Introduction to Computer Networks

Lab3 – Congestion Control (Tahoe)

111062109 黃庭曜

Contents

§1 Code explanation

Q1: How does the server send the packet with the correct sequence number to the client?
A1: The server sends packets with sequence numbers [`last_ack`, `last_ack+cwnd`). So I used a for-loop to send.

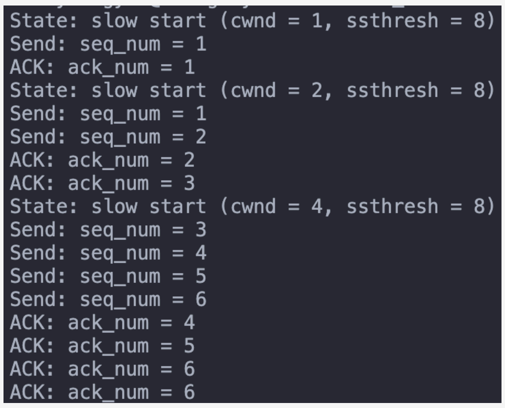
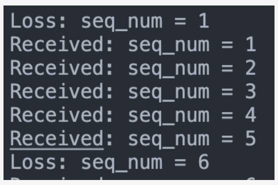
Q2: How to simulate packet loss?
A2: I used the function `packet_loss()`, which is provided. It's a random-based function that generates a floating point number ranging from 0 to 1, and if the number is larger than the experimental loss rate (here, 0.1), the function outputs false (packet not loss), else it outputs true (packet loss).

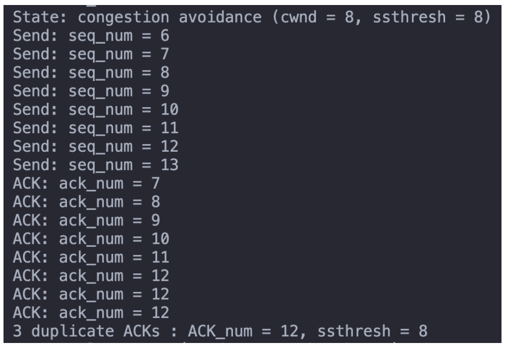
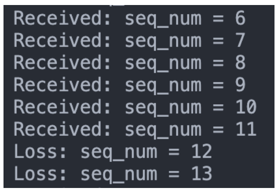
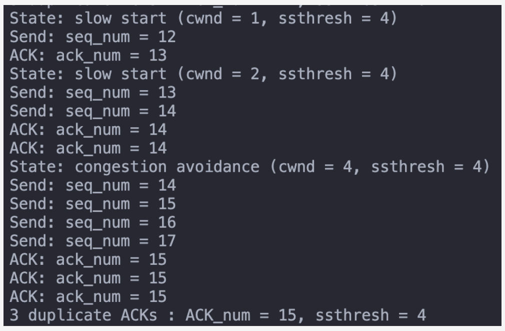
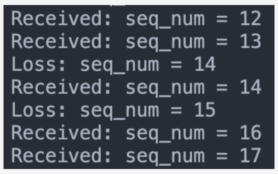
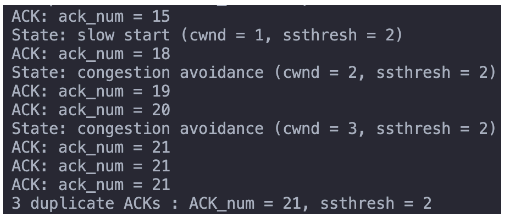
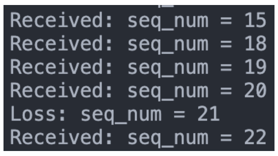
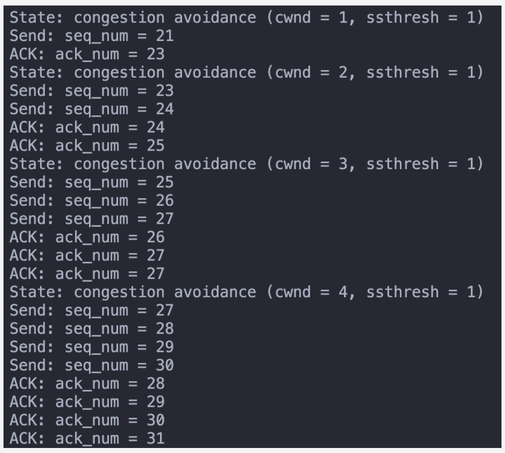
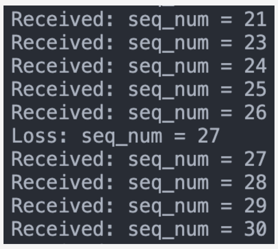
Q3: How to detect 3-duplicate ACKs?
A3: The integer "`last_ack`" indicates the most recent ack number received. If the client sends a packet with ack number equal to "`last_ack`", the "`dup_ack_count`" integer will be incremented. If the "`dup_ack_count`" reaches 3, 3-duplicate ACKs are detected. If the client sends a packet with ack different than "`last_ack`" (If there are no out-of-order packets, the received ack would always be larger than or equal to "`last_ack`"), the "`last_ack`" will be overwritten, and the "`dup_ack_count`" will be reset to 1.

Q4: How to update `cwnd` and `ssthresh`?
A4: If there are three duplicate ACKs, the `cwnd` will be reset to 1 in TCP Tahoe, and `ssthresh` will be set to half the previous value of `cwnd`. In normal conditions, The `cwnd` will be doubled if the final value of `cwnd` will not exceed `ssthresh` (slow start). If the `cwnd` isn't larger than the value of `ssthresh`, but `cwnd`×2 will, then `cwnd` will be set to `ssthresh`. If the value of `cwnd` is already larger than or equal to `ssthresh`, increment `cwnd` by 1 (congestion avoidance).

§2 Screenshot compiled results and explain

Server side	Client Side	Explanation:


`State: slow start (cwnd = 1, ssthresh = 8)`
`Send: seq_num = 1`
`ACK: ack_num = 1`
`State: slow start (cwnd = 2, ssthresh = 8)`
`Send: seq_num = 1`
`Send: seq_num = 2`
`ACK: ack_num = 2`
`ACK: ack_num = 3`
`State: slow start (cwnd = 4, ssthresh = 8)`
`Send: seq_num = 3`
`Send: seq_num = 4`
`Send: seq_num = 5`
`Send: seq_num = 6`
`ACK: ack_num = 4`
`ACK: ack_num = 5`
`ACK: ack_num = 6`
`ACK: ack_num = 6`
	
	`Loss: seq_num = 1`
	`Received: seq_num = 1`
	`Received: seq_num = 2`
	`Received: seq_num = 3`
	`Received: seq_num = 4`
	`Received: seq_num = 5`
	`Loss: seq_num = 6`
		Packets with sequence numbers 1~5 are successfully transmitted. Although the packet with sequence number 6 is lost, it doesn't trigger 3-duplicate ACKs since there's only 2 duplicate ACKs.

 State: congestion avoidance (cwnd = 8, ssthresh = 8) Send: seq_num = 6 Send: seq_num = 7 Send: seq_num = 8 Send: seq_num = 9 Send: seq_num = 10 Send: seq_num = 11 Send: seq_num = 12 Send: seq_num = 13 ACK: ack_num = 7 ACK: ack_num = 8 ACK: ack_num = 9 ACK: ack_num = 10 ACK: ack_num = 11 ACK: ack_num = 12 ACK: ack_num = 12 ACK: ack_num = 12 3 duplicate ACKs : ACK_num = 12, ssthresh = 8	 Received: seq_num = 6 Received: seq_num = 7 Received: seq_num = 8 Received: seq_num = 9 Received: seq_num = 10 Received: seq_num = 11 Loss: seq_num = 12 Loss: seq_num = 13	The packet 6 is retransmitted successfully. However, packet 12 is lost and it successfully triggered 3-duplicate ACKs with ACK_num = 12.
 State: slow start (cwnd = 1, ssthresh = 4) Send: seq_num = 12 ACK: ack_num = 13 State: slow start (cwnd = 2, ssthresh = 4) Send: seq_num = 13 Send: seq_num = 14 ACK: ack_num = 14 ACK: ack_num = 14 State: congestion avoidance (cwnd = 4, ssthresh = 4) Send: seq_num = 14 Send: seq_num = 15 Send: seq_num = 16 Send: seq_num = 17 ACK: ack_num = 15 ACK: ack_num = 15 ACK: ack_num = 15 3 duplicate ACKs : ACK_num = 15, ssthresh = 4	 Received: seq_num = 12 Received: seq_num = 13 Loss: seq_num = 14 Received: seq_num = 14 Loss: seq_num = 15 Received: seq_num = 16 Received: seq_num = 17	The packet 12 & 13 are retransmitted successfully in different rounds due to cwnd reset. Packet 14 is lost at first but retransmitted successfully without triggering 3-duplicate ACKs. Unfortunately, packet 15 is lost and it triggered 3-duplicate ACKs with ACK_num = 15.
 ACK: ack_num = 15 State: slow start (cwnd = 1, ssthresh = 2) ACK: ack_num = 18 State: congestion avoidance (cwnd = 2, ssthresh = 2) ACK: ack_num = 19 ACK: ack_num = 20 State: congestion avoidance (cwnd = 3, ssthresh = 2) ACK: ack_num = 21 ACK: ack_num = 21 ACK: ack_num = 21 3 duplicate ACKs : ACK_num = 21, ssthresh = 2	 Received: seq_num = 15 Received: seq_num = 18 Received: seq_num = 19 Received: seq_num = 20 Loss: seq_num = 21 Received: seq_num = 22	The packet 15 is retransmitted successfully. Since the packets 16 & 17 are successfully sent, the received ack is 18. The packet 21 is lost and it triggered 3-duplicate ACKs with ACK_num = 21.
 State: congestion avoidance (cwnd = 1, ssthresh = 1) Send: seq_num = 21 ACK: ack_num = 23 State: congestion avoidance (cwnd = 2, ssthresh = 1) Send: seq_num = 23 Send: seq_num = 24 ACK: ack_num = 24 ACK: ack_num = 25 State: congestion avoidance (cwnd = 3, ssthresh = 1) Send: seq_num = 25 Send: seq_num = 26 Send: seq_num = 27 ACK: ack_num = 26 ACK: ack_num = 27 ACK: ack_num = 27 State: congestion avoidance (cwnd = 4, ssthresh = 1) Send: seq_num = 27 Send: seq_num = 28 Send: seq_num = 29 Send: seq_num = 30 ACK: ack_num = 28 ACK: ack_num = 29 ACK: ack_num = 30 ACK: ack_num = 31	 Received: seq_num = 21 Received: seq_num = 23 Received: seq_num = 24 Received: seq_num = 25 Received: seq_num = 26 Loss: seq_num = 27 Received: seq_num = 27 Received: seq_num = 28 Received: seq_num = 29 Received: seq_num = 30	The packet 21 is retransmitted successfully. Although the packet 27 is lost at first, it is retransmitted successfully without triggering 3-duplicate ACKs. The other packets are transmitted without lost.