CMPSC 381
Data Communications and Networks
Spring 2016
Bob Roos
`http://cs.allegheny.edu/sites/rroos/cs381s2016`

Lab 8
17 March 2016
Due via Bitbucket on Tuesday, 29 March, 5 p.m
NOTE THE DIFFERENT DUE DAY AND TIME!

Summary: Congestion Control and RDT Problems (Second Exam Review)

Details: You'll solve several problems similar to (but perhaps a bit harder than) the kinds of things that may be asked on the second exam. Since the most complicated topics in the chapter are the protocols (for reliable data transfer and congestion control), these items are the focus of today's lab.

1. **[Prepare!]** To solve today's lab exercises, you need to *carefully study and understand* the finite state machine diagrams on pages 206–215 and page 275, as well as the corresponding sender/receiver time flow examples such as Figures 3.16, 3.18, etc. (These are mostly duplicated in the chapter 3 slides if you did not bring your textbook to the lab.)

   In particular, you should understand the meanings of commands such as "`rdt_send(data)`" or "`extract(rcvpkt,data)`" or "`cwnd = ssthresh+3*MSS`". While you will not be required to memorize any of these diagrams for the exam, you *will* be expected to understand the "big picture" (for instance, the three states of congestion control and when each occurs; the differences between "Go Back N" and "Selective Repeat" protocols; the difference between timeouts and triple duplicate acknowledgements; etc.).

2. **[Congestion Control (1).]** Create a file to hold your answers to this and the remaining questions in today's lab. **Be sure to put your name and the Honor Code Pledge at the top!**

   Figure 1 shows a point somewhere in the middle of a TCP session. Three segments are sent at once (`cwnd = 30 bytes`) and acknowledgements for all three are sent; the second ACK is lost. Assume that the maximum segment size, MSS, is 10 bytes and `ssthresh` is 64K bytes. The arrows are labeled with the values of the `sequence number` field and the length of the data (always 10 bytes in this example) in each TCP segment. Assume we are in the "slow start" state of the congestion control algorithm (figure 3.52, page 275—also on chapter 3 slides).

   (a) What is the value of the `acknowledgement number` field for the first acknowledgement (labeled ① in Figure 1)?

   (b) What is the value of the `acknowledgement number` field for the third acknowledgement (labeled ② in Figure 1)?

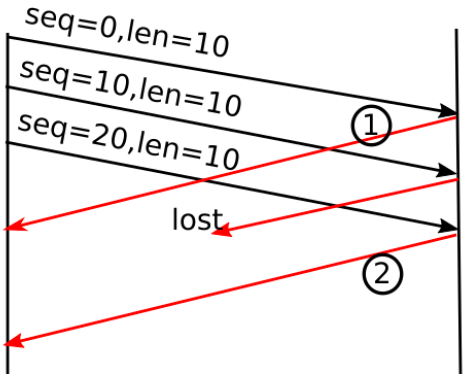

Figure 1: See problem 2; initially, `cwnd` = 30, `ssthresh` = 64K

(c) Assume that acknowledgement ② arrives *before* any of the seqments' timers expire. What is the new value of `cwnd` (measured in bytes)?

(d) What will be the sequence number of the next segment sent by the sender?

(e) Does the value of `ssthresh` change? If so, what is the new value?

(f) In which state of the congestion control algorithm are we after the events above take place?

3. **[Congestion Control (2).]** Figure 2 shows the same scenario as problem 2 with only one difference: acknowledgement ① arrives *after* sequence 0's timer expires (labeled ② in Figure 2).

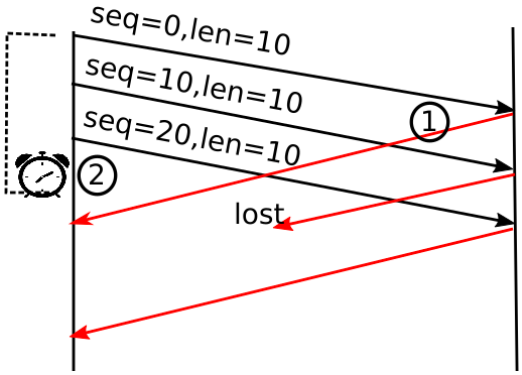

Figure 2: See problem 3; initially, `cwnd` = 30, `ssthresh` = 64K

(a) What is the new value of `cwnd` (measured in bytes)?

(b) What will be the sequence number of the next segment sent by the sender?

(c) Does the value of `ssthresh` change? If so, what is the new value?

(d) In which state of the congestion control algorithm are we after the events above take place?

4. **[Congestion Control (3).]** Figure 3 shows a point somewhere in the middle of a TCP session. Five segments are sent at once (`cwnd` = 50 bytes); the second segment is lost. Acknowledgements for the other four are sent. Assume that the maximum segment size, MSS, is 10 bytes and `ssthresh` is 64K bytes. Assume we are in the "slow start" state of the congestion control algorithm and that no timeouts occur..

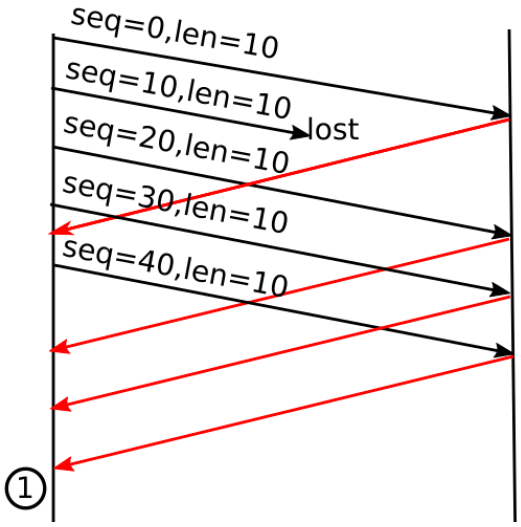

Figure 3: See problem 4; initially, `cwnd` = 50, `ssthresh` = 64K

   (a) After the last acknowledgement arrives (labeled ① in Figure 3), what is the new value of `ssthresh`?

   (b) What is the new value of `cwnd`?

   (c) What will be the sequence number of the next segment sent by the sender?

   (d) In which state of the congestion control algorithm are we after the events above take place?

5. **[Congestion Control (4).]** Go to the textbook website and look at the "Interactive Exercises" for chapter 3. The problem below is taken from the "TCP congestion window evolution" problem.

   Consider Figure 4 below, which plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT); see Figure 3.53 in the text. In the abstract model for this problem, TCP sends a "flight" of packets of size `cwnd` at the beginning of each time unit. The result of sending that flight of packets is that either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. In this problem, you are asked to reconstruct the sequence of events (ACKs, losses) that resulted in the evolution of TCP's `cwnd`.

   Consider the evolution of TCP's congestion window in the figure and answer the following questions. The initial value of `cwnd` is 1 and the initial value of `ssthresh` is 8.

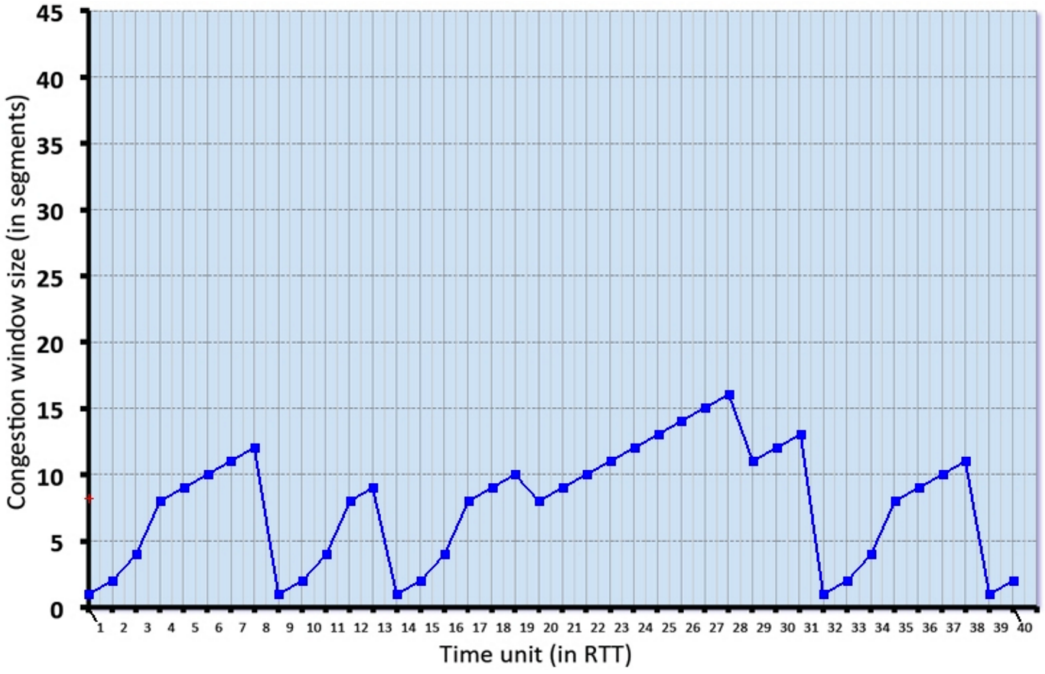

Figure 4: See problem 5

(a) Give the times at which TCP is in slow start, congestion avoidance and fast recovery at the start of a time slot, when the flight of packets is sent.

(b) Give the times at which the first packet in the sent flight of packets is lost, and indicate whether that packet loss is detected via timeout, or by triple duplicate ACKs.

(c) Give the times at which the value of `ssthresh` changes, and give the new value of `ssthresh`.

6. **[Reliable Data Transfer.]** Figure 5 shows the sender and receiver for a reliable data transfer protocol. We make the following assumptions:

   - the sender will never receive new requests to send until it has received an acknowledgement for the previously-sent data
   - packets (both data and ACKs) do not have sequence numbers
   - packets can be lost
   - there are no bit errors

   Briefly, the sender sends a packet and waits for an ACK. If no ACK is received within a certain time, the packet it resent. Once an ACK is received, the sender is ready to transmit new data. The receiver simply waits for a packet to arrive, delivers it up to the application layer, and sends an ACK for it.

   (a) Can data packets ever arrive out of order at the receiver? Explain your answer.

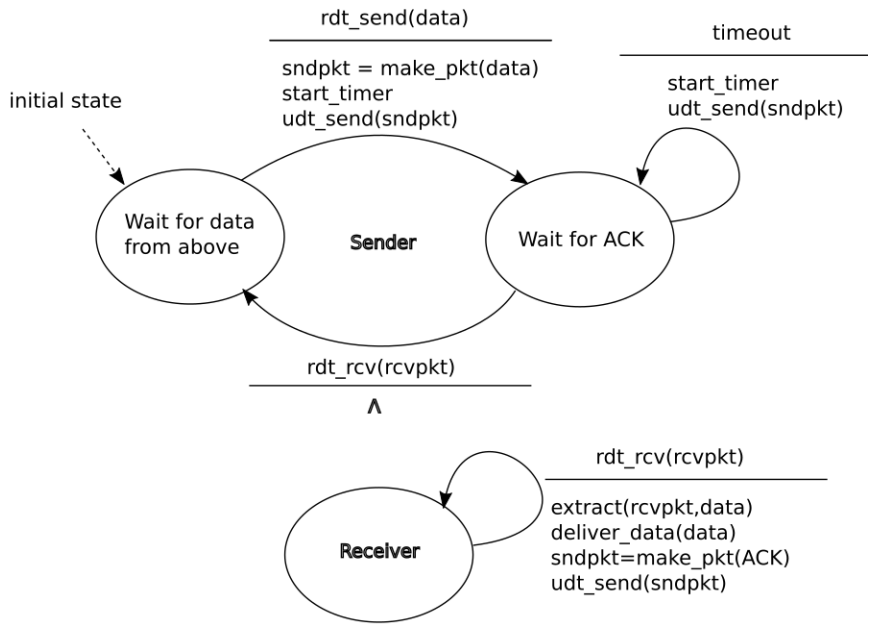

Figure 5: See problem 6

(b) Describe a scenario in which the receiver could deliver the same data twice to the application layer.

(c) Which of the following mechanisms could be used to prevent the scenario in part (b) (these would require adding more transitions and, in some cases, more states):

- Allow the receiver to send negative acknowledgements (NAKs)
- Add sequence numbers to the data packets
- Add a timeout event to the receiver

[Submit your work.] Make sure you have the following files in your lab8 folder:

- A PDF file with your answers to the questions. ***Make sure your name and the honor code pledge appear at the top of the file.***

Upload this folder to your Bitbucket repository by the lab deadline.