Extra Credit Project 1: TCP

Document Version 2.0

UPDATE REGARDING TESTING:
As I said in class, there will be no test box to test your TCP implementation, and you do not have to interoperate with other TCPs (though I'd give you some extra credit if you do). You have to develop your own test scenarios, and then take me through a demo that convinces me that your TCP works and performs well. I might suggest essentially the test configuration I show in the figure below, but you must implement the test box (i.e. traffic generator).
In the demo, you need to convince me that reliability works, and you need to convince me that your TCPs backoff correctly and do fair sharing. One way is to produce a graph which shows the sequence numbers for a number of flows. You should setup dummynet to constrict your throughput (and therefore drop packets from time to time).

NOTE THE TEXT BELOW IS NOT CHANGED FROM THE ORIGINAL POSTING, SO THE ABOVE PARAGRAPHS SUPERCEDE ANYTHING BELOW.

Worth 4 MEng credits. Must be done as a single-person project.

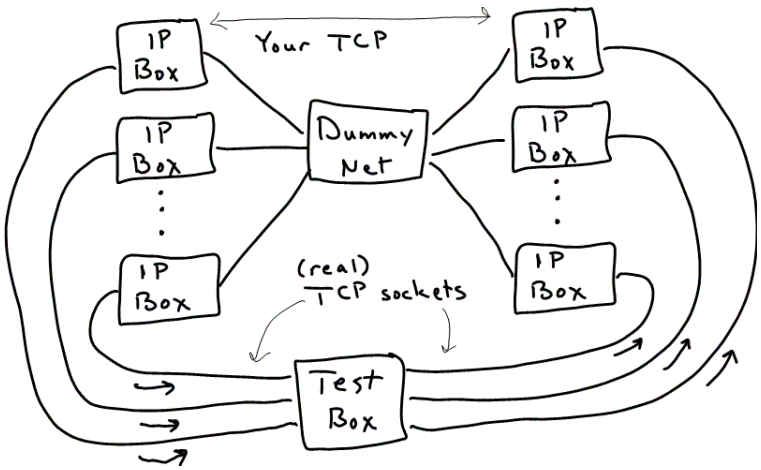
Goal: To implement a TCP layer above UDP

Task: You must build a near-fully functional TCP (RFC 793) layer, including the initial 3-way handshake, window flow control, sequencing (in order packet delivery), congestion control (including proper RTT estimation), and connection termination. Your congestion control must include slow start, congestion avoidance, fast retransmit and fast recovery (RFC 2581, sections 3.1 and 3.2). Your implementation is not required to have SACK (selective acknowledgement, RFC 2018), though it may. If you do implement SACK, you must be backwards compatible with non-SACK implementations.

You will implement TCP directly over the IP you wrote for project 2.

We will test your TCP by running it through a "dummynet" box. This is a box that emulates various network characteristics, like lost packets and delay. We will also expect your TCP to interoperate with other student's TCP. Specifically, we will put you in "testing groups" of three or four. You will be expected to insure that your implementation interoperates with the other implementations in your testing group (but you must not share code). The testing groups will be big enough that a bad implementation can be detected (for instance, if there are four implementations, and three work together but a fourth fails to work with the other three, then the bad one can be set aside).

We will test you by sending packets through your TCP implementations as shown below.



The IP boxes will run TCP connections with peer IP boxes via a dummy net. The IP boxes will establish TCP sockets with the test box. The Test box will send data over the TCP sockets to the IP boxes on the right, which will take this data and insert it into your TCP implementations. The IP boxes on the left will receive the data (possibly after retransmission because of losses induced by the dummynet) and feed the received data into the TCP sockets back to the Test box. Because the receiving IP Boxes will exert flow control on the sending IP boxes, the Test box cannot send into the IP boxes at full rate. The IP boxes will flow control the Test box by reading from the TCP socket only as fast as it can send into the TCP connection with the peer IP box.

The Test box will look for two things: First, that it receives from the left the same data it sent on the right. Second, it will expect to see fair sharing across the dummynet box. In other words, each of the data streams it sends in to the IP boxes should be sent at about the same rate.

Note: Further details will be released soon. This, however, should be enough to get you started.