

Transport Layer

NOTE: This assignment is due on a Friday, so you may ask for help on it after Wednesday's class, unlike previous assignments. However, there is no automatic extension on this assignment, so any assignments not handed in by the start of class on November 17th will receive no credit.

1. Demultiplexing

Out of the packets below, show which ones will be delivered to the same socket.

#	Source IP	Source Port	Dest IP	Dest Port	Protocol
1	1.2.3.4	1234	5.6.7.8	5678	UDP
2	1.2.3.4	4321	5.6.7.8	5678	UDP
3	1.2.3.4	1234	5.6.7.8	8765	UDP
4	1.2.3.4	1234	5.6.7.8	5678	TCP
5	1.2.3.4	4321	5.6.7.8	5678	TCP
6	1.2.3.4	4321	5.6.7.8	8765	TCP
7	1.2.3.5	4321	5.6.7.8	8765	TCP

2. Reliable Data Transfer

- Show how rdt3.0 can become unreliable if the network reorders packets. In particular, show a sequence of messages, states, and network loss / delay events in a diagram similar to Figure 3.16 (p213) of the textbook.
- Notice that when the sender of rdt3.0 receives a "duplicate ACK" for the last packet, this ACK is ignored. Suppose that instead, rdt3.0 implemented a version of Fast Retransmit and resent the packet whenever it received such an ACK. I.e. it would include the following transition rule in state "Wait for ACK0":

$$\frac{\text{rdtrcv(rcvpkt) \&\& notcorrupt(rcvpkt) \&\& isACK(rcvpkt,1)}}{\text{udtsend(sendpkt), start timer}}$$

Show how with this modification, a single delayed ACK from the receiver can result in unnecessary retransmissions of subsequent packets.

3. Slow Start

Assume a connection with RTT=100ms, MSS=1460 bytes. Ignoring overhead spent on headers, calculate the transfer time and the effective throughput for transferring a 500 KB file. Assume there are no losses and that both the connection bandwidth and the receiver window are infinite (and in particular, not limited to 64K).

- Calculate the transfer time for TCP using slow start
- Calculate the transfer time supposing TCP used additive increase, starting with a window size of 1 MSS and increasing it by 1 MSS every RTT.

4. Fairness

Suppose there are two connections, A and B, sharing a 10 Kbit link. Let us simplify the TCP congestion control protocol to include the following assumptions:

- 1 MSS = 1000 bits
 - the two connections increase their window sizes by 1 MSS per RTT, in lock-step, whenever their aggregate bandwidth is less than or equal to 10 Kbits
 - when the aggregate bandwidth is greater than 10 Kbits, both connections simultaneously decrease their window size to 1/2 of the previous size. The window is rounded up to the next MSS size.
- (a) Suppose A's window is currently 9000 bits and B's window is 1000 bits, and both their RTTs are 1s. Show how the window sizes change after 20s
- (b) Suppose A and B both have a window of 1000 bits, A has RTT of 1s and B has RTT of 2s. Show how the window sizes change after 20s. (Assume that whenever the aggregate bandwidth exceeds 10 Kbit, the next window of each of the connection will be cut in half.)

5. RTT estimation

You may want to use a spreadsheet or a computer program to calculate the answers to this question.

Recall that the RTT estimation algorithm is:

$$RTTEstimate = RTTEstimate \cdot (1 - \alpha) + RTTSample \cdot \alpha$$

Suppose that a TCP connection has experienced a constant RTT of 500ms, so that $RTTEstimate$ is now 500ms. Now suppose the RTT drops to 200ms for the next N measurements, and then returns to 500ms.

- (a) Consider the following strategy for setting the timeout:

$$Timeout = 2 \cdot RTTEstimate$$

What is the smallest value of N so that a premature timeout will occur when the RTT returns to 500ms? Let $\alpha = 0.1$ in this case.

- (b) Now suppose the timeout is set using a deviation estimate:

$$DeviationEstimate = DeviationEstimate \cdot (1 - \beta) + |RTTSample - RTTEstimate| \cdot \beta$$

$$Timeout = RTTEstimate + 4 \cdot DeviationEstimate$$

with $\beta = 0.25$. What will the timeout value be after N measurements of 200ms, using the value of N from the previous question? (Assume that $DeviationEstimate$ is 0 before the RTT drops to 200ms.)