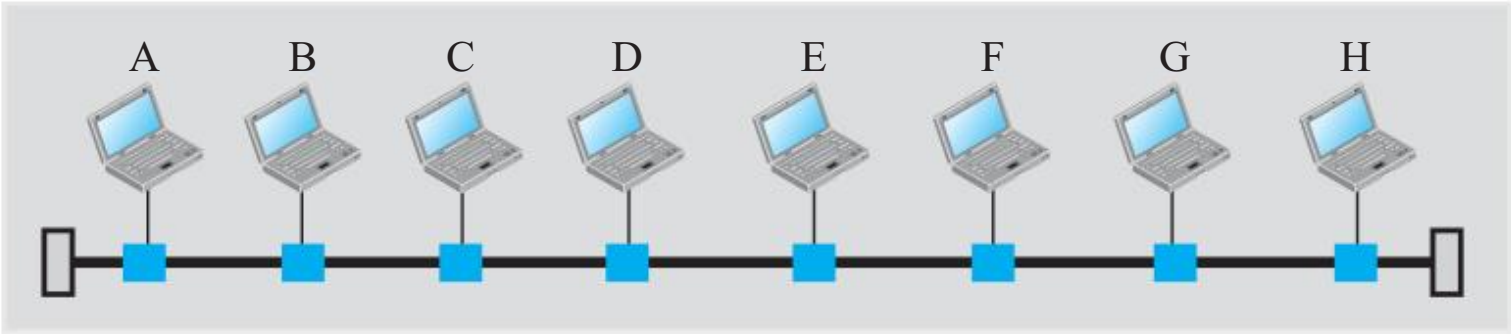


# Link layer, LANs: roadmap

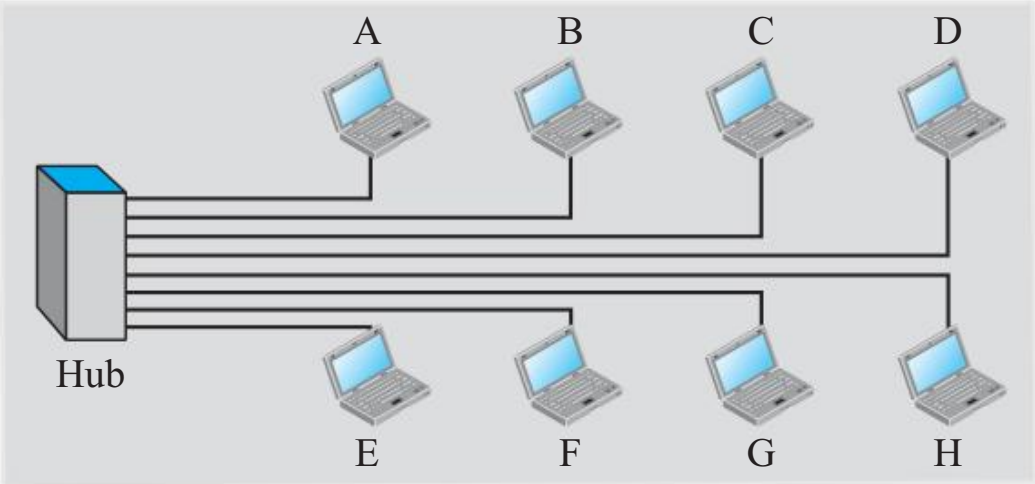
- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

- a day in the life of a web request

# Interconnection devices




a. A LAN with a bus topology using a coaxial cable




b. A LAN with a star topology using a hub

A HUB faces signal attenuation,
it just repeats signals on every outgoing links

# Hubs

- A hub is a **physical layer** device that acts on individual bits (not frames).
- It simply **repeats** (**boosts**) the received signal to all its outgoing interfaces.

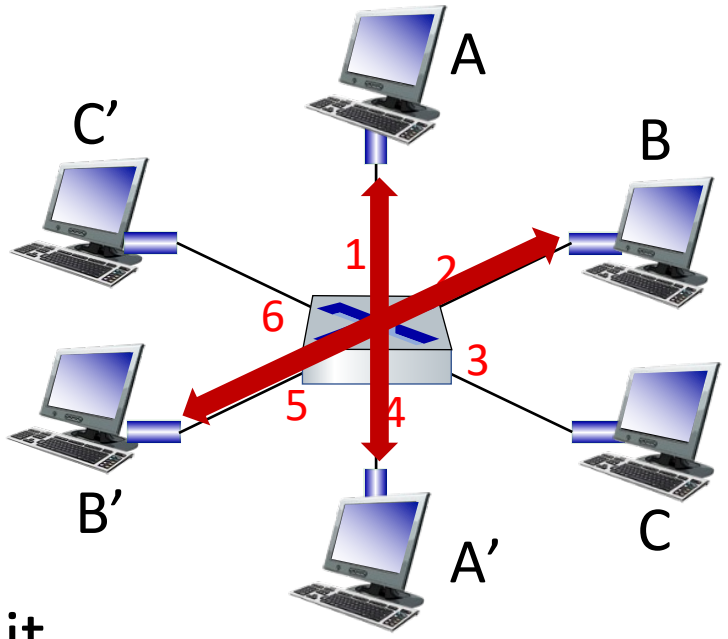
- An Ethernet with a hub-based start topology is a broadcast LAN
  - If a hub receives frames from two different interfaces at the same time, a collsion occurs and the frames must be retransmitted

# Ethernet switch

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

# Switch: multiple simultaneous transmissions

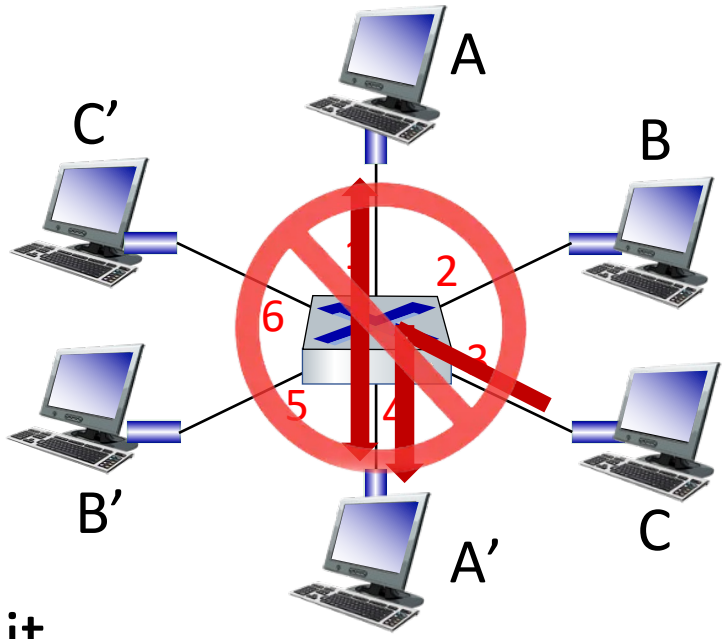
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions




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' can *not* happen simultaneously



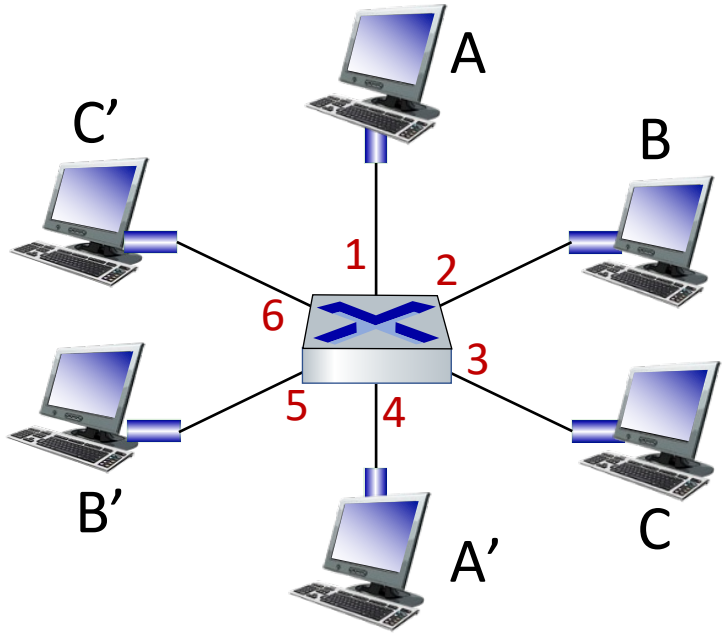

switch with six interfaces (1,2,3,4,5,6)

# 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!

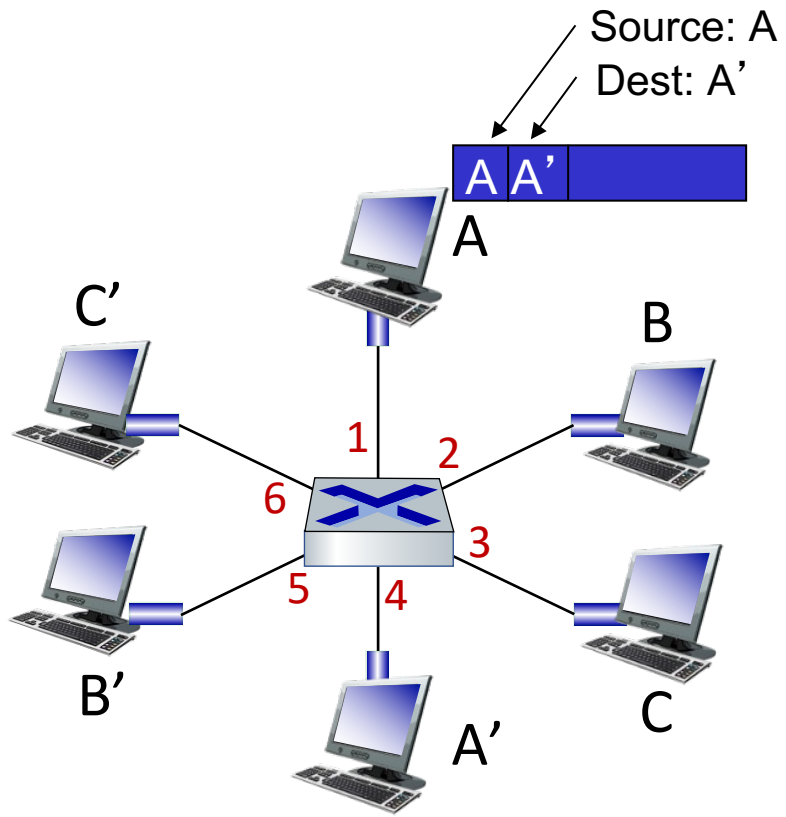



*Q:* how are entries created, maintained in switch table?

- something like a routing protocol?

# Switch: 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:

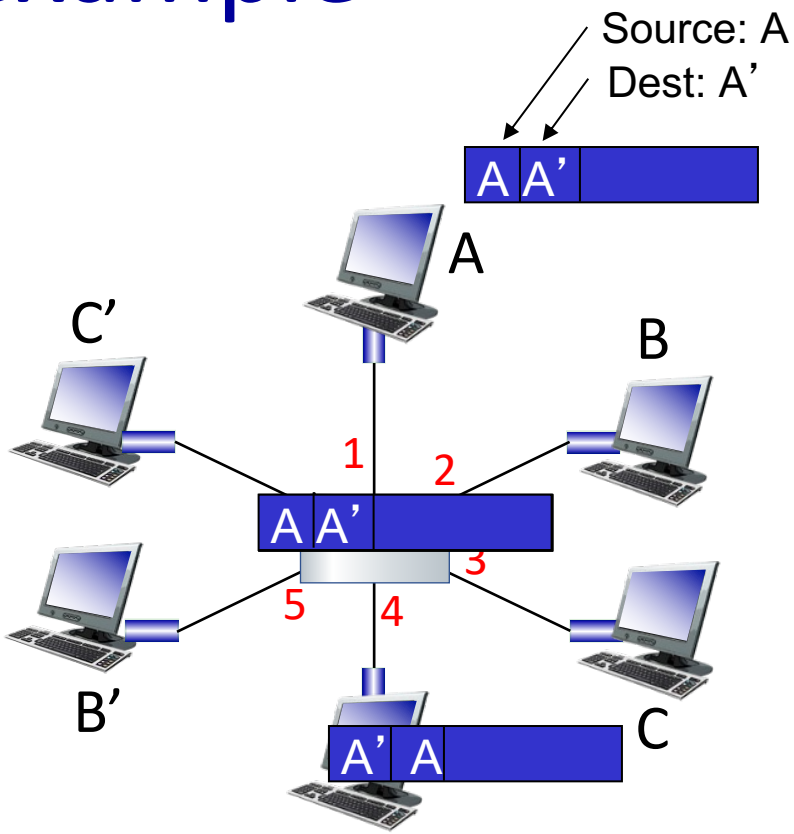
1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination

```
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 */
```

*If an entry age (time since last use) is equal to its TTL then the entry is evicted*

# Self-learning, forwarding: example

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

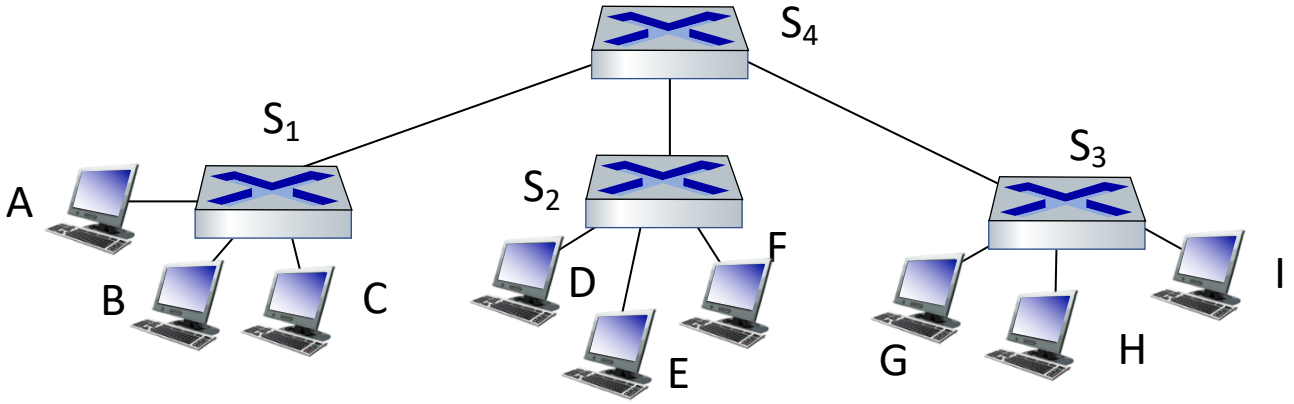



| MAC addr | interface | TTL |
|---|---|---|
| *A* | *1* | *60* |
| *A'* | *4* | *60* |

*switch table (initially empty)*

# Interconnecting switches

self-learning switches can be connected together:

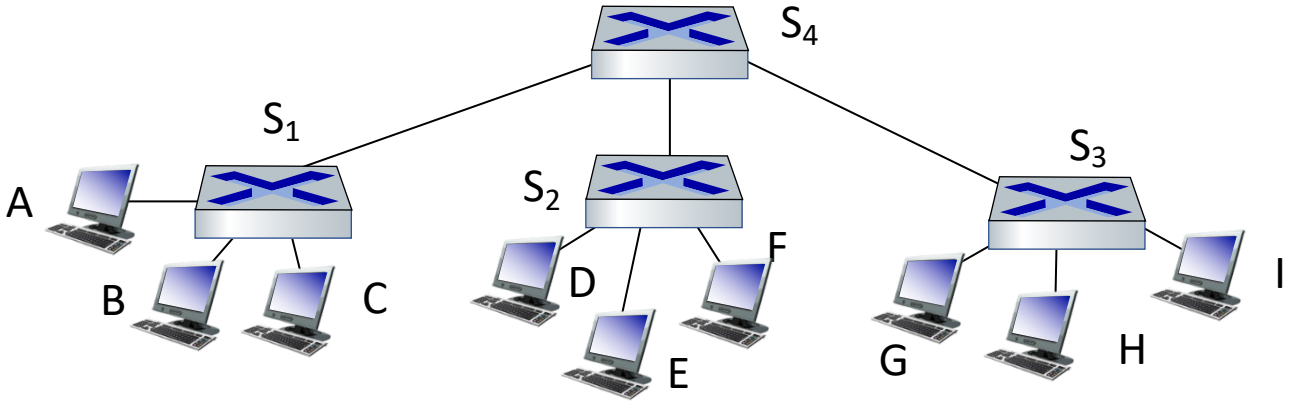



*Q:* sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

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

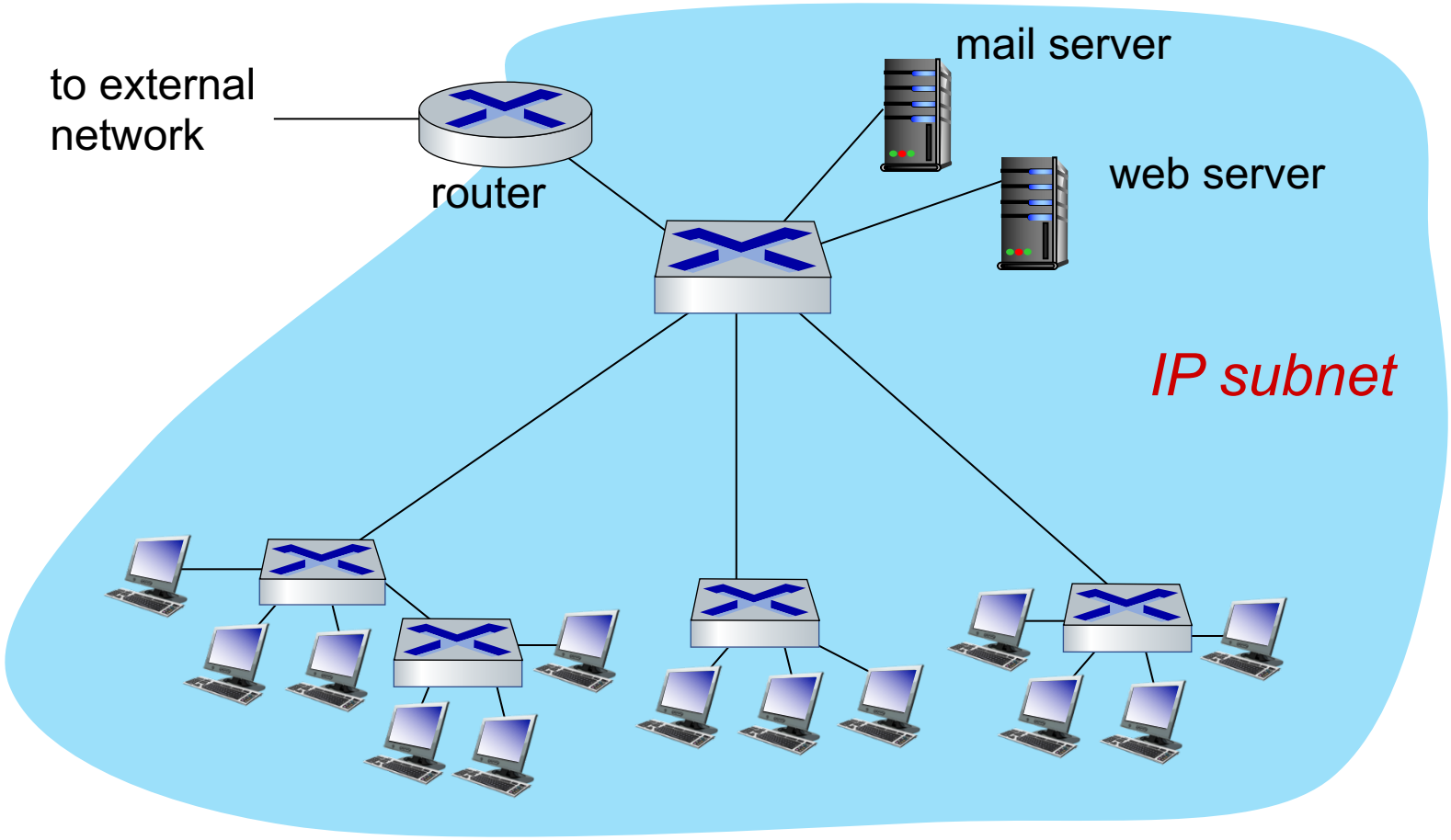
# Self-learning multi-switch example

Suppose C sends frame to I, I responds to C




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

# Small institutional network



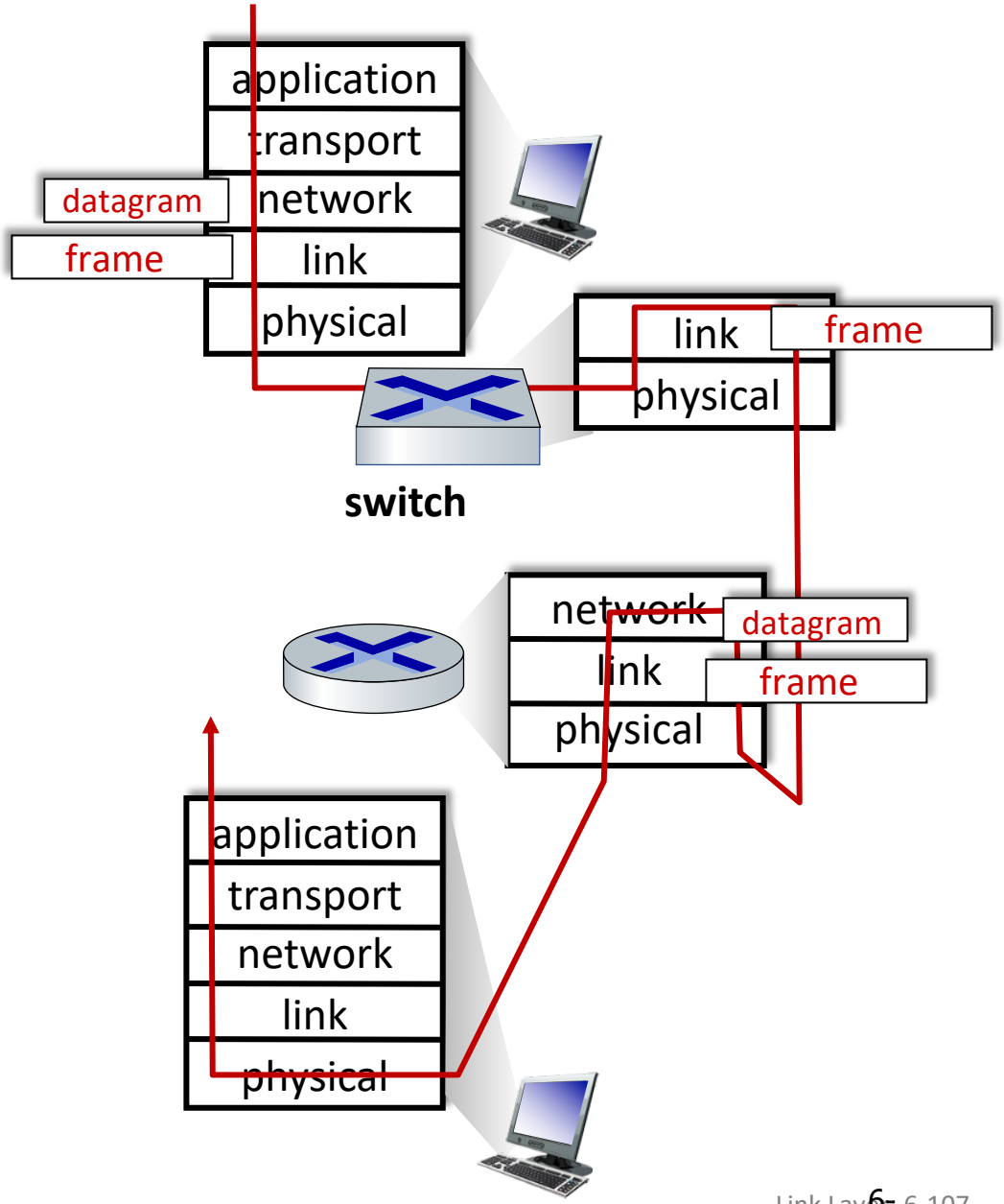
# Switches vs. routers

## 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



# Switches vs. routers

- Switches are plug and play
- Switches have less work to do, i.e., they are faster than routers
- Switches **must use spanning trees** to prevent cycles (only same interface packets can be dropped, no way to find cycles)
- Switches tables could be larger than routers'
- Routers may produce cycles only in case of wrong configurations (but IPv4 provides TTL field, and IP6 the hop limit field)
- Routers can be used in richer topologies, not just spanning trees
- Routers need IP address configuration (not plug and play)

# Switches vs. routers – when one or the other?

- Use switches in small networks (few hundreds of hosts with few LAN segments)
- Use routers in larger networks (thousands of hosts) that need more intelligent routing
- Note that switches may propagate broadcast storms

# Popular interconnection devices

| | Hubs | Routers | Switches |
|---|---|---|---|
| Traffic isolation | No | Yes | Yes |
| Plug and play | Yes | No | Yes |
| Optimal routing | No | Yes | No |

# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
- data center networking

- a day in the life of a web request

# Datacenter networks

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

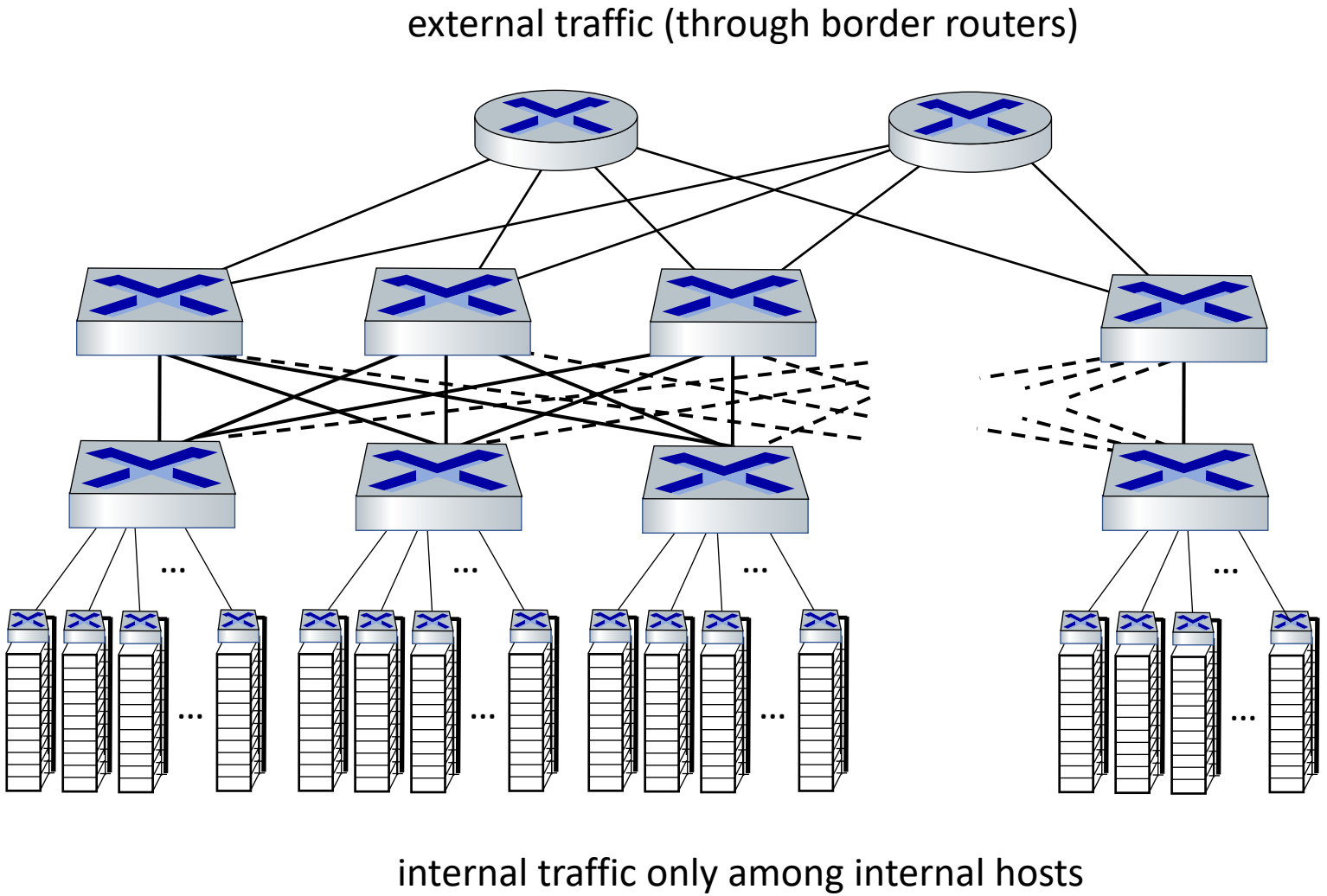
- e-business (e.g. Amazon)
- content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
- search engines, data mining (e.g., Google)

challenges:

- multiple applications, each serving massive numbers of clients
- 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

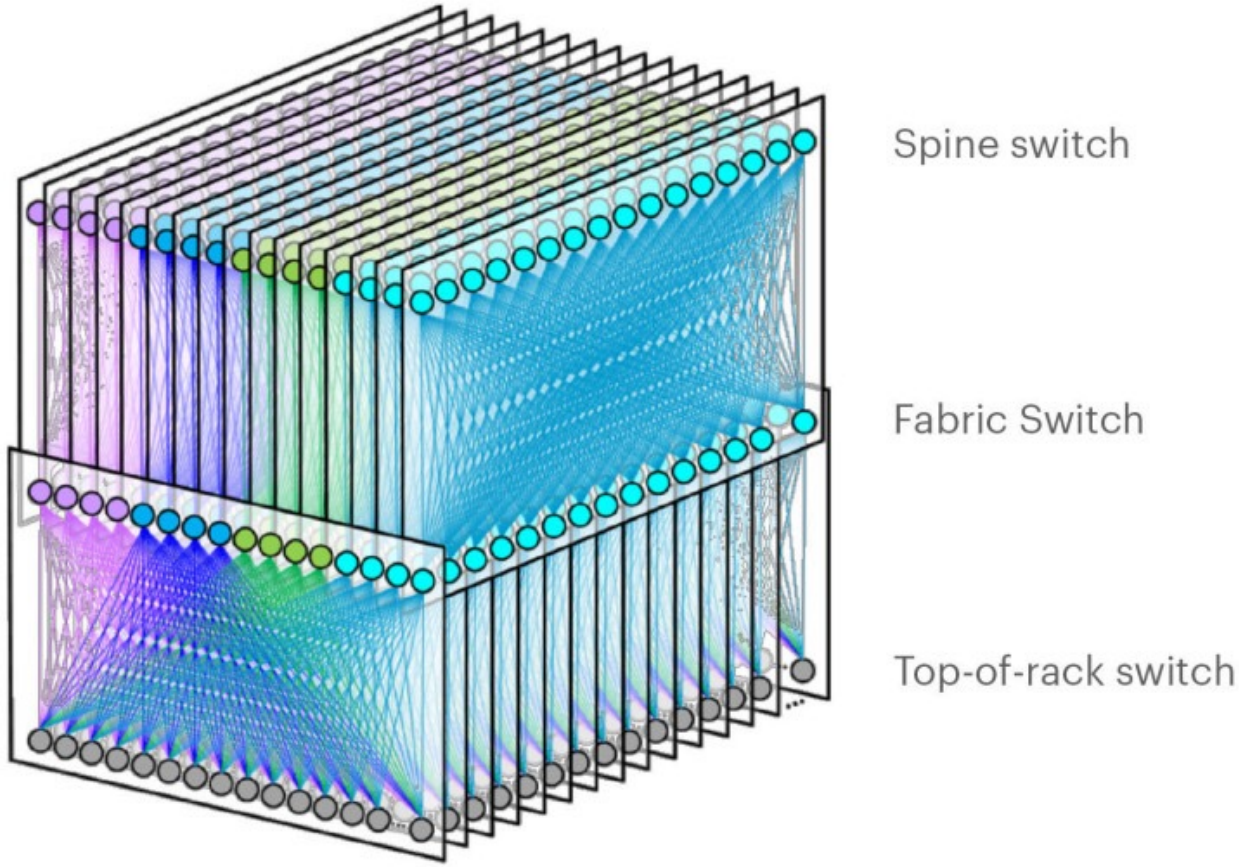
- one per rack
- 40-100Gbps Ethernet to blades

## Server racks

- 20- 40 server blades: hosts

# Datacenter networks: network elements

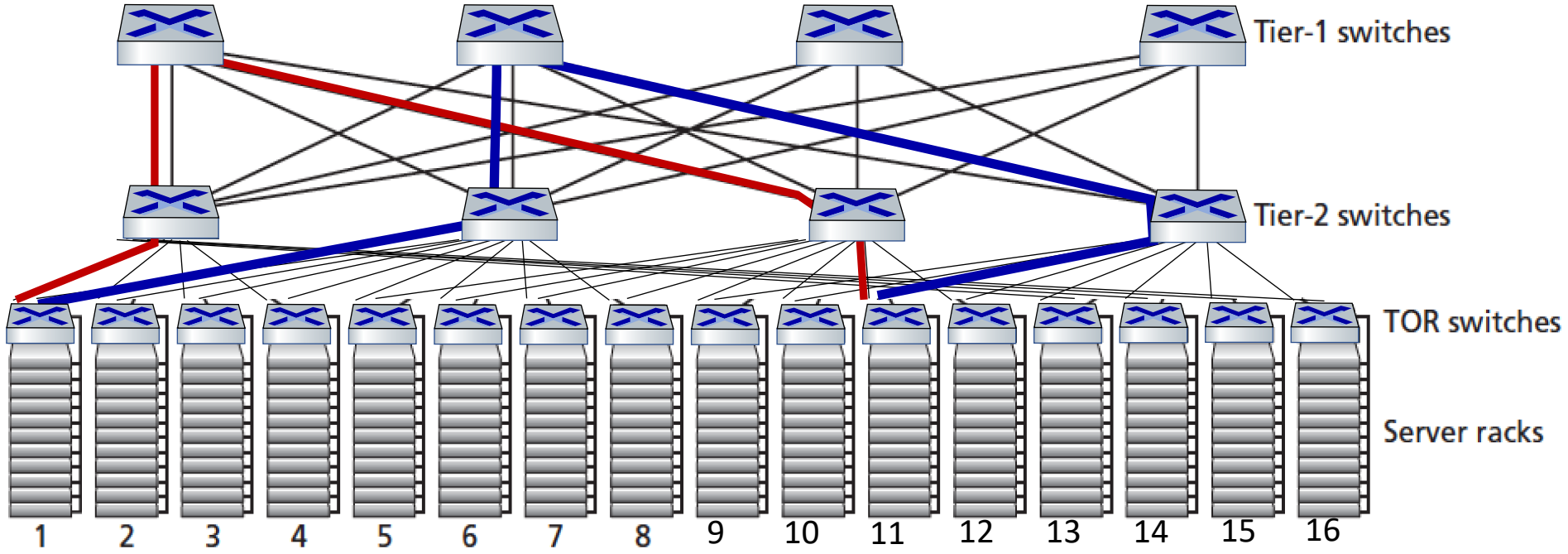
Facebook F16 data center network topology:




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

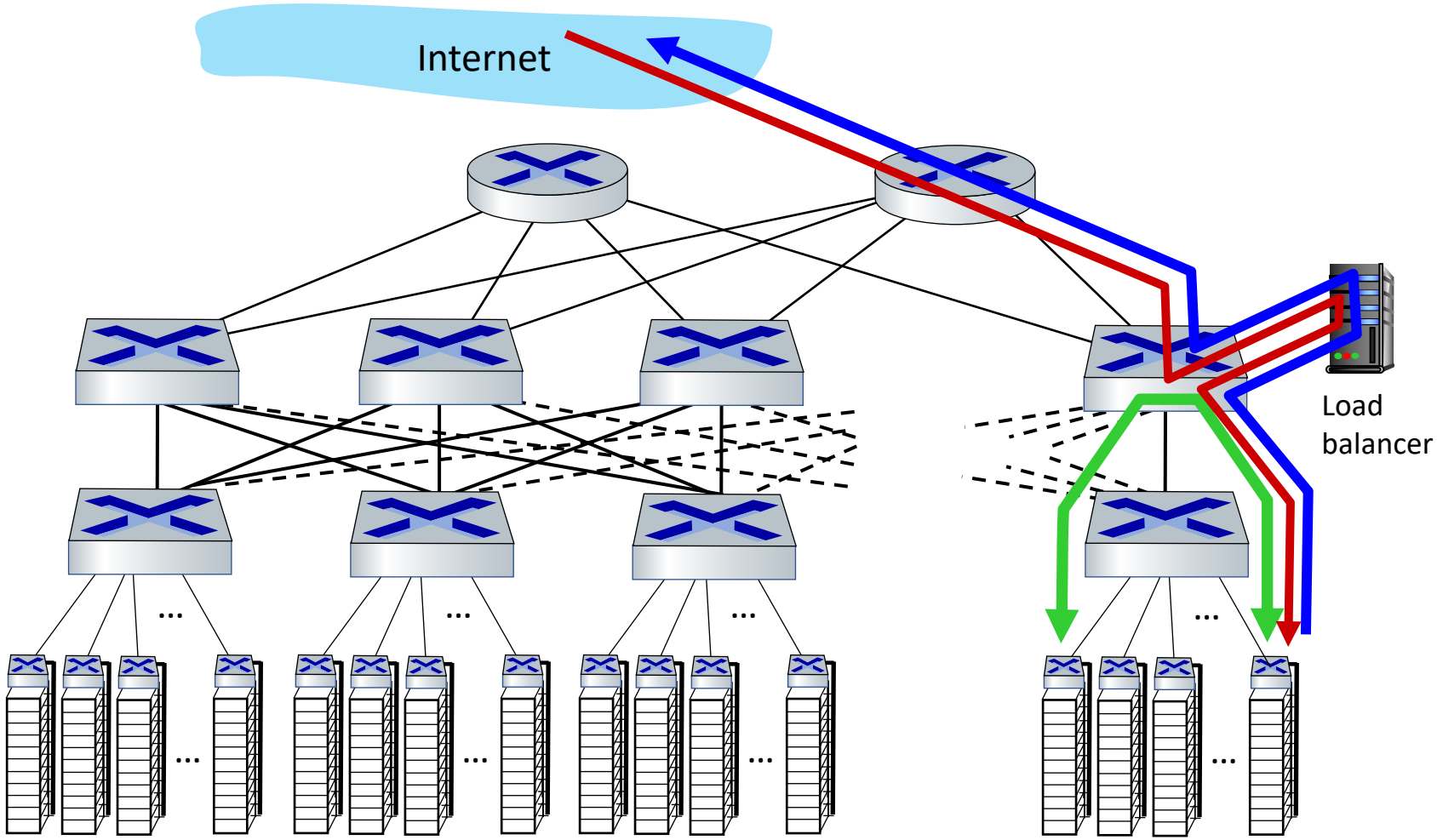
# Datacenter networks: multipath

- 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

# Datacenter networks: application-layer routing




## load balancer: application-layer routing

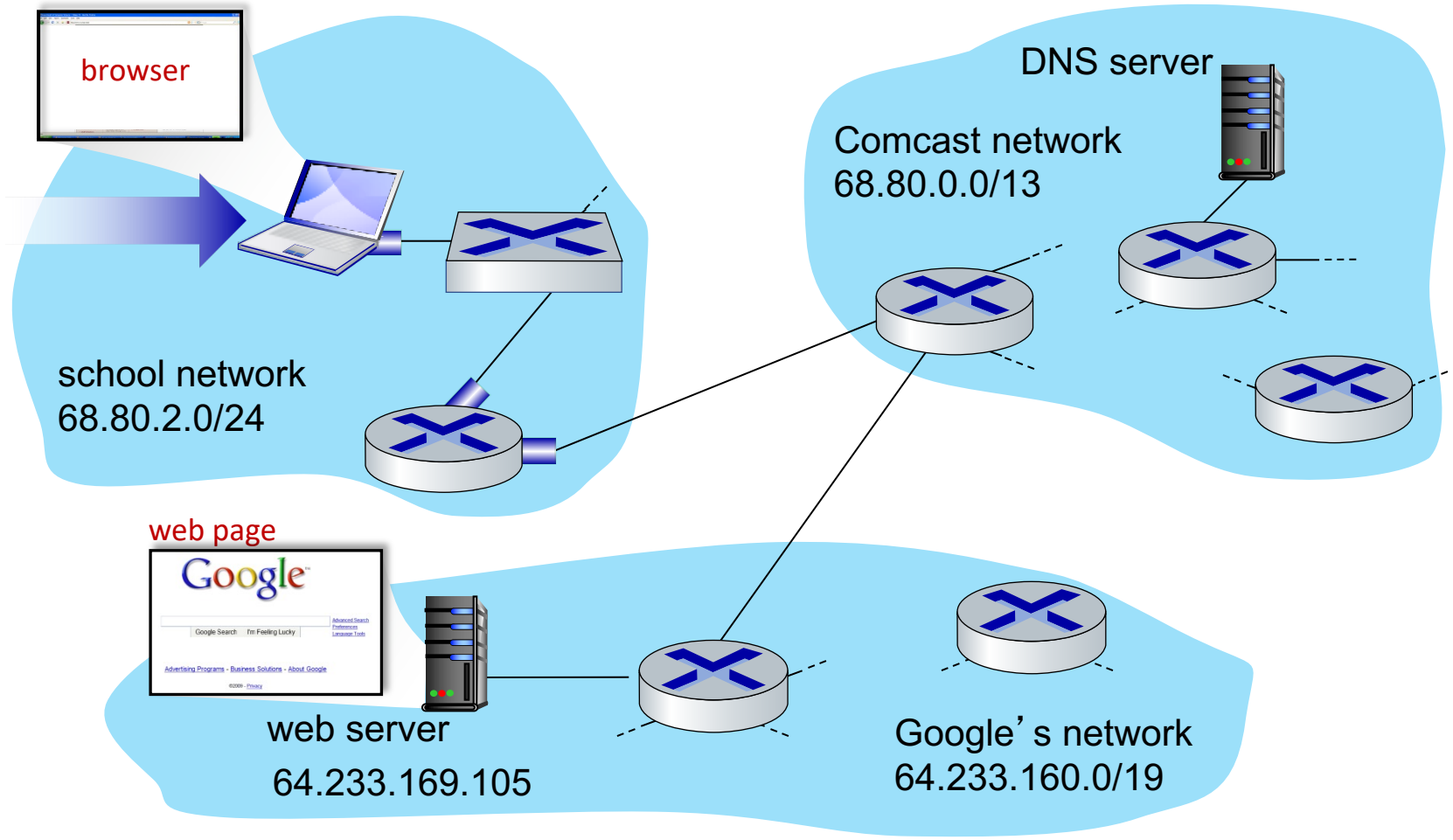
- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)

Load balancers are sometimes called **layer-4 switches when they consider TPC ports for balancing application load**
Load balancers provide **NAT services to provide a unique IP** to the external internet
Mostly based on SDN approach

# 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

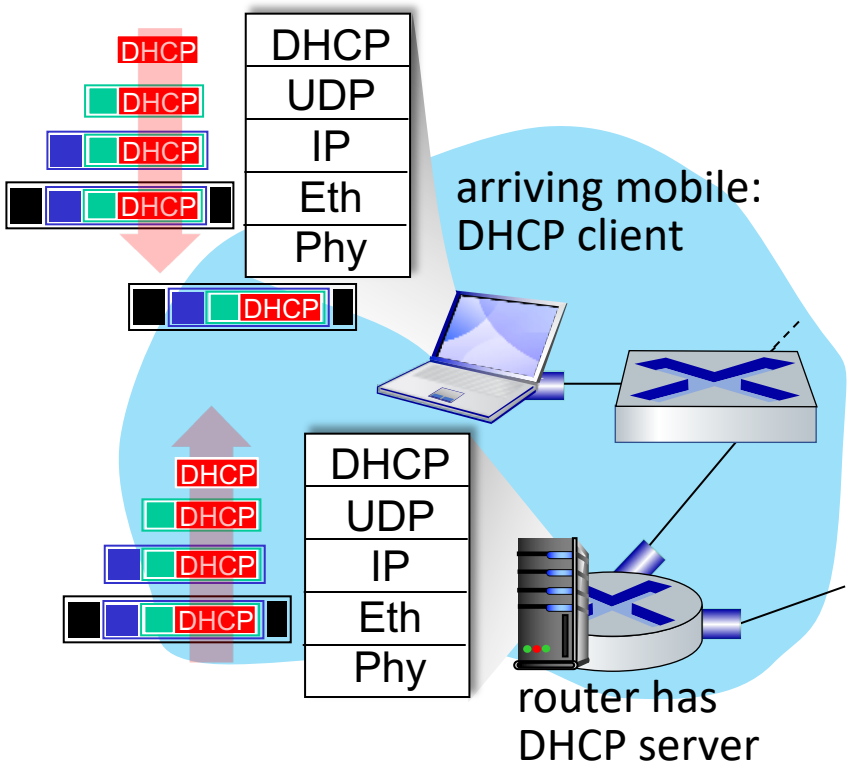



scenario:

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

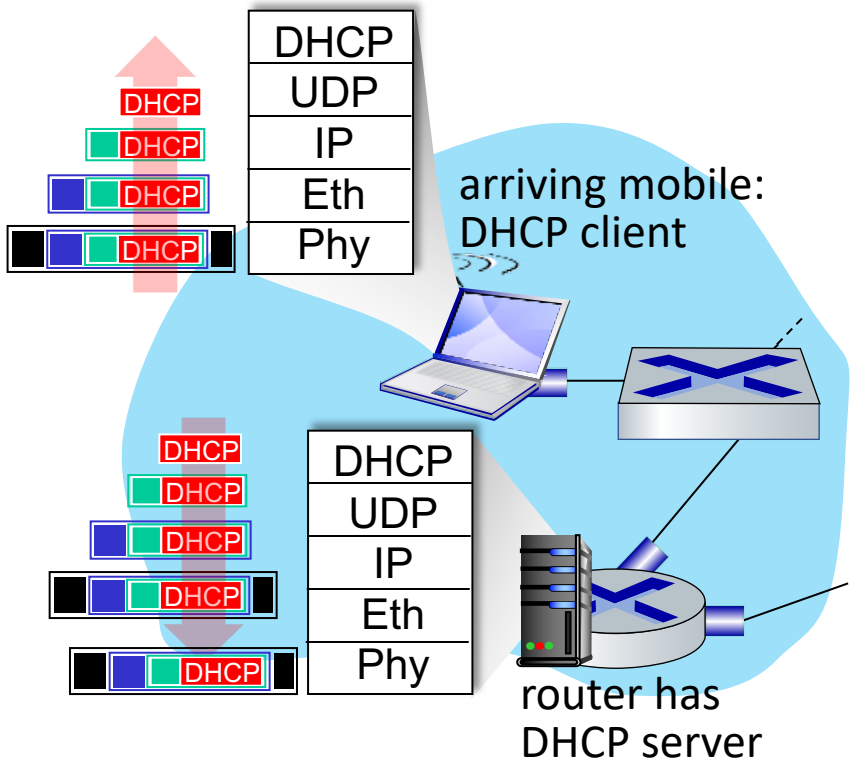
*Sounds simple!*

# 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 de-muxed to IP de-muxed, UDP de-muxed to DHCP

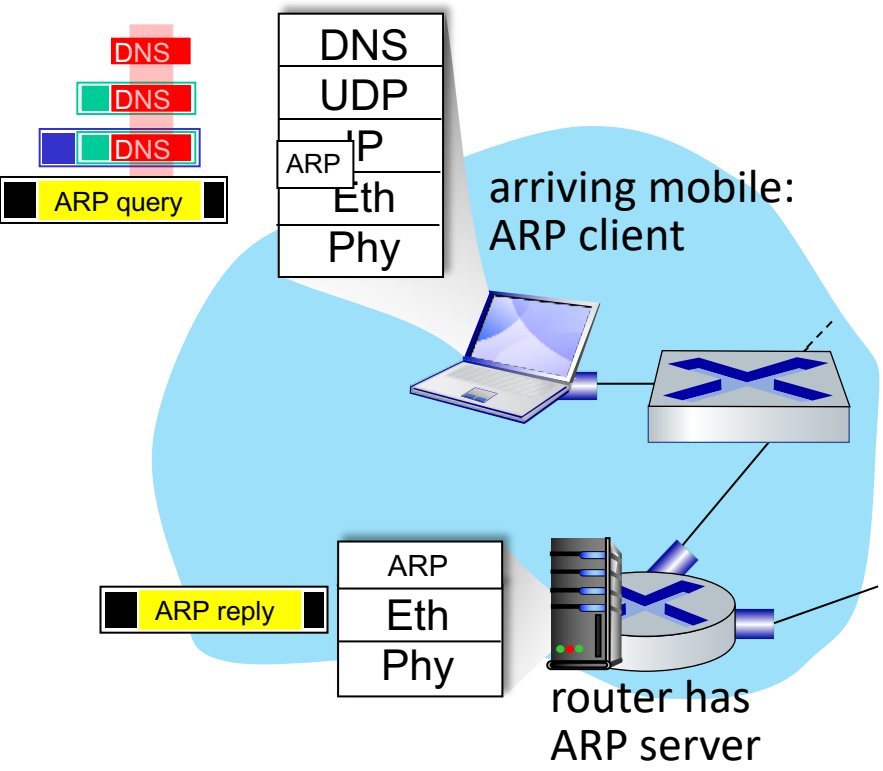
# A day in the life: connecting to the Internet




- 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

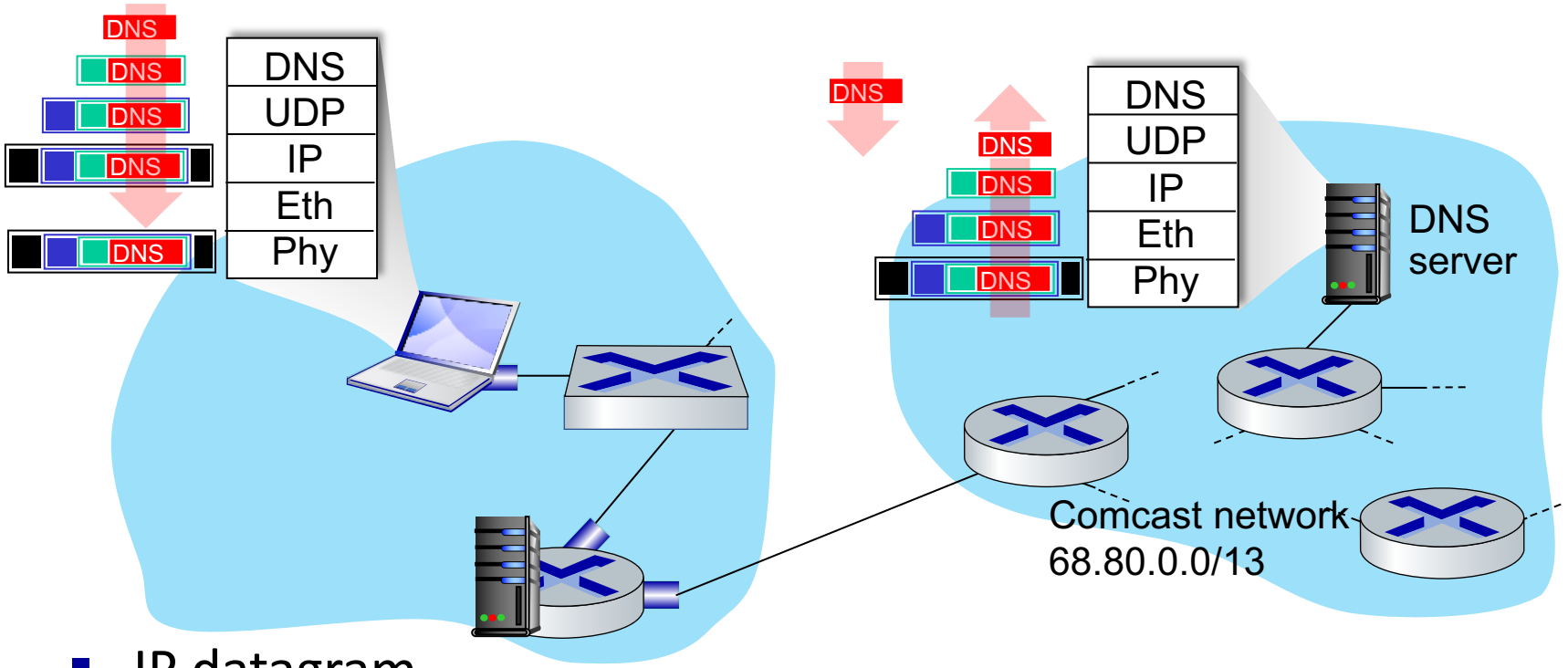
*Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router*

# 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

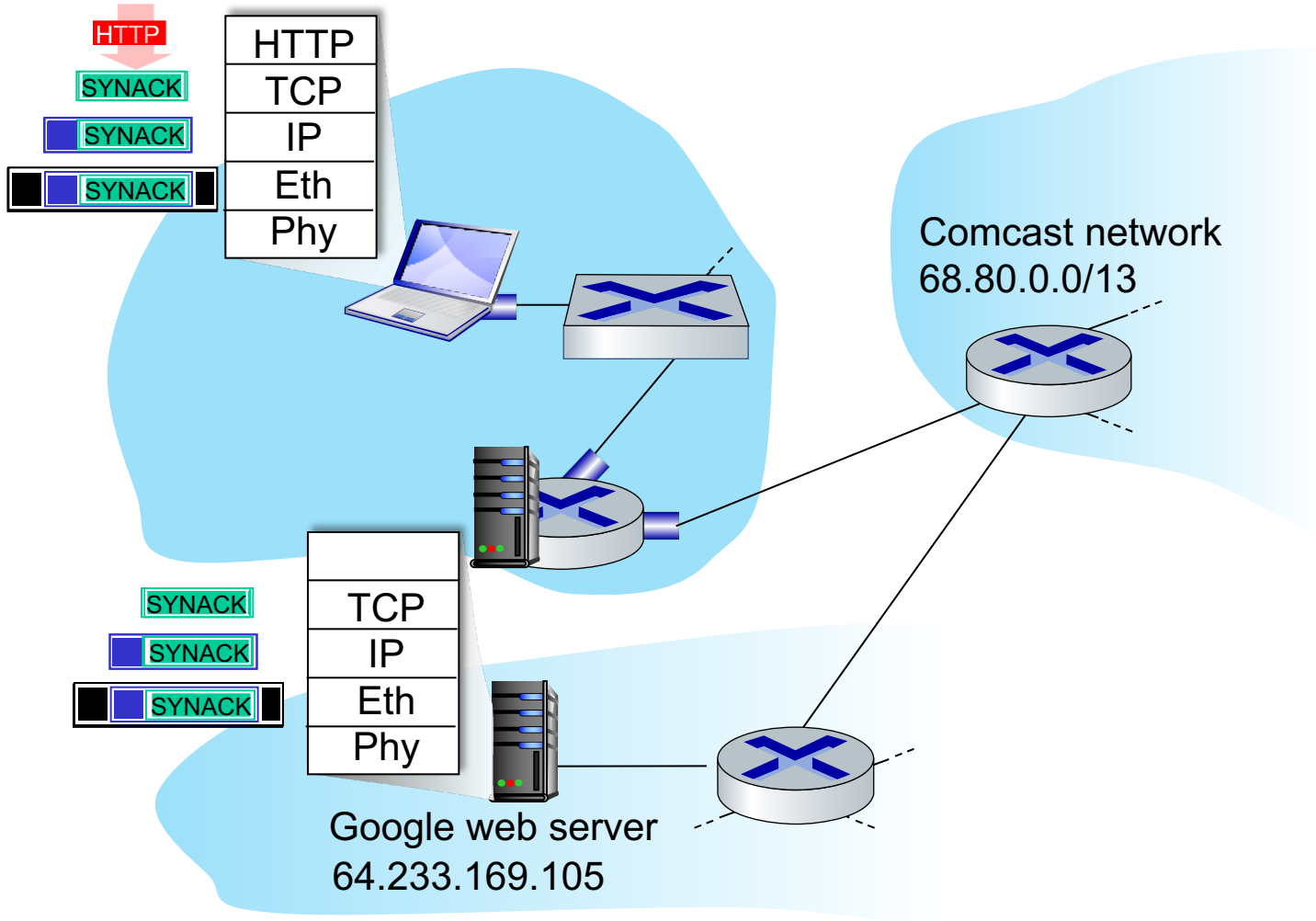



- de-muxed to DNS
- DNS replies to client with IP address of www.google.com

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

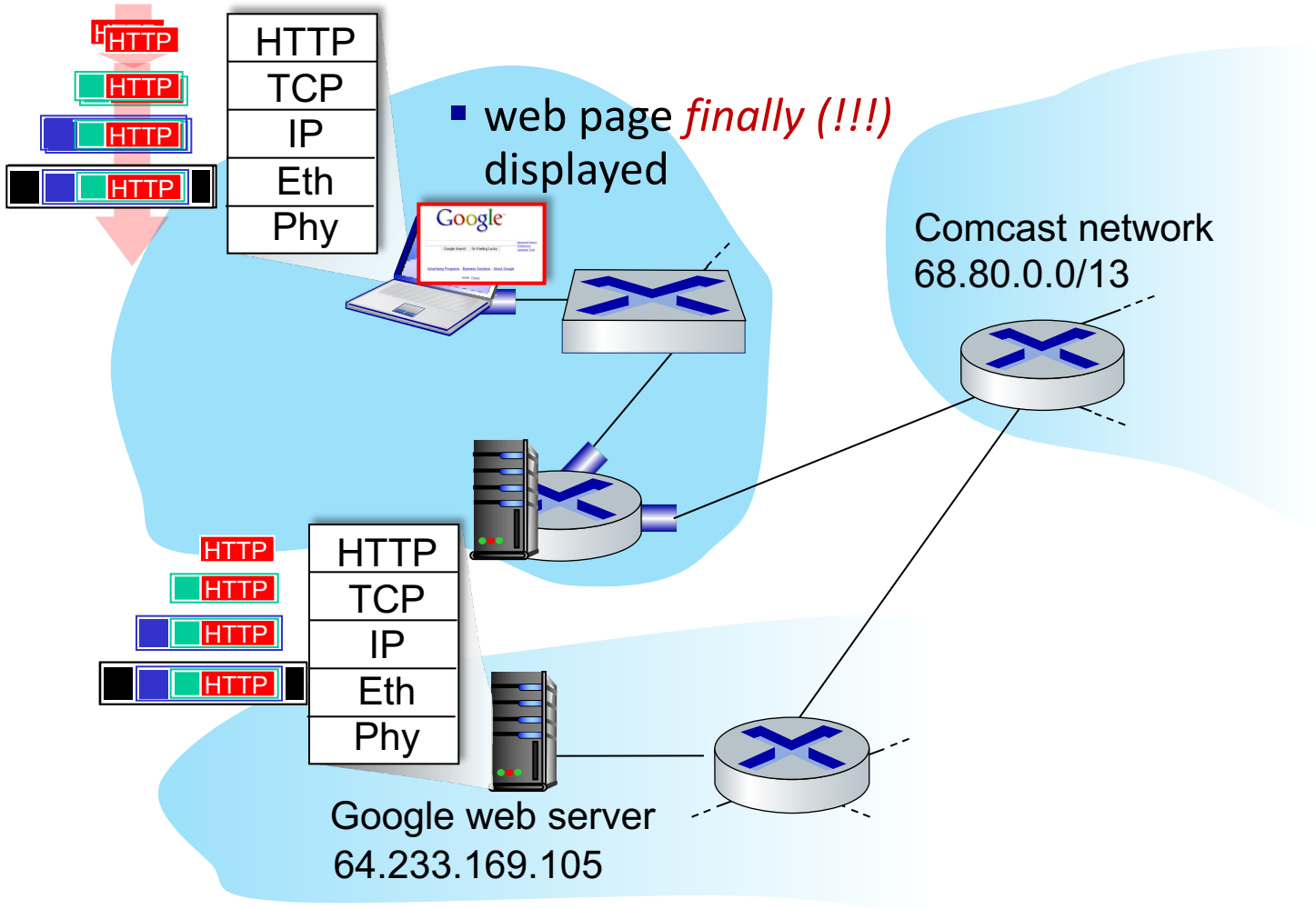
- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

# 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

# Chapter 6: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation, 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 6: let's take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice!

# Question 1

- Is it possible for two hosts belonging to two different LANs to have the same data link address (MAC)? What about their network address (IP)?

# Ex 2

- 4 stations belong to the same Ethernet hub-based LAN. The distances from the hub are 300m, 400m, 500m, and 700m. Which is the length of this network to be accounted when you calculate the (maximum) propagation delay of the network.

# Ex 3

- We have a network using the pure ALOHA protocol. The stations send frames of size 1000 bits, with a data rate of 1 Mbps. What is the vulnerable time of the network?

# Ex 4

- Assuming that the propagation delay in a broadcast network is 5 $\mu$s and that the trasmission time of a frame is 10 $\mu$s. How long does the first transmitted bit take to reach the destination? How long does the last bit take to reach the destination after the first bit arrival? How long is the network kept busy by this frame?

# Ex 5

- We have a noise which lasts for 2ms, in a network with data rate
  1. 1500 bps
  2. 12 Kbps
  3. 100 Kbps
  4. 100 Mbps

How would you evaluate the damage produced by the noise?

# Ex 6

- In a slotted ALOHA network consider two nodes A and B. Suppose that node A has more data to transmit than node B and that A's retransmission probability $p_A$ is greater than node B's retransmission probability $p_B$.

a) Provide a formula for node A's average throughput. What is the total efficiency (success in a slot) of the protocol with these two nodes?

b) If $p_A=2p_B$, is node A's average throughout twice as large as that of node B? Why or why not? If not, how can you choose $p_A$ and $p_B$ to make it happen?

c) In general, suppose there are N nodes, among which node A has retransmission probability 2p and all others have retransmission probability p. Provide expressions to compute the average thoughput of node A and of any other node.

# Ex 7

- In a Slotted Aloha network there are three stations: A,B, and C. In a time slot each station produces a frome with probabilities

$p_A$=0.2, $p_B$=0.3, and $p_C$=0.4.

What is the throughput of each station?

What is the network throughput?

# Ex 8 (problem 18 from KR)

P18. Suppose nodes A and B are on the same 10 Mbps broadcast channel, and the propagation delay between the two nodes is 325 bit times. Suppose CSMA/CD and Ethernet packets are used for this broadcast channel. Suppose node A begins transmitting a frame and, before it finishes, node B begins transmitting a frame. Can A finish transmitting before it detects that B has transmitted? Why or why not? If the answer is yes, then A incorrectly believes that its frame was successfully transmitted without a collision. *Hint*: Suppose at time $t = 0$ bits, A begins transmitting a frame. In the worst case, A transmits a minimum-sized frame of $512 + 64$ bit times. So A would finish transmitting the frame at $t = 512 + 64$ bit times. Thus, the answer is no, if B's signal reaches A before bit time $t = 512 + 64$ bits. In the worst case, when does B's signal reach A?

# Ex 9

- The transmission rate
  a) Denotes how fast a bit travels along a transmission channel
  b) Can be used to calculate the transmission delay
  c) Can be used to calculate the propagation delay
  d) Is used to calculate the vulnerable time in CSMA/CD

# Ex 9

- The transmission rate
  a) Denotes how fast a bit travels along a transmission channel (no, this depends on the speed)
  b) Can be used to calculate the transmission delay
  c) Can be used to calculate the propagation delay (wrong as the propagation delay depends on the network propagation speed, not on the rate)
  d) Is used to calculate the vulnerable time in CSMA/CD – wrong, in CSMA/CD it coincides with the propagation time, the trasmission time is used for slotted ALOHA and twice the tx time in the pure ALOHA)

# Ex 10

- Which of the following resource records are correct?
  a) [uniroma1.it ns1.garr.net. A]
  b) [it nameserver.cnr.it NS]
  c) [it 151.100.27.38 CNAME]
  d) [amazon.it pdns5.ultradns.info NS]
  e) [umb.edu ns2-umb.edu, A]

# Ex 10

- Which of the following resource records are correct?
  a) [uniroma1.it ns1.garr.net. A] (wrong, with an A record we expect to have an IP address)
  b) [it nameserver.cnr.it NS] (correct, for the domain it we are saying that the responsible DNS is nameserver.cnr.it)
  c) [it 151.100.27.38 CNAME] (wrong, we are not giving a canonical name of an alias, but we are giving an ip address)
  d) [amazon.it pdns5.ultradns.info NS] (correct, see before)
  e) [umb.edu ns2-umb.edu, A] (again, it is an A-record, it should contain an IP address]

# Ex. 11

- The retransmission timer of TCP:
  a) It coincides with the measured RTT
  b) It defines the waiting time of an ACK
  c) It is started for any segment sent within a sender window
  d) It is started only for the first packet of a sender window
  e) It considers the standard deviation of the RTT

# Ex. 11

- The retransmission timer of TCP:
  a) It coincides with the measured RTT - wrong
  b) It defines the waiting time of an ACK - correct
  c) It is started for any segment sent within a sender window
  d) It is started only for the first packet of a sender window
  e) It considers the standard deviation of the RTT

- Give a toy example of a network with K nodes and define the data structures that will be used by a DV or a LS routing algorithm. Motivate your response and provide clarifying examples of the algorithm execution.