TCP Header

20 Bytes is fixed

Size				
4 Bytes	Source Port (16 bits)		Destination Port (16 bits)	
4 Bytes	Sequence No. (32 bits)			
4 Bytes	Acknowledgement No. (32 bits)			
4 Bytes	HLEN (4 bits)	Reserved (6 bits)	Flags (6 bits)	Window / Advertisement Size (16 bits)
	Checksum (16 bits)			Urgent Pointer (16 bits)
(optional) 0 to 40 Bytes	Options (0 to 40 Bytes)			

Includes the TCP header, TCP data AND IP pseudo-header. (see UDP notes to know more)

- **HLEN** : (4 bits)
  - Header length. Max. no. can be $2^4-1 = 15$. But max. header size = 60. So, Scaling factor = 4 (Read similar to IPv4 header)

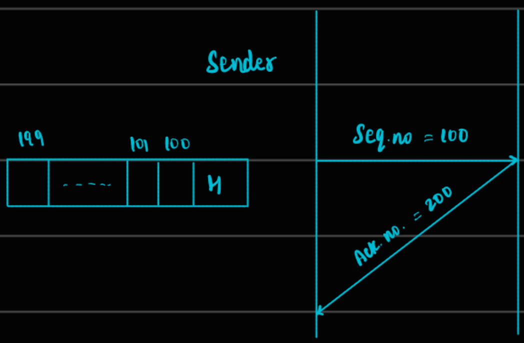
- **Port numbers** : (16 bits)
  - Assigned to a specific process. Well known port nos. from 0 to 1023 is assigned and controlled by IANA (Internet Assigned Numbers Authority). For eg.
    SMTP → 25 ; HTTP → 80 ; FTP → 20 and 21 ; DNS → 53
  - Port nos. from 1024 to 49151 are neither assigned or controlled by IANA. These are called registered or reserved port nos.
  - Port nos. from 49152 to 65,535 are freely available for a process to use. These are called dynamic port nos.

- **Sequence No.** : (32 bits)
  - TCP is a byte-stream protocol. So, every byte is associated with a sequence no.
  - This field defines the sequence no. of the FIRST DATA BYTE

- **Acknowledgement No** : (32 bits)
  - This field defines the seq. no. of the NEXT EXPECTED BYTE.
  - If the receiver has successfully received byte no. x from the

other party (sender), it returns byte no. (n+1) as the ack. no.

- Eg.



- There is a slight problem now. How does the receiver know that 199 is the last byte's seq. no? We only have HLEN in the header not total length. <u>Solution</u>: use total length from network layer

So, $\boxed{\text{Data length at TL} = \text{Total length (NL)} - \text{HLEN (NL)} - \text{HLEN (TL)}}$

- Eg. ① HLEN (IP) = 10 ; TL (IP) = 1000 and HLEN (TCP) = 5 ;
  Seq. no = 100 ; Ack. no. = ?

[Scaling factor of HLEN in both TCP and IP header = 4]

DL (TL) = TL (IP) - HLEN (IP) - HLEN (TCP)

= 1000 - 10(4) - 5(4) = 940

Last seq. byte no. = Seq. no. + DL (TL) - 1 = 1039

Ack. no. = Seq. no. + DL (TL) = 1040

- <u>Wrap Around Time</u>:

- Max. no. of seq. nos. $= 2^{32}$
- Max. size of data that can be transmitted $= 2^{32}$ bytes $=$ 4GB
- If data size > 4GB, we wrap the seq. no. around
- NOTE: Seq. no. is a TCP NEVER starts from 0. It is always randomly assigned

- Wrap around time (WAT) is the time taken for the seq. nos. to wrap around. WAT depends on Bandwidth. If bandwidth is given, make sure to convert it into Bytes since we are dealing with Bytes of data not bits

- Eg. ① $B = 1$MBps ; WAT = ?

  1MBps $= 10^6$ Bps

  10^6 B $\rightarrow$ 1sec

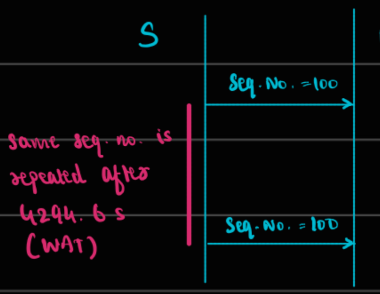
  $\Rightarrow$ 10^6 seq. no $\rightarrow$ 1 sec

  2^{32} seq. no $\rightarrow$ $2^{32}/10^6 = \boxed{4294.6 \text{ secs}}$

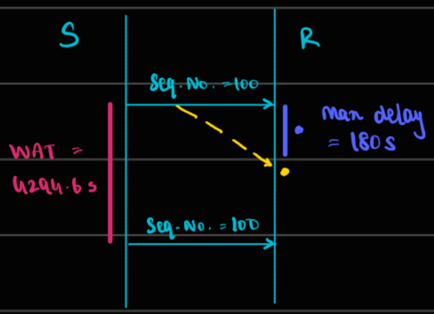
  ② $B = 1$GBps ; WAT = 4.294 secs

NOTE:
- lifetime of a packet in a network = 180 s = 3 mins.
- The packet is discarded from the network after it reaches its lifetime. network here includes all the routes the packet takes. So, regardless of where the packet is, if it exceeds the lifetime, that packet is discarded

So, in eg. ①, WAT = 4294.6s > lifetime

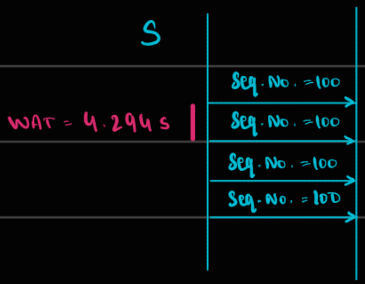


but say the 1st packet with seq. no. = 100 is delayed

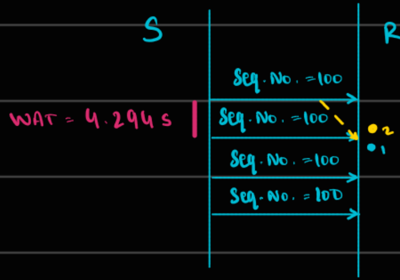


- → Delayed packet will reach after a MAXIMUM OF 180s. If it exceeds the lifetime, it is DISCARDED anyway. So, different packet with the same seq. no CAN NOT be received by the receiver

But in eg. ②, WAT = 4.294 s < lifetime



but say the 1st packet with seq. no. = 100 is delayed



- → In this case, receiver receives a different packet with the same seq. no. So, receiver accepts •1 and discards •2 as it considers •2 as a duplicate.

- Solution: Have more seq. nos. so that they don't repeat during the lifetime of 180 secs.
  (In this case, we don't want to compromise the bandwidth

⇒ min. no. of seq. nos. required to avoid WAT in the lifetime = lifetime × Bandwidth

⇒ min. no. of bits required in seq. no. = $\lceil \log_2(\text{lifetime} \times \text{Bandwidth}) \rceil$
" "

So, in eg. ②, min. no. of seq. no. bits = $\lceil \log_2(180 \times 10^9) \rceil$ = 38 bits

Eg. ① B = 200 mbps and seq. no. bits = 28 bits. WAT = ?

$B = 200 \times 10^6 / 8 = 25 \times 10^6$ Bps

25×10^6 Bytes ⟶ 1s

2^{28} Bytes ⟶ $2^{28} / 25 \times 10^6$ s = 10.73 s

$$WAT = \frac{\text{Total seq. no}}{\text{Bandwidth}}$$; Bandwidth is in Bytes per seconds

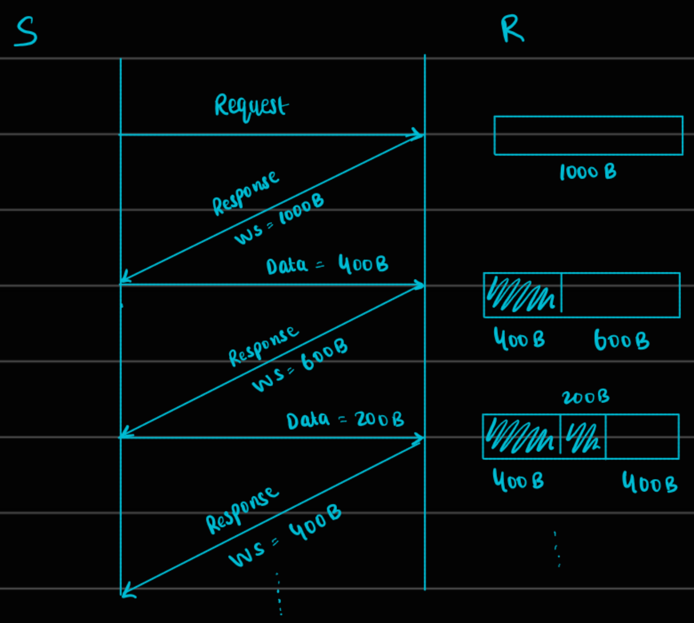
② B = 1Gbps ; 1^{st} seq. no. = 1234 ; WAT = ? (rounded to closest integer)

$$WAT = \frac{\text{Total seq. nos.}}{B \text{ (in Bps)}} = \frac{2^{32}}{10^9/8} \text{ (default no. of bits in seq. no. field)} = 34.35s = 34s$$

③ B = 1Gbps ; maximum segment lifetime (MSL) = 60s ; min. no. of bits to prevent wrap around during the MSL = ?

$$\text{min. no. of seq. no. bits required} = \lceil \log_2(\text{lifetime} \times B) \rceil = \lceil \log_2(10^9 \times 60/8) \rceil$$
$$= \lceil 32.8 \rceil = 33$$

- Window or Acknowledgement Size : (16 bits)
  - Is used for flow control. It is set by the receiver to indicate how much data it can receive from the sender
  - Sender NEVER sends more data than the window size. For eg.



TCP Congestion Control

- **How Congestion Occurs:**

    - A packet from sender may pass through several routers before reaching its final destination. A router has a buffer that stores the incoming packets, processes them and forwards them.
    - If a router receives a packet faster than it can process, congestion might occur and some packets could be dropped.
    - When a packet does not reach its destination, no ACK is sent for it. The sender has no choice but to re-transmit the lost packet. This may create more congestion and more dropping of packets, which means more re-transmission and more congestion.
    - A point maybe reached where the whole system collapses and no more data can be sent. TCP, therefore needs to find some way to avoid this situation.

- **Congestion Window:**

    - In TCP, senders window size is not only determined by the receiver but also by congestion in the network
    - The sender has 2 pieces of information: Receiver advertisement (receiver's window size) and congestion window size

    - Actual window size = min (congestion window size, receiver window size)

$$\Rightarrow \quad W_S = \min(W_C, W_R)$$

- Finding W_C and W_S: Exponential increase from 1 to $W_R/2$. After reaching $W_R/2$, additive increase: +1, +2, ...

Eg. $W_R = 1024$:

$W_C = 1$: send 1 packet to network. Ack received $\Leftrightarrow$ W_C is atleast 1

$\Rightarrow W_S = \min(1, 1024)$

$W_C = 2$ send 2 packets to network. Ack received $\Leftrightarrow$ W_C is atleast 2

$\Rightarrow W_S = \min(2, 1024) = 2$

$W_C = 4$ send 4 packets to network. Ack received $\Leftrightarrow$ W_C is atleast 4

$\Rightarrow W_S = \min(4, 1024) = 4$

⋮

$W_C = 512$: send 512 packets to network. Ack received $\Leftrightarrow$ W_C is atleast 512

$\Rightarrow W_S = \min(512, 1024) = 512$

$W_C = 513$: send 513 packets to network. Ack received $\Leftrightarrow$ W_C is atleast 513

$\Rightarrow W_S = \min(513, 1024) = 513$

$W_C = 514$: send 514 packets to network. Ack received $\Leftrightarrow$ W_C is atleast 514

$\Rightarrow W_S = \min(514, 1024) = 514$

⋮

- Congestion Control Algorithm: (It has 3 phases)

• 1 Slow Start : • 1 After 1 RTT, $W_c = 2 * W_c$

(Exp. increase) • 2 If an ACK arrives then $W_c = W_c + 1$ segment

• 2 Congestion Avoidance : • 1 After 1 RTT, $W_c = W_c + 1$ segment

(Additive increase) • 2 If an ACK arrives then $W_c = W_c + (1/W_c)$

• 3 Congestion Detection : • 1 Timeout Timer : Indicates severe congestion condition as

(multiplicative decrease) packets are getting dropped. In this case, new threshold value is set to HALF THE CURRENT WINDOW SIZE and the next transmission STARTS FROM 1st SEGMENT and the algorithm enters the slow start phase.

1st segment means the 1st value in the W_c progression. BUT start the new threshold value from 1.

• 2 3 duplicate ACKs : Indicates mild congestion condition. In this case, new threshold value is set to HALF THE CURRENT WINDOW SIZE and next transmission starts from new threshold and the algorithm enters into congestion avoidance phase

Eg. ① W_R = 128 KB ; Maximum Segment Size (MSS) = 1 KB ; Progression of W_c = ?

No. of segments = 128 / 1 = 128

Threshold = 128/2 = 64 segments (not $W_R/2$. Another form basically)

W_c = 1, 2, 4, 8, 16, 32, 64, 65, 66, ..., 128, 128, . .

② Same as ① but there are 3 duplicate ACKs at 68.

W_c = 1, 2, 4, 8, ..., 64, 65, 66, 67, $\boxed{68, 68, 68}$ → 3 duplicate ACKs

⇒ W_c = 68/2 = 34

So, $W_C = 1, 2, 4, 8, .., 64, 65, 66, 67, 68, 68, 68, 34, 35, 36, 37, 38, ...$

③ Same as ② but timeout timer is reached here

$$W_C = \overbrace{1, 2, 4, 8, .., 64, 65, 66, 67, 68, 68, 68, 34, 35, 36, 37}^{n}, 38, ...$$

Timeout timer ⇒ Start from 1

$W_C = n, 38, 1, 2, 4, 8, 16, 19$ [not 17, 18, 19. DIRECTLY goto $W_R/2$ since the next value $(32) > W_R/2$]

④ In slow start phase ; $W_C = 4$ mss ; sender gets 4 successful Ack of segments (no outstanding Acks) ; New W_C = ?

Slow start : On successful ACK , $W_C = W_C + 1$

⇒ New $W_C = 4 + 4 = 8$ mss

⑤ W_R = 16000 Bytes ; mss = 1000 Bytes ; How many RTTs will the sender need to send the full window ?

No. of segments = $16000/1000 = 16$

Threshold = $16/2 = 8$ segments (not $W_R/2$. Another form basically)

Progression of W_C : 1, 2, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16

Total no. of RTTs = 11 (DO NOT COUNT 16 obviously)

⑥ RTT = 10ms ; W_R = 24KB ; mss = 2KB ; How long does it take before the 1st full window can be sent ?

No. of segments = $24/2 = 12$ → Full window size in terms of segment

Threshold (TH) = $12/2$ = 6 segments

W_c : 1, 2, 4, 6 (TH), 7, 8, 9, 10, 11, 12 ⇒ No. of RTTs = 9

∴ Answer = 90 ms

OR (TH in terms of W_R)

$TH = W_R/2 = 24/2 = 12$

W_c : 2KB 4KB 8KB 12KB (TH) 14KB 16KB 18KB 20KB 22KB 24KB

⇒ No. of RTTs = 9 . ∴ Answer = 90ms

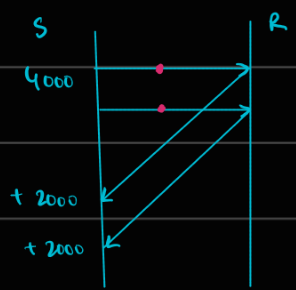
(7) W_c = 4KB ; W_R = 6KB ; last byte sent by the sender = 10240 ; last byte acknowledged by receiver = 8192 ; Current W_s = ?

$W_s = \min(W_c, W_R)$ = 4 KB = 4096 Bytes

(8) Packet size = 2000 Bytes ; In slow start phase, the current transmit window (W_s) = 4000 Bytes and transmitter receives 2 ACKs ; No packets are lost or timeout. Max. transmit window size (W_c) = 12000 Bytes. Max. possible value of current transmit window ?

Diagramatically, it makes sense but how can we substitute W_s in place of W_c in the formula ?

$\boxed{?}$

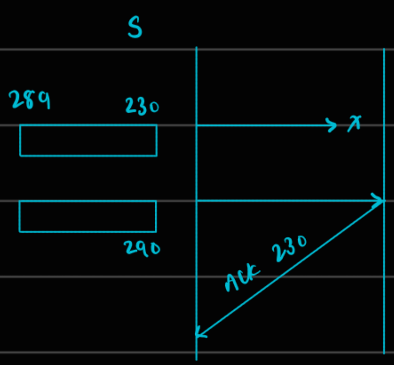


• → 4000 Bytes. So it means transmitter already sent 2 packets (as confirmed by the 2 ACKs)

Slow Start : After 1 ACK , $W_c = W_c + 1$ segment

⇒ $W_s = (4000 + 2000) + (2000)$ = 8000 Bytes

(9) Sender sends 2 segments back to back ; seq. no. of 1^{st} and 2^{nd} segments are 230 and 290 resp. ; 1^{st} segment was lost but 2^{nd} segment was received by the receiver ; No outstanding Acks ; X = data carried in the 1^{st} segment and Y = Ack no. sent by the receiver ; X, Y = ?



TCP can receive out of order packets but ACKS are ALWAYS IN ORDER

Clearly, $X = 290 - 230 = 60$

$Y = 230$

(10) Slow start phase : $W_c = 2$ mss and $TH = 8$ mss ; Timeout occurs at 5^{th} transmission. W_c at the end of 10^{th} transmission = ?

		TH/2	TH		
W_c :	2	4	8	9	10 ...
	1	2	3	4	5

Slow start : Timeout timer $\Rightarrow W_c = W_c + 1$ segment

Confusion : Should we take 1 or 2 as "1 segment"

Solution : ALWAYS TAKE IT AS 1

		TH/2	TH					TH/2				
$\Rightarrow W_c$:	2	4	8	9	10	1	2	4	5	6	7	8

$\therefore$ Answer = 7

(11) RTT = 100ms ; W_c = 32 KB when timeout occurs ; MSS = 2KB ; Time taken by the TCP connection to get back to 32 KB congestion window = ?

No. of segments = $32/2 = 16$; For RTT, we need 16 segments

$TH = No.\ of\ segments / 2 = 16/2 = 8$

			$TH/2$	TH									
W_c :	1	2	4	8	9	10	11	12	13	14	15	16	⇒ 11 RTTs required

∴ Answer = 1100 ms

(12) RTT = 6ms ; W_R = 50 KB ; TH = 32 KB ; Connection established at t = 0 ms ; No timeouts or errors during transmission ; W_c at the time (t + 60) ms after all the ACKs are processed = ? (t = 0 : slow start phase and mss = 2 KB)

(+2KB NOT +1KB since 1st segment is 2 KB)

				$TH/2$	TH						
W_c :	2KB	4KB	8KB	16KB	32KB	34KB	36KB	38KB	40KB	42KB	44KB
Time taken :	t+0	t+6	t+12	t+18	t+24	t+30	t+36	t+42	t+48	t+54	t+60

∴ Answer = 44KB

(13) W_c = 18KB and timeout occurs ; mss = 1KB ; Next 4 bursts are successful ; How big will be the window size in the 5th transmission ?

No. of segments = 18/1 = 18 ; TH = 18/2 = 9

Slow start : Timeout occurs : $W_c = W_c + 1$ segment

		1	2	3	4	5
W_c :	18KB	1KB	2 KB	4 KB	8 KB	9 KB

Timeout occurs

∴ Answer = 9 KB

In these type of questions, make sure to read it properly. Sometimes, it is AT timestep or AFTER timestep, etc.