**Transport Layer (Continued)**

* rdt3.0
  + Sender waits “reasonable” amount of time for ACK by setting timer
  + Network protocol limits use of physical resources
* Pipelined protocols (and sliding-window protocol)
  + GBN (Go-Back-N)
    - Sender: N packets, single timer for the oldest one
    - Receiver: no buffer, out-of-order packets are discarded, sends cumulative ack.
  + SR (Selective Repeat)
    - Sender: N packets, multi-timer for every packets,
    - Receiver: has buffer, can accept out-of-order packets, sends individual ack.
    - Delima solution: Ns + Nr ≤ 2^k
* Components of reliable protocols
  + Checksums (error detection)
  + Timers (loss detection)
  + Acknowledgments -- cumulative/selective
  + Sequence numbers -- duplicates, windows
  + Sliding windows (efficiency)
* TCP
  + Overview
    - Point-to-point -- one sender, one receiver
    - Reliable, in-order byte stream
    - Pipelined -- congestion and flow control set window size
    - Send and receive buffers
    - Full-duplex data -- bi-directional data flow in same connection
    - Connection-oriented -- handshaking
    - Flow controlled -- sender will not overwhelm receiver
  + TCP segment structure
    - Source port #, dest port #, sequence number, ack number …
    - TCP segment size



* + - * IP packet ≤ MTU (Maximum Transmission Unit)
      * TCP segment ≤ MSS (Maximum Segment Size) bytes
      * MSS = MTU - IP header - TCP header (≥ 20B)
  + Reliable
    - Checksum: computed over header and data
    - Sequence numbers are byte offsets
      * Sequence #: 1st byte in segment = k + ISN (initial sequence number)
    - Receiver sends cumulative acknowledgements (like GBN)
      * ACK: next expected byte = seq # + length(data)
    - Receivers can buffer out-of-order sequence packets (like SR)
    - Sender maintains a single retransmission timer (like GBN) and retransmits on timeout
      * RTT
        + EstimatedRTT = ( 1 - alpha) \* EstimatedRTT + alpha \* SampleRTT (alpha = 0.125)
        + TimeoutInterval = EstimatedRTT + 4\*DevRTT (“safety margin”)

Large variation in EstimatedRTT → larger “safety margin”

* + - * + DevRTT = (1 - beta) \* DevRTT + beta \* |SampleRTT - EstimatedRTT| (Beta = 0.25)
      * TCP sender events:
        + Data received from app

Create segment with seq #

Seq # is byte-stream number of first data byte in segment

Start timer if not already running

* + - * + Time out

Retransmit segment that caused timeout

Restart timer

* + - * + ACK received

If ack previously unacked segments

Update what is known to be ACKed

Start timer if there are still unacked segments

* + - Introduces fast retransmit: optimisation that uses duplicate ACKs to trigger early retransmission
      * If sender receives 3 duplicate ACKs for same data
  + Flow control
    - Receiver controls sender, so sender won’t overflow receiver’s buffer by transmitting too much/fast
    - Receiver “advertises” free buffer space by including rwnd value in TCP header of receiver-to-sender segments