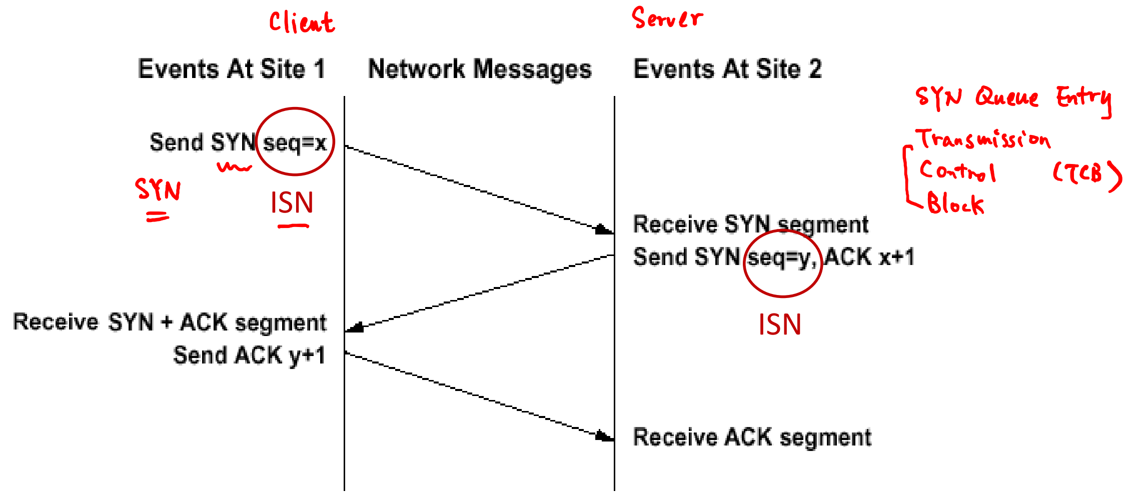
- TCP uses three-way handshake to establish a connection



"SYN Flood Attack"

ISN: Initial Sequence Number (a byte index) *

"SYN Cookies"

ISN: 32 bits

y: 5 bits | 3 | 24 bits

t = time mod 32

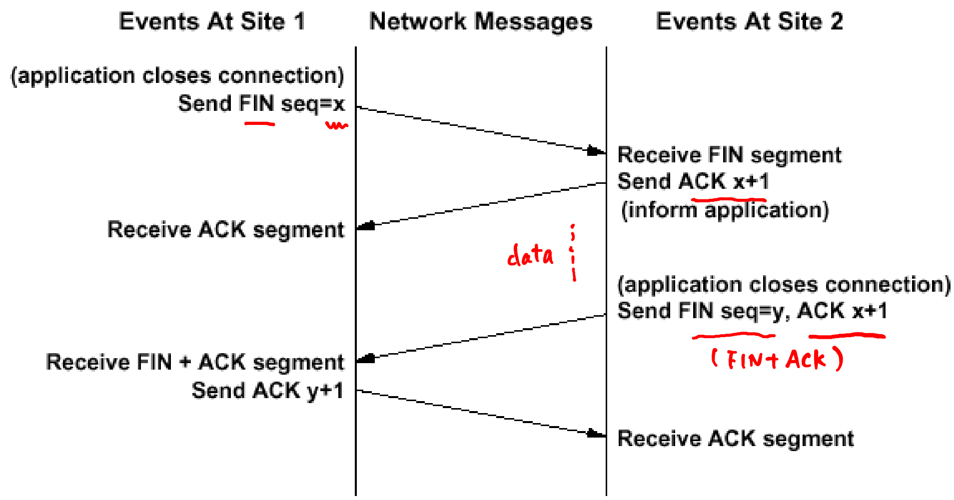
m: 3 bits : 1 of 8 possible options on MSS (Max Segment Size)

S = hash (server IP, server Port #, Client IP, Client Port #)

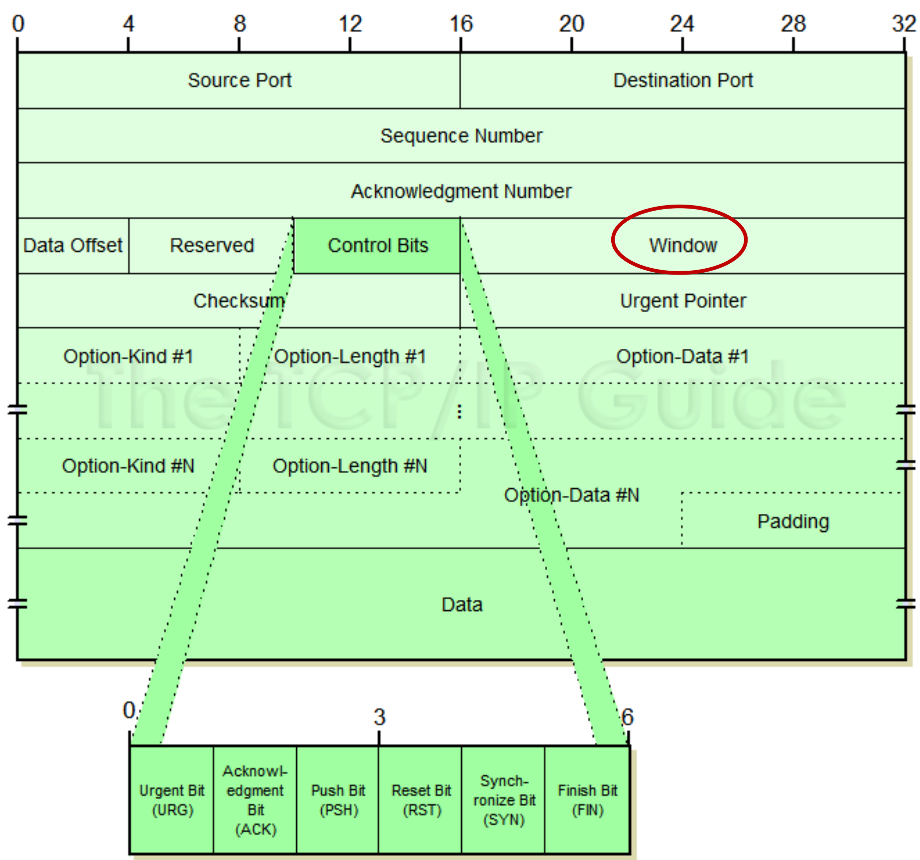
At server side: get ACK(y+1)

- check t against current time to see if connection has expired
- recompute S to make sure this is a valid cookie
- decode m to figure out the MSS option

- TCP uses modified three-way handshake to terminate a connection



3. TCP Flow Control



- TCP flow control prevents the sender from overwhelming the receiver with too much data
  - Receiver advertises the available buffer space (rwnd)

    "window" field
  - Sender makes sure that the amount of outstanding data (swnd) is less than the receiver-advertised buffer space
    - swnd ≤ rwnd

4. TCP Error Control

- TCP Error Control
  - Special version of Selective Repeat ARQ (SR ARQ)
    - ACKs only
    - No NAKs
    - Retransmit upon
      - Retransmission Timeout (RTO)
      - Reception of the 4th ACK with the same sequence number

        (3rd dup ACK)

        implicit NAK

5. TCP Retransmission Timeout (RTO)

- Every time TCP sends a segment, it starts a timer and waits for an ACK
- If the timer expires, TCP assumes that the segment was lost and retransmits it

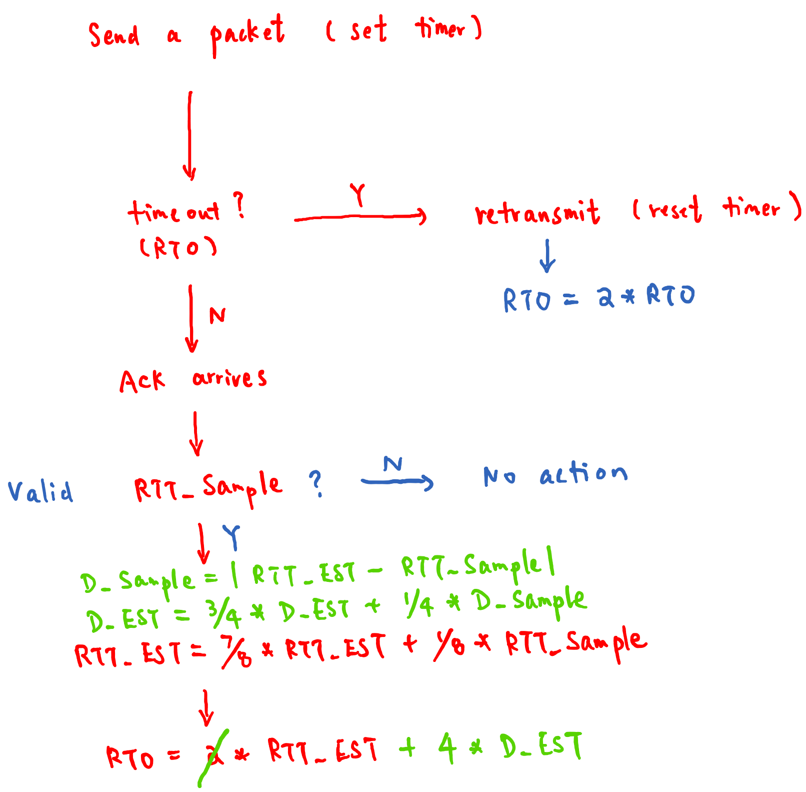
- TCP software must accommodate:
  - differences in the time required to reach various destinations
  - changes in time required to reach a given destination as traffic load varies

- TCP accommodates varying Internet delays by adapting the RTO

TCP RTO

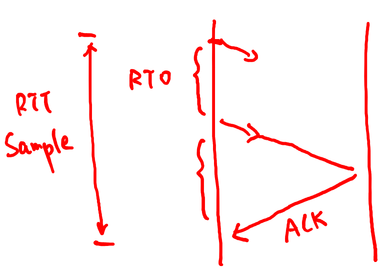
- Implications of poorly-selected RTO
  - RTO too small $\rightarrow$ unnecessary retransmissions
  - RTO too big $\rightarrow$ low throughput

- RTO must adapt to RTT
  - Using RTT estimates
    - RTT_EST = $\alpha \times$ RTT_EST + $(1-\alpha) \times$ RTT_Sample
    - α is typically 0.875
    - Exponential Weighted Average
  - RTO = $\beta \times$ RTT_EST, where β is typically 2

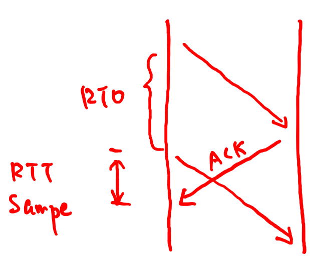


Problem #1: ACK Ambiguity

- ACK Ambiguity: which packet to associate with an ACK in case of retransmission?
  - With the original transmission?
  - With the most recent retransmission?



RTT_Sample > RTO
> True RTT



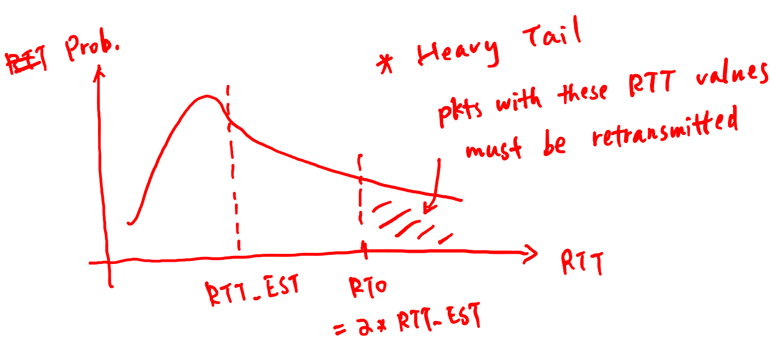
True RTT > RTO
> Sample RTT

Problem #1: ACK Ambiguity

- Karn's Algorithm
  - Update RTT_EST only with valid RTT samples that are associated with non-retransmitted packets
  - However, ignoring RTT of re-transmitted packets could lead to insensitivity to long delay
  - Hence, increase RTO upon each packet retransmission:
    - RTO = $\gamma \times$ RTO, where γ is typically 2
    - Until a valid RTT sample is obtained when RTO is reset using the newly-updated RTT_EST

Problem #2: High Variance in RTT

- RTT_EST as computed only gives a good mean
- High Variance in RTT
  - Packets with RTT larger than RTO = $\beta \times$ RTT_EST must be retransmitted



Problem #2: High Variance in RTT

- Jacobson's Algorithm (Modified RTO Computation)

[STEP 1] $D\_EST = \delta \times D\_EST + (1-\delta) \times \underbrace{|RTT\_Sample - RTT\_EST|}_{D\_Sample}$

- δ is typically 0.75

[STEP 2] $RTT\_EST = \alpha \times RTT\_EST + (1-\alpha) \times RTT\_Sample$

[STEP 3] $RTO = RTT\_EST + \eta \times D\_EST$

- η is typically 4

RTO	D_EST	RTT_EST	D_Sample	RTT_Sample
40	4	24		
				valid sample
45	5	25	8	32
90		no updates ;	reset timer	- - - - -----
				invalid
				invalid
49	5.75	26	8	33