TCP timeout and retransmission
TCP interactive flow algorithms

14강

TCP timeout

- One of the two means of detecting a packet loss
  - ACK not arriving until retransmission timeout (RTO)
- RTO = avg(RTT)+4*std(RTT)
  - Cf. Chevyshef inequality
  - Avg RTT is a low-pass filter output of RTT graph
  - Standard deviation is not used; actually mean deviation is used

TCP timeout

- RFC 2988 stipulates that max(1s, RTO) should be used
- RTO vs. RTT: see Fig. 14-3

Fast Retransmit

- The second means of detecting packet loss
  - 3 consecutive duplicate ACKs
  - E.g. 4320, 4320, 4320, 4320 → trigger FR
- It's faster than timeout method

Delayed ACK algorithm

- TCP suppresses ACKs by default
  - Uses 200ms timer to suppress
  - Waits until a packet in the opposite direction starts
  - It's because the placeholders for the two pieces of ACK information (ACK number + window size) are in the opposite direction data packet anyway

Delayed ACK algorithm

- If 200ms timer expires while suppressing ACK, the ACK segment should go
  - Such ACK segment has only header (no body carrying data in the opposite direction)
- If window size (i.e. available receive socket buffer space) is newly made larger than 2MSS, suppression should end
  - Let the TCP sender know that we can receive more! Let it send!

Delayed ACK algorithm

- Delayed ACK is on by default
- If you turn it off (e.g. by editing registry on Windows), all TCP connections will have Delayed ACK off
- Delayed ACK is to eliminate unnecessary ACK transmissions on the receiver side of a data channel
  - It runs on either side

Nagle's algorithm

- This is sender side story
- We do not want to generate "small" segments
  - Small = less than MSS: e.g. 1B, 1459B, etc.
- If a small segment has to be sent, check if there is any outstanding small segment unacknowledged yet
  - If so, wait until the ACK comes back

Nagle's algorithm

- Gather bytes in the send socket buffer in the mean time
- When the ACK for the prior small segment comes back, carry all the gathered bytes in one segment
  - If more than MSS bytes are gathered in the send socket buffer, you can always send a full segment irrespective of Nagle

Zero window ACK and persist timer

- If the receive socket buffer is full, window size=0 is sent in the ACK
  - This is a special condition, and the TCP sender notes it; run so called the “persist” timer
- The only way to break the sender free is sending an ACK with window size != 0
  - Then sender sends some data, gets an ACK, etc. etc.

Zero window ACK

- But what if the window advertisement ACK is lost?
  - All packets are equal in the Internet; it can die
- The sender is frozen
  - As it thinks window size is still 0 as no ACK with WS!=0 arrives
- The receiver has nothing to send more
  - It already sent an ACK; no retransmission of ACK in TCP

Zero window ACK

- That's why the sender runs the persist timer
- If the timer expires, send a "probe" packet
  - Carries at least 1 byte taken from the head of the send socket buffer (normal data)
  - TCP always allows such 1 byte packet to be send beyond the window

Zero window ACK

- If there is some space in the receive socket buffer, the probe data will be received and ACKed
  - Then the window opening ACK must have been lost indeed
  - This will break the livelock and set the TCP running again
- If the situation is the same, the receiver will send another zero window ACK

Silly Window Syndrome

- It doesn't mean that SWS is actually happening in today's Internet
- SWS means the sender transmits "small" packets
- Both the sender and the receiver have a preventive mechanism
  - Sender: Nagle

SWS avoidance

- What does receiver do?
  - It lies
  - It lies about the available receive socket buffer size
  - If it is less than 2MSS, it just says "0"
    - Otherwise the sender will get a small opening, and to fill the opening, it will have to send a small packet
- Note that the receiver mechanism is unnecessary if the Nagle is working fine at the sender or vice versa
  - We do not trust the other guy…that's our philosophy; do you?

Keepalive timer

- Not in the TCP spec
- The server can check if the client is still there (“half-open” or not?)
  - Because half-open only wastes server’s resource
- Sends a “probe”, but this is different from the probe used in the persist timer
  - Probe is an empty packet with an old sequence number (older than the client is expecting)
  - Receiver immediately sends an ACK with a correct ACK number if it is alive!