

source: computer-networks-webdesign.com



CSCI 4760 - Computer Networks Fall 2016

Instructor: Prof. Roberto Perdisci
perdisci@cs.uga.edu

These slides are adapted from the textbook slides by J.F. Kurose and K.W. Ross

Chapter 5: The Data Link Layer

Our goals:

- ▶ understand principles behind data link layer services:
 - ▶ error detection, correction
 - ▶ sharing a broadcast channel: multiple access
 - ▶ link layer addressing
 - ▶ reliable data transfer, flow control: *done!*
- ▶ instantiation and implementation of various link layer technologies

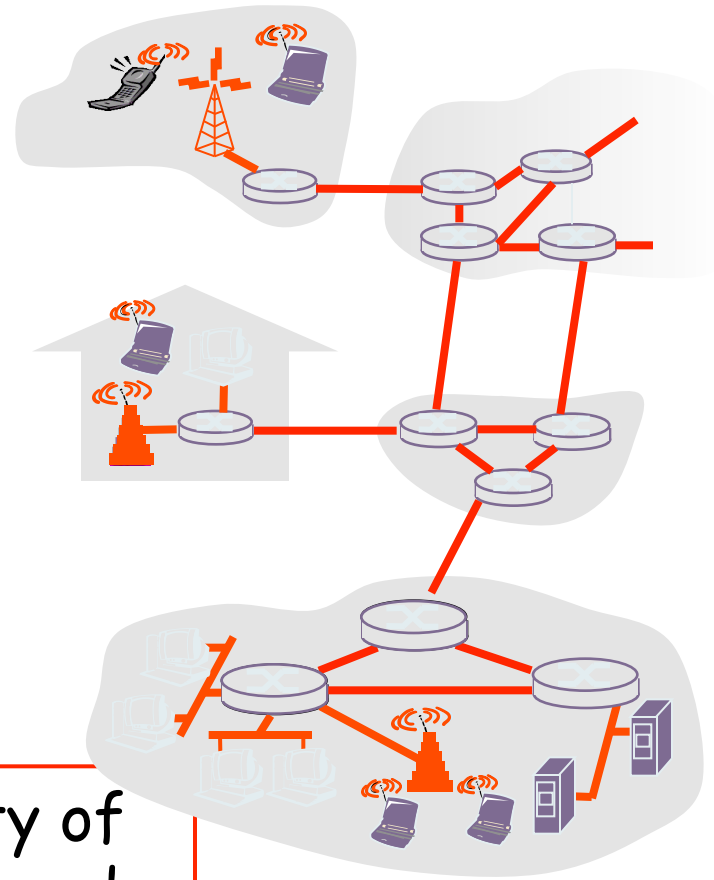
Link Layer

- ▶ **5.1 Introduction and services**
- ▶ 5.2 Error detection and correction
- ▶ 5.3 Multiple access protocols
- ▶ 5.4 Link-layer Addressing
- ▶ 5.5 Ethernet
- ▶ 5.6 Link-layer switches
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

Link Layer: Introduction

Some terminology:

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



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

Link layer: context

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

transportation analogy

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

Link Layer Services

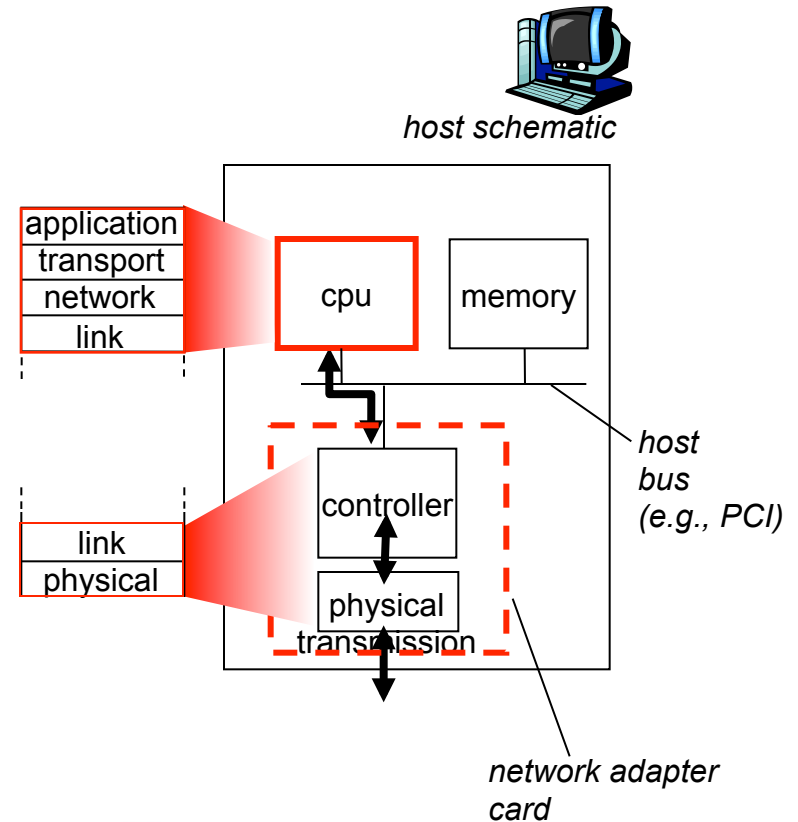
- ▶ *framing, link access:*
 - ▶ encapsulate datagram into frame, adding header, trailer
 - ▶ channel access if shared medium
 - ▶ “MAC” addresses used in frame headers to identify source, dest
 - ▶ different from IP address!
- ▶ *reliable delivery between adjacent nodes*
 - ▶ we learned how to do this already (chapter 3)!
 - ▶ seldom used on low bit-error link (fiber, some twisted pair)
 - ▶ wireless links: high error rates
 - ▶ Q: why both link-level and end-end reliability?

Link Layer Services (more)

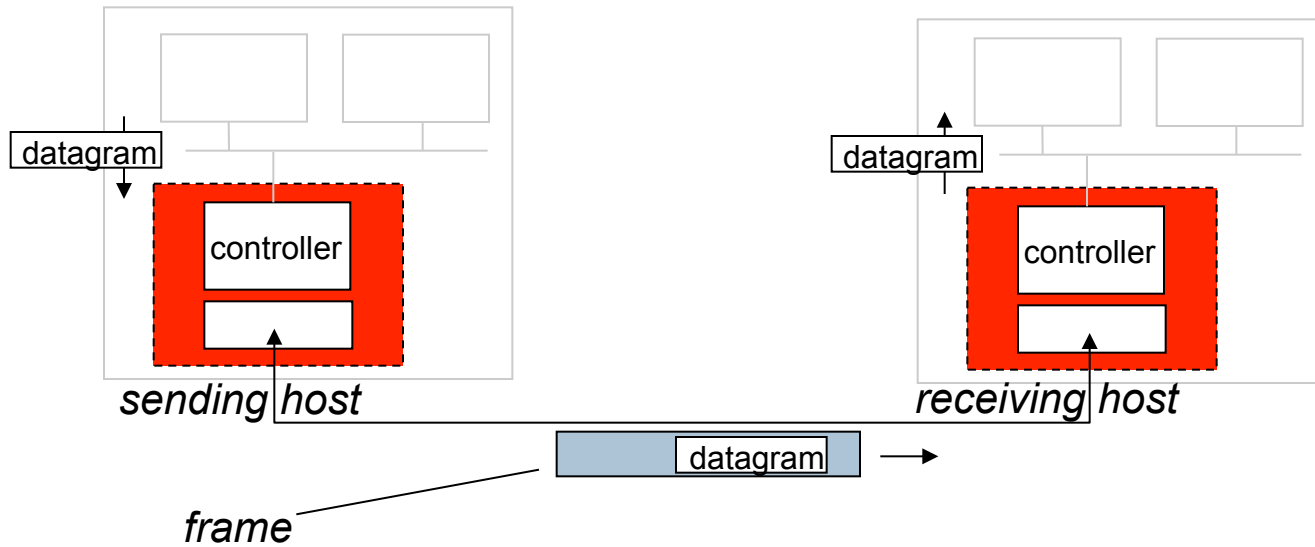
- ▶ **flow control:**
 - ▶ pacing between adjacent sending and receiving nodes
- ▶ **error detection:**
 - ▶ errors caused by signal attenuation, noise.
 - ▶ receiver detects presence of errors:
 - ▶ signals sender for retransmission or drops frame
- ▶ **error correction:**
 - ▶ receiver identifies *and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

Where is the link layer implemented?

- ▶ in each and every host
- ▶ link layer implemented in “adaptor” (aka *network interface card* NIC)
 - ▶ Ethernet card, PCMCIA card, 802.11 card
 - ▶ implements link, physical layer
- ▶ attaches into host’s system buses
- ▶ combination of hardware, software, firmware



Adaptors Communicating



▶ sending side:

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

▶ receiving side

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

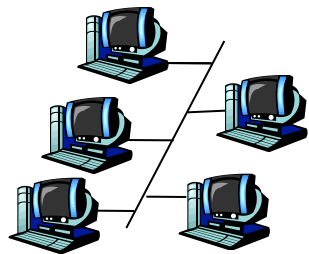
Link Layer

- ▶ 5.1 Introduction and services
- ▶ 5.2 Error detection and correction
- ▶ **5.3 Multiple access protocols**
- ▶ 5.4 Link-layer Addressing
- ▶ 5.5 Ethernet
- ▶ 5.6 Link-layer switches
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

Multiple Access Links and Protocols

Two types of “links”:

- ▶ point-to-point
 - ▶ PPP for dial-up access
 - ▶ point-to-point link between Ethernet switch and 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

Ideal Multiple Access Protocol

Broadcast channel of rate R bps

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: a taxonomy

MAC = Medium Access Control

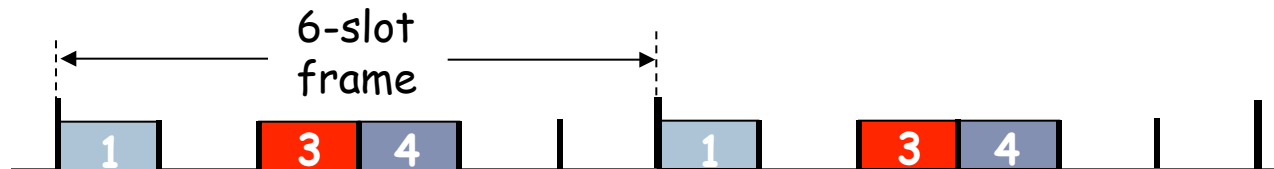
Three broad classes:

- ▶ **Channel Partitioning**
 - ▶ divide channel into smaller “pieces” (time slots, frequency, code)
 - ▶ allocate piece to node for exclusive use
- ▶ **Random Access**
 - ▶ channel not divided, allow collisions
 - ▶ “recover” from collisions
- ▶ **“Taking turns”**
 - ▶ nodes take turns, but nodes with more to send can take longer turns

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

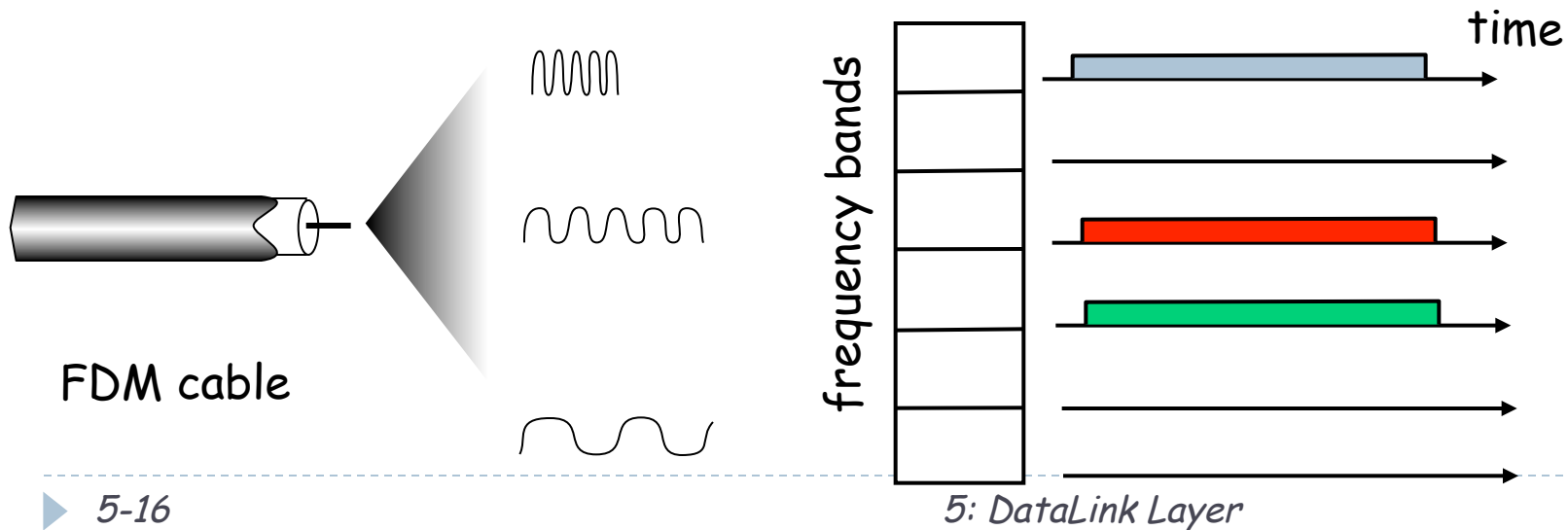
- ▶ access to channel in "rounds"
- ▶ each station gets fixed length slot (length = pkt trans time) in each round
- ▶ unused slots go idle
- ▶ example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

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



Random Access Protocols

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

Slotted ALOHA

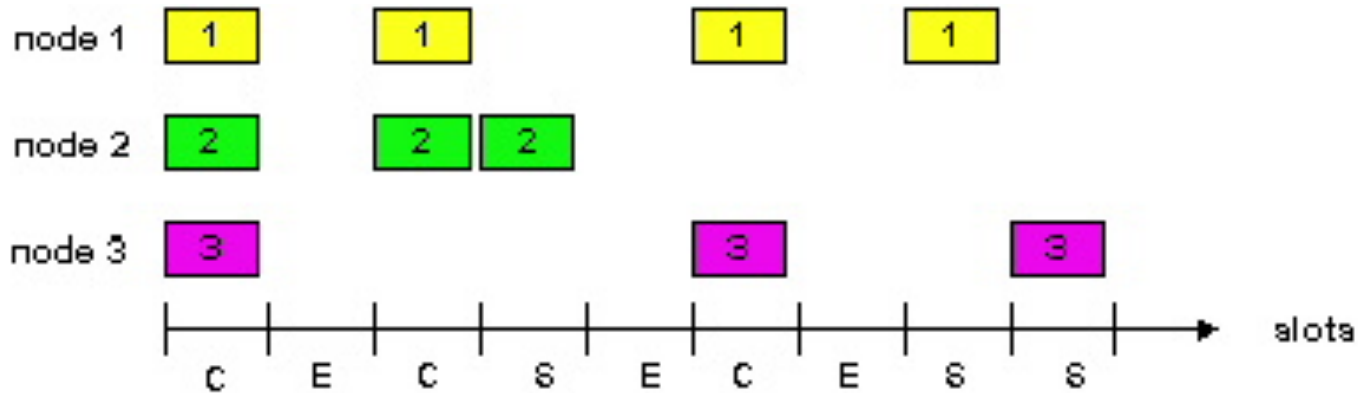
Assumptions:

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

Operation:

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

Slotted ALOHA



Pros

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

Cons

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

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:

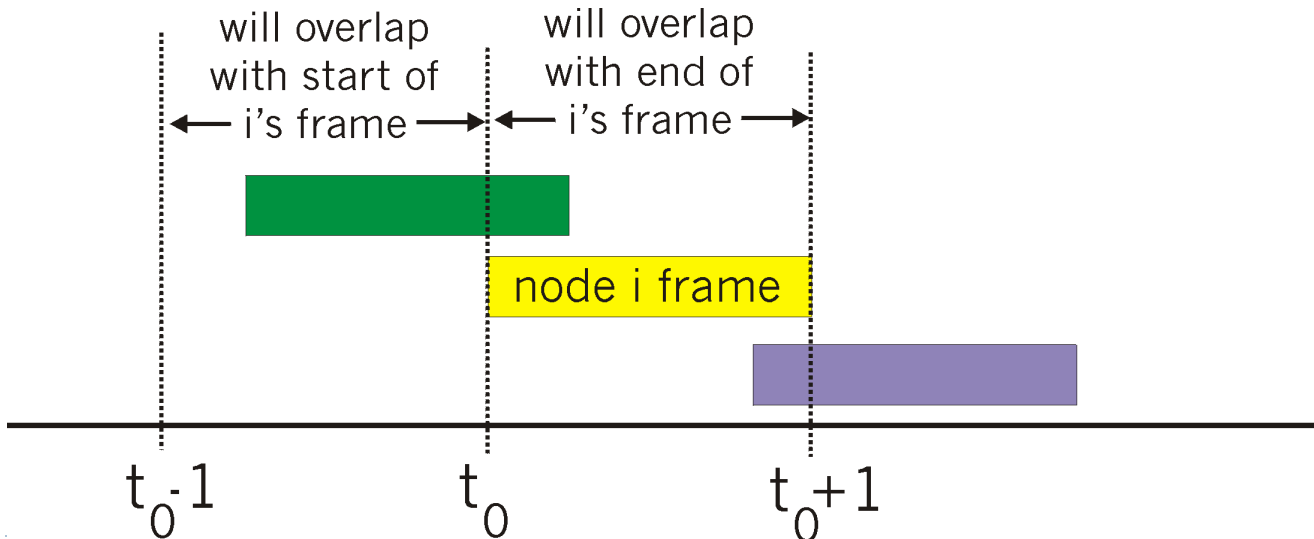
$$\text{Max efficiency} = 1/e = .37$$

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 $[t_0-1, t_0+1]$



Pure Aloha efficiency

$$\begin{aligned} P(\text{success by given node}) &= P(\text{node transmits}) \cdot \\ &\quad P(\text{no other node transmits in } [t_0-1, t_0]) \cdot \\ &\quad P(\text{no other node transmits in } [t_0, t_0+1]) \\ &= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} \\ &= p \cdot (1-p)^{2(N-1)} \end{aligned}$$

... choosing optimum p and then letting $n \rightarrow \infty$...

$$= 1/(2e) = .18$$

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!

CSMA collisions

collisions *can still occur*:

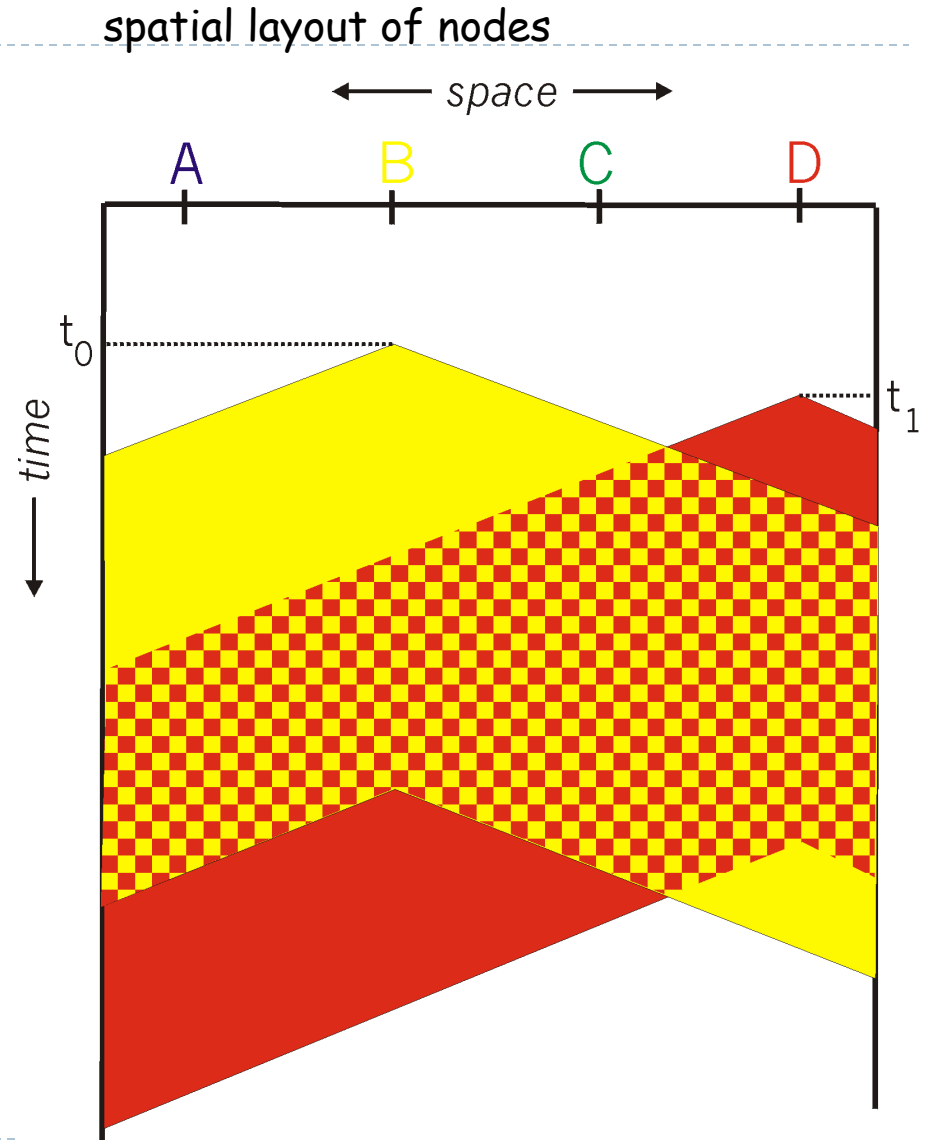
propagation delay means
two nodes may not hear
each other's transmission

collision:

entire packet transmission
time wasted

note:

role of distance & propagation
delay in determining collision
probability
"the longer the propagation
delay, the larger the chance of
collision"

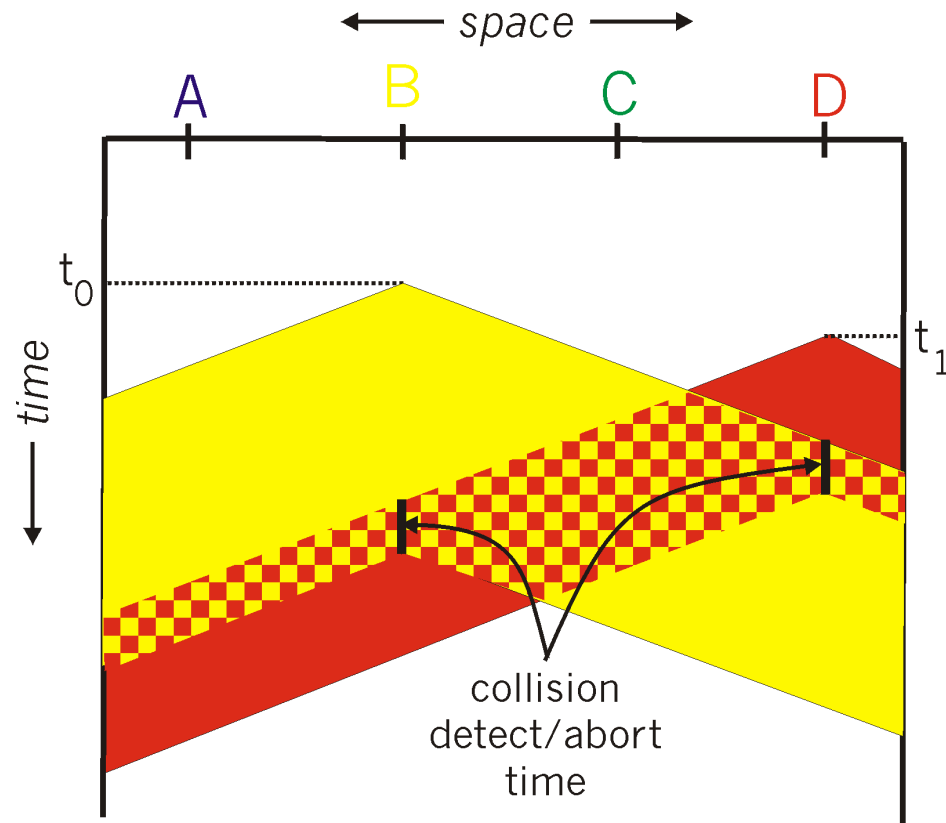


CSMA/CD (Collision Detection)

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



“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!

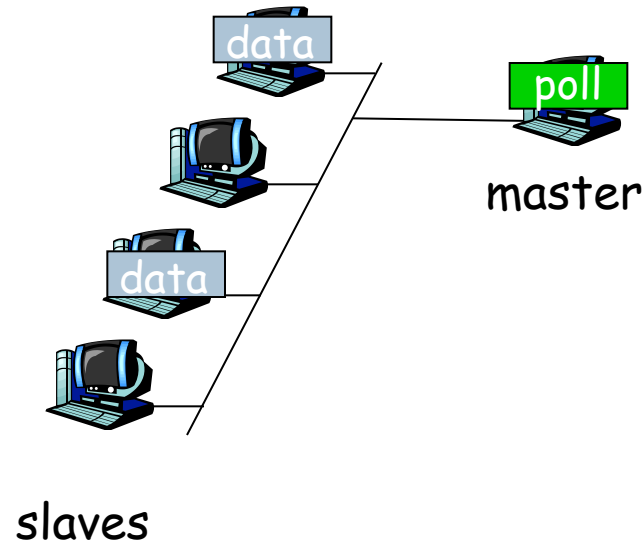
Remember the ideal requisites:

- 1) *Throughput = R bps when only one node has data to transmit*
- 2) *Throughput = R/M bps when M nodes have data to transmit*

“Taking Turns” MAC protocols

Polling:

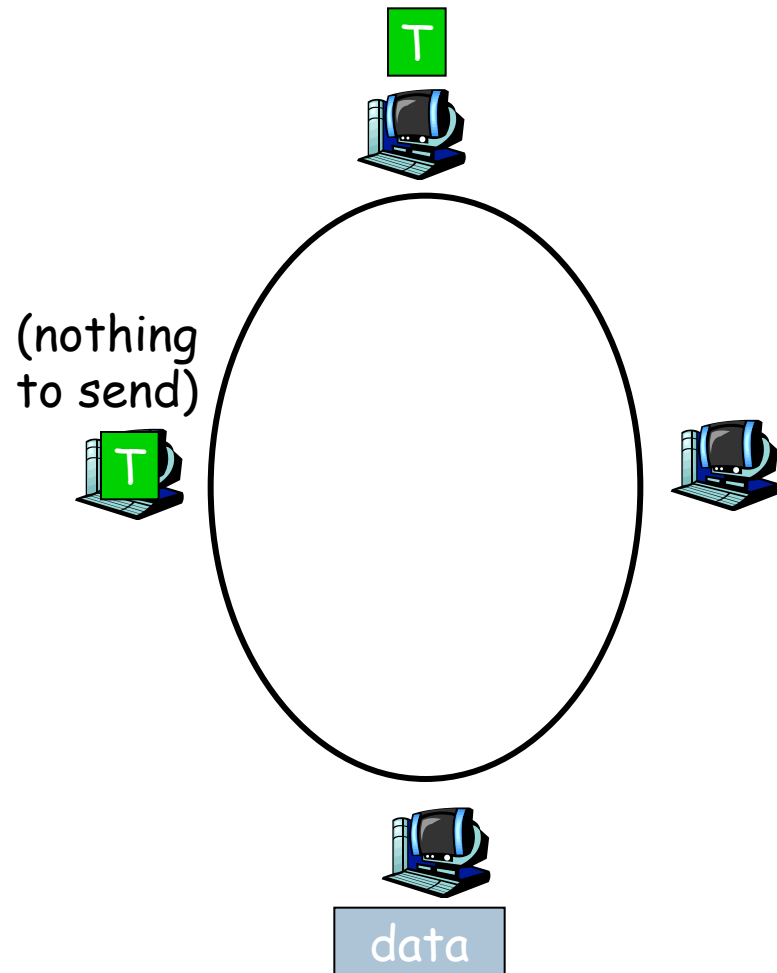
- ▶ master node “invites” slave nodes to transmit in turn
- ▶ typically used with “dumb” slave devices
- ▶ concerns:
 - ▶ polling overhead
 - ▶ latency
 - ▶ single point of failure (master)



“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)



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, IBM Token Ring

Link Layer

- ▶ 5.1 Introduction and services
- ▶ 5.2 Error detection and correction
- ▶ 5.3 Multiple access protocols
- ▶ **5.4 Link-Layer Addressing**
- ▶ 5.5 Ethernet
- ▶ 5.6 Link-layer switches
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

MAC Addresses and ARP

- ▶ **32-bit IP address:**

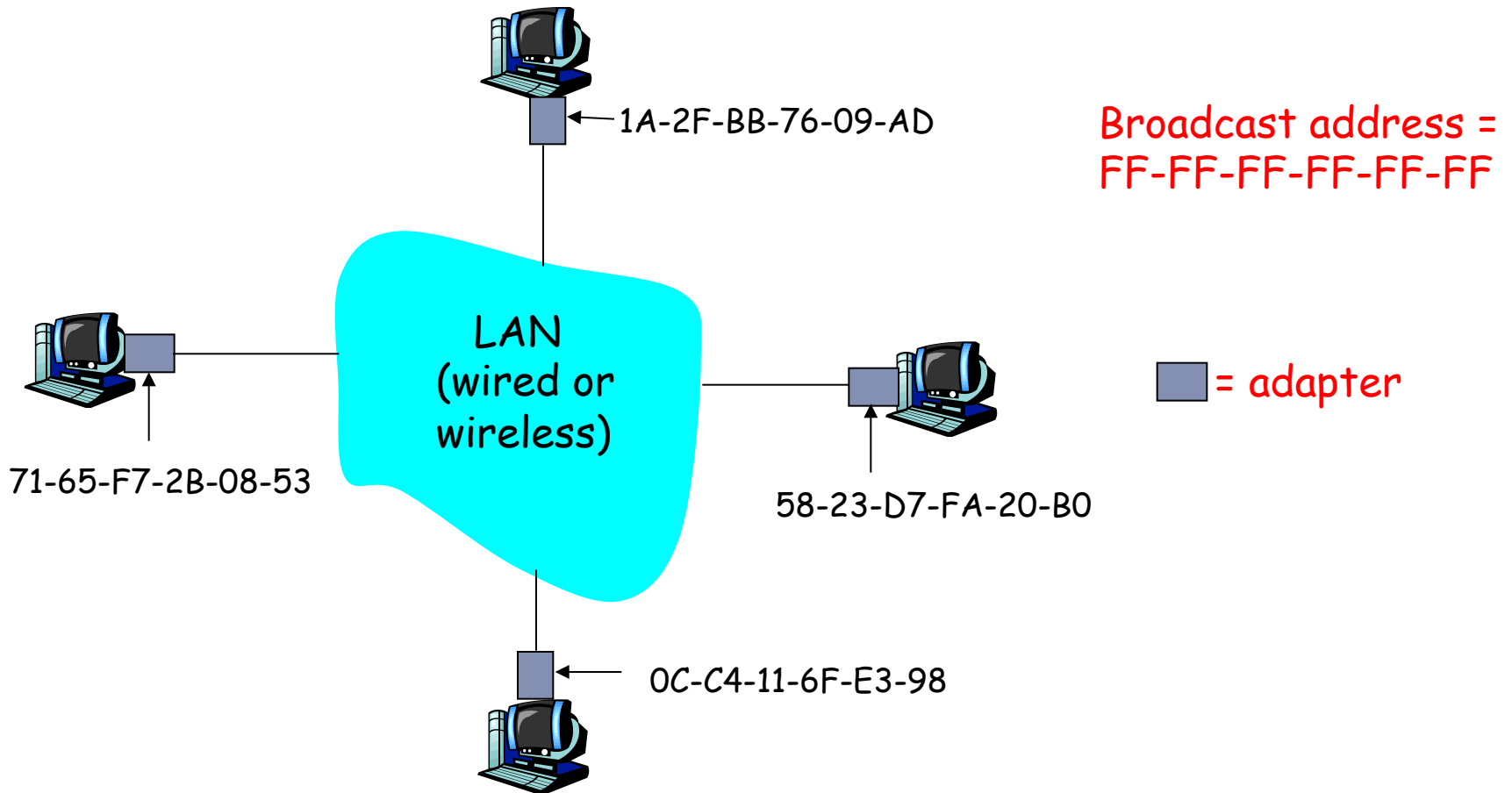
- ▶ *network-layer* address
- ▶ used to get datagram to destination IP subnet

- ▶ **MAC (or LAN or physical or Ethernet) address:**

- ▶ function: *get frame from one interface to another physically-connected interface (same network)*
- ▶ 48 bit MAC address (for most LANs)
 - ▶ 3 bytes for organization-specific prefix + 3 bytes to identify the card
 - ▶ burned in NIC ROM, also sometimes software settable

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



LAN Address (more)

- ▶ MAC address allocation administered by IEEE
- ▶ manufacturer buys portion of MAC address space (to assure uniqueness)
- ▶ analogy:
 - (a) MAC address: like Social Security Number
 - (b) IP address: like postal address
- ▶ MAC flat address → portability
 - ▶ can move LAN card from one LAN to another
- ▶ IP hierarchical address NOT portable
 - ▶ address depends on IP subnet to which node is attached

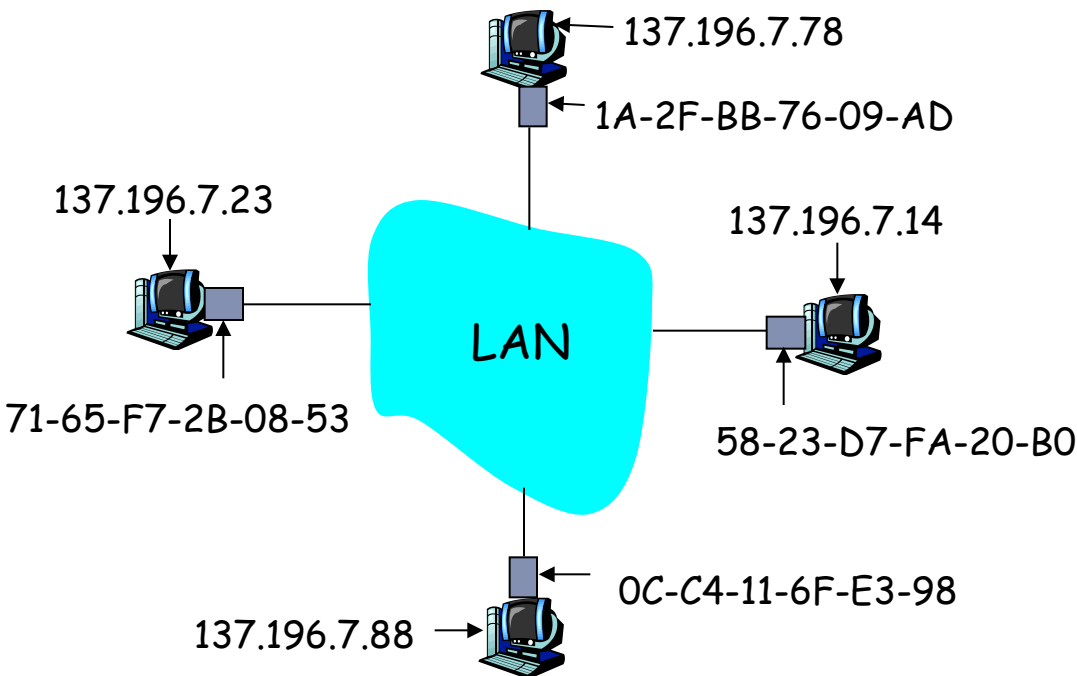
ARP: Address Resolution Protocol

Question: how to determine MAC address of B knowing B's IP address?

- ▶ Each IP node (host, router) on LAN has **ARP** table
- ▶ 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)



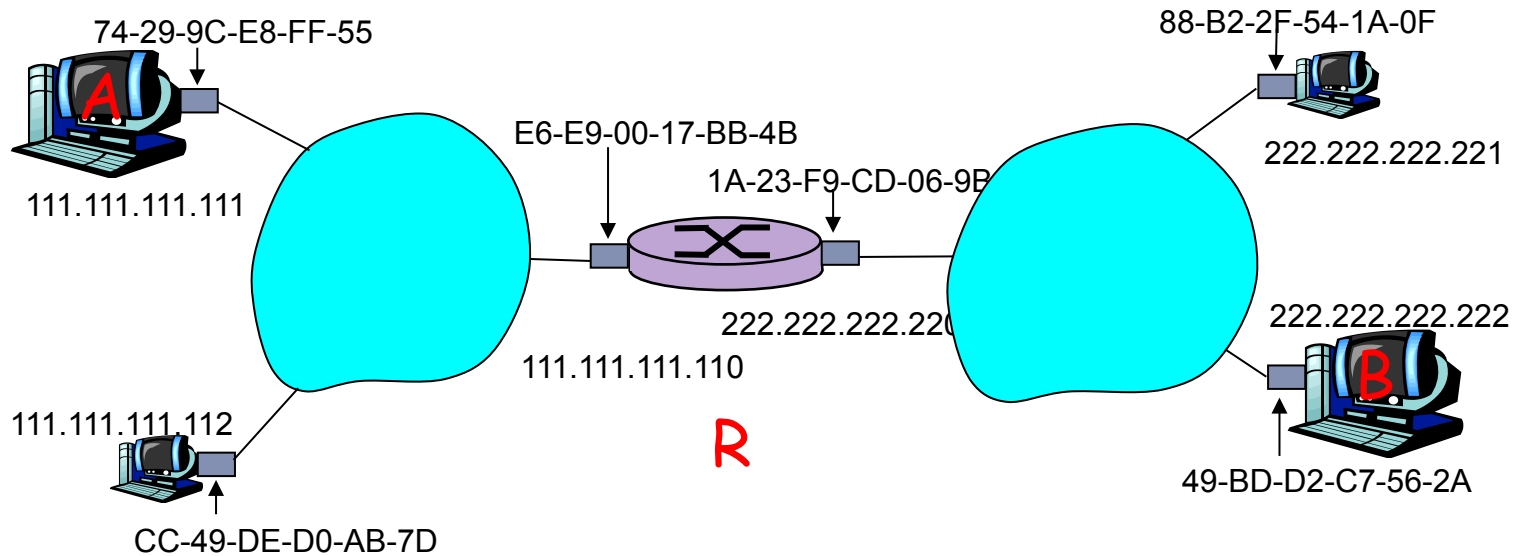
ARP protocol: Same LAN (network)

- ▶ A wants to send datagram to B, and 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 machines on LAN receive ARP query
- ▶ B receives ARP packet, replies to A with its (B's) MAC address
 - ▶ frame sent to A's MAC address (unicast)
- ▶ A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
 - ▶ soft state: information that times out (goes away) unless refreshed
- ▶ ARP is “plug-and-play”:
 - ▶ nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

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

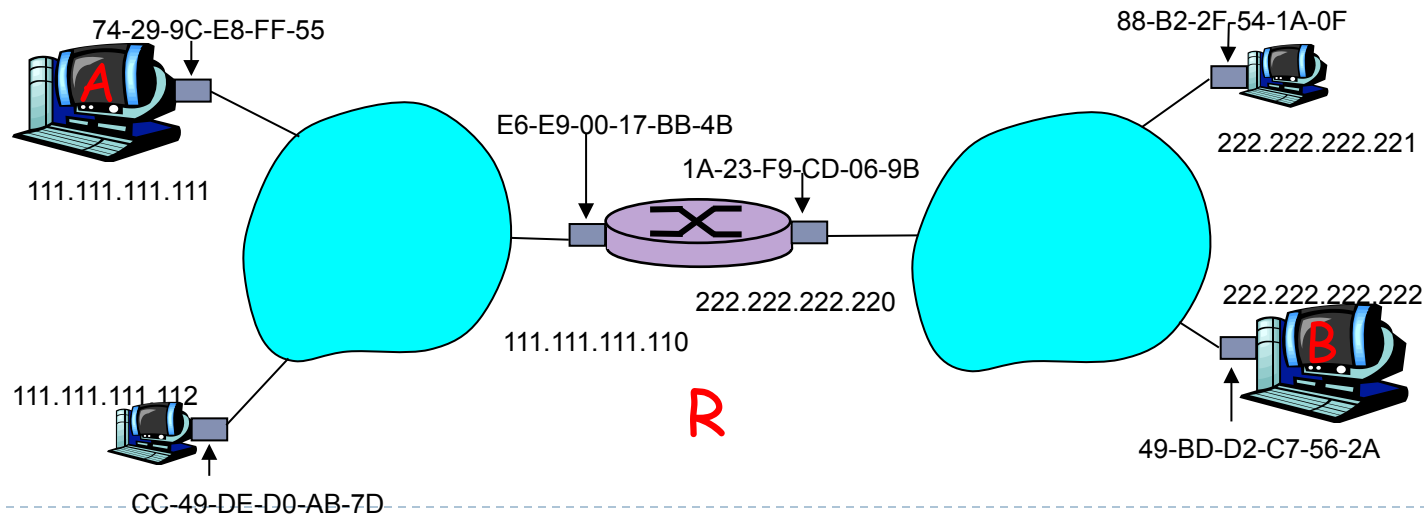
assume A knows B's IP address



- ▶ two ARP tables in router R, one for each IP network (LAN)

- ▶ A creates IP datagram with source A, destination B
 - ▶ A uses ARP to get R's MAC address for 111.111.111.110
-
- ▶ A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
 - ▶ A's NIC sends frame
 - ▶ R's NIC receives frame
 - ▶ R removes IP datagram from Ethernet frame, sees its destined to B
 - ▶ R uses ARP to get B's MAC address
 - ▶ R creates frame containing A-to-B IP datagram sends to B

This is a **really** important example - make sure you understand!



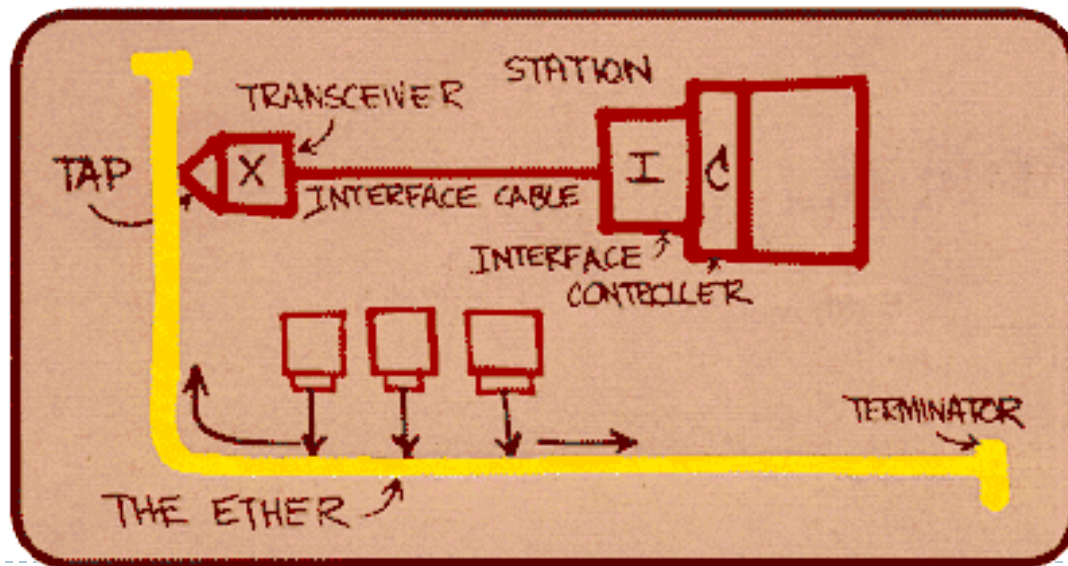
Link Layer

- ▶ 5.1 Introduction and services
- ▶ 5.2 Error detection and correction
- ▶ 5.3 Multiple access protocols
- ▶ 5.4 Link-Layer Addressing
- ▶ **5.5 Ethernet**
- ▶ 5.6 Link-layer switches
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

Ethernet

“dominant” wired LAN technology:

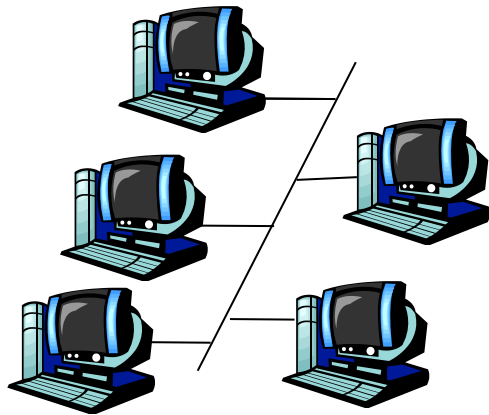
- ▶ cheap \$20 for NIC
- ▶ first widely used LAN technology
- ▶ simpler, cheaper than token LANs and ATM
- ▶ kept up with speed race: 10 Mbps – 10 Gbps



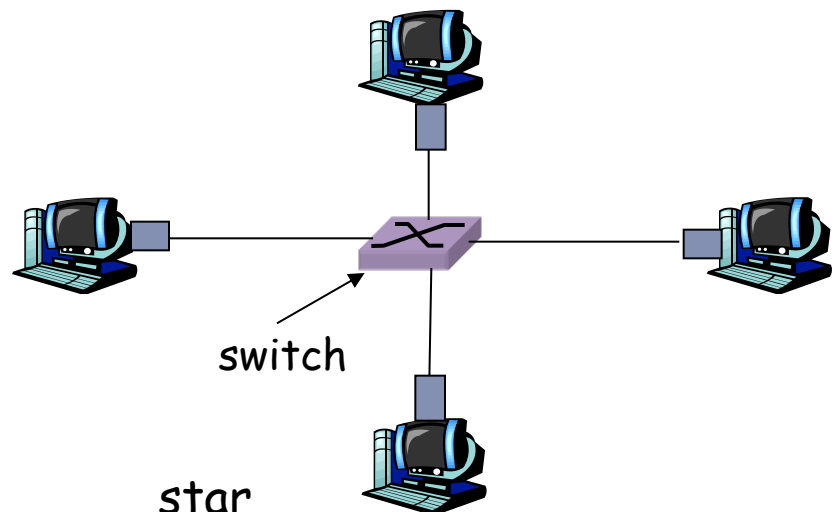
Metcalfe's Ethernet sketch

Star topology***

- ▶ bus topology popular through mid 90s
 - ▶ all nodes in same collision domain (can collide with each other)
- ▶ today: star topology prevails
 - ▶ active *switch* in center
 - ▶ each node runs a (separate) Ethernet protocol (nodes do not collide with each other)



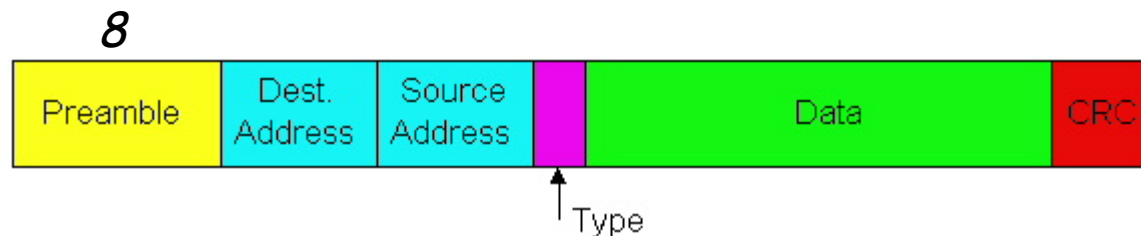
bus: coaxial cable



star

Ethernet Frame Structure

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

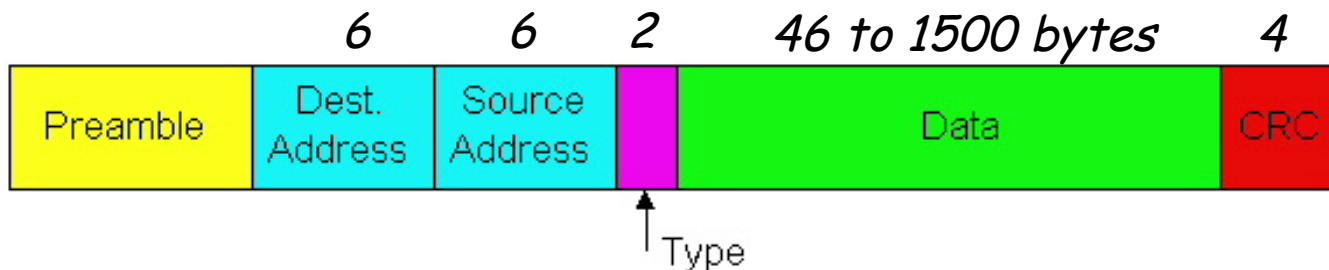


Preamble:

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

Ethernet Frame Structure (more)

- ▶ **Addresses:** 6 bytes
 - ▶ if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network layer protocol
 - ▶ otherwise, adapter discards frame
- ▶ **Type:** indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- ▶ **CRC:** checked at receiver, if error is detected, frame is dropped



Ethernet: Unreliable, connectionless

- ▶ **connectionless**: No handshaking between sending and receiving NICs
- ▶ **unreliable**: receiving NIC doesn't send acks or nacks to sending NIC
 - ▶ stream of datagrams passed to network layer can have gaps (missing datagrams)
 - ▶ gaps will be filled if app is using TCP
 - ▶ otherwise, app will see gaps
- ▶ Ethernet's MAC protocol: unslotted **CSMA/CD**

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 **exponential backoff**: after m th collision, NIC chooses K at random from $\{0, 1, 2, \dots, 2^m - 1\}$ (max $m=10$). NIC waits $K \cdot 512$ bit times, returns to Step 2

Ethernet's CSMA/CD (more)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits

Bit time: .1 microsec for 10 Mbps Ethernet ;
for $K=1023$, wait time is about 50 msec

Exponential Backoff:

- ▶ *Goal:* adapt retransmission attempts to estimated current load
 - ▶ heavy load: random wait will be longer
- ▶ first collision: choose K from $\{0,1\}$; delay is $K \cdot 512$ bit transmission times
- ▶ after second collision: choose K from $\{0,1,2,3\}$...
- ▶ after ten collisions, choose K from $\{0,1,2,3,4,\dots,1023\}$

CSMA/CD efficiency

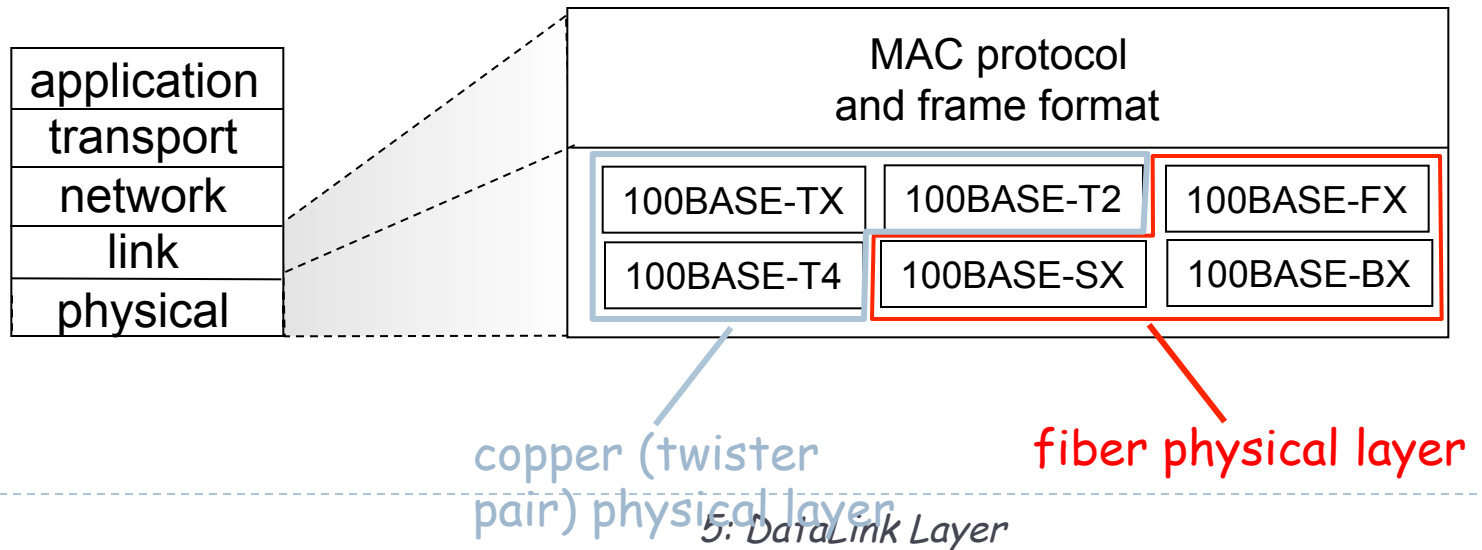
- ▶ T_{prop} = max prop delay between 2 nodes in LAN
- ▶ t_{trans} = time to transmit max-size frame

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

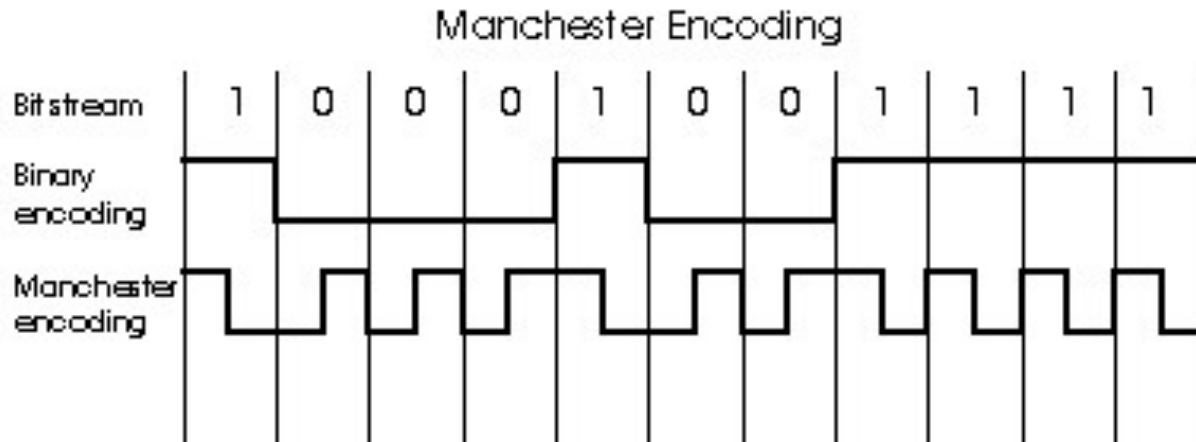
$$\text{efficiency} = \frac{1}{1 + 5t_{\text{prop}}/t_{\text{trans}}}$$

802.3 Ethernet Standards: Link & Physical Layers

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



Manchester encoding



- ▶ used in 10BaseT
- ▶ each bit has a transition
- ▶ allows clocks in sending and receiving nodes to synchronize to each other
 - ▶ no need for a centralized, global clock among nodes!
- ▶ **Hey, this is physical-layer stuff!**

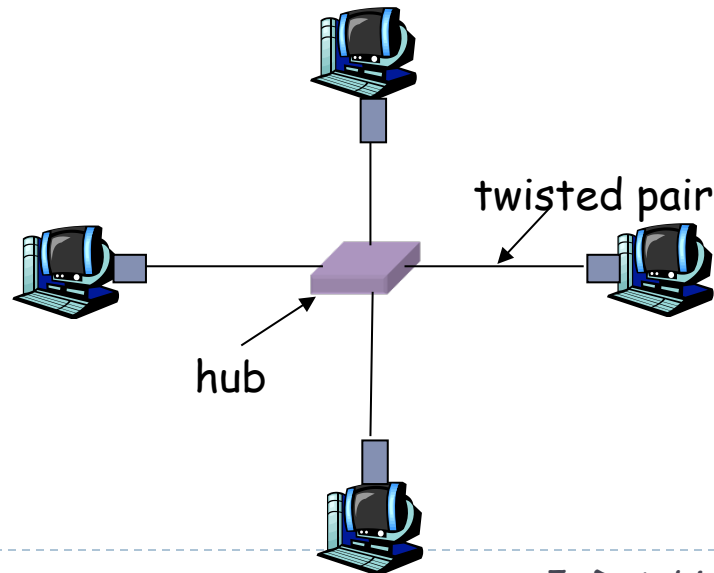
Link Layer

- ▶ 5.1 Introduction and services
- ▶ 5.2 Error detection and correction
- ▶ 5.3 Multiple access protocols
- ▶ 5.4 Link-layer Addressing
- ▶ 5.5 Ethernet
- ▶ 5.6 Link-layer switches, LANs, VLANs
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

Hubs

... physical-layer (“dumb”) repeaters:

- ▶ bits coming in one link go out *all* other links at same rate
- ▶ all nodes connected to hub can collide with one another
- ▶ no frame buffering
- ▶ no CSMA/CD at hub: host NICs detect collisions

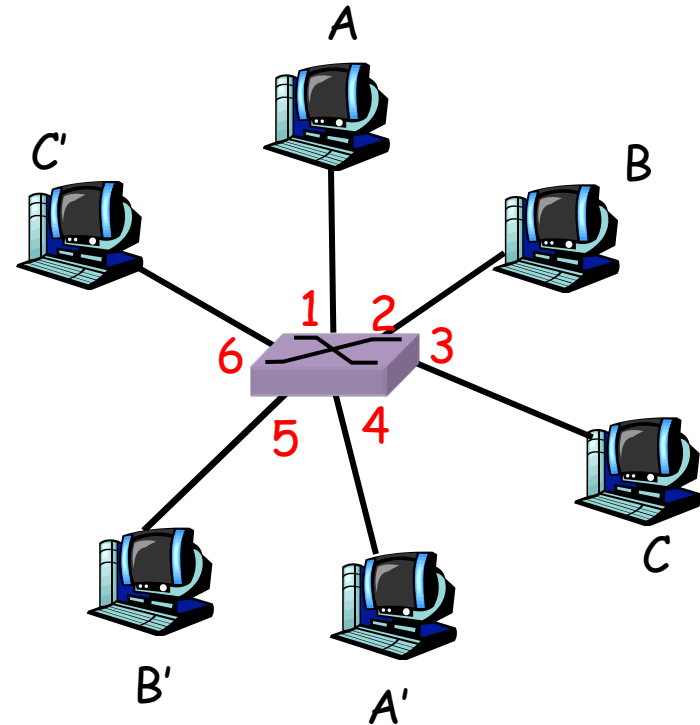


Switch

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

Switch: allows *multiple* simultaneous transmissions

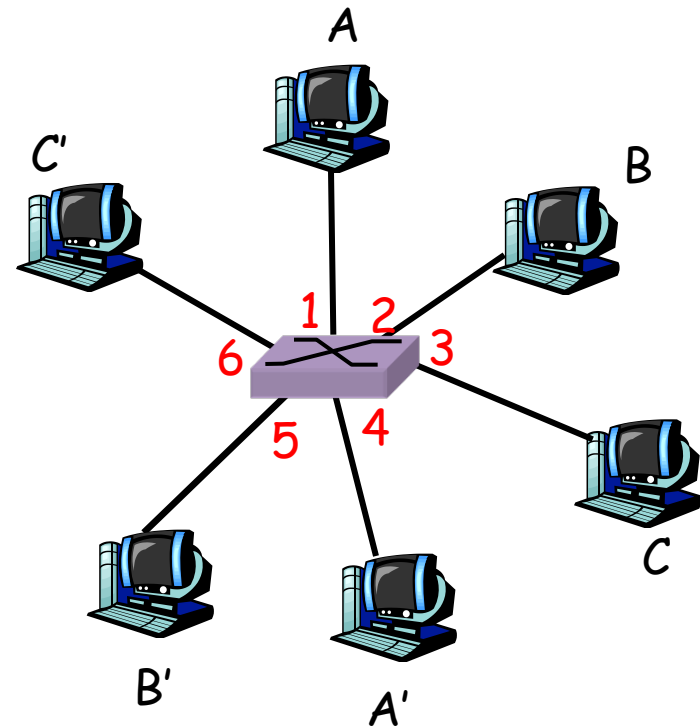
- ▶ hosts have dedicated, direct connection to switch
- ▶ switches buffer packets
- ▶ Ethernet protocol used on *each* incoming link, but no collisions; full duplex
 - ▶ each link is its own collision domain
- ▶ **switching**: A-to-A' and B-to-B' simultaneously, without collisions
 - ▶ not possible with dumb hub



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

Switch Table

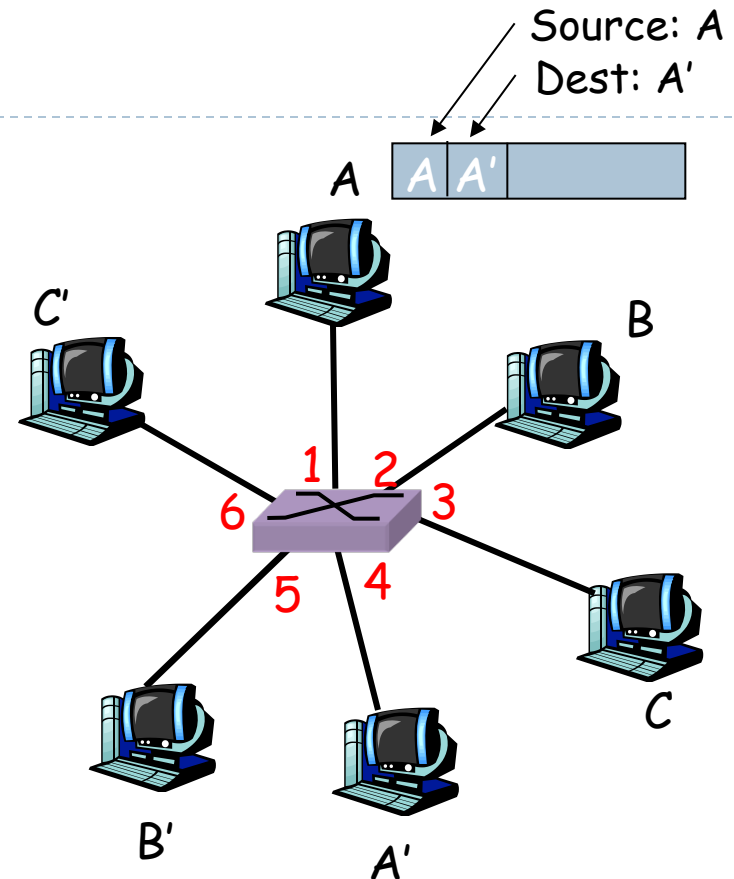
- ▶ **Q:** how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- ▶ **A:** each switch has a **switch table**, each entry:
 - ▶ (MAC address of host, interface to reach host, time stamp)
- ▶ looks like a routing table!
- ▶ **Q:** how are entries created, maintained in 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:

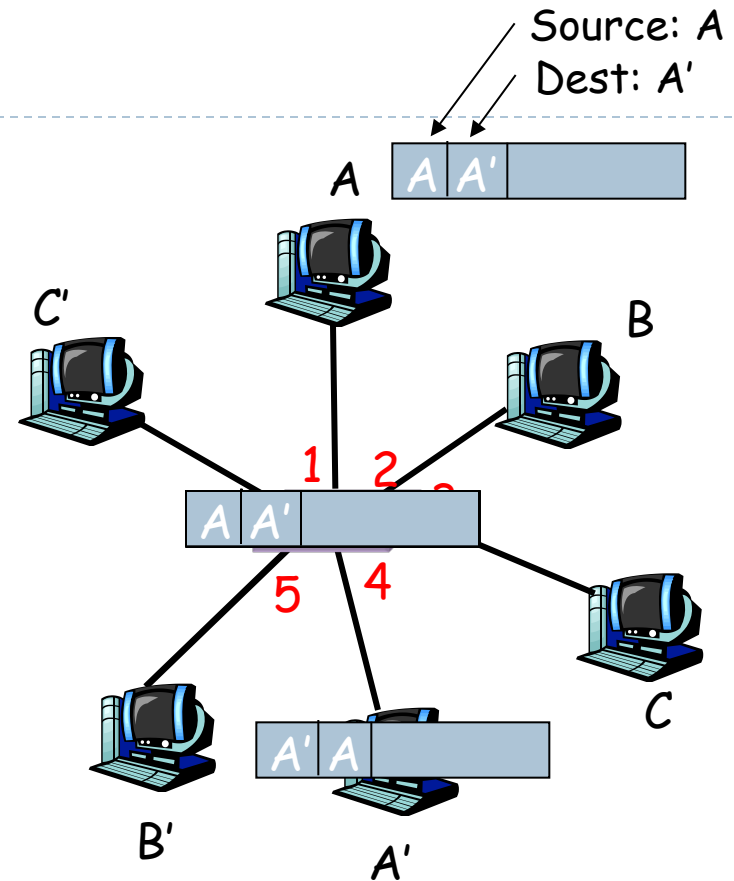
1. record link associated with sending host
2. index switch table using MAC dest address
3. **if** entry found for destination
 then {
 if dest on segment from which frame arrived
 then drop the frame
 else forward the frame on interface indicated
 }
 else flood



*forward on all but the interface
on which the frame arrived*

Self-learning, forwarding: example

- ▶ frame destination unknown: *flood*
- destination A location known: *selective send*

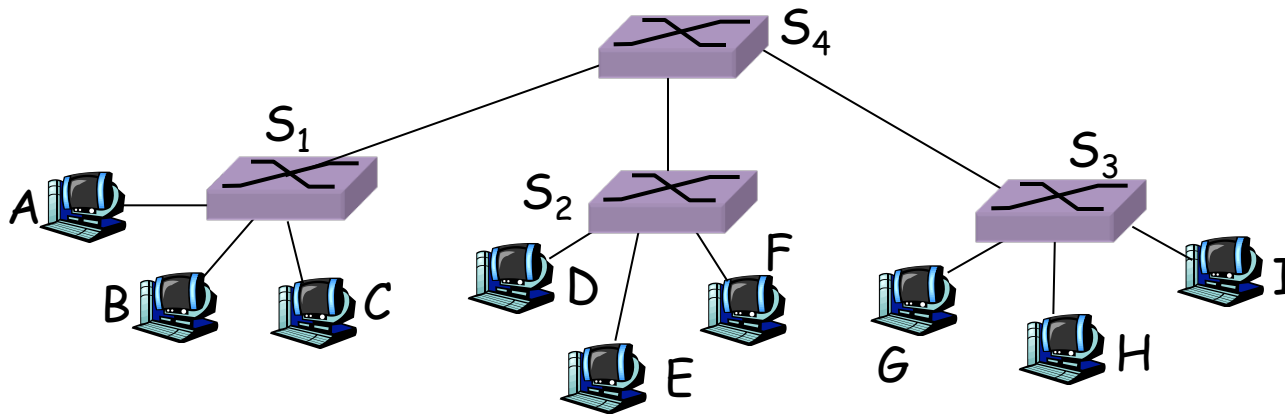


MAC addr	interface	TTL
A	1	60
A'	4	60

*Switch table
(initially empty)*

Interconnecting switches

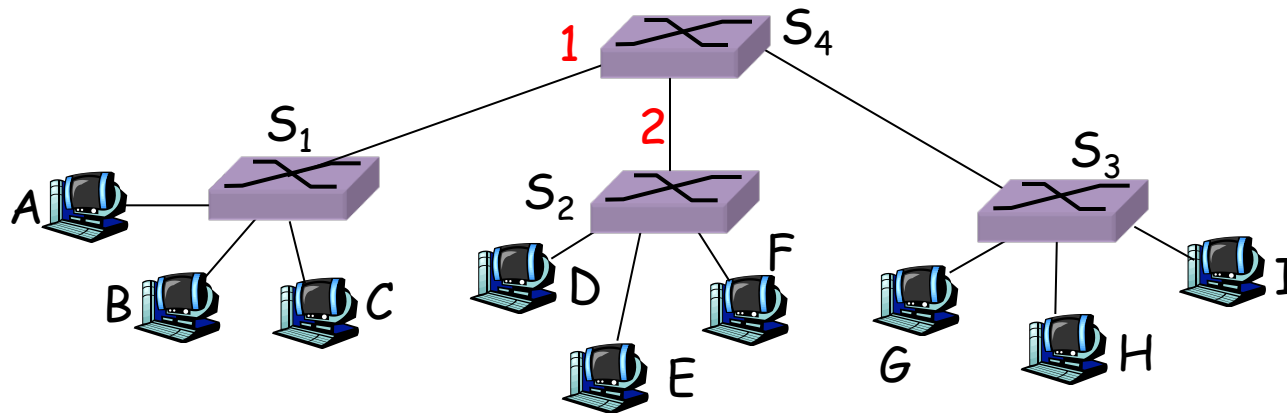
- ▶ switches can be connected together



- ❑ **Q:** sending from A to G - how does S₁ know to forward frame destined to F via S₄ and S₃?
- ❑ **A:** self learning! (works exactly the same as in single-switch case!)

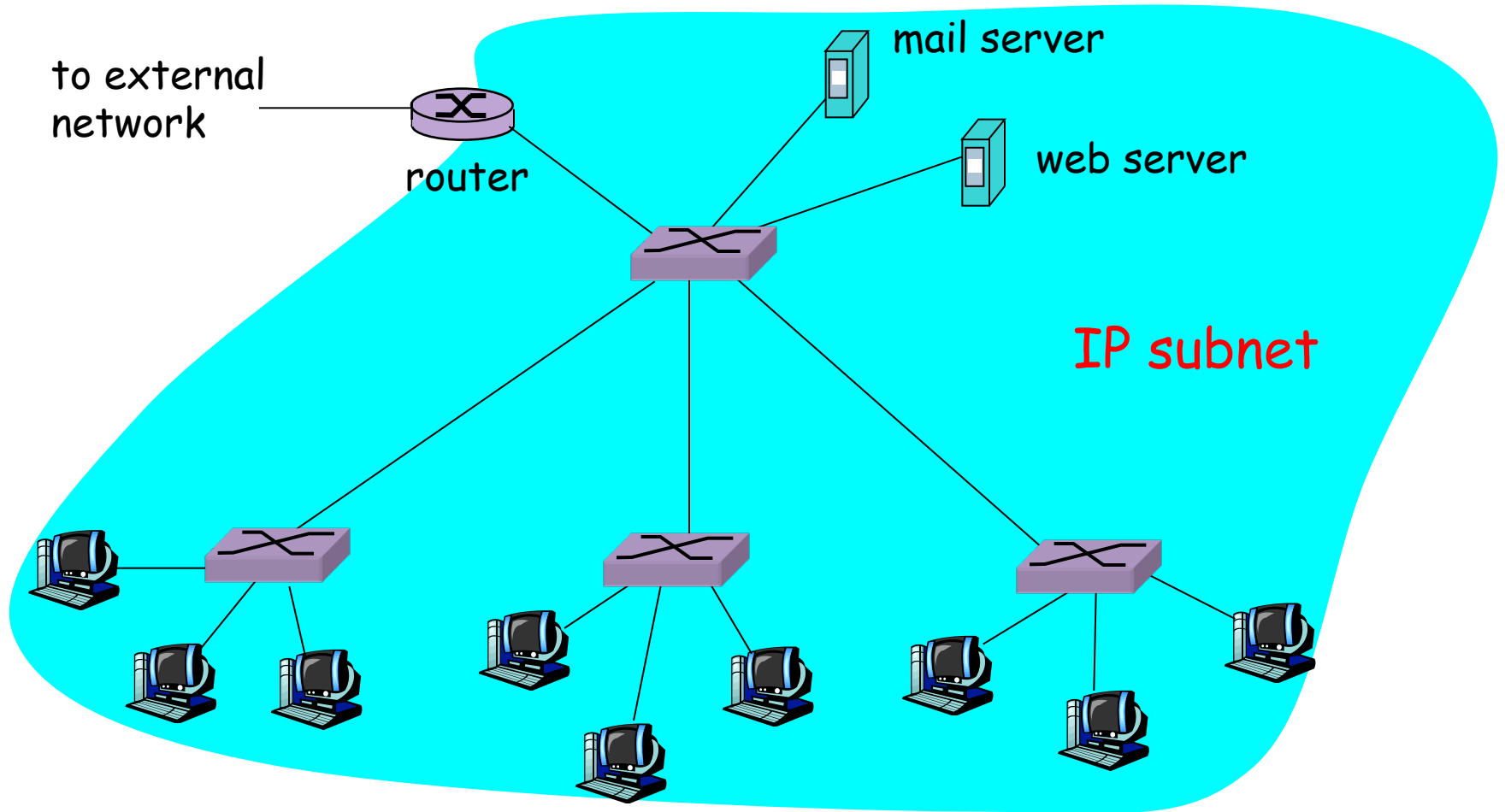
Self-learning multi-switch example***

Suppose C sends frame to I, I responds to C



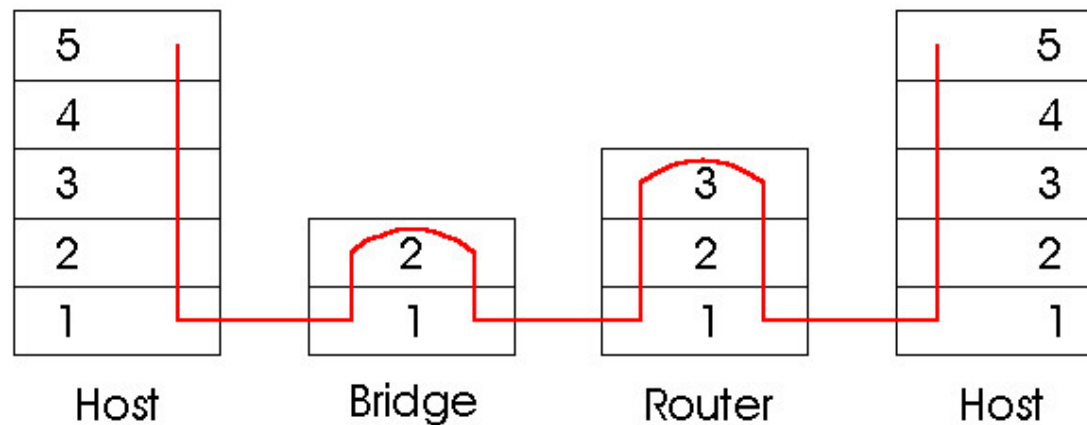
- Q: show switch tables and packet forwarding in S₁, S₂, S₃, S₄

Institutional network



Switches vs. Routers

- ▶ both store-and-forward devices
 - ▶ routers: network layer devices (examine network layer headers)
 - ▶ switches are link layer devices
- ▶ routers maintain routing tables, implement routing algorithms
- ▶ switches maintain switch tables, implement filtering, learning algorithms



Switch vs. ARP Poisoning

▶ Switch Poisoning

- ▶ Send Eth packets with spoofed src-MAC to the switch
- ▶ Objective: fill the MAC-to-NIC map
- ▶ Result: switch gets flooded, all frames will be *broadcasted* and therefore can be **sniffed!**

▶ ARP Poisoning

- ▶ Can be more targeted
- ▶ Objective: Poison the ARP table of a host X
- ▶ How? Attacker Y sends lots of spoofed ARP packets saying that the MAC address of the default gateway is actually Y's MAC
- ▶ Result: **Man-in-the-Middle Attack!**

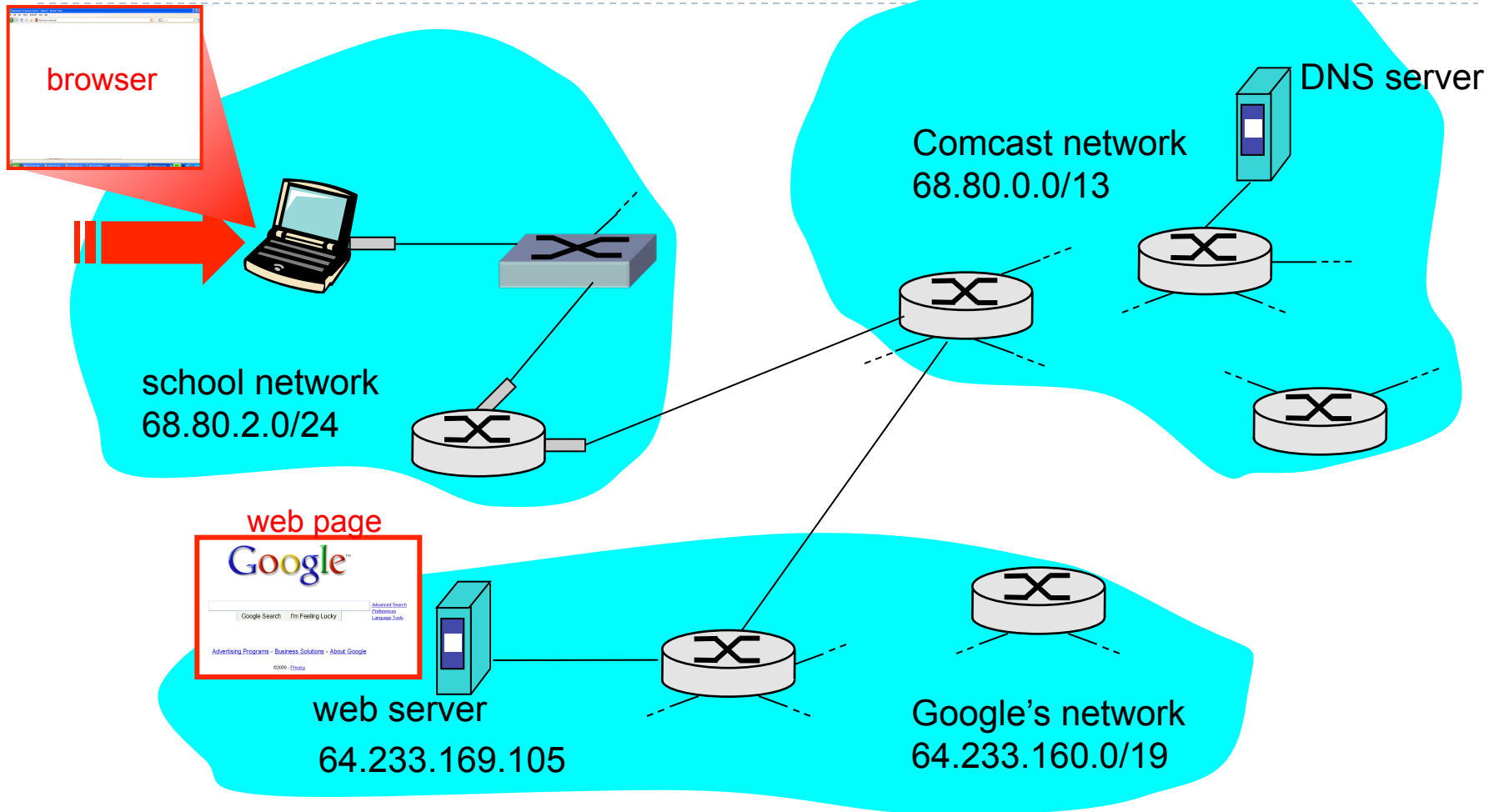
Link Layer

- ▶ 5.1 Introduction and services
- ▶ 5.2 Error detection and correction
- ▶ 5.3 Multiple access protocols
- ▶ 5.4 Link-Layer Addressing
- ▶ 5.5 Ethernet
- ▶ 5.6 Link-layer switches
- ▶ 5.7 PPP
- ▶ 5.8 Link virtualization: MPLS
- ▶ 5.9 A day in the life of a web request

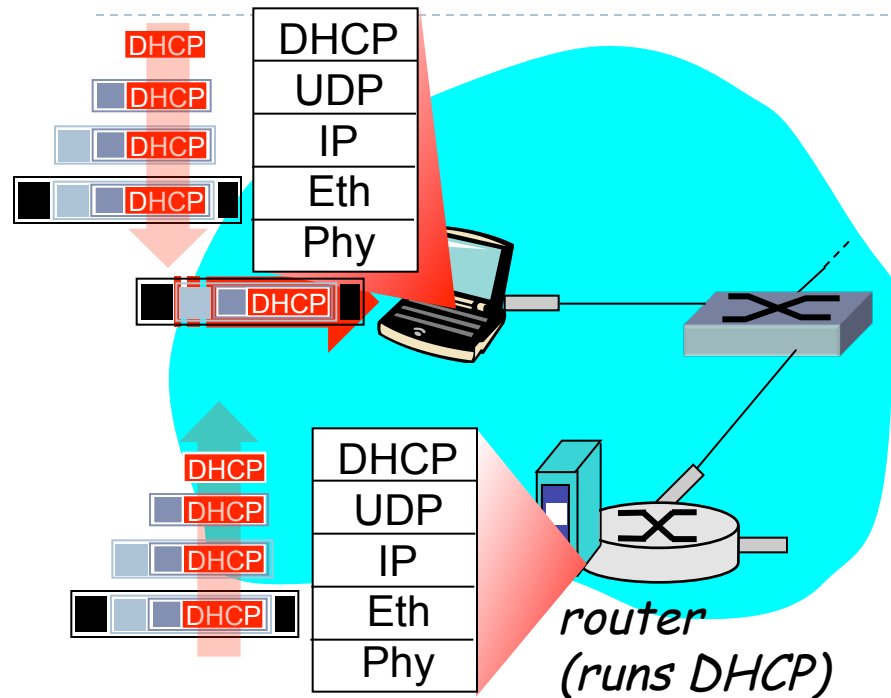
Synthesis: a day in the life of a web request

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

A day in the life: scenario

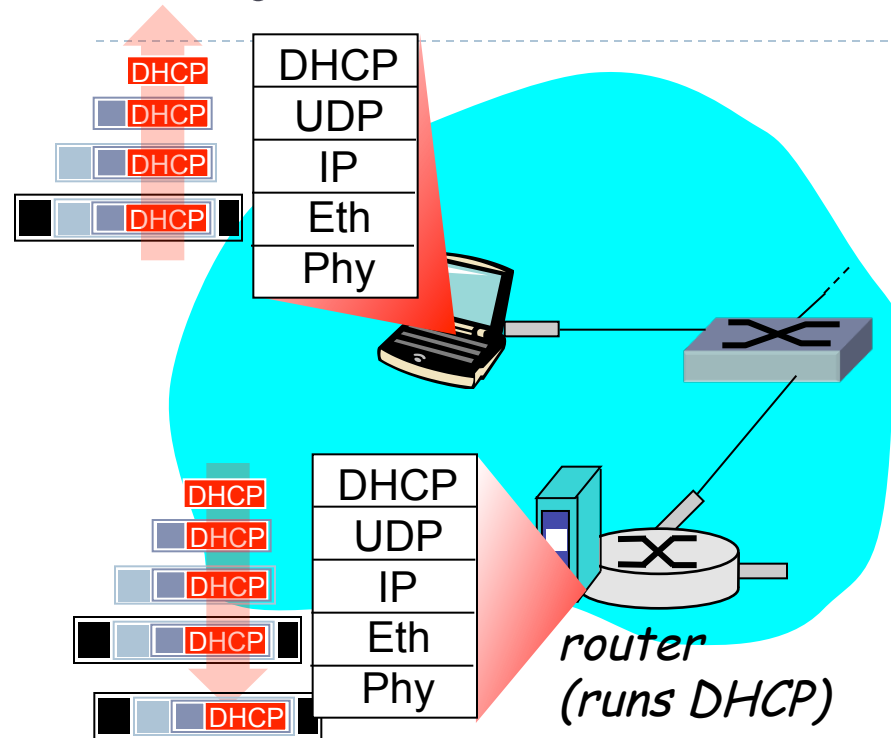


A day in the life... connecting to the Internet



- ▶ connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**
- DHCP request *encapsulated* in **UDP**, encapsulated in **IP**, encapsulated in **802.1** Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet *demux'ed* to IP demux'ed, UDP demux'ed to DHCP

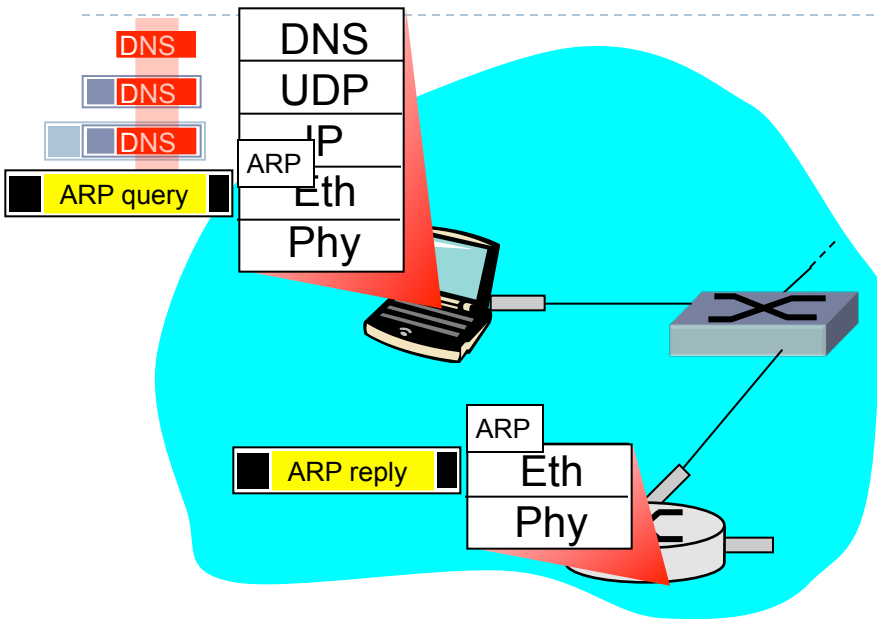
A day in the life... connecting to the Internet



- ▶ 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

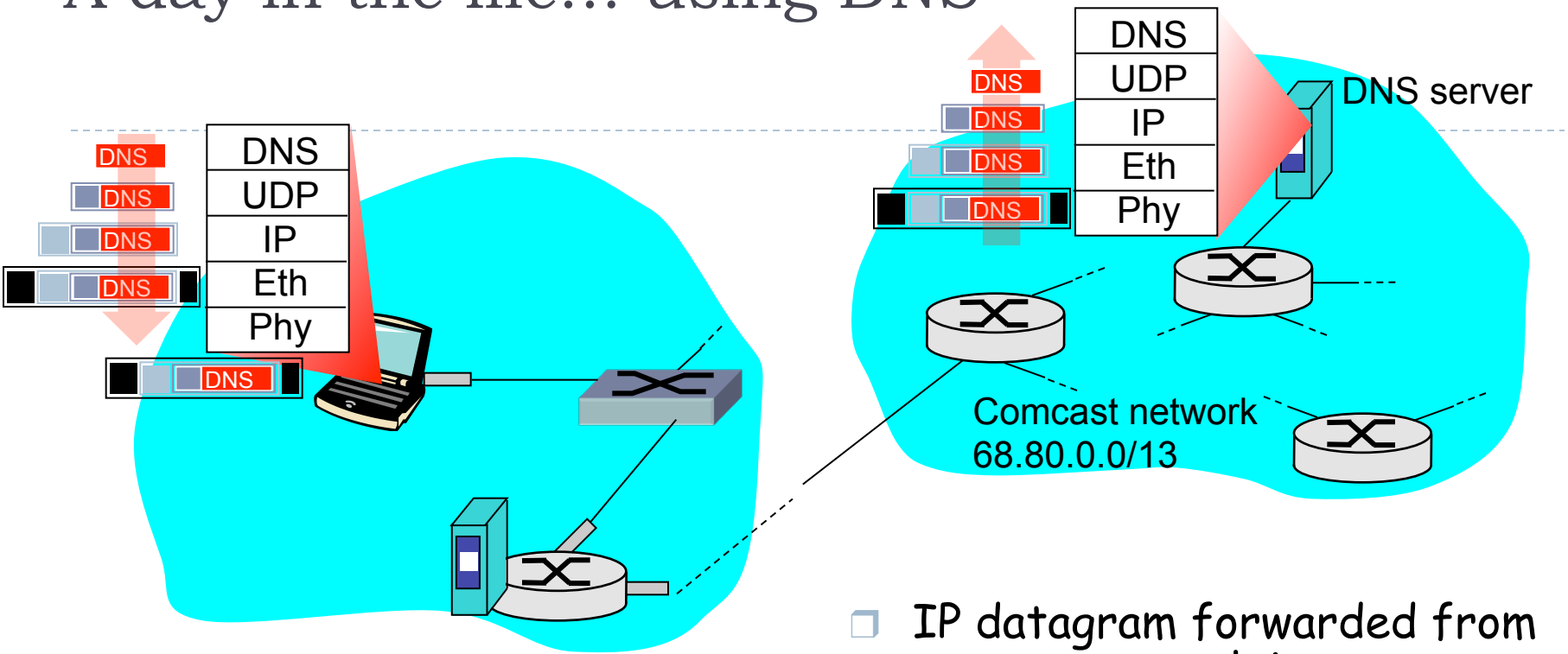
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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. In order to send frame to router, need MAC address of router interface: **ARP**
- **ARP query** broadcast, received by router, which replies with **ARP reply** giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life... using DNS

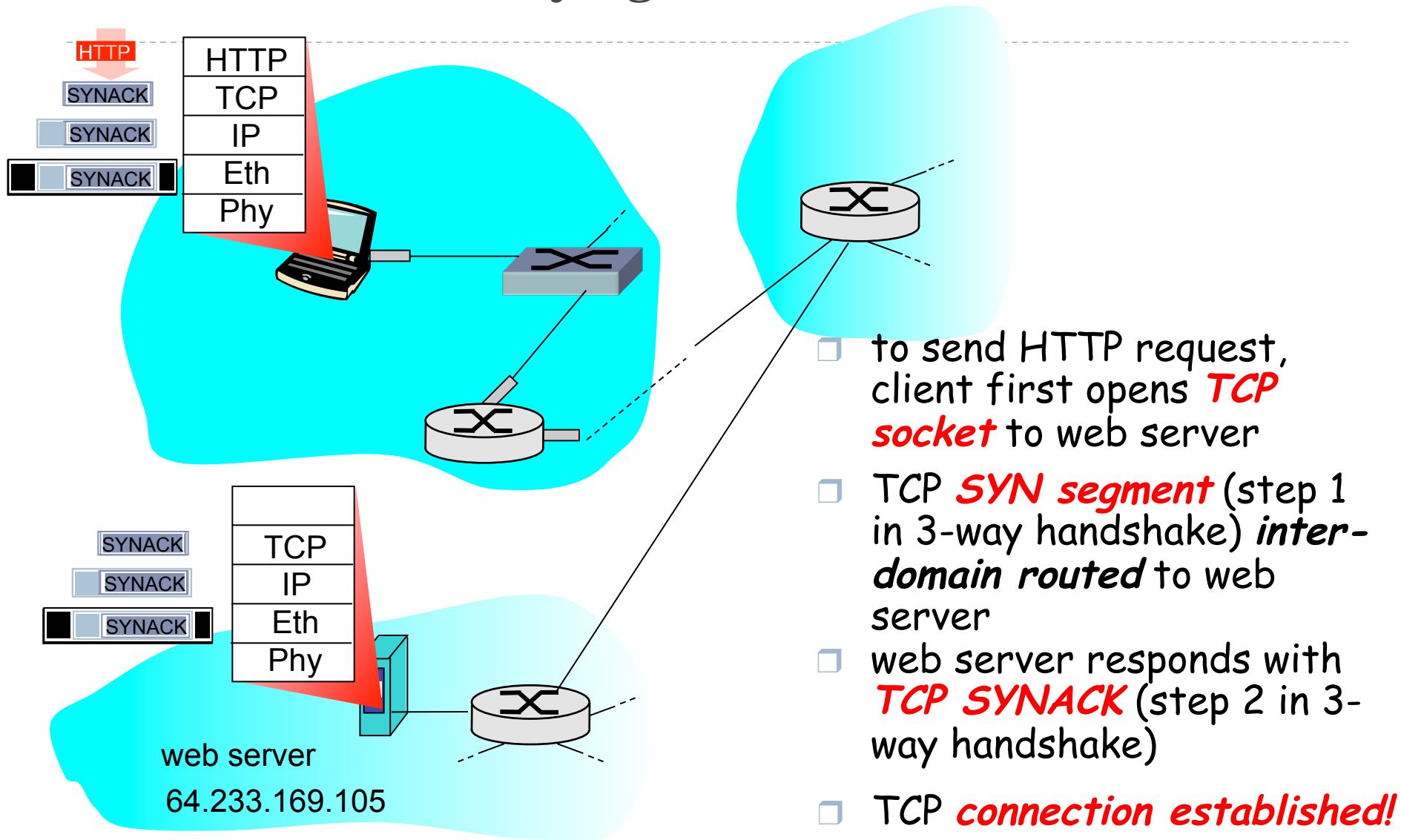


- ❑ IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

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

A day in the life...

TCP connection carrying HTTP



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

