Matthew Irvine

1001401200

Chapter 3 review questions

Section 3.1 – 3.3
R3. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A? (Mark 1)

Destination port = x

Source port = y

R7. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts? (Mark 1)

Yes, it will know the difference by the IP address.

R8. Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain. (Mark 5)

Sent to different sockets but do all have the same destination ip and port number. Basically, the server uses the client ip and port number to demultiplex the message to a particular socket along with using server ip and port number. Therefore, multiple connections can be using the same destination ip and port number, but the messages are sent to different sockets.

Section 3.4
R11.Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0, assuming that packets can be lost? Explain. (Mark 1)

Since the delay is known and packets can be lost then a situation may come where the packet is lost and a ACK is not sent back, therefore the packet would need to be resent thus a timer would be necessary, but we could just set it to how long the roundtrip delay is.

Section 3.5
R14.True or false? (Mark 7)

1. Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgments to Host A because Host B cannot piggyback the acknowledgments on data. false
2. The size of the TCP rwnd never changes throughout the duration of the connection. false

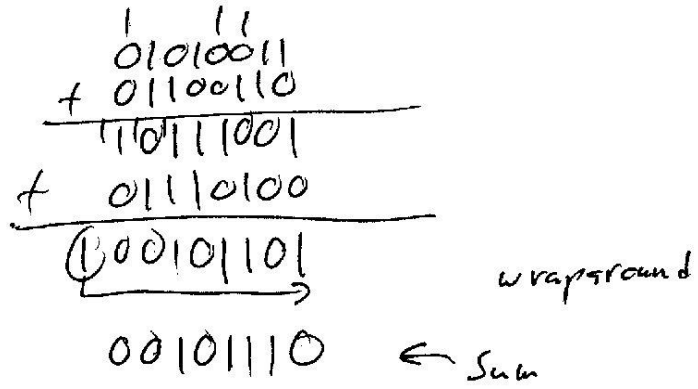
3. Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer. true
4. Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m + 1. true
5. The TCP segment has a field in its header for rwnd. true
6. Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. The current value of TimeoutInterval for the connection will necessarily be ≥ 1 sec. false
7. Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgment number is necessarily 42. false

Section 3.7
R18.True or false? Consider congestion control in TCP. When the timer expires at the sender,

the value of ssthresh is set to one half of its previous value. (Mark 1) false
(one half of cwnd)

Problems

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error? (Marks 5)



1's complement is the inverse of the sum above: 11010001

Because 1's complement of the sum is the same as the checksum which is used to detect errors.

If the receiver observes that the 1's complement of the sum is not equal to the checksum then there is an error.

No, 4 + 5 = 9 regardless of if it is base 10 or binary, thus if I change a single bit the one of these numbers will change while the other stay the same, making the equality false.

Yes, if two bit errors occur then that is like saying 4 + 5 = 9 -> 3 + 6 = 9 which are both true thus the checksum fails to recognize the 2 bit error.

P24.Answer true or false to the following questions and briefly justify your answer: (Marks 8)

1. With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window. True
2. With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window. true
3. The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1. true
4. The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1. true

P40.Consider Figure 3.58. Assuming TCP Reno i

s the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer. (Marks 11)

1. Identify the intervals of time when TCP slow start is operating.
    1. Transmission round = 1-6 and 23-26, we can see window size doubling.
2. Identify the intervals of time when TCP congestion avoidance is operating.
    1. Transmission round = 6 and 17, connection goes from exponential (or decreasing) to linearly increasing.
3. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
    1. Triple duplicate ACK, timeout sends cwnd to 1 and this data point is not equal to 1.
4. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
    1. Timeout, cwnd goes to 1 which is the result of a timeout.
5. What is the initial value of ssthresh at the first transmission round?
    1. ½ cwnd, initial setup starts ssthresh at ½ cwnd (or ½ of MSS, same thing).
6. What is the value of ssthresh at the 18th transmission round?
    1. Equal to cwnd, after dup ACK(at 17) ssthresh = cwnd/2 then after new ACK (at 18) cwnd = ssthresh.
7. What is the value of ssthresh at the 24th transmission round?
    1. ½ cwnd, timeout sets ssthresh = ½ cwnd, then sets cwnd to 1 for slow start.
8. During what transmission round is the 70th segment sent?
    1. 13, round 1 to round 6 have doubles each round for a cumulative 63 segments, add 7 for 70 segments in linear increase, thus 7 rounds + 6 rounds = 13.

9. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?
    1. Ssthresh = ½ cwnd, window size = ssthresh + 3. Ssthresh = 4 and windows size = 7.
10. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?
    1. Window size = 4 ssthresh = 21. Tahoe resets cwnd to 1 and sets ssthresh to half of the current congestion window, thus ssthresh = 21 while cwnd goes to 1 and after 2 rounds of doubling makes it 4.
11. Again, suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?
    1. 24 packets, 17th round starts with cwnd = 1 and doubles each round (never reaches ssthresh) which is 4 rounds of doubling (1->2->4->8->16) then inclusively add the 1 packet sent at round 22 (timeout resets cwnd to 1). Thus, 1+2+4+8+16+1 = 24.

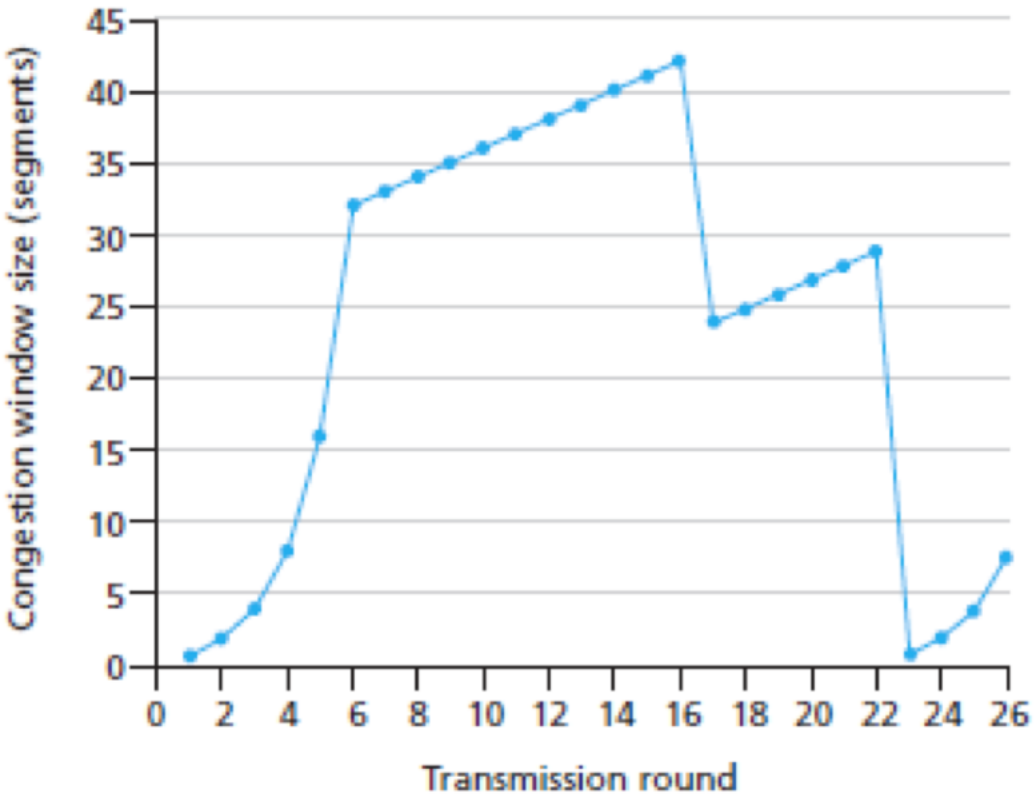


Figure 3.58 – TCP window size as a function of time