Announcements

- **Enrollment**
  - No one is left on the wait list

- **Reading**
  - Today: Chapter 5 (5.1-5.2)

- **Project #2**
  - Handout is on the web
  - Due on Wed Sept 26th (10 AM)
  - Reminder, not credit for late work
ATM Datalink Protocol

- **Cells (53 bytes)**
  - 5 byte header (4 bytes address plus 1 byte crc)
  - 48 byte payload
- **Header**
  - use CRC over the 32 bits of the header
- **How to find cell boundary?**
  - use shift register to check for valid checksum
    - 1/256 chance of a random match
  - use HUNT mode to increase chances
    - after a good cell, skip to the next cell boundary
    - must receive $\delta$ cells with checksum matches
- **Detecting loss of synchronization**
  - one bad cell is probably an error
  - many bad cells is likely a slip (loss of sync)
  - if $\alpha$ bad cells are seen in a row, switch to hunt mode
Simple Link Protocols

- **Stop-and-wait**
  - Sender
    ```
    while (1) {
        get frame from network layer;
        send frame;
        wait for ack;
    }
    ```
  - Receiver:
    ```
    while (1) {
        recv frame;
        send frame to network layer;
        send ack;
    }
    ```
  - Only one side active (sending) at once
  - Ensures rate matching
Sliding Window Protocol

- **Need to**
  - have multiple outstanding packets
  - limit total number of outstanding packets
  - permit re-transmissions to occur

- **Sliding Window**
  - permit at most N outstanding packets
  - when packet is ACK’d advance window to first non-ACK’d pkt

- **Retransmission**
  - Go-back N
    - when a packet is lost, restart from that packet
    - provides in-order delivery, but wastes bandwidth
  - Selective Retransmission
    - use timeout to re-sent lost packet
    - use NACK as a **hint** that something was lost
Sliding Window Example
Go-back N vs. Selective Retransmission

From: Computer Networks, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.
Pthreads

- Allows multiple threads of control on a process
- Basic operations:
  - `pthread_create(&threadId, attr, func, arg)`
    - creates a new thread
    - `threadid` is the id of the new thread
    - `attr` are special attributes of the thread (pass NULL)
    - `Func` is a pointer to a function to run
    - `arg` is an argument to that function

  - first thread of control must not exit (will kill other threads)
    - `pthread_join(threadId, status)`
      - wait for a specific thread to terminate
Using Locks for the Critical Section

- **Lock:**
  - if no thread has the lock mark it locked and return
  - if another thread has the lock, wait

- **Unlock:**
  - release the lock
  - if other threads waiting, notify one or all of them

- **Called mutexes in pthreads**
  - pthread_mutex is the data type
  - pthread_mutex_init used to initialize it
  - pthread_mutex_lock locks it
  - pthread_mutex_unlock releases it

- **Lock Grainularity**
  - want to lock enough to protect accesses
  - don’t want to lock too much to slow down the program