Announcements

- **Enrollment**
  - Now 6 on the waitlist

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

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

Ports:
- Port: 25
- Port: 80
- Port: 666
- Port: 1234

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

<[srcIp,srcPort], [destIp, destPort]>
Gigabit 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
Telco Data Networks

- **X.25**
  - low speed (up to 64kbps) packet switched network
  - provides connection oriented services
    - call an end-point and hold the connection

- **ISDN**
  - slow speed (up to 128kbps) network
  - runs over a single copper pair
  - still connection oriented

- **B-ISDN**
  - higher speed version of ISDN
  - connection oriented
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