Chapter 5
Data Link Layer

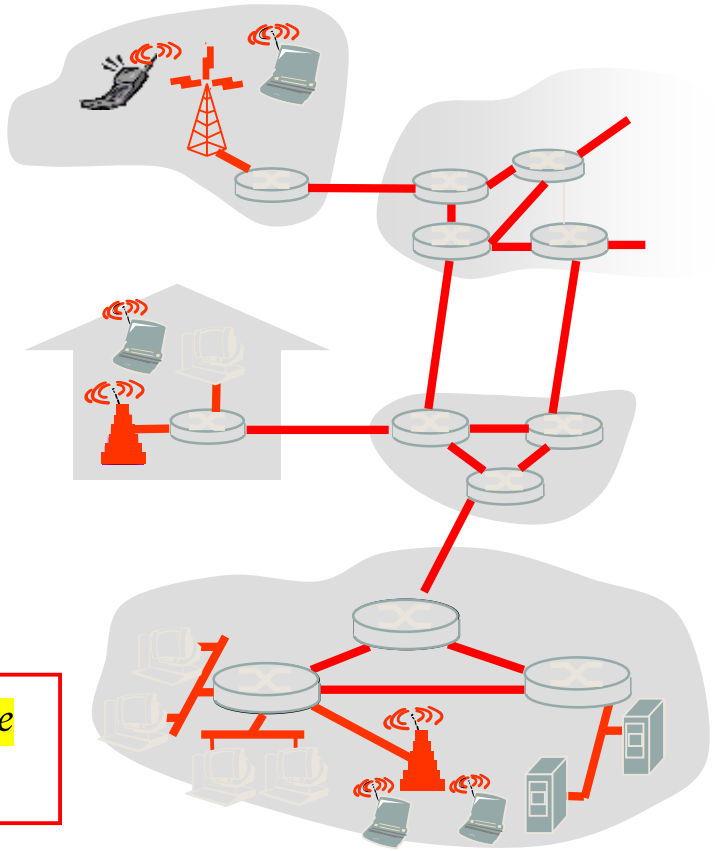
Chapter outlines

- Introduction
- Error detection and correction
- Multiple access protocols
- Link-layer Addressing
- Ethernet
- Link-layer switches
- PPP
- A day in the life of a web request

1. Introduction

Some terminology:

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



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

Internet layers

Layer	Packets	Services	Protocols
Application	Data (messages)	Provide services to Internet users (web, file transfer, email,…), form data chunks	HTTP, FTP, SMTP, POP3, IMAP, DNS,…..
Transport	Segment	Form segments, buffering, mux, demux, reliability, flow control, congestion control	TCP, UDP
Network	Datagrams	encapsulation to form datagrams, forwarding, routing	IP, RIP, OSPF, BGP, ….
Link	Frames	framing, link access, reliable delivery, error detection & correction, …	Ethernet, 802.11, PPP, ….

Internet layers

Layer	Where is the protocol implemented?	How is the protocol implemented?	Service basis
Application	host	Software as a part of OS	client-to-server Per-to-peer End-to-end
Transport	host	Software as a part of OS	Process-to-process through virtual (logical) direct path
Network	host and routers	Software, routing algorithms	Host-to-host through a physical network path
Link	network adaptors of nodes	A combination of hardware and software	adaptor-to-adaptor Node-to-node through a link

Transportation Analogy

- Trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram/frame
- travel path = route
- travel agent = routing algorithm
- transport segment = communication link
- transport mode (limo,plane,train)= link layer protocol

Link layer Protocols

- Define packets format (frames), messages exchanged, and the actions (services) taken during packets transfer
- Link layer protocols provide different services:
  - e.g., error detection, retransmission, flow control, and random access
- Different Protocols:
  - e.g., Ethernet, 802.11 WiFi, token ring, and PPP
- A frame is transferred by different link layer 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

Link Layer Services

- *framing*
  - encapsulate datagram into frame, adding header, trailer
- *link access*
  - Medium Access Control (MAC) protocol: specifies the rules by which a frame is transmitted onto the link
  - Two types of links: point-to-point links and broadcast link
  - Point-to-point: single sender at one end of the link and single receiver at the other end
  - Broadcast link: multiple nodes share the link (multiple access problem)
  - Channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest (different from IP address!)

Link layer service (more)

- *reliable delivery between adjacent nodes:*
  - we learned how to do this already (chapter 5)!
  - similar to that provided by TCP in transport layer (ACK, retransmission)
  - seldom used on low bit-error link (fiber, coax, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?
    - A: to provide local error correction on the link
- *flow control:*
  - Similar to transport layer, prevent the sending node on onside of the link from overwhelming the receiving node on the other side of the link by sending too many frames to fast

Link Layer Services (more)

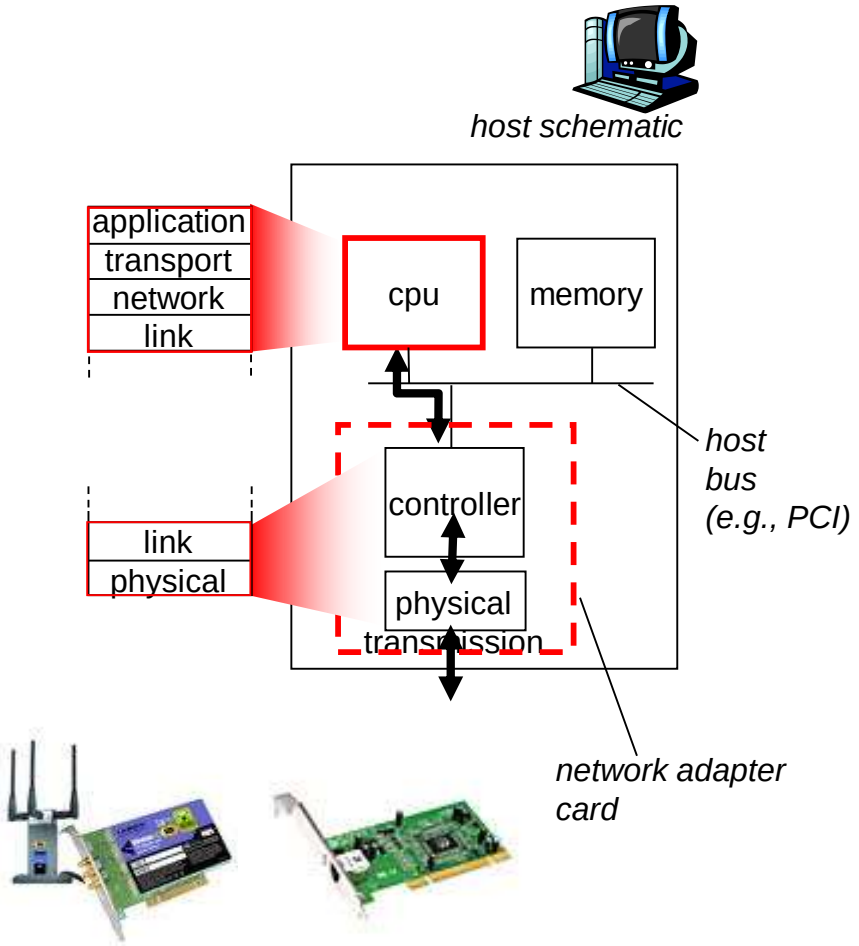
- *error detection*:
  - Bit errors is caused by signal attenuation and noise
  - Sender sends error detection bit in the frame
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
  - More sophisticated than check sum technique and hardware implemented
- error correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission

Link layer services (more)

- *half-duplex and full-duplex*
  - With half duplex, nodes at both ends of link can transmit, but not at same time
  - With full duplex, nodes at both ends of link may transmit packets at same time
- Reliability, flow control, and error detection services are provided by transport layer and link layer
  - Transport layer provides services between **two processes** on an end-to-end basis
  - Link layer provides services on a link between **two adjacent nodes** on a node-to-node basis

Where is the link layer implemented?

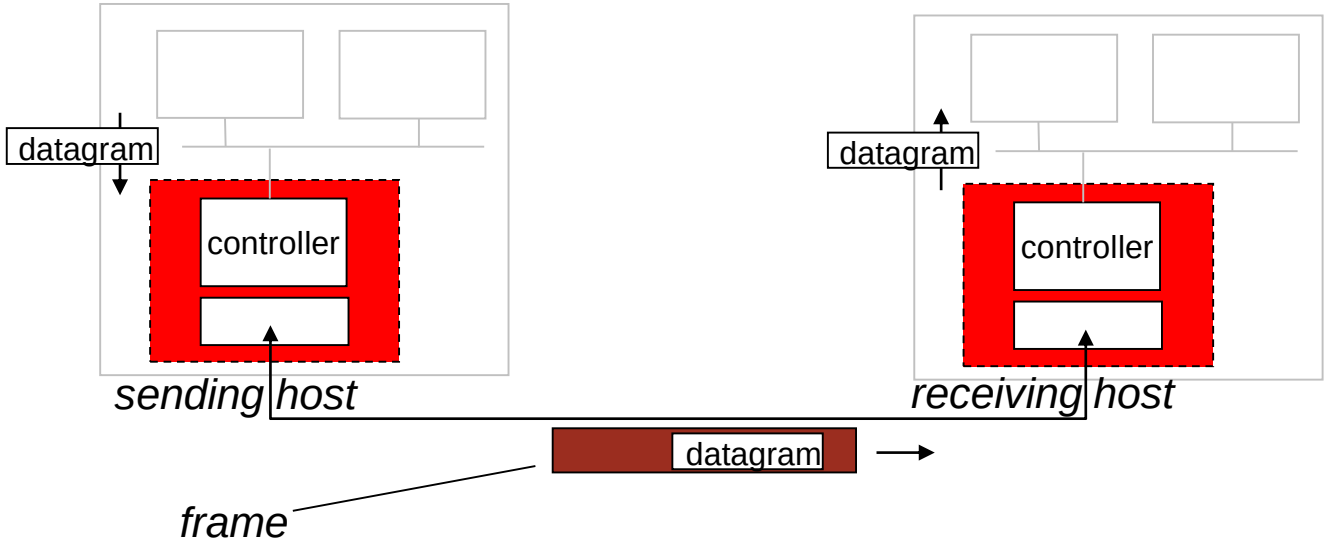
- In each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - Ethernet card, PCMCI card, 802.11 card
  - implements link, physical layer
- Controller is the heart of NIC
- Attaches into host's system buses



Where is the link layer implemented?

- Controller is a special purpose chip that implements most of link layer services
- Examples: Intel's 8524x implements Ethernet protocol
- Most of link layer services are hardware implemented
- Combination of hardware and software
- Link layer is the place in the protocol stack where SW meets HW
- HW implementation is faster than SW implementation

Adaptors Communicating

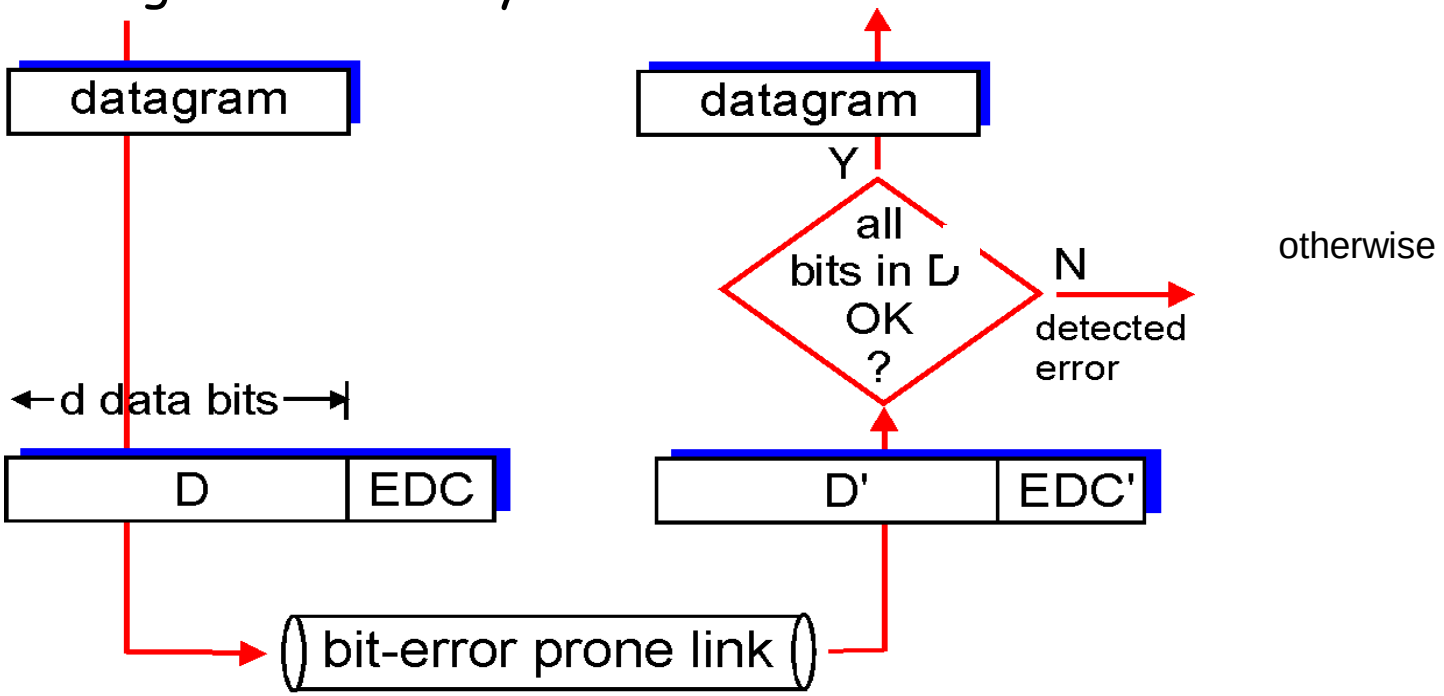


- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

2. Error Detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields
Error detection not 100% reliable!

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



Error detection protocols

- Parity checks
  - Simple protocol
  - Single bit-parity or two dimension bit-parity
- Checksum method
  - Used in transport layer
  - Software implemented
- Cyclic Redundancy Check (CRC)
  - Used in link layer
  - Hardware implemented

Parity Checking

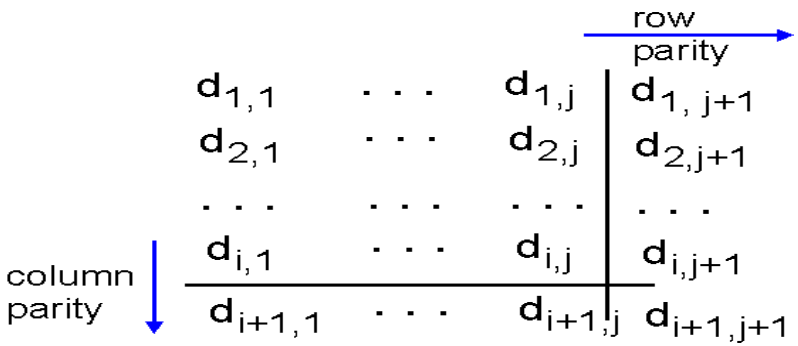
Single Bit Parity:

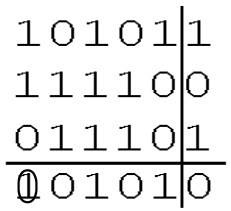
Detect single bit errors
Can not correct errors

0111000110101011 1
data even parity

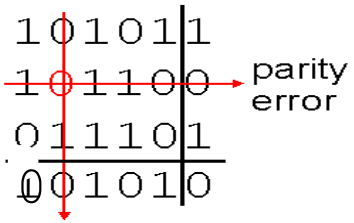
Two Dimensional Bit Parity:

Detect *and correct* single bit errors
Detect but not correct multiple errors





no errors



correctable single bit error

Internet checksum (review)

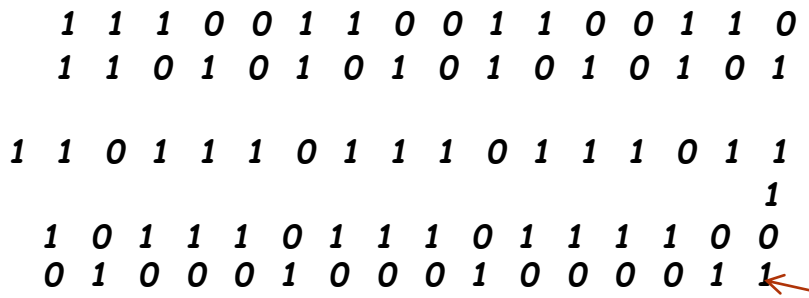
Goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer *only*)

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: 1's complement of the sum of 16-bit integers
- sender puts checksum value into UDP,TCP checksum field

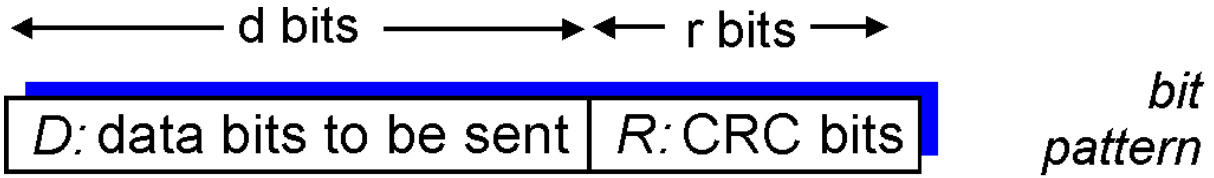
Receiver:

- compute the sum of received 16-bit integers sequence
- add computed sum to checksum field value
- Check whether the result is all 1 bits
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*



Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- choose r CRC bits, R, such that <D,R> exactly divisible by G
- sender sends <D,R>
- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



bit pattern

$\langle D,R \rangle = D * 2^r \text{ XOR } R$

mathematical formula

How to choose R

Want: <D,R> divisible by G

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

equivalently:

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

equivalently:

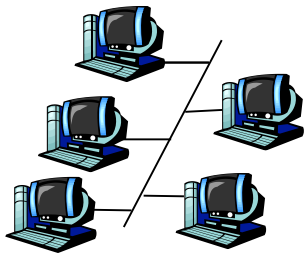
if we divide $D \cdot 2^r$ by G,
want remainder R

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

3. Multiple Access Links and Protocols

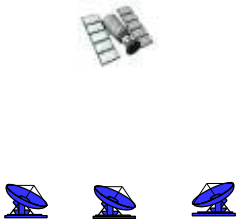
Two types of "links":

- point-to-point
  - PPP for dial-up access
  - High-level data link control (HDLC)
- broadcast (shared broadcast channel)
  - Ethernet --- LAN
  - 802.11 --- wireless LAN
  - Shared RF satellite



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 Problem

- single shared broadcast channel
- two or more simultaneous transmissions by nodes causes interference
- collision if node receives two or more signals at the same time
- how to coordinate the access of multiple sending/receiving nodes to a shared broadcast channel?
- nodes use multiple access protocol to regulate their transmission into the shared channel

Multiple Access Protocol

- Distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- Communication about channel sharing must use the channel itself!
  - no out-of-band channel for coordination
- Dozens of multiple access protocols have been implemented in a variety of link layer technologies

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 N nodes want to transmit, each can send at average rate R/N
3. fully decentralized (distributed):
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. Simple so that it is inexpensive to implement

Multiple Access Protocols: a taxonomy

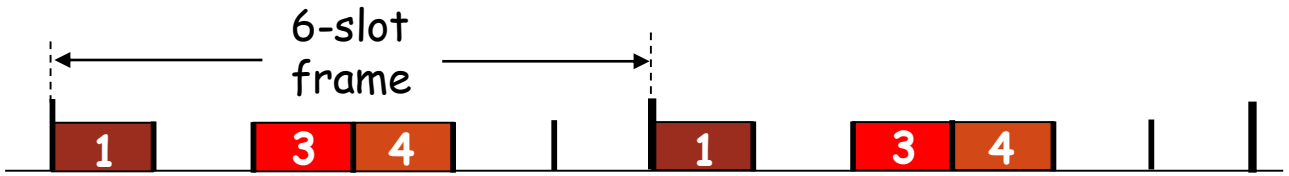
Three broad classes:

- Channel Partitioning
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate pieces to nodes for exclusive use (no collision)
  - Each node transmit at R/N
- Random Access
  - channel not divided, node transmit at full rate R
  - allow collisions
  - "recover" from collisions
- Taking turns
  - nodes take turns, but nodes with more to send can take longer turns

TDMA

TDMA: Time Division Multiple Access

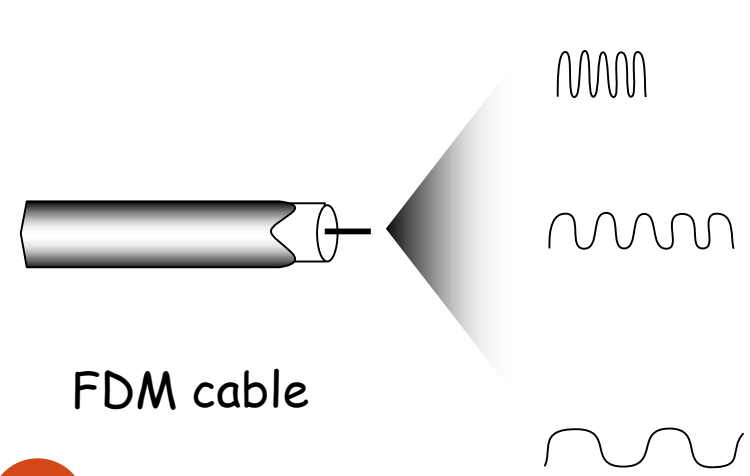
- access to channel in "rounds"
- each node gets fixed length slot (length = pkt trans time = L/R) in each round
- Each node is assigned R/N bandwidth
- unused slots go idle
- example: 6-nodes LAN, 1,3,4 have pkt, slots 2,5,6 idle

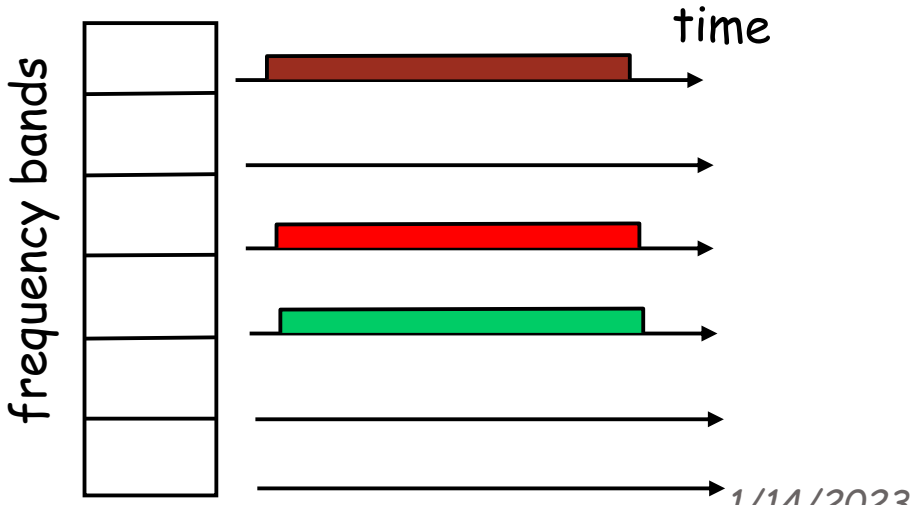


FDMA

FDMA: Frequency Division Multiple Access

- channel spectrum divided into frequency bands
- each node assigned fixed frequency band (R/N)
- unused transmission time in frequency bands go idle
- example: 6-nodes LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle





CDMA

CDMA: Code Division Multiple Access

- CDMA assigns a different code for each node
- Each node uses its unique code to encode that data bits it sends
- Different nodes can transmit simultaneously and yet have respective receivers correctly receive a sender's encoded data bits
- Used widely in wireless networks

Random Access Control Protocols

- When node has packet to send
  - transmit at full channel data rate R
  - no *a priori* coordination among nodes (send at random)
- Two or more transmitting nodes ➜ "collision",
- Random access protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., *via delayed retransmissions*)
  - the delays are random and independently chosen by nodes involved in collisions
  - It is possible that one node will pick a delay sufficiently less than other nodes and therefore, will be able to sneak its frame into the channel without collision

Random Access Protocol

- There are dozens of random access protocols
- Examples of random access MAC protocols:
  - slotted ALOHA
  - Pure (unslotted) ALOHA
  - Carrier Sense Multiple Access (CSMA), CSMA/CD, CSMA/CA

Slotted ALOHA

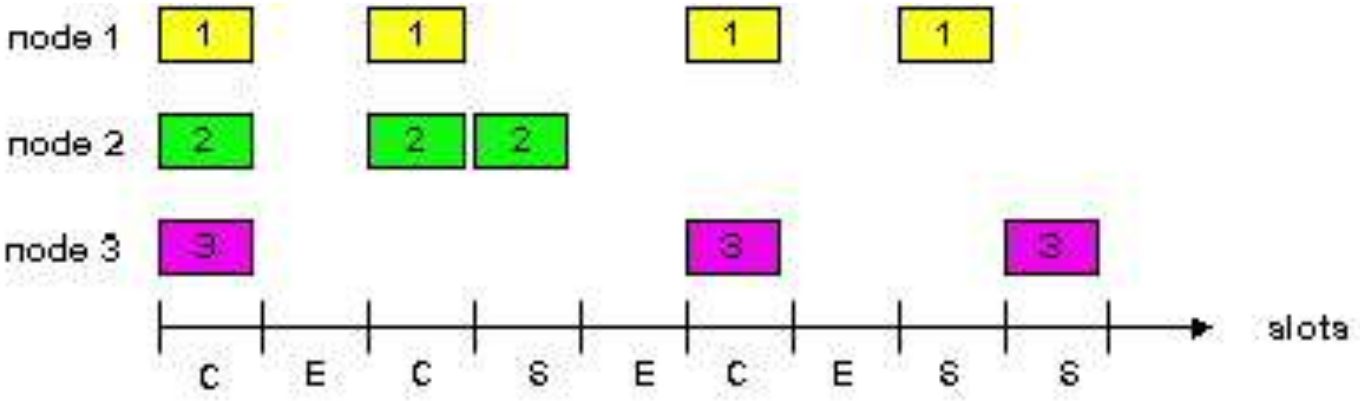
Assumptions:

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

Operation:

- when node obtains fresh frame, it 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)

- N nodes with many frames to send, each transmits in slot with probability p
- *Prob that a given node transmit* $= p$
- *Prob. that a given node does not transmit* $= 1-p$
- *Prob that other nodes do not transmit* $= (1-p)^{N-1}$
- *Prob that a given node has success in a slot* $= p(1-p)^{N-1}$
- *Prob that any node has a success* $= Np(1-p)^{N-1}$
- *Effiviency* $= \lim Np(1-p)^{N-1}$

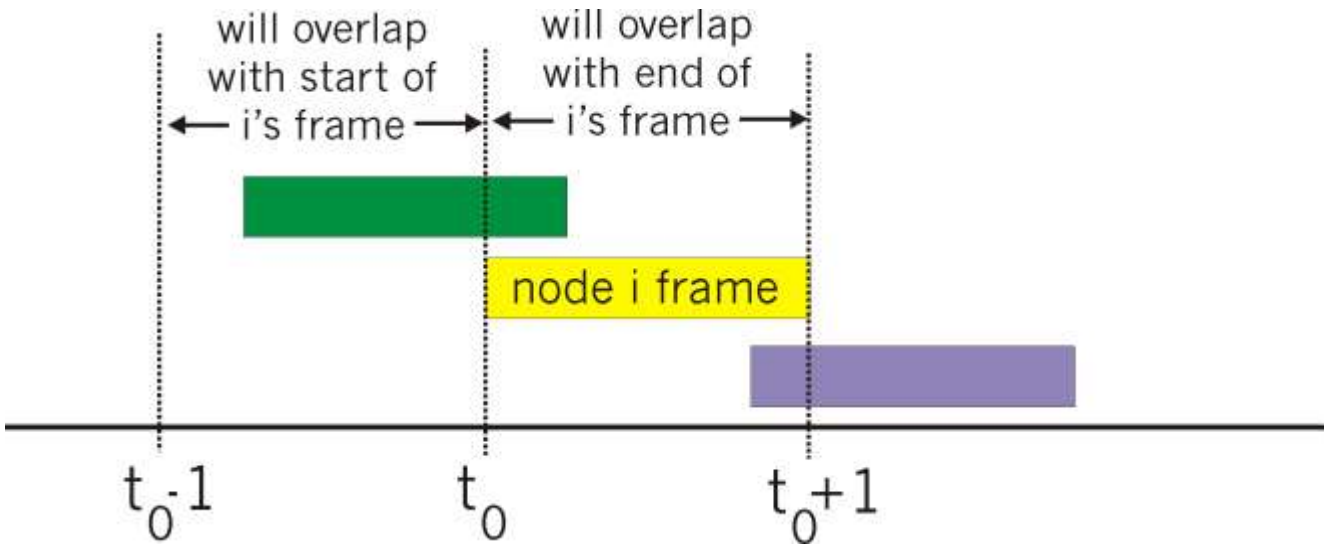
Slotted ALOHA efficiency

- 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, this gives: Max efficiency = 1/e = 0.37

- At best: channel used for useful transmissions 37% of time!
- Effective transmission rate =0.37 R
- 37% of slots go empty
- 26% of slots have collision

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

Probability that node transmits=p

prob. that other nodes do not transmit in $[t_0-1, t_0] = (1-p)^{N-1}$

prob. that other nodes do not transmit in $[t_0, t_0+1] = (1-p)^{N-1}$

prob. that a node succeeds in transmission $= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} = p \cdot (1-p)^{2(N-1)}$

prob. that any node succeeds in transmission $= Np \cdot (1-p)^{2(N-1)}$

… choosing optimum p and then letting N -> infty …

$= 1/(2e) = 0.18$

even worse than slotted Aloha!

CSMA (Carrier Sense Multiple Access)

CSMA/CA

- listen to channel ***before transmit***:
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission
- human analogy: don't interrupt others!

CSMA/CD (Collision Detection)

CSMA/CD:

- listen to channel ***while transmitting***
  - If another node transmits at the same time, defer transmit
- human analogy: if someone else begins talking at he same time, stop talking
- collision detection: measure signal strengths
  - collisions *detected* within short time
  - colliding transmissions aborted, reducing channel wastage

"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" MAC protocols

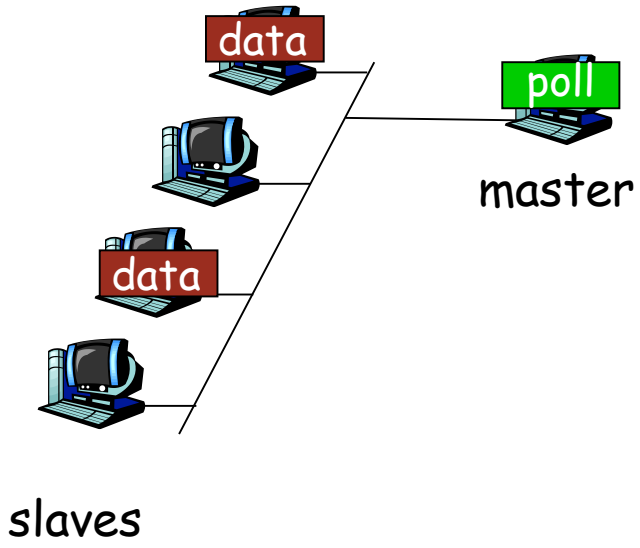
look for best of both worlds!

"Taking Turns" MAC protocols

- When only one node is active, transmission rate = R
- When N nodes are active, transmission rate = R/N
- Dozens of protocols
  - Polling protocol
  - Token-passing protocol

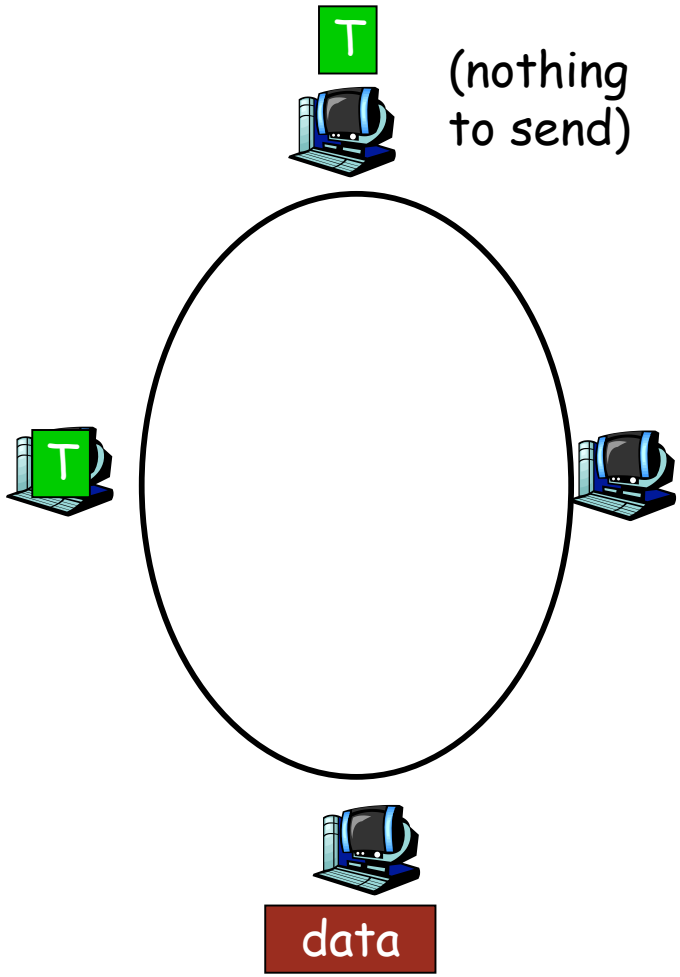
Polling protocol

- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- Concerns:
  - polling overhead (latency)
  - single point of failure (master)
- Examples: Bluetooth and 802.15



Token passing protocol

- Control token passed from one node to next sequentially.
- When a node receives the token, it holds onto it only when it has frames to send, otherwise, it immediately passes the token to the next node
- If a node does have frames to transmit when it receives the token, it sends up to the maximum number of frames and then forward the token to the next node
- Pros: decentralized and highly efficient
- Cons: token overhead (latency) or failure
- Examples: FDDI, 802.5



Summary of MAC protocols

- *channel partitioning,*
  - Time Division, Frequency Division, Code Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD, CSMA/CA
  - 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 (register) from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring

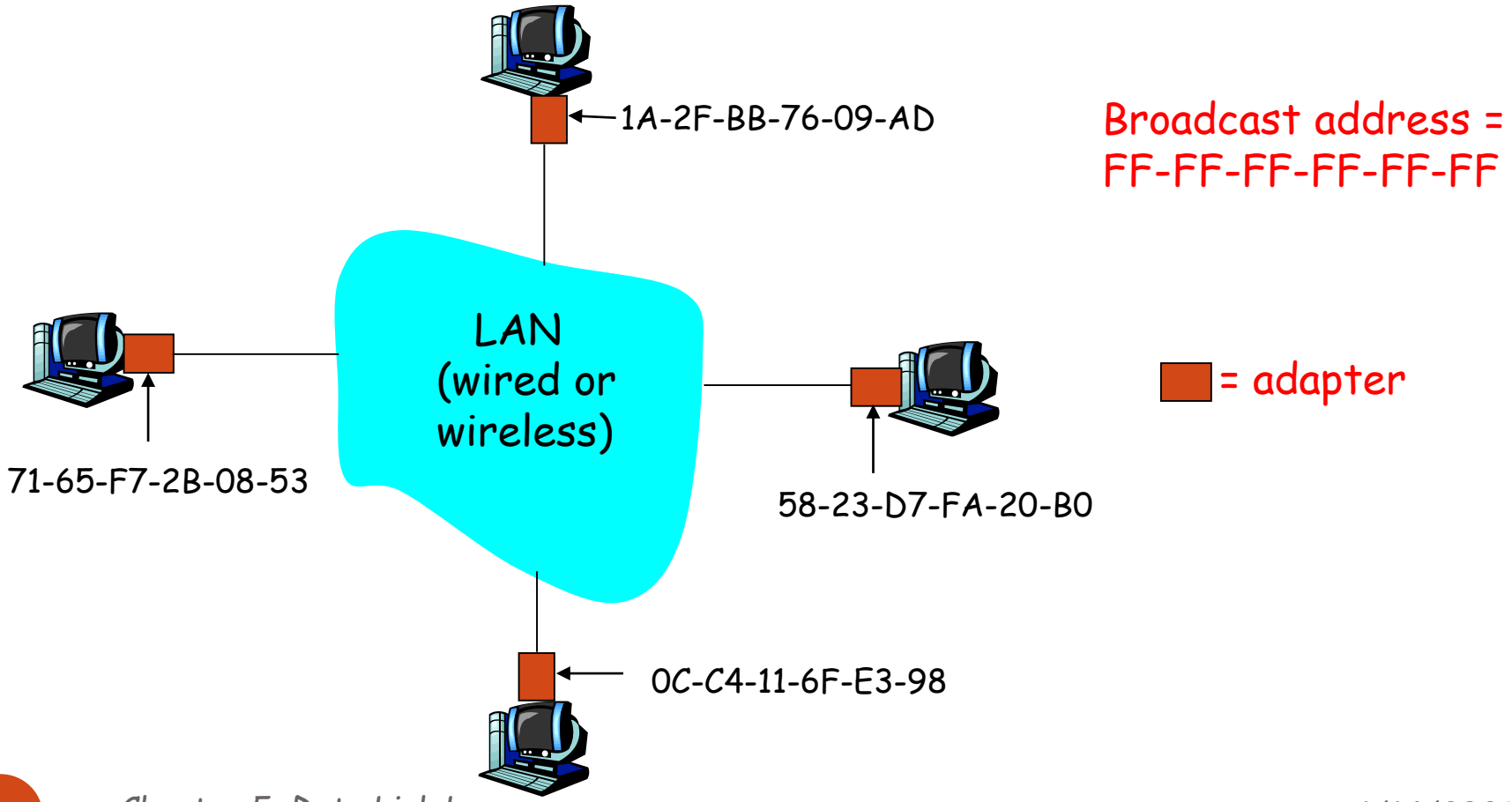
4. MAC Addresses

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

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

LAN Addresses

Each adapter on LAN has unique LAN address
IEEE set different MAC address for different adapter



LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion (24 bit) 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

IP address vs. MAC address

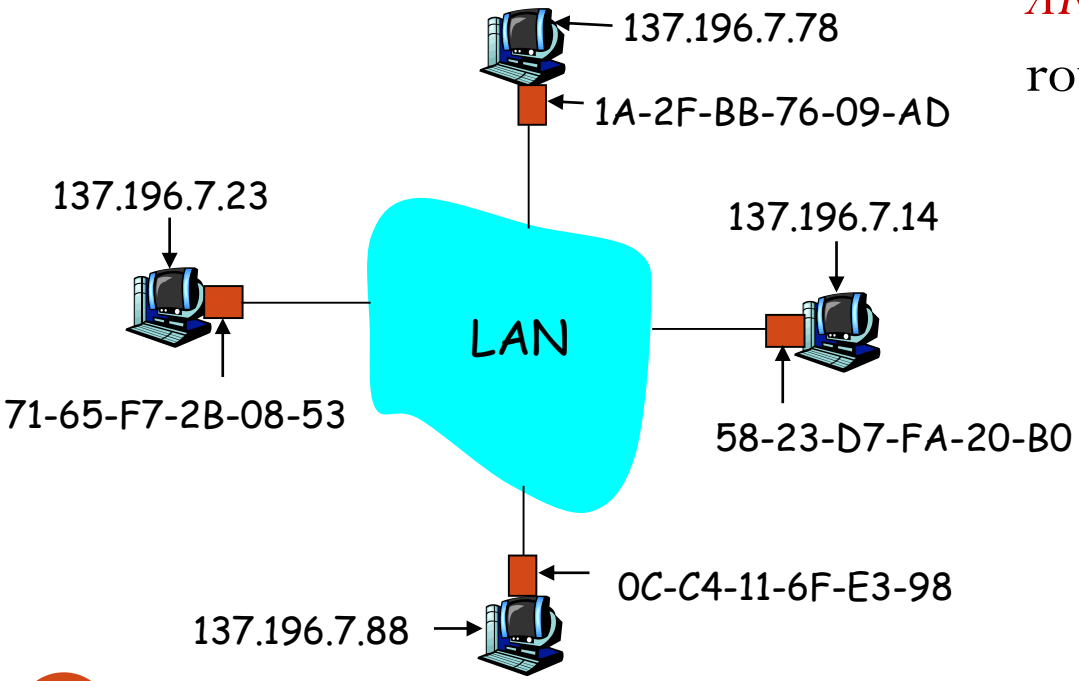
IP address	MAC address
Network layer address	Link layer address
32-bit (4byte)	48-bit (6byte)
Address each node (host or router) interface	Address each node network adapter (NIC)
Hierarchical (subnet part, host part) CIDR	flat
Dynamic (change if node moved to another subnet)	Fixed and permanent (does not change if node moved to another LAN)-portable
Dotted-decimal notation	Hexadecimal notation
Administered by ICANN and assigned to ISP	Administered by IEEE and assigned to manufacture
Determined by DHCP	Determined by ARP

ARP: Address Resolution Protocol

IP address	MAC address
137.196.7.78	1A-2F-BB-76-09-AD
137.196.7.88	0C-C4-11-6F-E3-98
137.196.7.23	71-65-F7-2B-08-53
137.196.7.14	58-23-D7-FA-20-B0

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

Using ARP table



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)

DNS vs. ARP

DNS	ARP
Translate from hostname to IP address	Translate from IP address to MAC address
Translate any where in the Internet	Translate within the subnet only (same LAN)

- 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)

IP address	LAN address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figure 5.4-3: A possible ARP table in node 222.222.222.220.

ARP: Address Resolution Protocol

- Node 137.196.7.78 wants to send an IP datagram to Node 137.196.7.88
- Node 137.196.7.78 uses its ARP table to find the MAC address of the dest. Node 137.196.7.88
- The MAC address of the dest. node 137.196.7.88

  is 0C-C4-11-6F-E3-98
- The adapter of node 137.196.7.78 construct a frame containing the MAC address 0C-C4-11-6F-E3-98 and send it to LAN

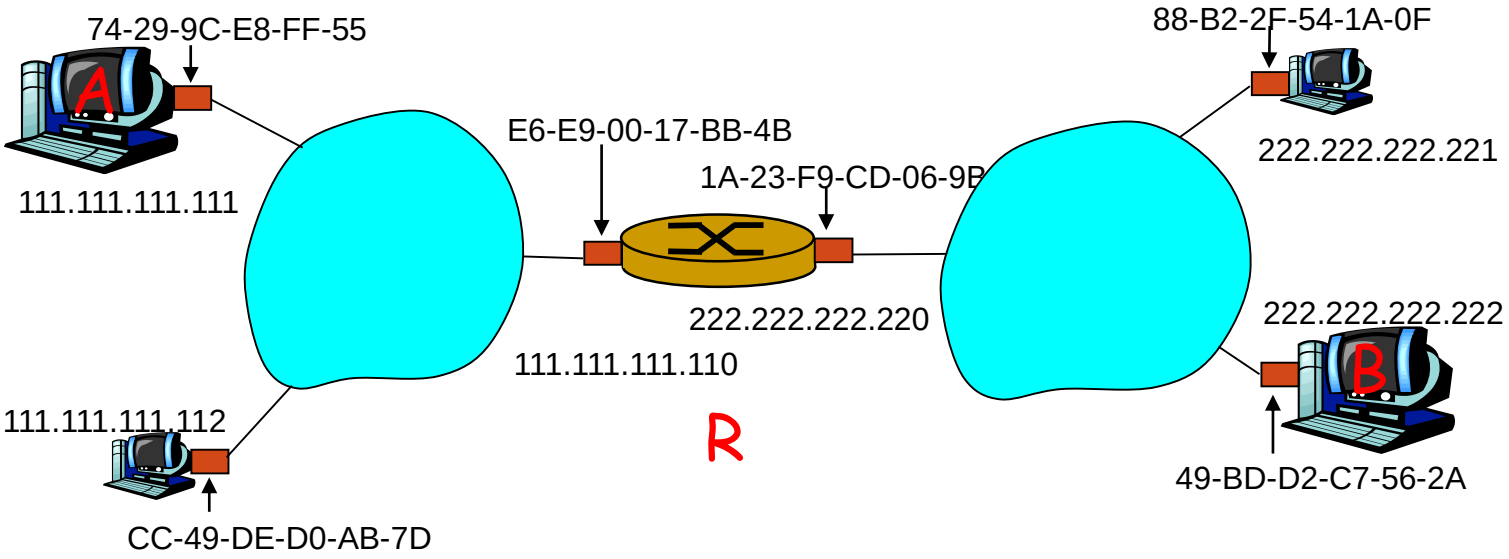
ARP protocol: Same LAN (network)

- 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

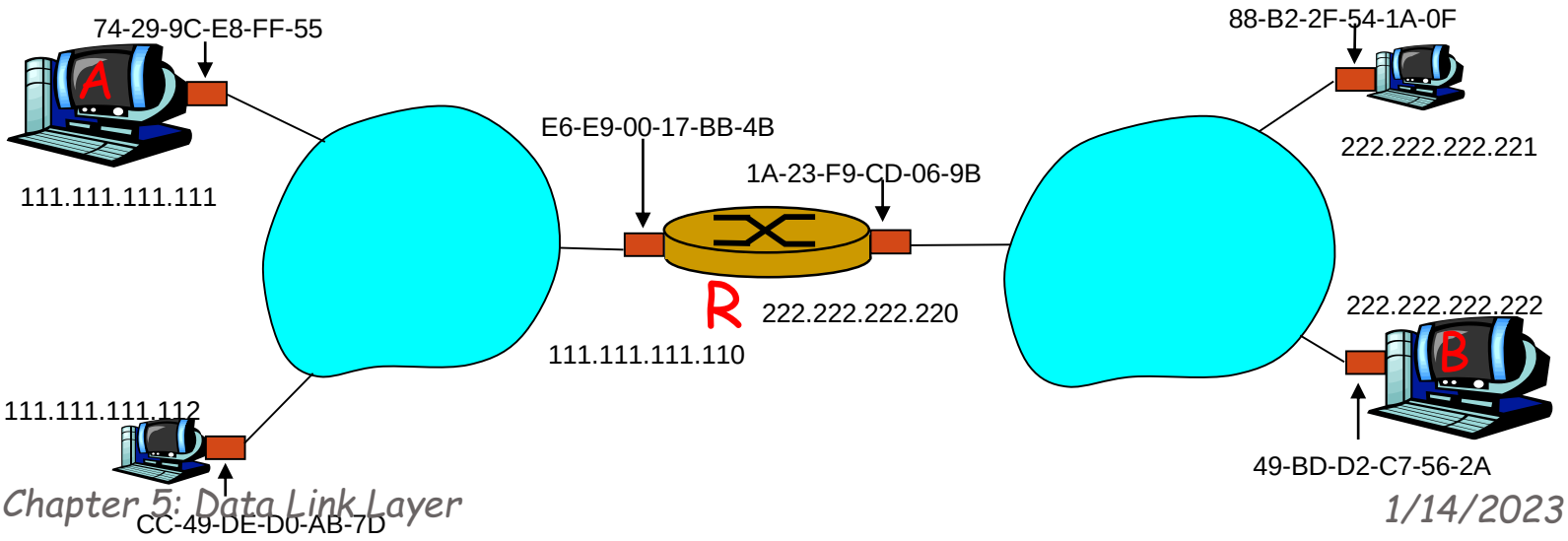
walkthrough: send datagram from A to B via R

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 its forwarding table to place the datagram on the correct interface
- R uses ARP to get B's MAC address
- R creates frame containing A-to-B IP datagram sends to B



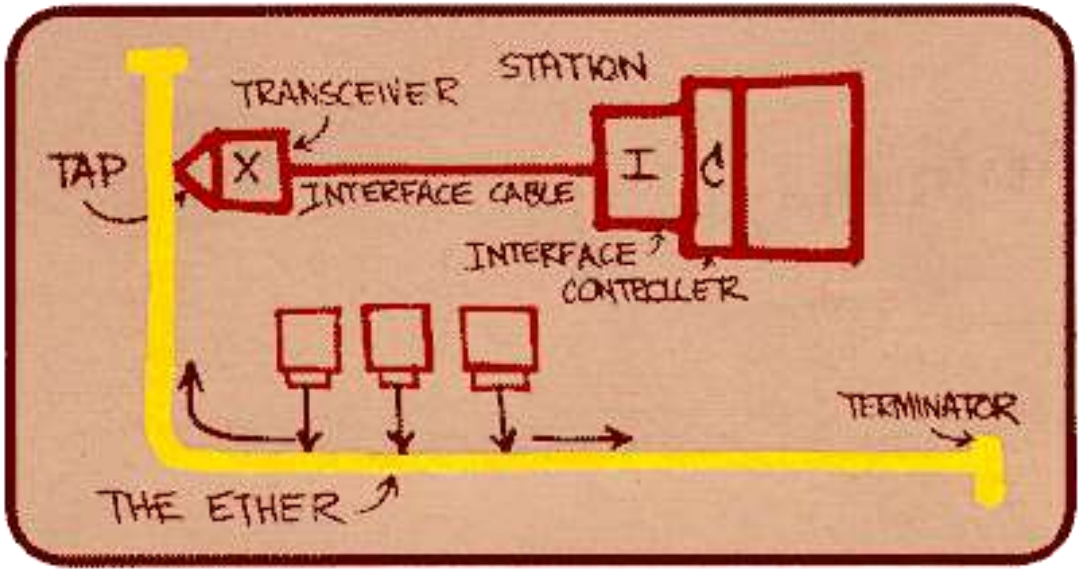
5. Ethernet

- Different wired LAN technology
  - Token ring LAN
  - Fiber Distributed Data Interface (FDDI) LAN
  - ATM LAN
  - **Ethernet LAN**
- Ethernet is the most widely used technology for LANs
- Ethernet for LANs is like Internet for global networking

Ethernet

"dominant" wired LAN technology:

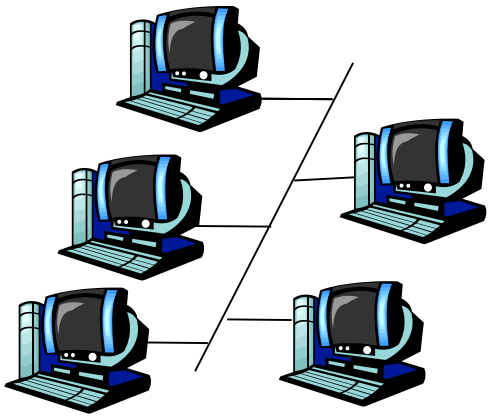
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



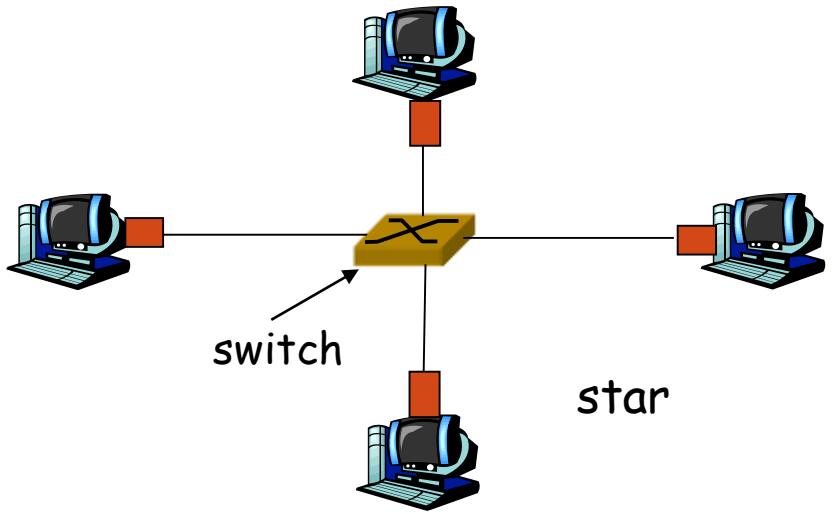
Metcalfe's Ethernet sketch

Star topology

- Bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - Hub at the center (collision occurs and node must retransmit)
  - active *switch* in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: coaxial cable

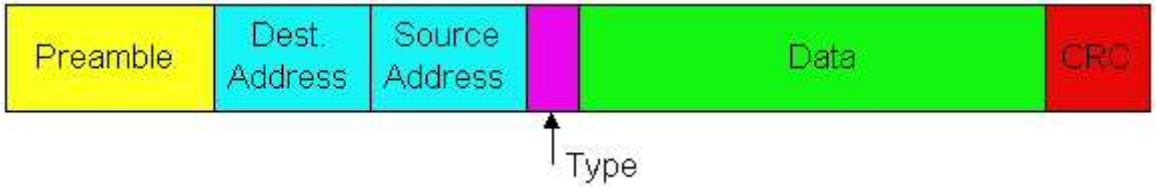


Hub and Switch

- A network hub contains multiple ports. When a a bit (or a frame) arrives at one port, it is copied ***unmodified*** to ***all*** ports of the hub for transmission
- A network switch is a device that forwards and filters datagrams between ports based on the MAC addresses in the packets
- This is distinct from a hub in that it only forwards the frames to ***the ports involved in the communication*** rather than all ports connected

Ethernet Frame Structure

- 802.1 protocol
- Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame and sends the frame down to the physical layer



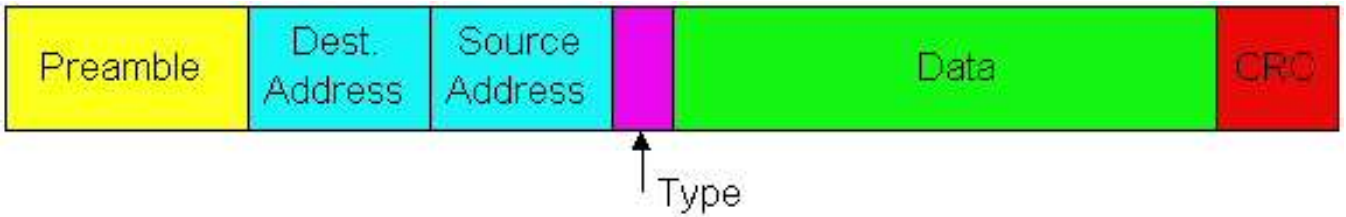
- At the receiver, the adaptor extracts IP datagram from the frame and forwards them up to the network layer

Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- Used to synchronize receiver's clock rate to sender's clock rate

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)
- Data: carries IP datagram (46-1500) bytes, MTU=1500
- CRC: checked at receiver, if error is detected, frame is dropped



Ethernet Protocol

- Connectionless
  - No handshaking between sending and receiving NICs
  - Similar to IP in Network layer and UDP in Transport layer
- Unreliable:
  - receiving NIC runs the frame through CRC check
  - receiving NIC doesn't send acks when the frame passes CRC check or nacks when it fails to sending NIC
  - when a frame fails CRC check, the receiving NIC ignores it
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP retransmission technique
  - otherwise, app will see gaps
- Ethernet's MAC protocol: CSMA/CD

Ethernet CSMA/CD mechanism

- An adapter may begin to transmit at any time, i.e., no slots are used.
- An adapter never transmits a frame when it senses that some other adapter is transmitting, i.e., it uses carrier-sensing.
- A transmitting adapter aborts its transmission as soon as it detects that another adapter is also transmitting, i.e., it uses collision detection.
- Before attempting a retransmission, an adapter waits a random time that is typically small compared to a frame time.

CSMA/CD efficiency

- efficiency of Ethernet is the long-run fraction of time during which frames are being transmitted on the channel without collisions when there is a large number of active nodes, with each node having a large number of frames to send.
- T_{prop} = max prop delay between 2 adapter in LAN
- t_{trans} = time to transmit max-size frame = 1.2 msec for 10 Mbps ethernet
- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity

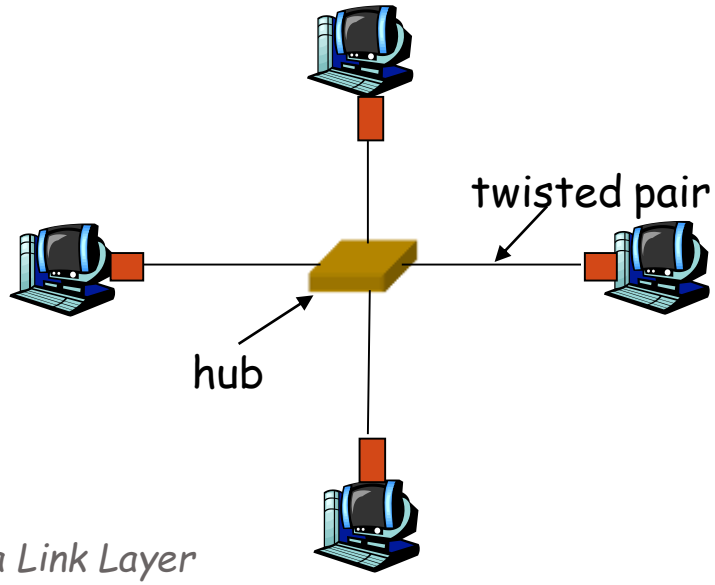
$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- better performance than ALOHA: and simple, cheap, decentralized!

6. Link layer Switches

hub 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

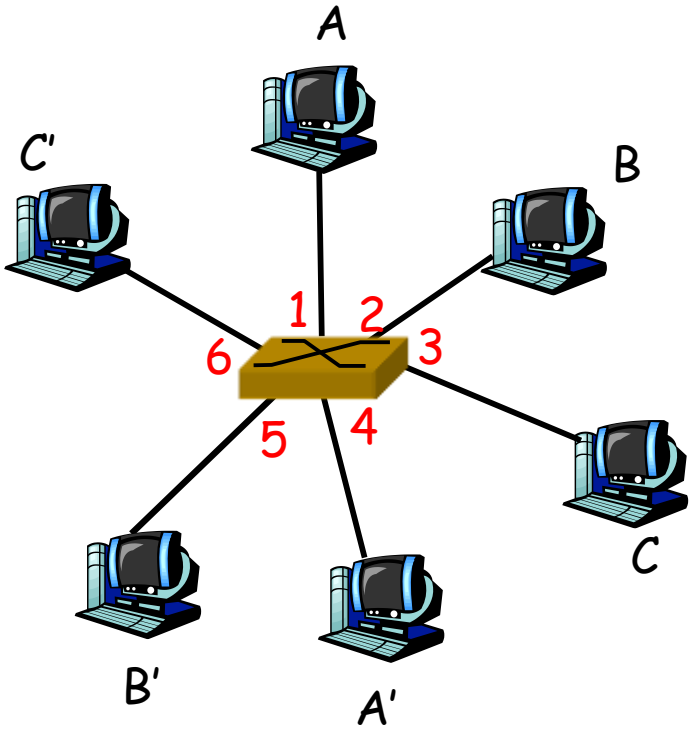


Switch

- link-layer device: smarter than hubs, take *active* role
  - Buffer frames
  - Filtering: determine whether a frame should be forwarded to some interface or just be dropped
  - forward 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
- *Allow multiple simultaneous transmission*
  - No collision
- *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
- *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)

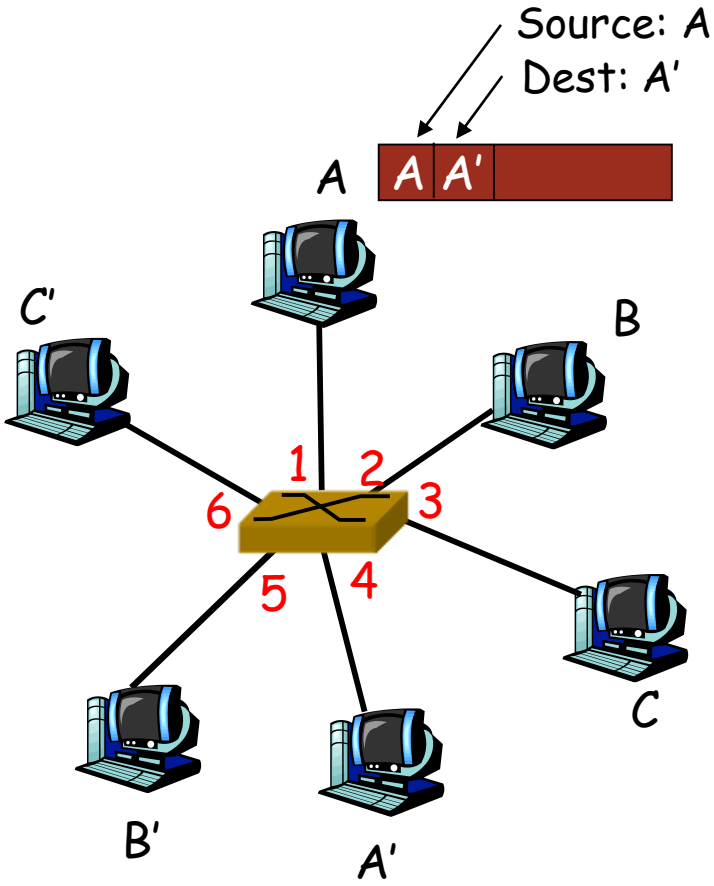
Switch Table

- *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?

Address	Interface	Time
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

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: self-learning

- The switch table is initially empty.
- For each frame received, the switch stores in its table: (1) the MAC address in the frame's *source address field*, (2) the interface from which the frame arrived, (3) the current time.
- If every node in the LAN eventually sends a frame, then every node will eventually get recorded in the table.
- When a frame arrives on one of the interfaces and the frame's destination address is in the table, then the switch forwards the frame to the appropriate interface.
- The switch deletes an address in the table if no frames are received with that address as the source address after a period of time (the *aging time). In this manner, if a PC is replaced by another PC (with a different adapter),* the MAC address of the original PC will eventually be wiped out of the switch table.

Switch self-learning

Address	Interface	Time
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

Host 01-12-23-34-45-56 connected to link 2 sent a frame

Address	Interface	Time
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
.....	.....	.....

Time 10:32 , Host 62-FE-F7-11-89-A3 not sending

MAC	Interface	Time
01-12-23-34-45-56	2	9:39
7c-BA-B2-B4-91-10	3	9:36

Switch: frame filtering/forwarding

When frame received:

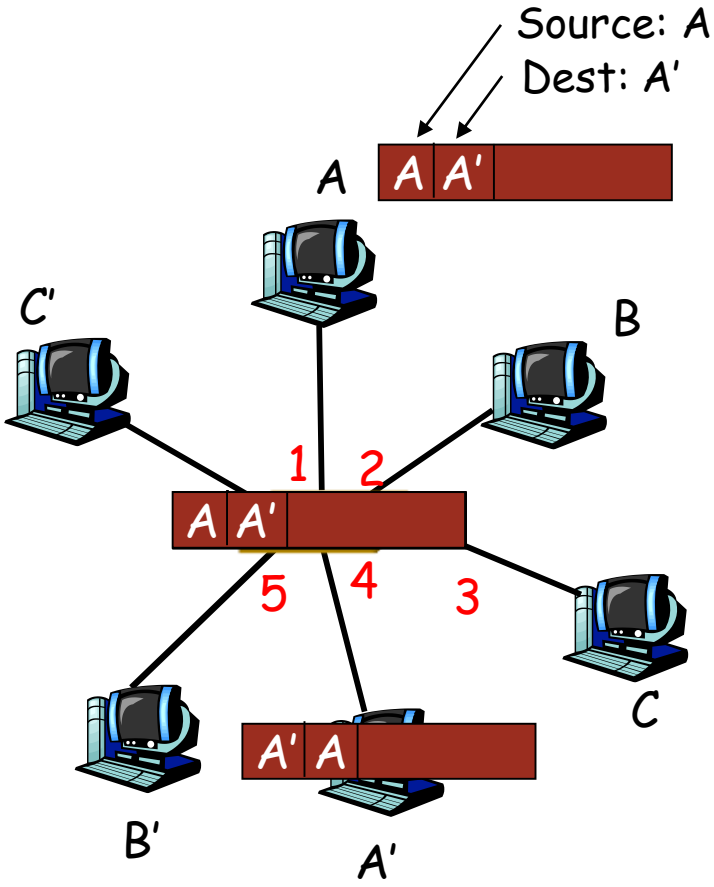
1. Record link associated with sending host, the MAC source address and TTL
2. Index switch table using MAC dest address of the frame
3. **if** entry found for destination address
   then {
      if dest on segment from which frame arrived
      then drop the frame
      else forward the frame on interface 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

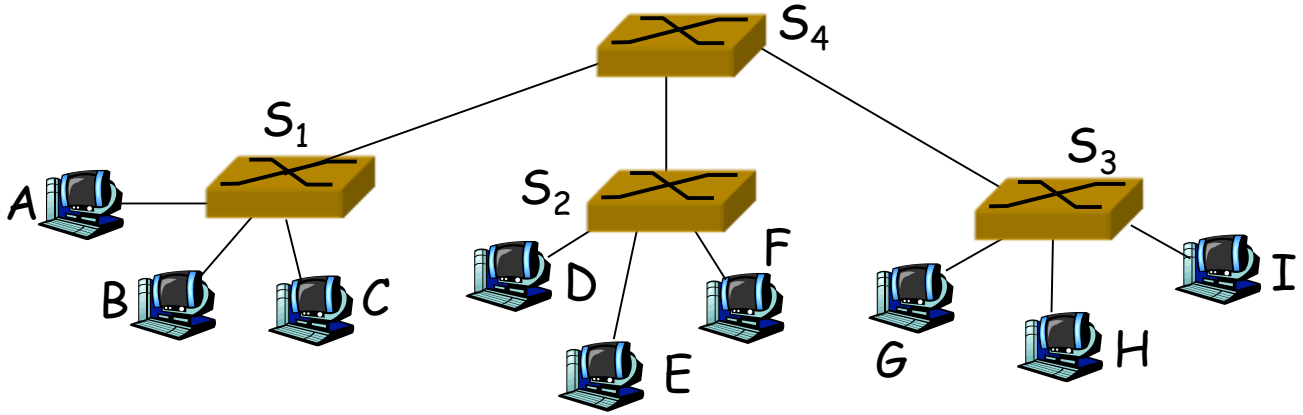


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

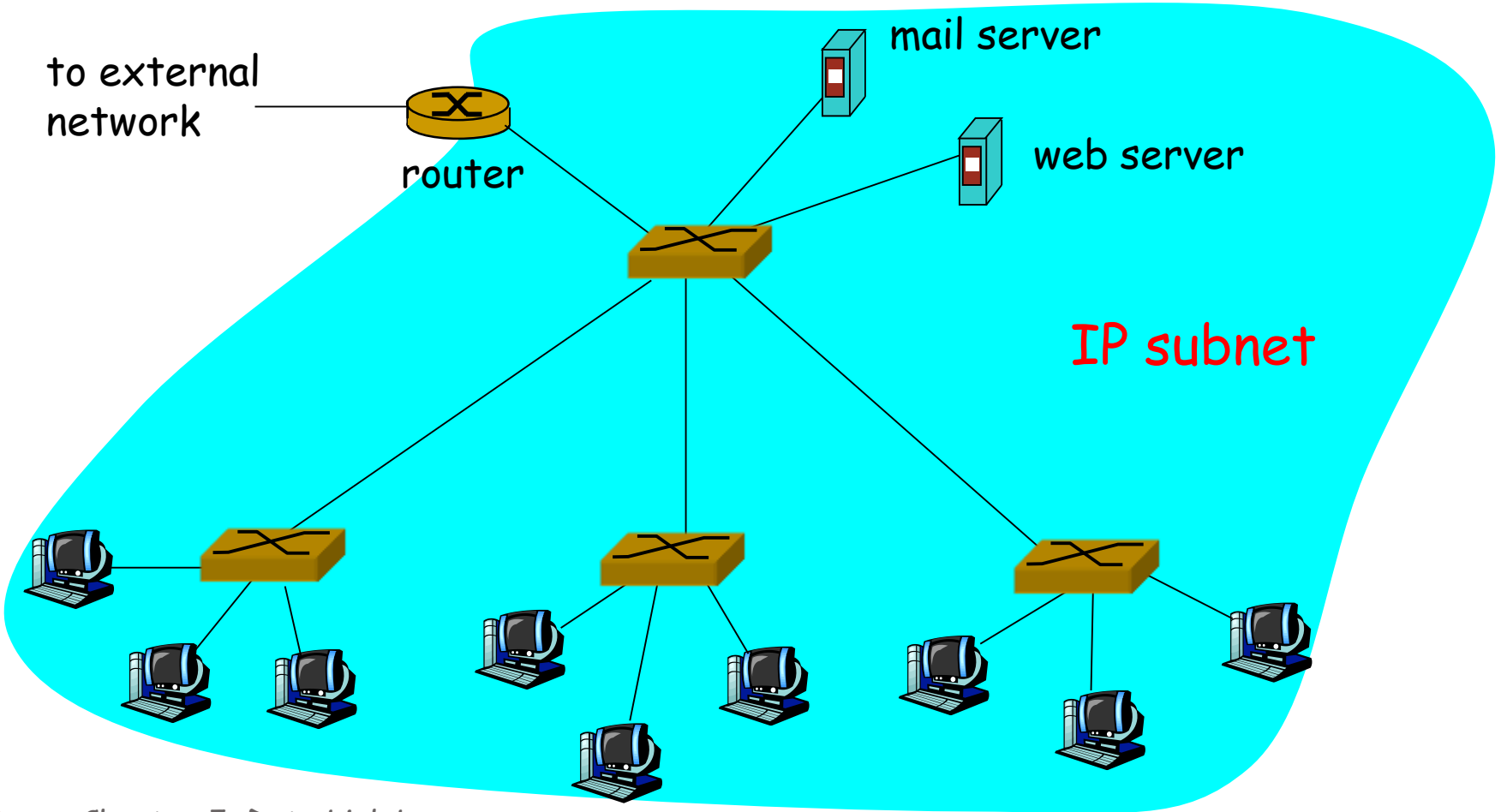
Interconnecting switches

- switches can be connected together



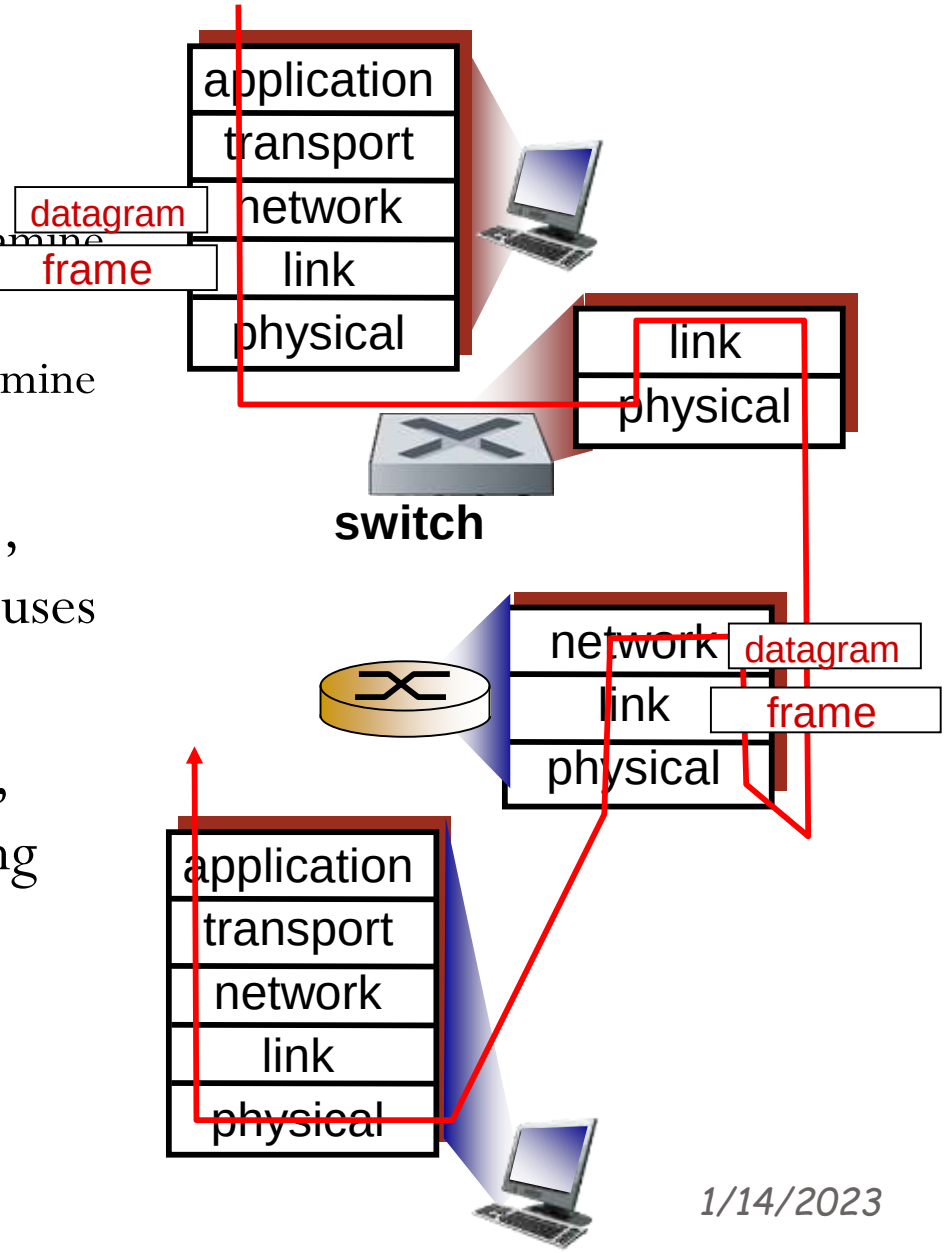
- Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?
- A: self learning! (works exactly the same as in single-switch case!)

Institutional network



Switches vs. Routers

- both store-and-forward devices
  - routers: network layer devices (examine network layer headers-datagarms)
  - switches are link layer devices (examine link layer header-frames)
- routers maintain forward tables, implement routing algorithms, uses IP address
- switches maintain switch tables, implement filtering, self learning algorithms, uses Mac address

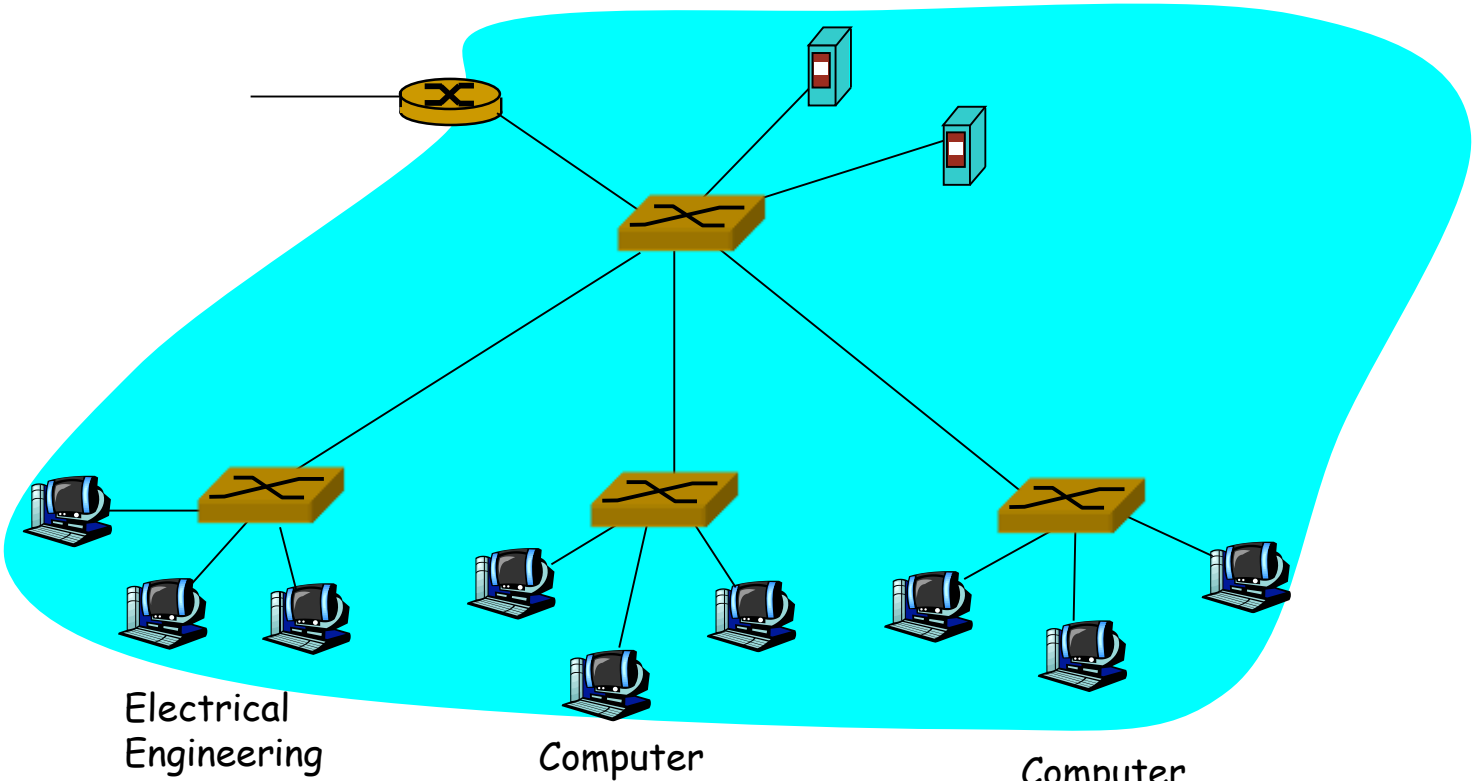


Hubs, Switches and Routers

	Hub	Router	Switch
Traffic isolation	No	Yes	Yes
Plug and play	Yes	No	Yes
Optimal routing	No	**Yes**	No

Virtual LANs

What's wrong with this picture?
Inefficient use of switches: each lowest level switch has only few ports in use. large number of switches is needed to isolate groups from each other

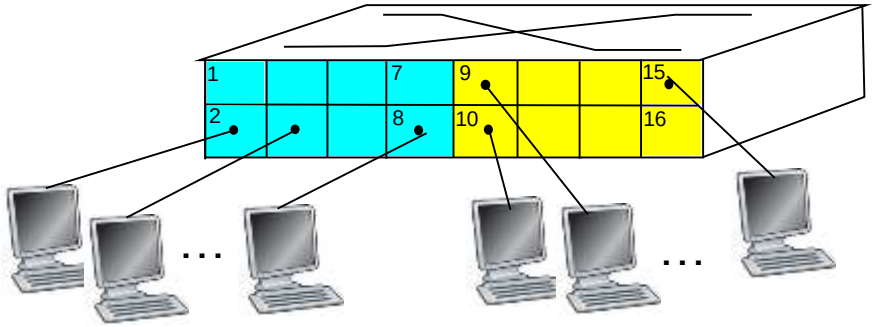


VLANs

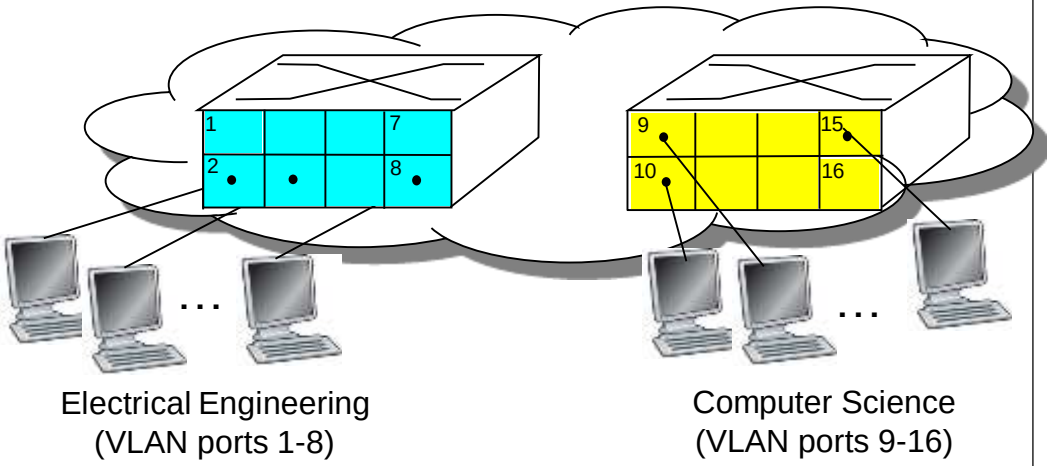
Virtual Local Area Network

Switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

Port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch .....



... operates as *multiple* virtual switches



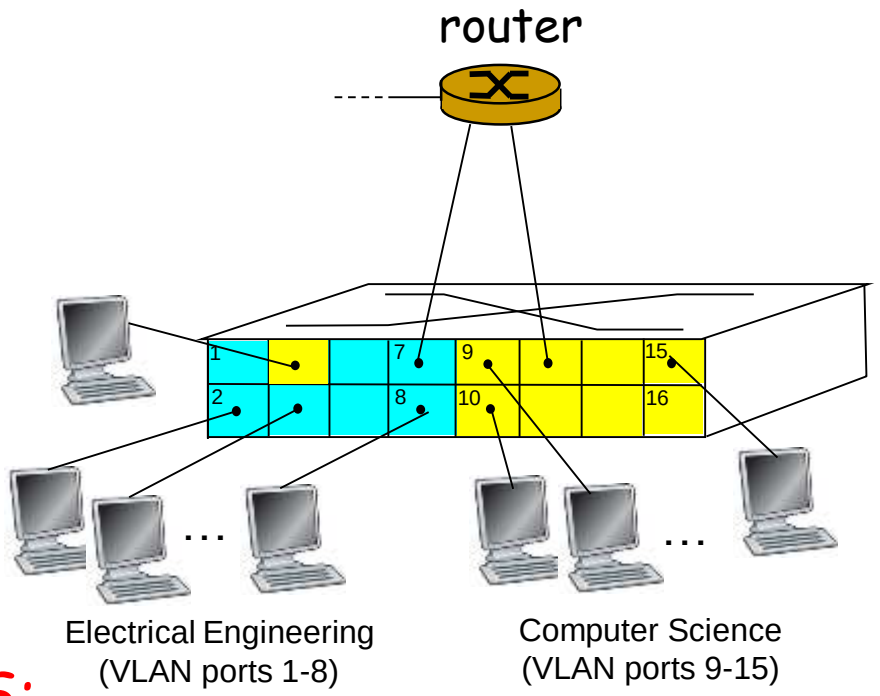
Port-based VLAN

- *traffic isolation:*

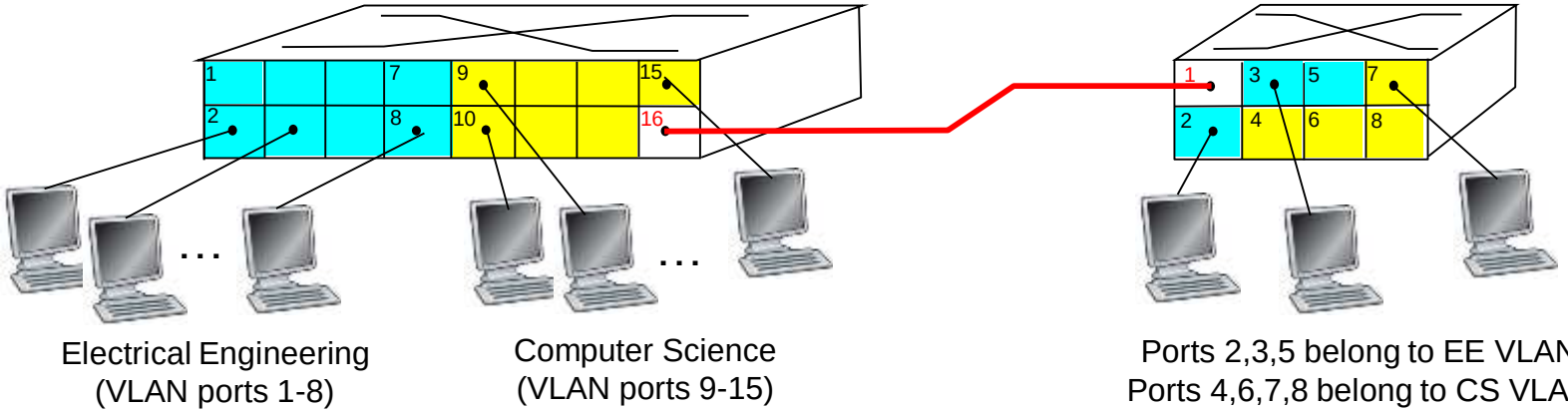
frames to/from ports 1-8 can only reach ports 1-8, can also define VLAN based on MAC addresses of endpoints, rather than switch port

- *dynamic membership:* ports can be dynamically assigned among VLANs

- *forwarding between VLANS:* done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers

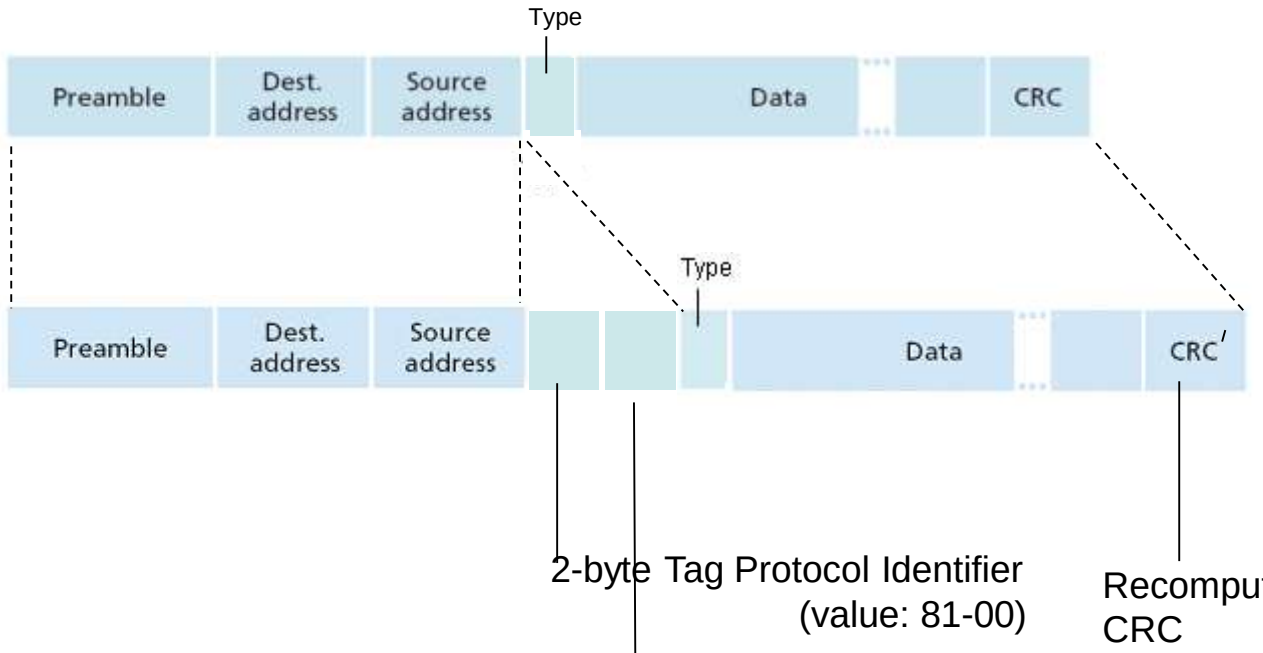


VLANS spanning multiple switches



- *trunk port:* carries frames between VLANS defined over multiple physical switches
  - frames forwarded within VLAN between switches must carry VLAN ID info
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format



7. Point to Point Data Link Control

- one sender, one receiver, one link: easier than broadcast link:
  - no Media Access Control (MAC)
  - no need for explicit MAC addressing
  - Point-to-point links: dialup link (56k), ISDN line, X2.5, …
- popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level Data Link Control (Data link used to be considered "high layer" in protocol stack!

PPP Design Requirements [RFC 1557]

- packet framing: encapsulation of network-layer datagram in data link frame
- multiple network layer protocol: carry network layer data of any network layer protocol (not just IP) *at same time* and ability to demultiplex upwards
- bit transparency: no constraints on network layer packet format
- error detection (no correction)
- connection liveness: detect, signal link failure to network layer
- network layer address negotiation: endpoint can learn/configure each other's network address
- multiple types of links: serial, parallel, synchronous, asynchronous, low speed, high speed, electrical, or optical
- simplicity

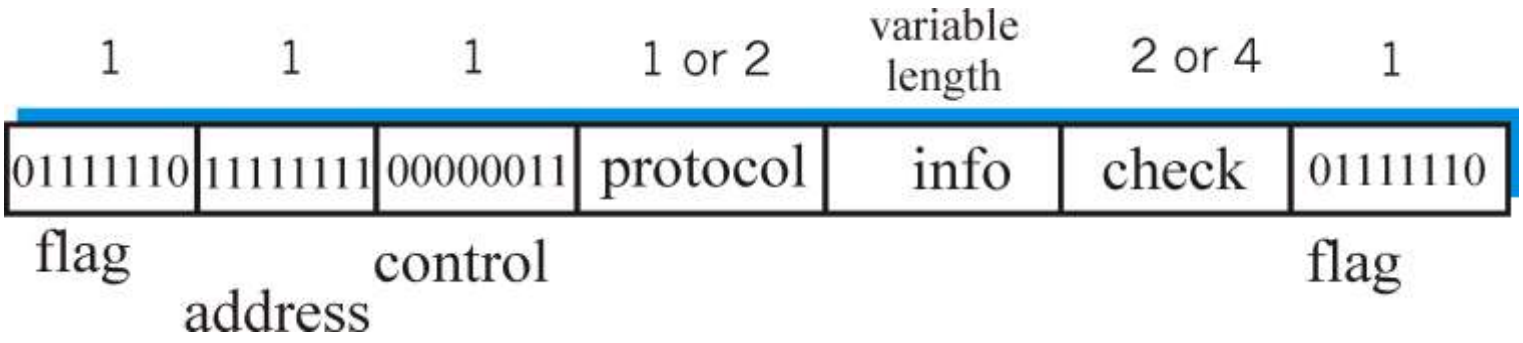
PPP non-requirements

- no error correction/recovery
- no flow control
- out of order delivery OK
- no need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering all relegated to higher layers!

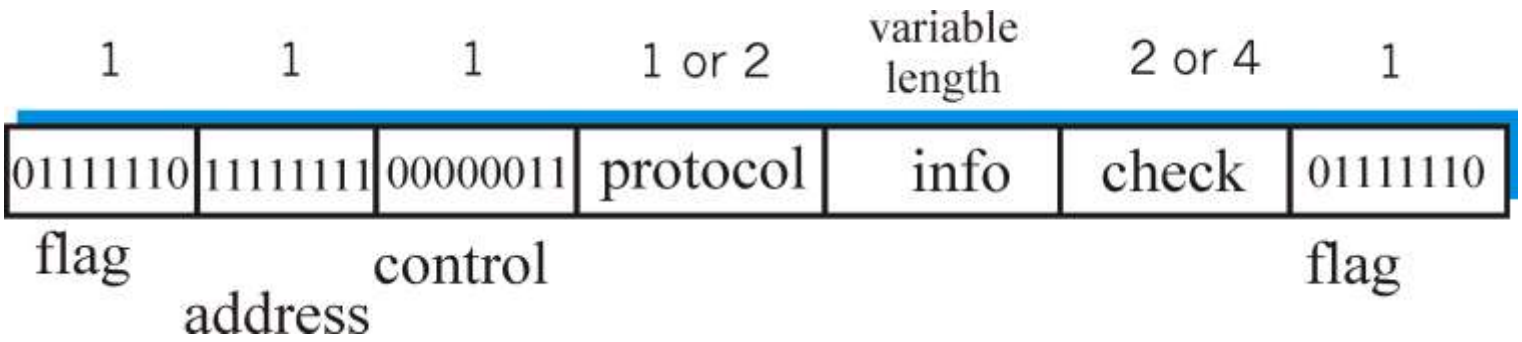
PPP Data Frame

- Flag: delimiter define the start and the end of (framing)
- Address: does nothing (fixed:only one option)
- Control: does nothing; in the future possible multiple control fields
- Protocol: upper layer protocol to which frame delivered (eg, IP=21, DECnet=27, AppleTalk=29)



PPP Data Frame

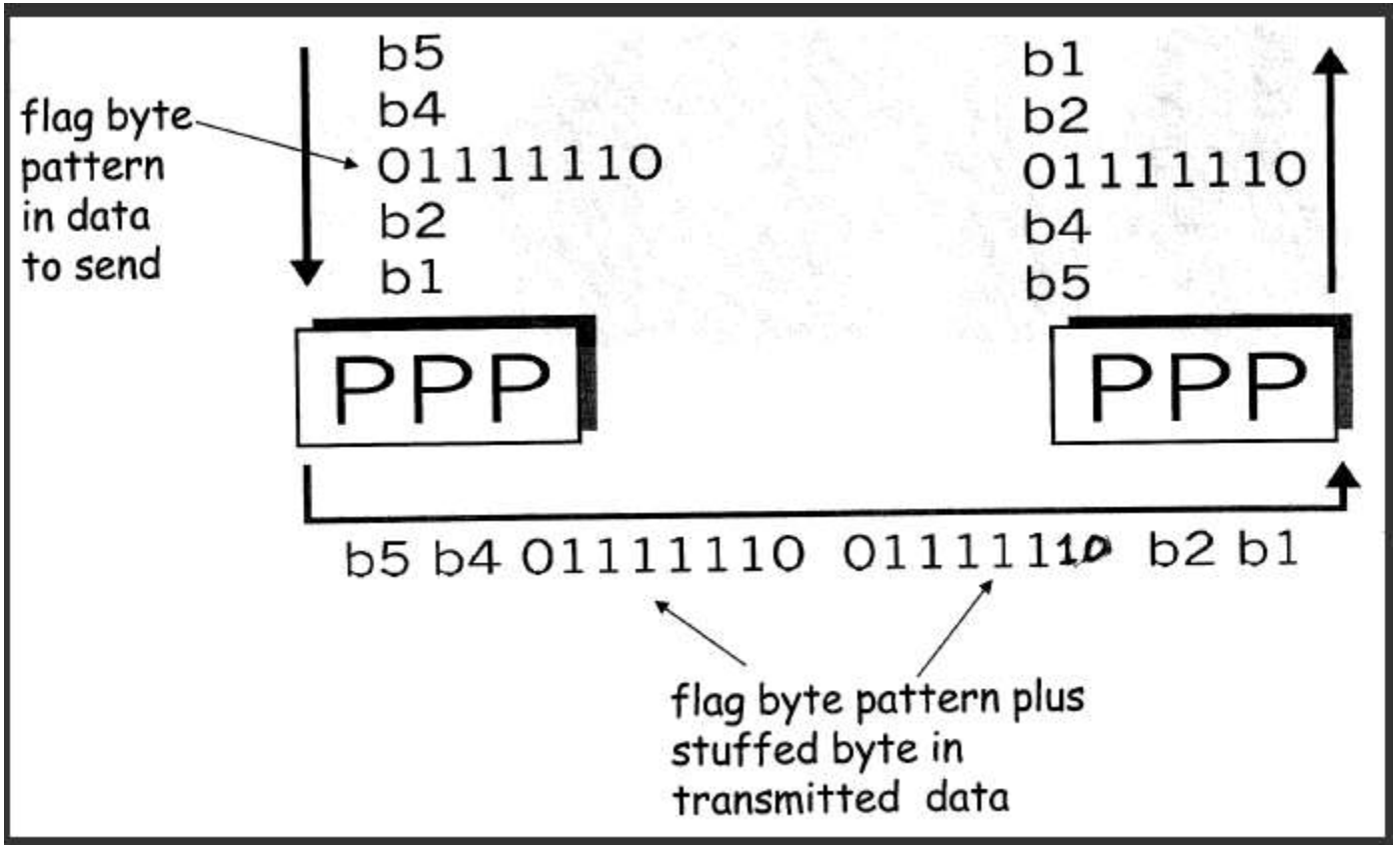
- info: upper layer data being carried
- check: cyclic redundancy check (CRC) for error detection



Byte Stuffing

- "data transparency" requirement: data field must be allowed to include flag pattern <01111110>
  - Q: is received <01111110> data or flag?
- Sender: adds ("stuffs") extra < 01111110> byte after each < 01111110> *data* byte
- Receiver:
  - two 01111110 bytes in a row: discard first byte, continue data reception
  - single 01111110: flag byte

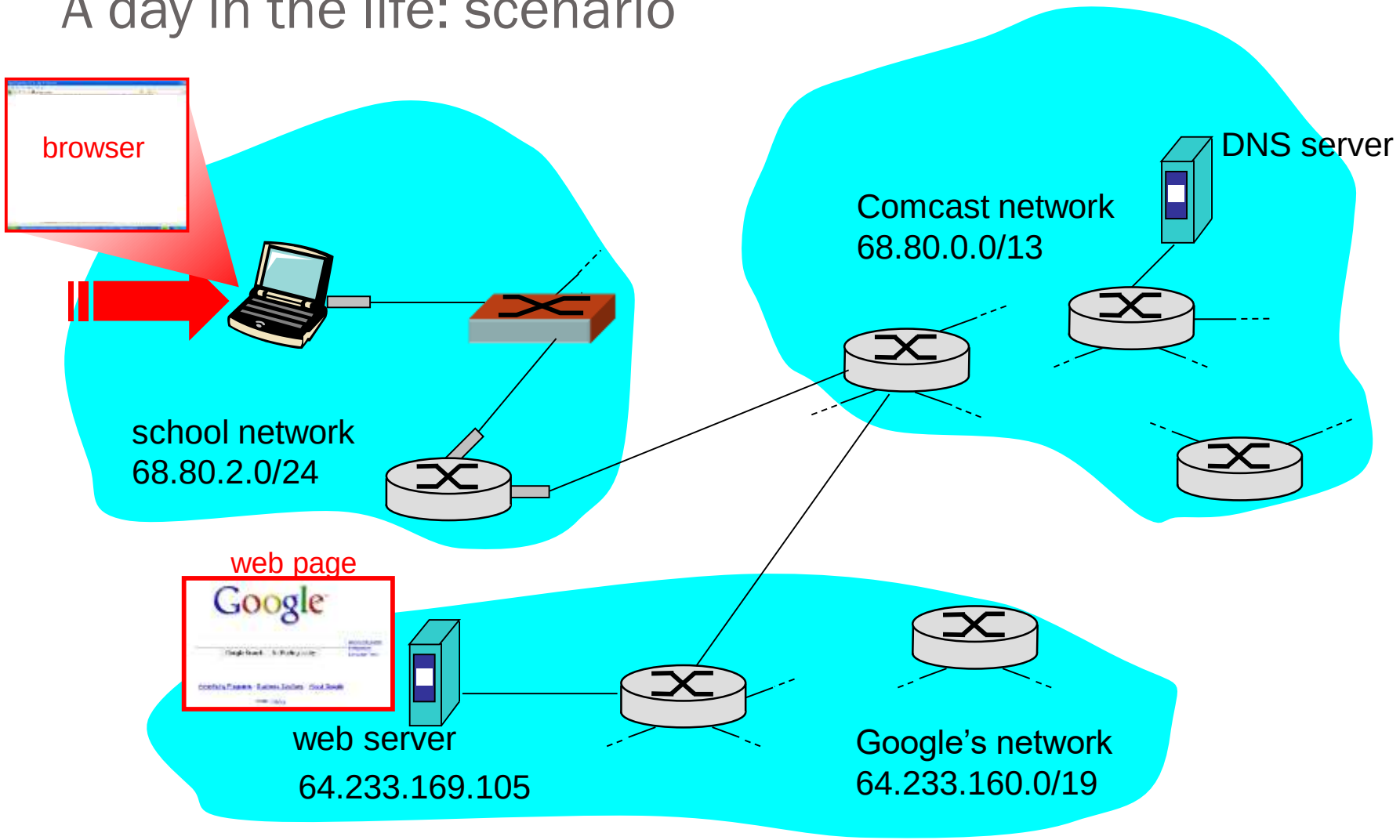
Byte Stuffing



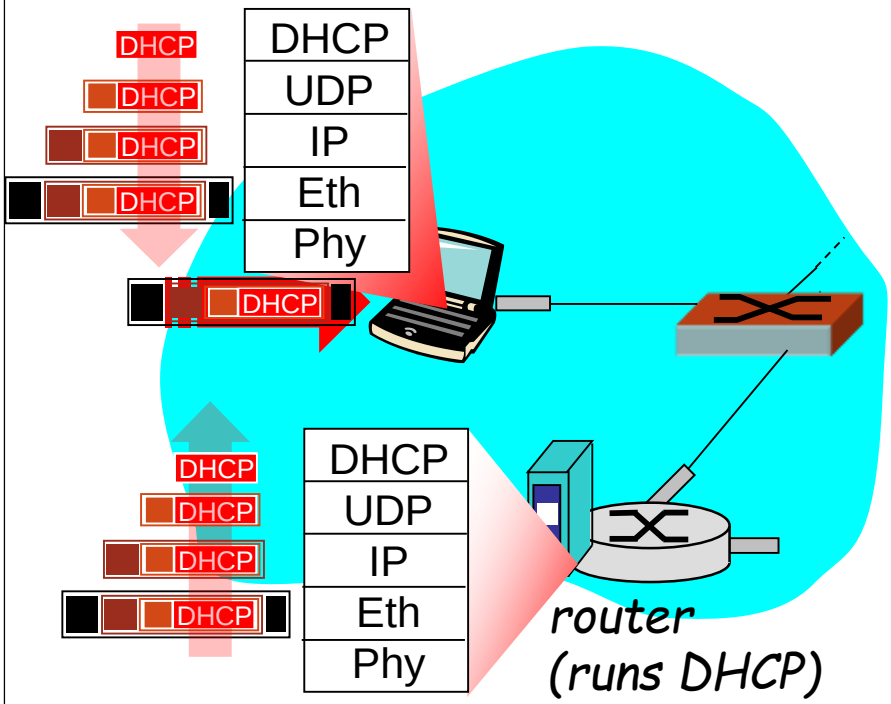
8. Synthesis: A day in the life of a web request

- 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

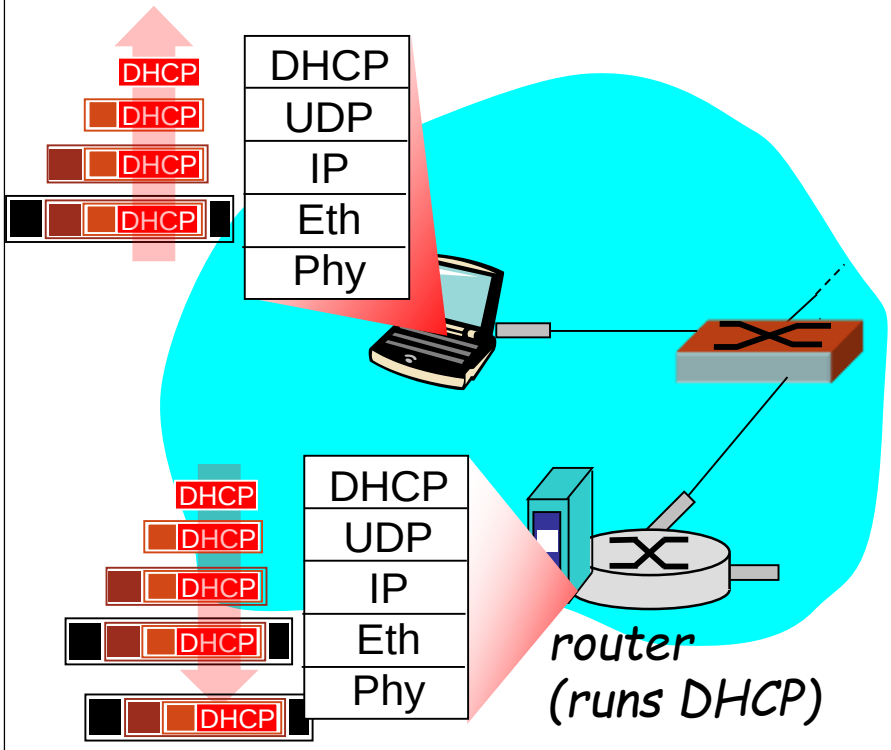


A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, IP addr of first-hop router, IP addr of DNS server: use ***DHCP***
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) 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



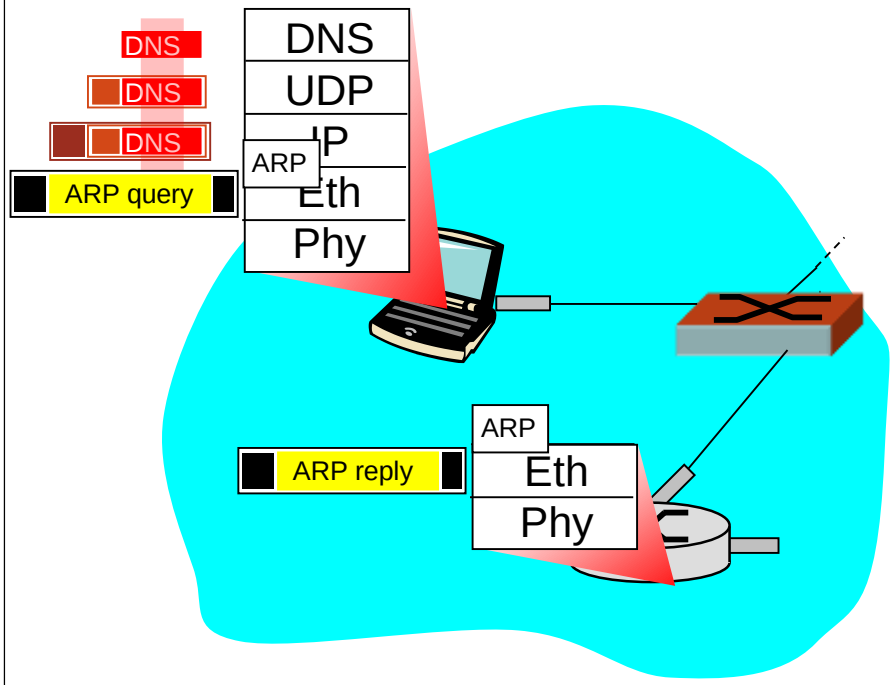
- DHCP server formulates ***DHCP ACK*** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- Encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client

- DHCP client receives DHCP ACK reply

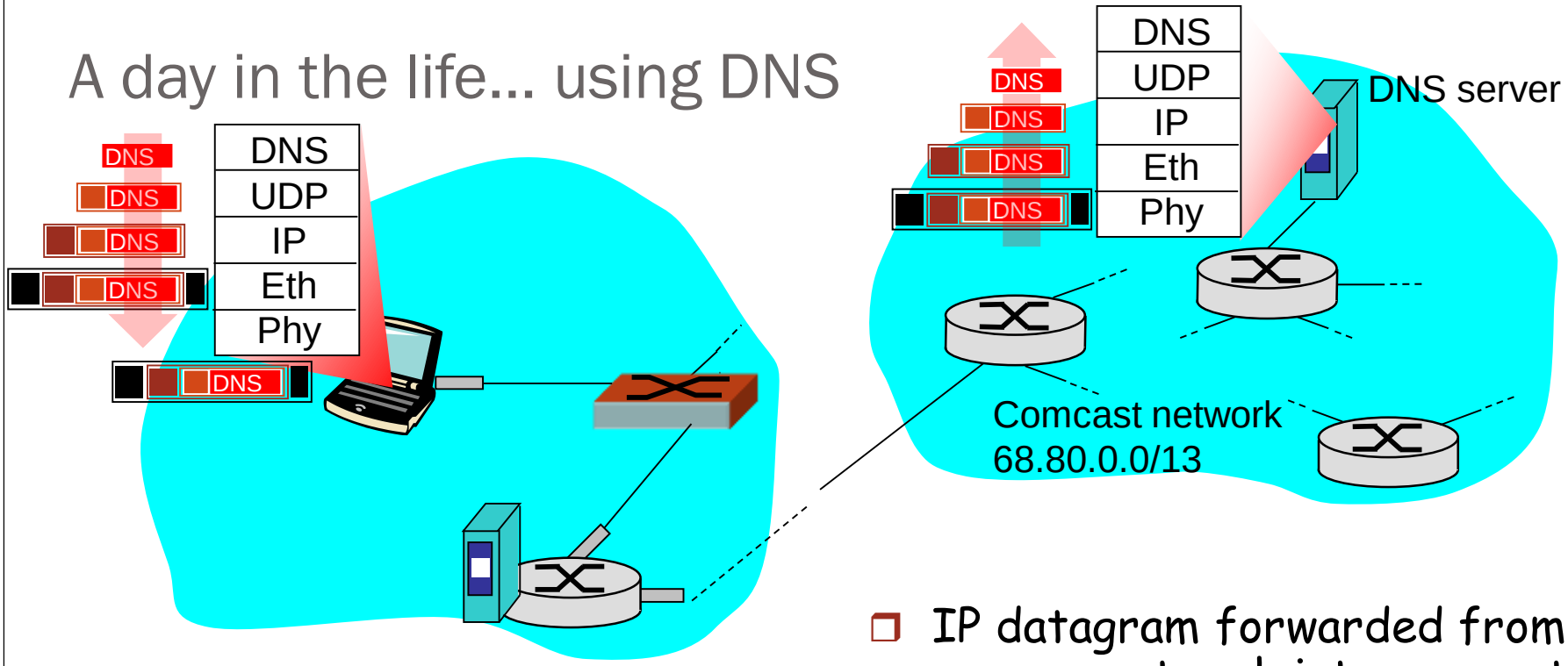
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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



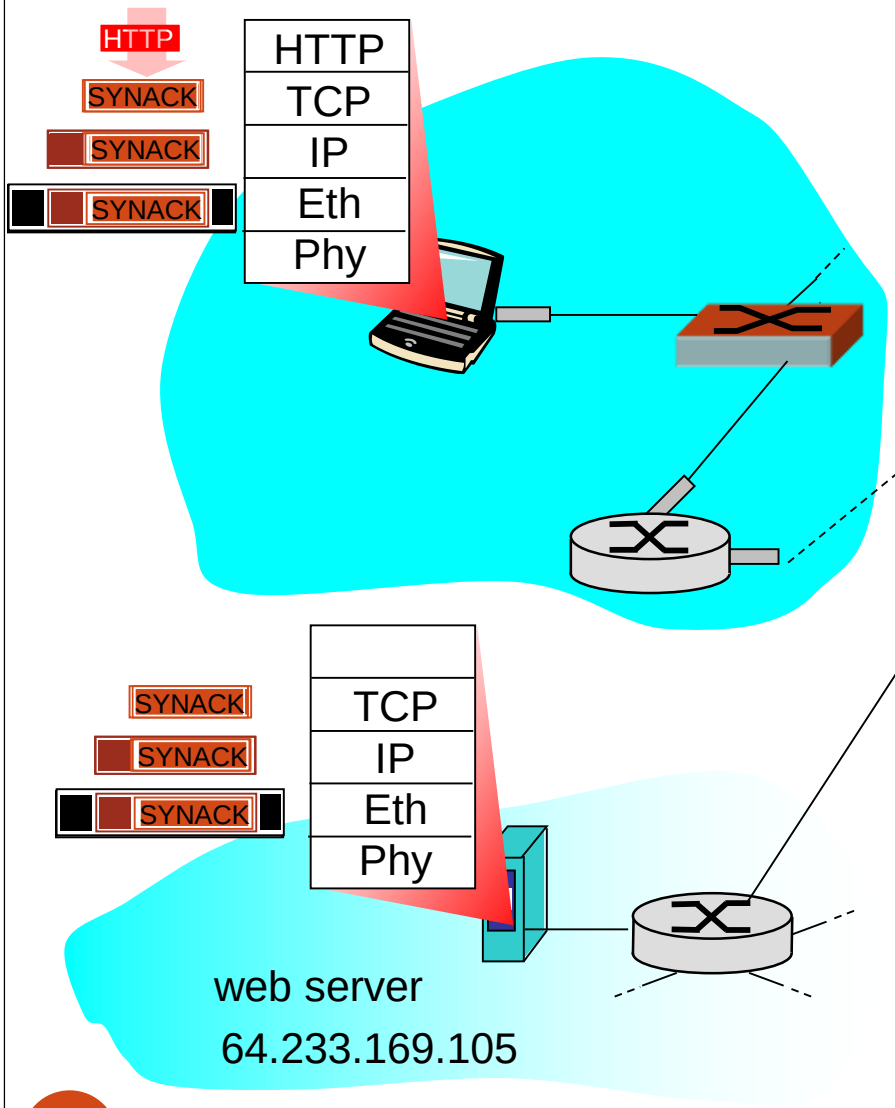
- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encasulated in Eth. In order to send frame to router, need MAC address of router interface: *ARP*
- *ARP query* broadcast, received by router, which replies with *ARP 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



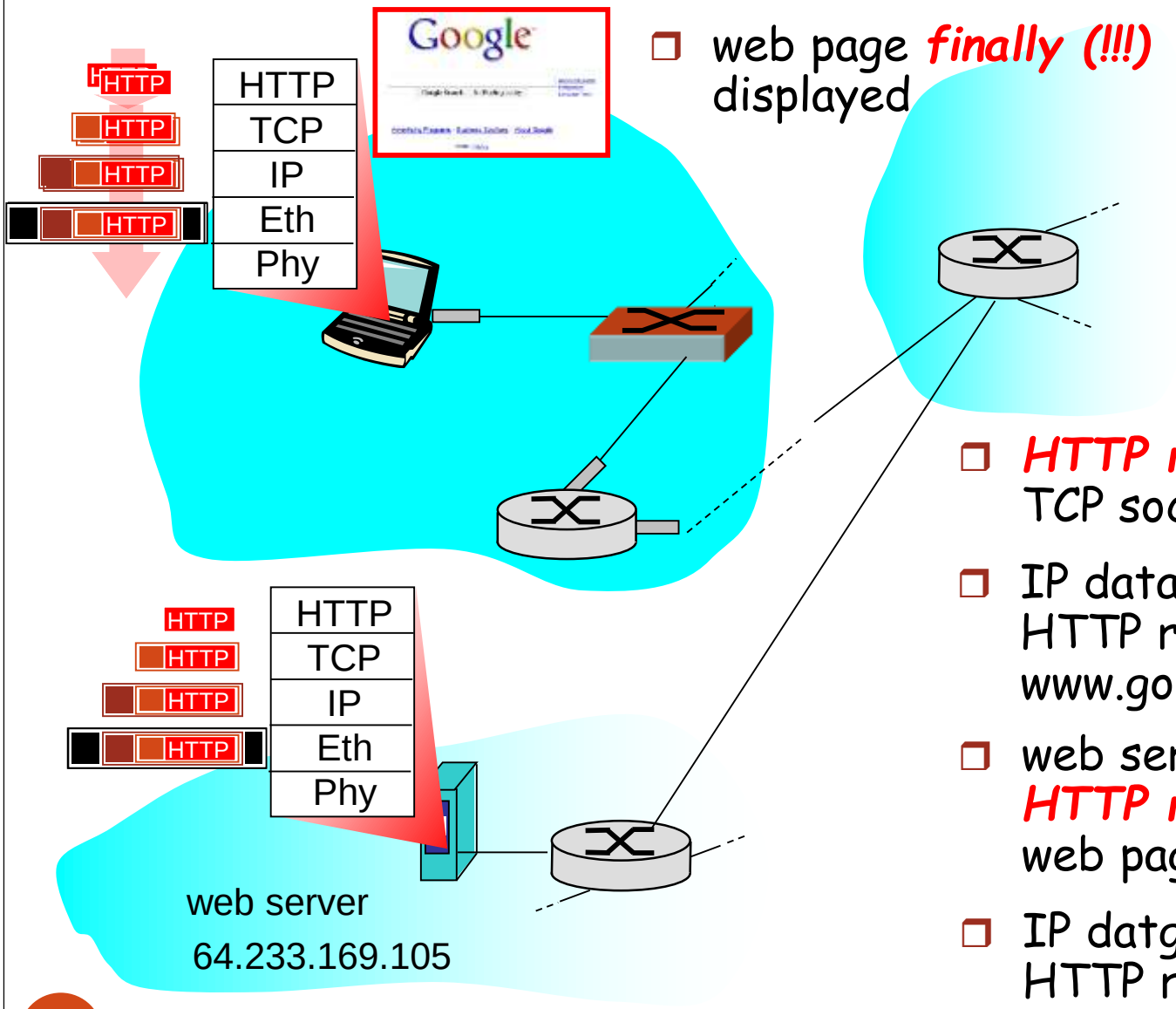
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by ***RIP, OSPF*** and/or ***BGP*** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

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



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

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 datgram containing HTTP reply routed back to client