Homework Problems and Discussion Questions

Chapter 3

Review Questions

Sections 3.1-3.3

1) 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 travelling from host B to host A?

2) Describe why an application developer may choose to run its application over UDP rather than TCP.

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

Section 3.5

4) True or False:

a) Host A is sending host B a large file over a TCP connection. Assume host B has no data to send A. Host B will not send acknowledgements to host A because B cannot piggyback the acknowledgementson data?

b) The size of the TCP RcvWindow never changes throughout the duration of the connection?

c) 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?

d) 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?

e) The TCP segment has a field in its header for RcvWindow?

f) Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. Then Timeout for the connection will necessarily be set to a value >= 1 sec.

g) Suppose host A sends host B one segment with sequence number 38 and 4 bytes of data. Then in this same segment the acknowledgement number is necessarily 42?

5) Suppose A sends two TCP segments back-to-back to B. The first segment has sequence number 90; the second has sequence number 110. a) How much data is the first segment? b) Suppose that the first segment is lost, but the second segment arrives at B. In the acknowledgement that B sends to A, what will be the acknowledgment number?

6) Consider the Telent example discussed in Section 3.5. A few seconds after the user types the letter 'C' the user types the letter 'R'. After typing the letter 'R' how many segments are sent and what is put in the sequence number and acknowledgement fields of the segments.

Section 3.7

7) Suppose two TCP connections are present over some bottleneck link of rate R bps. Both connections have a huge file to send (in the same direction over the bottleneck link). The transmissions of the files start at the same time. What is the transmission rate that TCP would like to give to each of the connections?

8) True or False: Consider congestion control in TCP. When a timer expires at the sender, the threshold is set to one half of its previous value?

Problems

1) Suppose client A initiates an FTP session with server S. At about the same time, client B also initiates an FTP session with server S. Provide possible source and destination port numbers for :

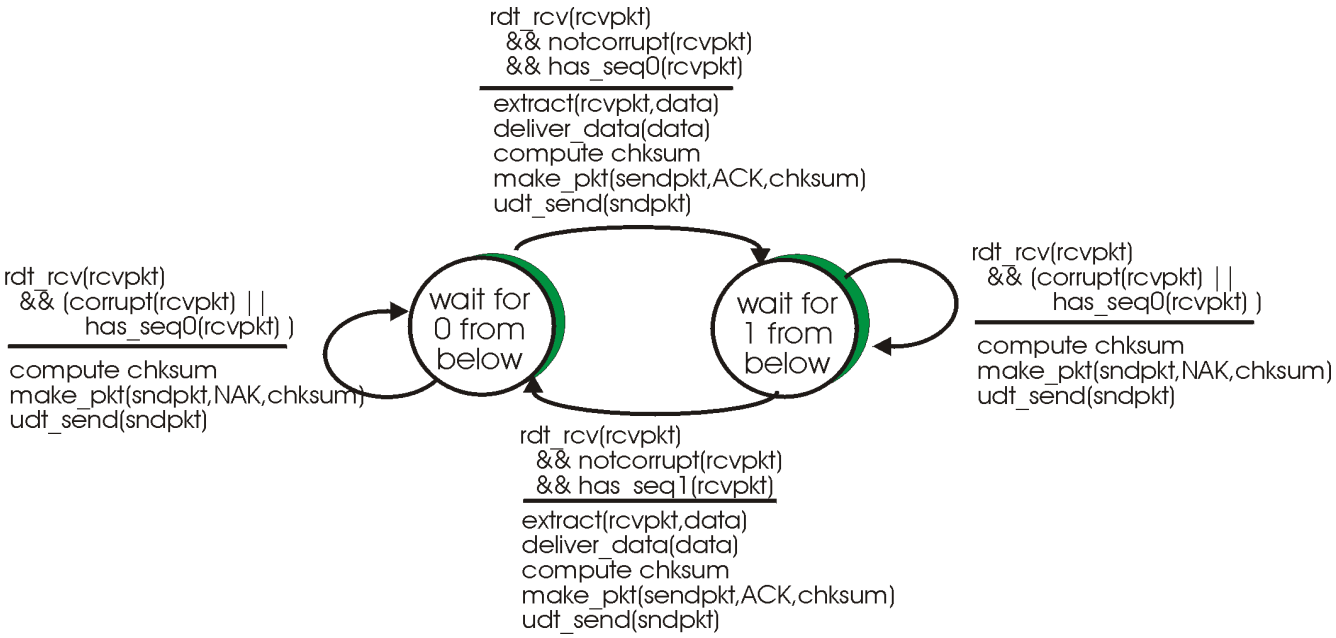
(a) the segments sent from A to S?
(b) the segments sent from B to S?
(c) the segments sent from S to A?
(d) the segments sent from S to B?

(e) If A and B are different hosts, is it possible that the source port numbers in the segments from A to S are the same as those from B to S? (f) How about if they are the same host?

2) UDP and TCP use 1's complement for their checksums. Suppose you have the following three 8-bit words: 01010101, 01110000, 11001100. What is the 1's complement of the sum of these words? Show all work. Why is it that UDP take the 1's complement of the sum, i.e., why not just use the sum? With the 1's 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?

3) Protocol rdt2.1 uses both ACK's and NAKs. Redesign the protocol, adding whatever additional protocol mechanisms are needed, for the case that only ACK messages are used. Assume that packets can be corrupted, but not lost. Give the sender and receiver FSMs, and a trace of your protocol in operation (using traces as in Figure \ref{fig57}). Show also how the protocol works in the case of no errors, and show how your protocol recovers from channel bit errors.

4) Consider the following (incorrect) FSM for the receiver for protocol rtd2.1.



Show that this receiver, when operating with the sender shown in Figure 3.4-5 can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur.

5) 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?

6) Draw the FSM for the receiver side of protocol rdt 3.0.

7) Give a trace of the operation of protocol rdt3.0 when data packets and acknowledgements packets are garbled. Your trace should be similar to that used in Figure 3.4-9.

8) 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.

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

10) Consider the cross-country example shown in Figure 3.4-10. How big would the window size have to be for the channel utilization to be greater than 90 %?

11) Design a reliable, pipelined, data transfer protocol that uses only negative acknowledgements. How quickly will your protocol respond to lost packets when the arrival rate of data ot the sender is low? Is high?

12) Consider transferring an enormous file of L bytes from host A to host B. Assumme an MSS of 1460 bytes.

a) What us the maximum length of L such that TCP sequence numbers are not exhausted? Recall that the TCP number field has four bytes.

b) For the L you obtain in (a), find how long it takes to transmit the file. Assme 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 10 Mbps link. Ignore flow control and congestion control, so A can pump out the segments back-to-back and continuously.

13) In Figure 3.5-5, we see that TCP waits until it has received three duplicate ACK before performing a fas 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?

14) Consider the TCP procedure for estimating RTT. Suppose that x = .1. Let $SampleRTT_1$ be the most recent sample RTT, let $SampleRTT_2$ be the next most recent sample RTT, etc. (a) For a given TCP connection, suppose 4 acknowledgements have been returned with corresponding sample RTTs $SampleRTT_4$, $SampleRTT_3$, $SampleRTT_2$, and $SampleRTT_1$. Express EstimatedRTT in terms of the four sample RTTs. (b) Generalize your formula for n sample round-trip times. (c) For the formula in part (b) let n approach infinity. Comment on why this averaging procedure is called an exponential moving average.

15) Refer to Figure 3.7-3 that illustrates the convergence of TCP's additive increase, multiplicative decrease algorithm. Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting additive increase additive decrease converge to an equal share algorithm? Justify your answer using a diagram similar to Figure 3.7-3.

16) Recall the idealized model for the steady-state dynamics of TCP. In the period of time from when the connection's window size varies from (W*MSS)/2 to W*MSS, only one packet is lost (at the very end of the period). (a) Show that the loss rate is equal to

$$L = \text{loss rate} = 1/[(3/8)*W^2 - W/4] .$$

(b) Use the above result to show that if a connection has loss rate L, then its average bandwidth is approximately given by:

$$\text{average bandwidth of connection} \sim 1.22 * MSS / (RTT * \text{sqrt}(L)).$$

17) Consider sending an object of size O = 100 Kbytes from server to client. Let S=536 bytes and RTT=100msec. Suppose the transport protocol uses static windows with window size W.

a) For a transmission rate of 28 Kbps, determine the minimum possible latency. Determine the minimum window size that achieves this latency.

b) Repeat (a) for 100 Kbps.

c) Repeat (a) for 1 Mbps.

d) Repeat (a) for 10 Mbps.

18) Suppose TCP increased its congestion window by two rather than by one for each received acknowledgement during slow start. Thus the first window consists of one segment, the second of three segments, the third of nine segments, etc. For this slow-start procedure:

a) Express K in terms of O and S.

b) Express Q in terms of RTT, S and R.

c) Express latency in terms of P = min(K-1,Q), O, R and RTT.

19) Consider the case RTT = 1 second and O = 100 kBytes. Prepare a chart (similar to the charts in Section 3.5.2) that compares the minimum latency (O/R + 2 RTT) with the latency with slow start for R=28Kbps, 100 Kbps, 1 Mbps and 10 Mbps.

20) True or False.

a) If a Web page consists of exactly one object, then non-persistent and persistent connections have exactly the same response time performance?

b) Consider sending one object of size O from server to browser over TCP. If O > S, where S is the maximum segment size, then the server will stall at least once?

c) Suppose a Web page consists of 10 objects, each of size O bits. For persistent HTTP, the RTT portion of the response time is 20 RTT ?

d) Suppose a Web page consists of 10 objects, each of size O bits. For non-persistent HTTP with 5 parallel connections, the RTT portion of the response time is 12 RTT ?

21) The analysis for dynamic windows in the text assumes that there is one link between server and client. Redo the analysis for T links between server and client. Assume the network has no congestion, so the packets experience no queueing delays. The packets do experience a store-and-forward delay, however. The definition of RTT is the same as that given in the section on TCP congestion control. (Hint: The time for the server to send out the first segment until it receives the acknowledgement is TS/R + RTT.)

22) Recall the discussion at the end of Section 3.7.3 on the response time for a Web page. For the case of non-persistent connections, determine a general expression for the *fraction* of the response time that is due to TCP slow start.

23) With persistent HTTP, all objects are sent over the same TCP connection. As we discussed in Chapter 2, one of the motivations behind persistent HTTP (with pipelining) is to diminish the affects of TCP connection establishment and slow start on the response time for a Web page. In this problem we investigate the response time for persistent HTTP. Assume that the client requests all the images at once, but only when it has it has received the *entire* HTML base page. Let M+1 denote the number of objects and let O denote the size of each object.

a) Argue that the response time takes the form *(M+1)O/R + 3RTT + latency due to slow-start.* Compare the contribution of the RTTs in this expression with that in non-persistent HTTP.
b) Assume that $K = \log_2(O/R+1)$ is an integer; thus, the last window of the base HTML file transmits an entire window's worth of segments, i. e., window K transmits 2^{K-1}segments. Let *P' = min{Q,K'-1}* and

$$K' = \left\lceil \log_2((M+1)\frac{O}{S}) + 1 \right\rceil$$

Note that K' is the number of windows that cover an object of size (M+1)O and P' is the number of stall periods when sending the large object over a single TCP connection. Suppose (incorrectly) the server can send the images without waiting for the formal request for the images from the client. Show that the response time is that of sending one large object of size (M+1)O:

$$approx\ response\ time = 2RTT + \frac{(M+1)O}{R} + P'\left[RTT + \frac{S}{R}\right] - (2^{P'} - 1)\frac{S}{R}$$

c) The actual response time for persistent HTTP is somewhat larger than the approximation. This is because the server must wait for a request for the images before sending the images. In particular, the stall time between the Kth and (K+1)st window is not $[S/R + RTT - 2^{K-1}(S/R)]^+$ but is instead RTT. Show that

$$response\ \ time = \ 3RTT + \frac{(M+1)O}{R} + P'\left[RTT + \frac{S}{R}\right] - (2^{P'} - 1)\frac{S}{R} - \left[\frac{S}{R} + RTT - \frac{S}{R}2^{K-1}\right]^+$$

24) Consider the scenario of RTT = 100 msec, O = 5 Kbytes, and M= 10. Construct a chart that compares the response times for non-persistent and

persistent connections for 28 kbps, 100 kbps, 1 Mbps and 10 Mbps. Note that persistent HTTP has substantially lower response time than non-persistent HTTP for all the transmission rates except 28 Kbps.

25) Repeat the above question for the case of RTT = 1 sec, O = 5 Kbytes , M= 10. Note that for these parameters, persistent HTTP gives a significantly lower response time than non-persistent HTTP for all the transmission rates.

26) Consider now non-persistent HTTP with parallel TCP connections. Recall that browsers typically operate in this mode when using HTTP/1.0. Let X denote the maximum number of parallel connections that the client (browser) is permitted to open. In this mode, the client first uses one TCP connection to obtain the base HTML file. Upon receiving the base HTML file, the client establishes M/X sets of TCP connections, with each set having X parallel connections. Argue that the total response time takes the form:

response time = (M+1)O/R + 2(M/X+1) RTT + latency due to slow-start stalling.

Compare the contribution of the term involving RTT to that of persistent connections and non-persistent (non-parallel) connections.

Discussion Questions

1) Consider streaming stored audio. Does it make sense to run the application over UDP or TCP? Which one does RealNetworks use? Why? Are there any other streaming stored audio products? Which transport protocol do they use and why?

Programming Assignment

In this programming assignment, you will be writing the sending and receiving transport-level code for implementing a simple reliable data transfer protocol - for either the alternating bit protocol or a Go-Back-N protocol. This should be FUN since your implementation will differ very little from what would be required in a real-world situation.

Since you presumably do not have standalone machines (with an OS that you can modify), your code will have to execute in a simulated hardware/software environment. However, the programming interface provided to your routines (i.e., the code that would call your entities from above (i.e., from layer 5) and from below (i.e., from layer 3)) is very close to what is done in an actual UNIX environment. (Indeed, the software interfaces described in this programming assignment are much more realistic that the infinite loop senders and receivers that many textbooks describe). Stopping/starting of timers are also simulated, and timer interrupts will cause your timer handling routine to be activated.

You can find full details of the programming assignment, as well as C code that you will need to create the simulated hardware/software environment at http://gaia.cs.umass.edu/kurose/transport/programming_assignment.htm