DIPARTIMENTO DI INFORMATICA
Via Salaria 113 - 00198, Roma

SAPIENZA UNIVERSITÀ DI ROMA

Chapter 5
Data Link Layer

Reti di Elaboratori

Corso di Laurea in Ingegneria Informatica

Università degli Studi di Roma "La Sapienza"

Canale A-L

Prof.ssa Chiara Petrioli

Parte di queste slide sono state prese dal materiale associato al libro *Computer Networking: A Top Down Approach* , 5th edition.

Thanks also to Antonio Capone, Politecnico di Milano, Giuseppe Bianchi and Francesco LoPresti, Un. di Roma Tor Vergata

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

5: DataLink Layer 5a-2

Link Layer

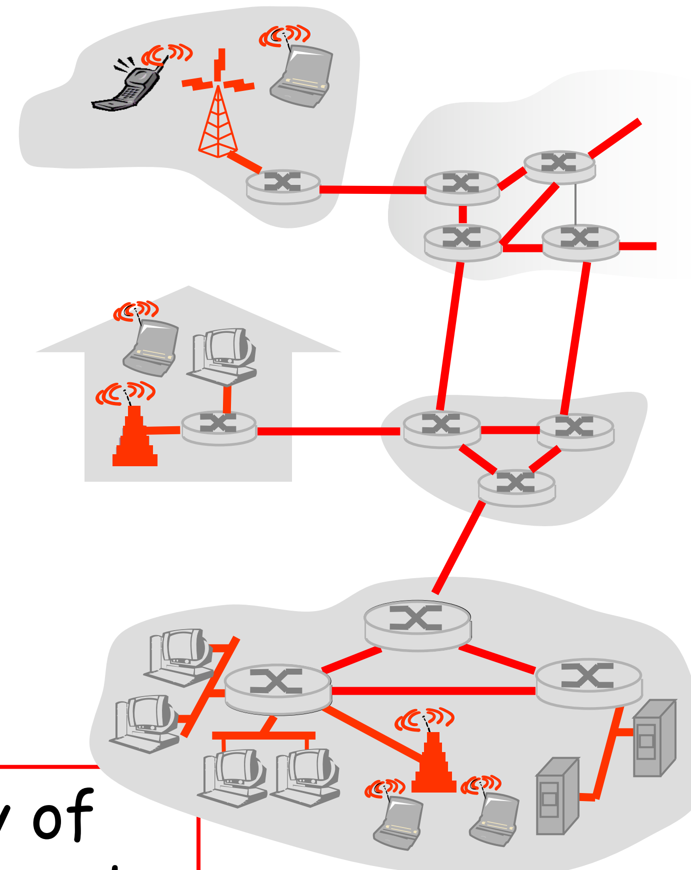
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches

5: Link Layer 5-3

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

5a-5

Link Layer Services

- *Framing:* understand where a frame starts and ends
- *link access*
  - **channel access if shared medium**
    - **avoids or limits the effect of collisions over a broadcast channel**
- addressing
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *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

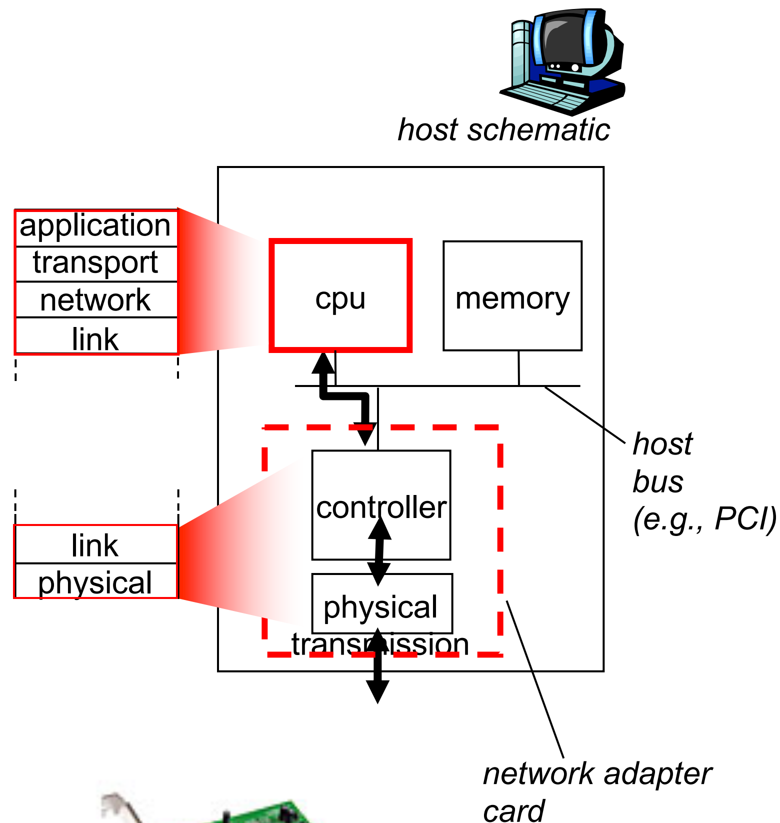
Link Layer Services (more)

- *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?
- *flow control:*
  - pacing between adjacent sending and receiving nodes

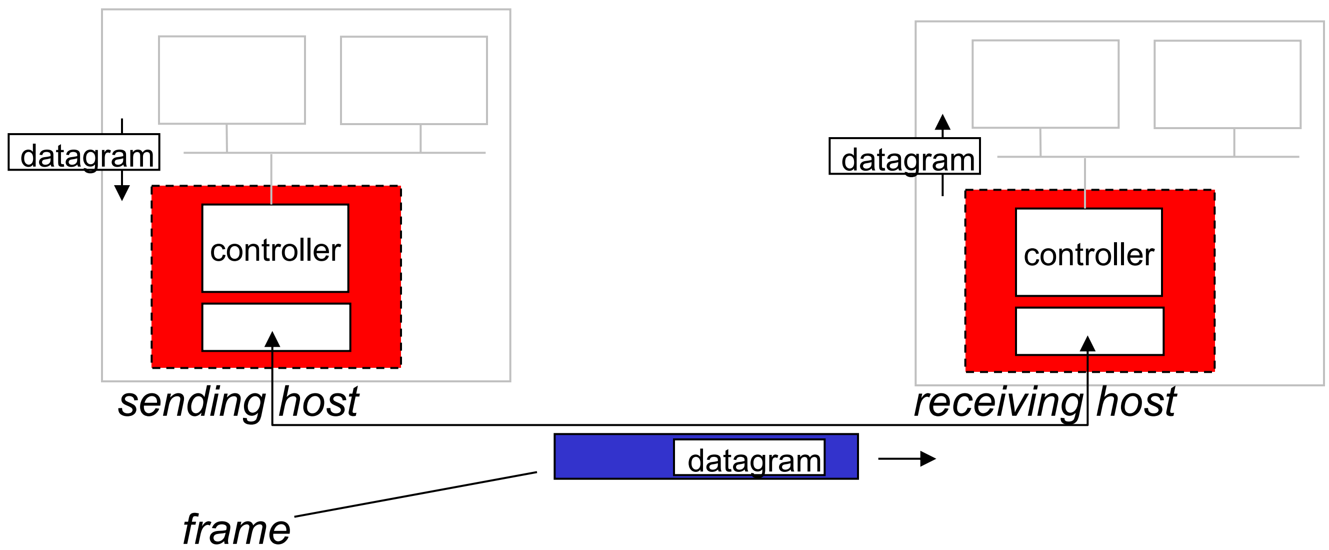
5: DataLink Layer 5a-7

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
- 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

5: DataLink Layer 5a-9

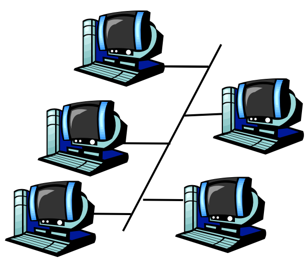
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple 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

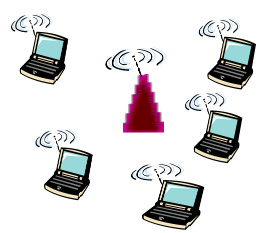
Multiple Access Links and Protocols

Two types of "links":

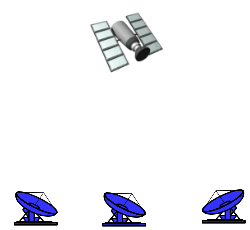
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



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

5: DataLink Layer 5a-12

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

5a-13

MAC Protocols: a taxonomy

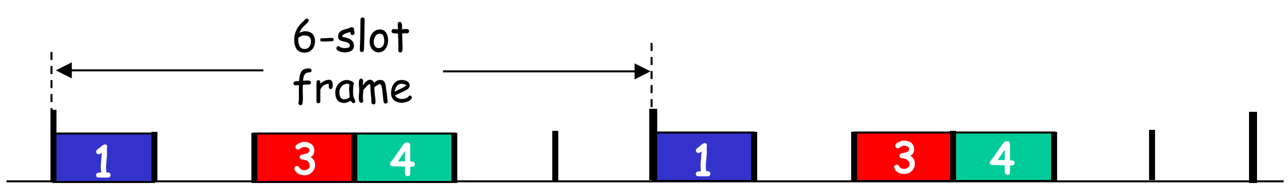
Three broad classes:

- 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

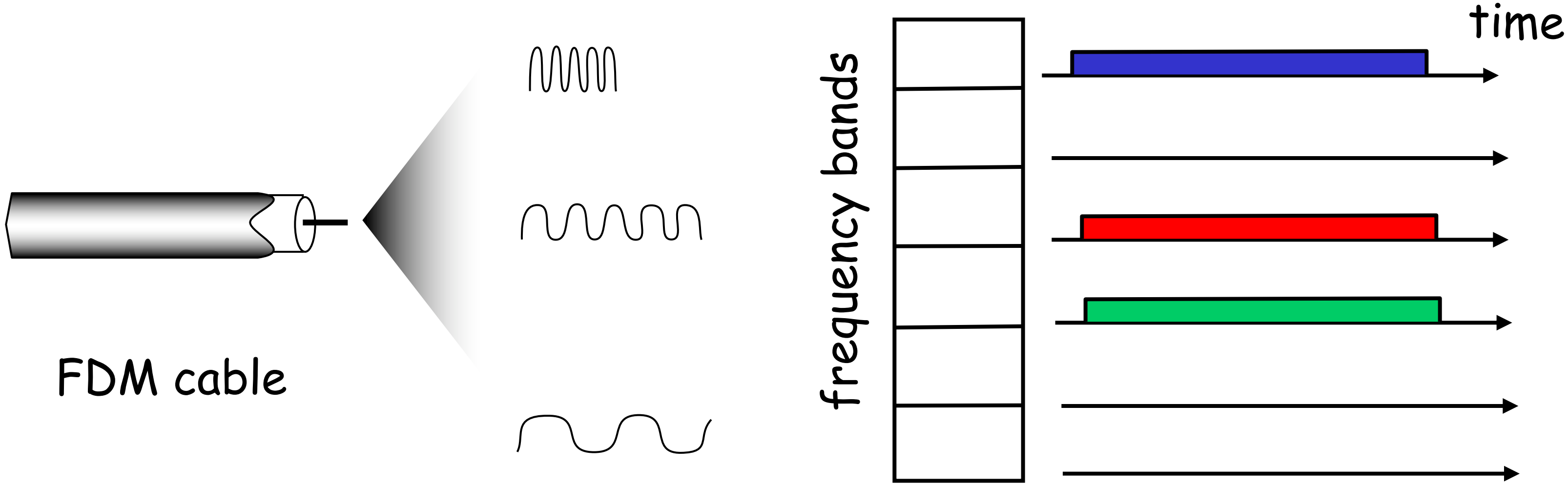
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



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



5: DataLink Layer 5a-16

TDMA/FDMA Vs. Ideal Multiple Access Protocol

Broadcast channel of rate R bps

1. when one node wants to transmit, it can send at rate R. → NOT MET BY TDMA/FDMA
2. when M nodes want to transmit, each can send at average rate R/M → MET BY TDMA/FDMA
3. fully decentralized:
   - **no special node to coordinate transmissions**
   - **no synchronization of clocks, slots**
4. simple

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

5: DataLink Layer 5a-18

Slotted ALOHA

Assumptions:

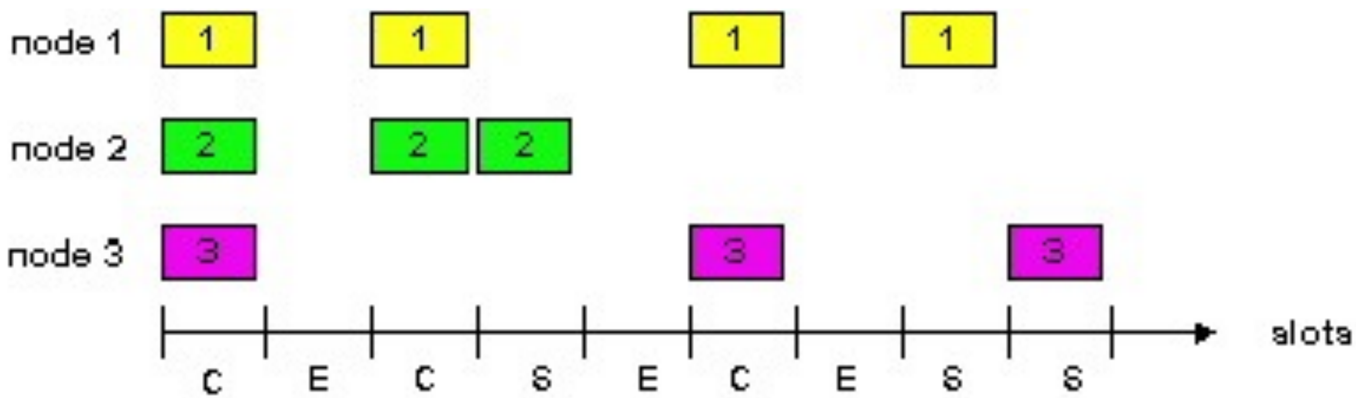
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

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

5: DataLink Layer 5-19

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:

Max efficiency = 1/e = .37

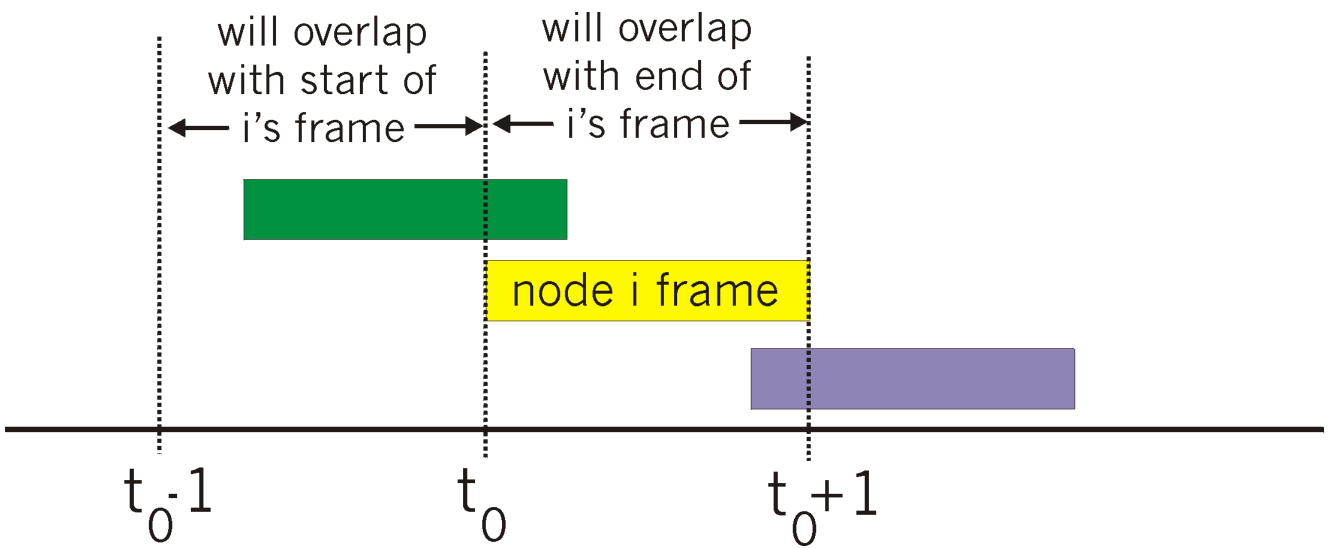
At best: channel used for useful transmissions 37% of time!

!

5: DataLink Layer 5a-21

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 $[t_0-1, t_0+1]$



5a-23

Pure Aloha efficiency

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

P(no other node transmits in $[p_0-1,p_0]$ ·

P(no other node transmits in $[p_{0,}\ p_0+1,]$

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

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

… choosing optimum p and then letting n -> infty ...

= 1/(2e) = .18

even *worse* than slotted Aloha!

5: DataLink Layer 5a-24

CSMA (Carrier Sense Multiple Access)

CSMA: listen before transmit:

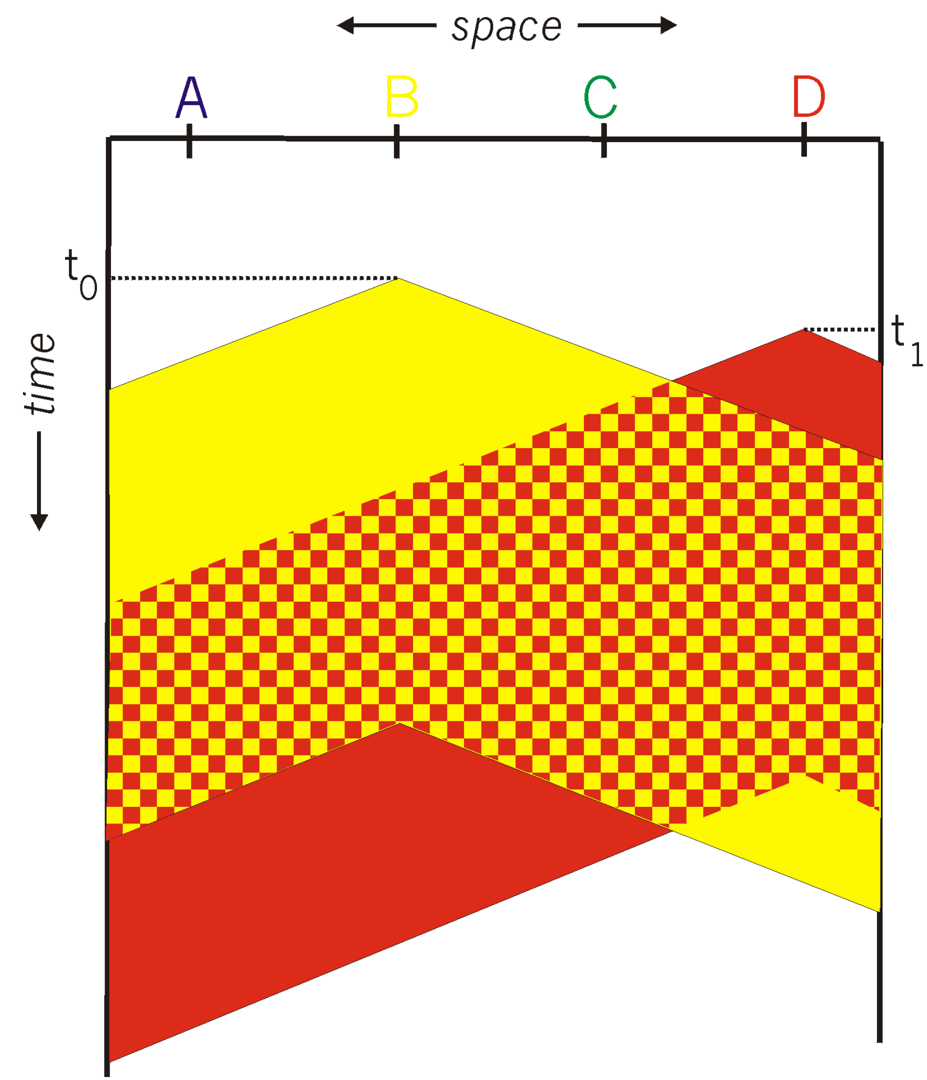
If channel sensed idle: transmit entire frame

- If channel sensed busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

spatial layout of nodes



collisions *can* still occur:

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

collision:

entire packet transmission time wasted

note:

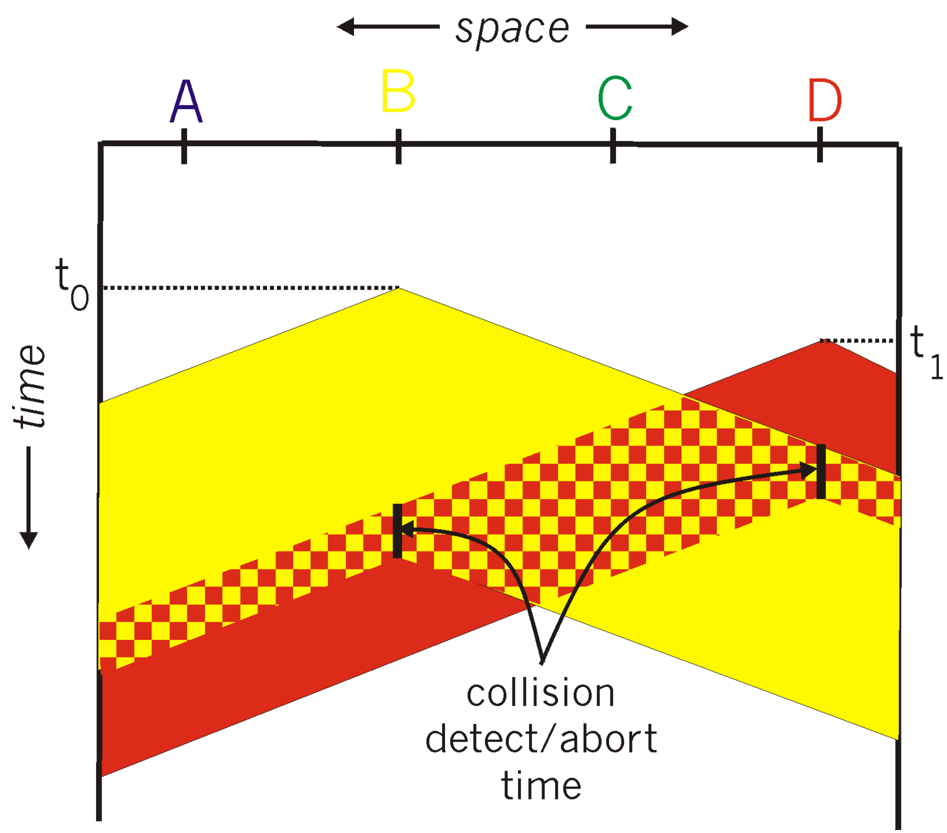
role of distance & propagation delay in determining collision probability

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

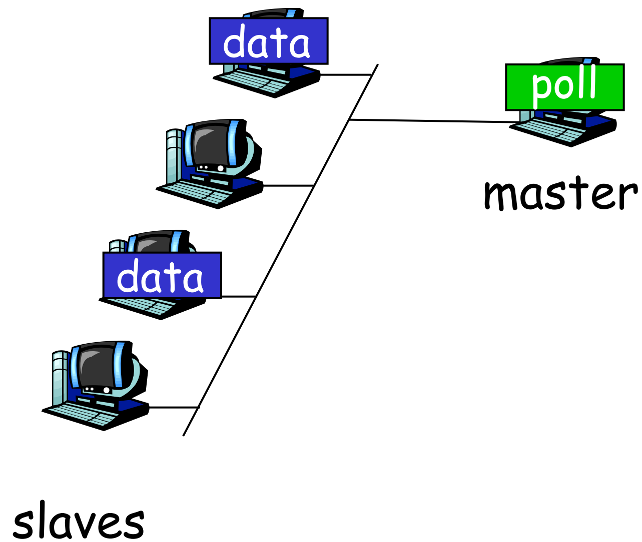
look for best of both worlds!

5: DataLink Layer 5a-29

"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)

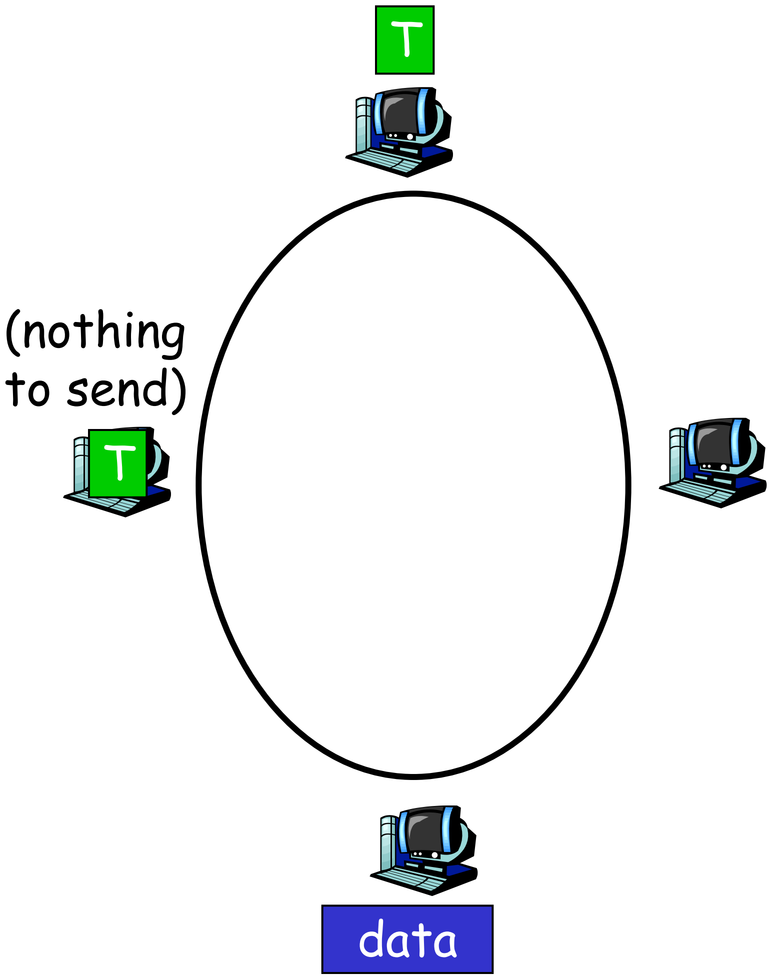


5a-30

“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

5: DataLink Layer 5a-32

Link Layer Services--framing

- PHY layer accepts only a raw bit stream and attempts to deliver to destination

  011000110000110000000010011000000100001

  - Communication is not necessarily error free
  - Multiplexing of different flows of information
    - → Data link layer breaks the bit stream up into discrete frames (FRAMING) and computes the checksum for each frame (ERROR DETECTION)

5: DataLink Layer 5a-33

Link Layer Services

Framing:

- encapsulate datagram into frame, adding header, trailer
- How to delimit frames:
  - We cannot count on some time gap (strong synch requirement and jitter requirement)
  - Character count: A field in the header specifies the number of characters in the frame (OK but loose synch in case of transmission error)
  - Starting and ending characters with character stuffing
    - ES ASCII character sequence DLE STX (Data Link Escape Start of TeXt)…DLE ETX (ETX=End of TeXt)
    - What if binary data are transmitted with sequences corresponding to DLE STX or SLE ETX occurring in the data?
    - Character stuffing: before transmitting add DLE before each of
      - such sequences in the data: DLE STX→DLE DLE STX

5: DataLink Layer 5a-34

Link Layer Services

Framing:

- encapsulate datagram into frame, adding header, trailer
- How to delimit frames:
  - Starting and ending flags with bit stuffing
    - Each frame begins and ends with a special bit pattern, e.g. 01111110 (**flag** sequence)
    - Techniques to avoid problems in case the flag sequence appears in data: whenever data link layer encounters five consecutive ones in the data add a 0 bit in the outgoing bit stream (removed at the other end of the link)→bit stuffing
    - Es.: (a) 01101111111111111110010
    - (b) 011011111**0**11111**0**11111**0**10010

Stuffed bits

Link Layer Services

Framing:

- encapsulate datagram into frame, adding header, trailer
- How to delimit frames:
  - Physical layer coding variations
    - For instance if Manchester encoding used a High-High or Low-Low sequence
  - A combination of character count and one of the other typically used

Link Layer

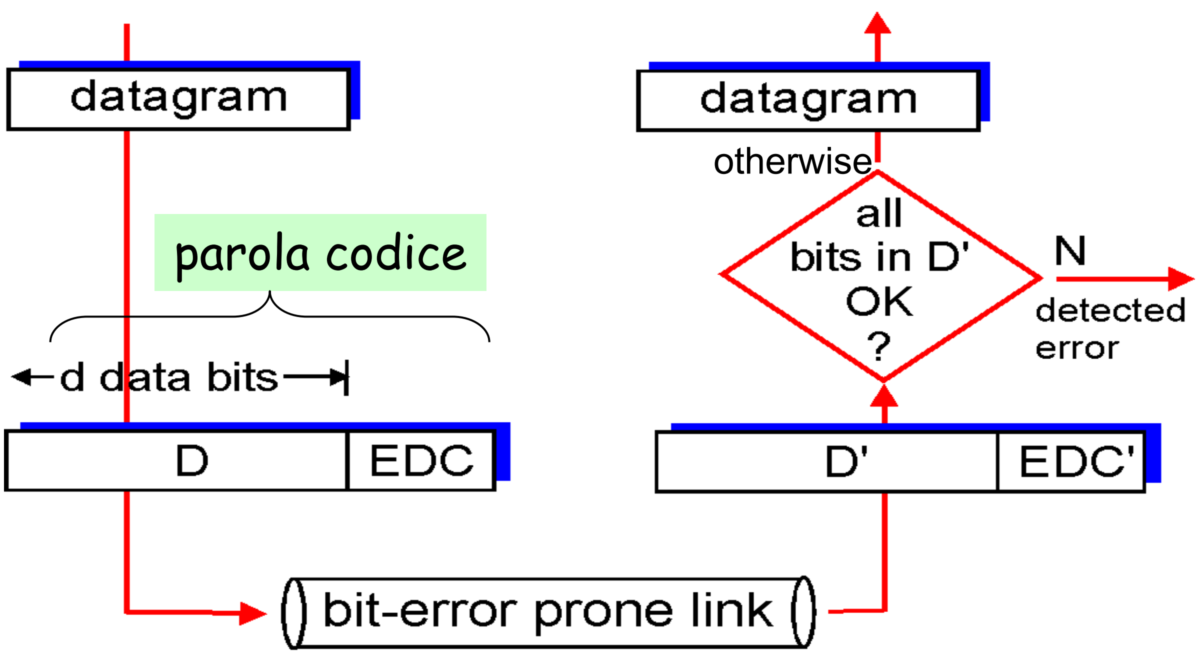
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

5: DataLink Layer 5a-37

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



5: DataLink Layer 5a-38

Distanza di Hamming

- Date due parole codice e.g., 10001001 e 10110001 è possibile determinare in quanti bit 'differiscano' (XOR delle due parole e contate il numero di 1 del risultato)
  - Il numero di posizioni nelle quali le due parole di codice differiscono determina la loro distanza di Hamming
  - Se due parole codice hanno una distanza di Hamming d ci vorranno d errori sui singoli bit per tramutare una parola di codice nell'altra
  - Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è n=m+r sono possibili 2^m messaggi dati ma non tutte le 2^n parole codice
    - la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice

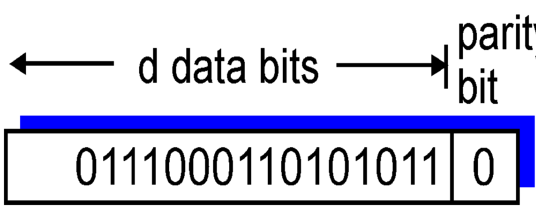
Distanza di Hamming

- Date due parole codice e.g., 10001001 e 10110001 è possibile determinare in quanti bit 'differiscano' (XOR delle due parole e contate il numero di 1 del risultato)
  - Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è $n=m+r$ sono possibili 2^m messaggi dati ma non tutti 2^n parole codice
    - la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice
  - Per fare il detection di d errori serve un codice con distanza di Hamming d+1
  - Per correggere d errori serve un codice con distanza di Hamming 2d+1

Parity Checking

Single Bit Parity:

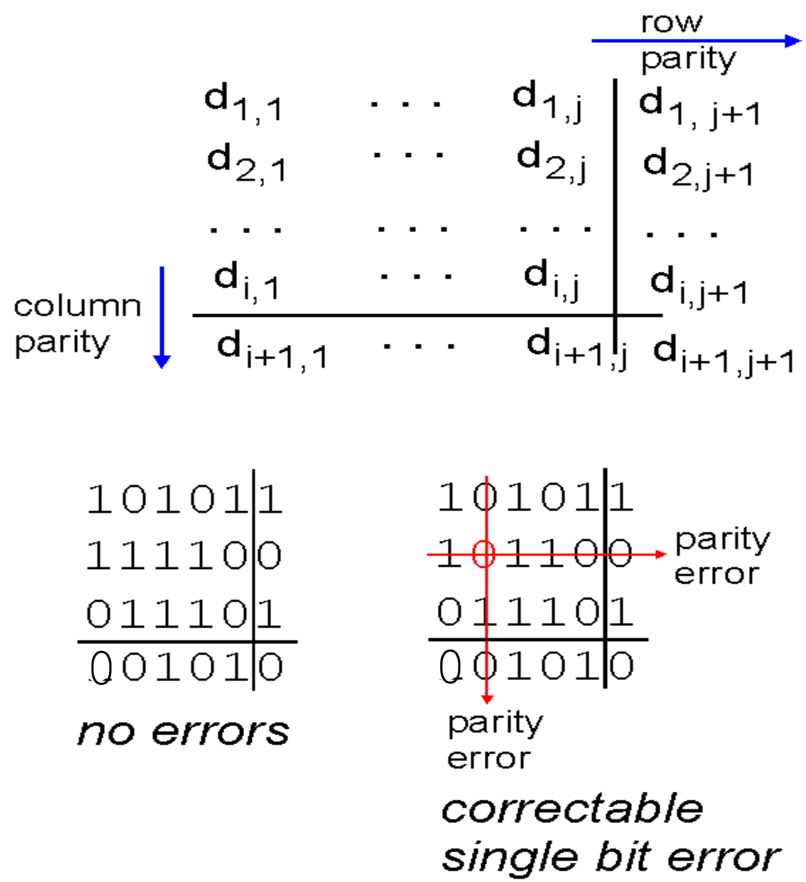
Detect single bit errors



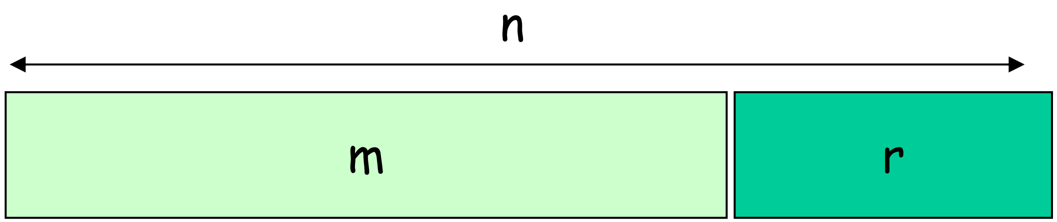
Schema di parità dispari:
Il mittente include un bit addizionale e sceglie il suo valore in modo che il numero di uno nei d+1 bit sia dispari

Two Dimensional Bit Parity:

Detect *and correct* single bit errors



Quanta ridondanza serve per correggere errori singoli?

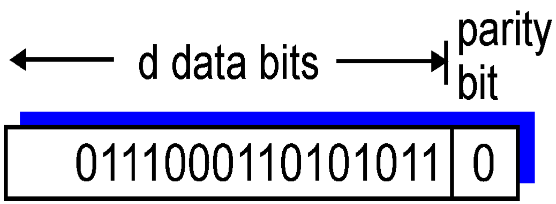


- 2^m messaggi legali
- Ciascuna parola codice legale ne ha n a distanza 1
- Ciascuno dei 2^m messaggi legali deve avere (n+1) sequenze di bit a lui associate
  - $(n+1)2^m <= 2^n$
  - $n=m+r$
  - $(m+r+1) <=2^r$
  - → Lower bound su r

Parity Checking

Single Bit Parity:

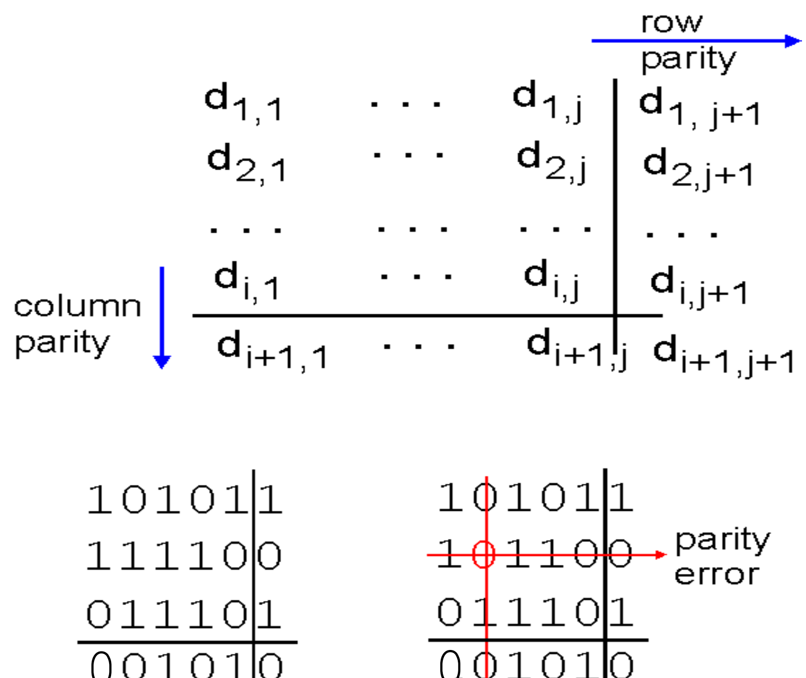
Detect single bit errors



Schema di parità dispari: Il mittente include un bit addizionale e sceglie il suo valore in modo che il numero di uno nei d+1 bit sia dispari

Two Dimensional Bit Parity:

Detect *and correct* single bit errors



Schemi semplici possono essere sufficienti nel caso di errori casuali
Cosa si può fare nel caso di errori a burst?

- Maggiore ridondanza
- Interleaving

Internet checksum (review)

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

Sender:

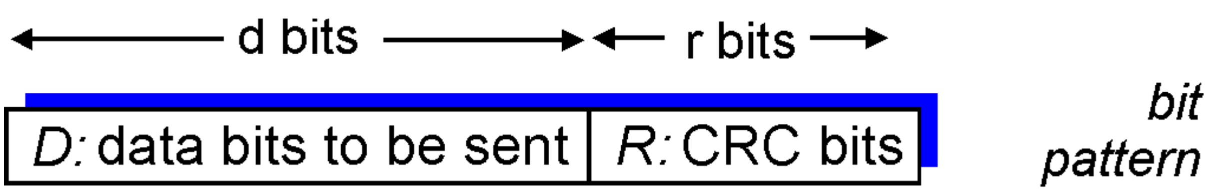
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - 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
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides received <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)



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

mathematical formula

CRC

- r è l'ordine del polinomio generatore $G(x)$
- Appendi r bit zero al messaggio $M(x)$ che ora corrisponde a $x^r M(x)$
- dividi $x^r M(x)$ per $G(x)$ modulo 2
- Sottrai (modulo 2) il resto della divisione da $x^r M(x)$ → si ottiene $T(x)$, il risultato da trasmettere

- In ricezione controlla che il resto della divisione per $G(x)$ sia 0
- Estrai la parte di messaggio $M(x)$

→Individua un burst di fino a r errori

Prestazioni di CRC

TABLE I
CRC AND TCP CHECKSUM RESULTS
(256 BYTE PACKETS ON SYSTEMS AT STORTEK)

system	code	% remaining	splices
st05	Total		7186841747
46411 files	Caught by Header		3593444113
4856193 pkts	Identical data		17498067
(98-05-04)	Remaining splices		3575899567
	Missed by CRC	0.0000000000	0
	Missed by TCP	0.0459554853	1643322
st11	Total		6306945748
45627 files	Caught by Header		3152782063
6896637 pkts	Identical data		22324135
(98-05-04)	Remaining splices		3131839550
	Missed by CRC	0.0000000319	1
	Missed by TCP	0.0610412816	1911715
st23	Total		4920441461
29444 files	Caught by Header		2459789331
4372688 pkts	Identical data		50703652
(98-05-04)	Remaining splices		2409948478
	Missed by CRC	0.0000000830	2
	Missed by TCP	0.0568444518	1369922
st25	Total		8748322301
38187 files	Caught by Header		4372322214
9531889 pkts	Identical data		65900443
(98-05-04)	Remaining splices		4310099644
	Missed by CRC	0.0000000464	2
	Missed by TCP	0.1103037608	4754202
st27	Total		5012189213
22319 files	Caught by Header		2505005350
5461908 pkts	Identical data		16574413
(98-05-04)	Remaining splices		2490609450
	Missed by CRC	0.0000000402	1
	Missed by TCP	0.0439271199	1094053

TABLE II
CRC AND TCP CHECKSUM RESULTS (256 BYTE PACKETS ON SYSTEMS AT SICS)

system	code	% remaining	splices
sics.se	Total		3183838883
/src1	Caught by Header		1594737950
48,817 files	Identical data		11000914
3,520,967 pkts	Remaining splices		1578100019
(11-24-97)	CRC	0.0000000000	0
	TCP	0.0411719151	649734
sics.se	Total		2902904306
/src2	Caught by Header		1450715240
11,492 files	Identical data		12039586
3,162,423 pkts	Remaining splices		1440149480
(11-24-97)	CRC	0.0000000000	0
	TCP	0.0344980161	496823
sics.se	Total		12074080447
/src3	Caught by Header		6031140841
7,845 files	Identical data		12062020
13,097,058 pkts	Remaining splices		6030877586
(12-17-97)	CRC	0.0000000000	0
	TCP	0.0088341538	532777
sics.se	Total		5025946678
/src4	Caught by Header		2512845921
33,912 files	Identical data		22171407
5,496,043 pkts	Remaining splices		2490929350
(12-17-97)	CRC	0.0000000000	0
	TCP	0.0198888017	495416
sics.se	Total		21107489268
/iss1	Caught by Header		10557354562
204,601 files	Identical data		126239615
23,178,376 pkts	Remaining splices		10423895091
(12-17-97)	CRC	0.0000000192	2
	TCP	0.2238580377	23334727

Prestazioni di CRC

st27	Total		5012189213
22319 files	Caught by Header		2505005350
5461908 pkts	Identical data		16574413
(98-05-04)	Remaining splices		2490609450
	Missed by CRC	0.0000000402	1
	Missed by TCP	0.0439271199	1094053
st29	Total		5756622285
57299 files	Caught by Header		2878637775
6314509 pkts	Identical data		19999951
(98-05-04)	Remaining splices		2857984559
	Missed by CRC	0.0000000350	1
	Missed by TCP	0.0552609704	1579350
st49	Total		5696462431
17663 files	Caught by Header		2846361632
6196298 pkts	Identical data		16371605
(98-05-04)	Remaining splices		2833729194
	Missed by CRC	0.0000000000	0
	Missed by TCP	0.0766246826	2171336
st51	Total		4584391161
16864 files	Caught by Header		2290882985
4990431 pkts	Identical data		14136325
(98-05-04)	Remaining splices		2279371851
	Missed by CRC	0.0000000000	0
	Missed by TCP	0.0693654262	1581096
st52	Total		8309068498

sics.se	Total		21107489268
/issl	Caught by Header		10557354562
204,601 files	Identical data		126239615
23,178,376 pkts	Remaining splices		10423895091
(12-17-97)	CRC	0.0000000192	2
	TCP	0.2238580377	23334727
sics.se	Total		6560349785
/opt	Caught by Header		3286741967
141,453 files	Identical data		152672075
7,312,235 pkts	Remaining splices		3120935743
(11-24-97)	CRC	0.0000000320	1
0.2% executables	TCP	0.1703438788	5316323
sics.se	Total		8630623470
/solaris	Caught by Header		4318348898
98,211 files	Identical data		92736322
9,502,013 pkts	Remaining splices		4219538250
(12-17-97)	CRC	0.0000000474	2
	TCP	0.1068534691	4508723
sics.se	Total		33661656216
/cna	Caught by Header		16832727499
248,611 files	Identical data		196026754
36,859,417 pkts	Remaining splices		16632901963
(12-17-97)	CRC	0.0000000180	3
	TCP	0.1866982627	31053339

LAN Addresses and ARP

32-bit IP address:

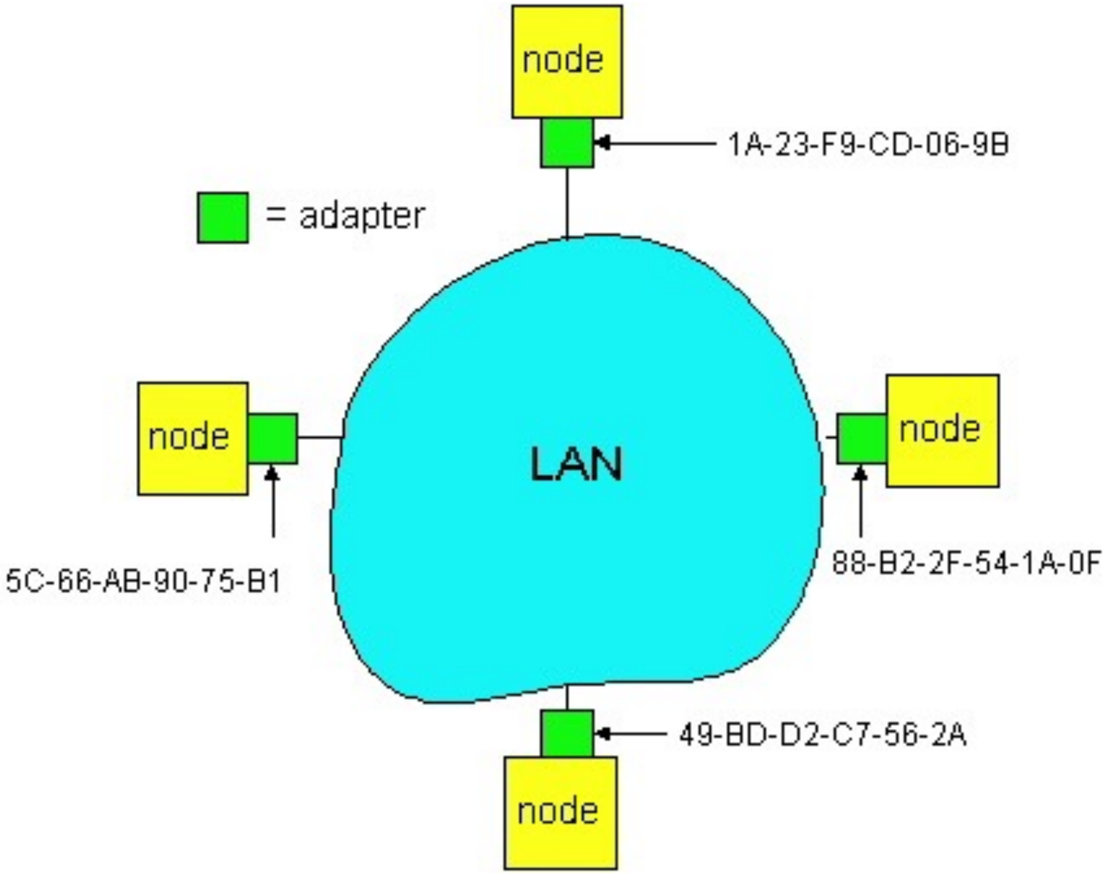
- *network-layer* address
- used to get datagram to destination IP network (recall IP network definition)

LAN (or MAC or physical or Ethernet) address:

- used to get datagram from one interface to another physically-connected interface (same network)
- 48 bit MAC address (for most LANs) burned in the adapter ROM

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



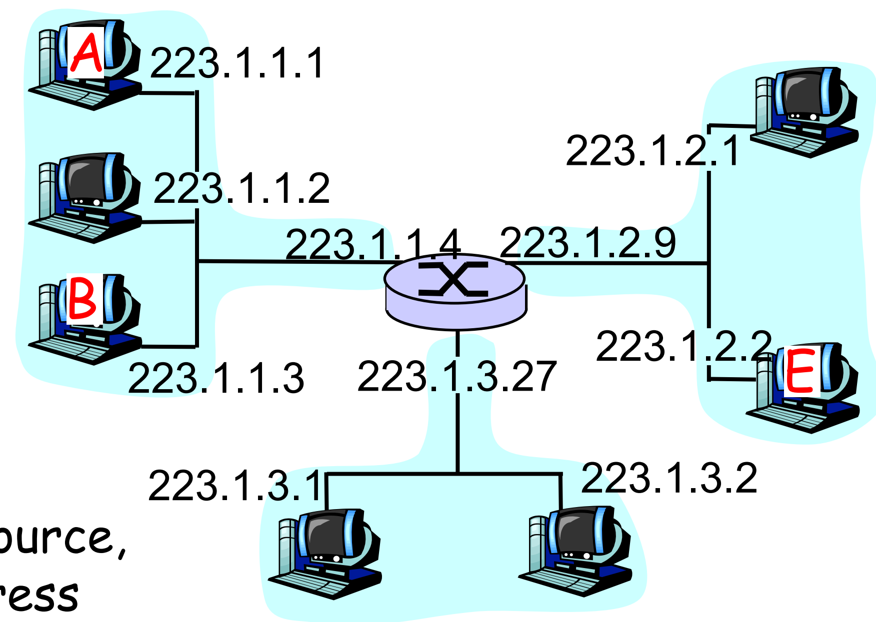
LAN Address (more)

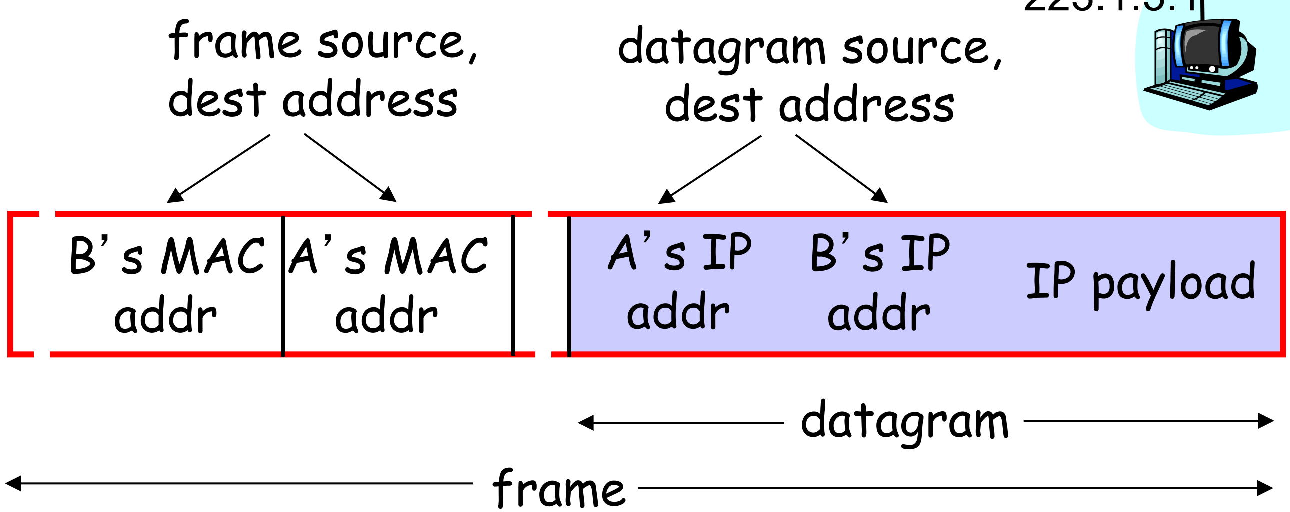
- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
- (a) MAC address: like Social Security Number
- (b) IP address: like postal address
- MAC flat address => portability
  - can move LAN card from one LAN to another
- IP hierarchical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

Recall earlier routing discussion

Starting at A, given IP datagram addressed to B:

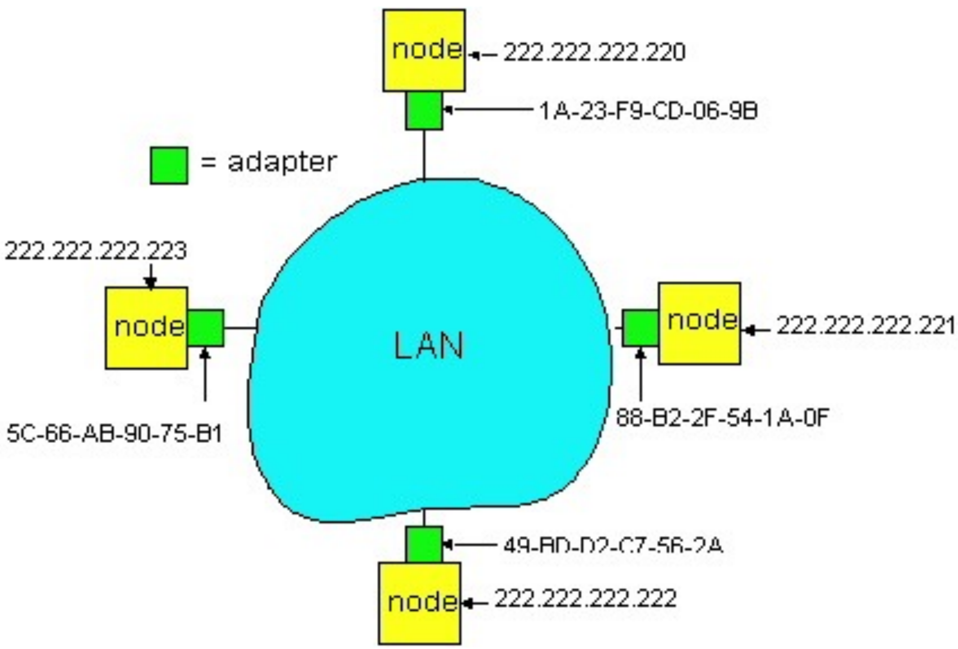
- look up net. address of B, find B on same net. as A
- link layer send datagram to B inside link-layer frame





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

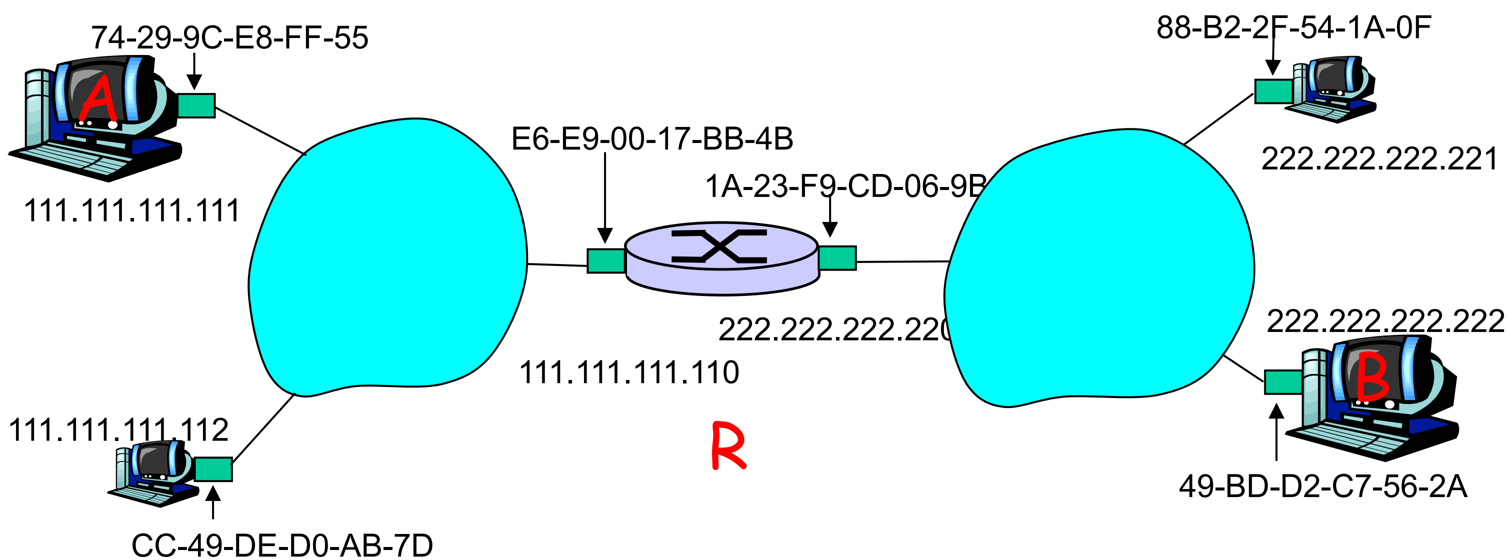
- A wants to send datagram to B, and A knows B's IP address.
- Suppose B's MAC address is not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC 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
  - USED to save ARP messages: if a receive an ARP message I cache all the informations associated to it
- 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

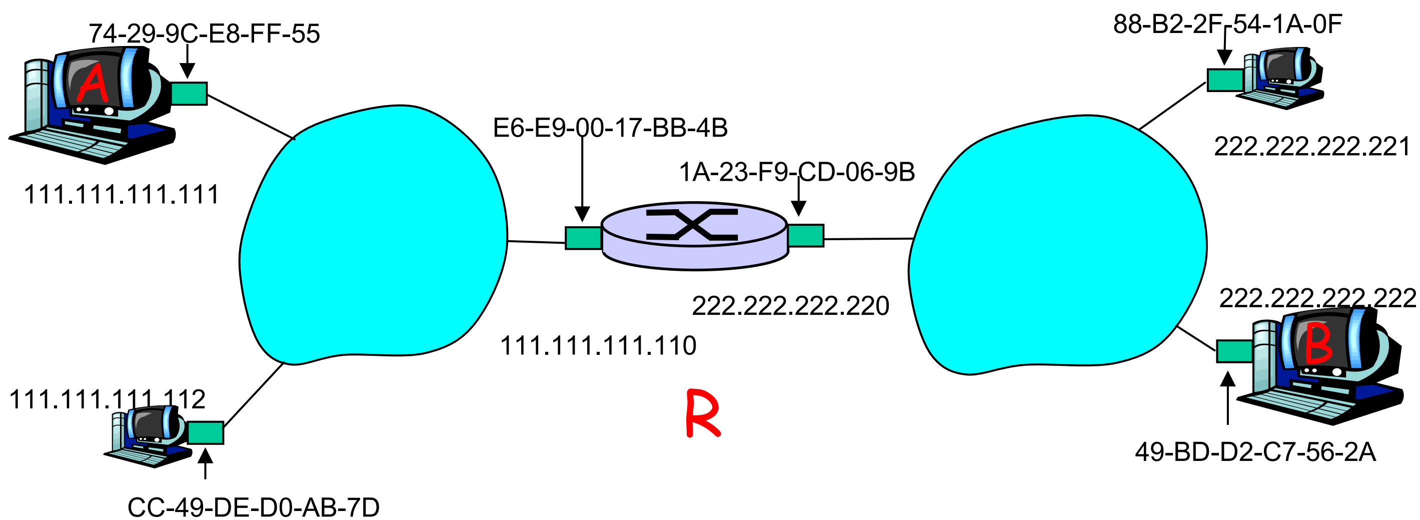


- two ARP tables in router R, one for each IP network (LAN)

5: DataLink Layer 5a-55

- 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

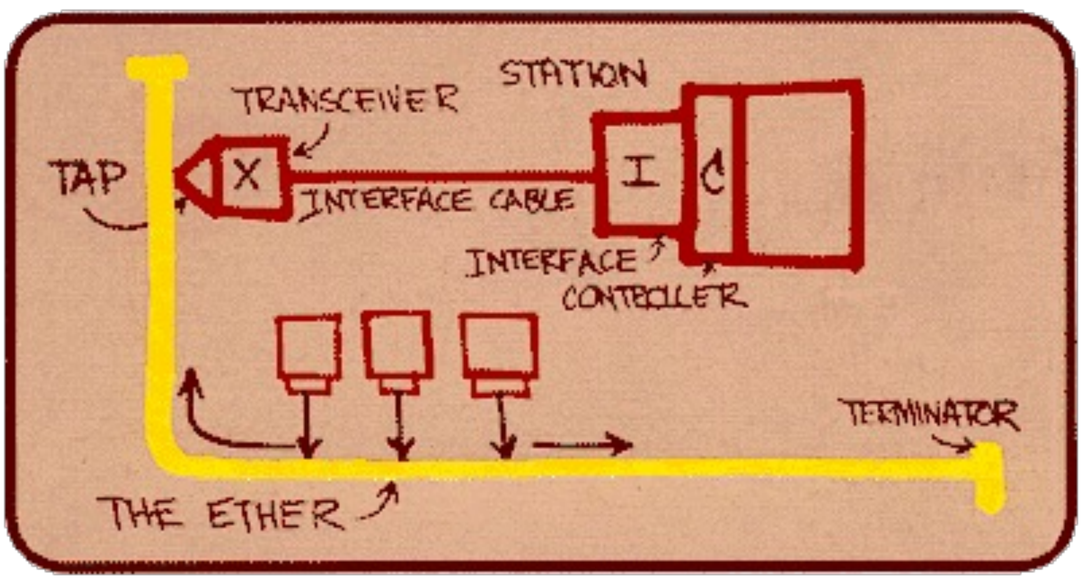
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

5: DataLink Layer 5a-57

Ethernet

"dominant" wired LAN technology:

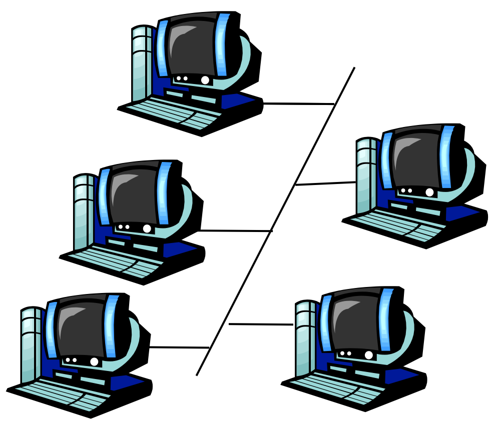
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



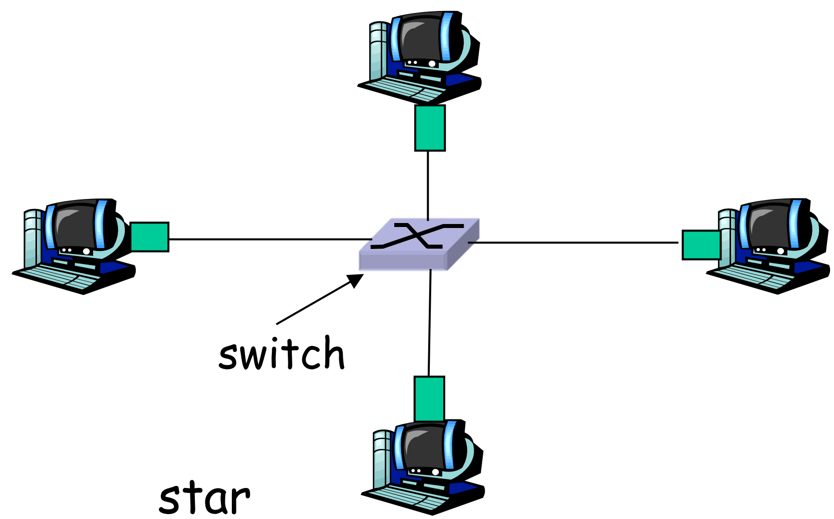
Metcalfe's Ethernet sketch

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 "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



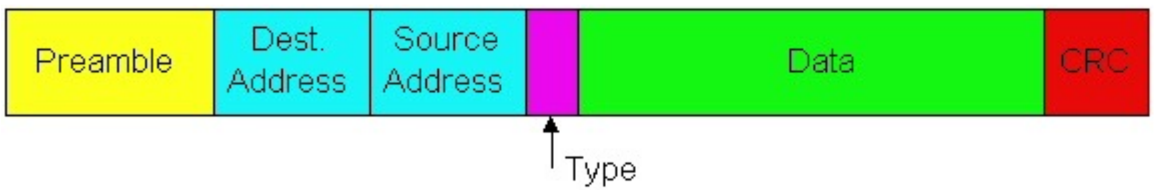
bus: coaxial cable



star

Ethernet Frame Structure

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



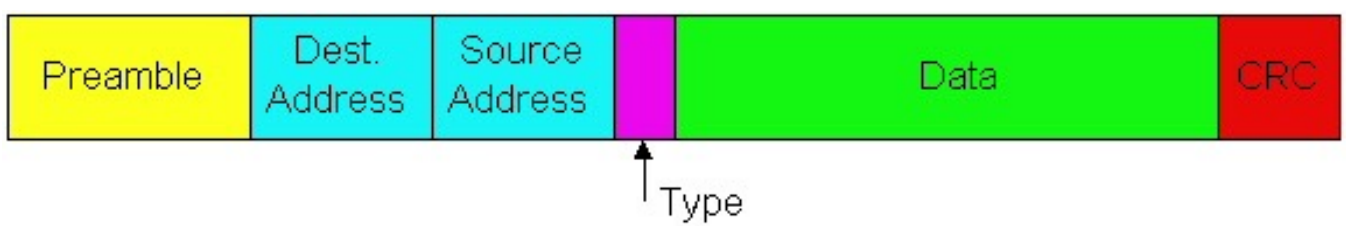
Preamble:

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

5: DataLink Layer 5a-60

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

5a-62

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 $\{0,1,2,\ldots,2^m-1\}$. NIC waits K·512 bit times, returns to Step 2

5a-63

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· 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,…,1023}

5: DataLink Layer 5a-64

CSMA/CD efficiency

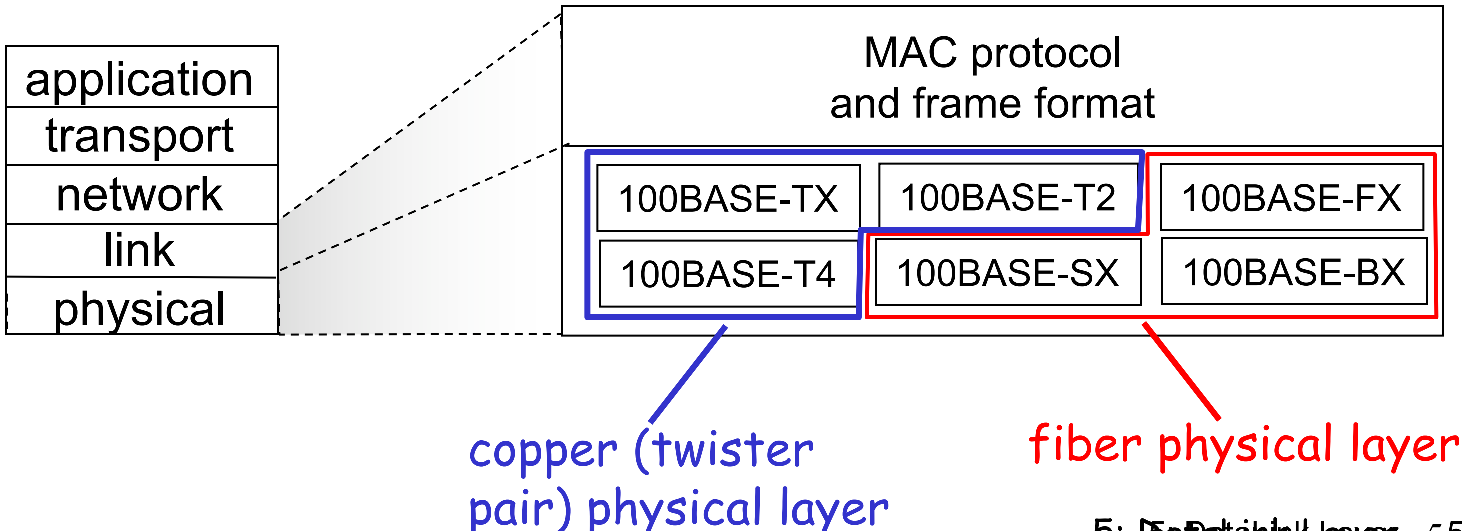
- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

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

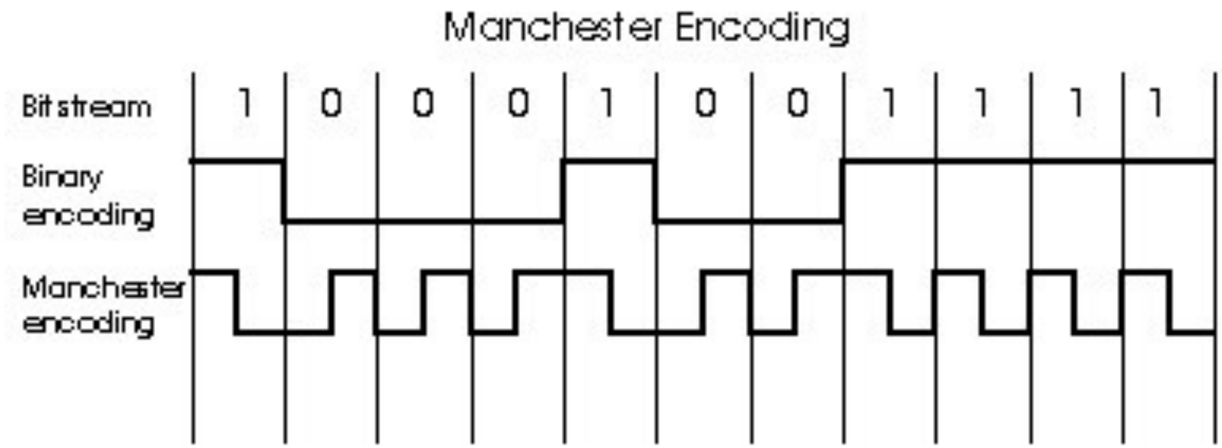
- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 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!

5: DataLink Layer 5a-67

Ethernet: some numbers..

- Slot time 512 bit times (di riferimento, la tras missione NON e' slottizzata!!)
- Interframegap 9.6 micros
- Number of times max for retransmitting a frame 16
- Backoff limit ($2^{\text{backoff limit}}$ indicates max length of the backoff interval): 10
- Jam size: 48 bits
- Max frame size: 1518 bytes
- Min frame size 64 bytes (512 bits)
- Address size: 48 bits

Link Layer

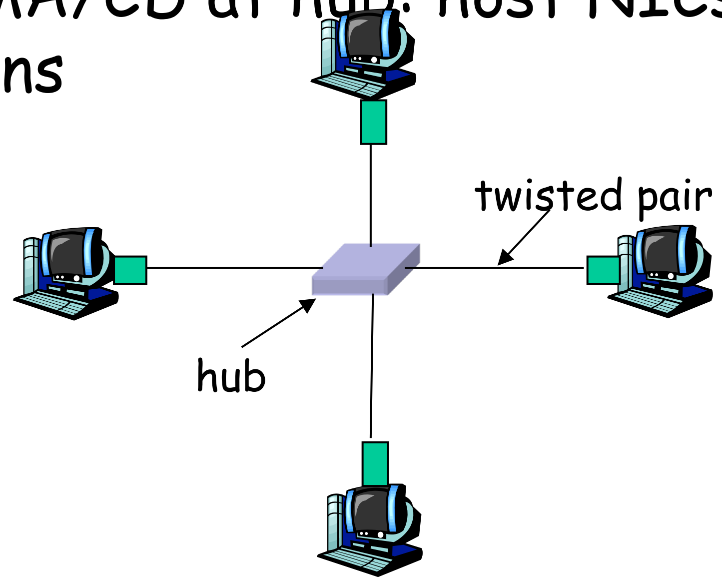
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches, LANs

5: DataLink Layer 5a-70

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



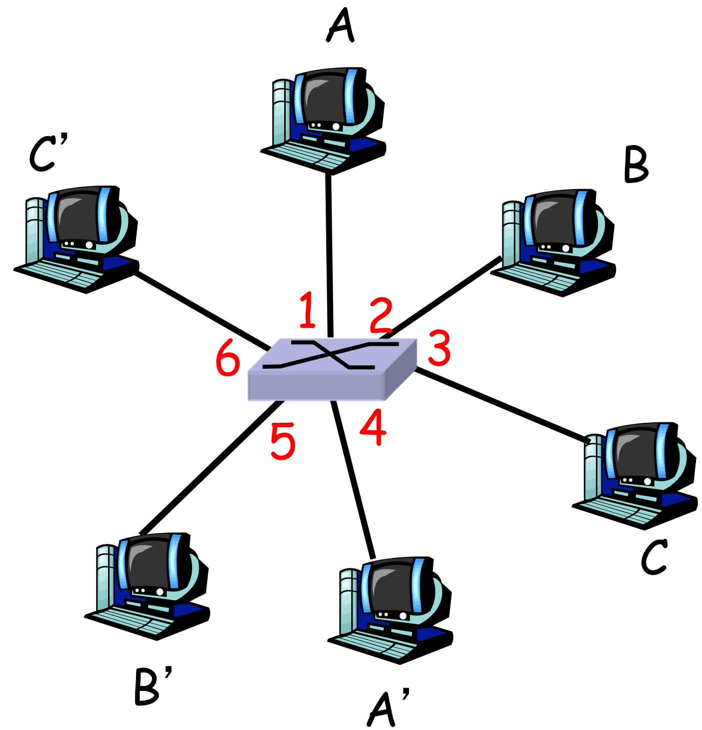
Switch

- 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
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

5: DataLink Layer 5a-72

Switch: allows *multiple* simultaneous transmissions

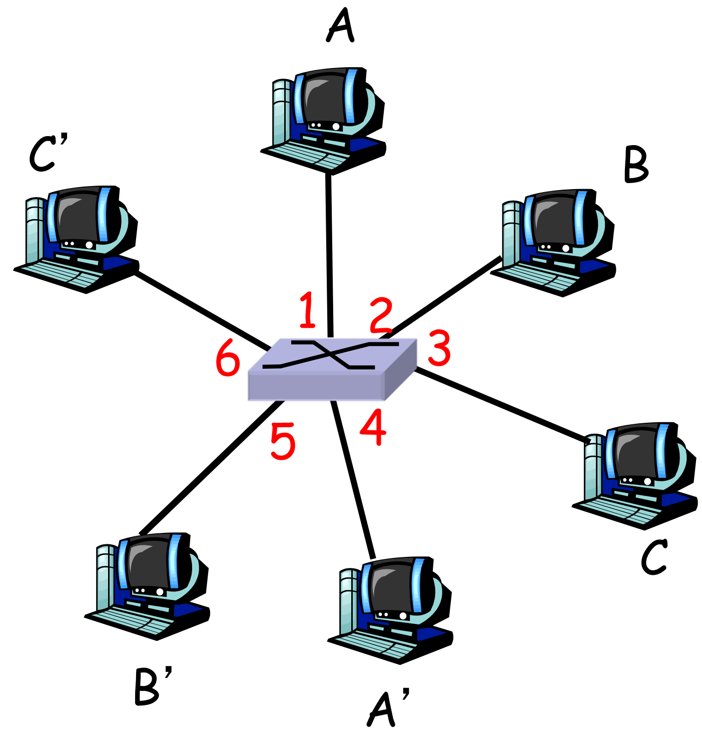
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



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

Switch Table

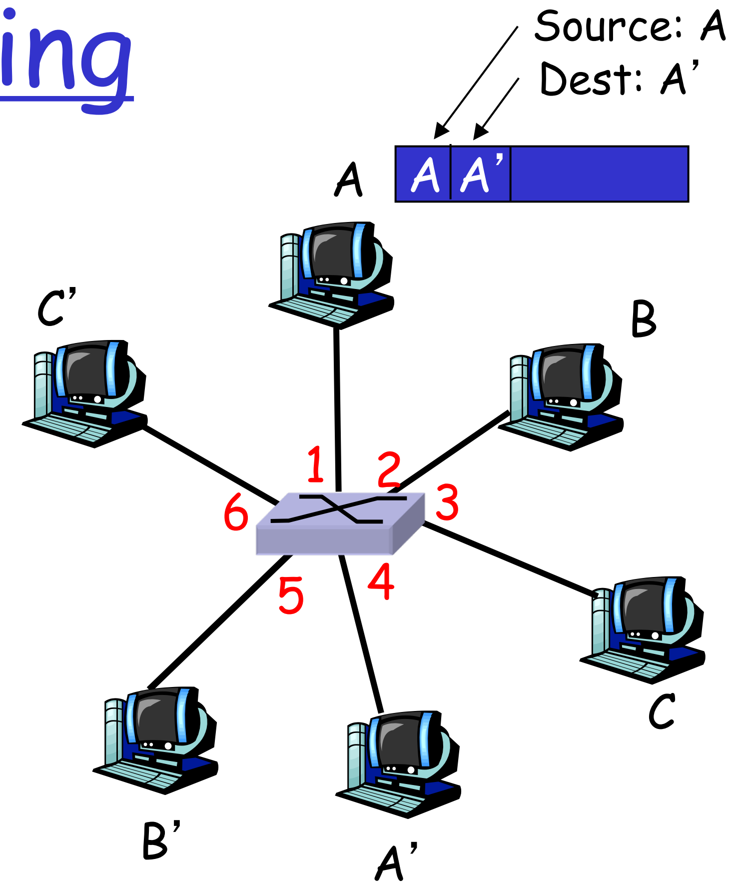
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  -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

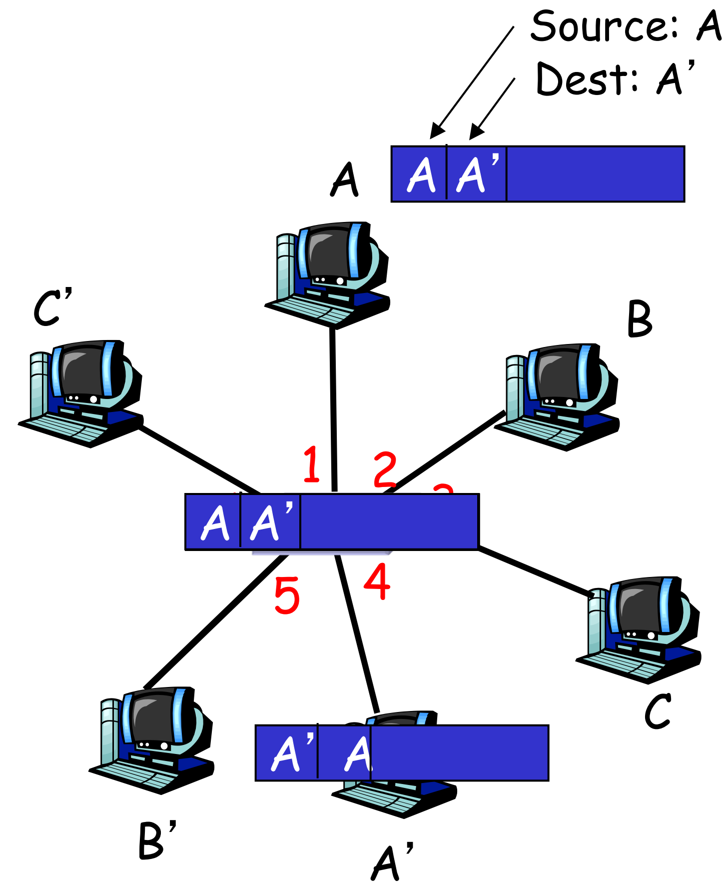
1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
   then {
   if dest on segment from which frame arrived
   then drop the frame
   else forward the frame on interface indicated
   }
   else flood

forward on all but the interface on which the frame arrived

5: DataLink Layer 5a-76

Self-learning, forwarding: example

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

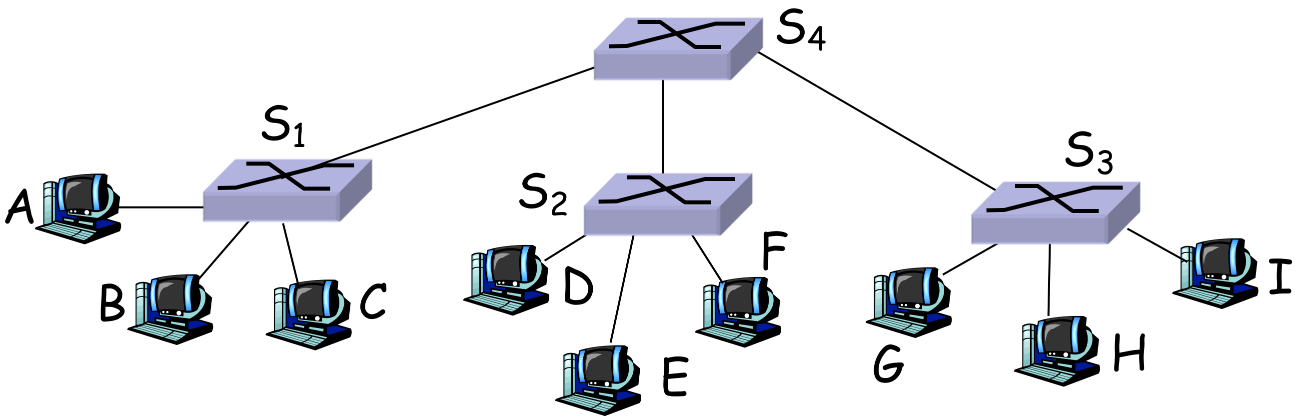


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting switches

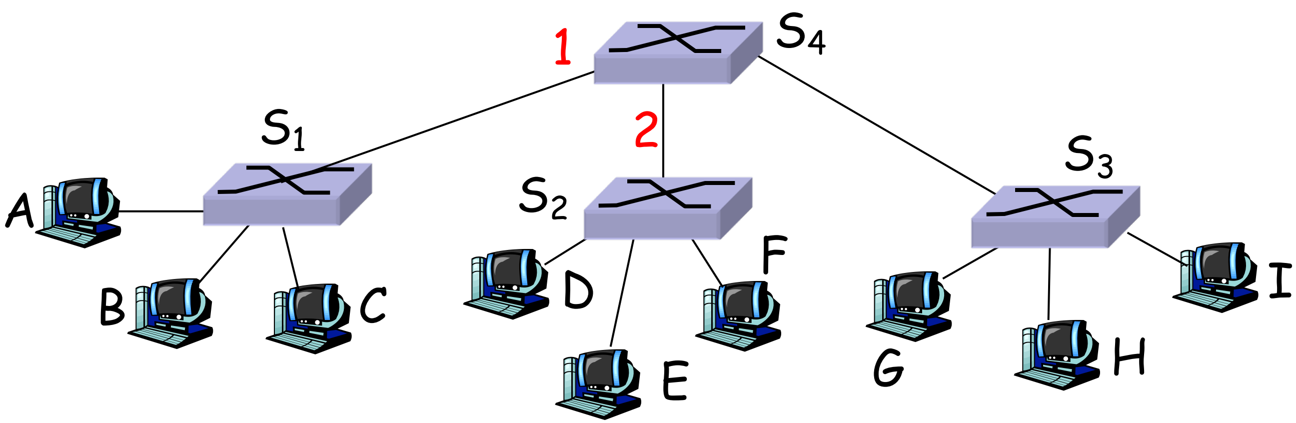
- switches can be connected together



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

Self-learning multi-switch example

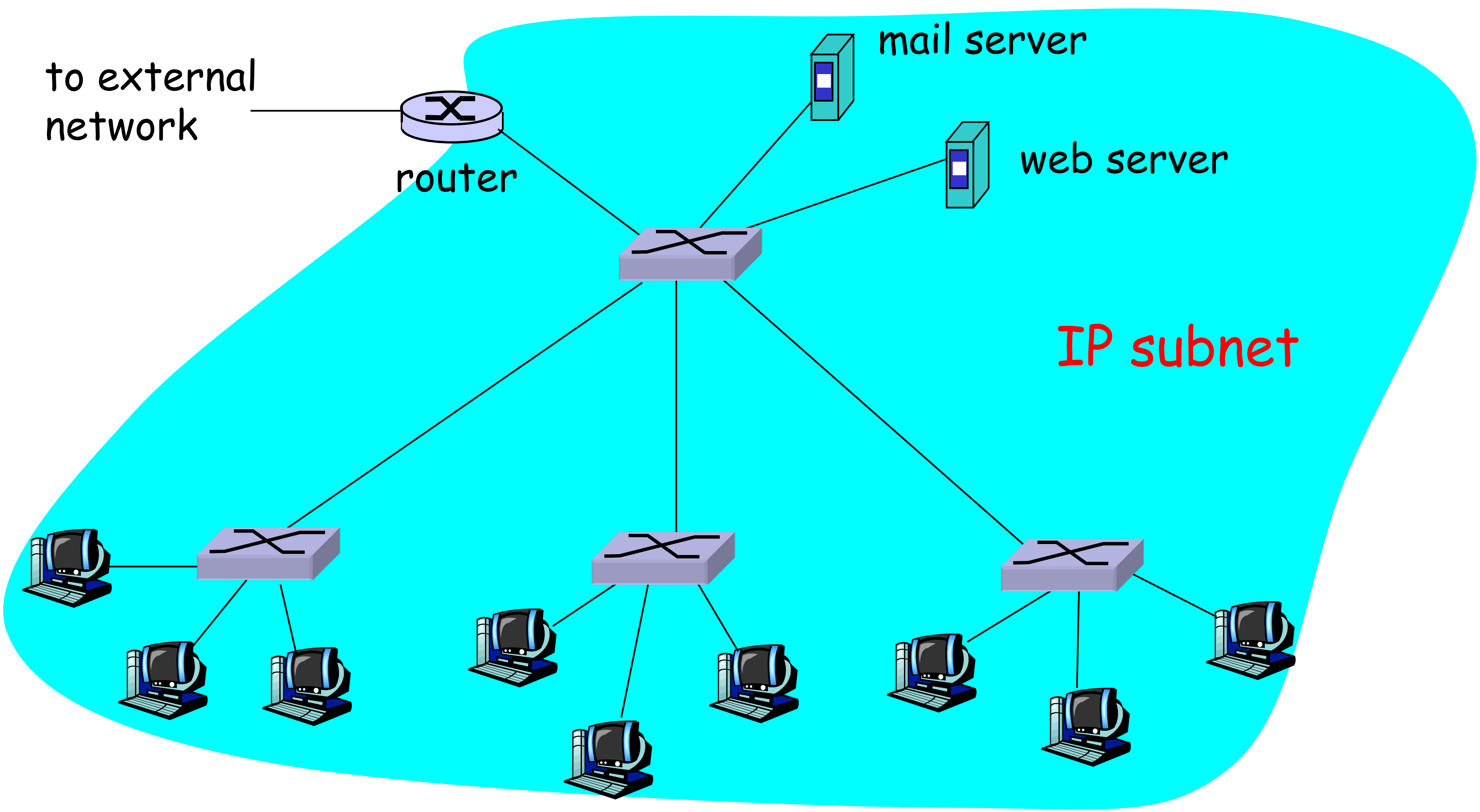
Suppose C sends frame to I, I responds to C



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

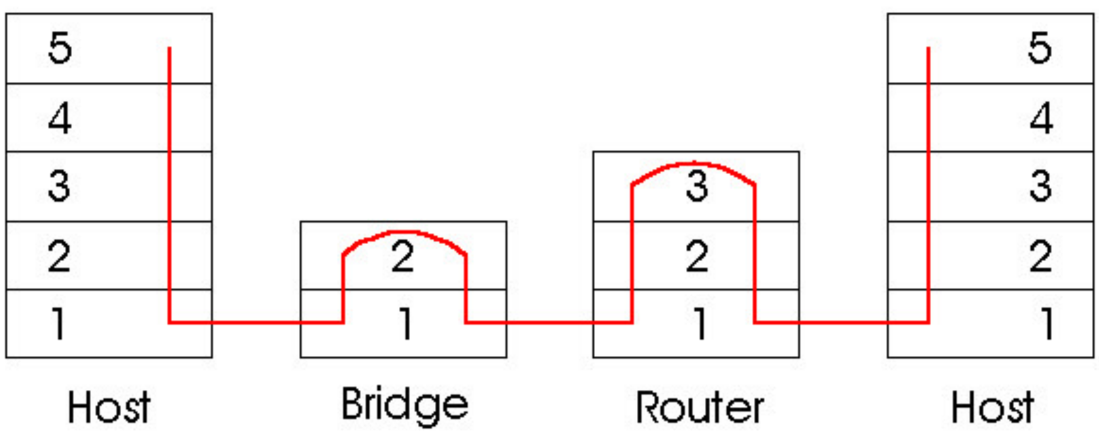
5: DataLink Layer 5a-79

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



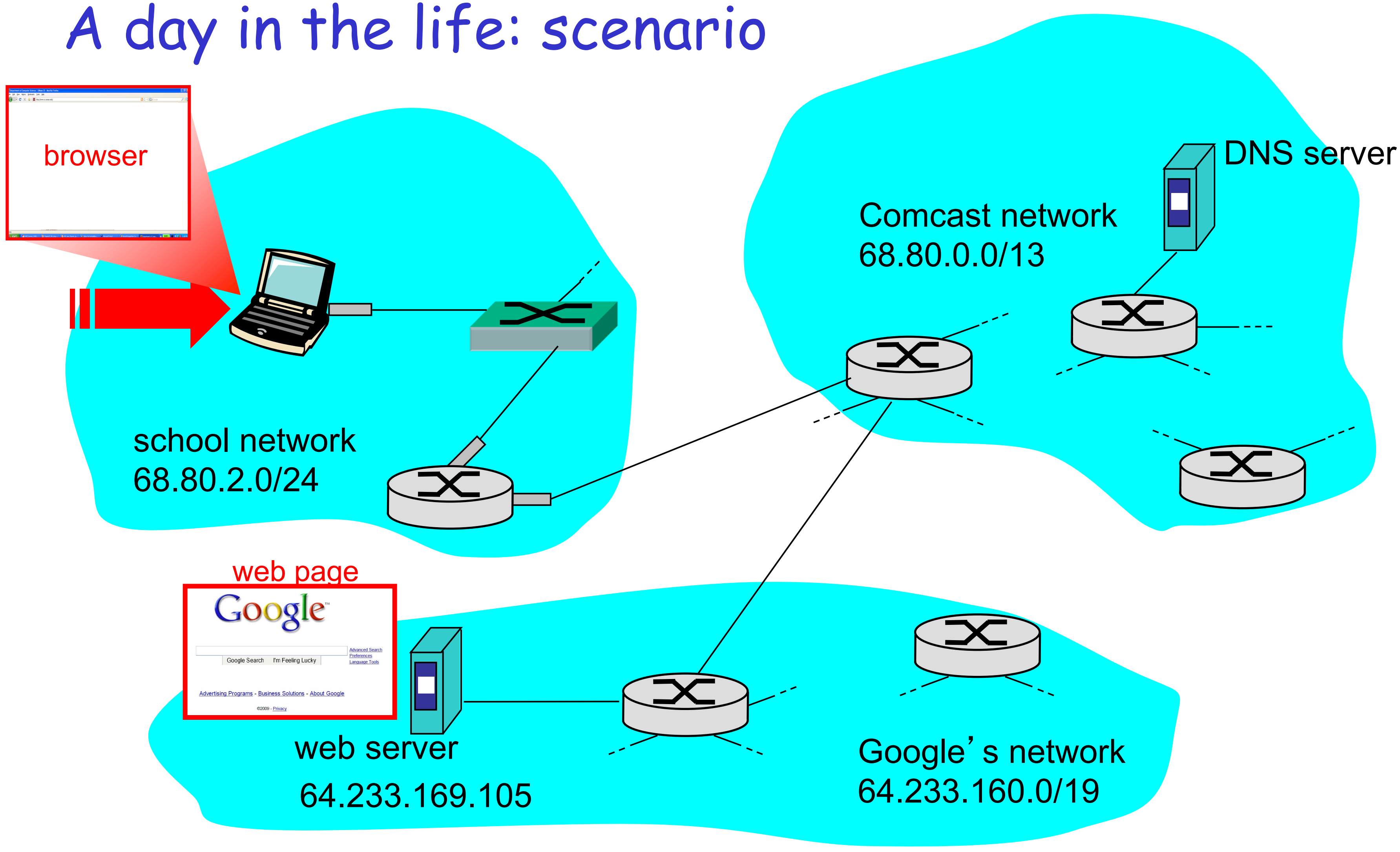
5a-81

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

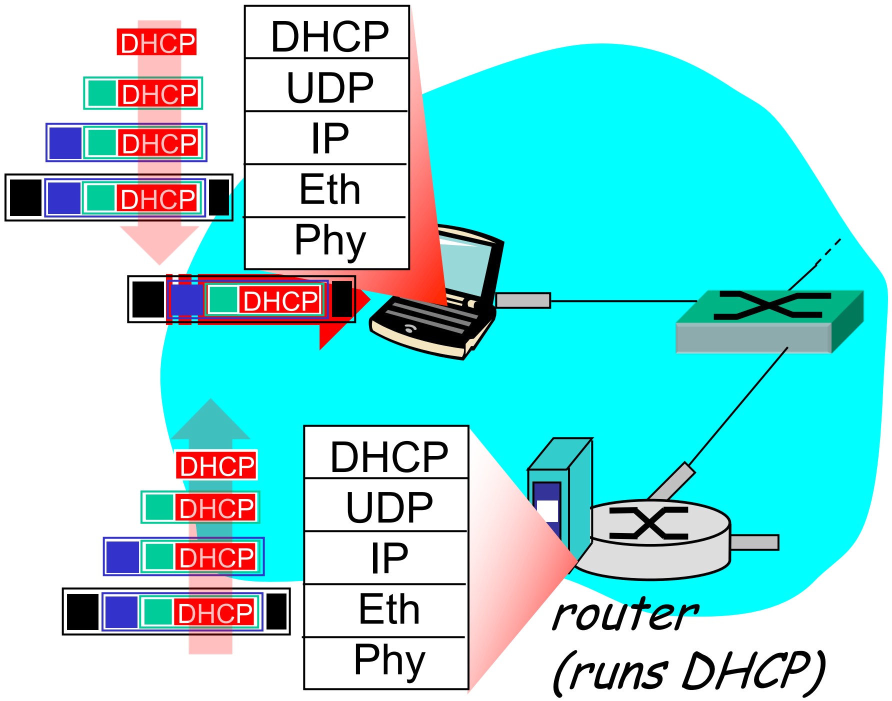
5: DataLink Layer 5a-82

A day in the life: scenario



5: DataLink Layer 5-83

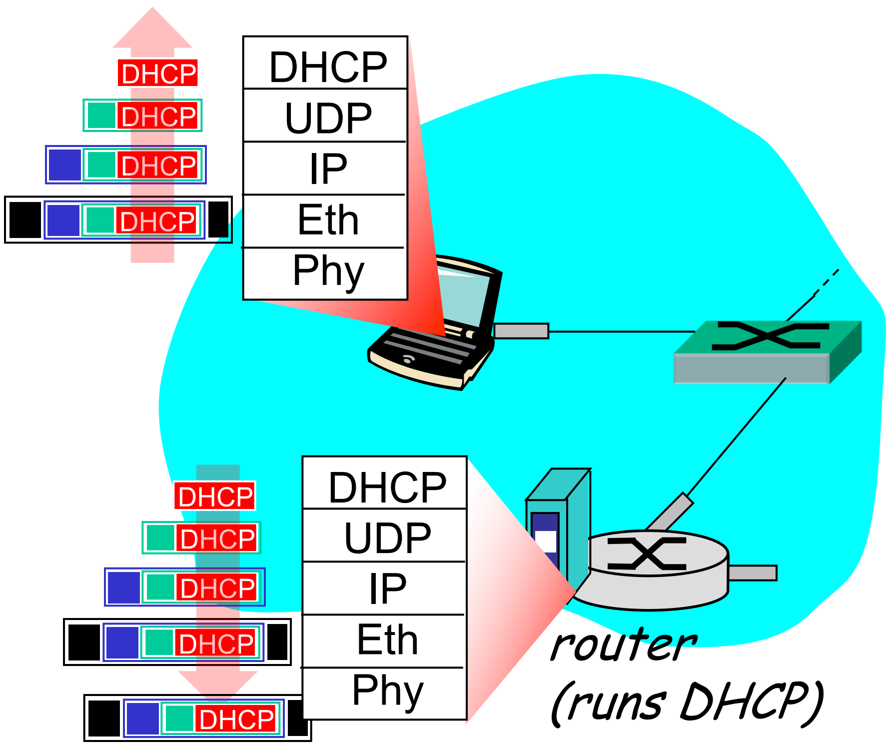
A day in the life… connecting to the Internet



- 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: FFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- Ethernet ***demux'ed*** to IP demux'ed, UDP demux'ed to DHCP

5: DataLink Layer 5-84

A day in the life… connecting to the Internet

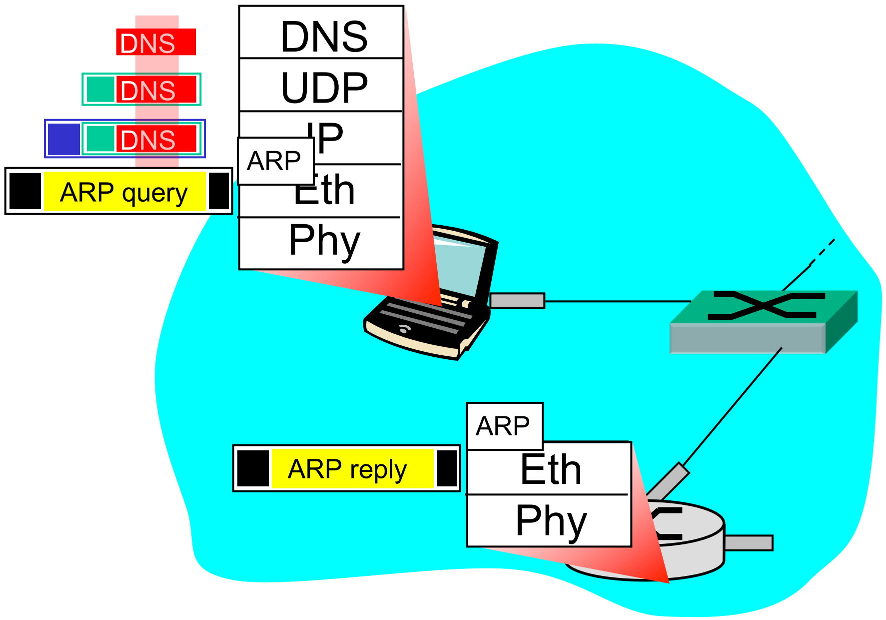


- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (*switch learning*) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

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

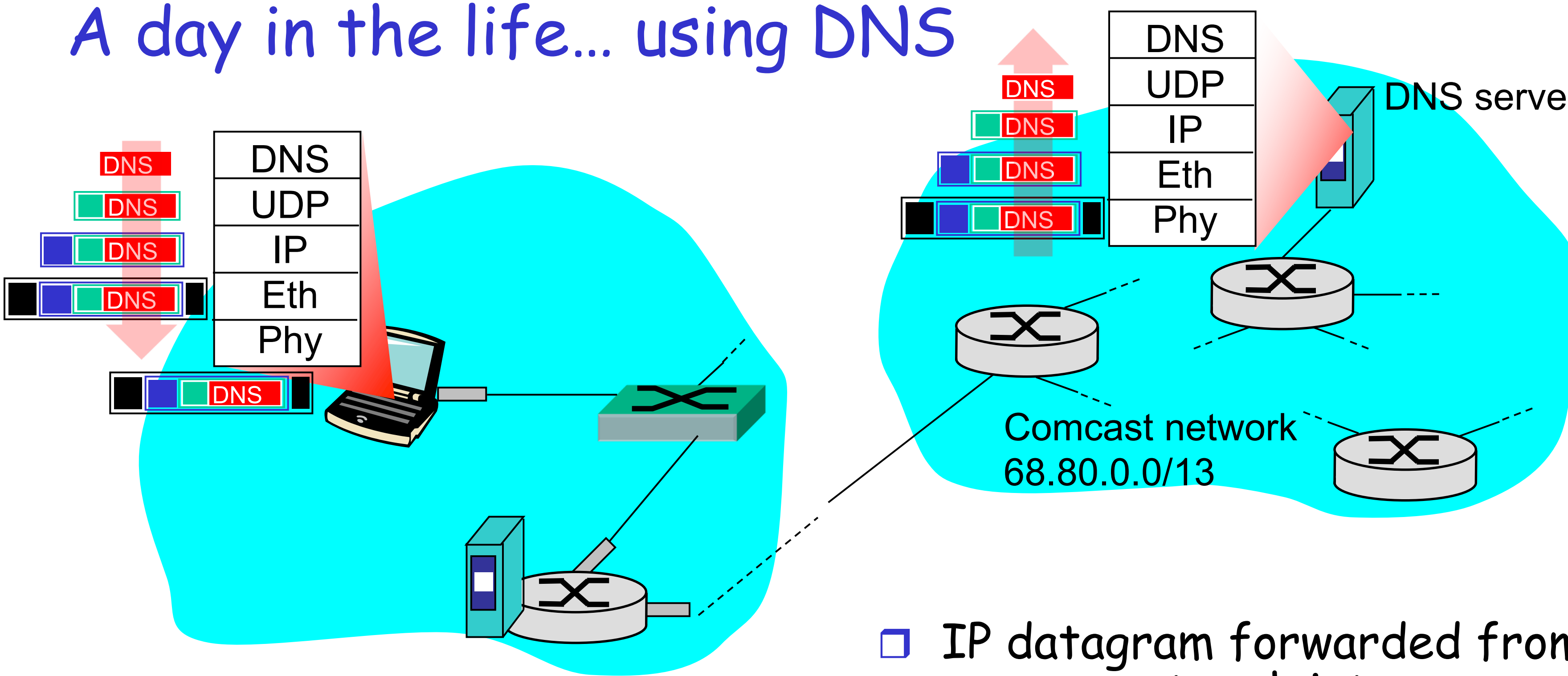
5: DataLink Layer 5a-85

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

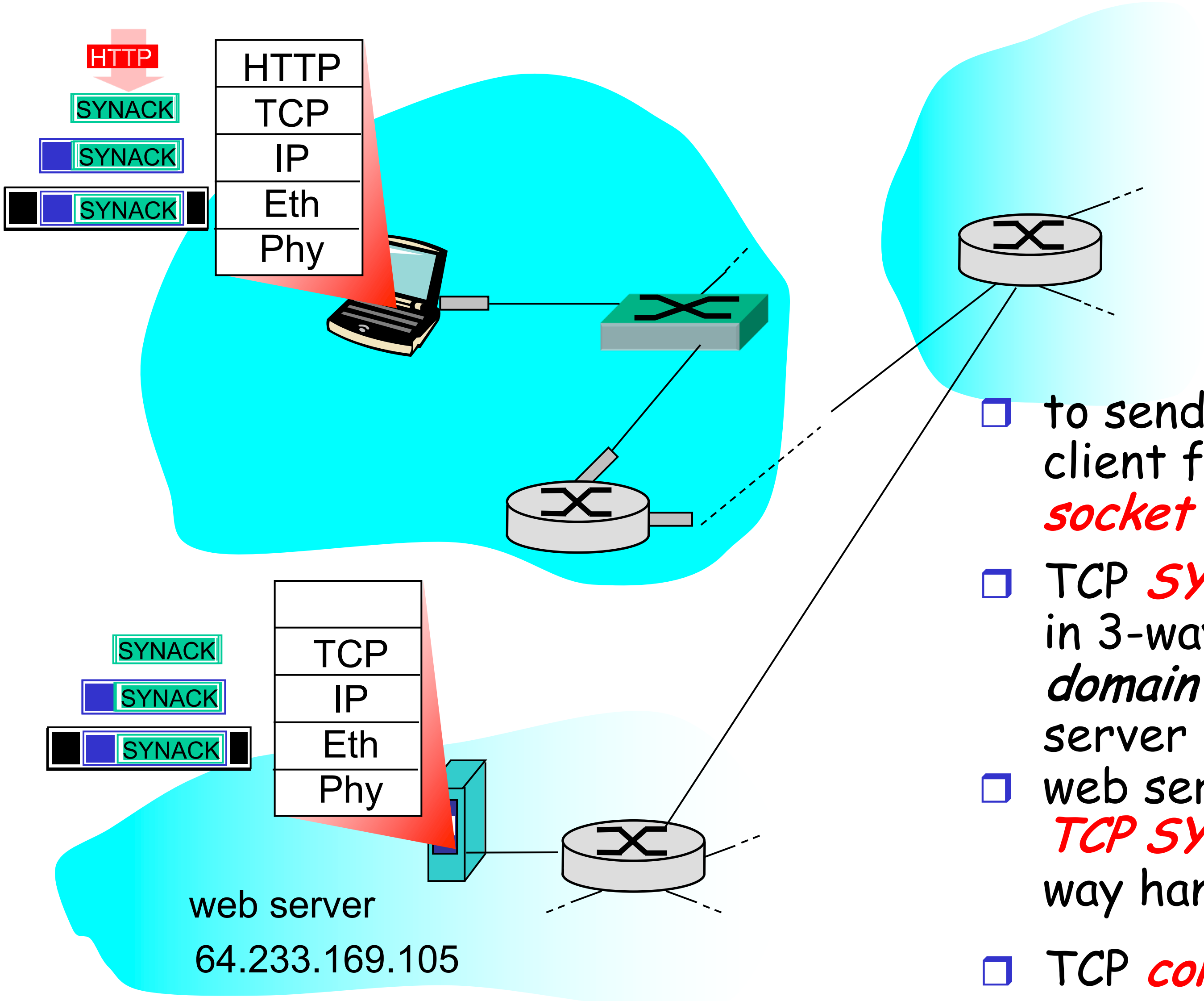
A day in the life… using DNS



- 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, 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

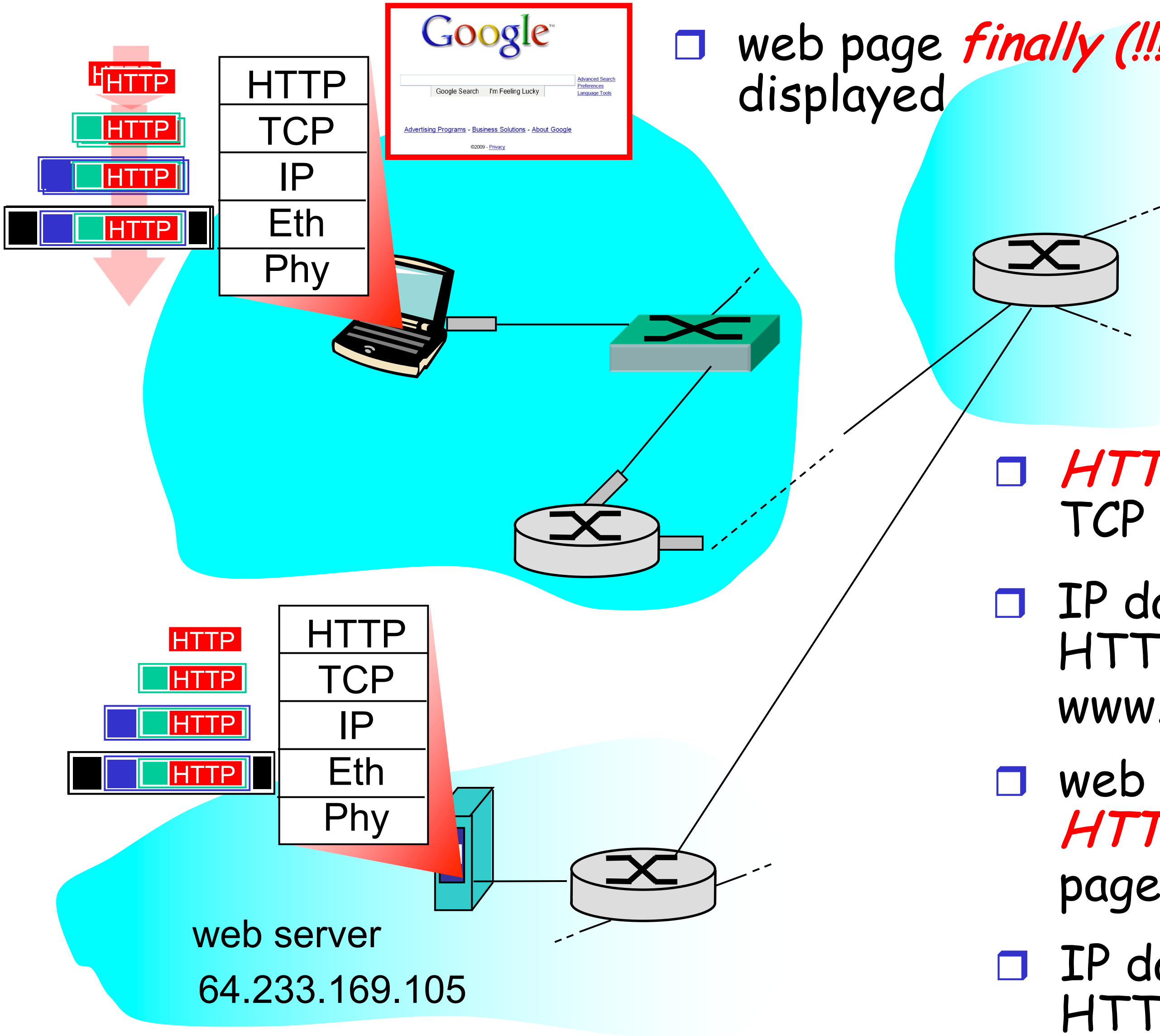
A day in the life… TCP connection carrying HTTP



- 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!*

5: DataLink Layer 5-88

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

5: DataLink Layer 5a-89

Chapter 5 outline

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 LAN addresses and ARP
- 5.5 Ethernet
- 5.6 Hubs, bridges, and switches
- 5.7 Wireless links and LANs
- 5.8 PPP
- 5.9 ATM
- 5.10 Frame Relay