Lecture 12

24 September 2021 23:14

- Recap → TCP properties
  → TCP Header Structure
- Simple version of TCP (without) ⇒ Flow control ✓, Cong. control ✓
- State Diagram of TCP sender
- Few Example scenarios with timing Diagrams.

Recap - properties of TCP

1. Connection-oriented

Control information (3-way handshake)

↳ what are the buffer sizes they want to use? ✓

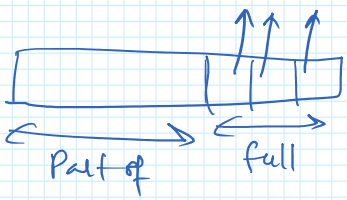
↳ Exchange what init seq. # to use? ✓

Reliable
- Maintain State variables like window
- Ack Packets
- Previous Pckt sent

2. **Point-Point**
   - one sender to one receiver
   - It is not designed to support Multicasting / Broadcasting

3. **Flow-control** → TX won't overwhelm Rx.

Sender has to Adaptively adjust its window size based on current avl. space in the Rx Buffer!!



4. **Full-Duplex**

When a TCP conn is established b/w two devices, it means that

⇒ over same TCP conn.

TCP Header Structure

- Process Management ⇒ Dest port # } 16 bit
  Source Port # } 16 bit

- Process Management → Dest port # } — 16 bit
  → Sender Port # } — 16 bit
- Reliability → Checksum — 16 bit
  → Seq num — 32 bit
  → Ack num — 32 bit
- Flow-Control → recv. window — 16 bit
- Flags → SYN
  → FIN
  → RST
  → URG
  → PSH
  → Ack = 1
- Urgent Data pointer — 16 bit
- Optional Header
- Data Field

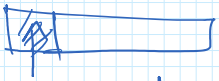
Typical Header size = 20 bytes.

Seq num & Ack num

✓ Cumulative Acks

✓ Seq num based byte-stream count

Note: NOT BASED on segment number



Byte-count of first byte in that segment.

- Go Back N : Ack # N. ⇒ all pkts till 'N' are received in-order.

TCP : Ack # N ⇒ all byte till byte-count "N-1" have been recv. in-order.

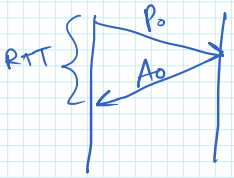
- In other words, it just means, the TCP is waiting for the next segment which is expected to start with byte-count "N". ✓

Timeout & Re-Transmissions

byte-count W ✓

Time out & Re-Transmissing

- Time-out : to re-transmit lost packets
- what value of Time-out?



- Time-out $\geq$ Typical RTT

?? -> who will give us this?

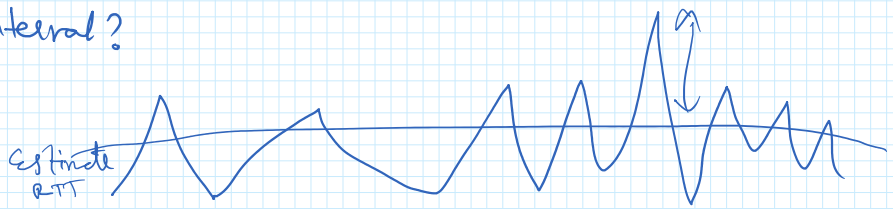
- $SampleRTT_0 - P_0$
  $SampleRTT_1 - P_1$
- $EstimateRTT_t = W_t\, SampleRTT_t + W_{t-1}\, SampleRTT_{t-1} + \dots$

$= \alpha\, SampleRTT + \alpha^2\, SampleRTT_{t-1} + \alpha^3 SR_{t-2} \dots$

($\alpha \to 1/8$, $\alpha^2 \to (1/8)^2$)

$= \alpha\, SampleRTT_t + (1-\alpha)\, EstimateRTT_{t-1}$

Time-out interval?



Time-out = EstimateRTT + Buffer

Buffer → 4 · StDevRTT → $|EstimateRTT - SampleRTT|$

StdDev.

TCP Reliable Data Transport

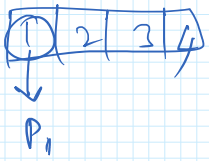
✓ cum. Ack ✓

'merits' to understand when to re-transmit ✓

— Time[illegible]

~~needs~~ to understand when to re-transmit ✓

— Timer associated with the oldest unAck pckt. ✓

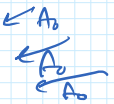


Till now, it looks like GoBackN.

But, It will differ from goback n in the following aspects.

→ Timeout occurs: Re-transmit only the first segment in window

✓ → Some versions of TCP will re-transmit upon receiving some duplicate Ack



→ No particular information about what TCP should do at Rx if out-of-order Pckts come.

✓ → MOST TCP version will store out-of-order Pckts in a buffer.

Simple version of TCP → No Flow Control / Cong. Control
→ No re-transmission for dupAcks

Sender TCP

① — App sends msg
— seq# = 1st byte of the data in that Segment ✓
— byte-out of the
— NEXTSEQNUM = NEXTSEQNUM + len(current segment)

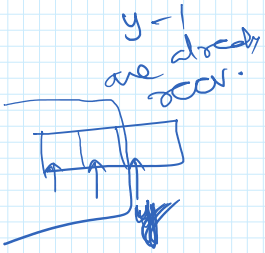
NEXTSEQ = 90
App msg has 20 bytes of data
NEXTSEQ = 90 + 20

② Time-out occur
— Re-transmit the first pckt in window
= the oldest unAck pct.

- Restart timer.

③ Ack Received (y is seq. num of this Ack)

- If 'y' is already ack before
  {
  Do nothing
  }
- If 'y' corresponds to some unack pckt,
  {
  - update window accordingly.
  - Start timer if there are some unack Packets.
  }



State Diagram

(Picture from Book slide)

TCP sender (simplified)