Data Link Layer

- Today: LANs other than Ethernet
  - token rings
  - switched networks
  - cellular technology
- remaining issues
  - error detection
  - reliable transmission

Token Rings

- example: FDDI
- nodes arranged in ring
- all frames travel around the ring in the same direction
- ring acts like broadcast medium
  - all nodes see all frames
  - need algorithm to decide when to transmit

The Token

- special "token" frame circulates on ring
- must grab the token before sending
- to send a frame
  - remove token from circulation
  - send data frame
  - (some token rings: wait for acknowledgement to come back)
  - re-inject token

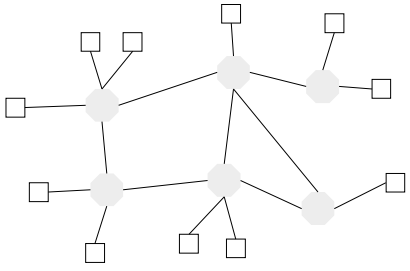
Token Ring Evaluation

- unlike Ethernet, can operate at full capacity
- but:
  - more expensive
  - have to wait for token before sending, even if network is idle
  - adding a host hurts performance, even if that host is silent
  - trickery required to recover from corrupted token

Switched LANs

- example: Myrinet
- point-to-point links connected with switching hardware
- advantages
  - total bandwidth scales as hosts are added
  - "cookie-cutter" approach

Switched LANs



Myrinet Details

- arbitrary topology
- hosts map the network initially
  - remap periodically (soft state)
- source routing: sender determines path through network
  - path encoded in header
- built from 8-way or 16-way switches

Myrinet vs. Alternatives

- performs well
  - same as 1 Gb/sec Ethernet, which is newer
- tolerant of configuration changes
  - like 10BaseT
  - unlike others
- moderate cost
  - more than Ethernet
- great for researchers
  - programmable adaptor

Error Detection

- Internet checksum
  - weak, but simple to code
- CRC
  - based on nontrivial math
  - strong
  - easy to build in hardware

Internet Checksum

```
short checksum(short buf[]){
   int sum = 0;

   for(int i=0; i<buf.length; ++i){
      sum += buf[i];
      if(sum & 0xffff0000){
         sum &= 0xffff;
         ++sum;
      }
   }
   return ~(sum & 0xffff);
}
```

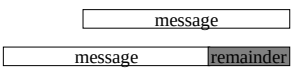
- complement of the ones-complement sum
- Misses some common errors
  - rearranged words
  - complementary errors in consecutive words

CRC (Cyclic Redundancy Code)

- based on polynomial math, modulo 2
  - addition = exclusive-or (modulo 2)
- think of a bit-string as representing a polynomial
  - i'th bit is on ==> x^i term in polynomial
- pick a "magic polynomial" C(x)
- transmit a bit-string (polynomial) that is divisible by C(x)

CRC

- to make a polynomial divisible by C(x)
  - let k = degree of C(x)
  - multiply message by x^k to get Q(x)
  - compute the remainder Q(x) % C(x)
  - P(x) = Q(x) - (Q(x) % C(x))



CRC

- sender transmits P(x)
- receiver verifies result is divisible by C(x)
  - if not, message was corrupted
- strength depends on properties of C(x)
  - popular values of C(x)
    - » CRC-8: x^8+x^2+x+1
    - » CRC-16: $x^{16}+x^{15}+x^2+1$
    - » CRC-32: $x^{32}+x^{26}+x^{23}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Computing CRC in Hardware

- compute remainder incrementally
  - start with zero
  - grow message one bit at a time
  - to shift in a bit:
    - » multiply polynomial by x (left shift)
    - » add one if shifted-in bit is one (bit flip)
    - » take remainder mod C(x)
      - after tricks, boils down to one AND and one XOR per bit

Reliable Transmission

- build reliable communication on top of unreliable
- acknowledgement, timeout, retransmission
- known as ARQ (Automatic Repeat reQuest)
  - lousy acronym, and a misnomer
- will discuss three variants of ARQ

Stop-and-Wait

- don't send a packet to a host until it has acknowledged the previous packet
  - covered in first lecture
  - recall: use one-bit sequence number on packets
  - implemented in Assignment 1
- advantage: simple
- disadvantage: poor use of bandwidth
  - especially if hosts are far apart

Efficiency of Stop-and-Wait

- assume
  - packet size = S (bytes)
  - network bandwidth = B (bytes/second)
  - delay between hosts = D (seconds)
- time per packet is T = 2D + S/B

$$\text{effective bandwidth} = \frac{S}{T} = \frac{B}{1 + \frac{2BD}{S}}$$

Bandwidth-Delay Product

- says how much data could be in transit at any moment
- for maximum efficiency, want to have this much data in transit
- "keep the pipe full"
- better ARQ variants fill the pipe
- in practice, often hard to figure out the bandwidth-delay product
  - adaptive algorithms

Improving Stop-and-Wait

- run several stop-and-wait protocols at once between a pair of hosts
  - "logical channels"
- use them in round-robin fashion
- packet (or ack) header says which channel the packet belongs to
- number of logical channels chosen big enough to fill the pipe

Performance

- with C logical channels, effective bandwidth is

$$\frac{B}{1+\frac{2BD}{CS}}$$

- by making C big enough, can use available bandwidth fully

Sliding Window

- extend stop-and-wait to allow multiple unacknowledged packets to be outstanding
  - limited number: "window size" W
- give each packet a sequence number
  - ack carries sequence number
- sender must have space to buffer W packets
- equivalent to logical channels
  - harder to understand, but more often discussed

Sliding-Window Details

- sequence numbers must be large enough to represent 2W distinct values
  - note: logical channel used lg(W) bits to identify channel, plus 1-bit sequence number
- different ways to deal with reordering, dropping
  - receiver can ignore (simple)
  - receiver can remember (efficient)