CS348 Computer Networks
Lab Excercises 6

Indian Institute of Technology, Patna

March 14, 2016

Instructions: For this assignment you have to use the simulator that you have developed for assignments 1 and 2. You have to submit the codes in a tgz file with name labAssign6.tgz. The submission date is 20.03.2016.

Problem

Consider a packet switched network that has n sources, all sending packets (at a same fixed rate) to a common sink through a single switch. The system uses a TCP based transport protocol. The buffer sizes of the *switch* and *sink* are Q_s and Q_r (packets) respectively. Assume that the buffers at the *sources* are theoretically infinite sized. The RTT values of the sources depend on their link bandwidth to the sink and can vary for each source. Further, assume that packets from source to the sink can get dropped randomly with a probability p. You have to implement the following two TCP protocols:

1. AIMD: Assume that packets are acknowledged by their IDs and not by bytes as is done in TCP. In the acknowledgement, the switch sends the ID of the next expected packet in sequence. Whenever a duplicate ack is received by a source, it reduces its cwnd to half and then increases linearly following the AIMD protocol.

2. CUBIC: Assume the parameters C and β on your own. The time elapsed is the difference between the last window reduction time and the current time.

For both these cases, start with a window size of 1. You need not use any timeout value as dropped packets are detected by duplicate acks. Plot the following graphs for 2 cases, (a) both Q_s and $Q_r \to \infty$ and (b) they are finite but of same size. Further assume that there are 2 sources only

1. The window size of the sources with respect to time.

2. The average throughput of the sources with respect to the loss rate p.

3. The average throughput ratio of the sources with respect to the ratio of their average RTT values.

[**Important**] Design your code in such a way that you can implement as many TCP protocols as you want without touching the lower level functionalities.