CSC / ECE 573 Internet Protocols, Fall 2005

Homework #3

Due Date

- Part I is due Friday, September 23, at 11:45 PM
- Part II is due Monday, October 3, at 11:45PM

Instructions

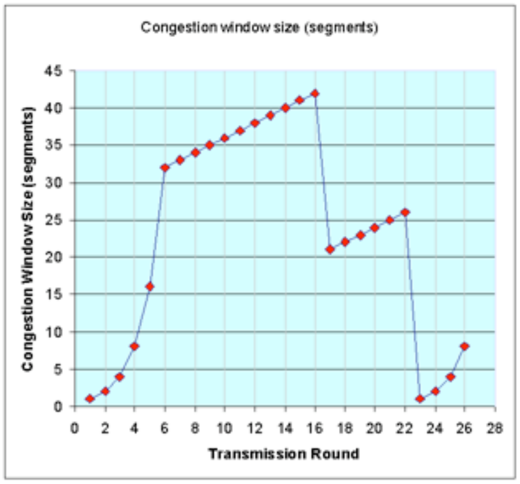
- Homeworks should be submitted individually. We will use the standard submit utility for our class to submit all work, which means your work must be prepared electronically.
- Put your name, the assignment number, and date at the top of the first page. Put solutions in order (don't make the TA hunt for your solution).
- Do not plagiarize; that means, do not copy content from any source without permission from the instructor, and if permitted, acknowledge the source.
- This homework is worth a total of 200 points (100 points for part I, 100 points for part II)

PART I Problems

1. Suppose if a UDP datagram with non-zero checksum failed the checksum test at the receiver (i.e., there are one or more bit errors), an ICMP error message was sent back to the source, containing as the payload 28 bytes of the corrupted datagram (20 bytes IP header and 8 bytes UDP header). Does the sender have everything it needs at that point to correct the problem and resend the UDP datagram? Explain.
2. Would you consider UDP or TCP "better" for the following functions, and why?
   * A "quote of the day" service
   * Mail transfer
   * Dynamic host configuration (DHCP)
   * Transmission of a TV program from a video server to thousands or millions of receivers
3. Suppose you wanted to use UDP to transmit some data to a receiver, and you wanted to ensure that each datagram was received in order and only once. However, reliability was not required (i.e., it was not necessary to ensure all datagrams were received), nor was flow control or congestion control necessary. What would need to be added to the UDP header to provide such capabilities, and what behavior would you need to add to the sender and receiver?
4. Why does the UDP header have a length field, but the TCP header doesn't? Is it unnecessary, or is there some capability of UDP that is not present in TCP?
5. The following is a dump of a TCP header in hexadecimal format:
   05320017 00000001 00000000 500207FF 00000000
   * What is the source port number, in decimal?
   * What is the destination port number, in decimal?
   * What is the sequence number, in decimal?
   * What is the acknowledgment number, in decimal?
   * What is the length of the header, in decimal?
   * What is the type of the segment?
   * What is the window size, in decimal?
6. Enumerate the states in the TCP state-transition diagram which hosts A and B visit during the following sequence of events and specify the time intervals which they spend in each state. Each segment is identified by an integer ID in parentheses.
   * Time 0: A and B are communicating in the ESTABLISHED state.
   * Time 25 ms: A closes and sends a FIN (1).
   * Time 225 ms: A times out its first FIN (1) and retransmits another FIN (2).
   * Time 226 ms: B receives FIN (1) from A and sends an ACK (3).
   * Time 228 ms: B closes and sends a FIN (4).
   * Time 235 ms: B receives FIN (2) from A and sends an ACK (5).
   * Time 247 ms: A receives ACK (5) from B.

* Time 250 ms: A receives ACK (3) from B.
* Time 428 ms: B times out its first FIN (4) and retransmits another FIN (6).
* Time 445 ms: A receives FIN (6) from B and sends an ACK (7).
* Time 467 ms: B receives ACK (7).

7. Suppose it was suggested to use a two-way handshake to establish connections rather than a three-way handshake. (In other words, the third message was not required) Are deadlocks now possible (neither party proceeds, both are waiting for the other)? Given an example or show that none exist.
8. If a server goes through the SYN_SENT state, who has initiated the TCP connection *establishment*, the client or the server? If a server goes through the CLOSE_WAIT state, who has initiated the TCP connection *termination*, the client or the server?
9. A client C and a server S establish a connection. The MSS is 1000, and the initial sequence number is 2000 from C to S and 7500 from S to C. S has a buffer that will hold a maximum of 4000 bytes, while C's buffer will store 10000 bytes. Show the Sequence Number and Ack Number of each segment below, and the Window Size as well.
   * C sends SYN segment (#C1)
   * S sends SYN/ACK segment (#S1) of (#C1)
   * C sends ACK segment (#C2) of (#S1)
   * C sends one data segment (#C3) with ack of (#S1)
   * S sends ACK segment (#S2) of #C3
   * C sends two more data segments (#C4 and #C5)
   * S sends ACK segment (#S3) of #C4
   * C sends 3 more data segments (#C6, #C7, and #C8)
   * S sends ACK segment (#S4) of #C5 and #C6
   * C sends FIN segment (#C9)
10. The MSL for a TCP connection is 5 seconds. What is the minimum length the sequence number field can be to avoid the "overlapping segments" problem if the connection transmits at a steady 1000 segments / second, where each segment payload (not counting IP and TCP headers) is 1000 bytes long?
11. Assume RTT and MDEV at some point have been calculated to be 100 ms and 60 ms, respectively. What will RTT, MDEV, and RTO be after each of the following packet round-trip times are measured? Use .875 as the weight for RTT, .25 as the weight for MDEV, and using Jacobson's algorithm, Karn's algorithm, and exponential backoff.
   * 80 ms
   * 230 ms
   * 30 ms
   * (ACK times out and segment is retransmitted)
   * (retransmitted segment ACK again times out and segment is retransmitted)
   * 70 ms (for the second retransmitted segment)
   * 220 ms
12. An application at host A generates one byte of data every .1 s, starting at time 0. The round-trip time for acknowledgment of a data packet is exactly .31 sec (assume no variation occurs), and the receiver advertises a window of size 10 at all times. Using Nagle's algorithm, and assuming the application sends a total of 15 bytes, show the time at which data packet will be sent, and how many bytes of data each will contain. Answer the same question without the use of Nagle's algorithm. Which is better, and why?
13. A TCP receiver has a buffer of size 3072 bytes, and a maximum segment size of 1024. The receiver buffer has 2048 bytes in it when a segment of size 1024 arrives. If the bytes in the buffer are removed by the application 1 every millisecond, how long will it be before this segment is acknowledged, using Clark's algorithm? If the receiver buffer had 0 bytes in it when the segment was received, what would your answer be instead?
14. What is the bandwidth-delay product for a 100 * 10^6 bits/s channel to a geostationary satellite? Assume the round-trip TCP delay over this channel is 1200 ms. If the packets are all 1500 bytes (including IP and TCP headers), how big should the transmission window be in packets to just saturate this channel, and what will the data throughput be? (ignore for this problem any framing or transmission overhead)
15. Suppose segments of fixed size are sent on a connection initially having a window size of 5 segments. The following sequence of events occurs:
   - Segments 1 through 4 are sent
   - Segments 1 through 2 are acknowledged, and the window size is grown to 11 segments (i.e., 11 unacknowledged segments may be sent)
   - Segments 5 through 10 are sent
   - Segments 1 through 7 are acknowledged, and the window size is changed to 10 segments
   * How many empty "slots" are there in the window at this point (i.e., how many more segments can be sent without any further acknowledgment being received)?
   * Is it possible at this point for the receiver to send an acknowledgment of segments 1 through 9 with a window size of 6 segments? Why or why not?
16. Consider the following plot of TCP window size as a function of time.



Assuming fast retransmit and recovery are used, answer the following questions. In all cases, justify your answer.

1. Identify the intervals of time when TCP slow start is operating.
2. Identify the intervals of time when TCP congestion avoidance is operating.
3. After the 16th transmission round, is segment loss detected by a triple duplicate ACK, or by a timeout?
4. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK, or by a timeout?
5. What is the initial value of ssthresh at the first transmission round?
6. What is the value of ssthresh at the 18th transmission round?
7. What is the value of ssthresh at the 24th transmission round?
8. During what transmission round is the 70th segment sent?
9. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of cwnd and ssthresh?

17. Using the slow-start congestion control algorithm, show what cwnd and ssthresh are (in bytes) after each of the following events, assuming MSS=1024, cwnd = 2048, and ssthresh = 5120 initially.
    * maximum-sized segment (#1) is sent and acknowledged
    * Two more maximum-sized segments (#2 and #3) are sent and acknowledged
    * 4 more maximum-sized segments (#4, #5, #6, #7) are sent and #4 is acknowledged
    * segments #5-#7 time out and are retransmitted
    * #4 is acknowledged again
    * #4 is acknowledged again
    Answer this question, if it matters, for two cases: with, and without fast retransmit and fast recovery.
18. Consider the effect of using slow-start on a line with a 20-msec round-trip time and no congestion. The receiver window is 12KB and the maximum segment size is 1KB. How long does it take before the first full window (i.e., equal to the receiver window size) can be sent? Assume data is removed from the receiver buffer as soon as it arrives.
19. Design and describe a retransmission protocol based on "NACKs" (negative acknowledgements), i.e., the receiver sends a NACK only when it *doesn't* receive something it was expecting. Obviously the receiver manages RTO timers rather than the sender. You do not have to indicate how RTT, MDEV, or RTO are calculated. Are there any advantages to using NACKs instead of ACKs? Disadvantages?
20. Why does the RED algorithm tend to avoid synchronizing the congestion response of multiple connections sharing a single bottleneck link in the network? Under what conditions might the RED active queue management algorithm provide *worse* aggregate throughput than the drop-tail policy?

PART II Problems

1. The following commands were executed, and the resulting traffic was captured:
   ping www.ietf.org
   tracert www.ncsu.edu
   tracert www.ietf.org
   Answer the following questions using hw3-capa-filt2.bin:
   1. What's in the echo requests sent by ping? What do you notice about identifiers? About Sequence Numbers? About the Data?
   2. What's in the echo replies?
   3. tracert on Windows does not work the way I described it in lecture. What is the difference?
2. The following commands were executed:
   ping www.ietf.org with varying payload sizes (1500, 1460, 1300, 1200), and DF flag set. For all except the last, I got a message saying "Packet size too large, but DF set"

ping www.ietf.org with record route option set
Answer the following questions using hw3-capg-v3.bin (*instead of* hw3-capg.bin).
1. What is the approximate RTT?
2. *Nothing* got transmitted for the "packet size too large" cases. What do you conclude?
3. With record route, where is it recorded in the reply, and what is recorded? What is in the request?

3. The following commands were executed:
connected to remote-linux.eos.ncsu.edu using ssh
cat'ed a file to the screen
killed the process while it was running
Answer the following questions using hw3-capd-filt.bin (Note: in ethereal, if you want to focus on TCP only, and don't care about the application (SSH), then use "Analyze >> Enabled Protocols >> (Disable SSH)"):
   1. Between time 14 and 56.063, how much data was transferred to 152.14.62.49, and what was the effective average data transfer rate in that direction?
   2. What happened (in the network) when the process was killed?
4. The following commands were executed:
connected to a remote machine (in Australia, machine 203.16.234.19!)
started file transfer to the local machine (192.168.0.103) and interrupted it at some point
Answer the following questions using hw3-capb-tcpsumm-v2.xls and hw3-capb-filt-v3.bin: (Note: again, in ethereal, you may want to press use "Analyze >> Enabled Protocols >> (Disable HTTP)"):
   1. How much data was transferred from the remote machine to the local machine? What is the approximate transfer rate?
   2. Graph the cumulative bytes transferred vs. time elapsed for the first 80 segments transferred to to the local machine (this is already shown in the Excel file as "chart1"). What do you conclude; is slow-start in operation?
   3. In general, how many segments were ACK'ed with a single ACK? Was there variation?
   4. Does the local machine advertised window size change much? Is the receiver speed a limiting factor?
   5. What is a "window update"? Why do you think it occurs?
   6. When ethereal indicates packets were lost, were they really not delivered to the receiver, or could ethereal just not keep up with the transfer rate?
   7. What are the common segment lengths from sending (remote) machine? the maximum? * What is the approximate RTT to this server? Is there much variation in the RTT?
5. The following commands were executed:
connected to a remote machine
Initiated a large ftp transfer from another site, with a high transfer rate
Cancelled this transfer after a little while
Answer the following questions using hw3-caph-filt.bin and hw3-caph-tcpsumm.txt:
   1. Why are there so many retransmissions? Could this also be an artifact (i.e., a mistake) due to ethereal failing to capture all the arriving packets or departing acknowledgements? Why or why not?
   2. Why do you think the server is alternating transmission of segments of size 1260 and 191 bytes?
   3. At time 7.105 there was a keepalive. I found this keep-alive very strange, given the state of the connection. What is the state of the connection?
6. The following commands were executed:
initiated a streaming audio broadcast to my computer
Answer the following questions using hw3-capi-filt.bin:
   1. I expected to see UDP in use for this application, but instead TCP is being used. Why? (or why isn't UDP being used?)
   2. How large are the segments being transferred?
   3. Graph the receiver (152.14.62.49) window size; how does it behave for this application? What happens when the window is full (advertised window size goes to 0?

Created on September 16 , 2005
Last Modified October 3, 2005
Maintained by Douglas S. Reeves