

COMP 3331/9331: Computer Networks and Applications

Week 9
Data link Layer

Reading Guide: Chapter 6, Sections 6.2 – 6.4, 6.7



UNSW
SYDNEY

Australia's
Global
University

**“Having the lecture
recordings up – absolute
lifesaver when it comes
to revision”**

**Shape
OUR
Future**

Tell us about your experience and
shape the future of education at UNSW.



Click the link in Moodle now



*Complete your myExperience and shape
the future of education at UNSW.*

Click the  Experience link in Moodle

or login to myExperience.unsw.edu.au

(use z1234567@**ad.unsw.edu.au** to login)

The survey is confidential, your identity will never be released

Survey results are not released to teaching staff until after your results are published

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 Switched LANs

- addressing, ARP
- Ethernet
- switches

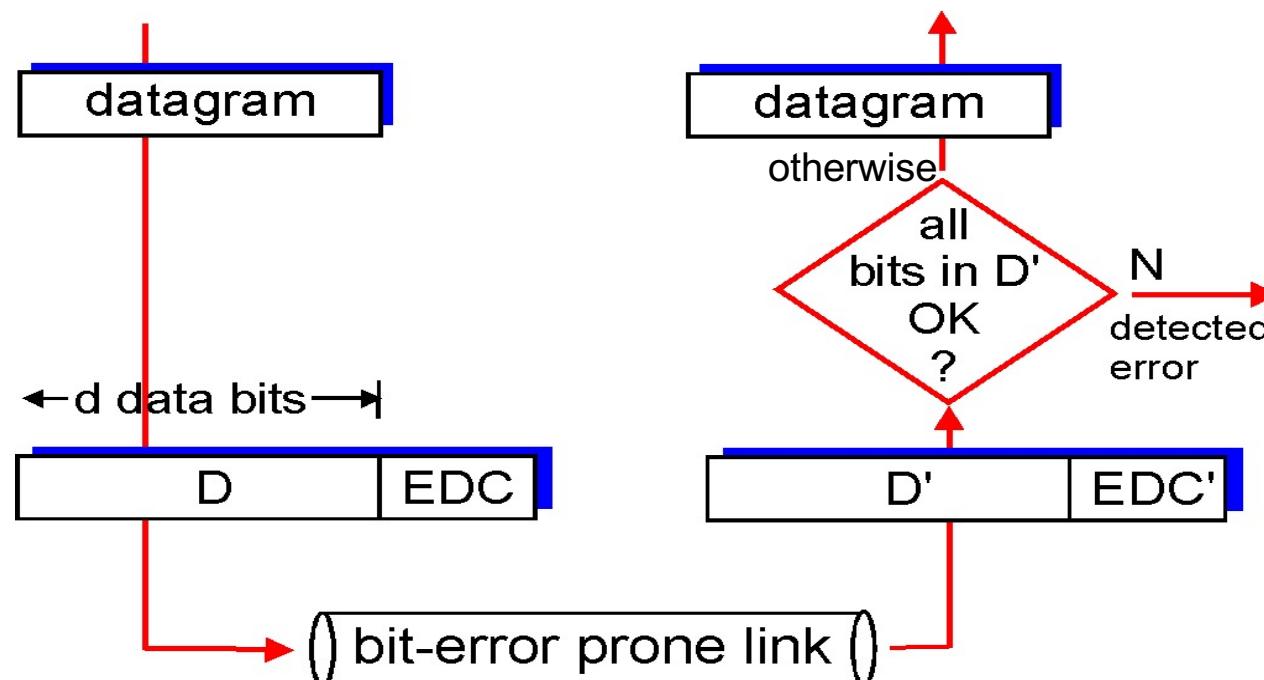
6.7 a day in the life of a
web request

Error detection

EDC= Error Detection and Correction bits (redundancy)

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

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



Error Detection

- Error coding
- Add check bits to the message bits to let some errors be detected and some be corrected
- How to structure the code to detect many errors with few check bits and modest computation?
- A simple code
 - Send two copies of the same message : **101101**
 - Error if the copies are different : **101100**
 - How many errors can it correct? 0
 - How many errors can it detect? At most 3
 - How many errors will make it fail? Specific 2-bit errors
 - What is the overhead? 100% (wrt original message)

Simple Parity - Sender

Example uses even parity

- Suppose you want to send the message:
 - 001011011011000110010
- For every d bits (e.g., $d = 7$), add a parity bit:
 - 1 if the number of one's is odd
 - 0 if the number of one's is even

Message chunk	Parity bit
0010110	1
1101100	0
0110010	1

– 00101101101100001100101

Simple Parity - Receiver

- For each block of size d :
 - Count the number of 1's and compare with following parity bit.
- If an odd number of bits get flipped, we'll detect it (can't do much to correct it).
- Cost: One extra bit for every d
 - In this example, 21 \rightarrow 24 bits.

Two-Dimensional Parity

Example uses even parity

- Suppose you want to send the same message:
 - 001011011011000110010
- Add an extra parity byte, compute parity on “columns” too.
- Can detect 1, 2, 3-bit (and some 4-bit) errors

	Message chunk	Parity bit
	0010110	1
	1101100	0
	0110010	1
Parity byte:	1001000	0

Forward Error Correction

Example uses even parity

- With two-dimensional parity, we can even *correct* single-bit errors.

The diagram illustrates a 4x9 grid of binary bits. A blue arrow labeled "Parity bits" points to the bottom row, which is labeled "Parity byte". The grid contains the following data:

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

A blue circle highlights the value "0" in the fifth column of the second row. A blue arrow labeled "Parity bits" points to the bottom row, which is labeled "Parity byte". The "Parity byte" row contains the values: 1, 1, 1, 1, 1, 1, 0, 0.

Exactly one bit has been flipped. Which is it?

In practice

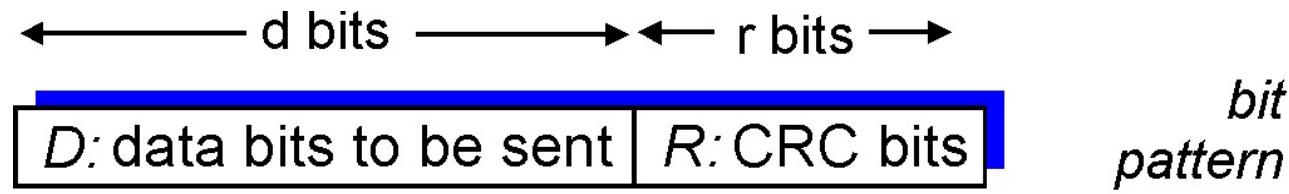
- Bit errors occur in bursts.
- We're willing to trade computational complexity for space efficiency.
 - Make the detection routine more complex, to detect error bursts, without tons of extra data
- Insight: We need hardware to interface with the network, do the computation there!

Error Detection and Correction

- **Checksum**
 - Sum up data in N-bit words
 - Internet Checksum uses 16-bit words
- **How well checksum works?**
 - How many errors can it detect/correct?
- **What have we gained as compared to parity bit?**
 - Can now detect all burst errors up to 16

Cyclic redundancy check

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



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

mathematical formula

Cyclic redundancy check

- Sender operation
 - Extend D data bits with R zeros
 - Divide by generator G
 - Keep remainder, ignore quotient
 - Adjust R check bits by the remainder
- Receiver Procedure
 - Divide received frame by G and check for zero remainder

A Note on Modulo-2 Arithmetic

- All calculations are modulo-2 arithmetic
- No carries or borrows in subtraction
- Addition and subtraction are identical and both are equivalent to XOR
 - $1011 \text{ XOR } 0101 = 1110$
 - $1011 - 0101 = 1110$
 - $1011 + 0101 = 1110$
- Multiplication by 2^k is essentially a left shift by k bits
 - $1011 \times 2^2 = 101100$

CRC example

want:

$$D \cdot 2^r \text{ XOR } R = nG$$

equivalently:

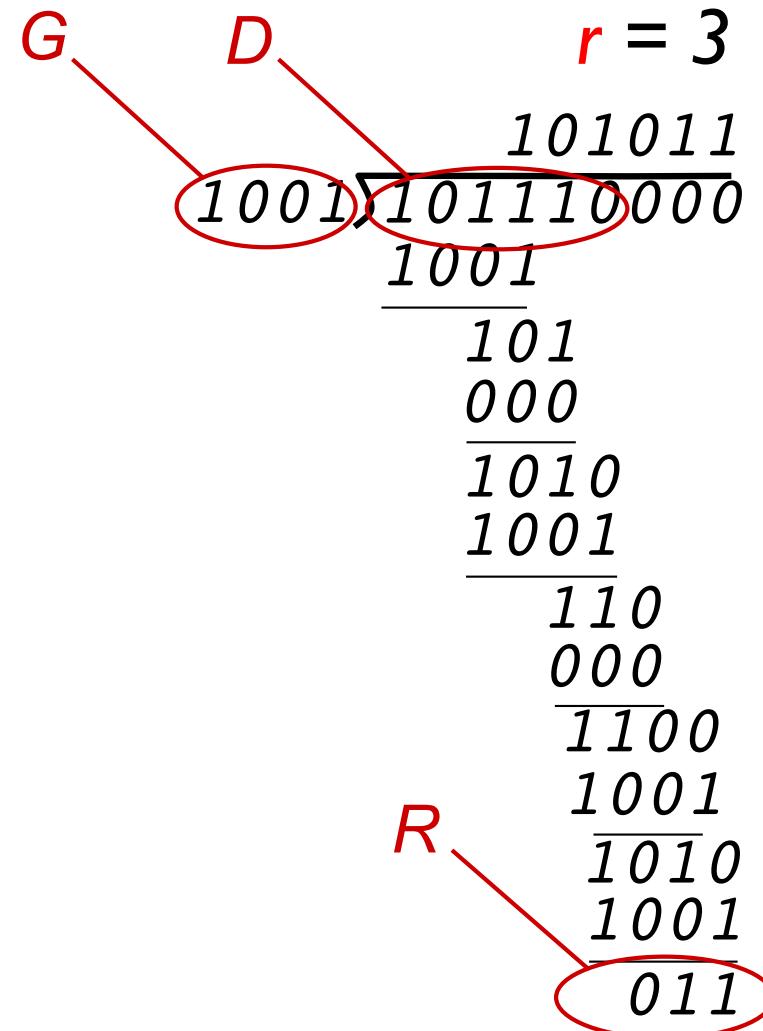
$$D \cdot 2^r = nG \text{ XOR } R$$

equivalently:

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

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

At the sender



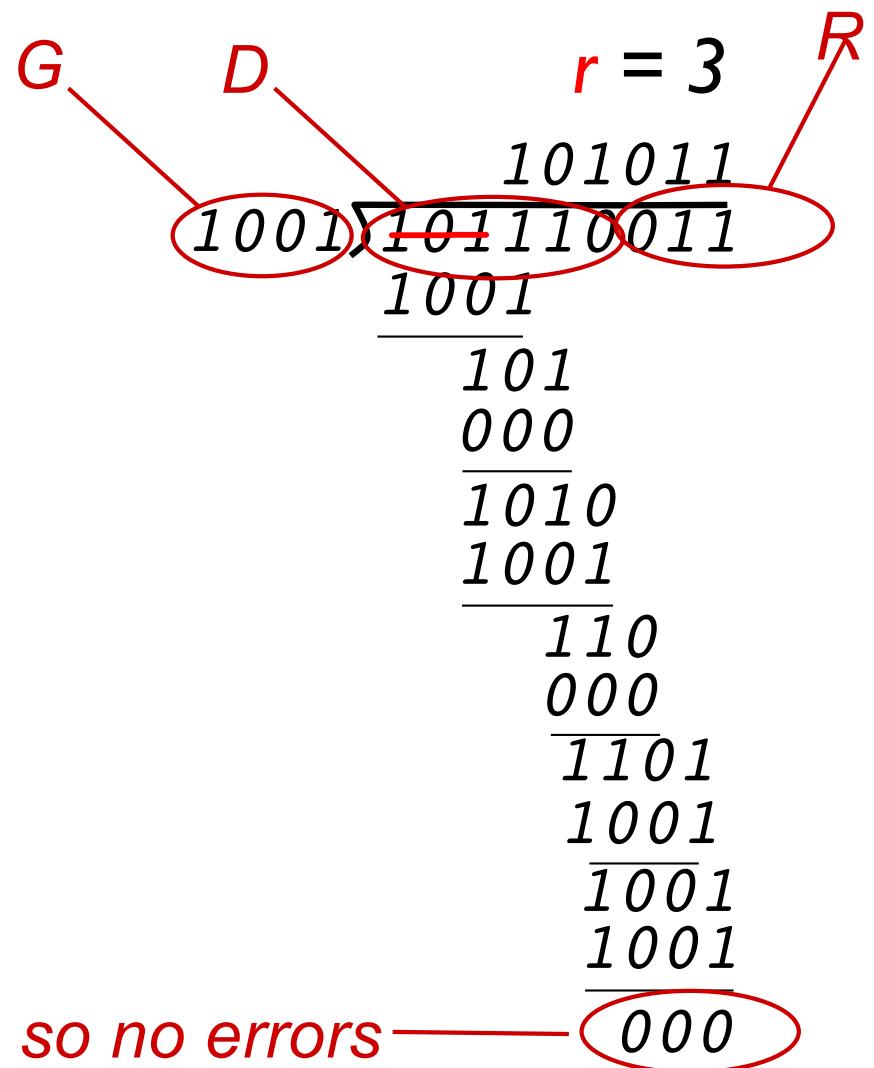
Sender sends $\langle D, R \rangle$ into the channel

CRC example

Receiver divides the received frame and divides by G and checks if the remainder is zero.

In this example, there are no errors, so the receiver receives $\langle D, R \rangle$ (from previous slide)

At the receiver



Quiz: Error Detection/Correction



- ❖ Can these schemes respectively correct any bit errors: Internet checksums, two-dimensional parity, cyclic redundancy check (CRC)
 - a) Yes, No, No
 - b) No, Yes, Yes
 - c) No, Yes, No
 - d) No, No, Yes
 - e) No, No, No

ANSWER: C

Link layer, LANs: outline

6.1 introduction, services

6.7 a day in the life of a
web request

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 Switched LANs

- addressing, ARP
- Ethernet
- switches

Multiple access links, protocols

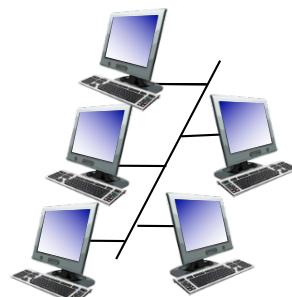
two types of “links”:

- ❖ **point-to-point**

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

- ❖ ***broadcast (shared wire or medium)***

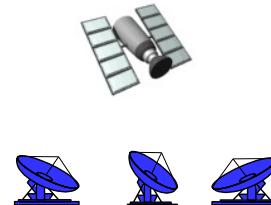
- old-fashioned Ethernet
- upstream HFC
- 802.11 wireless LAN



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)



humans at a
cocktail party
(shared air, acoustical)

Multiple access protocols

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

multiple access protocol

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

An ideal multiple access protocol

given: broadcast channel of rate R bps

requirements:

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

MAC protocols: taxonomy

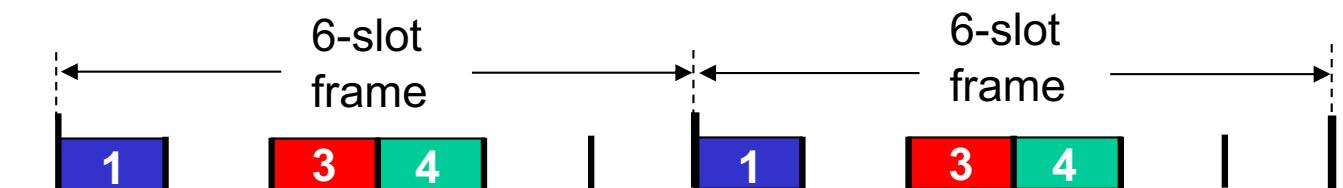
three broad classes:

- ❖ *channel partitioning*
 - divide channel into smaller “pieces” (time slots, frequency, code)
 - allocate piece to node for exclusive use
- ❖ *random access*
 - channel not divided, allow collisions
 - “recover” from collisions
- ❖ *“taking turns”*
 - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

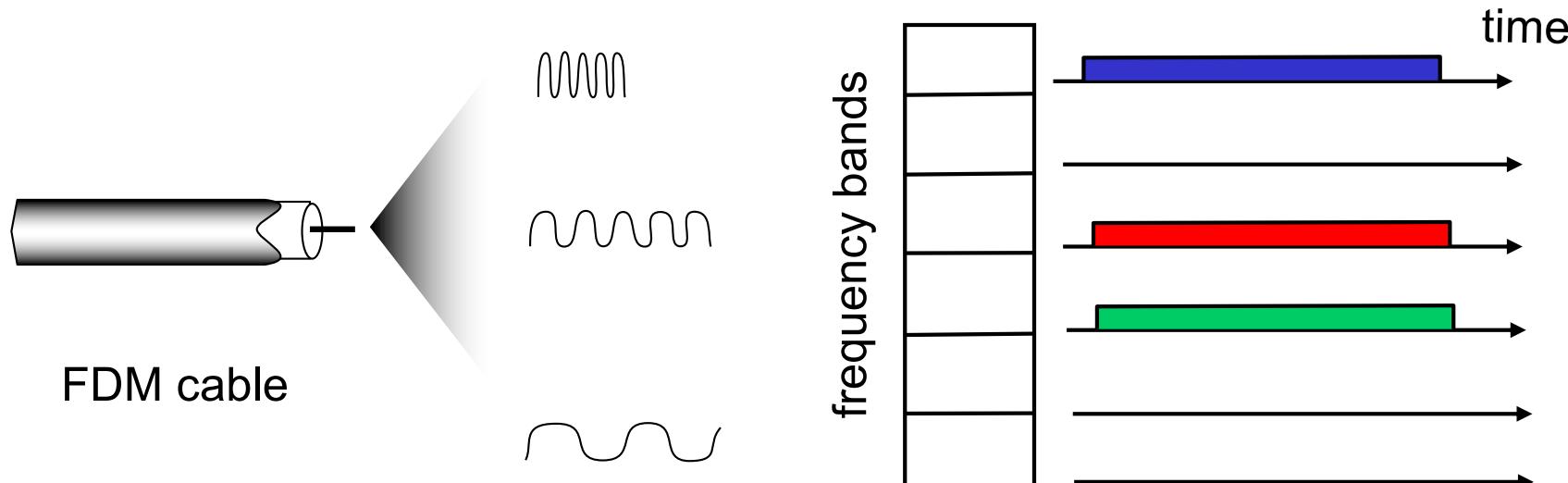
- ❖ access to channel in "rounds"
- ❖ each station gets fixed length slot (length = pkt trans time) in each round
- ❖ unused slots go idle
- ❖ example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

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



Quiz: Does channel partitioning satisfy ideal properties ?



1. if only 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 (fairness)
3. fully decentralized:
 - no synchronization of clocks, slots
 - no special node to coordinate transmissions
4. simple

- A. 0
B. 1
C. 2
D. 3
E. 4

ANSWER: C

2 and 4 from above are satisfied

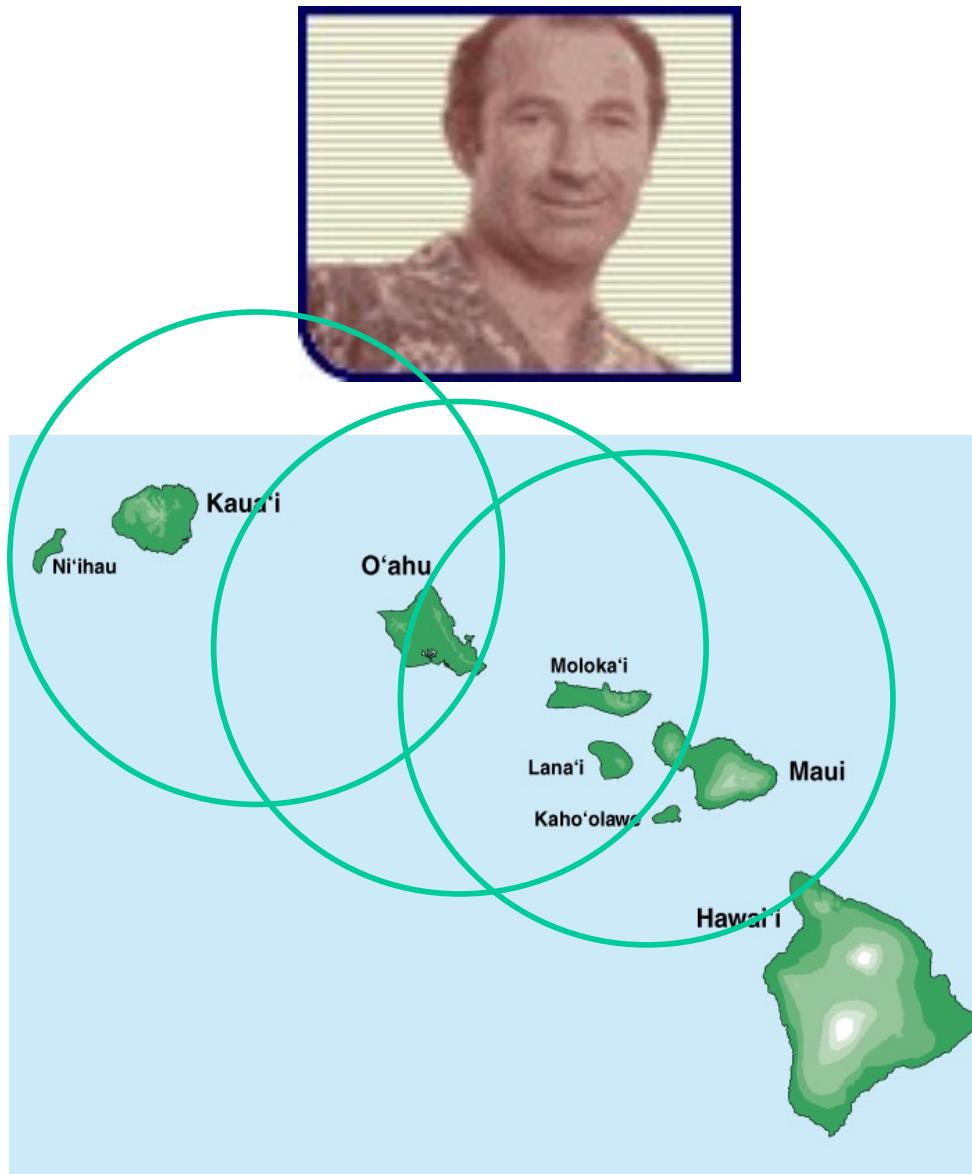
*Assuming $M=N$ (no of nodes on the network)
if $M < N$, then 2 is not satisfied)*

(Which ones?)

Random access protocols

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

Where it all Started: AlohaNet



- ❖ Norm Abramson left Stanford in 1970 (***so he could surf!***)
- ❖ Set up first data communication system for Hawaiian islands
- ❖ Central hub at U. Hawaii, Oahu

Slotted ALOHA

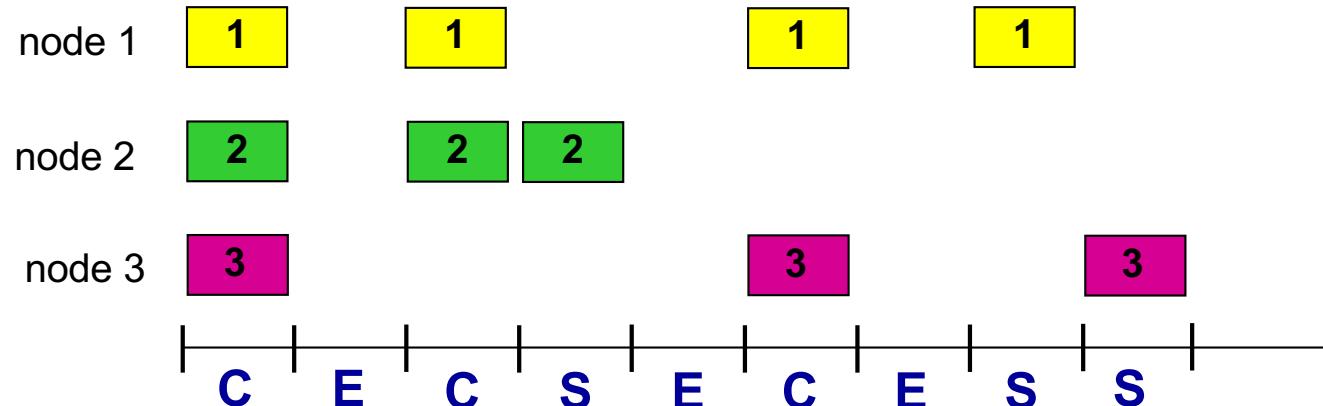
assumptions:

- ❖ 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 slot, all nodes detect collision

operation:

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

Slotted ALOHA



Pros:

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

Cons:

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

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 \underline{p}
- ❖ prob that given node has success in a slot = $p(1-p)^{\underline{N-1}}$
- ❖ prob that *any* node has a success = $\underline{Np(1-p)^{N-1}}$

- ❖ max efficiency: find $\underline{p^*}$ that maximizes $Np(1-p)^{N-1}$
- ❖ for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

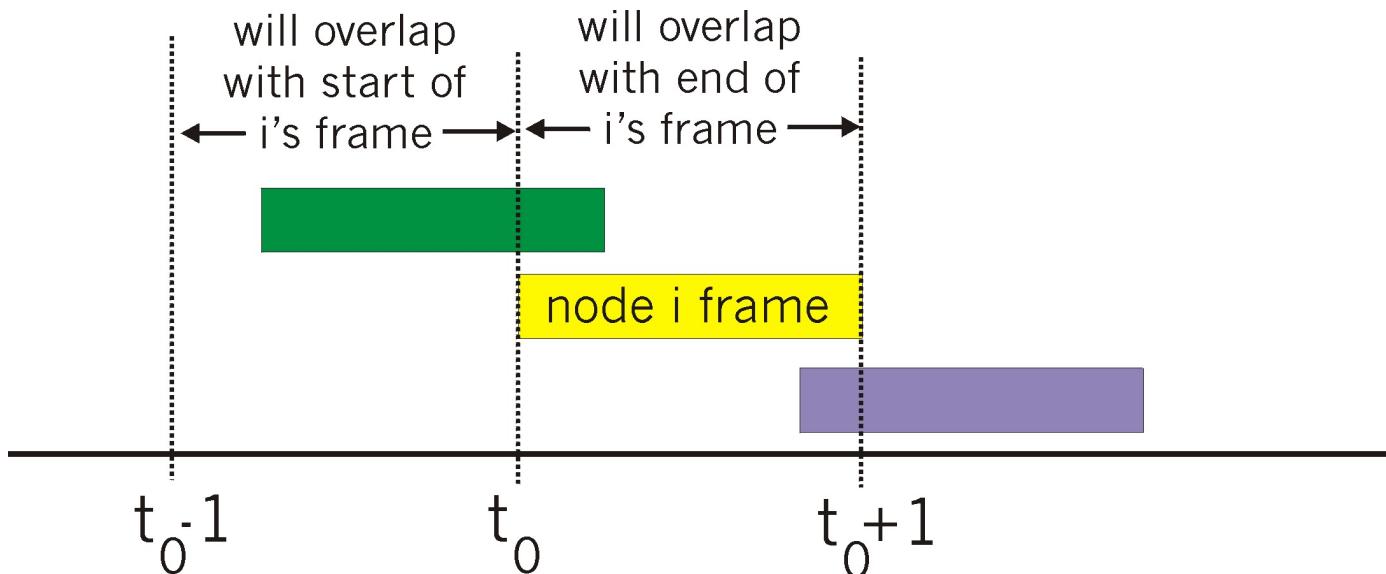
$$\text{max efficiency} = \underline{1/e = .37}$$

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

!

Pure (unslotted) ALOHA

- ❖ unslotted Aloha: simpler, no synchronization
- ❖ when frame first arrives
 - transmit immediately
- ❖ collision probability increases:
 - frame sent at t_0 collides with other frames sent in $[t_0 - l, t_0 + l]$



Pure ALOHA efficiency

$$\begin{aligned} P(\text{success by given node}) &= P(\text{node transmits}) \cdot \\ &\quad P(\text{no other node transmits in } [t_0-l, t_0]) \cdot \\ &\quad P(\text{no other node transmits in } [t_0, t_0+l]) \\ &= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} \\ &= p \cdot (1-p)^{2(N-1)} \\ \dots \text{ choosing optimum } p \text{ and then letting } n &\rightarrow \infty \\ &= l/(2e) = .18 \end{aligned}$$

even worse than slotted Aloha!

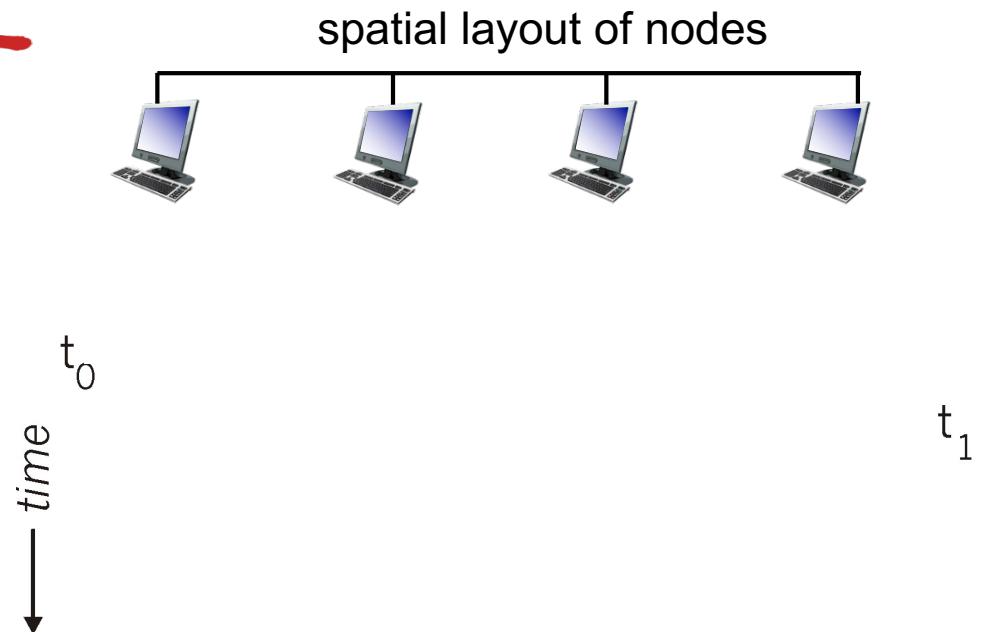
CSMA (carrier sense multiple access)

CSMA: listen before transmit:

- if channel sensed idle: transmit entire frame
 - ❖ if channel sensed busy, defer transmission
 - ❖ human analogy: don't interrupt others!
 - ❖ Does this eliminate all collisions?
 - No, because of nonzero propagation delay

CSMA collisions

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



*CSMA reduces but does not
eliminate collisions*

Biggest remaining problem?

Collisions can be detected earlier!

•

CSMA/CD (collision detection)

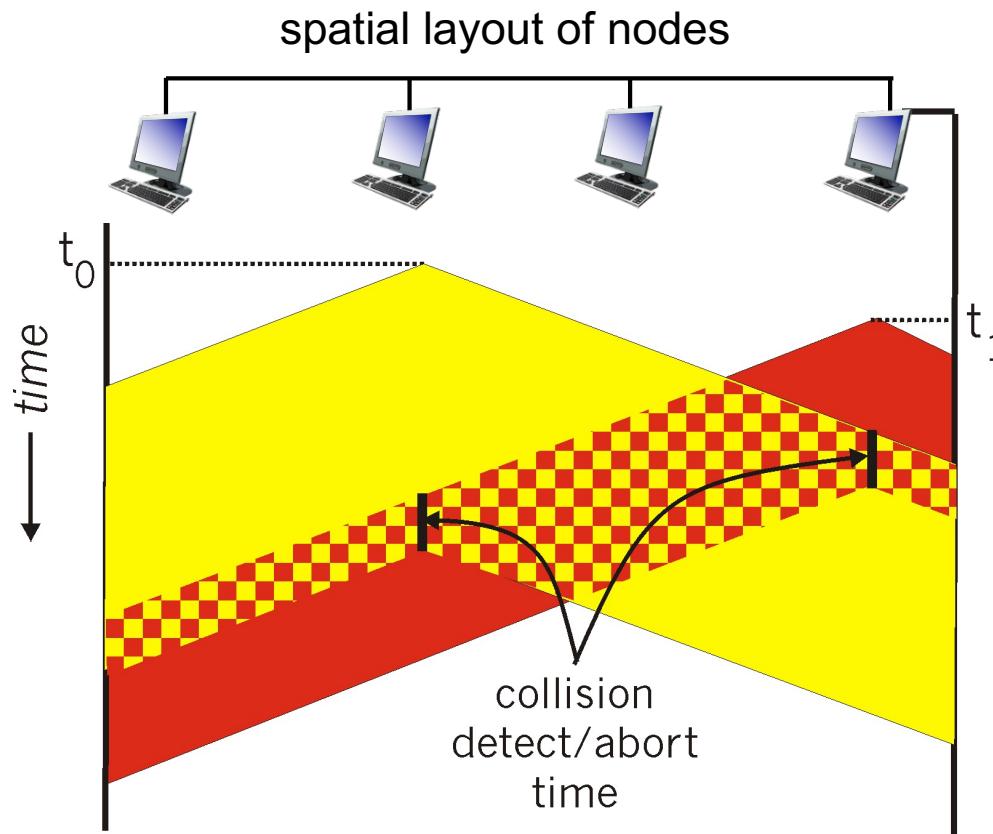
CSMA/CD: carrier sensing, deferral as in CSMA

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

CSMA/CD (collision detection)

Note: for this to work, need restrictions on minimum frame size and maximum distance.

Why?



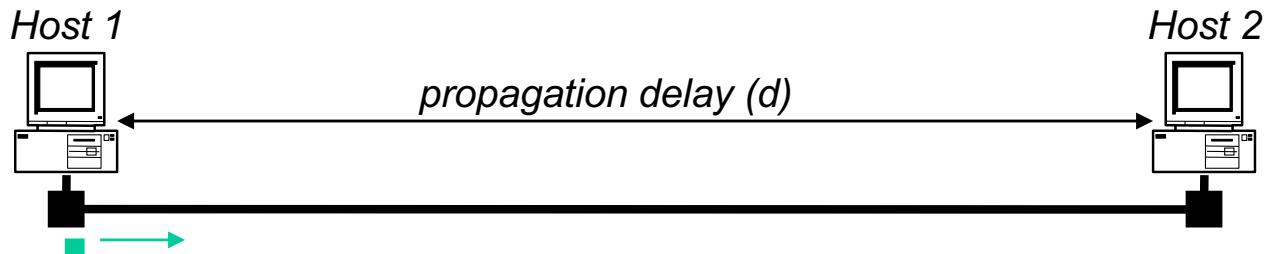
http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/csmacd/csmacd.html

Minimum Packet Size

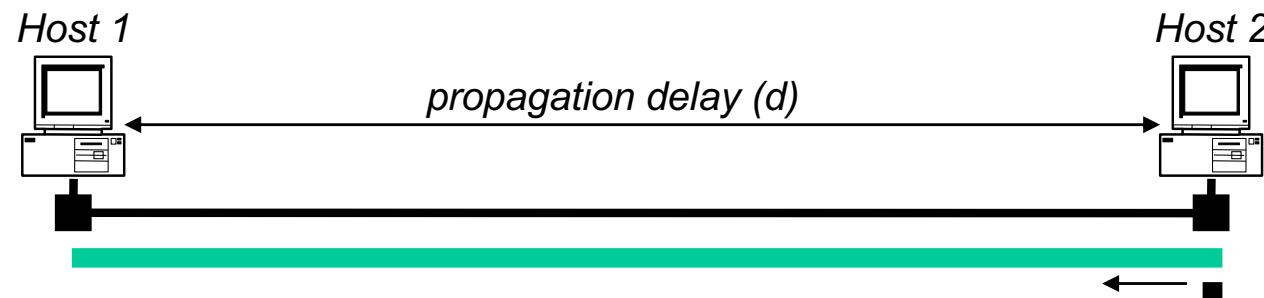
- ❖ Why enforce a minimum packet size?
- ❖ Give a host enough time to detect collisions
- ❖ In Ethernet, minimum packet size = 64 bytes (two 6-byte addresses, 2-byte type, 4-byte CRC, and 46 bytes of data)
- ❖ If host has less than 46 bytes to send, the adaptor pads (adds) bytes to make it 46 bytes
- ❖ What is the relationship between minimum packet size and the length of the LAN?

Limits on CSMA/CD Network Length

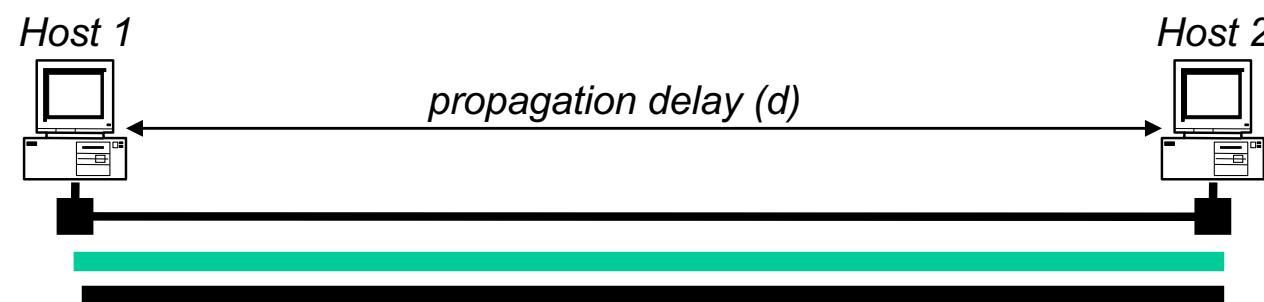
a) Time = t ; Host 1 starts to send frame



b) Time = $t + d$; Host 2 starts to send a frame, just before it hears from host 1's frame



c) Time = $t + 2*d$; Host 1 hears Host 2's frame → **detects collision**



$$\text{min_frame_size}/\text{bandwidth} = 2 * \text{LAN length}/\text{propagation speed}$$

$$\begin{aligned} \text{For } 10 \text{ Mbps LAN, LAN length} &= (\text{min_frame_size}) * (\text{propagation_speed}) / (2 * \text{bandwidth}) = \\ &= (8 * 64B) * (2 * 10^8 \text{ mps}) / (2 * 10^7 \text{ bps}) = 5120 \text{ m approx} \end{aligned}$$

What about 100 mbps? 1 gbps? 10 gbps?

Ethernet CSMA/CD algorithm

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

Quiz: Does CSMA/CD satisfy ideal properties ?



1. if only 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 (fairness)
3. fully decentralized:
 - no synchronization of clocks, slots
 - no special node to coordinate transmissions
4. simple

www.zetings.com/salil

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

Answer: D

1, 3 and 4 are satisfied

2 is not satisfied as bandwidth is wasted due to collisions when multiple nodes are transmitting (neglect the overheads for channel sensing)

(Which ones?)

“Taking turns” MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, I/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

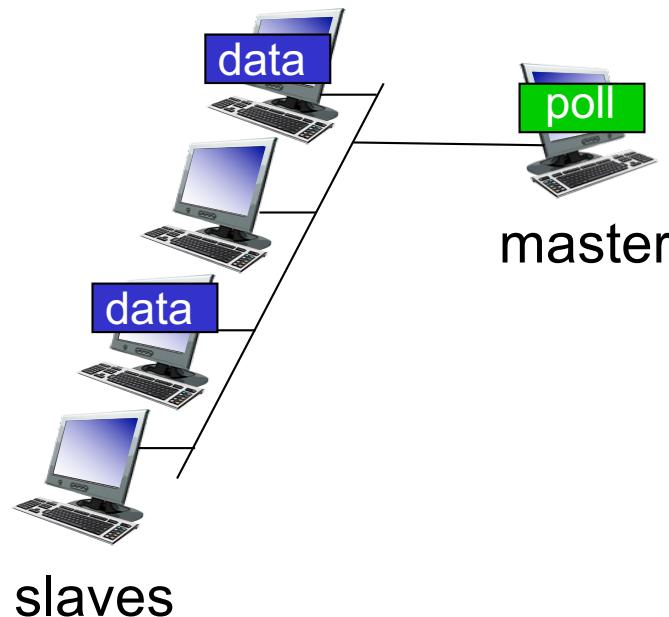
“taking turns” protocols

look for best of both worlds!

“Taking turns” MAC protocols

polling:

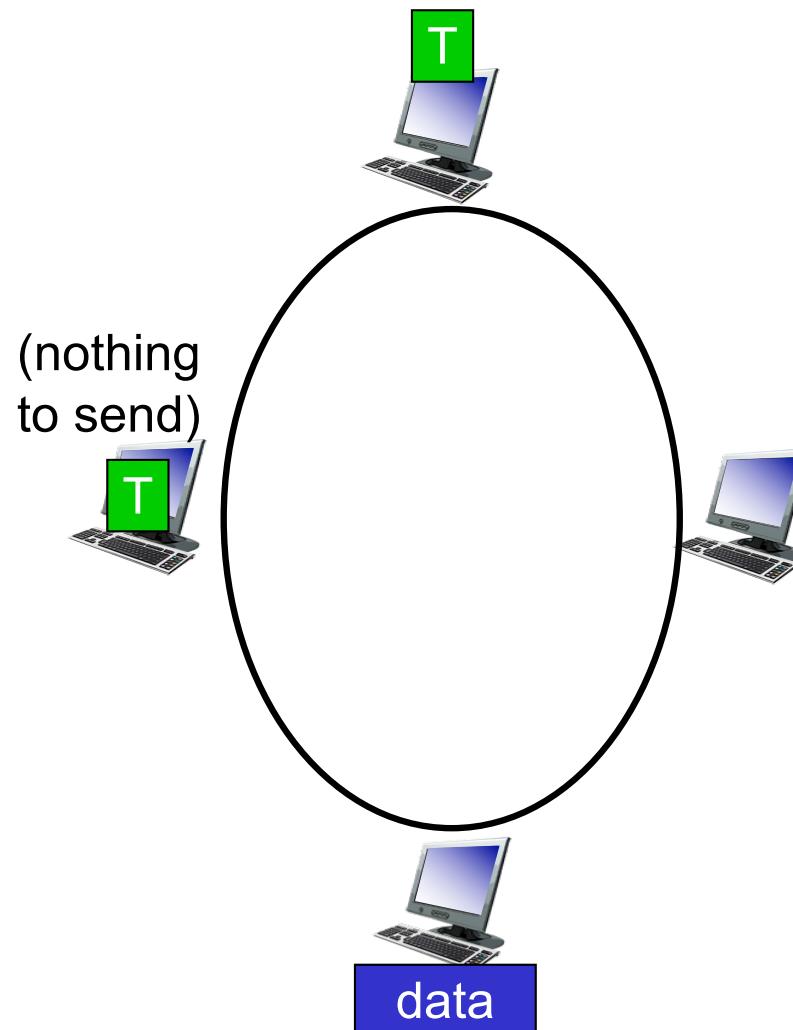
- ❖ master node “invites” slave nodes to transmit in turn
- ❖ typically used with “dumb” slave devices
- ❖ concerns:
 - polling overhead
 - latency
 - single point of failure (master)



“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)



Quiz: Does taking turns satisfy ideal properties ?



1. if only 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 (fairness)
 3. fully decentralized:
 - no synchronization of clocks, slots
 - no special node to coordinate transmissions
 4. simple
- A. 0
B. 1
C. 2
D. 3
E. 4
- (Which ones?)
- www.zeetings.com/salil
- Answer: D**
1, 2 and 4 are satisfied
(neglect the overheads for polling and token passing)

Summary of MAC protocols

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

Link layer, LANs: outline

6.1 introduction, services

6.7 a day in the life of a
web request

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 Switched LANs

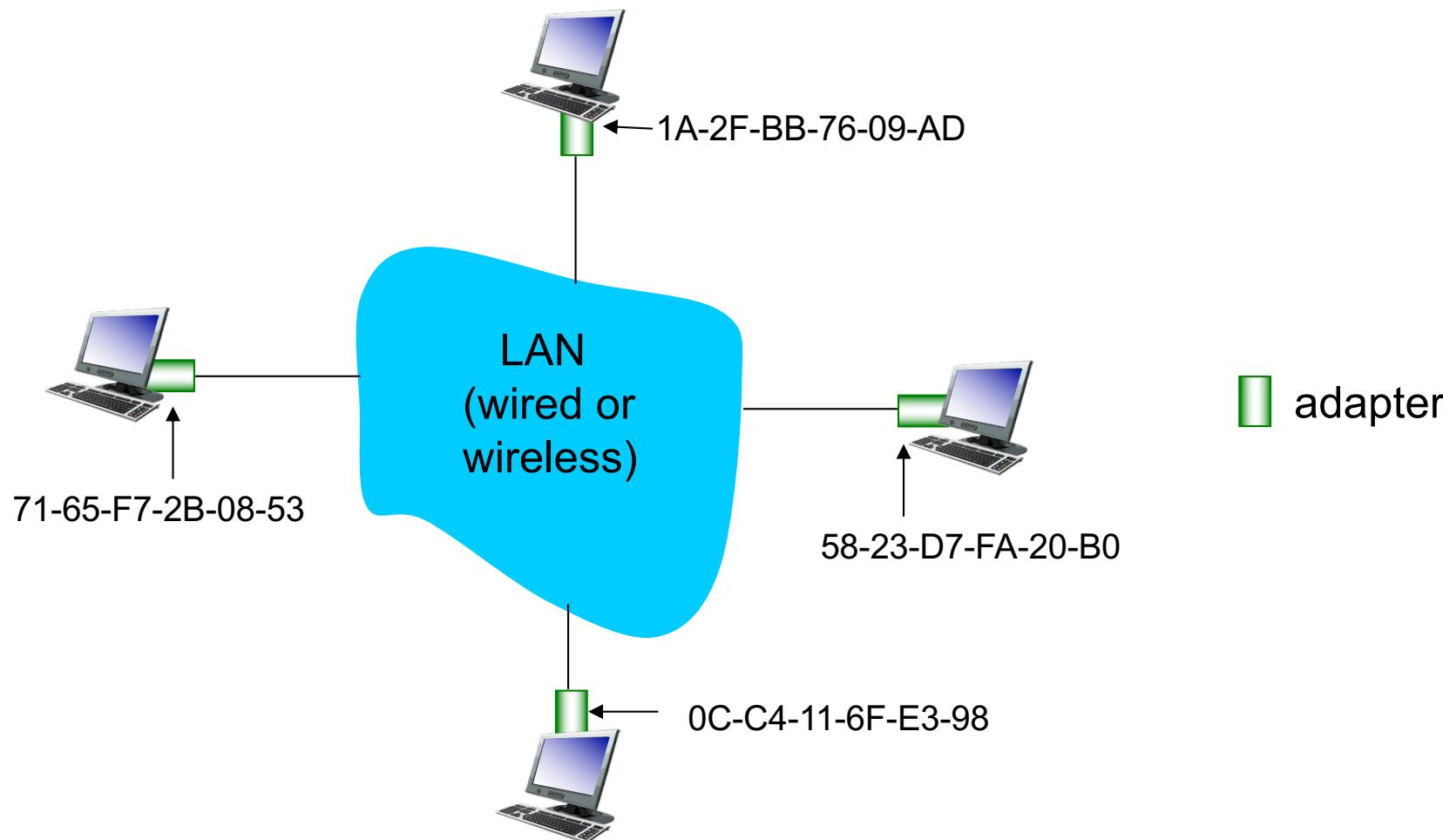
- addressing, ARP
- Ethernet
- switches

MAC addresses and ARP

- ❖ 32-bit IP address:
 - network-layer address for interface
 - used for layer 3 (network layer) forwarding
- ❖ MAC (or LAN or physical or Ethernet) address:
 - function: *used ‘locally’ to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
 - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
 - e.g.: IA-2F-BB-76-09-AD
 - hexadecimal (base 16) notation
 - (each “number” represents 4 bits)

LAN addresses and ARP

each adapter on LAN has unique *LAN* address



LAN addresses (more)

- ❖ MAC address allocation administered by IEEE
- ❖ manufacturer buys portion of MAC address space (to assure uniqueness)
- ❖ MAC flat address → portability
 - can move LAN card from one LAN to another
- ❖ IP hierarchical address *not* portable
 - address depends on IP subnet to which node is attached

MAC Address vs. IP Address

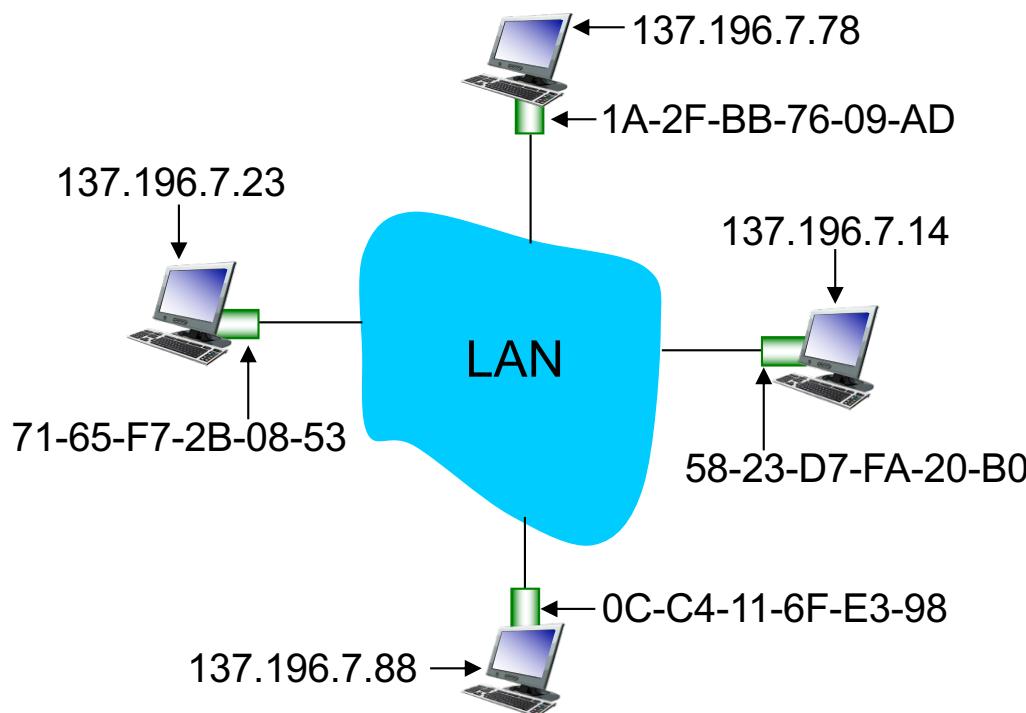
- ❖ MAC addresses (used in link-layer)
 - Hard-coded in read-only memory when adapter is built
 - Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
 - Portable, and can stay the same as the host moves
 - Used to get packet between interfaces on same network
- ❖ IP addresses
 - Configured, or learned dynamically
 - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
 - Not portable, and depends on where the host is attached
 - Used to get a packet to destination IP subnet

Taking Stock: Naming

Layer	Examples	Structure	Configuration	Resolution Service
App. Layer	www.cse.unsw.edu.au	organizational hierarchy	~ manual	DNS
Network Layer	129.94.242.51	topological hierarchy	DHCP	ARP
Link layer	45-CC-4E-12-F0-97	vendor (flat)	hard-coded	

ARP: address resolution protocol

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



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

- IP/MAC address mappings for some LAN nodes:
<IP address; MAC address; TTL>
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

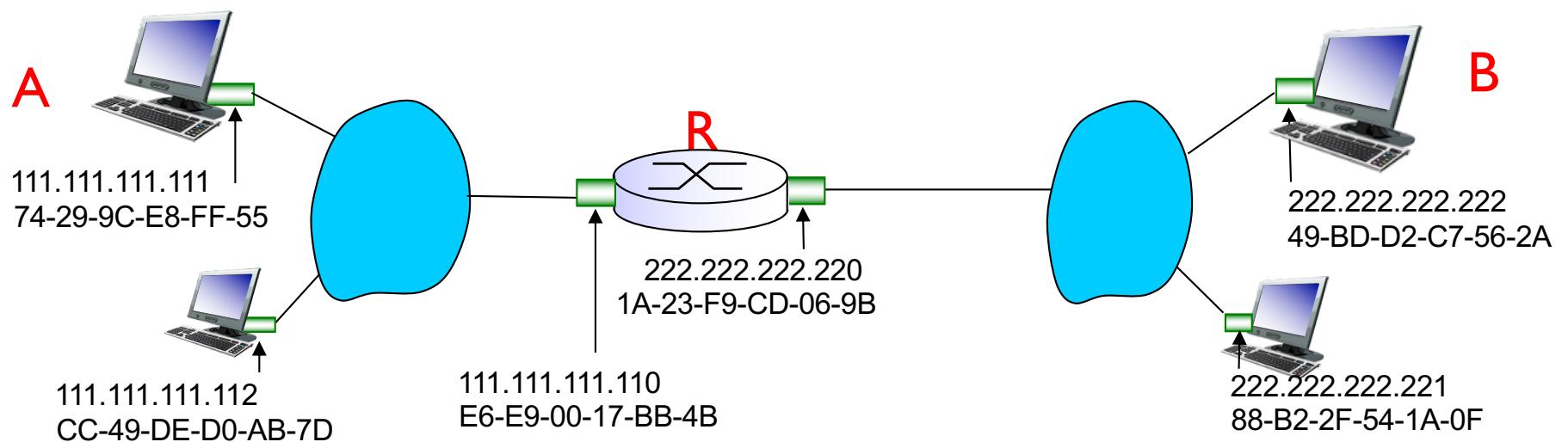
ARP protocol: same LAN

- ❖ A wants to send datagram to B
 - B's MAC address not in A's ARP table.
- ❖ A **broadcasts** ARP query packet, containing B's IP address
 - dest MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query
- ❖ B receives ARP packet, replies to A with its (B's) MAC address
 - frame sent to A's MAC address (unicast)
- ❖ A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
 - soft state: information that times out (goes away) unless refreshed
- ❖ ARP is “plug-and-play”:
 - nodes create their ARP tables *without intervention from net administrator*
- ❖ Only the node that responds to an ARP query caches the IP-MAC address mapping in its ARP table for the source of the query
 - In above example, B will add an ARP entry for A, but other nodes on the LAN will NOT

Addressing: routing to another LAN

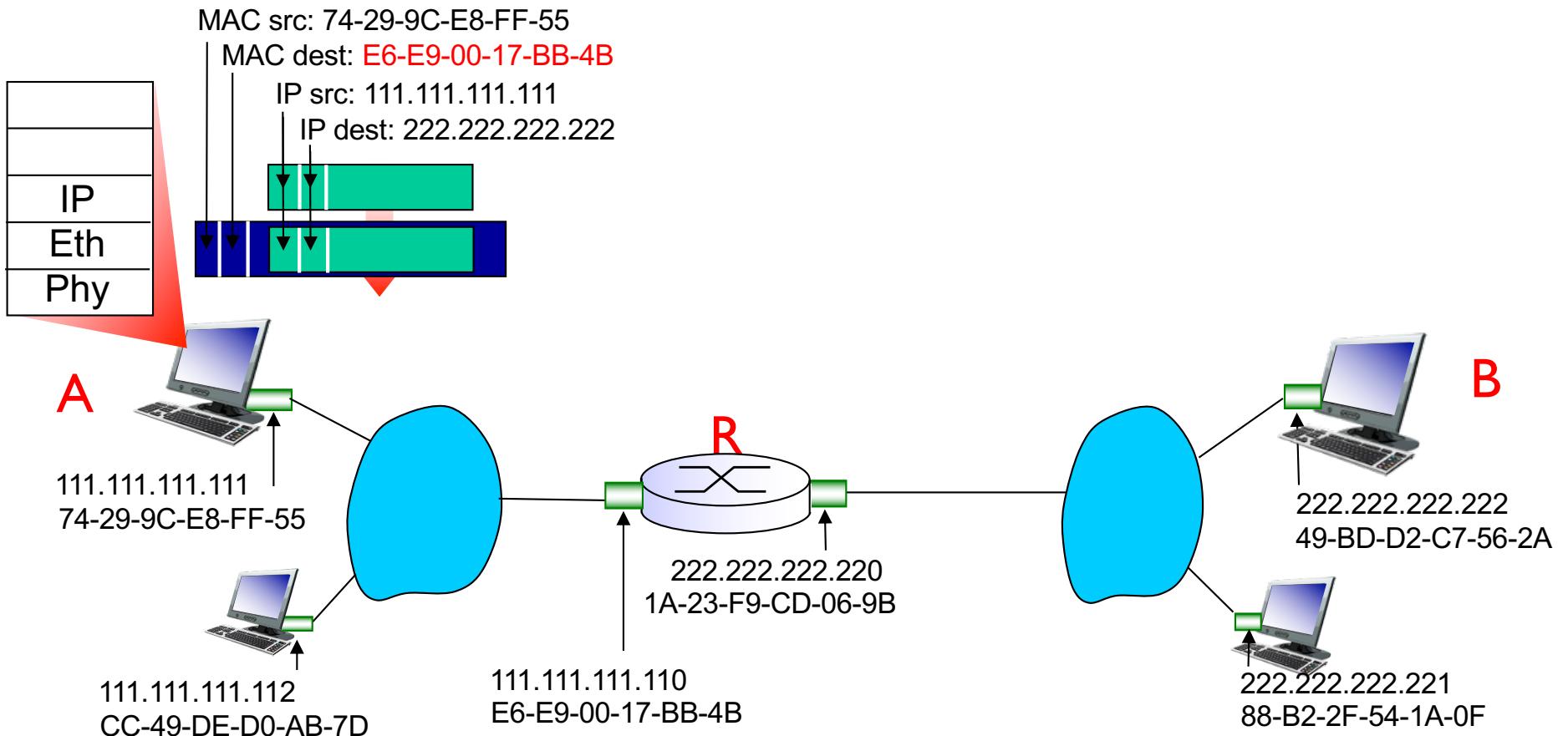
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address (how?)
 - How does A know B is not local (i.e., connected to the same LAN as A) ?
 - Subnet Mask (discovered via DHCP)
- assume A knows IP address of first hop router, R (how?)
 - Default router (discovered via DHCP)
- assume A knows R's MAC address (how?)
 - ARP



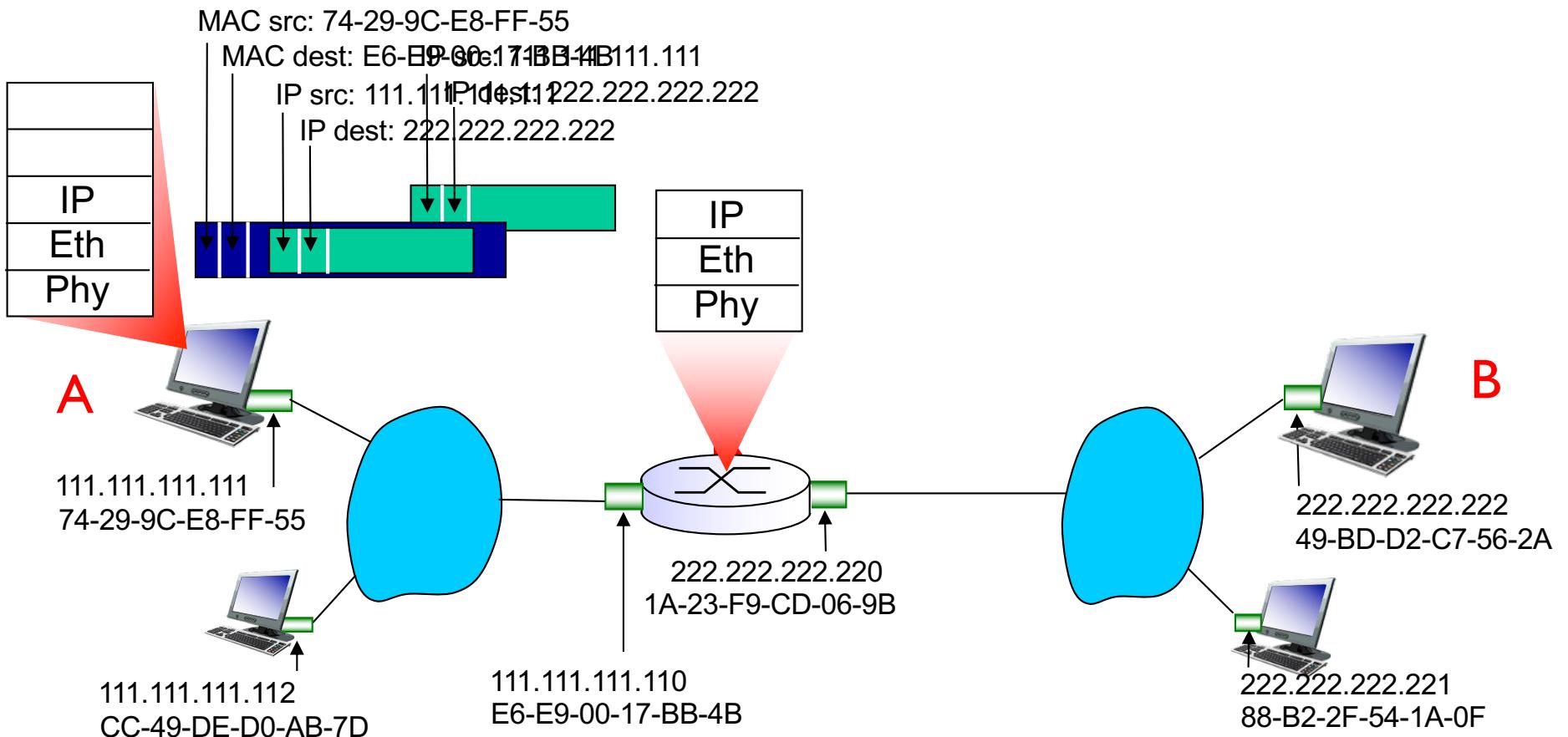
Addressing: routing to another LAN

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



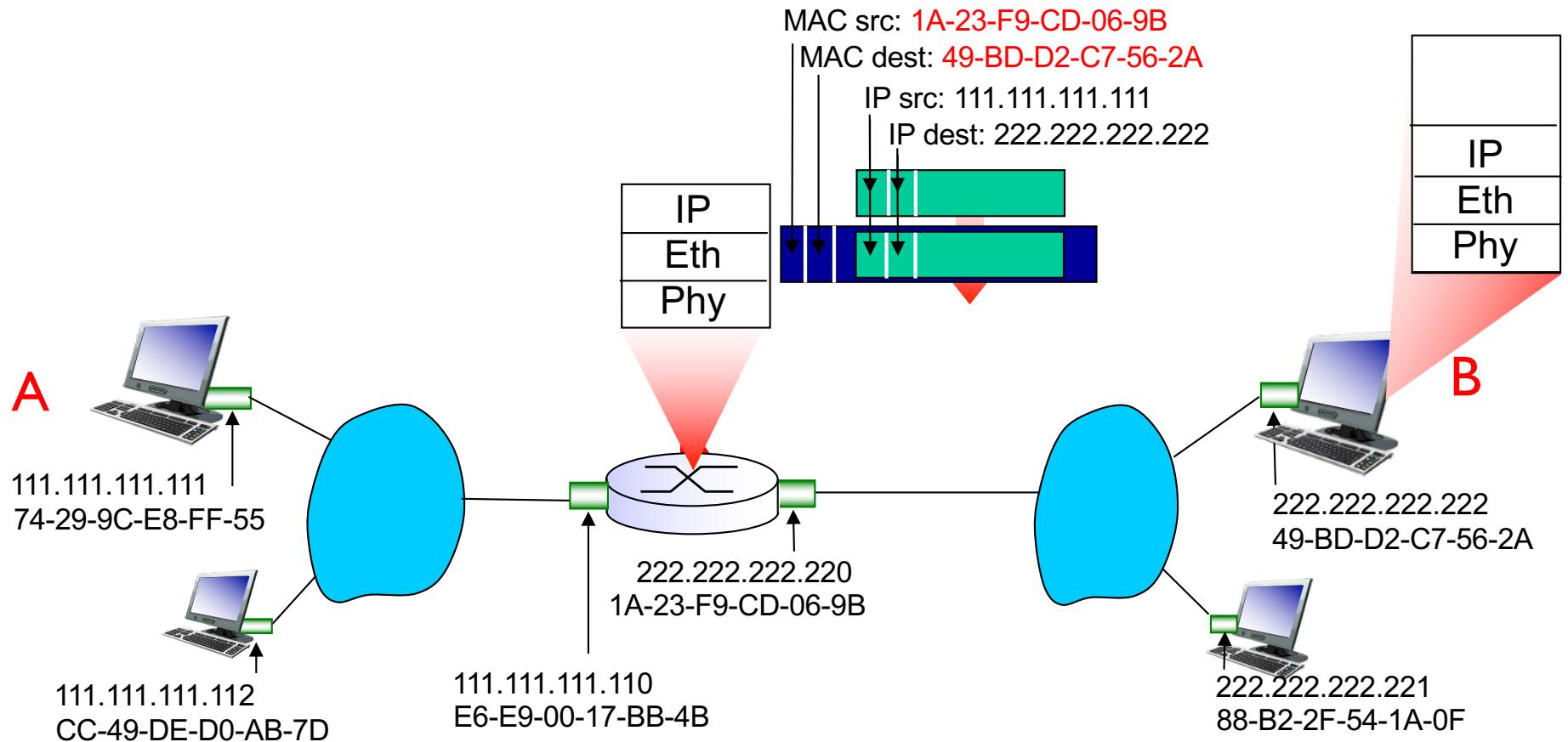
Addressing: routing to another LAN

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



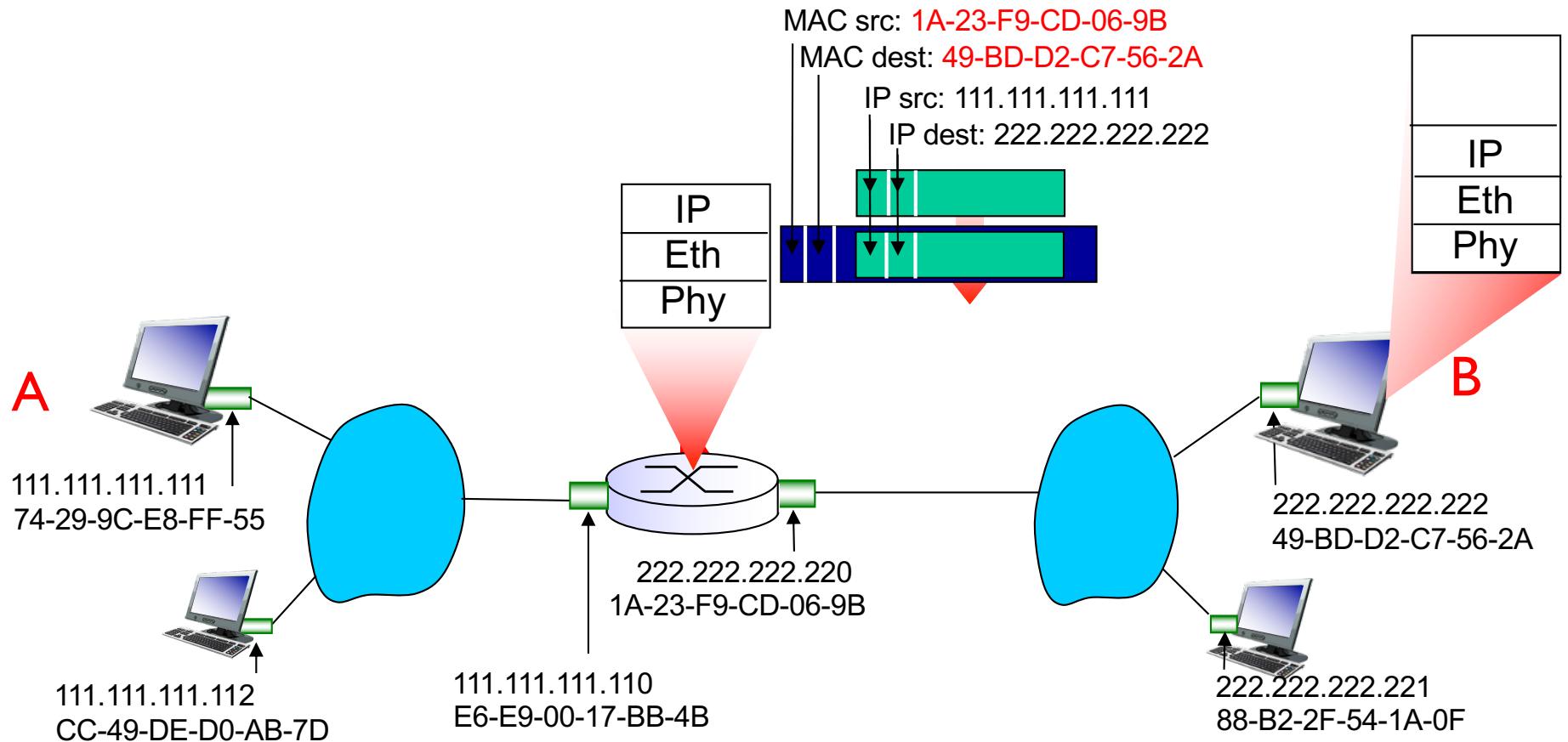
Addressing: routing to another LAN

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



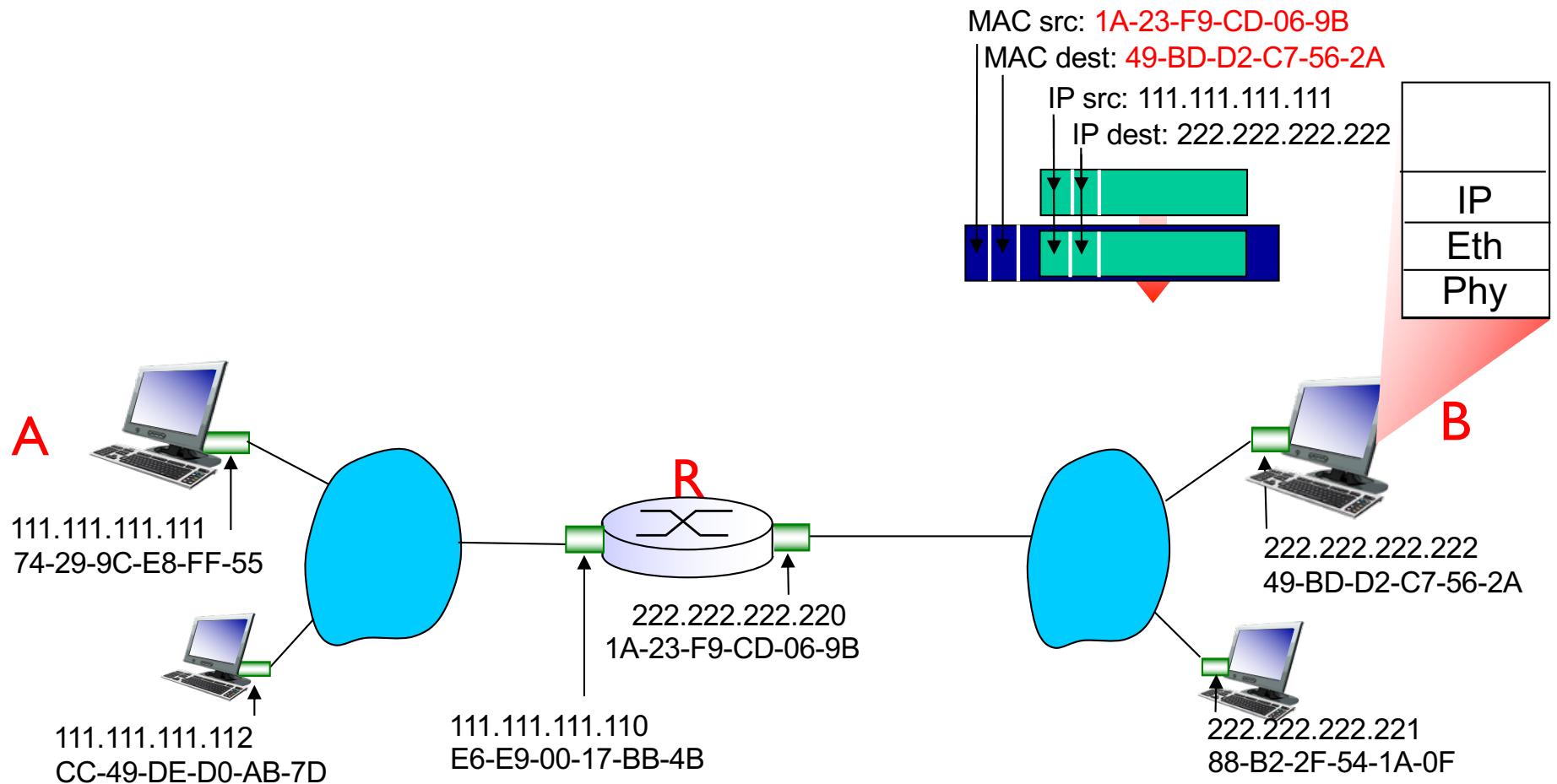
Addressing: routing to another LAN

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



Addressing: routing to another LAN

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



Example ARP Table

C:\Windows\system32\cmd.exe		
C:\Users\admin>arp -a		
Interface: 192.168.150.155 --- 0xb	Internet Address	Physical Address
	192.168.150.2	00-10-db-82-4d-52
	192.168.150.10	00-0e-7f-af-6d-b8
	192.168.150.24	00-0f-fe-25-74-40
	192.168.150.32	00-0b-cd-6e-b8-2c
	192.168.150.36	00-0f-fe-3a-aa-3f
	192.168.150.42	00-0f-fe-87-1e-98
	192.168.150.48	00-0e-7f-63-8d-d1
	192.168.150.54	00-16-35-ae-3b-a9
	192.168.150.58	00-16-35-ae-39-53
	192.168.150.60	00-21-63-68-e9-29
	192.168.150.62	00-0f-fe-9b-e8-38
	192.168.150.78	00-0f-fe-3a-a7-d7
	192.168.150.90	00-0e-7f-f2-f8-e8
	192.168.150.92	00-0f-fe-3a-a7-96
	192.168.150.98	00-0f-fe-85-8d-6b
	192.168.150.114	00-0e-7f-6c-81-25
	192.168.150.144	00-22-5f-12-67-a2
	192.168.150.156	00-0f-fe-d1-7e-1e
	192.168.150.157	00-0f-fe-d1-7e-1e
	192.168.150.159	00-06-1b-c2-e1-f3
	192.168.150.208	00-19-66-32-53-25
	192.168.150.219	00-00-aa-8c-be-07
	192.168.150.221	00-0e-7f-64-5f-d0
	192.168.150.255	ff-ff-ff-ff-ff-ff
	224.0.0.22	01-00-5e-00-00-16
	224.0.0.251	01-00-5e-00-00-fb
	224.0.0.252	01-00-5e-00-00-fc
	224.0.1.134	01-00-5e-00-01-86
	239.255.255.250	01-00-5e-7f-ff-fa

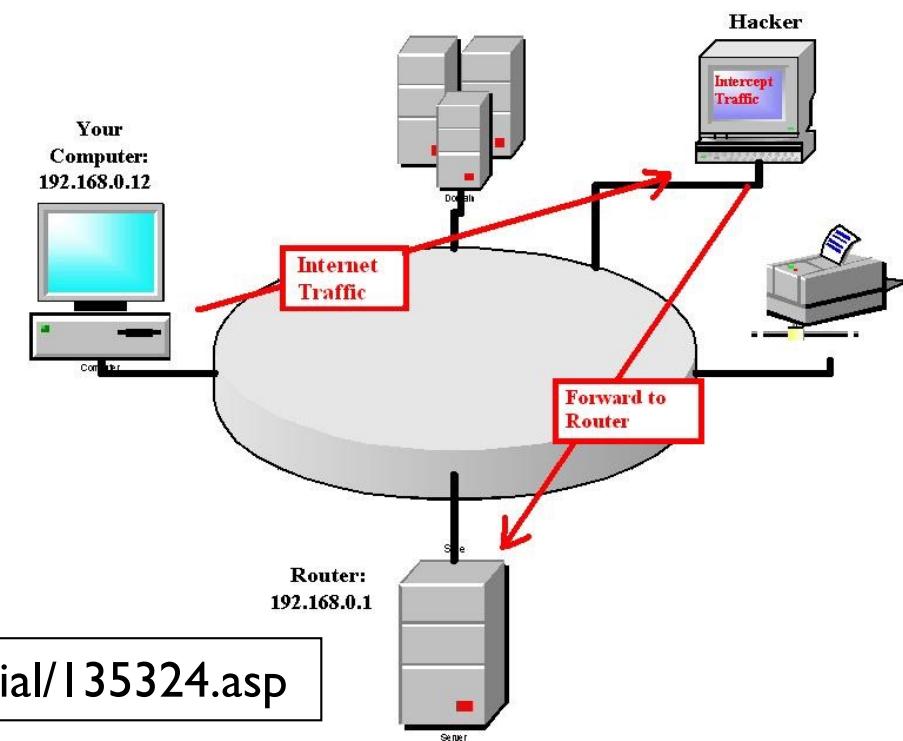
Security Issues: ARP Cache Poisoning



- ❖ Denial of Service - Hacker replies back to an ARP query for a router NIC with a fake MAC address
- ❖ Man-in-the-middle attack - Hacker can insert his/her machine along the path between victim machine and gateway router
- ❖ Such attacks are generally hard to launch as hacker needs physical access to the network

Solutions -

- Use Switched Ethernet with port security enabled (i.e., one host MAC address per switch port)
- Adopt static ARP configuration for small size networks
- Use ARP monitoring tools such as ARPWatch



<http://www.watchguard.com/infocenter/editorial/135324.asp>

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

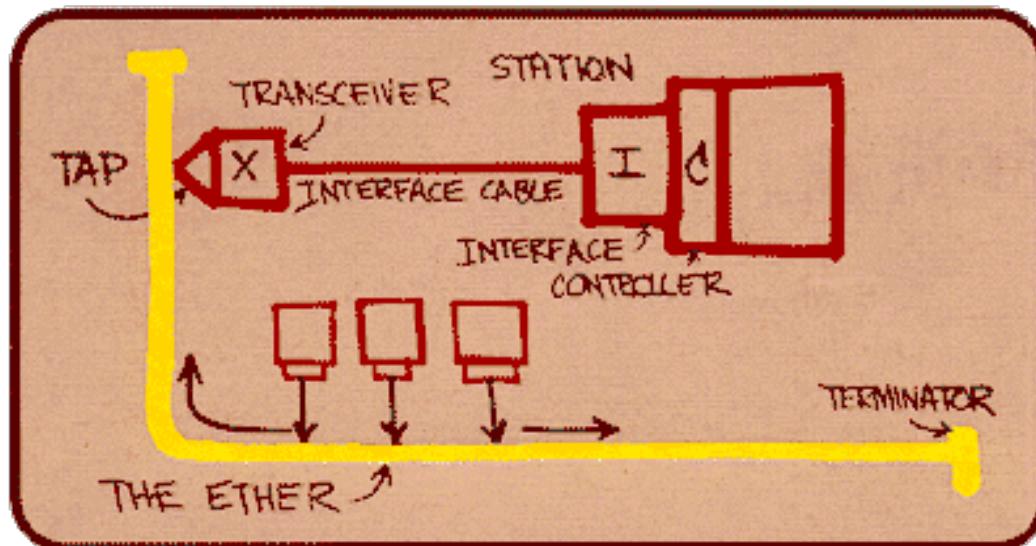
6.4 LANs

- addressing, ARP
- Ethernet
- switches

6.7 a day in the life of a
web request

Ethernet

Bob Metcalfe, Xerox PARC, visits Hawaii and gets an idea!



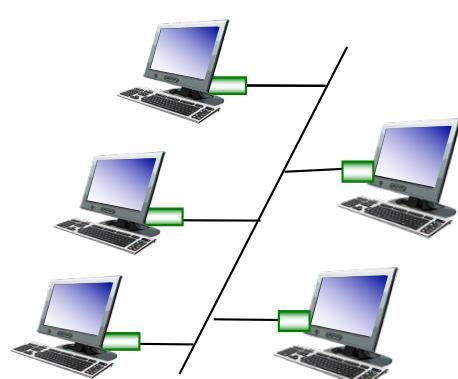
Metcalfe's Ethernet sketch

“dominant” wired LAN technology:

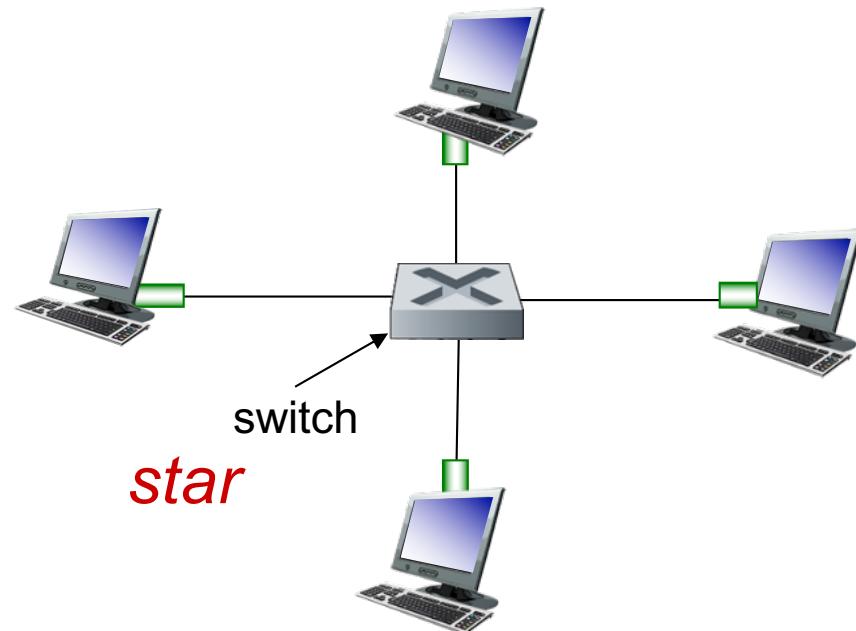
- ❖ first widely used LAN technology
- ❖ simpler, cheaper than token LANs and ATM
- ❖ kept up with speed race: 10 Mbps – 10 Gbps

Ethernet: physical topology

- ❖ *bus*: popular through mid 90s
 - all nodes in same collision domain (can collide with each other)
 - CSMA/CD for media access control
- ❖ *star*: prevails today
 - active *switch* in center
 - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)
 - No sharing, no CSMA/CD



bus: coaxial cable



Ethernet frame structure

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

Preamble 7 Bytes	SFD 1 Byte	Dest MAC 6 Bytes	Source MAC 6 Bytes	Type/Length 2 Bytes	Payload 46-1500 Bytes	FCS/CRC 4 Bytes	Inter Frame Gap
---------------------	---------------	---------------------	-----------------------	------------------------	--------------------------	--------------------	-----------------

preamble:

- ❖ Start of frame is recognized by
 - Preamble : Seven bytes with pattern 10101010
 - Start of Frame Delimiter (SFD) : 10101011
- ❖ used to synchronize receiver, sender clock rates
- Inter Frame Gap is 12 Bytes (96 bits) of idle state
 - 0.96 microsec for 100 Mbit/s Ethernet
 - 0.096 microsec for Gigabit/s Ethernet

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., ARP, Novell IPX, AppleTalk)
- ❖ **CRC:** cyclic redundancy check at receiver
 - error detected: frame is dropped



Ethernet: unreliable, connectionless

- ❖ **connectionless**: no handshaking between sending and receiving NICs
- ❖ **unreliable**: receiving NIC does not send acks or nacks to sending NIC
 - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- ❖ Ethernet's MAC protocol: unslotted **CSMA/CD with binary backoff**

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches

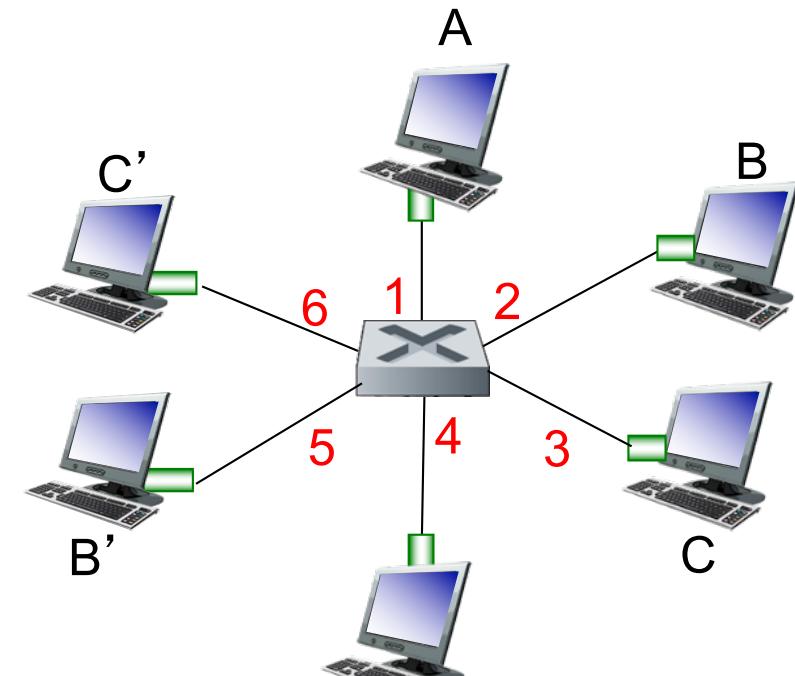
6.7 a day in the life of a
web request

Ethernet switch

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

Switch: multiple simultaneous transmissions

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

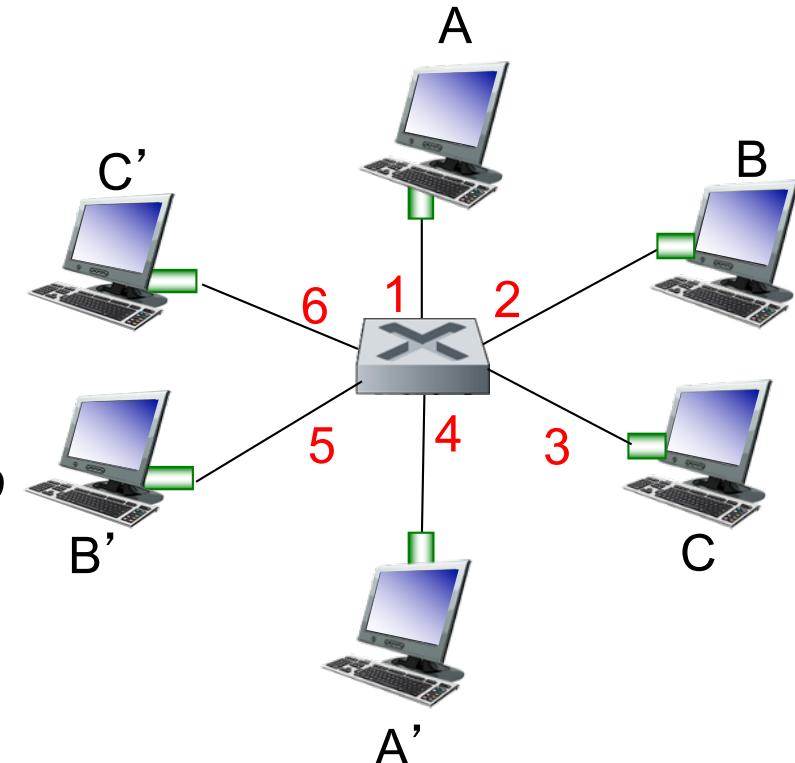


*switch with six interfaces
(1,2,3,4,5,6)*

Switch forwarding table

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

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



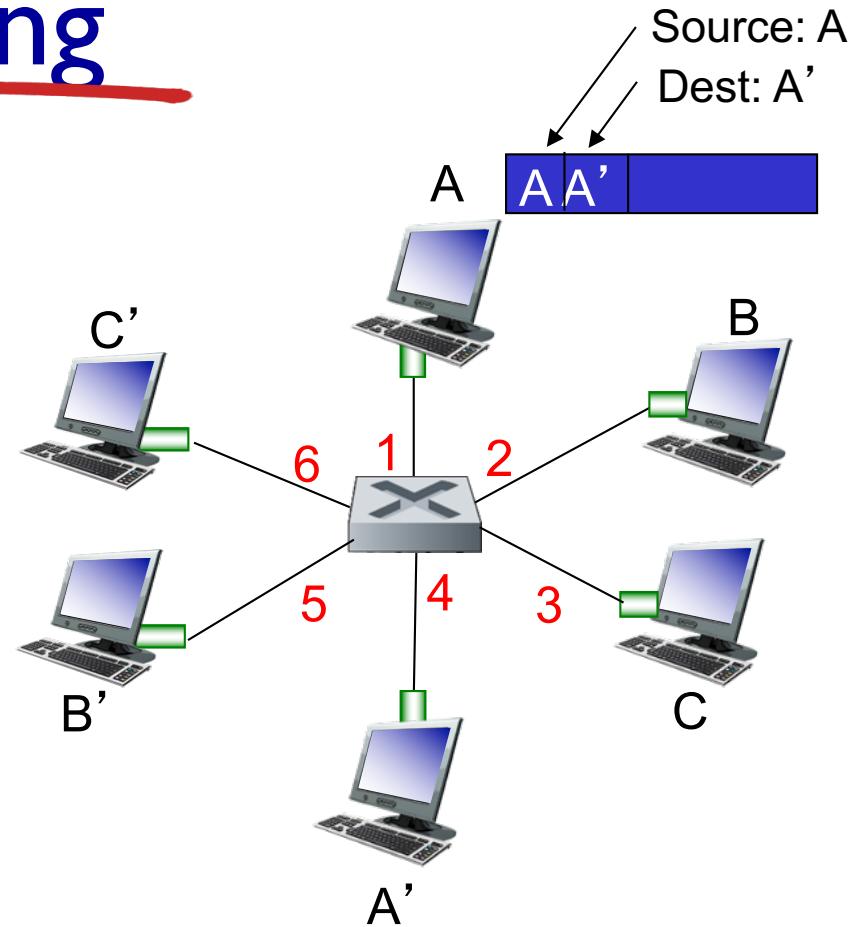
Q: how are entries created, maintained in switch table?

- something like a *routing protocol?*

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

Switch: self-learning

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



MAC addr	interface	TTL
A	1	60

*Switch table
(initially empty)*

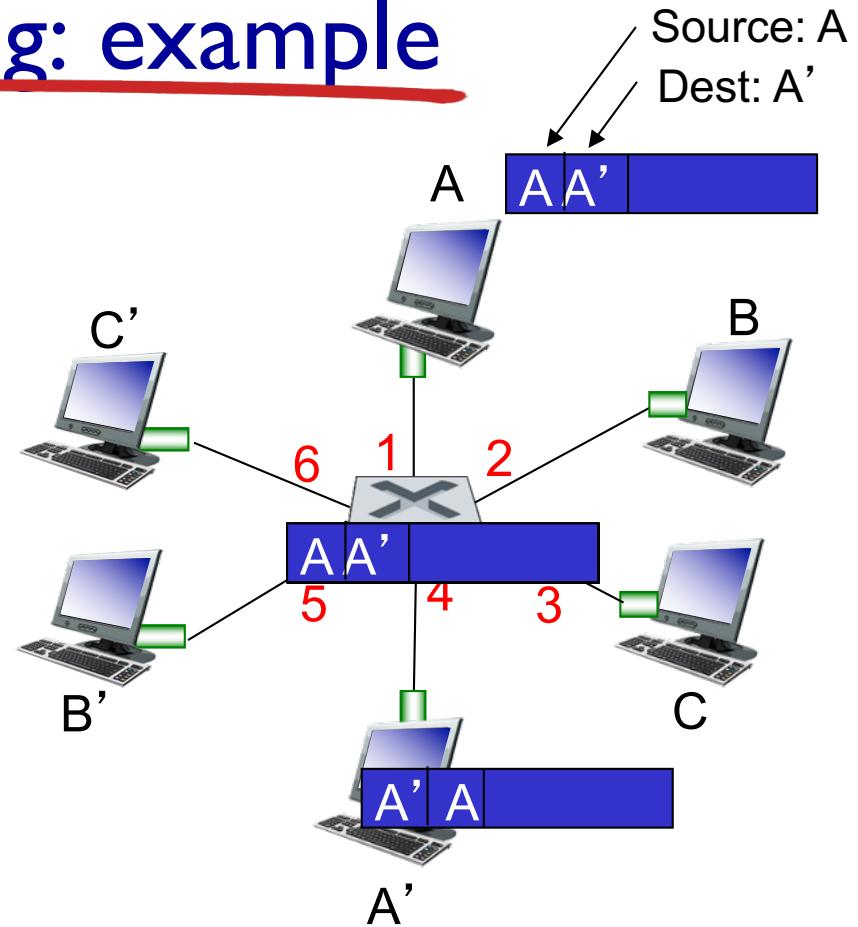
Switch: frame filtering/forwarding

when frame received at switch:

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

Self-learning, forwarding: example

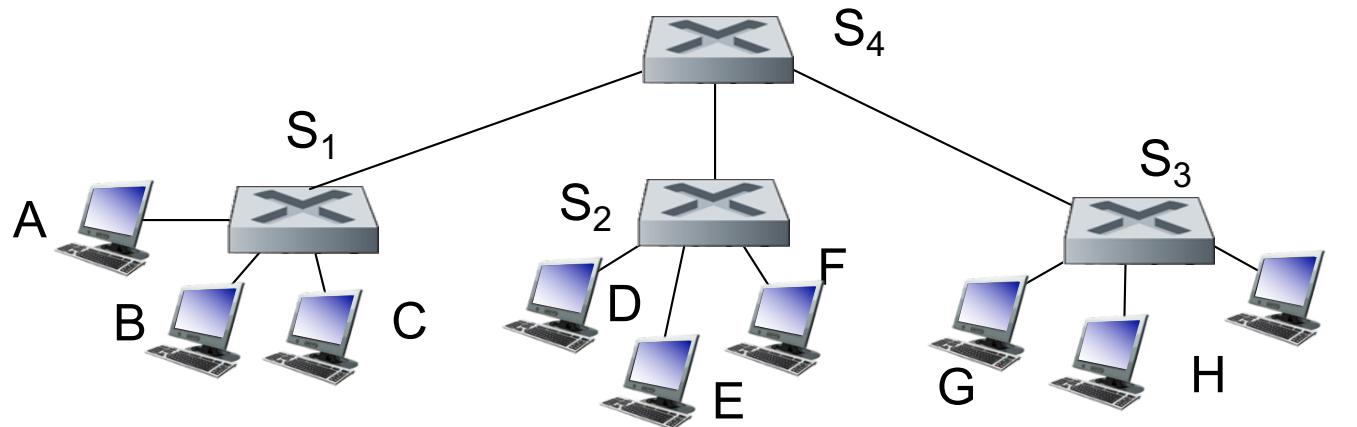
- ❖ frame destination, A' , location unknown: *flood*
- ❖ destination A location known: *selectively send on just one link*



*switch table
(initially empty)*

Interconnecting switches

- ❖ switches can be connected together



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

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

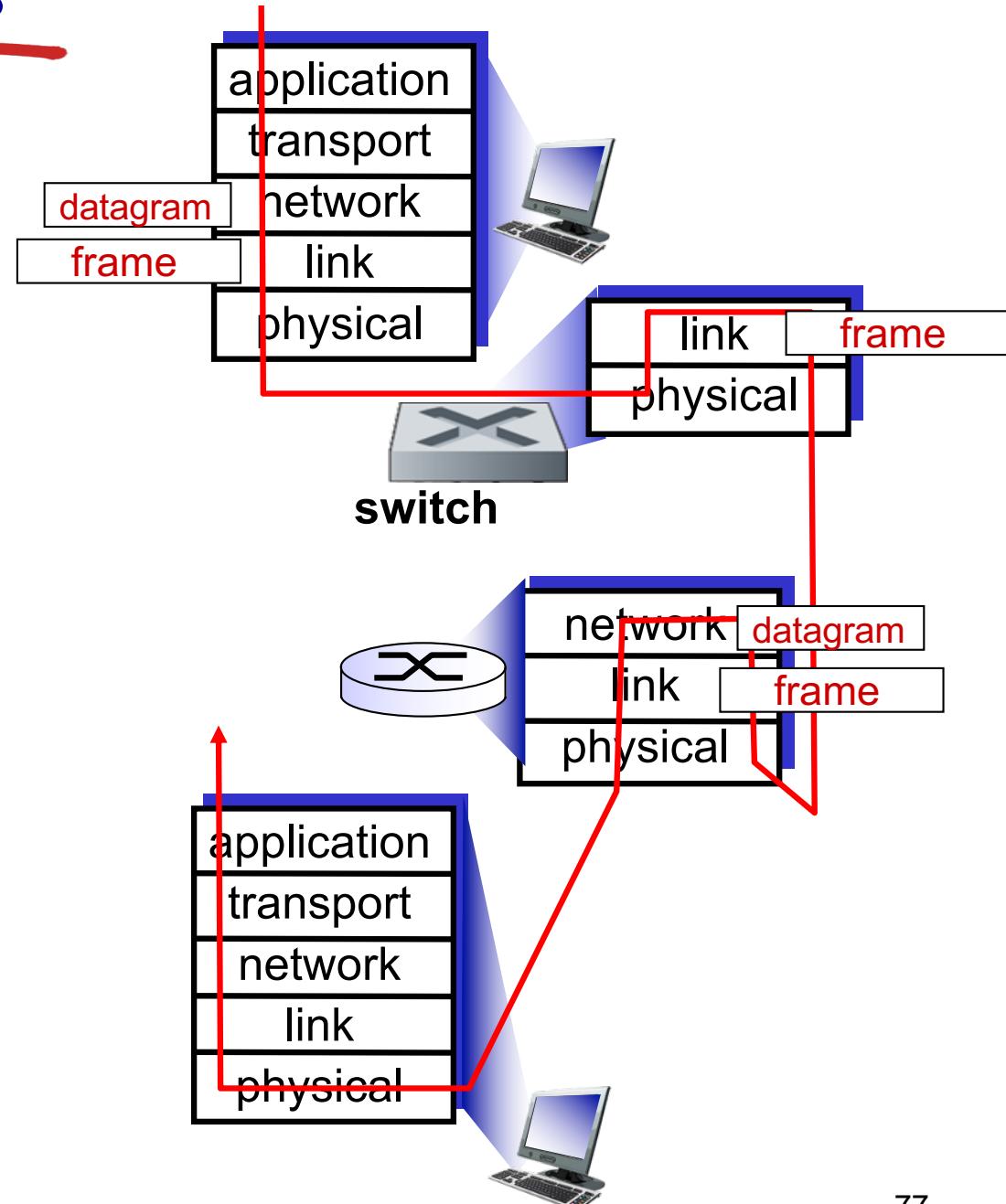
Switches vs. routers

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



Security Issues

- ❖ In a switched LAN once the switch table entries are established frames are not broadcast
 - Sniffing frames is harder than pure broadcast LANs
 - Note: attacker can still sniff broadcast frames and frames for which there are no entries (as they are broadcast)
- ❖ Switch Poisoning: Attacker fills up switch table with bogus entries by sending large # of frames with bogus source MAC addresses
- ❖ Since switch table is full, genuine packets frequently need to be broadcast as previous entries have been wiped out

Quiz

- ❖ A switch can
 - A. Filter a frame
 - B. Forward a frame
 - C. Extend a LAN
 - D. All of the above

Answer: D

Quiz

- ❖ The _____ will typically change from hop to hop, but the _____ will typically remain the same

Answer: D

- A. Source MAC address, destination MAC address
- B. Source IP address, destination IP address
- C. Source & destination IP addresses, source & destination MAC addresses
- D. Source & destination MAC addresses, source & destination IP addresses

Link layer, LANs: outline

6.1 introduction, services

6.7 a day in the life of a
web request

6.2 error detection,
correction

6.3 multiple access
protocols

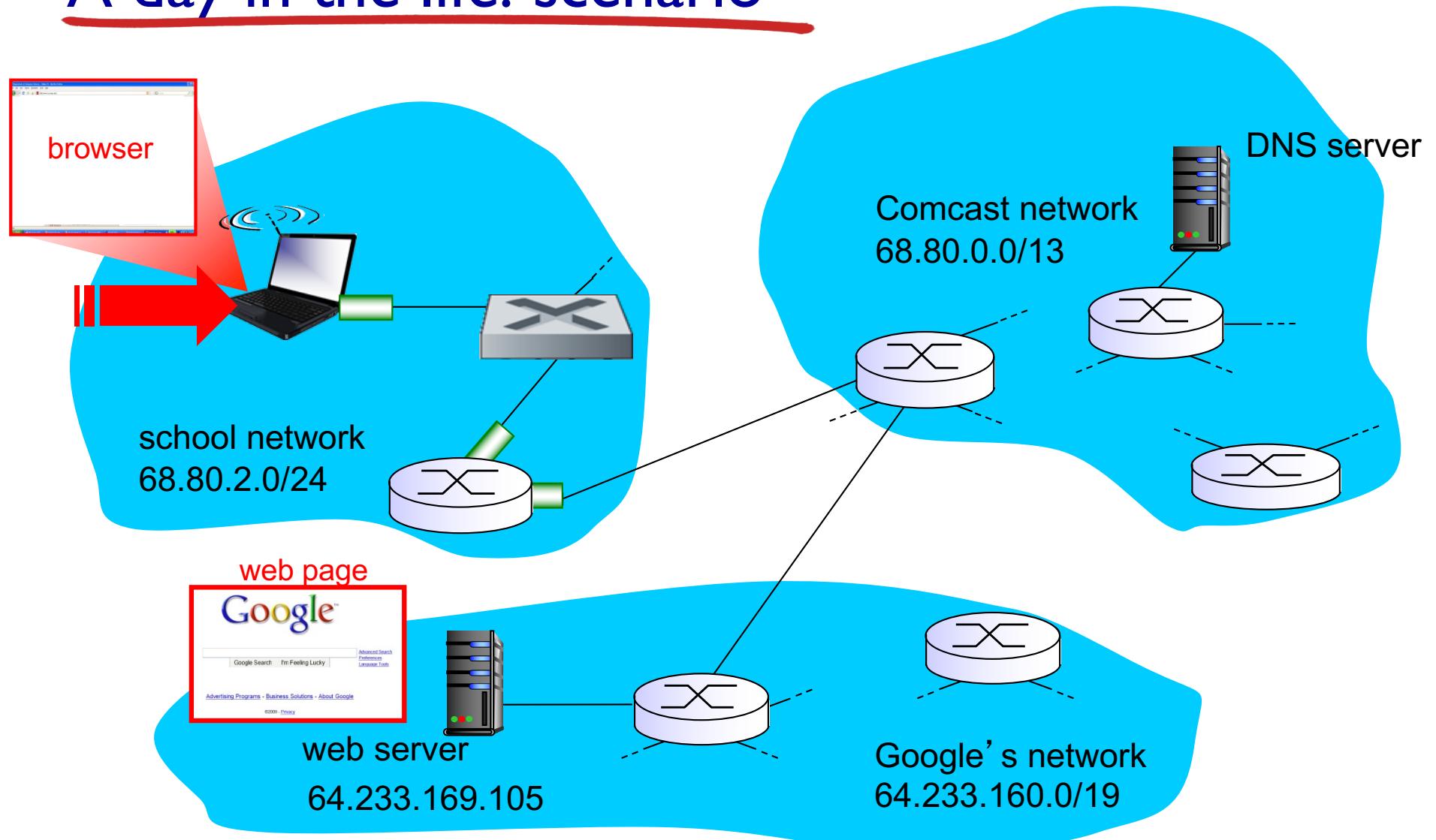
6.4 LANs

- addressing, ARP
- Ethernet
- switches

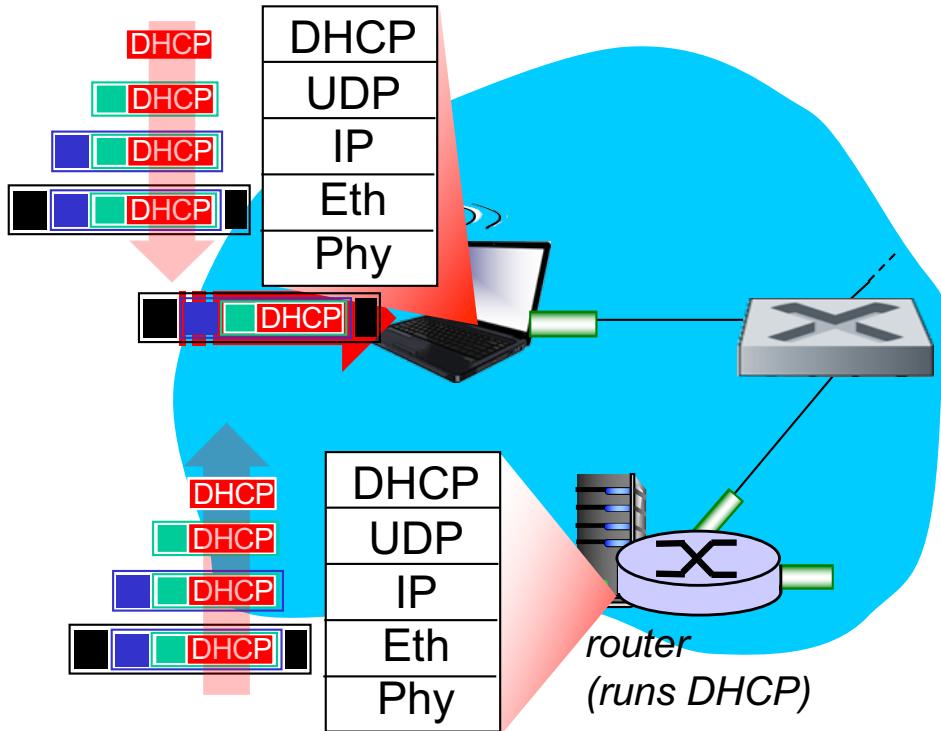
Synthesis: a day in the life of a web request

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

A day in the life: scenario

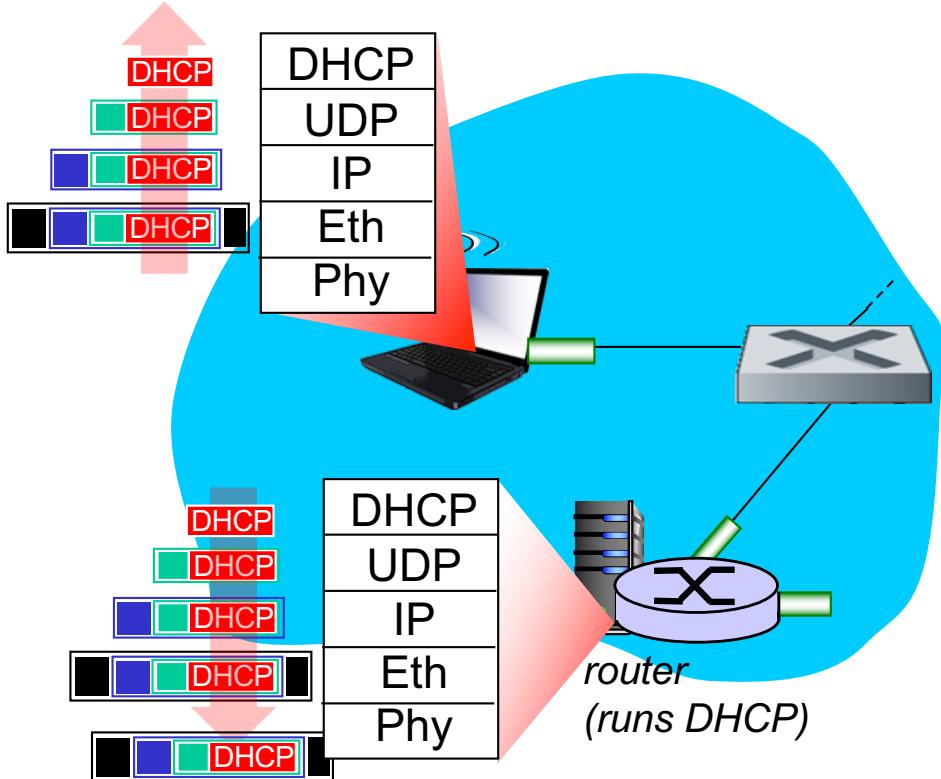


A day in the life... connecting to the Internet



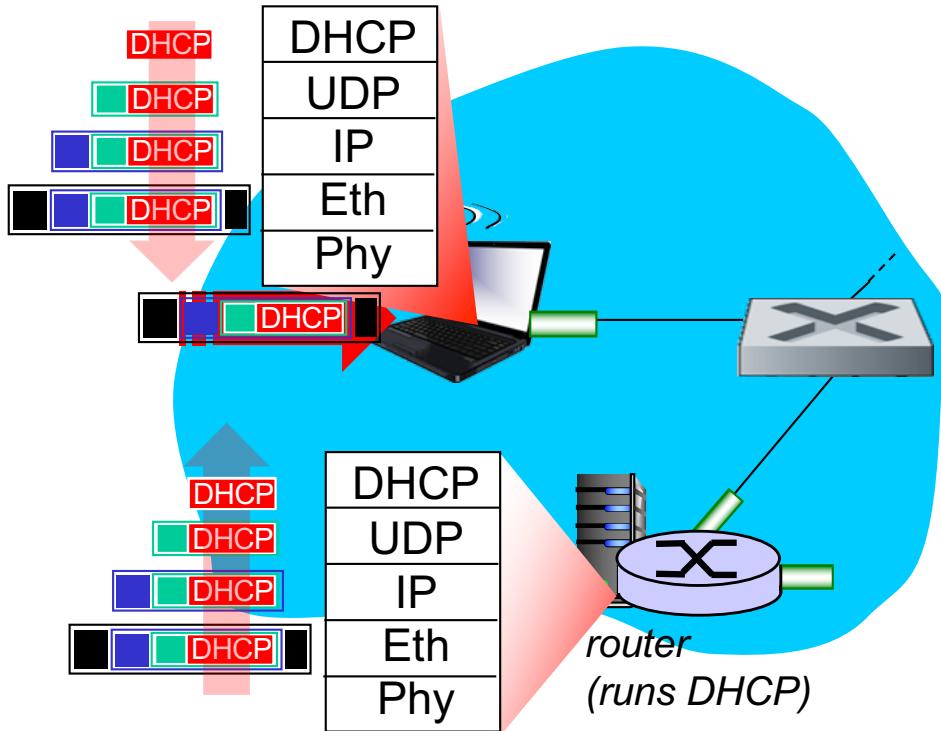
- ❖ connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**
- ❖ DHCP **Discover** Message *encapsulated* in **UDP**, *encapsulated* in **IP**, *encapsulated* in **802.3** Ethernet
- ❖ Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received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

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



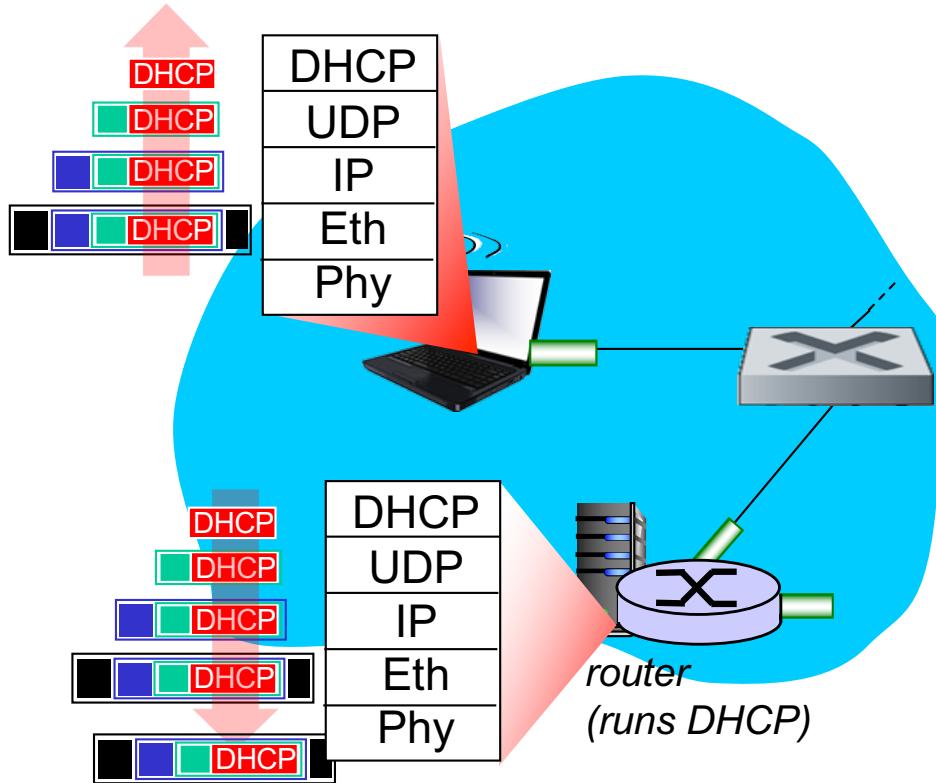
- ❖ DHCP server formulates *DHCP Offer* message containing client's IP address
- ❖ encapsulation at DHCP server; frame again broadcasted on LAN
- ❖ DHCP client receives DHCP Offer message

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



- ❖ The client initiates **DHCP Request** message
- ❖ 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

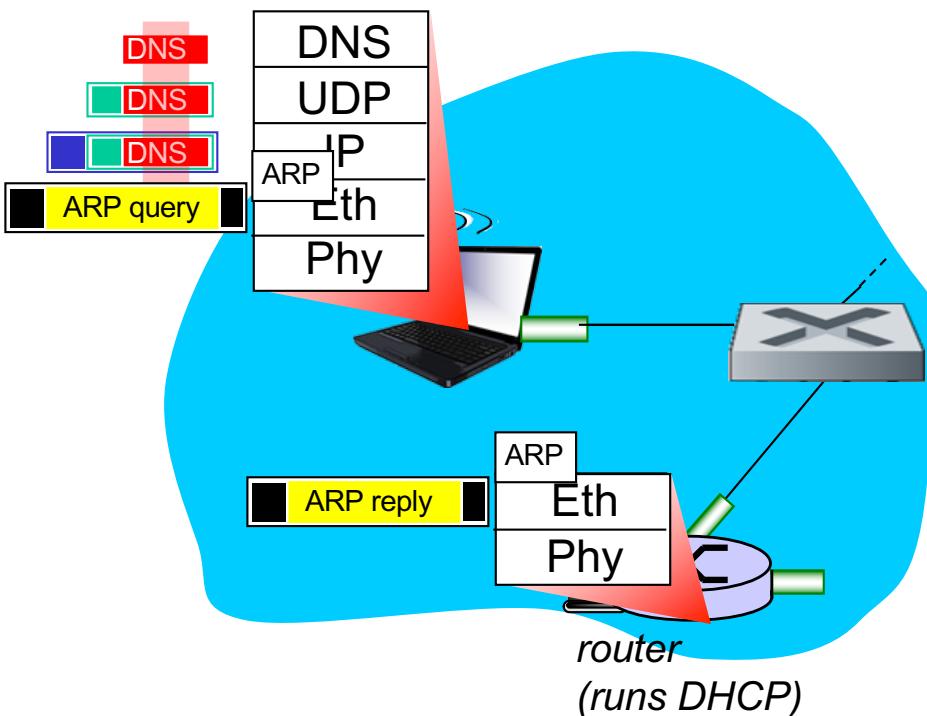
A day in the life... connecting to the Internet



- ❖ 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 broadcasted through LAN,
- ❖ DHCP client receives DHCP ACK reply

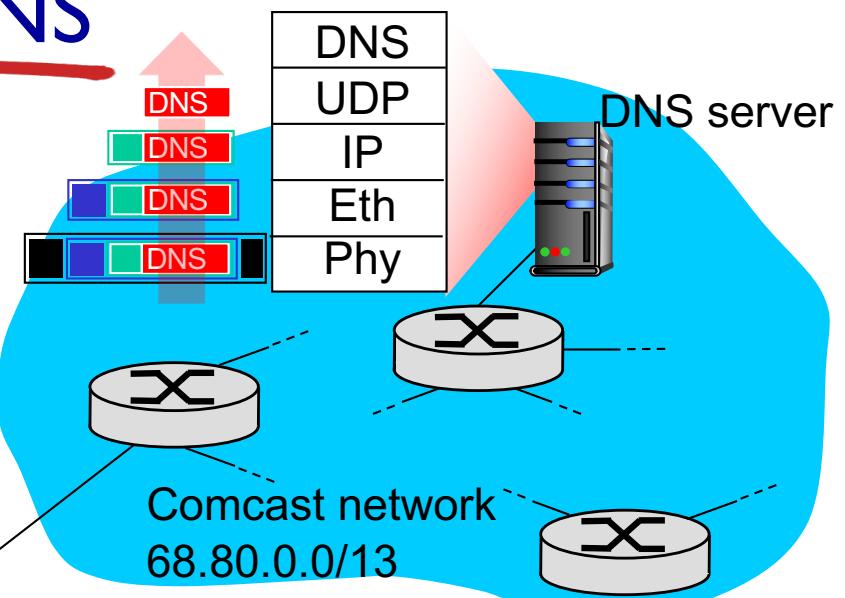
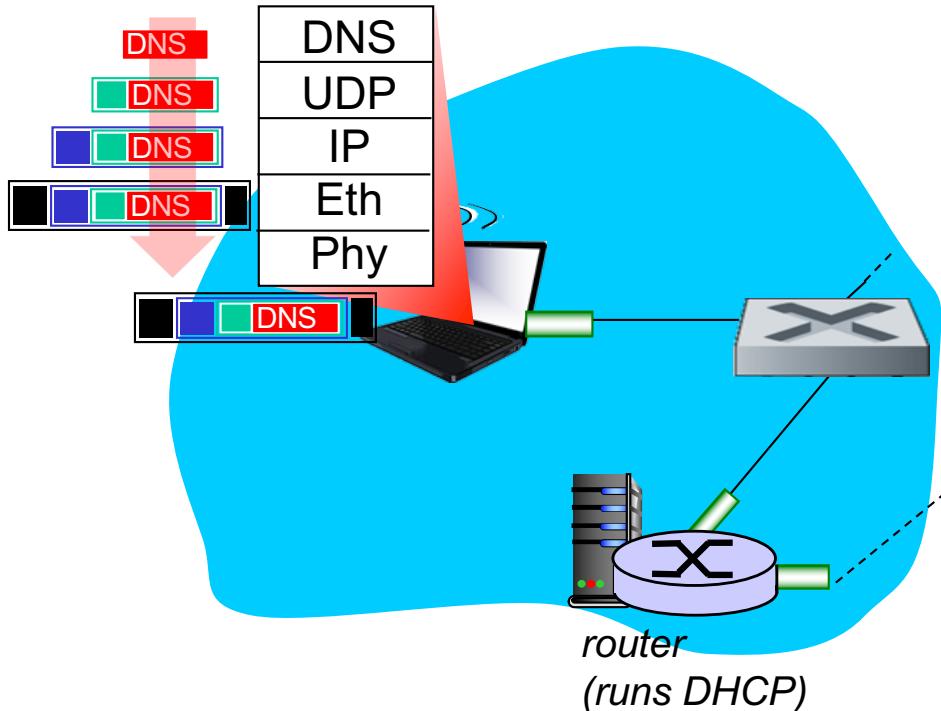
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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



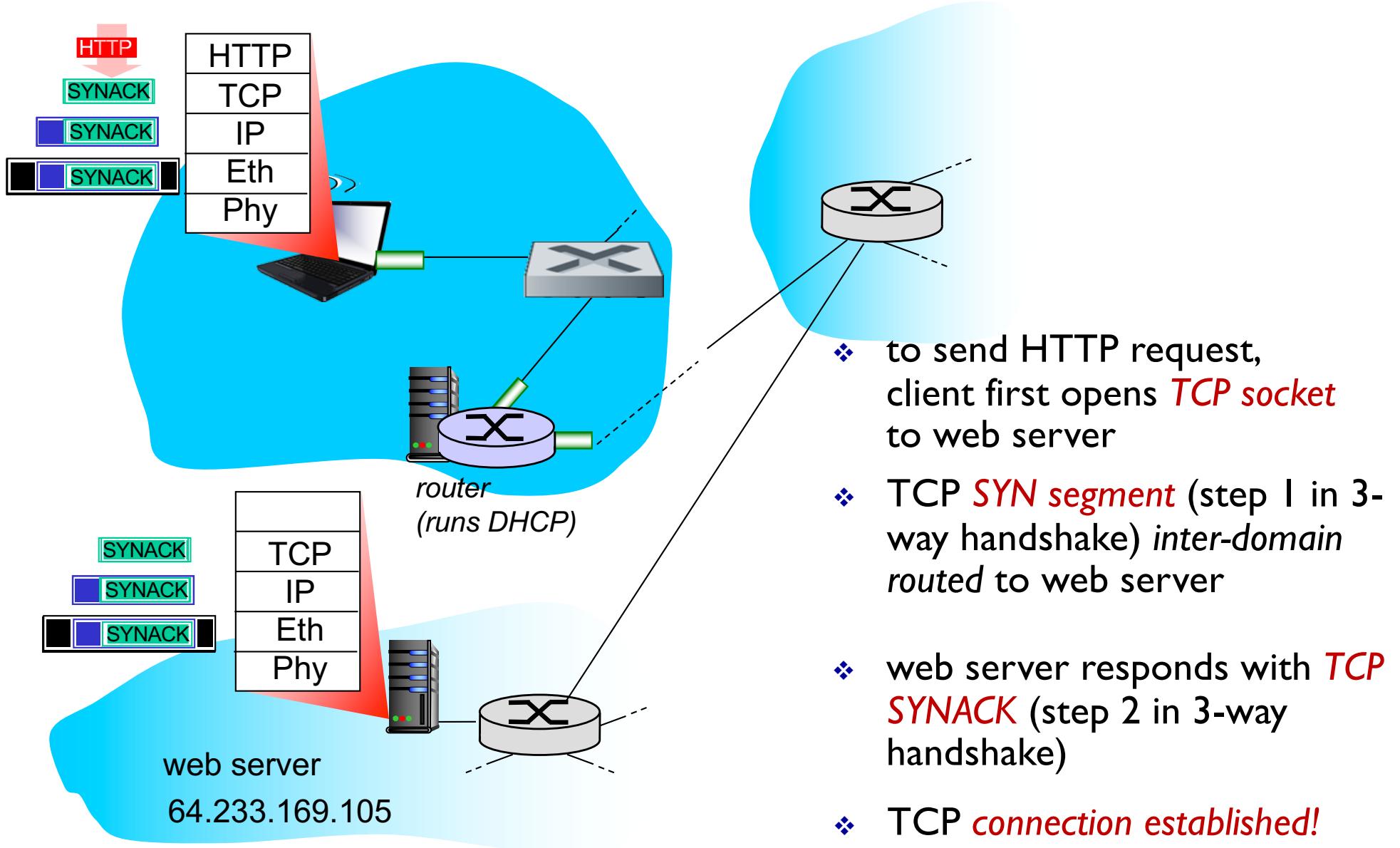
- ❖ before sending **HTTP** request, need IP address of www.google.com: **DNS**
- ❖ DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to DNS server, need MAC address of first hop router: **ARP**
- ❖ **ARP query** broadcast, received by router, which replies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

A day in the life... using DNS

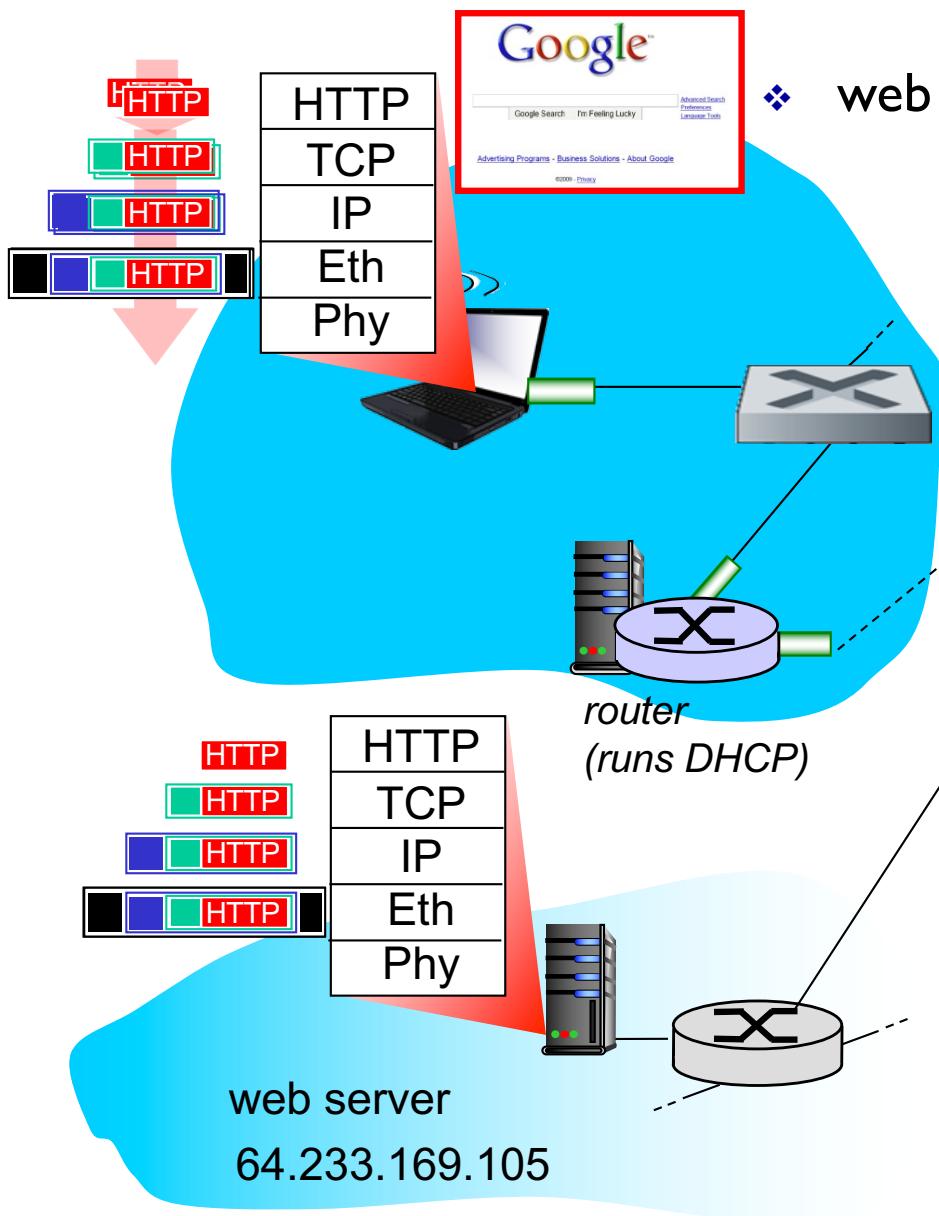


- ❖ IP datagram containing DNS query forwarded via LAN switch from client to first hop router
- ❖ IP datagram forwarded from first hop router in campus network into Comcast network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- ❖ demux'ed to DNS server
- ❖ DNS server replies to client with IP address of www.google.com

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



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



❖ web page *finally (!!?)* displayed

- ❖ *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

Link Layer: Summary

- ❖ principles behind data link layer services:
 - error detection, correction
 - sharing a broadcast channel: multiple access
 - link layer addressing
- ❖ instantiation and implementation of various link layer technologies
 - Ethernet
 - switched LANS

Link Layer: let's take a breath

- ❖ journey down protocol stack *complete* (except PHY)
- ❖ solid understanding of networking principles, practice
- ❖ could stop here but *lots* of interesting topics!
 - **Wireless**
 - **Security**