Chapter 5 - Link Layer

Sunday, April 17, 2016 9:50 AM

→ transfering information b/w physically adjacent nodes
→ sending frames across individual links
↳ link layer packet

two types of link-layer channels
Broadcast → connect multiple hosts (LAN, wifi, satellite, etc)
→ requires "medium access protocol" to coordinate

Point-to-point → connection b/w two routers/hosts

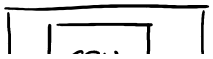
Services provided by Link-layer

Framing → encapsulating datagrams
Link access → medium access control (MAC) protocol
specifies rules/coordination of transmitting frames
Reliable delivery → guarantees error-free delivery, uses similar tactics as TCP (ACKs/retransmissions)
→ good for wifi & error-prone links
→ better to correct error locally than wait for end-to-end retransmission by TCP

Error Detection/Correction
↳ bit errors introduced by signal attenuation & EM noise
↳ reciever may be able to detect/correct bit errors
→ more sophisticated than network/transport layer
→ implemented in hardware

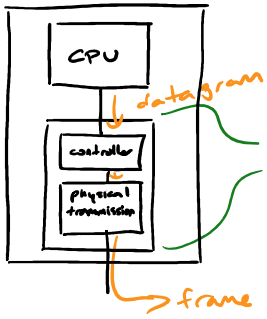
Flow control
↳ can buffer output to not overwhelm link

Where is link-layer implemented?
↳ exists on all hosts in NETWORK ADAPTER/NIC chip
→ encapsulates, adds error checking bits
provides flow control, etc.
→ extracts datagram, checks for errors,
passes to upper layer



most of link layer is hardware

passes to upper layer



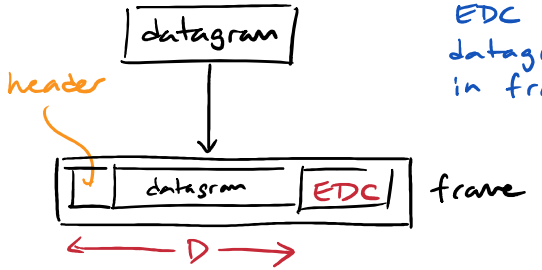
Network Adapter

most of link layer is hardware
↳ addressing information & error handling
& controller interrupts done via software

ERROR DETECTION/CORRECTION

↳ often provided by link layer
↳ provided in transport layer too sometimes

data bits, D, are appended with error detection and correction bits, EDC



EDC not only there to protect datagram but also information in frame header

DETECTION METHODS

parity
checksumming → typically transport layer only
cyclic redundancy check → typically link layer

PARITY CHECKS

↳ single parity bit check
→ adds one bit and chooses it so there are an odd or even # ones
→ detects if an odd number of bit errors occured. ~50% accuracy

↳ two dimensional parity
→ bits split into rows/columns & parity assigned to each row/column

data sent

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

data recieved

1	0	1	0	1	1
1	0	1	1	0	0 ← this parity wrong
0	1	1	1	0	1
0	0	1	0	1	

↑ this parity wrong

therefore can detect & correct that bit error

CHECKSUM

→ take each byte of data and treat as 16-bit integer
→ sum these all together
→ take ones complement (ie. swap all the bits)
→ store result in checksum field

reciever then does same thing & checks if they have same checksum value
→ if so error detected

requires little overhead

CYCLICAL REDUNDANCY CHECK (CRC)

→ typically used today
→ sender & reciever agree on R+1 bits called generator, G
↳ leftmost bit of G is 1
→ sender appends R bits to end of data, D, such that the bits D+R is exactly divisible by G

ie. $R = \text{remainder}\left(\frac{D \cdot 2^r}{G}\right)$ where r is length of G-1

→ receiver divides D+R by G and gets a remainder, an error has been detected

probability of error detected = $1 - 0.5^r$
where r is the length of a cluster of bit errors "bursts"

Multiple Access Links & Protocols

with broadcast links, there are multiple sending & recieving nodes to a shared channel

↳ two nodes send at same time the signals get mixed together & recievers can't understand
→ "collision"

↳ so must coordinate transmissions of nodes

IDEAL SCENARIO
→ when there's one node, it can transmit at full rate
→ when there are multiple nodes, they get equal share of full rate
→ decentralized so no coordinator & clocks not synchronized
→ simple, easy to implement

MAC PROTOCOLS

Channel Partioning

TDM/FDM → allocate time slot or frequency slot to each node
→ bad because wasted bandwidth when a node isn't transmitting

Random Access

↳ simple but inefficient
↳ most widely used

→ node transmits at full rate until a collision
→ if collision, node waits random amount of time then retransmits

Slotted ALOHA

→ time divided into slots equal to amount of time to transmit one packet
→ nodes are synchronized

→ when a node has a frame to send, it waits until beginning of next slot then sends

→ if there is a collision, node knows before the end of the slot & waits random delay

good b/c:
- if only one node, allows full rate
- mostly decentralized, nodes act independently
- simple

bad b/c:
- clocks must be synchronized
- wasted/empty time slots during random delay
  - ↳ max $\frac{1}{e} = 37\%$ of time slots used
    - ie. 100 Mbps cable used at 37 Mbps

Pure ALOHA

→ clocks not synchronized
→ nodes don't use slots
  → much more collisions happen

good b/c:
- allows full rate if one node
- completely decentralized
- simple

bad b/c:
- more collisions so more wasted bandwidth/time

max $\frac{1}{2e} = 18.5\%$ half as efficient as slotted ALOHA is cost of decentralization

CSMA/CD

→ similar to ALOHA but instead of transmitting as soon as frame to send it waits until channel is idle → "carrier sensing"
→ if collision is detected, stop transmitting immediately

efficiency depends on how fast nodes detect collision
↳ t_{prop} (max delay b/w nodes)

and time to transmit max-size frame
↳ t_{trans}

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

→ delay controlled via binary exponential backoff

↳ one collision, delay b/w $k=0$ & $k=1$
↳ two collisions, delay b/w $k=[0,3]$,
etc.

delay = $k \cdot$ bit times

amount of time needed to send certain # bits into channel (512 w/ ethernet)

SWITCHED LOCAL AREA NETWORKS

→ addressing done by MAC addresses (usually 6 Bytes long)
→ usually permanent

link layer uses MAC not IP to address
↳ can find MAC address of hosts within LAN from their IP address with ARP

address resolution protocol

→ host creates ARP table

ETHERNET

→ typical MTU 1500 bytes
→ connectionless → no handshake

→ unreliable → no ACKs

- uses CRC error checking
- starts with preamble of 8 bytes to synch clocks
- MAC protocol is CSMA/CD

SWITCHES

→ link layer routers
→ transparent
→ plug & play

has MAC address / interface table

↳ switch eliminates collisions
→ buffer frames so only one segment per time