Announcements

- Reading
  - Today: 3.1-3.3
  - Tuesday: 3.5-3.6
  - section 3.4 was covered before (during session)
Data Link Layer

- **Goal:** transmit error free frames over the physical link
- **Sample Issues:**
  - how big is a frame?
  - can I detect an error in sending the frame?
  - what demarks the end of the frame?
  - how to control access to a shared channel?
- **Examples:**
  - Ethernet framing
Frames

- **Slice Raw bit stream up into frames**
  - need to have manageable unit of transmission

- **Frame Boundary**
  - How do we know when a frame ends?
  - Character count
    - header indicates number of bytes
    - problem: what if the header is corrupt, can’t tell end of frame
  - Special character
    - ASCII: DLE STX ... DLE STE
    - need to use character stuffing to send DLE characters
      - send two DLE to indicate a DLE
  - Special bit pattern - no longer tied to ASCII
    - 01111110 - indicates end of frame
    - need to use bit stuffing to send 01111110 as data
      - insert 0 after 5 1’s
    - use link level invalid bit patterns
      - some bits may not be valid
Other Link Functions

- **Error Control**
  - may want to do sequence numbers and re-transmission
  - this introduces overhead, but useful if probability of failure is high

- **Flow Control**
  - provide rate matching between sender and receiver
  - sender has rules about when it can send: credits, etc.
Error Correcting Codes

- Idea: add redundant information to permit recovery
  - this is the dual of data compression (remove redundancy)

- Hamming distance (n)
  - number of bit positions that differ in two words
  - key idea: need n single bit errors to go from one word to the other
  - to detect d errors, need a hamming distance of d+1 from any other valid word.
  - to recover d errors, need a hamming distance of 2d + 1
    - any error of d bits is still closer to correct word

- Parity bit
  - ensure that every packet has an odd (or even) # of 1’s
  - permits detection of one 1 bit error
Error Codes (cont.)

- **Error Recovery**
  - Given m bits of data and r bits of error code
  - Want to correct any one bit error
  - There are n words one bit from each valid message
    - so need n+1 words for each valid message
    - thus \((n + 1) 2^m \leq 2^n\)
    - but \(n = m + r\) so \((m + r + 1) \leq 2^r\)

- **Hamming Code**
  - recovers from any one bit error
  - number bits from left (starting at 1)
    - power of two bits are parity
    - rest contain data
  - bit is checked by all parity bits in its sum of power expansion
    - bit 11 is used to compute parity bits 1, 2, and 8v
Hamming Code Example

<table>
<thead>
<tr>
<th>Char</th>
<th>ASCII</th>
<th>Hamming</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1001000</td>
<td>00110010000</td>
</tr>
<tr>
<td>a</td>
<td>1100001</td>
<td>10111001001</td>
</tr>
<tr>
<td>m</td>
<td>1101101</td>
<td>11101010101</td>
</tr>
<tr>
<td>l</td>
<td>1101001</td>
<td>01101011001</td>
</tr>
</tbody>
</table>

• Burst Errors
  – can send hamming codes by column rather than row
  – if use k rows, then can detect any burst error up to k bits
    • uses kr bits to check a block km bits long
Error Detection

- **Less bits are required**
  - if errors are infrequent, then this works better
  - assumes that re-transmission is possible

- **Cyclic Redundancy Codes (CRC)**
  - Use a generator function $G(x)$ of degree $r$
  - let $M'$ be the message with $r$ 0's on the end of it
  - divide $M'$ into $G(x)$ and compute remainder
    - use this as the $r$ bit CRC code
  - a code with $r$ bits will detect all burst errors less than $r$ bits
  - several $G$'s are standardized
    - CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$
    - CRC-16 = $x^{16} + x^{15} + x^2 + 1$
    - CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$
  - 16 bit CRC will catch
    - all single and double bit errors
    - all errors with an odd number of bits
    - all burst errors of length less than 16
CRC Example

Frame: 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after appending 4 zero bits: 1 1 0 1 0 1 1 0 0 0 0

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 0
Data Link Protocols

- **Stop And Wait**
  - send a frame
  - wait for ACK
  - need sequence number to tell re-transmission from next packet
    - lost ACK vs. lost frame

- **Sliding Window**
  - sequence numbers
  - can send up to window size number frames
  - Retransmission
    - Go Back N
    - Selective repeat