**Reliable Transmission**

- Because there may be uncorrectable errors (no matter what ECC scheme is used), how can the sender be sure that the receiver got the data?
  - Receiver must tell sender
  - Common scheme: positive acknowledgements (ACKs)

![Diagram of Reliable Transmission]

**Timeouts / Automatic Repeat Request (ARQ)**

- If no ACK comes back, the sender must re-send the data (ARQ)
  - When is the sender sure that no ACK is coming back?
    - Because as a practical matter delays are very difficult to bound, in
      - Sender chooses some reasonable timeout — if the ACK isn’t back in
        that much time, it assumes it will never get an ACK, and re-sends

![Diagram of Timeouts / Automatic Repeat Request (ARQ)]
Duplicate Detection: Sequence Numbers

- So that the receiver can detect (and discard) duplicates, distinct frames are given distinct sequence numbers
  - E.g., 0, 1, 2, 3, ...
- When a frame is re-sent, it is re-sent with the same sequence number as the original
- The receiver keeps some information about what sequence numbers it has seen, and discards arriving packets that are duplicates

Stop-and-Wait Protocol

- Sender doesn’t send next packet until he’s sure receiver has last packet
- The packet/ACK sequence enables reliability
- Sequence numbers help avoid problem of duplicate packets

Stop & wait sequence numbers

- Sequence numbers enable the receiver to discard duplicates
- ACKs must carry sequence number info as well

Problem with Stop-And-Wait: Performance

- Problem: “keeping the pipe full”
  - If the bandwidth-delay product is much larger than a packet size, the sender will be unable to keep the link busy
- Example
  - 1.5Mbps link x 45ms RTT = 67.5kb (8KB)
  - 1KB frames imply 1/8th link utilization
- Solution: allow multiple frames “in flight”
Solution: Allow Multiple Frames in Flight

- This is a form of pipelining

Flow Control

- Flow control:
  - Receiver needs to buffer data until it can be delivered to higher layers
  - If more data is sent before the receiver is ready, it will overwrite it, causing the receiver to lose data.
  - Additionally, if a frame is lost, the receiver will receive frames "out of order". It wants to buffer those frames to avoid retransmission, but cannot deliver them to the client until the missing frame is re-sent and received.
  - Finally, sender needs to buffer frames in case it has to resend them.

- Flow control is the notion that the sender must limit the rate at which it transmits to something below the raw bandwidth of the link.

- A common, important approach to flow control is the sliding window protocol.

Sliding Window Protocol

- There is some maximum number of un-ACKed frames the sender is allowed to have in flight.
  - We call this the "window size".
  - Example: window size = 2

When the window is full, each ACKed frame allows the sender to send one more frame.
Sliding Window: Sender

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

  Direction of slide

  ... A B C D E F G H I J K ... 

  LAR  LFS 

- Advance sas when ACK arrives
- Buffer up to sas frames

Sliding Window: Receiver

- Maintain three state variables
  - receive window size (RWS)
  - largest frame sequence number acceptable (LFA)
  - last frame received (LFR)
- Maintain invariant: LFA - LFR <= SWS

  i-loop

  ... A B C D E F G H I J K ... 

  LFA  LFR 

- Frame SeqNum arrives:
  - if LFR < SeqNum <= LFA: accept + send ACK
  - if SeqNum > LFA: discard

ACKs

- Send cumulative ACKs
  - send ACK for largest frame such that all frames less than this have been received
  - Why?
- Send an ACK each time a packet with SeqNum in the window arrives
  - even if you've seen that packet already
  - Why?

Sliding Window Example
Sliding Window Summary

- Sliding window is best known algorithm in networking
- First role is to enable reliable delivery of packets
  - Timeouts and acknowledgements
- Second role is to enable in order delivery of packets
  - Receiver doesn’t pass data up to app until it has packets in order
- Third role is to enable flow control
  - Prevents sender from overflowing receiver’s buffer

Sequence Number Space

- SegNum field is finite; sequence numbers wrap around
- Sequence number space must be larger than number of outstanding frames
- SWS <= MaxSeqNum-1 is not sufficient
  - Suppose 3-bit SegNum field (0-7)
  - SWS=7
    - Sender transmits frame 0.6
    - Arrives successfully, but ACK lost
    - Sender retransmits 0.6
    - Receiver expecting 7, 0, 5, but receives the original incarnation of 0.5
- SWS <= (MaxSeqNum+1)/2 is correct rule
- Intuitively, SegNum “slides” between two halves of sequence number space