Answers to Sample Questions on Transport Layer

1) Which protocol – Go-Back-N or Selective-Repeat - makes more efficient use of network bandwidth? Why?

Answer: Selective repeat makes more efficient use of network bandwidth since it only retransmits those messages lost at the receiver (or prematurely timed out). In Go-Back-N, the sender retransmits the first lost (or prematurely timed out) message as well as all following messages (without regard to whether or not they have been received).

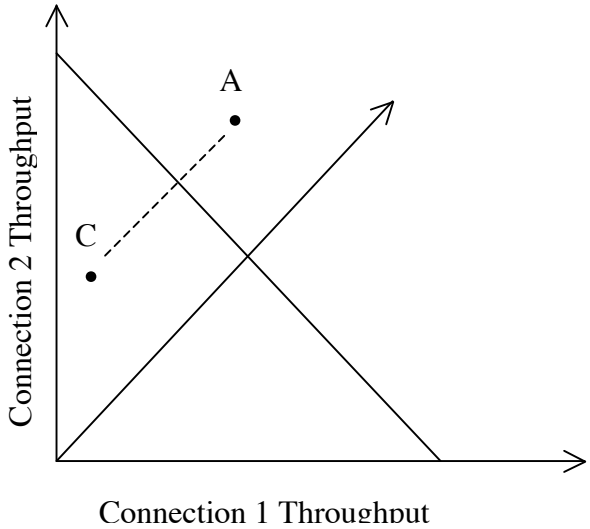
2) Consider a reliable data transfer protocol that uses only negative acknowledgements. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Answer: In a NAK only protocol, the loss of packet *x* is only detected by the receiver when packet *x+1* is received. That is, the receiver receives *x-1* and then *x+1,* only when *x+1 is* received does the receiver realizes that *x* was missed. If there is a long delay between the transmission of x and the transmission of *x+1,* then it will be a long time until *x* can be recovered, under a NAK only protocol.

On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

3) Refer to Figure 3.56 of the textbook (or a similar figure from the Week 5 lecture note) which illustrates the convergence of TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm. Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting AIAD (Additive Increase Additive Decrease) algorithm converge to an equal share algorithm? Justify your answer using a similar figure.

Answer: Refer to figure below**.** Point A is unstable resulting in packet loss. In this case, both the flows will additively decrease their window resulting in movement along the 45-degree line (dotted line in above figure) to point A. Eventually when they reach point C, the sum of the throughputs will be less than the channel capacity and so the two flows will again additively increase their windows resulting in increase in throughput along the same dotted line. Again, this oscillation along the dotted line will continue. Hence fairness will not be achieved and connection 2 will receive an unfairly large share of the link bandwidth. Hence AIAD is not used by TCP.



4) Consider two TCP senders. They are at different sending hosts and go to different destinations, but pass through a common bottleneck link (that is the only bottleneck link on either of their paths). What does is mean to say that TCP provides fair sharing of bandwidth at the bottleneck link? Suppose the RTTs of the two connections are very different. Is TCP "fair" in this case? Justify your answer.

Answer: TCP is fair is that in the long term, each receiver will eventually see the same throughput at that link. The TCP window size grows at a rate that is proportional to the RTT. If the RTT's are different, then the two TCP senders will grow their windows at different rates (even if they start with the same window size). Hence, this could possibly result in unfair sharing of the link.

5) If the RTT from London to Cape Sydney is 120ms and all links in the network have a 155 Mbits/second data-rate, how much data can fit in the "pipe"? Express your answer in bytes.

Answer: 120 ms x 155 Mbits/sec = 18.6 x 10^6 bits = 2,325,000 bytes will fit in the pipe.

6) Consider the GBN and SR protocols. Suppose the sequence number space of size *k*. What is the largest allowable sender size window that will avoid the occurrence of the problems such as that in Figure 3.27 (of the textbook or as discussed towards the end of Week 4 lecture slides).

Answer: In order to avoid the scenario of Figure 3.27, we want to avoid having the leading edge of the receiver's window (i.e., the one with the "highest" sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows.

Suppose that the lowest-sequence number that the receiver is waiting for is packet *m*. In this case, its window is *[m, m+w-1]* and it has received (and ACKed) packet *m-1* and the *w-1* packets before that, where *w* is the size of the window. If the sender has yet received none of those *w* ACKs, then ACK messages with values of *[m-w, m-1]* may still be

propagating back. If the sender has received no ACKs with these ACK numbers, then the sender's window would be *[m-w, m-1]*.

Thus, the lower edge of the sender's window is *m-w*, and the leading edge of the receiver's window is *m+w-1*. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate *2w* sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.

7) Solve Problem 40 from Chapter 3 of the textbook. (Note: If you have an older edition of the textbook, this is the problem concerning the plot of the TCP window size as a function of time and which has (a) – (i) sub-questions)

Answer:

a) TCP slow start is operating in the intervals [1,6] and [23,26]
b) TCP congestion avoidance is operating in the intervals [6,16] and [17,22]
c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.
d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.
e) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.
f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.
g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion windows size is 26. Hence the threshold is 13 during the 24th transmission round.
h) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8-15 are sent in the 4th transmission round; packets 16-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.
i) The threshold will be set to half the current value of the congestion window (8) when the loss occurred and congestion window will be set to the new threshold value + 3 MSS . TCP halves the value of cwnd but adds 3 MSS for good measure to account for triple duplicate ACKs received. Thus the new values of the threshold and window will be 4 and 7 respectively.

8) In protocol rdt 3.0. the ACK packets flowing from the receiver to the sender do not contain any sequence numbers. What do you think is the reason for this?

Answer: To best answer this question, consider why we needed sequence numbers in the first place. We saw that the sender needs sequence numbers so that the receiver can tell if a data packet is a duplicate of an already received data packet. In the case of ACKs, the sender does not need this info (i.e., a sequence number on an ACK) to tell detect a duplicate ACK. A duplicate ACK is obvious to the rdt3.0 receiver, since when it has received the original ACK it transitioned to the next state. The duplicate ACK is not the ACK that the sender needs and hence is ignored by the rdt3.0 sender.

9) Draw the FSM (finite state machine) for the receiver side of protocol rdt 3.0.

Answer: The sender side of protocol rdt3.0 differs from the sender side of protocol 2.2 in that timeouts have been added. We have seen that the introduction of timeouts adds the possibility of duplicate packets into the sender-to-receiver data stream. However, the receiver in protocol rdt.2.2 can already handle duplicate packets. (Receiver-side duplicates in rdt 2.2 would arise if the receiver sent an ACK that was lost, and the sender then retransmitted the old data). Hence the receiver in protocol rdt2.2 will also work as the receiver in protocol rdt 3.0.

10) Two 16-bit words 1011 0101 1010 1000 and 0101 1001 0000 0101 are received, along with another 16-bit word, 1101 0001 0101 0001, which is the UDP checksum of the first two words. Will the receiver detect an error?

Answer: The 1's complement of the sum of the first two words is 0000 1110 1010 1110 and the UDP checksum should be 1111 0001 0101 0001. This is not the same as the received UDP checksum. The receiver will detect an error.

11) Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

Answer: An application developer may not want its application to use TCP's congestion control, which can throttle the application's sending rate at times of congestion. Often, designers of IP telephony and IP videoconference applications choose to run their applications over UDP because they want to avoid TCP's congestion control. Also, some applications do not need the reliable data transfer provided by TCP. If these applications require reliable data transfer, then the application layer protocol will have to provide for reliability.

12) Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110. (EXAM PROBLEM FROM PREVIOUS SEMESTERS)

(a) How much data is in the first segment?

Answer: 20 bytes

(b) Suppose the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number?

Answer: The acknowledgement number will be 90.

13) Consider a TCP connection between Host A and Host B. Suppose that the TCP segments travelling 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 travelling from Host B to Host A?

Answer: Source port number y and destination port number x.

14) Because of the connection-oriented nature of TCP, a connection setup phase is required at the beginning of each session, as well as a connection tear-down phase at the end of the session. Enumerate the events below in the order they occur as host A opens a

TCP connection to host B, transmits data and then closes the connection. Write a 1 next to the event that occurs first and continue like that until all occurring events are enumerated (the first event has been enumerated for you). You may assume that no segments are lost. Also indicate at which host the event happens. Please note that there might be events listed below that are not a part of the above data transfer and hence should not be enumerated. (EXAM PROBLEM FROM PREVIOUS SEMESTERS)

Answer:

Order	Host	Event
8	A	Send an ACK segment
4	A & B	Do the rest of the data exchange
11	A	Close the connection
6	B	Send an ACK segment
5	A	Send a FIN segment
1	A	Send a SYN segment
7	B	Send a FIN segment
		Send a RST segment
2	B	Send a SYN-ACK segment
9	A	Enter the TIME-WAIT state
3	A	Send an ACK+DATA segment
10	B	Close the connection

15) Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely sure that no bit errors have occurred? Explain. Would things be different with TCP?

Answer: No, the receiver cannot be absolutely certain that no bit errors have occurred. This is because of the manner in which the checksum for the packet is calculated. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1 then even if these get flipped to 1 and 0 respectively, the sum still remains the same. Hence, the 1s complement the receiver calculates will also be the same. This means the checksum will verify even if there was transmission error.

Since TCP uses the same checksum mechanism, the above would hold true with TCP as well.

16) Consider the cross-country pipelined RDT example shown in Figure 3.17. How big would the window size have to be for the channel utilisation to be greater than 90%?

Answer: It takes 8 microseconds (or 0.008 milliseconds) to send a packet. in order for the sender to be busy 90 percent of the time, we must have

$$util = 0.9 = (0.008n) / 30.016$$

or n approximately 3377 packets.