Data Link Layer

CS 3516 – Computer Networks

Chapter 5: The Data Link Layer

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!* in *Ch3*)
- Instantiation and implementation of various link layer technologies

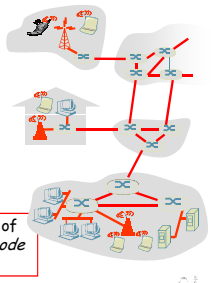
Link Layer

- 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

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 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 hop = communication link
- Transportation mode = link layer protocol
- Travel agent = routing algorithm

Link Layer Services

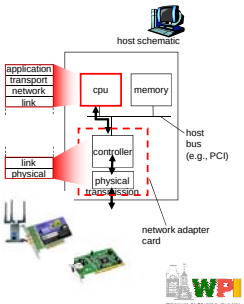
- Framing, link access
  - Encapsulate datagram into *frame*, adding header, trailer
  - Channel access if shared medium
  - Medium Access Control (MAC) addresses used in frame headers to identify source and dest
    - Different from IP address!
- Reliable delivery between adjacent nodes
  - We learned how to do this already! (in ch3)
  - Seldom used on low bit-error link (fiber, some twisted pair)
  - Used for *wireless* links with high error rates

Link Layer Services (more)

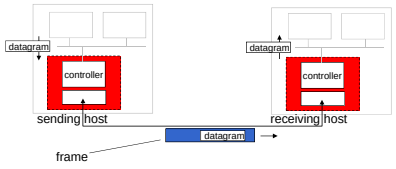
- Flow control
  - Pacing between adjacent sending and receiving nodes
- Error detection
  - Errors caused by signal attenuation, noise.
  - Receiver detects presence of errors
    - Signals sender for retransmission or drops frame
- Error correction
  - Receiver identifies *and corrects* bit error(s) without resorting to retransmission
- Half-duplex and full-duplex
  - With half duplex, nodes at both ends of link can transmit, but not at same time

Where is 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, and 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

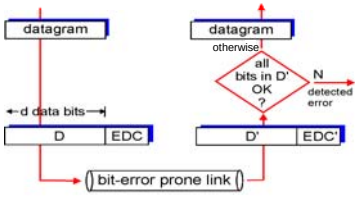
Data Link Layer

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

Error Detection

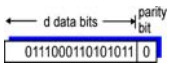
EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - Protocol may miss some errors, but rarely
  - Larger EDC field yields better detection and correction

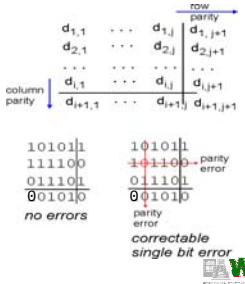


Simple - Parity Checking

Single Bit Parity:
Detect single bit errors



Two Dimensional Bit Parity:
Detect and correct single bit errors



Internet Checksum (review)

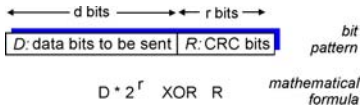
Goal: detect "errors" (e.g. flipped bits) in transmitted packet

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 (CRC)

- 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 <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)



CRC Example – Choosing R

Want:

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

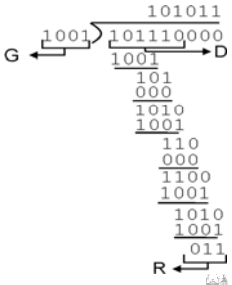
Equivalently:

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

Equivalently:

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

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



CRC Standards

- Defined for 8, 12, 16 and 32 bit genrators (G)
- CRC-32 adopted by many IEEE link-layer protocols uses generator:
  - G_{crc-32} = 100000100110000010001110110110111
- Detects *all* errors burst less than 33 bits
- Detects *all* odd number bit errors
- Burst errors greater than 33 bits with probability 1-0.5^r

Data Link Layer

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

Multiple Access Links and Protocols

Two types of "links":
- point-to-point (not shared)
  - 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

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 determines how nodes share channel (i.e. determine when/who 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 (no overhead)
3. Fully decentralized
   - No special node to coordinate transmissions
   - No synchronization of clocks, slots
4. Simple

MAC Protocols: a Taxonomy

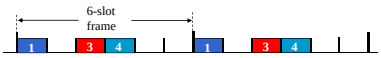
Three broad classes:

- Channel Partitioning
  - Divide channel into smaller "pieces" (time slots, frequency)
  - 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 perhaps take longer turns

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

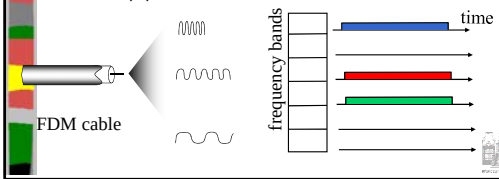
- Access to channel in "rounds"
- Each station gets fixed length slot (length = pkt trans time) in each round
- Unused slots go idle
- Example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
- Unused transmission time in frequency bands go idle
- Example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random Access Protocols

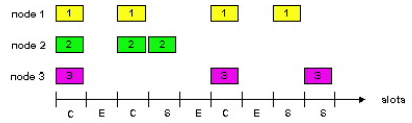
- When node has packet to send
  - Transmit at full channel data rate R
  - No *a priori* coordination among nodes
- Two or more transmitting nodes ➜ "collision"
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions (e.g. via delayed retransmissions)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

Slotted ALOHA

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

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

Slotted ALOHA



Pros
- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only slots in nodes need to be in sync
- Simple

Cons
- Collisions, wasting slots
- Idle slots
- Nodes may be able to detect collision in less than time to transmit packet
- Clock synchronization

Slotted Aloha Efficiency

Efficiency : long-run fraction of successful slots (many nodes, all with many frames to send)

- Suppose: *N* nodes with many frames to send, each transmits in slot with probability *p*
- Prob that given node has success in a slot = $p(1-p)^{N-1}$
- Prob that *any* node has a success = $Np(1-p)^{N-1}$

- Max efficiency: find p' that maximizes $Np(1-p)^{N-1}$
- For many nodes, take limit of $Np'(1-p')^{N-1}$ as N goes to infinity, gives:

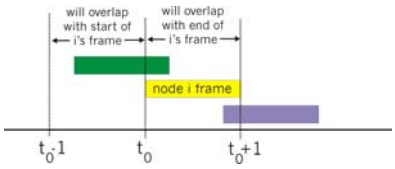
Max efficiency = 1/e ~ .37

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

!

Pure (Unslotted) ALOHA

- Unslotted Aloha: simpler, no synchronization
- When frame first arrives
  - Transmit immediately
- Collision probability increases:
  - Frame sent at t_0 collides with other frames sent in $[t_0-1,t_0+1]$



Pure Aloha Efficiency

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 $[p_0-1,p_0]$

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

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

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

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

Even worse than slotted Aloha!

CSMA (Carrier Sense Multiple Access)

CSMA: listen before transmit
- If channel sensed idle → transmit entire frame
- If channel sensed busy → defer transmission

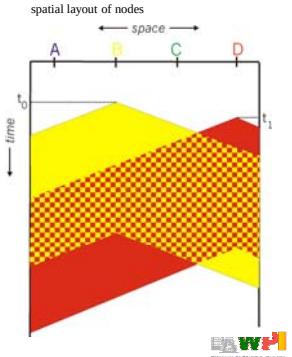
- Human analogy: someone else talking? → Don't interrupt!

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

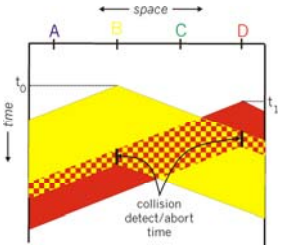


CSMA/CD (Collision Detection)

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

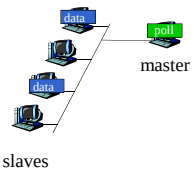
Channel partitioning MAC protocols
- Share channel *efficiently* and *fairly* at high load
- Inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

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!

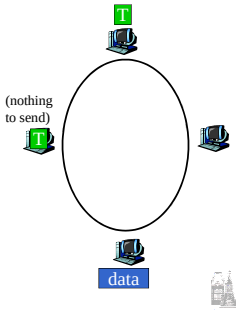
"Taking Turns" MAC protocols

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*
  - 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

Data Link Layer

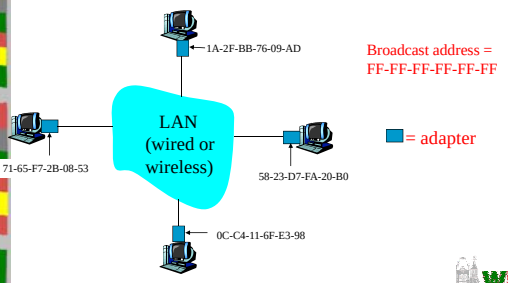
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

MAC Addresses

- 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

Each adapter on LAN has unique LAN address

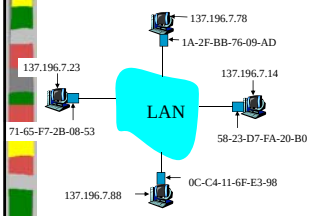


LAN Address (more)

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

ARP: Address Resolution Protocol

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



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

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

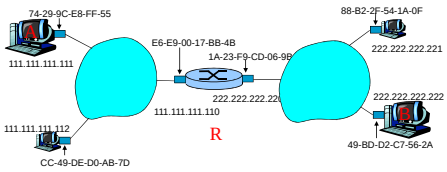
ARP Protocol: Same LAN

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: Routing to Another LAN

Walkthrough: send datagram from A to B via R

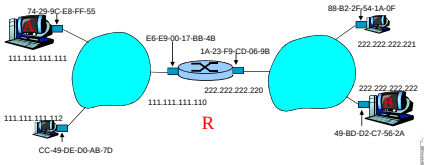
assume A knows B's IP address



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

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

This is a really important example – make sure you understand!



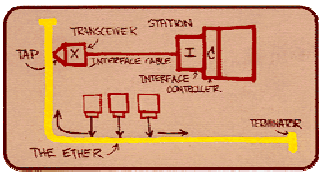
Data Link Layer

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

Ethernet

Dominant wired LAN technology:

- Cheap ($20) for NIC
- First widely used LAN technology
- Simpler, cheaper than token LANs and ATM
- Kept up with speed race: 10 Mbps – 10 Gbps

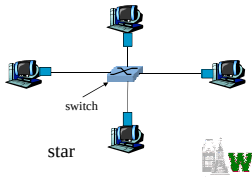


Metcalfe's Ethernet sketch

Topology (Bus and Star)

- 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 (contrast with *hub*)
  - Each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)

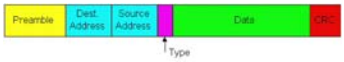
bus: coaxial cable



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

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 or nacks to sending NIC
  - Stream of datagrams passed to network layer can have gaps (missing datagrams)
  - Gaps will be filled if app is using TCP
  - Otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters **exponential backoff**: after *m*th collision, NIC chooses *K* at random from {0,1,2,…,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: .1 microsec 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}

CSMA/CD Efficiency

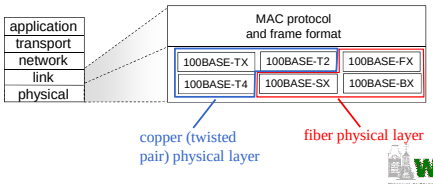
- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

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

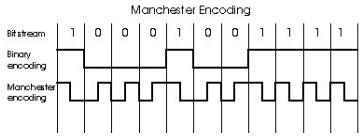
- Efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- Better performance than ALOHA: and simple, cheap, decentralized!

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



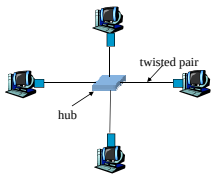
- Used in 10BaseT
- Each bit has a transition
- Allows clocks in sending and receiving nodes to synchronize to each other
  - No need for a centralized, global clock among nodes!
- Hey, this is physical-layer stuff!

Data Link Layer

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

Hubs

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

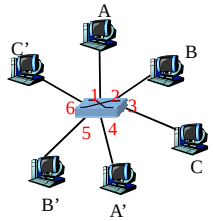


Switch

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

Switch: Allows *Multiple* Simultaneous Transmissions

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



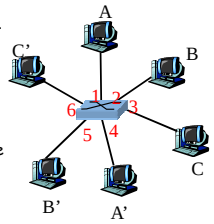
switch with six interfaces (1,2,3,4,5,6)

Switch Table

- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)

looks like a routing table!

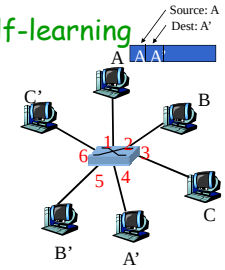
- Q: how are entries created, maintained in switch table?
  - something like a routing protocol?



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

Switch: Self-learning

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



MAC addr	interface	TTL
A	1	60

Switch table (initially empty)

Switch: Frame Filtering / Forwarding

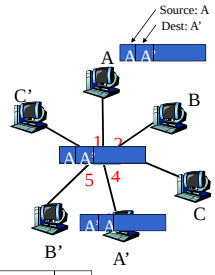
When frame received:

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

forward on all but the interface on which the frame arrived

Self-learning, Forwarding: example

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

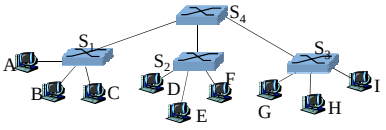


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting Switches

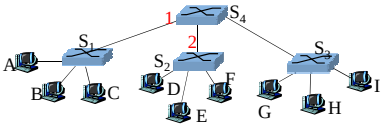
- Switches can be connected together



- Q: sending from A to G - how does 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!)

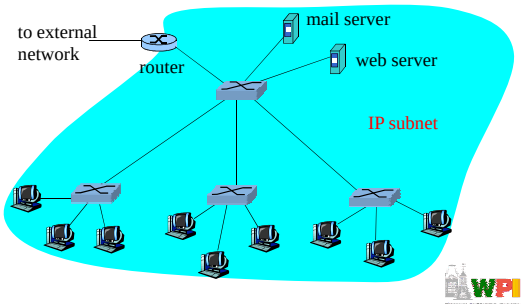
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



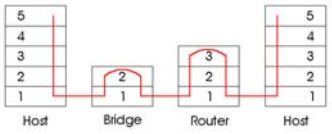
- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

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



Data Link Layer

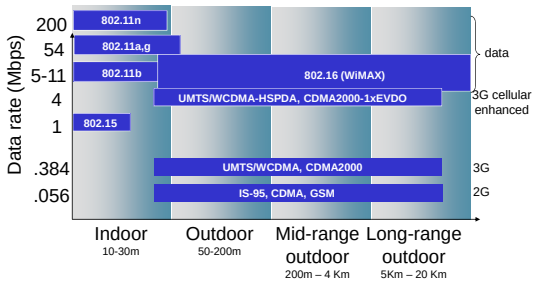
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

But First! Elements of Wireless (WiFi)

- Note some key characteristics of Wireless that differ from wired
- 802.11 (WiFi) as contrast to 802.3 (Ethernet)

(Bits of Ch 6.1 – 6.3)

Characteristics of Selected Wireless Link Standards



Wireless Link Characteristics (1)

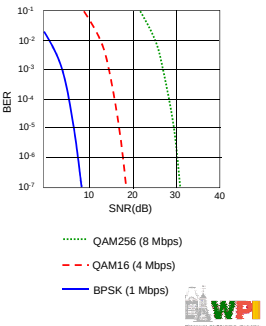
Differences from wired link ….

- decreased signal strength: radio signal attenuates as it propagates through matter (path loss)
- interference from other sources: standardized wireless network frequencies (e.g., 2.4 GHz) shared by other devices (e.g., phone); devices (motors) interfere as well
- multipath propagation: radio signal reflects off objects ground, arriving ad destination at slightly different times

…. make communication across (even a point to point) wireless link much more "difficult"

Wireless Link Characteristics (2)

- SNR: signal-to-noise ratio
  - larger SNR – easier to extract signal from noise (a "good thing")
- *SNR versus BER tradeoffs*
  - *Given physical layer:* increase power → increase SNR → decrease BER
  - *Given SNR:* choose physical layer that meets BER requirement, giving highest thruput
    - SNR may change with mobility: dynamically adapt physical layer (modulation technique, rate)

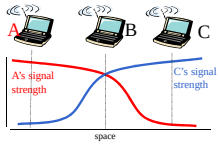


Wireless Network Characteristics

Multiple wireless senders and receivers create additional problems (beyond multiple access):

Hidden terminal problem
- B, A hear each other
- B, C hear each other
- A, C can not hear each other

means A, C unaware of their interference at B



Signal attenuation:
- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

IEEE 802.11 Wireless LAN

802.11b
- 2.4-5 GHz unlicensed spectrum
- up to 11 Mbps
- direct sequence spread spectrum (DSSS) in physical layer
  - all hosts use same chipping code

802.11a
- 5-6 GHz range
- up to 54 Mbps

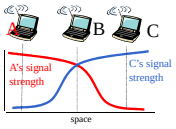
802.11g
- 2.4-5 GHz range
- up to 54 Mbps

802.11n: multiple antennae
- 2.4-5 GHz range
- up to 200 Mbps

- All use CSMA/CA for multiple access
- All have base-station and ad-hoc network versions

IEEE 802.11: multiple access

- Avoid collisions: 2+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - don't collide with ongoing transmission by other node
- 802.11: *no* collision detection!
  - difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - can't sense all collisions in any case: hidden terminal, fading
  - goal: *avoid collisions:* CSMA/C(ollision)A(voidance)



IEEE 802.11 MAC Protocol: CSMA/CA

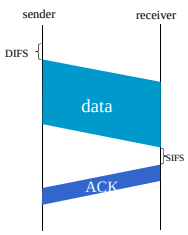
802.11 sender

1 if sense channel idle for **DIFS** then
  transmit entire frame (no CD)

2 if sense channel busy then
  start random backoff time
  timer counts down while channel idle
  transmit when timer expires
  if no ACK, increase random backoff interval, repeat 2

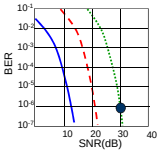
802.11 receiver

- if frame received OK
  return ACK after **SIFS** (ACK needed due to hidden terminal problem)



802.11: Advanced Capabilities

Rate Adaptation
- Base station, mobile dynamically change transmission rate (physical layer modulation technique) as mobile moves, SNR varies



1. SNR decreases, BER increase as node moves away from base station
2. When BER becomes too high, switch to lower transmission rate but with lower BER

More Wireless!

- Power management
- Other protocols: Zigbee, 3G, WiMax …
- Mobility
- Security

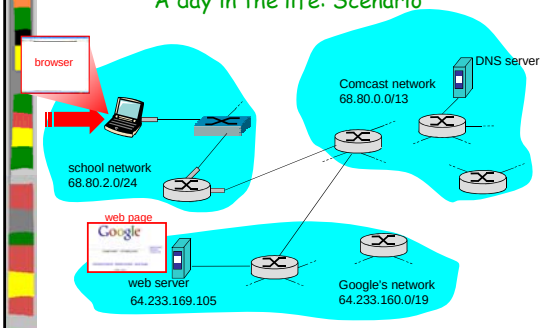
Link Layer

- 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

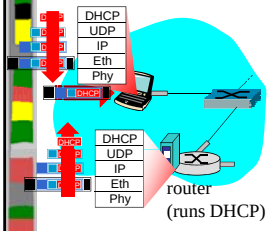
Synthesis: a day in the life of a Web request

- Journey down protocol stack complete!
  - Application, Transport, Network, Data 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

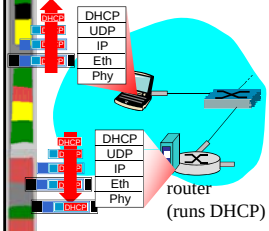


A day in the life… connecting to the Internet



- 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

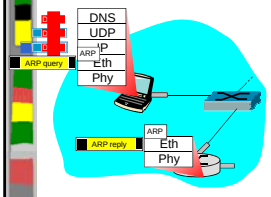
A day in the life… connecting to the Internet



- 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

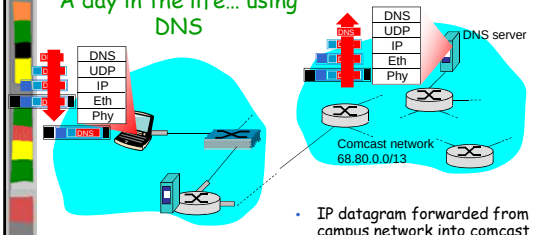
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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



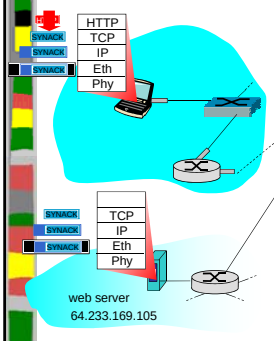
- Before sending *HTTP* request, need IP address of `www.google.com`: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encasulated in Eth. In order to send frame to router, need MAC address of router interface: **ARP**
- **ARP query** broadcast, received by router, which replies with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



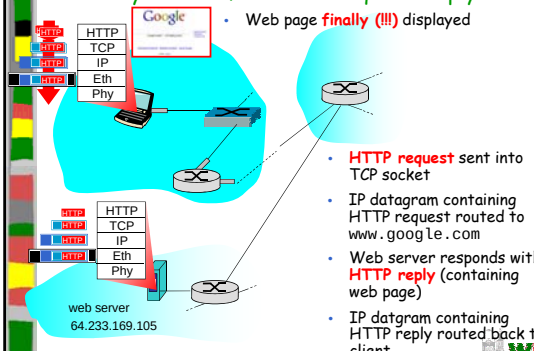
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by **RIP, OSPF, IS-IS** and/or **BGP** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client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

Chapter 5: Summary

- Principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- Instantiation and implementation of various link layer technologies
  - Addressing
  - Ethernet
  - Switched LANS
- Synthesis: a day in the life of a web request

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!
  - Wireless
  - Multimedia
  - Security
  - Network management