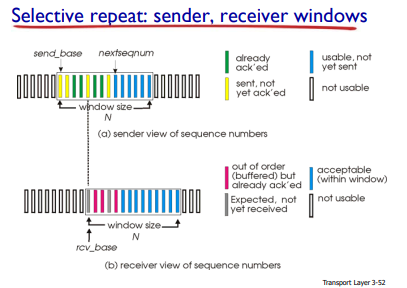
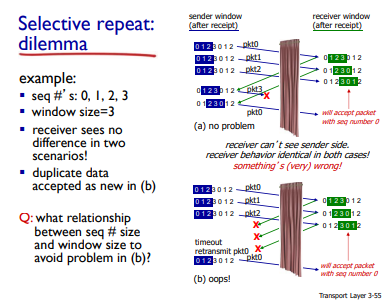
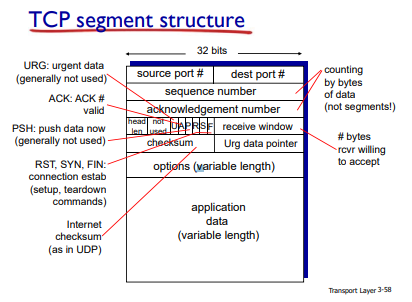
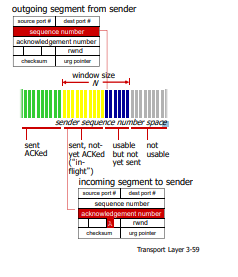
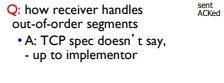
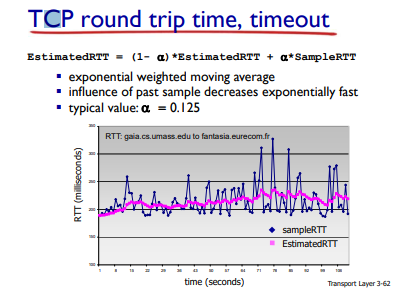
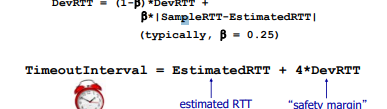
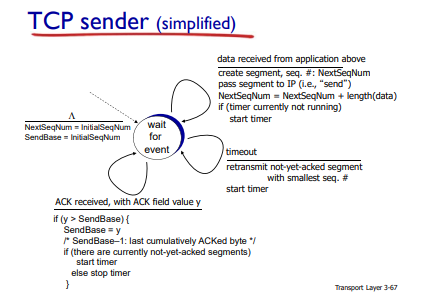
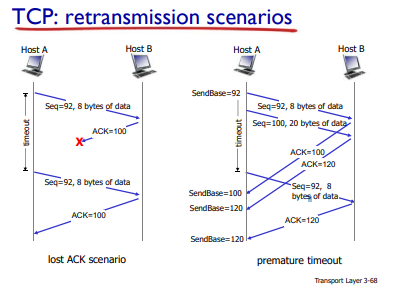
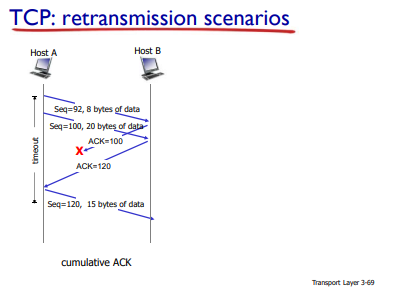
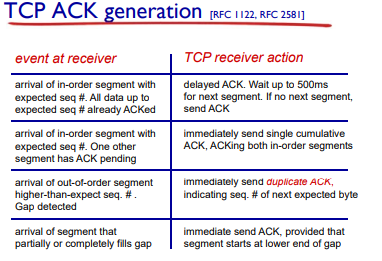
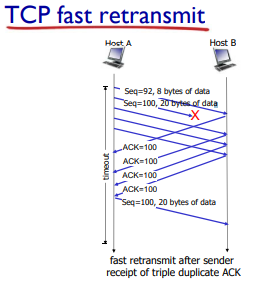
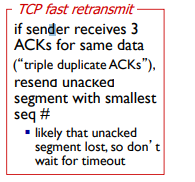
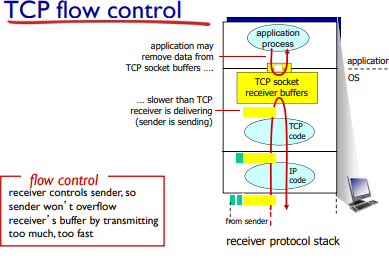
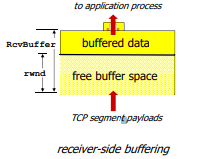
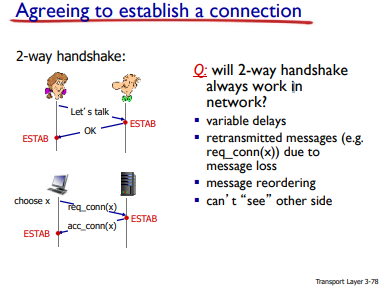
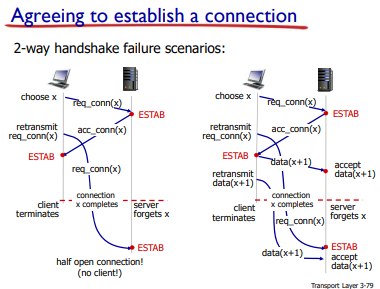
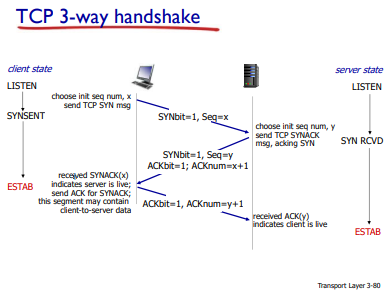
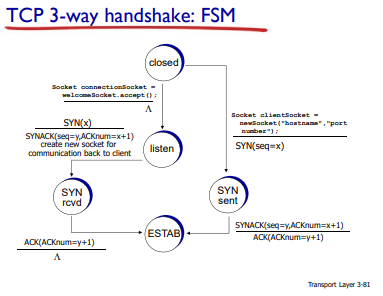
* Selective Repeat
  + Receiver individually acknowledge all correctly received pkts
    - Buffed pkts, as needed, for eventual in-order deliver to upper layer
  + Sender only resends pkts for which ACK not received
    - Sender timer for each unACKed pkt
  + Sender window
    - N consecutive seq #’s
    - Limits seq #s of sent, unACKed pts
  + 
  + 
* TCP Overview
  + point - to - point: one send, one receiver
  + Reliable, in order byte stream
    - No “message boundaries”
  + Pipelined: TCP congestions and flow control set window size
  + Full duplex data:
    - Bi directional data flow in same conection
    - MSS: maximum segment size
  + Connection-orientated
    - handshaking(exchange of control msgs) inits sender, receivers state beore data exchange
  + Flow controlled
    - Sender will not overwhelm receiver
  + 
* TCP sequence numbers, ACKs
  + Sequence numbers:
    - Bytes stream “number” of first byte in segment’s data
  + Acknowledgements:
    - Sequence numbers of next byte expected from other side
    - Cumulative ACK
  + 
* TCP round trip time, timeout
  + How to set TCP timeout value?
    - Longer than RTT
      * But RTT varies
    - Too short: premature timeout, unnecessary retransmissions
    - Too long: slow reaction to segment loss
  + How to estimate RTT?
    - SampleRTT: measured time from segment transmission until ACK receipt
      * Ignore retransmissions
    - Sample RRT will vary, want estimated RTT smoother
      * Average several recent measurement, no tsure current SampleRTT
  + Round Trip Time Formulae
  + 
  + Timeout Interval: EstimatedRTT plus “safety margin”
    - Large variation in EstmatedRTT -> larger safety margin
  + Estimated SampleRTT deviation from EstimatedRTT:
    - 
* TCP Reliable Data transfer
  + TCP Creates rdt service on top of IP’s unreliable service
    - Pipelined segments
    - Cumulative acks
    - Single retransmission timer
  + Retransmission triggered by:
    - Timeout events
    - Duplicat ACKs
* TCP Sender events:
  + Data rcvd from app:
    - Create segment with seq #
    - Seq # is byte-stream number of first data byte in segment
    - Start timer if not already running
      * Think of timer as for oldest unacked segment
      * Expiration interval: TimeOutInterval
  + Timeout:
    - Retransmit segment tha caused timeout
    - Restart timer
  + Ack rcvd:
    - If ack acknowledges previously unacked segments
      * Update what is known to be ACKed
      * Start timer if there are still unACKed segments
  + 
  + 
  + 
* TCP ACK Generation
  + 
* TCP Fast retransmit
  + Timeout period: often relatively long:
    - Long delay before resending lost packet
  + Detect lost segments via duplicate ACKs
    - Sender often sends many segments back to back
    - If segment is lost, there will likely be many duplicate ACKs.
* 
* TCP Flow Control
  + 
  + Receiver “advertises” free buffer space by including **rwnd** value in TCP header of receiver to sender segments
    - RcvBuffer: size set via socket options (typically default is 40% bytes)
    - Many operating systems auto adjust RcvBuffer
  + Sender limits amount of unacked, (“in-flight”) data to receiver’s rwnd value
  + Guarantees receive buffer will not overflow
  + 
* Connection Management
  + Before exchanging data, sender/receiver “handshake”:
    - Agree to establish connection (each knowing the other wiling to establish connection)
    - Agree on connection parameters
* 
* 
* 
* 
* TCP: closing a connection
  + Client, server each close their side of connection
    - Send TCP segment with FIN bit = 1
  + Respond to received FIN with ACK
    - On receiving FIN, ACK can be combined with own FIN
  + Simultaneous FIN exchanges can be handled
* 