Note

Please ask your lab demonstrator about any questions you have regarding previous labs. It will be beneficial for you to have a good foundation early on in the semester, and we are here to help!

Please familiarise yourself with the cnet website at http://www.csse.uwa.edu.au/cnet/.There is an introduction on the homepage that will help you with the initial tasks.If you are running cnet on the university iMacs, you may need to prepare an alias for the executable, which is located at /Volumes/cslinux/bin/cnet

Task 1: (files required: stopandwait.c, PIGGYBACK)

For this exercise, using the cnet network simulator:

Add piggybacking to the standard stop-and-wait protocol.
Modify the topology file, changing the attributes of message delivery, bandwidth, frame corruption and loss, to determine under which conditions piggybacking offers an advantage.

Remember, to perform a valid scientific experiment (which, after all, is what we're doing), we need to hold all conditions/attributes constant and vary just one over time, perform the same experiment for both the stop-and-wait and piggybacking protocols.

We will be asking questions of a similar format to Labsheet 2. If you haven't completed that lab yet please do so now.

Run the following questions with the seed value of 19, and using 90 seconds for the STOPANDWAIT file. The -T command for cnet should speed things up for you.

As in Labsheet 2, develop a plot to help you to visually compare the results of your experiment. The easiest approach is to modify the plot-to-html.sh shellscript from Labsheet 2 to plot multiple curves on the same graph (you may like to learn about the 'join' program to merge the results of the two simulations).

Task 2: (files required: stopandwait.c, SLIDINGWIN)

Sliding-window protocols are Data Link Layer protocols that support multiple outstanding frames between sender and receiver. Again, the goal is to minimize the waiting time required in the standard stop-and-wait protocol.

In general, sliding-window protocols have the properties that:

- The sender has a sending window consisting of a sequence (an array) of frames that have been sent but not acknowledged.
- The sender's window size grows as more frames are sent but not yet acknowledged.
- The receiver has a receiving window consisting of frames it is willing to accept. The receiver's window size remains constant.
- Each frame being sent has a sequence number from 0 to 2n-1 (which fits in >n bits). Stop-and-wait has n=1.
- A window size of 1 implies that frames are only accepted in order, and thus an error is detected if an expected frame is missed (a 'later' frame arrives first).

The simplest variant of a sliding-window protocol is termed the go-back-N protocol, in which the receiver simply discards (ignores) all frames after a bad one. This corresponds to a receiver window size of 1.

After it learns of a lost (or corrupted frame), the sender retransmits all unacknowledged frames - it "goes back N frames" and starts again. Notice that in the presence of significant errors that bandwidth is wasted because the receiver only 'buffers' a single frame.

Following the same experimental approach as in the first exercise, use cnet to implement the go-back-N protocol, and determine under which conditions it is an improvement over the stop-and-wait protocol.