University College London
Department of Computer Science
3035/GZ01: Networked Systems
Kyle Jamieson and Brad Karp
Individual Coursework 3
Distributed: 31st October 2013; Due: 19th November 2013, 4:05 PM

Answer all of the following problems. Either handwritten or typeset solutions are fine, but if you write by hand, please ensure your answers are legible. Please show all work! We cannot award credit for correct answers if their complete derivation isn't shown. Please state clearly all assumptions you make while solving a problem. This coursework is worth 8% of the total marks for 3035/GZ01.

Please monitor the 3035/GZ01 Piazza site during the period between now and the due date for the coursework. Any announcements (*e.g.,* helpful tips on how to work around unexpected problems encountered by others) will be sent to the list.

Hand-in instructions: Hand in hardcopy for your solutions at the start of lecture at 4:05 PM on the 19th of November. There is no provision for electronic submission of this coursework. Any late submissions should be handed in at the 5th floor reception desk in the CS department (in MPEB).

Collaboration: Collaboration is *not permitted* on this problem set; you may not discuss the problems or their solutions with anyone else (whether or not the other person is taking the class), apart from the instructors and teaching assistants. All work you submit must be your own, produced without consulting a solution written by any other person. You may of course refer to all lecture notes and readings, and any other materials you wish (textbooks, papers, or material found on the Internet, provided it does not contravene the prohibition from looking at another person's solution).

Late days: If you wish to use any late days on this assignment, you *must* state how many days you wish to use clearly at the top of the first page. If your written assignment does not include a request to use late days, you will not be permitted to use date days on the assignment later. (Full late-day policies are stated on the 3035/GZ01 web site.)

1. **TCP and High-Bandwidth Paths**

   Your physicist friend C. King Higgs is running an experiment on the Large Hadron Collider (LHC) nuclear physics apparatus at CERN. He has enlisted your help as someone knowledgeable in networking in sending the data from Switzerland, where the LHC is located, to his own university laboratory at Caltech in California.

   The experiments your friend is conducting will generate measurement data at the impressive rate of 5 Gbits/s. He has arranged two 10 Gbits/s links, one to connect each of CERN and Caltech to the same international Internet Service Provider's (ISP's) backbone, and this backbone has sufficient capacity to carry the 5 Gbits/s data flow between CERN and Caltech.

You decide to use TCP over the Internet to carry the data between CERN and Caltech over this ISP's network. Suppose that the round-trip time (RTT) between CERN and Caltech is 200 ms. Throughout this question, ignore the bandwidth overhead of TCP and IP headers, and assume that the sender sends 1460 user data bytes per TCP segment.

(a) How large would the receiver's advertised window and the sender's send window both need to be at a minimum in order to achieve a throughput of 5 Gbits/s? In your answer, ignore TCP congestion control—that is, assume that the congestion window is larger in size than the receiver's advertised window and sender's send window at all times. Justify your answer by explaining why TCP requires that window size.

[2 marks]

(b) Now, let's consider some of the effects of TCP congestion control, and in particular the congestion window (`cwnd`), on this data transfer. Assume that the path between sender and receiver drops no packets, that the receiver's advertised window and sender's send window are not limiting factors, and that the TCP receiver does not use delayed ACKs. At the start of the connection between CERN and Caltech, how long will it take for the sender to reach the window size that will allow the sender to send at 5 Gbits/s (the window size that you found in the previous part of this question)? Justify your answer by explaining any relevant TCP congestion control behaviors that are invoked at the start of a connection.

[2 marks]

(c) Suppose that the sender is currently using the exact full congestion window (`cwnd`) size needed to sustain 5 Gbits/s throughput. During the sending of this window's worth of data, a router along the path between sender and receiver drops a single packet of the sender's data. Assume that the sender detects this loss upon receiving three duplicate ACKs, and adjusts its congestion window (`cwnd`) accordingly. How long (in seconds) will it take before the sender's window once again reaches the exact full congestion window (`cwnd`) size needed to sustain a throughput of 5 Gbits/s? In your answer, you may assume that the network drops no packets after the single loss described above.

[2 marks]

(d) Assume that there is no loss between the sender and receiver, and that all window sizes are such that the sender is able to send at 5 Gbits/s. How long would it take for the sender to cycle through the entire possible range of TCP sequence numbers for sent packets, and begin sending packets with sequence numbers it has already used to send previous packets on the same connection?

[2 marks]

(e) Suppose that the 5 Gbits/s data flow between CERN and Caltech lasts hours. Given the answer to the previous part of this question, what best-effort packet delivery phenomenon present in the Internet would cause the reassembled byte stream produced by the TCP receiver at Caltech to differ from the original byte stream passed to TCP by the sending application at CERN? Justify your answer by explaining exactly how this phenomenon could cause the receiver to receive a different byte stream than what was sent.

[2 marks]

2. TCP Retransmissions, RTT Estimation, and Short Flows

A TCP sender carries out estimation of a connection's round-trip time using the improved method advocated by Van Jacobson in his classic paper, Congestion Avoidance and Control, assigned for GZ01. Recall that the sender samples the round-trip time between when each data packet is sent and when the ACK returns for that data packet. It does so by recording the time on its local clock when a data packet was sent, and recording the time on its local clock when the corresponding ACK returns.

(a) Suppose the sender retransmits a packet after a timeout. In such cases, the sender cannot accurately sample the round-trip time for a packet that it retransmits. Why not? Justify your answer by explaining the problem that could arise, and give an example of a sequence of events that could occur in the best-effort Internet that would cause the TCP sender to arrive at an incorrect single sample of the RTT for a packet that it retransmits.

[2 marks]

(b) Suggest a different mechanism that TCP could use to sample the round-trip time experienced by a packet that would work correctly, even for packets that are retransmitted. You are allowed to add a fixed-size additional field to the TCP header. In your answer, describe what the sender writes into this additional field, what the receiver does with the contents of this additional field, and why this method yields correct results for retransmitted packets.

[5 marks]

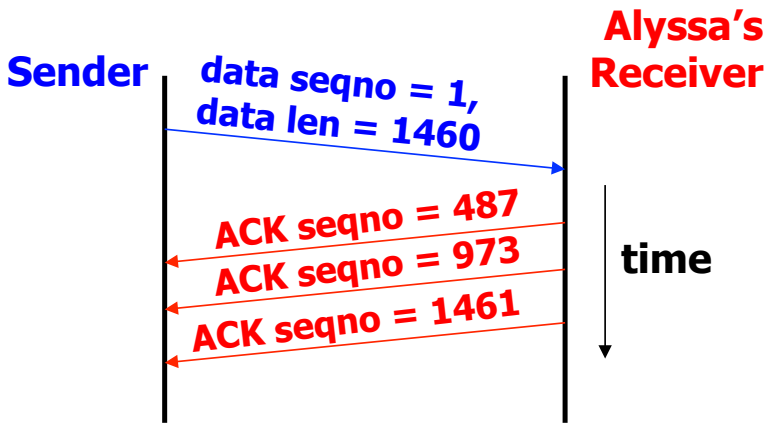
(c) A busy web server sends predominantly short files of under 8 KB in size to web clients using TCP. A measurement study finds that when this web server retransmits data packets with TCP, 56% of the retransmissions are after a retransmit timeout (RTO), and 44% of the retransmissions are done using fast retransmit. Assuming that this web server fully and correctly implements the congestion control and avoidance algorithms specified in Van Jacobson's paper, Congestion Avoidance and Control, why is it retransmitting after RTOs so often, instead of using fast retransmit?

[3 marks]

3. **Faster Downloads with TCP**

TCP sequence number and ACK number fields in the TCP header are expressed in bytes. However, most implementations of the sending side of TCP congestion control assume that when an ACK returns, it is acknowledging an entire segment of data.

(a) Your friend Alyssa P. Hacker wants to speed up TCP transfer times for her downloads. In particular, she'd like to receive an entire download of a file hundreds of kilobytes long in a little more than one round-trip time (after the initial three-way handshake), despite TCP's congestion control algorithm. She tells you she has implemented an "enhanced" TCP receiver, and shows you the following protocol time diagram, which begins immediately after her connection's 3-way handshake:



(In the diagram, Alyssa's TCP receiver sends the three TCP ACKs shown back-to-back in time, one immediately after the other.)

Given the details of TCP's behavior stated at the beginning of this question, what will the protocol time diagram look like after the sender receives this traffic from Alyssa's TCP receiver?

[5 marks]

(b) Based on the protocol time diagram Alyssa has shown you, state a general technique that a TCP receiver can use to trick a TCP sender into sending very quickly (far more quickly than TCP congestion control allows). Assume that the network path between the server and Alyssa's machine is of sufficient capacity to carry a burst of packets of arbitrary length from the server without dropping any packets.

[3 marks]

(c) How would you change the TCP sender's implementation to defend against the anti-social behavior of Alyssa's TCP sender?

[2 marks]

4. Suppose an attacker sends many SYN packets (*i.e.,* the first packet in TCP's usual 3-way handshake) to a server, all to the same destination port, but each with a different source port. He sends one such packet every 10 ms, and continues this behavior for hours. He never sends any packets apart from these SYN packets (*i.e.,* he does not attempt to pursue any of these handshakes beyond this first step). To hide his own identity, he chooses an unused IP address from elsewhere on the Internet, and sets the source IP address on each of these packets to be this unused IP address.

   A typical operating system (OS) implements the server side of TCP connection establishment as follows:

   When an application on a server asks the OS to listen for incoming TCP connections to a port, the OS allocates a *listen queue* for that TCP port. Each entry in this queue stores state about one incoming connection that has begun, but not yet completed, the full 3-way handshake. This queue is typically fairly short: it's only on the order of ten entries long. While the queue is full, no new incoming connection requests to the port can be processed, because there's nowhere to record state about the progress of the new incoming connection request.

   Entries in this queue are freed in one of two ways:

   - when a connection completes the 3-way handshake successfully, OR
   - when the connection fails to complete the 3-way handshake within a long timeout period (on the order of minutes, to allow any severely delayed messages from the remote host to arrive)

   (a) What effect will the attacker's behavior have on legitimate users who wish to connect to the server at the same destination port that is under attack? Refer specifically to the details of the 3-way handshake used by TCP and any relevant aspects of the server's handshake implementation described above in your answer.

   Assume that no packets are dropped in the middle of the network. Assume that the total rate at which legitimate users attempt to connect to the server at the destination port is far less than once every 10 ms.

   [2 marks]

   (b) Q. D. Fender proposes a change (described below) to the server's OS implementation of TCP's three-way handshake. He doesn't change the TCP packet format, nor the OS's TCP code on clients. Your task in this part of the question is to decide whether Q. D.'s 3-way handshake implementation behaves any better than the original implementation described in the previous part of this question, and why or why not.

   Q. D.'s scheme makes use of a *cryptographic hash function,* $y = H(x)$. For the purposes of this question, $H()$ has the following properties:

   - The length of the output given by $H()$ is 24 bits, regardless of the length of the input to $H()$.

- Given the output of $H(x)$, it's computationally infeasible to determine x.
- The input x to $H(x)$ can be of arbitrary length.
- We denote the concatenation of data items by a vertical bar |, such that, *e.g.*, $H(a|b|c)$ refers to computing $H()$ on the entire concatenation of a to b to c.

Q. D.'s scheme is implemented at the receiver, R, which wishes to accept incoming TCP connections to destination port p. It works as follows:

- *R does not have any queue* to hold state for connections in progress.
- Suppose R receives the first packet in the three-way handshake from S, and that S's TCP source port is q. R replies to S with a packet whose sequence number is the concatenation of the following pieces of data:
  - first 8 bits: t, a time counter; t increments once every 64 seconds on R, and wraps from 255 back to 0
  - next 24 bits: $H(S|q|R|p|t|x)$, where x is a secret 16-byte string chosen randomly by the server R each time the server R reboots, and all other variables are as defined above
- When R receives the next packet from S in the three-way handshake, with ACK number a, and with TCP source port q and TCP destination port p, it does as follows:
  - Let $b = a - 1$
  - set t_{ACK} to top 8 bits of b, and y to low 24 bits of b
  - if $t - 5 <= t_{\text{ACK}} <= t$, and $y == H(S|q|R|p|t_{\text{ACK}}|x)$, record the connection from S as ESTABLISHED, and notify the appropriate listening application running on R of the new connection
  - otherwise, ignore the packet received; do not record any new connection

Assume that the table for ESTABLISHED connections is very large, and never fills up.

i. How will legitimate (non-source-address-spoofed) connection requests be processed by Q. D.'s scheme? Be specific: show how the three-way handshake proceeds with Q. D.'s scheme running on the receiver, with reference to the relevant TCP header fields.

[4 marks]

ii. How will source-address-spoofed connection requests like those discussed in part (b) of this question be processed by Q. D.'s scheme? Again, refer to the relevant TCP header fields during the handshake in your answer.

[4 marks]

Problem set total: 40 marks