Announcements

- **Enrollment**
  - Now 11 on the waitlist
  - Will not be expanding class

- **Reading**
  - Chapter 3 (3.1-3.3)
Project #1 Notes

- **Ports**
  - End-points for communication
  - How to identify a processes rather than a machine

\[
\langle [\text{srcIp}, \text{srcPort}], [\text{destIp}, \text{destPort}] \rangle
\]

Debugging
- learn to use the debugger (ladebug)
- check that what you send it what you think you send
- print data just before it is sent
High-speed Networking Testbeds

- The Internet was taking, now what is next?
- A series of small projects to test new ideas
  - a “government gigabit” (622 Mbps)
- Issues:
  - the speed of light is fixed
    - round-trip coast to coast is 40msec
  - need for very high speed point-to-point connections
    - tele-medicine
    - video
    - coupling high-end computational resources
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?
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 \((n + 1) \cdot 2^m \leq 2^n\)
    - 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 8