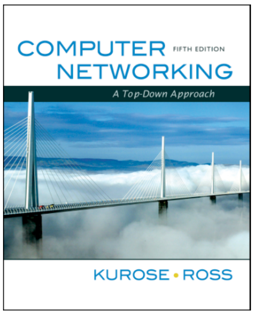
Introduction to Computer Networking

Guy Leduc

Chapter 5
Link Layer and LANs



Computer Networking: A Top Down Approach,
5^{th} edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Chapter 5: The Data Link Layer

Our goals:

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

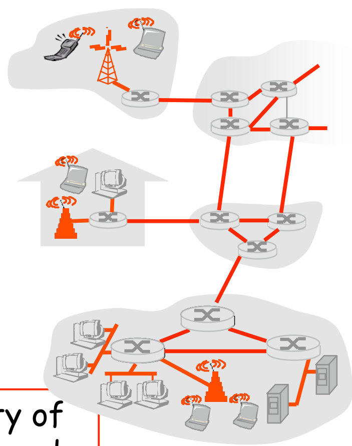
Link Layer

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

Link Layer: Introduction

Some terminology:

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



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

Link layer: context

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

transportation analogy

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

Link Layer Services

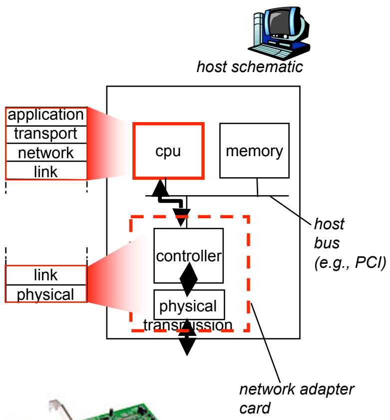
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already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

Link Layer Services (more)

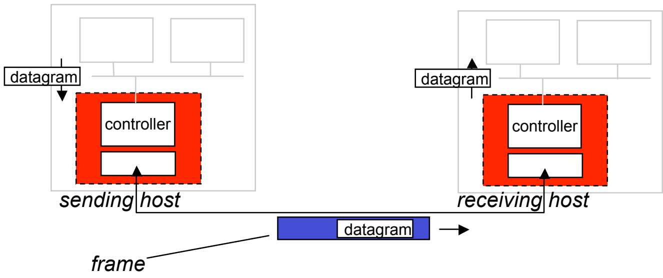
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - drops frame
    - if recovery needed: signals sender for retransmission
- error correction:
  - receiver identifies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

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - Ethernet card, 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

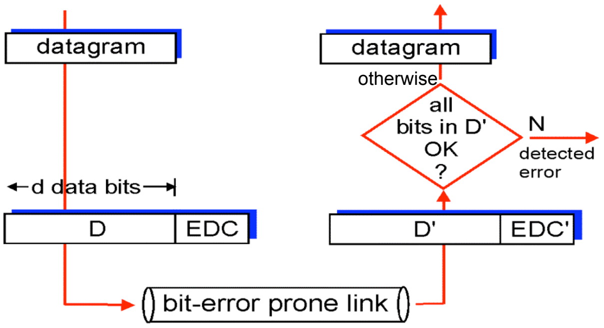
Link Layer

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

Error Detection

EDC= Error Detection (and sometimes Correction) bits (redundancy)
D = Data protected by error checking, may include header fields

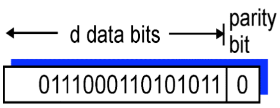
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection (and correction)



Parity Checking

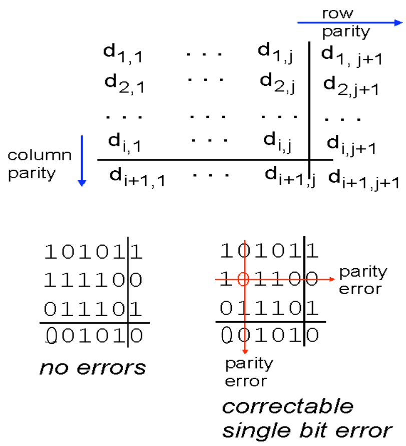
Single Bit Parity:

Detect single bit errors



Two Dimensional Bit Parity:

Detect *and correct* single bit errors



Internet checksum (review)

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

Sender:

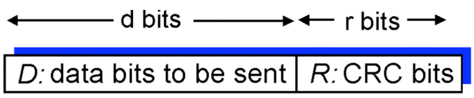
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

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?*

Checksumming: 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 (in base-2 arithmetic)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than or equal to r bits (see later)
- widely used in practice (Ethernet, 802.11 WiFi, PPP, ATM)



bit pattern

$D * 2^r$ XOR R

mathematical formula

CRC Example

Want:

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

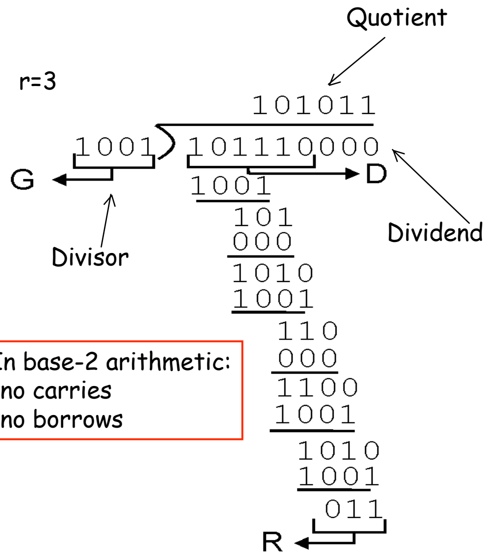
equivalently:

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

equivalently:

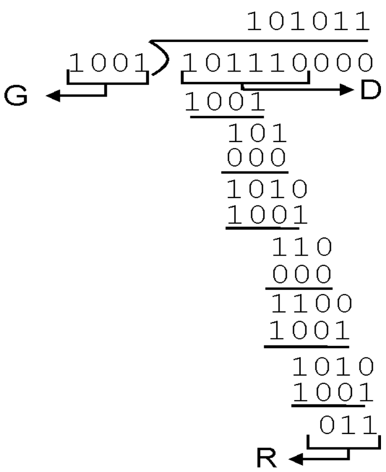
if we divide $D \cdot 2^r$ by G (in base-2 arithmetic), want remainder R

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



In base-2 arithmetic:
-no carries
-no borrows

CRC Example: the polynomial view



$D(x) = x^5 + x^3 + x^2 + x$

r=3

$G(x) = x^3 + 1$

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

Transmitted frame:

$T(x) = D(x) \cdot x^r - R(x)$

Is divisible by:

$G(x)$

Why does it work?

- We have $T(x) = D(x) \cdot x^r - R(x)$
  - Therefore, the remainder of $T(x) / G(x)$ is zero!
  - This is easy to check at the receiver, provided that the sender and the receiver agree on a certain $G(x)$
- Suppose some errors occur during transmission.
  - The received frame is $T(x) + E(x)$
  - The receiver will then calculate the remainder of $(T(x) + E(x)) / G(x)$
  - This remainder is equal to the remainder of $E(x) / G(x)$
  - If there is an error such that $E(x)$ is not divisible by $G(x)$, it is detected!
- The choice of $G(x)$ is thus very important!

Example of CRC

- Generator = $G(x) = x^{16} + x^{12} + x^5 + 1$
- $G(x)$ detects every $E(x)$ consisting of an odd number of error bits
  - Because $G(x)$ is a factor of $x + 1$ (in base-2 arithmetic, $G(1) = 0$), and a polynomial $E(x)$ with an odd number of terms is not divisible by $x+1$ (in base-2 arithmetic, $E(1) = 1$ for such $E(x)$)
- More generally, a $G(x)$ composed of an even number of terms detects every error consisting of an odd number of error bits
  - At least as good as a parity bit!
  - Could a parity bit be seen as a trivial CRC? Any $G(x)$ to suggest?
- $G(x)$ detects every 2-bit error (in any place in the frame)
- $G(x)$ detects every error burst of length ≤ 16 bits
  - An error burst of length n (≥ 2) is 1 erroneous bit, followed by n-2 bits (correct or not), followed by an erroneous bit
  - So an error burst of length ≤ 16 bits is an $E(x) = E'(x) \cdot x^k$, with neither $E'(x)$, nor x^k being divisible by $G(x)$, because $E'(x)$ is of degree ≤ 15, while $G(x)$ is of degree 16.
- More generally, a $G(x)$ of degree r detects every r-bit error bursts
- $G(x)$ detects 99,997% of the 17-bit error bursts (if errors are random)
- $G(x)$ detects 99,998% of the 18-bit error bursts (if errors are random)

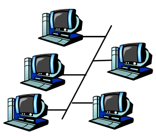
Link Layer

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

Multiple Access Links and Protocols

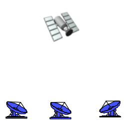
Two types of "links":

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



shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple Access protocols

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

multiple access protocol

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

Ideal Multiple Access Protocol

Broadcast channel of rate R bps

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

MAC Protocols: a taxonomy

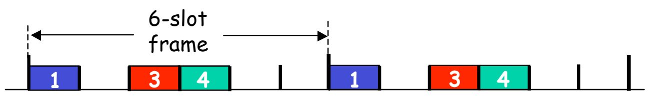
Three broad classes:

- 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

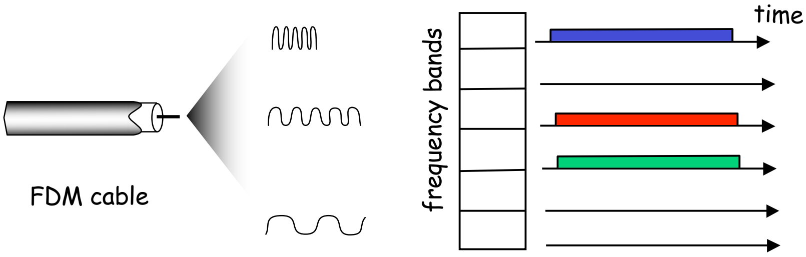
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

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



Random Access Protocols

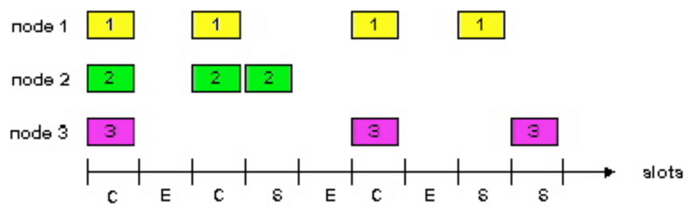
- When node has packet to send
  - transmit at full channel data rate R
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)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

Slotted ALOHA

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

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

Slotted ALOHA



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

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

Slotted Aloha efficiency

Efficiency is the long-run fraction of successful slots when there are many nodes, each with many frames to send

- Suppose N nodes with many frames to send, each transmits in slot with probability *p*
  - *Note: not exactly slotted ALOHA!*
- prob that node 1 has success in a slot = $p(1-p)^{N-1}$
- prob that any node has a success = $Np(1-p)^{N-1}$

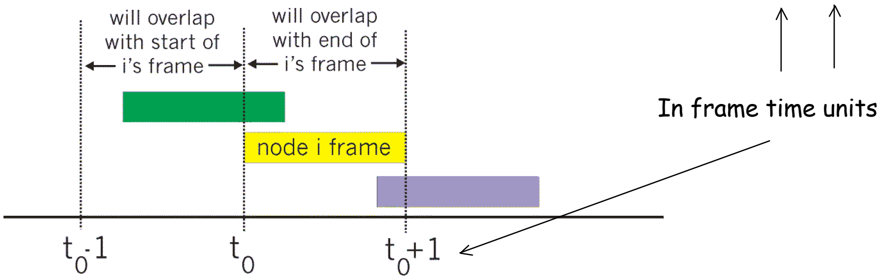
- For max efficiency with N nodes, find p* that maximizes $Np(1-p)^{N-1}$
  - $p^* = 1/N$
- For many nodes, take limit of $Np^*(1-p^*)^{N-1} = (1-1/N)^{N-1}$ as N goes to infinity, it gives $1/e = 0.37$

$$\lim_{N\to\infty}\left(1-\frac{G}{N}\right)^N = e^{-G}$$

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

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



Pure Aloha efficiency

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

P (success by <u>any</u> node) = $Np \cdot (1-p)^{2(N-1)}$

… choosing optimum p and then letting n go to infinity ...

= 1/(2e) = 0.18

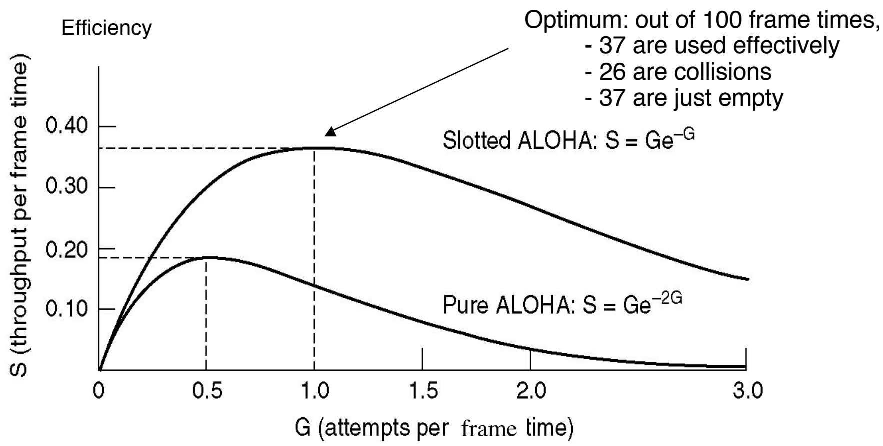
Even *worse* than slotted Aloha!

Efficiency wrt average traffic load

- Let $G = pN$ be the average aggregated traffic load (or demand) per frame time
  - = nr. of transmission attempts per frame time
  - N stations sending one frame with probability p in every frame time
- Efficiency
  - Slotted ALOHA: $Np(1-p)^{N-1} = G \cdot (1-G/N)^{N-1}$
  - ALOHA: $Np(1-p)^{2(N-1)} = G \cdot (1-G/N)^{2(N-1)}$
- Efficiency (for a given G) when N >>
  - Slotted ALOHA: $G \cdot e^{-G}$
  - ALOHA: $G \cdot e^{-2G}$
  - If $G \ll 1$: efficiency ≈ G, perfect

$$\lim_{N\to\infty}\left(1-\frac{G}{N}\right)^N = e^{-G}$$

ALOHA versus Slotted ALOHA



CSMA (Carrier Sense Multiple Access)

CSMA: listen before transmit:

If channel sensed idle: transmit entire frame

- If channel sensed busy, defer transmission
  - LBT: Listen Before Talking (and deference)

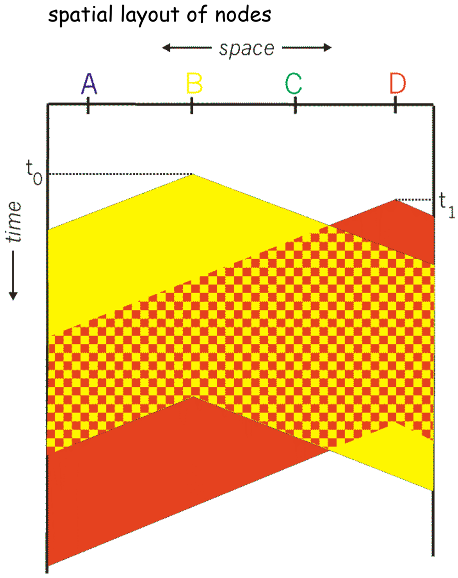
- human analogy: don't interrupt others!

CSMA collisions

collisions *can* still occur:
propagation delay means two nodes may not hear each other's transmission

collision:
entire packet transmission time wasted

note:
role of distance & propagation delay in determining collision probability



p-persistent CSMA

persistent CSMA
(if channel is busy, listen until it is freed)

non persistent CSMA
(if channel is busy, program a new attempt later)

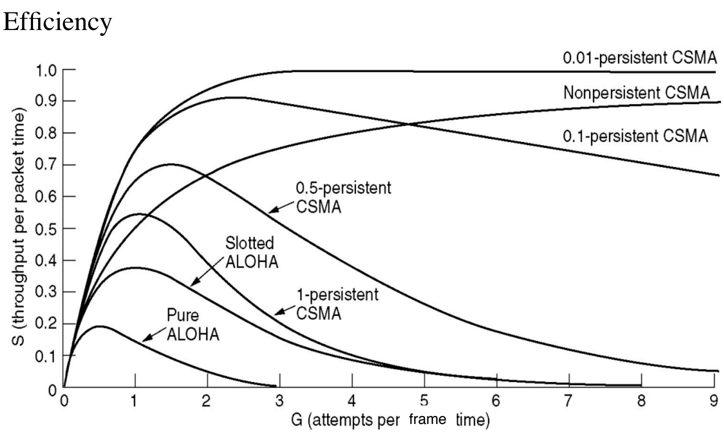
p-persistent CSMA:

While true do
 if channel is free
 then {
  with probability p: immediate transmission; or
  with probability 1-p: stay idle during at least propagation time (τ) }
 else listen until the channel is freed.

Trade-off between efficiency and delay

- This introduces a useless delay at low loads
- But the efficiency of the channel is better at high loads

Efficiency versus load for various random access protocols



From Computer Networks, by Tanenbaum © Prentice Hall

Engineering a CSMA network

B = Channel Data Rate (so-called Bandwidth)	(bps)
F = (Maximum) Frame size	(bits)
L = Length of the channel	(m)
c = Propagation speed	(m/s)

τ = Propagation delay = L / c	(s)
T = Transmission delay = F / B	(s)

a = τ / T = BL / cF

Let a = 1% → F = 100 BL / c = ± 5 10^{-7} BL (with c = ± 200,000 km/s)

Let B = 10 Mbps, L = 2.5 km → F = 12,500 bits (= 1,562.5 bytes)

This is roughly the Ethernet frame size

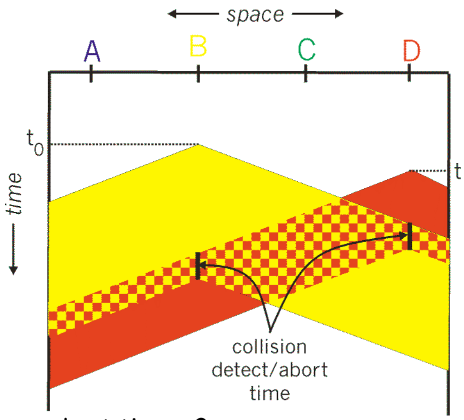
CSMA/CD (Collision Detection)

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

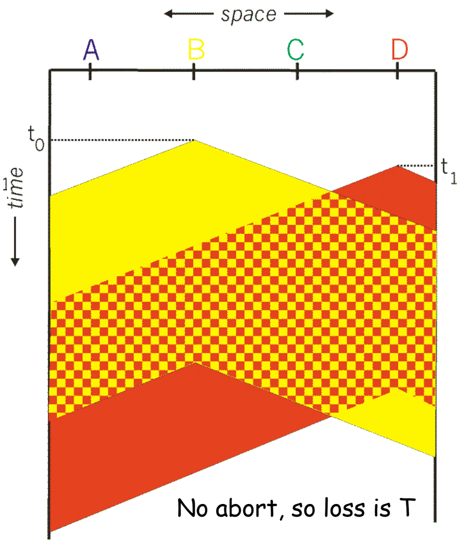
CSMA/CD collision detection

CSMA/CD

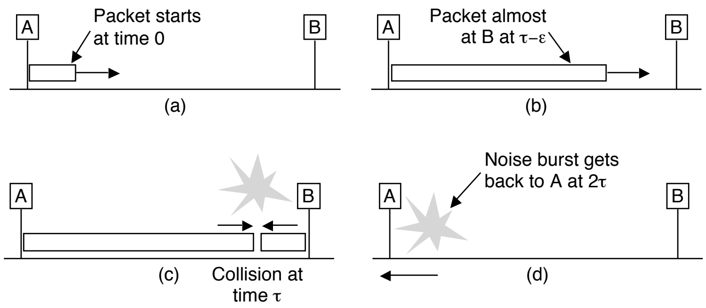


Lost time $\leq 2\tau$
(worst case: 2 max propagation times when the 2 stations are at the 2 ends)
+ detect/abort time

CSMA



Minimal frame size with CSMA/CD



- To detect collision, the sender must still be transmitting when the collision propagates back to it. So the condition is:
  $T > 2\tau$, which means $F/B > 2\,L/c$,
  which leads to a minimal $F_{min} = 2BL/c = \pm\, BL\, 10^{-8}$ bits
- Let $B = 10$ Mbps, $L = 2.5$ km $\rightarrow$ $F_{min} = \pm\, 250$ bits (= ± 32 bytes)
  - Ethernet has chosen 64 bytes = 512 bits (extra margin due to other delays)

"Taking Turns" MAC protocols

channel partitioning MAC protocols:
- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

Random access MAC protocols
- efficient at low load: single node can fully utilize channel
- high load: collision overhead

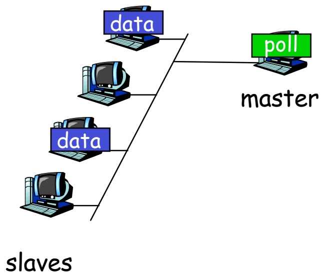
"taking turns" protocols

look for best of both worlds!

"Taking Turns" MAC protocols

Polling:

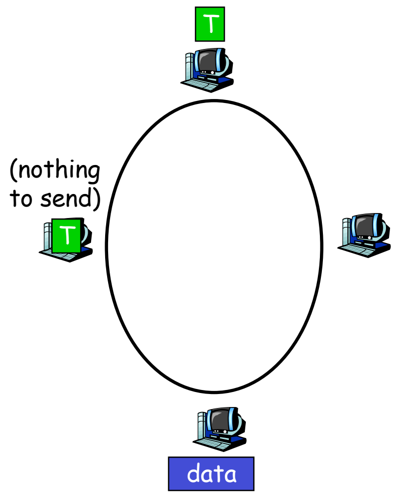
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking Turns" MAC protocols

Token passing:

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



Summary of MAC protocols

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

Link Layer

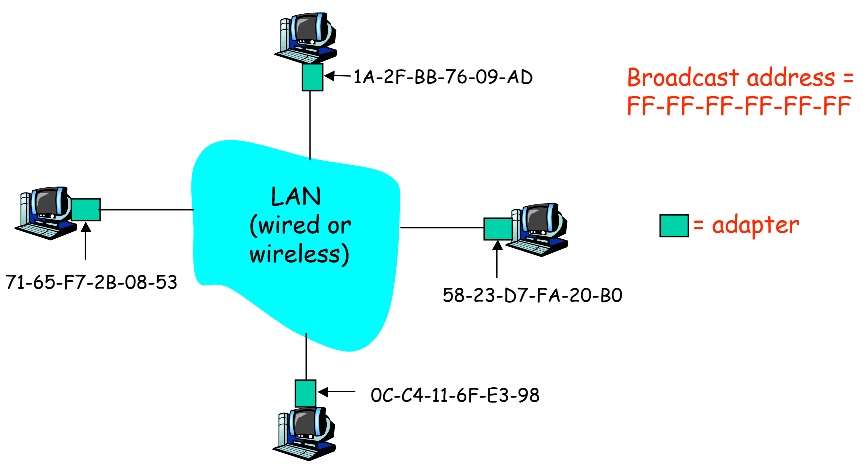
- 5.1 Introduction and services
- 5.2 Error detection
- 5.3Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link Virtualization
- 5.9 A day in the life of a web request

MAC Addresses and ARP

- 32-bit IP address:
  - *network-layer* address
  - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48-bit MAC address (for most LANs)
    - burned in NIC ROM, also sometimes software settable

LAN Addresses and ARP

Each adapter on LAN has unique LAN address

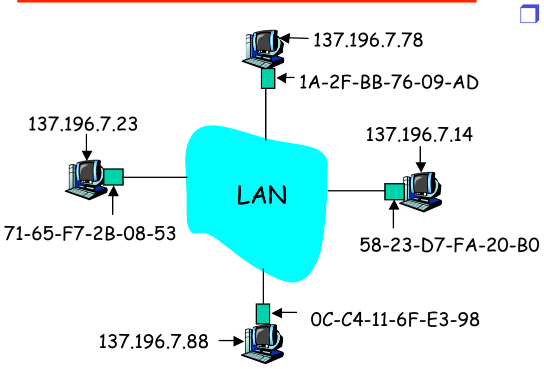


LAN Address (more)

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

ARP: Address Resolution Protocol

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



- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC address mappings for some LAN nodes

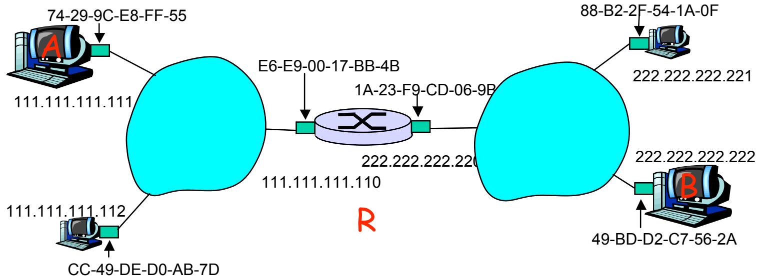
  < IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) B's 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
- All other stations had also cached A's IP-to-MAC pair!
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

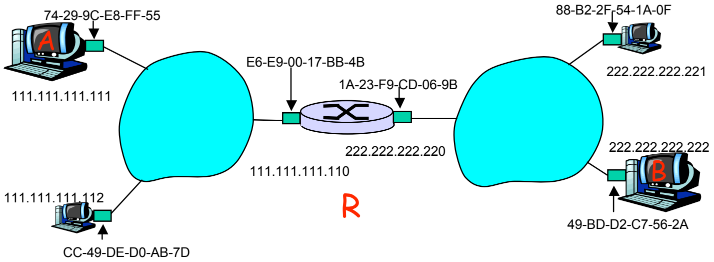
walkthrough: send datagram from A to B via R
assume A knows B's IP address



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

- A creates IP datagram with source A, destination B
- In routing table at source Host, find R's IP = 111.111.111.110
- 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 it's destined for 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!



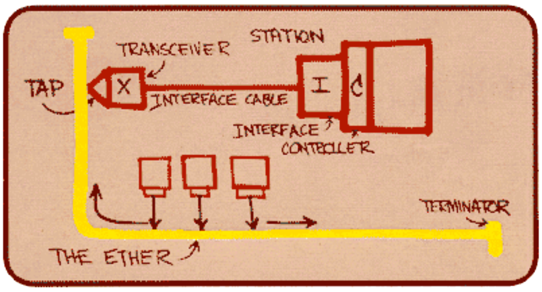
Link Layer

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

Ethernet

"dominant" wired LAN technology:

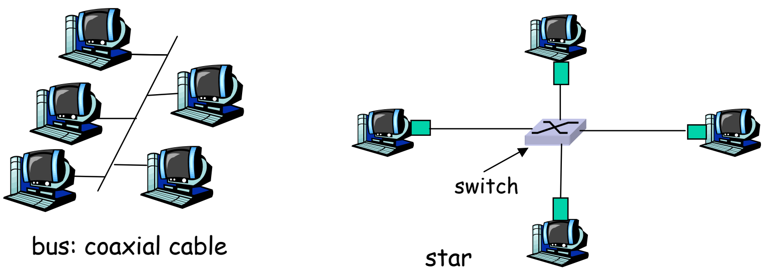
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe's Ethernet sketch

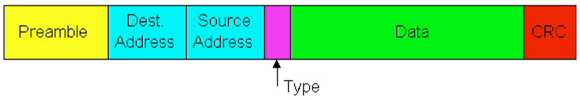
Star topology

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



Ethernet Frame Structure

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

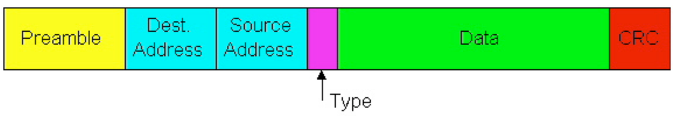


Preamble:

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

Ethernet Frame Structure (more)

- Addresses: 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- Type: indicates higher layer protocol (mostly IP but others possible, e.g., 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 nor nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting (collision), 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

Ethernet's CSMA/CD (more)

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

Bit time: 0.1 μsec for 10 Mbps Ethernet;
for K=1023, wait time is about 50 msec

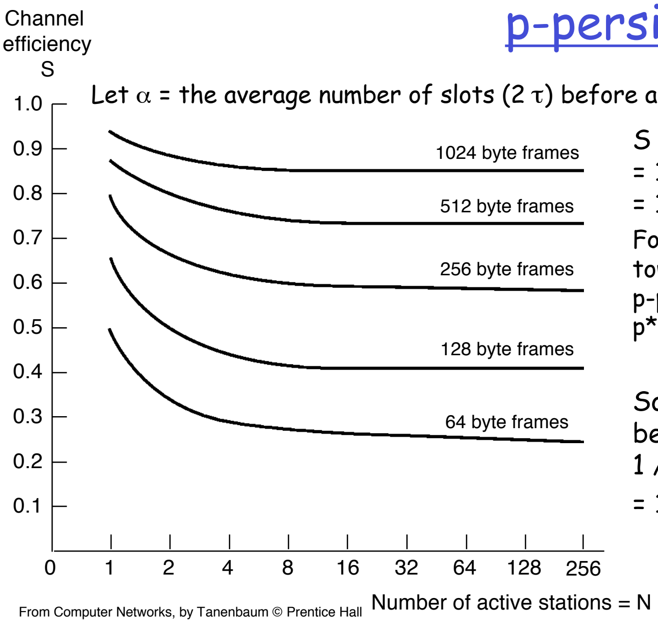
See/interact with Java applet on AWL Web site: highly recommended!

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}
- with this backoff algorithm, Ethernet is sort of p-persistent CSMA with an adaptive p!

Channel efficiency of an adaptive p-persistent CSMA

Let α = the average number of slots (2τ) before any successful transmission



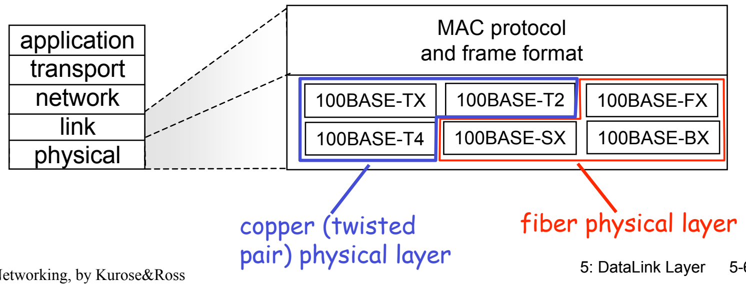
$S = T / (T + \alpha \times 2\tau)$
$= 1 / (1 + \alpha \times 2\tau/T)$
$= 1 / (1 + \alpha \times 2BL/cF)$

For large N, α converges towards e (±2.7) for a p-persistent CSMA with p* = 1/N (ideal case)

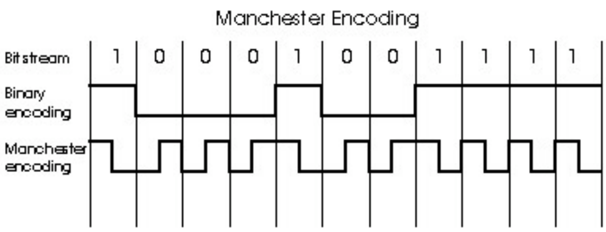
So, for N >>, S cannot be better than
$1 / (1 + e \times 2\tau/T)$
$= 1 / (1 + 5.4\ \tau/T)$

802.3 Ethernet Standards: Link & Physical Layers

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

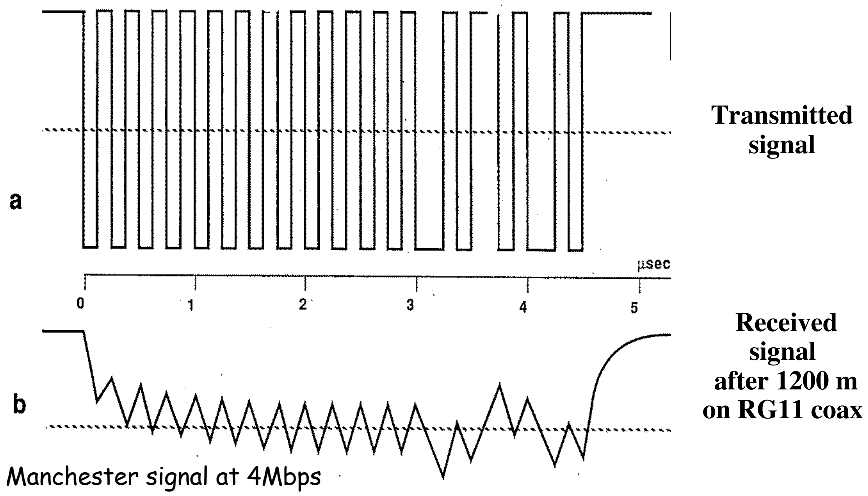


Manchester encoding



- 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!

Manchester encoding - Signal distortion



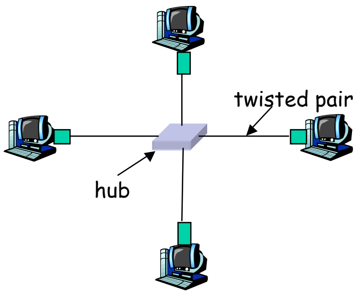
Manchester signal at 4Mbps

Link Layer

Hubs

… physical-layer ("dumb") repeaters:

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

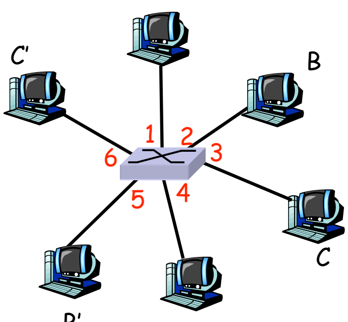


Switch

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

Switch: allows *multiple* simultaneous transmissions

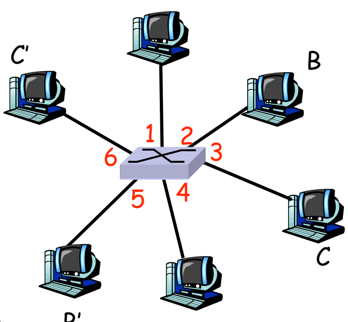
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



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

Switch Table

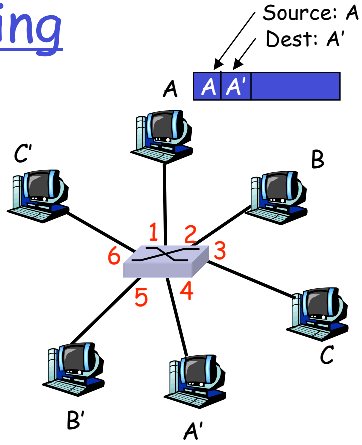
- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- Is a MAC forwarding table!
  - Looks like an IP forwarding table
- Q: how are entries created, maintained,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



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

Switch: self-learning

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



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Switch: frame filtering/forwarding

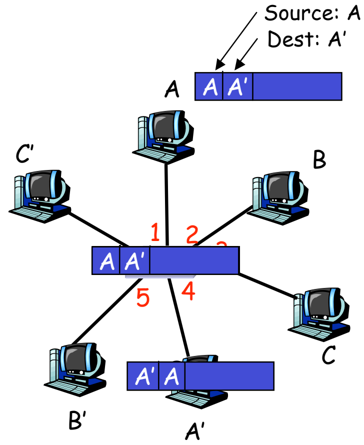
When frame received:

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

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

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

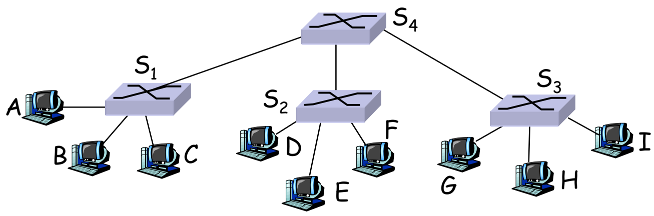


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting switches

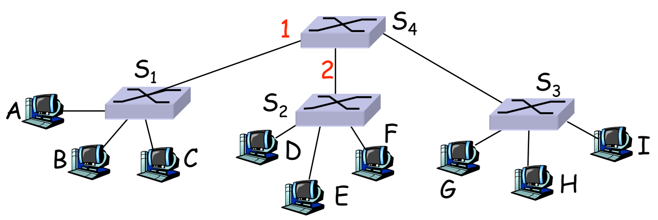
- switches can be connected together



- *Q:* sending from A to F - how does S_1 know how to forward frame destined for F via S_4 and S_2?
- *A:* self learning! (works exactly the same as in single-switch case!)

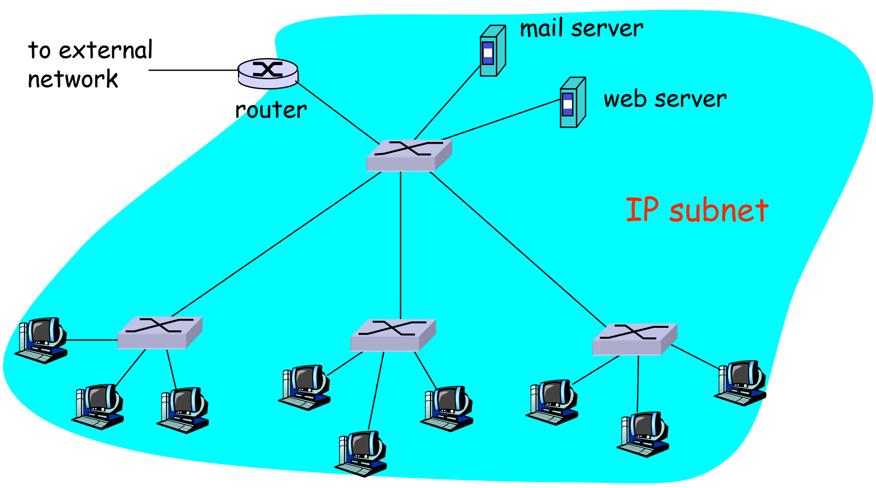
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C

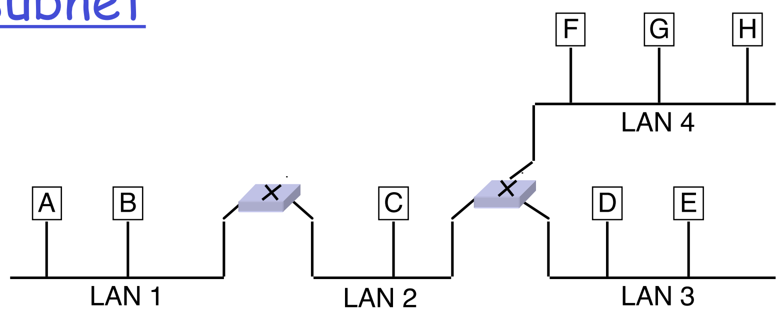


- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network



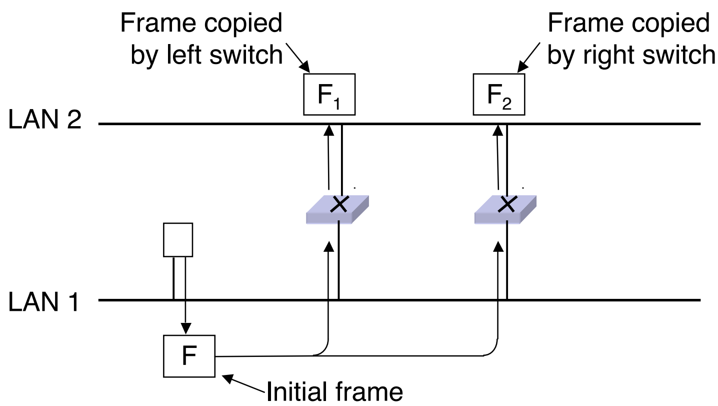
More than one switch in a subnet



- All principles seen so far are applicable to a subnet with several switches, if there is no cycle in the topology
- No cycle = no redundancy in case of failure
  - Part of subnet can be disconnected

From Computer Networks, by Tanenbaum © Prentice Hall

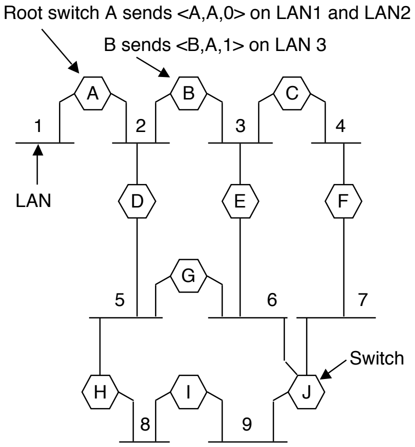
Problem with cycles



- Problem: Frame F (whose destination address is unknown to both switches) will loop.
- Solution: build a logical spanning tree topology over the real topology

From Computer Networks, by Tanenbaum © Prentice Hall

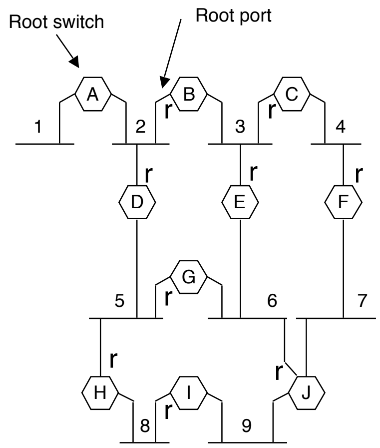
Spanning tree (1)



- Build a logical tree reaching all LANs
- 1. Determine the root switch (smallest switch id)
  - Switches regularly flood control messages (BPDUs) on all their output ports:
  - BPDU = <Source switch id, root as assumed, distance to root>
  - All switches will soon discover the root id

From Computer Networks, by Tanenbaum © Prentice Hall

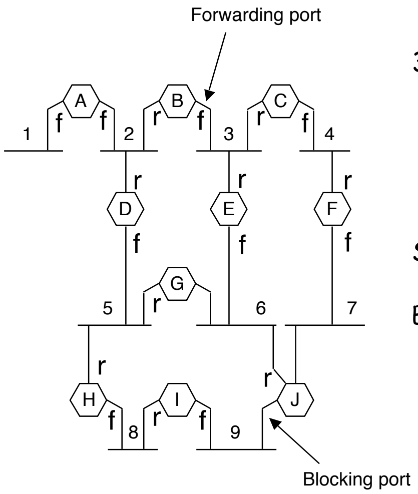
Spanning tree (2)



- 2. Build the tree
  - By continuously receiving these BPDUs (possibly on several ports), a switch knows its distance to the root and which port leads to the root by that shortest distance (root port, r)

From Computer Networks, by Tanenbaum © Prentice Hall

Spanning tree (3)



3. Decide if non-root ports are (data) forwarding or (data) blocking

   A port is "forwarding (f)" on a given LAN if the BPDUs this switch sends on this LAN are "smaller" than those other switches (would) send.

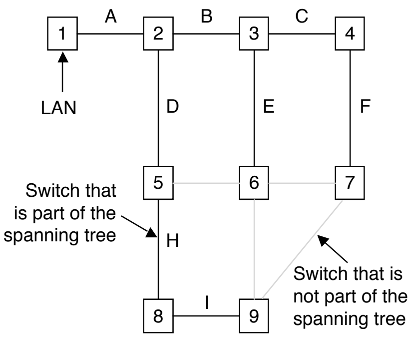
Smaller = shorter distance, or equal distance and smaller switch id

Example:

on LAN6, E sends <E,A,2>, G would send <G,A,2>, and J would send <J,A,3>

So E is elected to be the only one to forward frames on LAN6: J is too far away from the root, and G > E

The resulting spanning tree



Routing on a spanning tree is not optimal!

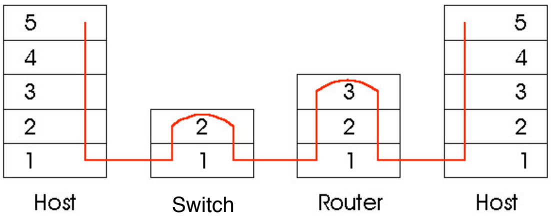
Same spanning tree for all source-destination pair!

Compare to layer-3 routing

- Some switches (e.g., G and J) are not part of the tree
  - Another case could be that some ports of some switches are blocking
- They could become part of the tree if another switch or port would fail (leading to no refresh of BPDUs)
- So, switches have to listen to BPDUs on blocking ports to detect failures

Switches vs. Routers

- both store-and-forward devices
  - routers: network layer devices (examine network layer headers)
  - switches are link layer devices
- routers maintain IP routing and IP forwarding tables, implement routing algorithms
- switches maintain MAC forwarding tables, implement filtering, learning and spanning tree algorithms



Summary comparison

	hubs	routers	switches
traffic isolation	no	yes	yes
plug & play	yes	no	yes
optimal routing	no	yes	no
cut through	yes	no	yes *Some do*

Link Layer

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

Point to Point Data Link Control

- one sender, one receiver, one link: easier than broadcast link:
  - no Media Access Control
  - no need for explicit MAC addressing
  - e.g., dialup link, ISDN line, ADSL line
- 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
  - carry network layer data of any network layer protocol (not just IP) *at same time*
  - ability to demultiplex upwards
- bit transparency: must carry any bit pattern in the data field
- 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

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 (framing)
- Address: does nothing (only one option)
- Control: does nothing; in the future possible multiple control fields
- Protocol: upper layer protocol to which frame delivered (e.g., PPP-LCP, IP, IPCP, etc)

1	1	1	1 or 2	variable length	2 or 4	1
01111110	11111111	00000011	protocol	info	check	01111110
flag	address	control				flag

PPP Data Frame

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

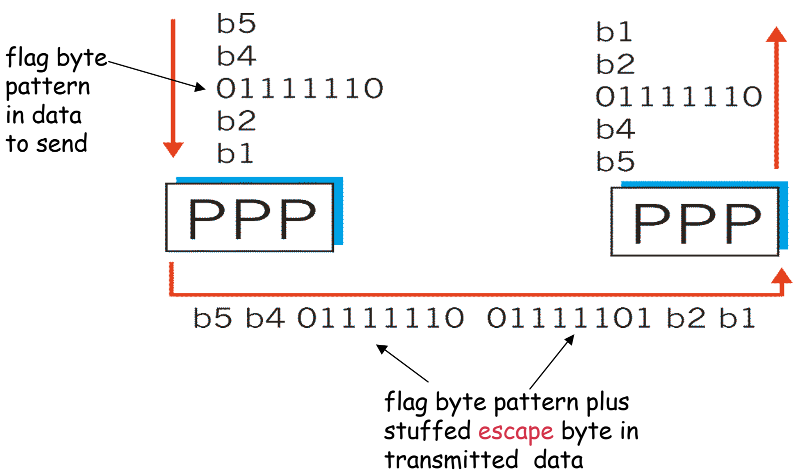
1	1	1	1 or 2	variable length	2 or 4	1
01111110	11111111	00000011	protocol	info	check	01111110
flag	address	control				flag

Byte Stuffing - Principle

- "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 - Variant of PPP



Bit stuffing

Used in other data link protocols
Use same flag as delimiter = 01111110

The original data (the frame payload)

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

Bit stuffing: the data as they appear on the line

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

Stuffed bits

Bit stuffing rule for that flag: add a 0 after five 1's, always!
So no flag pattern in the payload

The data as they are stored in the receiver's memory after destuffing

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

Rule: remove 0 after five 1's. If six 1's in a row, check if it is a flag.

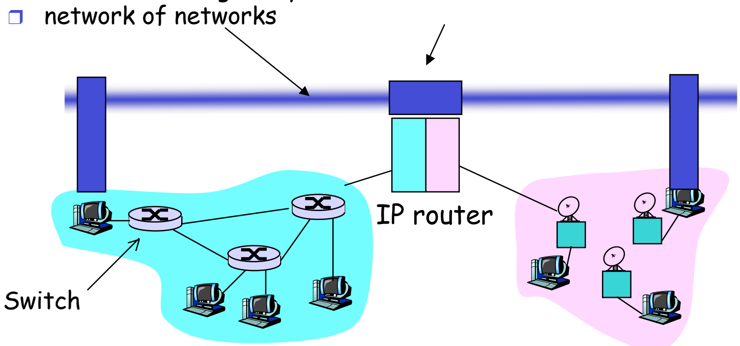
Link Layer

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

The Internet: virtualizing networks

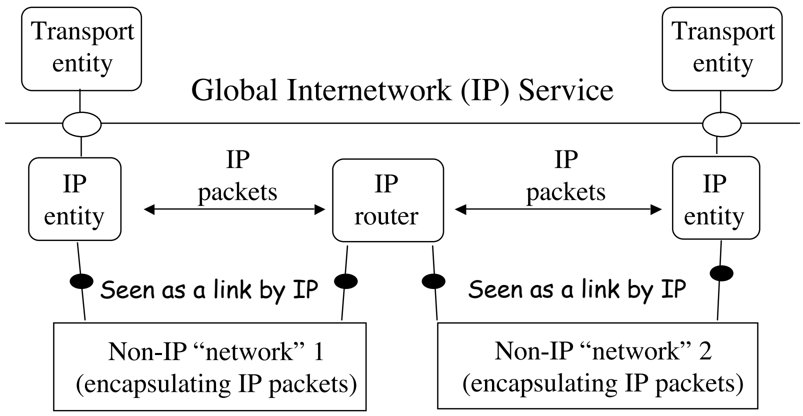
Internetwork layer (IP):
- addressing: internetwork appears as a single, uniform entity, despite underlying local network heterogeneity
- network of networks

Router:
- "embed IP packets in local packet format or extract them"
- route (at IP level) to next IP router



Those networks are seen as virtual links by IP routers

IP over anything (not just links or isolated LANs)



Examples:
-Network 1 can be a subnet composed of one or several Ethernet switches
-Network 2 can be a large ATM or MPLS backbone network

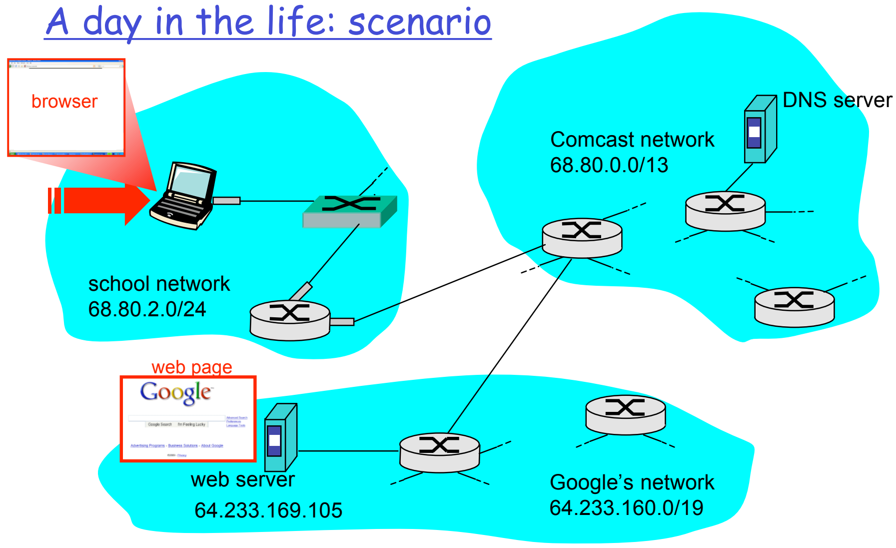
Link Layer

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

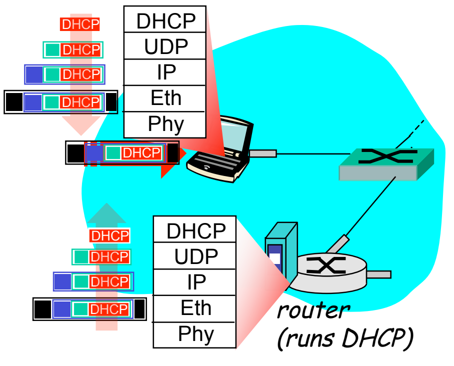
Synthesis: a day in the life of a web request

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

A day in the life: scenario

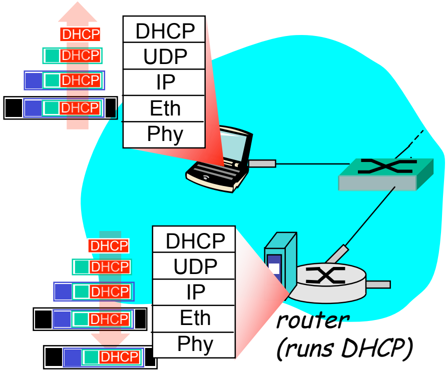


A day in the life… connecting to the Internet



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

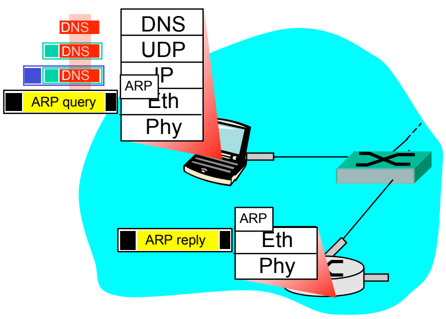
A day in the life… connecting to the Internet



- 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

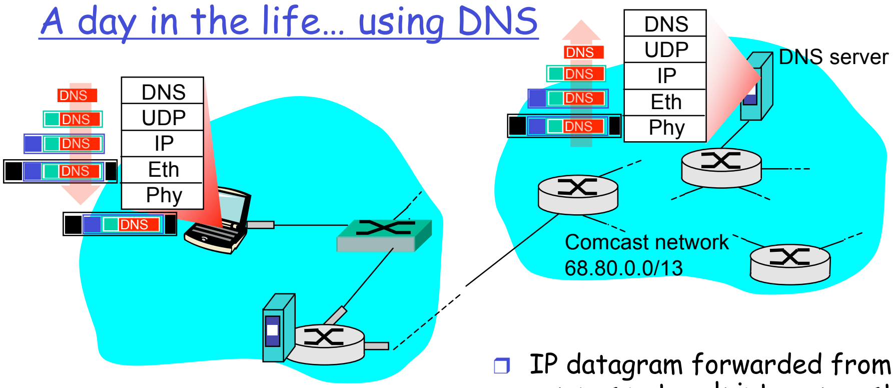
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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



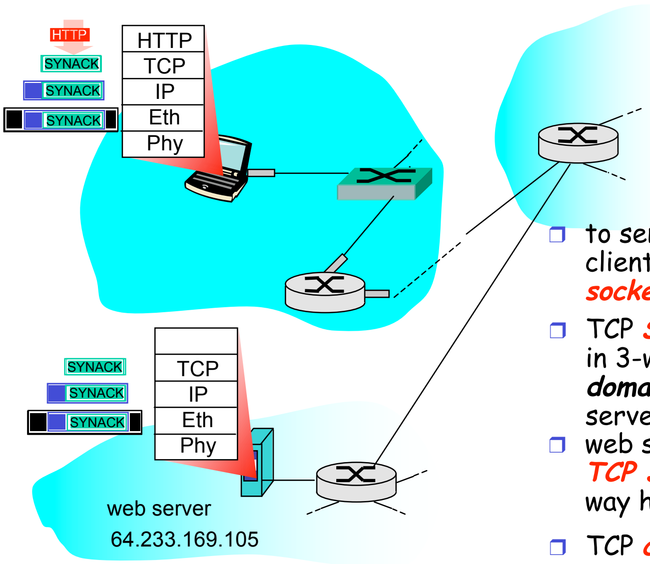
- before sending ***HTTP*** request, need IP address of www.google.com: ***DNS***
- DNS query created, encapsulated in UDP, encapsulated in IP, encasulated in Eth. In order to send frame to router, need MAC address of router interface: ***ARP***
- ***ARP query*** broadcast, received by router, which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



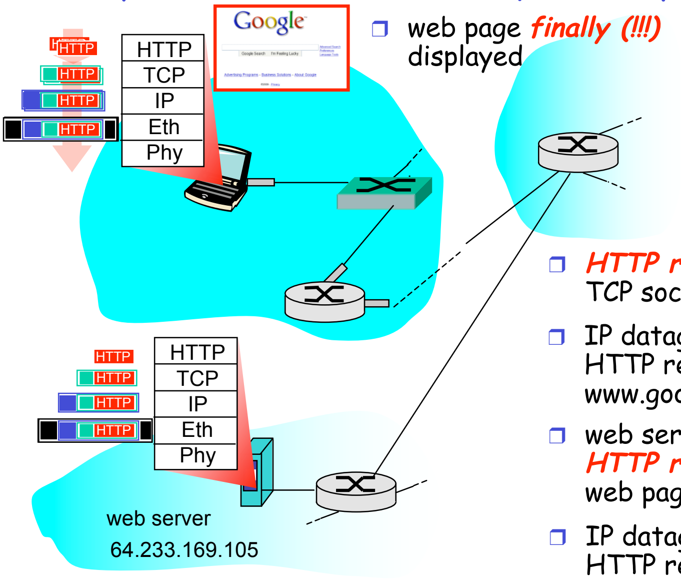
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by ***RIP, OSPF, IS-IS*** and/or ***BGP*** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies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

A day in the life… HTTP request/reply



- web page *finally (!!!)* displayed
- *HTTP request* sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with *HTTP reply* (containing web page)
- IP datagram containing HTTP reply routed back to client

Chapter 5: Summary

- principles behind data link layer services:
  - error detection
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANs
    - learning, building spanning tree
  - PPP
  - Link virtualization: networks as a link layer

Chapter 5: let's take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice
- ….. could stop here …. but *lots* of interesting topics!
  - Multicasting
  - Wireless networks
  - Mobility
  - Multimedia networking
  - Secure networking
  - Network management