1 Introduction

This assignment is to implement reliable packet delivery protocol on top of UDP. The protocol should ensure reliable delivery of packets given that the types of errors are:

- Lost packets,
- Out-of-order delivery of packets,
- Corrupted packets, and
- Duplicate packets.

Assuming that the network is able to deliver *some* packets.

1.1 Part I

Since the network you’d be using will probably be fairly reliable, you will need to create your own faults. To do this you will write two procedures:

**sendUdp** does the actual sending of UDP packets and in addition will drop packets, corrupt packets, reorder packets, and duplicate packets. The packet errors can come from a file, specifying what’s to be done, or generated via probabilities and using a random number generator.

**receiveUdp** receives a UDP packet.

Here the **sendUdp** routine is part of the sender and the **receiveUdp** routine is part of the receiver.

In addition to the above, the first part of the assignment will also require the program:
• to set an alarm (on the sender side) for each packet,
• to determine when a packet did not return within the time period,
• to detect what happened to the packet, and
• to acknowledge packets back to sender.

The purpose of this code is to generate the packet errors (on the sender side) and then
determine what happened (on the receiver side): Recovering from the errors happens in
Part II.

After the packet is sent the sender waits $T$ time for packet to be acknowledge, and then
sets off an alarm. There should be two ways to determine $T$:

• The round-trip-time plus six times the standard deviation.
• Alternatively, the alarm should be set for a fixed time.

Note that this part is an attempt to deal with random events, both timing and packet
errors, in two ways: A deterministic mode to enable testing and a random mode to mimic
the real world.

2 Part II

You are to implement selective repeat for a window size of $W$ packets, which is a parameter.
You should produce:

• simulation scripts to test out your code,
• stress testing results, and
• documentation about 1) how your program is designed, 2) how your program is tested,
  and 3) what does and doesn’t work.

3 turnin

The turnin command is:

$ turnin -c cs450 -p a2_part# <cs-login>

The part# refers to the part number of the assignment (part1 or part2). To turnin your
source and README files, store them in a directory with the same name as your cs login
and turnin the entire directory. Also make sure you include a makefile that creates two
executable named sender_part# and receiver_part# (# is either 1 or 2). (And please do
not turnin any binaries or object code).