Problem 1

Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.

Answer:

Host A sends data into the TCP receiver buffer at a rate as high as 120 Mbps. And Host B removes data from the TCP receive buffer at a rate of 50 Mbps, which is slower than the former.

Hence, the receive buffer begin to fill up at a rate of about 50 Mbps.

And once the buffer has completely filled, Host B will send a message to Host A to stop sending data until Host B can remove data from the buffer.

Host B will then send a TCP segment to Host A, informing it to continue sending data.

The buffer will fill up again and this process will repeat until all of the data has been sent from Host A to Host B.

Problem 2

Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting AIAD algorithm converge to an equal share algorithm? Justify your answer using a graphical diagram similar to Slide 119 of the lecture (Version 4/29/2021).

Answer:

Suppose that instead of multiplicative decrease, TP decreased the window size by a constant amount.
Then the result diagram is as follows:

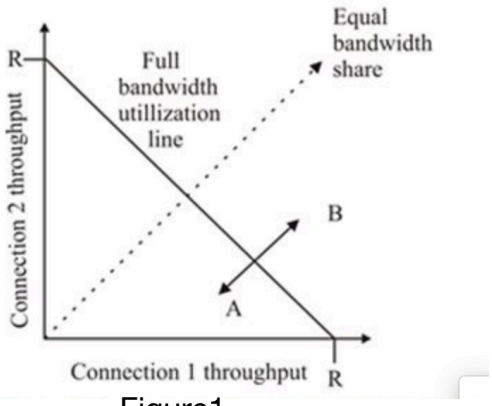


Figure1

The ratio of the linear decrease on loss between connection 1 and 2 is the same. (The loss ratior of the linear decrease is 1, as well as the loss ratior of the linear increase)

Therefore, in this case, the throughputs never move off of the AB line segment

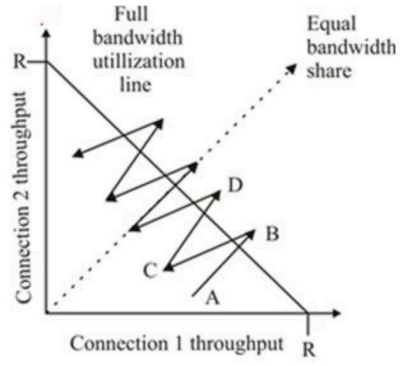


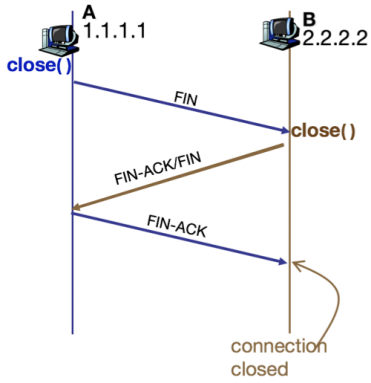
Figure2

In figure 2, the ratio of the linear decrease on loss between connection 1 and connection 2 is 2:1.
Hence, whenever there is a loss, connection 1 decreases its window by twice the amount of connection 2.
The full link bandwidth is 2.
In the end, after enough losses, and subsequent increases, that connection 1's throughput will go to 0, and the full link bandwidth will be allocated to connection 2.

So If TCP decreased the window size by a constant amount, the resulting AIAD algorithm will not converge to an equal share algorithm.

Problem 3

A sends a TCP FIN message to B to close the TCP connection with B, the TCP header of A's FIN message is shown below. When B receives A's TCP FIN, it also decides to close the connection, so B sends a combined FIN and FIN-ACK message, whose TCP header is also shown below. Please fill in all the fields with a question mark in this TCP header.



... src 1.1.1.1, dst: 2.2.2.2								
s_port: 2008							d_port: 5670	
seq_no: 980								
ack_no: 3120								
header length	not used	0	1	0	0	0	1	rcv_window: 200
checksum: ...								0 (ignore this field)

A → 1

... src 2.2.2.2, dst: 1.1.1.1								
s_port: ?							d_port: ?	
seq_no: ?								
ack_no: ?								
header length	not used	0	?	0	0	?	?	rcv_window: 400
checksum: ...								0 (ignore this field)

B → A

Answer:

... src 2.2.2.2								dst 1.1.1.1
s port: 5670							d port: 2008	
seq no: 3120								
ack no: 981								
header length	not used	0	1	0	0	0	1	rcv window: 400
checksum: ...								0 (ignore this field)

Problem 4

Suppose that three measured SampleRTT values are 106 ms, 120 ms, and 150 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, assuming that the value of EstimatedRTT was 100 ms just before the first of these three samples were obtained. Compute also the DevRTT after each sample is obtained, assuming the value of DevRTT was 5 ms just before the first of these three samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained. Round your calculation results to two decimals (e.g., 123.45).

Answer:

Typically, $\alpha = 0.125$, $\beta = 0.25$

1st:

$EstimatedRTT = \alpha * SampleRTT + (1-\alpha) * EstimatedRTT = 0.125 * 106 + (1-0.125) \times 100 = 100.75\ ms$

$DevRTT = (1-\beta) * DevRTT + \beta * |SampleRTT - EstimatedRTT| \approx 5.06\ ms$

$TimeoutInterval = EstimatedRTT + 4 * DevRTT = 100.75 + 4 * 5.06 = 120.99\ ms$

2nd:

$EstimatedRTT = 0.125 * 120 + (1-0.125) * 100.75 = 103.16\ ms$

$DevRTT = 0.75 * 5.06 + 0.25 * |120 - 103.16| = 8.01\ ms$

$TimeoutInterval = 103.16 + 8.01 * 4 = 135.20\ ms$

3rd:

$EstimatedRTT = 0.125 * 150 + 0.875 * 103.15 \approx 109.02\ ms$

$DevRTT = 0.75 * 8.01 + 0.25 * |150 - 109.02| \approx 16.25\ ms$

$TimeoutInterval = 109.02 + 4 * 16.25 \approx 174.02\ ms$

Problem 5

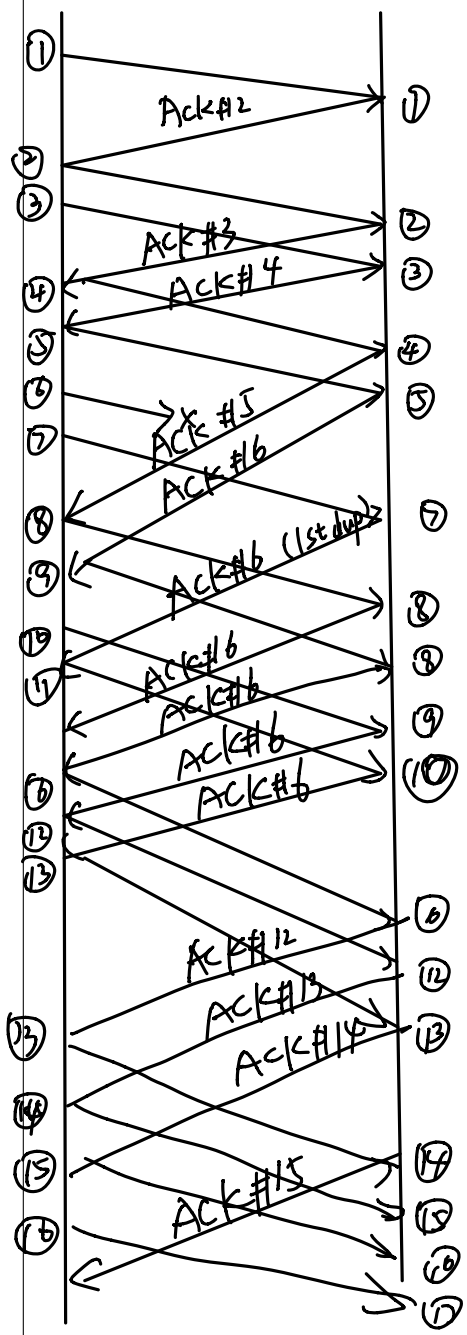
Consider the evolution of a TCP connection with the following characteristics. Assume that all the following algorithms are implemented in TCP congestion control: slow start, congestions avoidance, fast retransmit and fast recovery, and retransmission upon timeout. If ssthresh equals to cwnd, use the slow start algorithm in your calculation.

- The TCP receiver acknowledges every segment in cumulative way, and the sender always has data segments available for transmission.
- The RTT is 100 ms for all transmissions, consists of the network latency of 60 ms in sending a segment (header and payload) from the sender to the receiver and 40 ms in sending an acknowledgment (header only) from the receiver to the sender. Ignore packet-processing delays at the sender and the receiver.
- Initially ssthresh at the sender is set to 5. Assume cwnd and ssthresh are measured in segments, and the transmission time for each segment is negligible.
- Retransmission timeout (RTO) is initially set to 500ms at the sender and is unchanged during the connection lifetime.
- The connection starts to transmit data at time t = 0, and the initial sequence number starts from 1. TCP segment with sequence number 6 is lost once (i.e., it sees segment loss during its first transmission). No other segments are lost during transmissions.

What are the values for cwnd and ssthreshold when the sender receives the TCP ACK with number 15? Show your intermediate steps or your diagram in your solution.

Problem 5 continued on next page...

Answer:



	cwnd	ssthresh	Alg:
	1	5	slow start
		⋮	⋮
	1+1=2		
	2+1=3		
	3+1=4	⋮	
	4+1=5	5	No action
	5+1=6	(6/2)=3	Fast retransmit (FR & / fast recovery FR)
	⋮		
	3+3=6	3	
	6+1=7		
	7+1=8		
	⋮	⋮	
			FR&FR → slow start.
	3+1=4		
	4	3	Congestion avoidance.

∴ when receive Ack#15, sshthresh=3, cwnd=4