COSC 4377 – Networking - Kevin B Long

# interlocking-uh-m-186.eps

Homework #5

Due 11:59**pm**, Friday, 29 March 2019

Some problems due later, see below

Multiple submissions accepted.

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Peoplesoft ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (10 pts) Complete the 7th Wireshark lab on TCP
2. (8 pts) Use the Flow Control animation at <https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/flow-control/index.html> to observe a simple flow-control application.

Let’s step through an example if you need it; you can skip and go to the question in part (a) if you like.

Step 1 – the App Layer had 16KB to send. It has just handed the sender TCP buffer 2KB, leaving it with 14KB remaining. Note that the top part of the diagram with the two tall rectangles labeled Host A and B are the Application Layer, The boxes below connected by the pair of horizontal lines are the Transport Layer programs running on the two hosts.

What else is happening in this picture? A lot!

* They negotiated a window size of 2048 bytes (see ①). This was the minimum of rwnd (what the receiver’s window has room to accept) and cwnd (what the sender wanted to send).
* During A’s handshake with B, A announced he would start with sequence #0 for data (see ②).
* The Sender’s TCP buffer is full, the receive buffer is empty.



Step 2: Packet is in flight from Sender to Receiver



Step 3: Packet arrives safely at the receiver (host B), and sends back an Acknowledgment (in red). Several things to note:

* Host A’s application delivers the next 2K to the sender’s TCP window.
* The Receiver’s TCP buffer is full, and will remain that way as the receiver waits for the Application in the layer above to take delivery of some or all of the buffer of data. This assumes TCP is willing; it may buffer data out-of-order. We hope there would be room left for the out-of-order packet!
* The rwnd variable (called “WIN” in this animation) is set to 0 by the receiver, and is sent back in the ACK, to let the sender know not to send any more data. The receiver will never update this value, so it’s on the sender’s shoulders to periodically send a probe (more on this in the next step).
* The packet coming back contains an ACK value of 2048, announcing the receiver’s successful receipt of all bytes up to this point, and its readiness to now receive bytes starting at sequence number 2048. In our case, we know 0-2047 were sent, but since we don’t usually start with 0 (instead we choose a random number), you can’t depend on knowing how many bytes have already been sent by looking at the sequence number.



Step 4: Upon receiving the news that the receiver’s buffer is full, the sender will probe by sending 1-byte payloads from its window of data to hopefully get an ACK and an updated rwnd value advertising more room. Until the receiver has room, it will not acknowledge receipt of this byte, and the sender’s timer will expire and it will continue to try for a while. It is possible the receiver has data for the sender itself. After all, there is an app running above him too. The receiver will continue to update the sender on the status of its window this way also.



Step 5: By the time our probe arrives at the receiver, we’re in luck; the App has accepted the entire buffer of data, all 2KB, so our probe is ACK’d with WIN=2048, indicating the receiver has 2K of room again. Note again that the 1-byte probe is ignored in this simulator, so the ACK is still 2048, not 2049 as it would be in reality.



Step 6: We’re back in business; the second buffer of 2K at the sender is now on the wire; it will soon take another 2K from the application and wait to send. But the 2K on the wire, our second large packet, will arrive at the receiver and fill its buffer again, and the cycle will repeat.



OK, so with that as the explanation let’s get to the questions!

An app originally had 12KB to send. Both send and receive TCP buffer sizes are 2KB. At the moment, there remains only 2KB in the app’s buffer to be handed down to TCP. Also, the sender has just put the contents of his buffer onto the network headed towards the receiver. Answer the following questions:

1. What is the sequence number be found in the TCP Header? \_\_\_\_\_\_\_\_\_\_\_
2. What will the length of the payload be? \_\_\_\_\_\_\_\_\_\_\_
3. What size of receiver buffer will the receiver advertise back in his acknowledgment?

\_\_\_\_\_\_\_\_\_\_\_\_

1. What will the sender need to transmit after this packet before continuing with the last 2K?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (4x4=16 pts) Recursive/Iterative DNS Queries

Go to the simulator at <https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/recursive-iterative-queries-in-dns/index.html>

There are four types of scenarios shown, and two options for the local name server and the root name server. The names and labels are a little different than what we saw in the book but the simulator is showing us the difference between iterative and recursive DNS queries.

Answer the following questions:

1. Which scenario type always results in the greatest number of DNS queries, regardless of the server settings?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Which has the shortest?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. If you wish to minimize the number of queries handled by the root server across all caching scenarios, what settings do you need?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

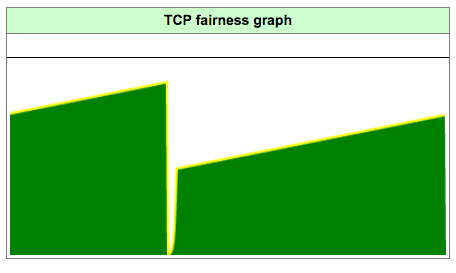
1. If you wish to hide as much of the identity of your DNS infrastructure from the local name servers as possible, which query method should you choose? Iterative or Recursive?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (4x4=16 pts) TCP Congestion Control

Access the simulator at <https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/tcp-congestion/index.html> and answer the following questions:

Consider the following graph:



1. There are 3 situations in the animation that can cause this dip in the bandwidth being used by a single workstation. What are they?
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ii. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

iii. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Add a second workstation and let the simulation run for a few moments, and then experiment introducing errors. When will the bandwidth consumption be at its lowest?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (26 pts) TCP Retransmissions

This question is based generally on the simulation available here:

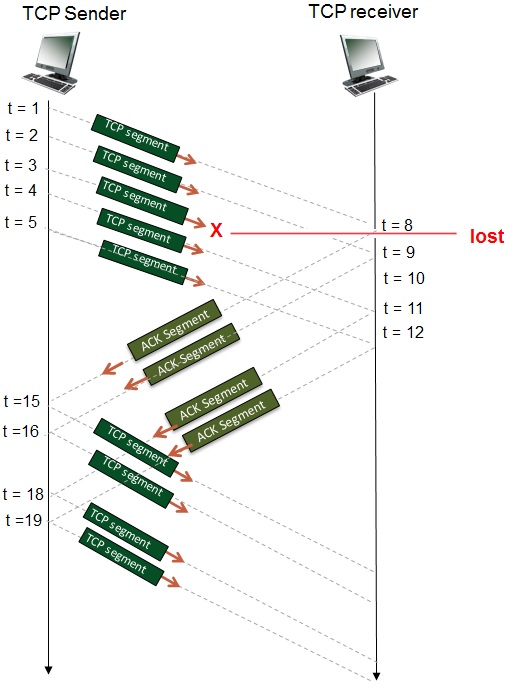
<http://gaia.cs.umass.edu/kurose_ross/interactive/tcp_retrans.php>

However, the simulation has a lot of errors in the answers. So take a look at how it works, but don’t take the answers to heart. Remember what acknowledgment we send when we receive a packet out-of-order; this is based on TCP Reno just as our lectures have been.

Consider the figure below in which a TCP sender and receiver communicate over a connection in which the sender-to-receiver segments may be lost.

* The TCP sender has window size of 5 segments. The diagram shows when they are transmitted. Let’s also number the segments starting with 1.
* The initial sender-to-receiver sequence number is 100
* All sender-to-receiver segments each contain 100 bytes.
* The delay between the sender and the receiver is 7 time units (the numbers are correct even though it looks like 8 comes before 5 – just ignore that) So for example, the first segment arrives at the receiver at t=8, and a receiver-to-sender ACK for this segment arrives at the TCP sender at t=15.
* There is no processing delay in the receiving host, so it acknowledges every packet as soon as it arrives.

As shown in the figure, one of the five segments in the initial round is lost between the sender and the receiver, but none of the ACKs are lost. Answer the following questions:

TCP Congestion Control 

1. (1 pts) Will we receive any duplicate ACKs? If so, how many and with what sequence number(s)?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (1 pts) In the beginning, the window starts with what sequence number? This is the value of cwnd.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 pts) In the beginning, what is the highest sequence number inside the window? Remember, the window begins at cwnd and continues for the number of segments in the window multiplied by the size of each segment.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (5 pts) What is the sequence number for each of the first five packets sent?

|  |  |
| --- | --- |
| Time | Seq # |
| Time 1 |  |
| Time 2 |  |
| Time 3 |  |
| Time 4 |  |
| Time 5 |  |

1. (5 pts) What is the acknowledgment number sent for each of the first 5 segments (packets)?

|  |  |
| --- | --- |
| Segment | Seq # |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

1. (2 pts) After the first ACK from this round is received back at the sender, what will the value of cwnd be? Assume the window size remains fixed.

\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 pts) After all the ACKs from this round that will be received are received, what will the value of cwnd be, and the last byte number in the window? Assume the window size remains fixed.

cwnd:\_\_\_\_\_\_\_\_\_\_\_\_ last byte:\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 pts) What must happen in order for transmissions to continue? Select all that apply.

☐ Packet 3 must be retransmitted due to a triple-duplicate ACK

☐ Packet 3 must be retransmitted due to a timeout (timer expiration)

☐ All packets in the window must be resent, segment #1-5

☐ All packets in the window must be resent, segment #3

☐ All packets in the window must be resent, segment #3-7

☐ All packets in the window must be resent, segment #3-9

☐ All packets in the window must be resent, segment #5-9

1. (2 pts) What will happen when the missing packet 3 is finally received? Select the best one from each group.

☐ The receiver will acknowledge with byte #300

☐ The receiver will acknowledge with byte #500

☐ The receiver will acknowledge with byte #700

☐ The receiver will acknowledge with byte #900

☐ Other. The receiver will acknowledge with byte #\_\_\_\_\_\_\_\_\_\_\_

☐ The receiver will deliver packet 3 to the application

☐ The receiver will deliver packets 3-5 to the application

☐ The receiver will deliver packets 3-7 to the application

☐ The receiver will deliver packets 3-9 to the application

☐ Other. The receiver will deliver to the application packets \_\_\_\_\_\_\_\_\_\_

1. (2 pts) If you change the missing packet to be packet 2, will your answer for part h change? Explain.

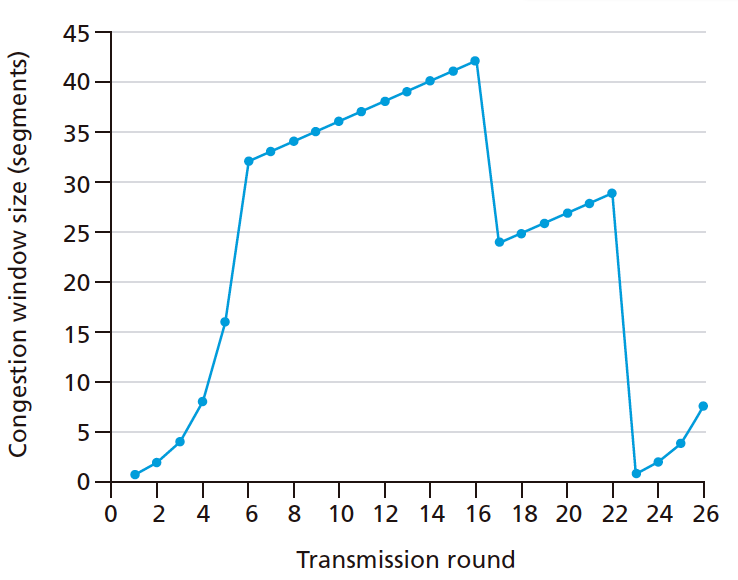
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 pts) Did you spot any serious problems with the diagram (and the simulator)?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (24 pts) Consider figure following diagram.



We are running TCP Reno.

1. (2 pts) In what mode is our sender at transmission round 1?

☐ Slow Start ☐ Congestion Avoidance

1. (2 pts) What is the initial value of ssthresh? \_\_\_\_\_\_\_\_\_\_
2. (2 pts) What’s the significance of this value? In other words, what changes as we cross that threshold?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 pts) What kind of loss is depicted first, and during which transmission round is it detected?

☐ Triple-Duplicate ACK ☐ Timeout Round?\_\_\_\_\_\_

1. (2 pts) What is the value of ssthresh after that loss? \_\_\_\_\_\_\_\_
2. (2 pts) What mode are we in after the first loss?

☐ Slow Start ☐ Congestion Avoidance

1. (2 pts) What kind of loss is depicted last, and during which transmission round is it detected?

☐ Triple-Duplicate ACK ☐ Timeout Round?\_\_\_\_\_\_

1. (2 pts) How many packets are transmitted in the first 10 rounds?

\_\_\_\_\_\_\_\_\_\_\_

If your MSS was set to 100 bytes, in what round on this graph would we have finished sending 21,000 bytes’ worth of segments?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (8 pts) Finally, use what you’ve learned to label the graph showing where the various points at which ssthresh changes. Include round 1 so you can label the initial value. Each time you change ssthresh, also list the value of cwnd. You can list it in tabular format instead of you like. Not all rows may be used.

|  |  |  |
| --- | --- | --- |
| Round | ssthresh | cwnd |
| 1 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |