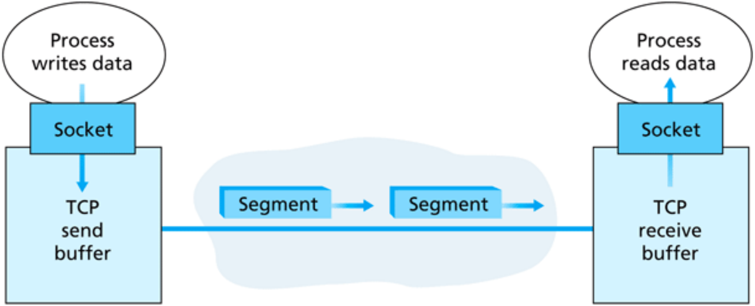
TCP Reliability

Daniel Zappala

CS 460 Computer Networking
Brigham Young University

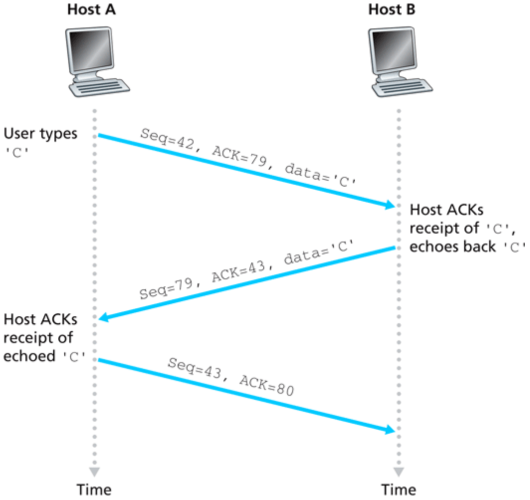
How does TCP implement reliable transfer?

TCP Segmentation

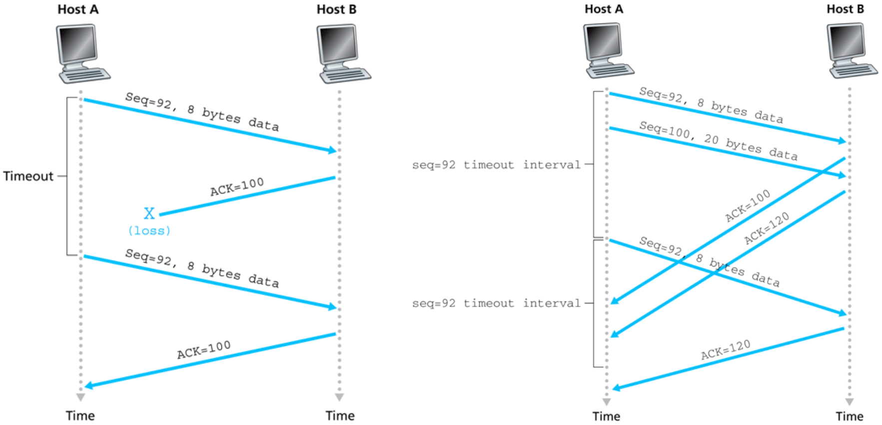


Sequence and ACK Numbers

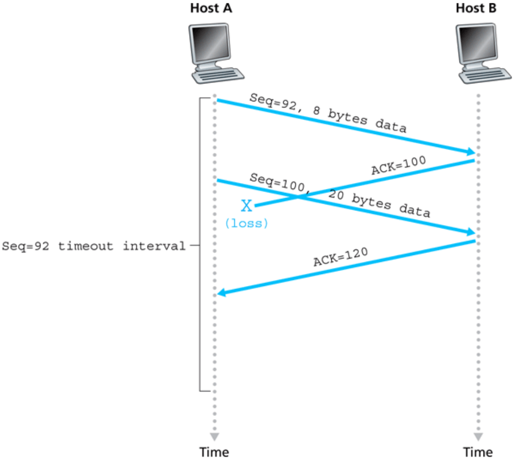
- sequence number: byte number of this segment within the byte stream
- ACK number: sequence number of next byte expected from sender



TCP Retransmission Scenarios



TCP Retransmission Scenarios (continued)



Generating TCP ACKs

Receiver Event	TCP Action
Arrival of in-order segment with expected sequence number. All previous data ACKed.	Delayed ACK. Wait up to 500 ms for next segment. If no next segment, send ACK.
Arrival of in-order segment with expected sequence number. One other segment has ACK pending.	Immediately send single cumulative ACK that covers both in-order segments.
Arrival of out-of-order segment, sequence number larger than expected. Gap detected.	Immediately send duplicate ACK, indicating sequence number of next expected byte.
Arrival of segment that partially or completely fills gap.	Immediately send ACK, provided that segment starts at lower end of gap.

RTT Estimation

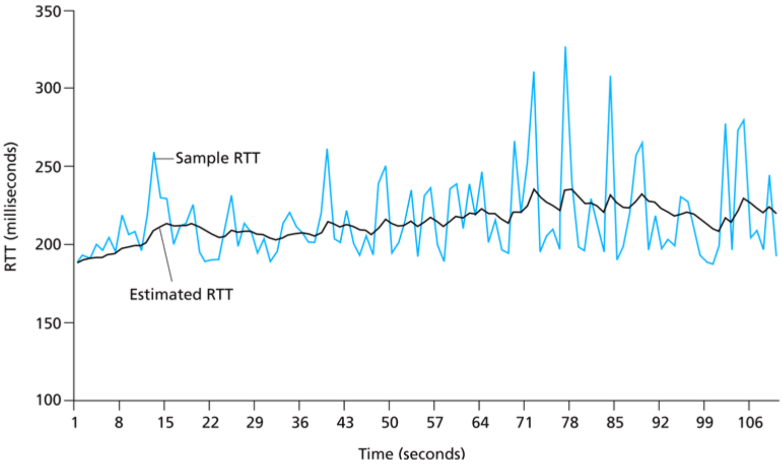
Setting the TCP Timer

- how long should the TCP timer be?
  - needs to be longer than RTT
  - too short: premature timeout - duplicate segments
  - too long: slow reaction to segment loss
- **problem: RTT can vary dramatically, depending on queueing delay**

Estimating RTT

- SampleRTT: measured time from segment transmission until ACK is received
  - ignore retransmissions due to loss
- average measurements of SampleRTT over time
- $\mathit{EstimatedRTT} = (1 - \alpha) * \mathit{EstimatedRTT} + \alpha * \mathit{SampleRTT}$
  - EWMA: exponential weighted moving average
  - influence of past samples decreases exponentially fast
  - typical value $\alpha = 0.125$
  - produces a smooth estimate of RTT

Example



- sample delay from a machine at UMASS to a machine at Eurecom in France

Fine-Tuning the RTT Estimate

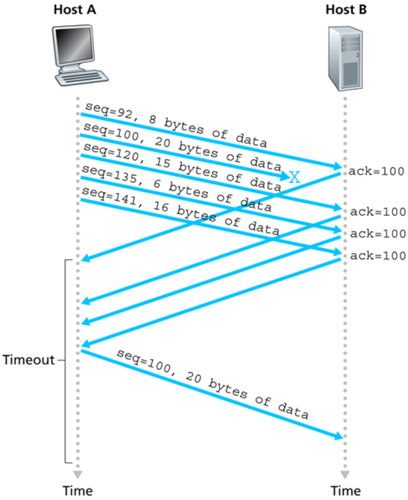
- calculate the deviation of the SampleRTT from the EstimatedRTT
- $DevRTT = (1 - \beta) * DevRTT + \beta * |SampleRTT - EstimatedRTT|$
  - typical value $\beta = 0.25$
- $TimeoutInterval = EstimatedRTT + 4 * DevRTT$
  - large variation in estimated RTT provides a larger safety margin

Fast Retransmit

Fast Retransmit

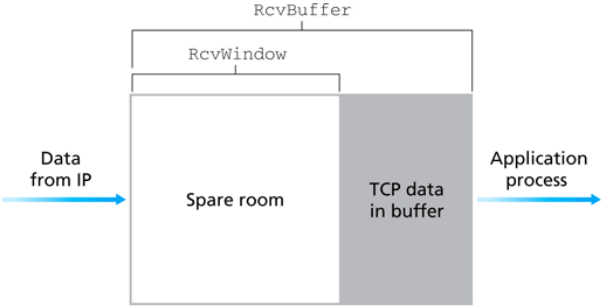
- problem: timeout period is generally very long
  - long delay when recovering lost packet
  - big performance hit
- solution: detect lost segments with duplicate ACKs
  - duplicate acks when one packet in a window is lost
  - if sender gets 3 duplicate ACKs, assume packet loss
  - fast retransmit before timer expires
- why not switch to selective ACKs?
  - requires only a small change to TCP sender

Fast Retransmit Example



Flow Control

Flow Control



- receiver stores incoming packets in a receive buffer
- application pulls data from the buffer
- TCP can't control how fast data is removed from buffer, so it may fill up
- flow control: ensure sender doesn't overflow the receiver's buffer

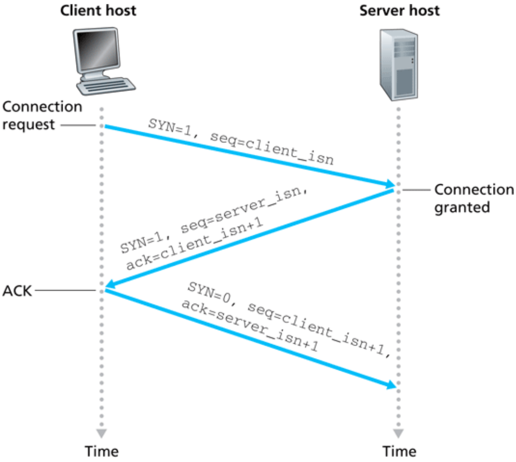
Advertised Window

- TCP receiver advertises a RcvWindow to sender, which is equivalent to the spare room in its buffer
- $RcvWindow = RcvBuffer - (LastByteRecvd - LastByteRead)$
- sender limits un-ACKed data to RcvWindow

Connection Management

Opening a Connection: Three-Way Handshake

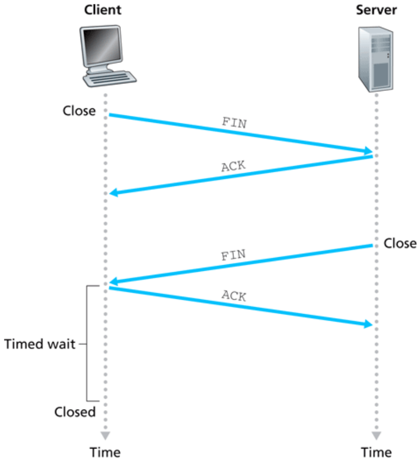
1. client sends TCP SYN segment
   - client specifies initial sequence number, MSS, RcvWindow
   - no data
2. server responds with SYN/ACK segment
   - server allocates buffers
   - server specifies initial sequence number, MSS
3. client responds with ACK segment
   - may contain data



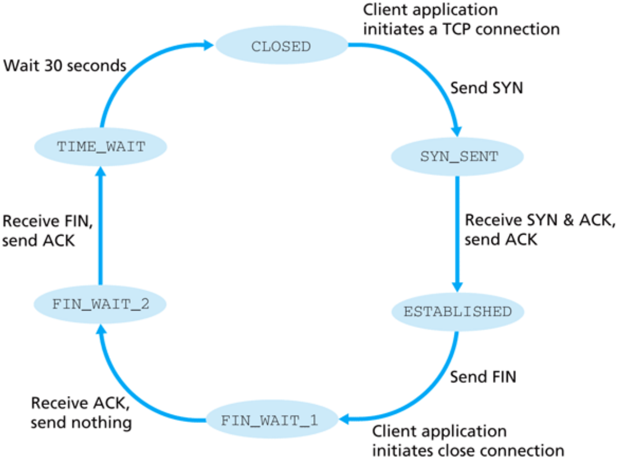
Closing a Connection

1. client sends TCP FIN segment to server
2. server responds with ACK
3. server sends FIN segment
4. client responds with ACK
    - enters timed wait
    - responds to FINs with ACK

- lots of variations, e.g. combining FIN/ACK
- various scenarios lead to different timers being set

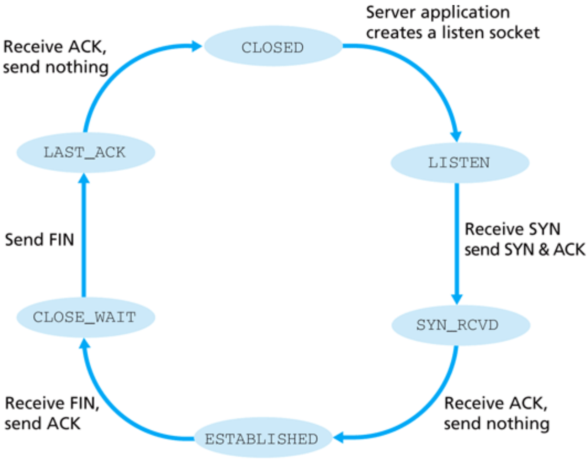


TCP Client Connection States



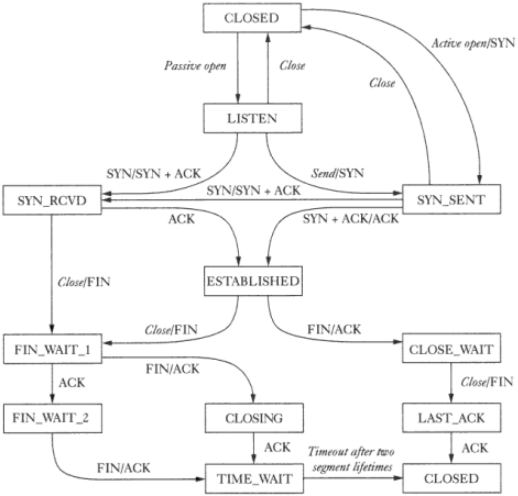
- grossly simplified

TCP Server Connection States



- grossly simplified

TCP State Transitions



TCP Header

- sequence and ACK number: count in terms of bytes
- flags
  - A: ACK number is valid
  - R: RST: reset connection
  - S: SYN: establish connection
  - F: FIN: close connection
  - U: URG: urgent data, typically not used
  - P: PSH: push (send) data immediately, used for TELNET
- receive window: number of bytes receiver can accept