
CN Assignment 2 & 3

P1. Suppose Client A requests a web page from Server S through HTTP and its socket is associated with port 16500.

- a. What are the source and destination ports for the segments sent from A to S?
- b. What are the source and destination ports for the segments sent from S to A?
- c. Can Client A contact to Server S using UDP as the transport protocol?
- d. Can Client A request multiple resources in a single TCP connection?

P2. Consider Figure 3.5 in the text book. What are the source and destination port values in the segments flowing from the server back to the clients' processes? What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 10101100, 10011001, 10001011. 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?

P4. Assume that a host receives a UDP segment with 10100010 00001101 (we separated the values of each byte with a space for clarity) as the checksum. The host adds the 16-bit words over all necessary fields excluding the checksum and obtains the value 11001101 11110010. Is the segment considered correctly received or not? What does the receiver do?

P5. 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 certain that no bit errors have occurred? Explain.

P6. Consider our motivation for correcting protocol rdt2.1. Show that the receiver, shown in Figure 3.57 in the textbook, when operating with the sender shown in Figure 3.11 in the textbook, can lead the sender and receiver to enter into a deadlock state, where each is

waiting for an event that will never occur.

P7. In protocol rdt3.0, the ACK packets flowing from the receiver to the sender do not have sequence numbers (although they do have an ACK field that contains the sequence number of the packet they are acknowledging). Why is it that our ACK packets do not require sequence numbers?

P8. Draw the FSM for the receiver side of protocol rdt3.0.

P9. Give a trace of the operation of protocol rdt3.0 when data packets and acknowledgment packets are garbled. Your trace should be similar to that used in Figure 3.16.

P10. Consider a channel that can lose packets but has a maximum delay that is known. Modify protocol rdt2.1 to include sender timeout and retransmit. Informally argue why your protocol can communicate correctly over this channel.

P11. Consider the rdt2.2 receiver in Figure 3.14, and the creation of a new packet in the self-transition (i.e., the transition from the state back to itself) in the Wait-for-0-from-below and the Wait-for-1-from-below states: `sndpkt=make_pkt(ACK,1,checksum)` and `sndpkt=make_pkt(ACK,0,checksum)`. Would the protocol work correctly if this action were removed from the self-transition in the Wait-for-1-from-below state? Justify your answer. What if this event were removed from the self-transition in the Wait-for-0-from-below state? [Hint: In this latter case, consider what would happen if the first sender-to-receiver packet were corrupted.]

P12. The sender side of rdt3.0 simply ignores (that is, takes no action on) all received packets that are either in error or have the wrong value in the acknum field of an acknowledgment packet. Suppose that in such circumstances, rdt3.0 were simply to retransmit the current data packet. Would the protocol still work? (Hint: Consider what would happen if there were only bit errors; there are no packet losses but premature timeouts can occur. Consider how many times the n th packet is sent, in the limit as n approaches infinity.)

P13. Assume Host A is streaming a video from Server B using UDP. Also assume that the network suddenly becomes very congested while Host A is seeing the video. Is there any way to handle this situation with UDP? What about with TCP? Is there any other

option?

P14. Consider a stop-and-wait data-transfer protocol that provides error checking and retransmissions but uses only negative acknowledgments. Assume that negative acknowledgments are never corrupted. Would such a protocol work over a channel with bit errors? What about over a lossy channel with bit errors?

P15. Consider the cross-country example shown in Figure 3.17. How big would the window size have to be for the channel utilization to be greater than 98 percent? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

P16. Suppose an application uses rdt 3.0 as its transport layer protocol. As the stop-and-wait protocol has very low channel utilization (shown in the crosscountry example), the designers of this application let the receiver keep sending back a number (more than two) of alternating ACK 0 and ACK 1 even if the corresponding data have not arrived at the receiver. Would this application design increase the channel utilization? Why? Are there any potential problems with this approach? Explain.

P17. Consider two network entities, A and B, which are connected by a perfect bi-directional channel (i.e., any message sent will be received correctly; the channel will not corrupt, lose, or re-order packets). A and B are to deliver data messages to each other in an alternating manner: First, A must deliver a message to B, then B must deliver a message to A, then A must deliver a message to B and so on. If an entity is in a state where it should not attempt to deliver a message to the other side, and there is an event like `rdt_send(data)` call from above that attempts to pass data down for transmission to the other side, this call from above can simply be ignored with a call to `rdt_unable_to_send(data)`, which informs the higher layer that it is currently not able to send data. [Note: This simplifying assumption is made so you don't have to worry about buffering data.] Draw a FSM specification for this protocol (one FSM for A, and one FSM for B!). Note that you do not have to worry about a reliability mechanism here; the main point of this question is to create a FSM specification that reflects the synchronized behavior of the two entities. You should use the following events and actions that have the same meaning as protocol rdt1.0 in Figure 3.9: `rdt_send(data)`, `packet = make_pkt(data)`, `udt_send(packet)`, `rdt_rcv(packet)`, `extract(packet, data)`, `deliver_data(data)`. Make sure your protocol reflects the strict alternation of sending between A and B. Also, make sure to indicate the initial states for A and B in your FSM descriptions.

P18. In the generic SR protocol that we studied in Section 3.4.4, the sender transmits a message as soon as it is available (if it is in the window) without waiting for an acknowledgment. Suppose now that we want an SR protocol that sends messages two at a time. That is, the sender will send a pair of messages and will send the next pair of messages only when it knows that both messages in the first pair have been received correctly. Suppose that the channel may lose messages but will not corrupt or reorder messages. Design an error-control protocol for the unidirectional reliable transfer of messages. Give an FSM description of the sender and receiver. Describe the format of the packets sent between sender and receiver, and vice versa. If you use any procedure calls other than those in Section 3.4 (for example, `udt_send()`, `start_timer()`, `rdt_rcv()`, and so on), clearly state their actions. Give an example (a timeline trace of sender and receiver) showing how your protocol recovers from a lost packet.

P19. Suppose Host A and Host B use a GBN protocol with window size $N = 5$ and a long-enough range of sequence numbers. Assume Host A sends six application messages to Host B and that all messages are correctly received, except for the first acknowledgment and the fifth data segment. Draw a timing diagram (similar to Figure 3.22), showing the data segments and the acknowledgments sent along with the corresponding sequence and acknowledge numbers, respectively.

P20. Consider a scenario in which Host A and Host B want to send messages to Host C. Hosts A and C are connected by a channel that can lose and corrupt (but not reorder) messages. Hosts B and C are connected by another channel (independent of the channel connecting A and C) with the same properties. The transport layer at Host C should alternate in delivering messages from A and B to the layer above (that is, it should first deliver the data from a packet from A, then the data from a packet from B, and so on). Design a stop-and-wait-like error-control protocol for reliably transferring packets from A and B to C, with alternating delivery at C as described above. Give FSM descriptions of A and C. (Hint: The FSM for B should be essentially the same as for A.) Also, give a description of the packet format(s) used.

P21. Suppose we have two network entities, A and B. B has a supply of data messages that will be sent to A according to the following conventions. When A gets a request from the layer above to get the next data (D) message from B, A must send a request (R) message to B on the A-to-B channel. Only when B receives an R message can it send a data (D) message back to A on the B-to-A channel. A should deliver exactly one copy of each D message to the layer above. R messages can be lost (but not corrupted) in the A-to-B channel; D messages, once sent, are always delivered correctly. The delay along

both channels is unknown and variable. Design (give an FSM description of) a protocol that incorporates the appropriate mechanisms to compensate for the loss-prone A-to-B channel and implements message passing to the layer above at entity A, as discussed above. Use only those mechanisms that are absolutely necessary.

P22. Consider the GBN protocol with a sender window size of 4 and a sequence number range of 1,024. Suppose that at time t , the next in-order packet that the receiver is expecting has a sequence number of k . Assume that the medium does not reorder messages. Answer the following questions:

- a. What are the possible sets of sequence numbers inside the sender's window at time t ? Justify your answer.
- b. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t ? Justify your answer.

P23. Give one example where buffering out-of-order segments would significantly improve the throughput of a GBN protocol.

P24. Consider a scenario where Host A, Host B, and Host C are connected as a ring (i.e., Host A to Host B, Host B to Host C, and Host C to Host A). Assume that Host A and Host C run protocol rdt3.0, while Host B simply relays all messages received from Host A to Host C. Does this arrangement enable reliable delivery of messages from Host A to Host C? Can Host B tell if a certain message has been correctly received by Host A?

P25. Consider the Telnet case study in Section 3.5.2. Assume a Telnet session is already active between Host A and Server S. The user at Host A then types the word "Hello."

- a. How many TCP segments will be created at the transport layer of Host A?
- b. Is there any guarantee that each segment will be sent into the TCP connection as soon as it is created?
- c. Does TCP provide any mechanism that can be useful for an interactive Telnet session?
- d. Would UDP offer a viable alternative to TCP for Telnet sessions over a reliable channel?

P26. Consider transferring an enormous file of L bytes from Host A to Host B. Assume an MSS of 536 bytes.

- a. What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has 4 bytes.
- b. For the L you obtain in (a), find how long it takes to transmit the file. Assume that a

total of 66 bytes of transport, network, and data-link header are added to each segment before the resulting packet is sent out over a 155 Mbps link. Ignore flow control and congestion control so A can pump out the segments back to back and continuously.

P27. Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

- a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?
- b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?
- c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?
- d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.

P28. 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.

P29. SYN cookies were discussed in Section 3.5.6. a. Why is it necessary for the server to use a special initial sequence number in the SYNACK? b. Suppose an attacker knows that a target host uses SYN cookies. Can the attacker create half-open or fully open connections by simply sending an ACK packet to the target? Why or why not? c. Suppose an attacker collects a large amount of initial sequence numbers sent by the server. Can the attacker cause the server to create many fully open connections by sending ACKs with those initial sequence numbers? Why?

P30. Consider the network shown in Scenario 2 in Section 3.6.1. Suppose both sending hosts A and B have some fixed timeout values. a. Argue that increasing the size of the finite buffer of the router might possibly decrease the throughput (lout). b. Now suppose both hosts dynamically adjust their timeout values (like what TCP does) based on the buffering delay at the router. Would increasing the buffer size help to increase the throughput? Why?

P31. Suppose that the five measured SampleRTT values (see Section 3.5.3) are 106 ms, 120 ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained.

Compute also the DevRTT after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

P32. Consider the TCP procedure for estimating RTT. Suppose that $\alpha = 0.1$. Let SampleRTT1 be the most recent sample RTT, let SampleRTT2 be the next most recent sample RTT, and so on.

a. For a given TCP connection, suppose four acknowledgments have been returned with corresponding sample RTTs: SampleRTT4, SampleRTT3, SampleRTT2, and SampleRTT1. Express EstimatedRTT in terms of the four sample RTTs.

b. Generalize your formula for n sample RTTs.

c. For the formula in part (b) let n approach infinity. Comment on why this averaging procedure is called an exponential moving average. P33. In Section 3.5.3, we discussed TCP's estimation of RTT. Why do you think TCP avoids measuring the SampleRTT for retransmitted segments? P34. What is the relationship between the variable SendBase in Section 3.5.4 and the variable LastByteRcvd in Section 3.5.5?

P35. What is the relationship between the variable LastByteRcvd in Section 3.5.5 and the variable y in Section 3.5.4? P36. In Section 3.5.4, we saw that TCP waits until it has received three duplicate ACKs before performing a fast retransmit. Why do you think the

TCP designers chose not to perform a fast retransmit after the first duplicate ACK for a segment is received? P37. Compare GBN, SR, and TCP (no delayed ACK). Assume that the timeout values for all three protocols are sufficiently long such that 5 consecutive data segments and their corresponding ACKs can be received (if not lost in the channel) by the receiving host (Host B) and the sending host (Host A) respectively. Suppose Host A sends 5 data segments to Host B, and the 2nd segment (sent from A) is lost. In the end, all 5 data segments have been correctly received by Host B.

- a. How many segments has Host A sent in total and how many ACKs has Host B sent in total? What are their sequence numbers? Answer this question for all three protocols.
- b. If the timeout values for all three protocols are much longer than 5 RTT, then which protocol successfully delivers all five data segments in shortest time interval?

P38. In our description of TCP in Figure 3.53, the value of the threshold, $ssthresh$, is set as $ssthresh = cwnd/2$ in several places and $ssthresh$ value is referred to as being set to half the window size when a loss event occurred. Must the rate at which the sender is sending when the loss event occurred be approximately equal to $cwnd$ segments per RTT? Explain your answer. If your answer is no, can you suggest a different manner in which $ssthresh$ should be set?

P39. Consider Figure 3.46(b). If $l'in$ increases beyond $R/2$, can $lout$ increase beyond $R/3$? Explain. Now consider Figure 3.46(c). If $l'in$ increases beyond $R/2$, can $lout$ increase beyond $R/4$ under the assumption that a packet will be forwarded twice on average from the router to the receiver? Explain.

P40. Consider Figure 3.58. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

a. Identify the intervals of time when TCP slow start is operating.

b. Identify the intervals of time when TCP congestion avoidance is operating.

c. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

d. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?