Direct link networks

- Encoding – How do you transmit one bit?
- Framing – How to delineate frames?
- Error detection
- Reliability – mostly for point-to-point
- Media Access Control (MAC) – for shared media
Physical media

- Signals propagate over a physical medium
  - E.g.: Copper wires, fiber, space

- Can modulate the signal in some way
  - Signal can be viewed as wave
  - Change amplitude (e.g., turn light switch on and off)
  - Change frequency, phase

- Speed is usually speed of light
  - In vacuum $3 \times 10^8$ m/s, in copper $\sim 2 \times 10^8$
How fast can you transmit information

- Depends on frequency, and signal/noise ratio

- **Shannon:** \( C = B \log_2(1 + S/N) \)
  - \( C \) is channel capacity in bps
  - \( B \) is bandwidth of line
  - \( S \) and \( N \) are avg. signal & noise power

- **Example: Telephone line**
  - 3 KHz b/w, 30 db S/N = \( 10^{30/10} = 1000 \)
  - \( C \approx 30 \) Kbps (so 56 Kbps modems need better S/N ratio)

- **Intuition for Shannon**
  - \( V \) voltage levels encode \( \log_2 V \) bits
  - To sample sine wave, need at least two samples per period
Now assume we can somehow modulate signal

How to encode binary data onto signals?

One approach: Non-return to zero (NRZ)

- Transmit 0 as low, 1 as high
NRZ drawbacks

- Consecutive 1s or 0s are problematic
- Non signal could be interpreted as 0s (or vice versa)
- "Baseline wander" problem
  - Where is threshold between low and high?
  - Could compare signal to average value, but avg. will drift
- Clock recovery problem
  - For long stretch of consecutive bits, might miscount
Alternate Encodings

- **Non-return to Zero Inverted (NRZI)**
  - Encode 1 with transition from current signal
  - Encode 0 by staying at same level
  - At least solves problem of consecutive 1s

- **Manchester**
  - XOR of NRZ symbol with clock
  - But now lose 50% efficiency
Illustration

<table>
<thead>
<tr>
<th>Bits</th>
<th>0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ</td>
<td><img src="image" alt="NRZ Waveform" /></td>
</tr>
<tr>
<td>Clock</td>
<td><img src="image" alt="Clock Waveform" /></td>
</tr>
<tr>
<td>Manchester</td>
<td><img src="image" alt="Manchester Waveform" /></td>
</tr>
<tr>
<td>NRZI</td>
<td><img src="image" alt="NRZI Waveform" /></td>
</tr>
</tbody>
</table>
4B/5B

- Every 4 bits of data encoded in a 5-bit code (see Table 2.4 in text)

- Remaining 13 codes used for other purposes
  - 11111 – line idle, 00000 – line dead, 00100 halt

- 5-bit codes selected to have no more than one leading 0 and no more than two trailing 0s
  - thus, never get more than three consecutive 0s

- resulting 5bit codes are transmitted using NRZI

- achieves 80% efficiency
Framing

- Break sequence of bits into a frame
- Typically implemented by network adaptor
Sentinel-based framing

- **Byte-oriented protocols (e.g., BISYNC, PPP)**
  - Place special bytes (SOH, ETX) at beginning, end of message

```
  SYN  SYN  SOH  Header  STX  Body  ETX  CRC
  8     8     8     8       8    16
```

- **What if ETX appears in the body?**
  - Escape ETX byte by prefixing DEL byte
  - Escape DEL byte by prefixing DEL byte
  - Technique known as *character stuffing*
Bit-oriented protocols

- View message as stream of bits, not bytes
- Can similarly use sentine (e.g., HDLC)

- HDLC begin/end sequence 01111110

- Use *bit-stuffing* technique to escape 01111110
  - When transmitting message, always append 0 to five consecutive 1s
  - Receiver uses bit to distinguish from begin/end sequence, strips bit out of payload
Counter approach

- **Drawback of “stuffing techniques”**
  - Length of frame depends on contents of payload

- **Alternative: Put length in header (e.g., DDCMP)**

  ![Frame Layout Diagram]

  - **Danger: Framing errors**
    - What if high bit of count gets corrupted
    - Adds 8K to length of frame, may lose many frames
    - CRC checksum helps detect error
Clock-based framing

- E.g., SONET: Synchronous Optical Network
  - Each frame is 125μs long
  - Look for header every 125μs
  - Encode w. NRZ, but scramble payload to help ensure lots of transitions
Error detection

• **Basic idea: Use a checksum**
  - Compute small checksum value, like a hash of the packet
  - If packet chances, checksum is likely to be wrong

• **Good checksum algorithms**
  - Want several properties, e.g., detect any single bit error
  - Details in a later lecture

• **Next problem: If discarding bad packets, how to ensure reliable delivery?**
Acknowledgements and Timeouts

- **Stop and wait approach**
  - Send packet, wait
  - Receive packet, send ACK
  - Receive ACK, send next packet
  - Don’t receive ACK, timeout and retransmit

- **Problems**
  - Might duplicate packet
  - Can’t keep pipe full (remember bandwidth-delay product)
Duplicates

- Can solve problem with 1-bit counter
  - Place in both Frame and ACK
  - Receiver knows if duplicate of last frame
  - Sender won’t interpret duplicate old ACK as for new packet
Sliding window protocol

- Still have problem of keeping the pipe full
  - Can generalize with > 1-bit counter
  - Allow multiple outstanding (unACKed) frames
  - Upper bound on unACKed frames, called window
SW sender

- Assign sequence number to each frame (SeqNum)
- Maintain three state variables:
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame sent (LFS)

\[
\begin{align*}
\text{LAR} & \leq \text{SWS} \\
\text{LFS} & \leq \text{SWS}
\end{align*}
\]

- Maintain invariant: LFS – LAR \(\leq\) SWS
- Advance LAR when ACK arrives
- Buffer up to SWS frames
**SW receiver**

- **Maintain three state variables**
  - receive window size (RWS)
  - largest acceptable frame (LAF)
  - last frame received (LFF)

- **Maintain invariant:** \( LFA - LFR \leq RWS \)

- **Frame SeqNum arrives:**
  - if \( LFR < SeqNum \leq LFA \) accept
  - if \( SeqNum \leq LFR \) or \( SeqNum > LFA \) discarded

- **Send cumulative ACKs**
Sequence number space

- How big should RWS be?
  - At least 1, no bigger than SWS

- How many distinct sequence numbers needed
  - If RWS=1, need at least SWS+1
  - If RWS=SWS, need at least 2SWS+1
Ethernet

- One of the most popular LAN network types

- CSMA/CD
  - carrier sense
  - multiple access
  - collision detection

- Frame format (Manchester encoding)
Ethernet 2

- **Addresses**
  - unique, 48-bit unicast address assigned to each adapter
  - example: 00:07:e9:52:c3:93
  - broadcast: all 1s
  - multicast: first bit is 1

- **Bandwidth:** 10Mbps, 100Mbps, 1Gbps

- **Length:** 2500m (500m segments with 4 repeaters)

- **Problem:** Distributed algorithm to provide fair access
Transmit algorithm

- If line is idle
  - send immediately
  - upper bound message size of 1500 bytes
  - must wait $9.6\mu s$ between back-to-back frames

- If line is busy
  - wait until idle and transmit immediately
  - called 1-persistent (special case of p-persistent)
Handling collisions

- If collision
  - jam for 32 bits, then stop transmitting frame
  - minimum frame is 64 bytes (header + 46 bytes of data)
  - delay and try again

- \( n \)th time: \( k \times 51.2 \mu s \), for \( k \leftarrow R \{0 \ldots 2^{\min(n,10)} - 1\} \)
  - 1st time: 0 or 51.2\( \mu s \)
  - 2nd time: 0, 51.2, or 102.4\( \mu s \)
  - 3rd time: 51.2, 102.4, or 153.6\( \mu s \)

- give up after several tries (usually 16)

- Min packet size determines max net length...
Token ring

- Idea: Frames flow around ring
- Capture special “token” bit pattern to transmit
Interface cards

- Danger if host dies, can wedge net
- Hardware typically has relays
  - In interface cards, or multi-station access units (MASUs)
Token ring frames

- **Frame format (Manchester encoding)**

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>48</th>
<th>48</th>
<th>Variable</th>
<th>32</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start delimiter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dest addr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Src addr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End delimiter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **First two bits of token differ from preamble by one bit**
  - Can just flip bit as you grab the token, so don’t need to buffer more than one byte at host

- **Token maintenance**
  - Various complications required to recover from lost token, etc.