# Assignment 2

# 3P01

**{Due Date Mar.23 2015, 4:00. Late date Mar.26 2015, 4:00}**

**Objective:**

Using Sockets, open a UDP connections between 1 source and 1 receiver. To help secure the data integrity of the connection, with will add flow control and error detection.

Assignment:

**Part A**

Using the socket structure in the book and from A1, create a UDP connection between a source (S) and receiver (R). Set a sufficient buffer size so that 1 can type in a small message and send it to R.

UDP has very little in the way of flow control or error correction and is uni-directional.

Have a reverse connection created from R to S, with sufficient buffer capacity. This should also be a UDP connection. Test that you can send messages in both directions.

In short, both S and R should have the capacity to send and receive UDP messages on separate socket connection.

**Part B**

Consider implementing flow control with the above connections. Let us limit a frame to 8 characters. Thus to send a large message, it would be broken into 8 character chunks and sent.

S will always be sending to R, R will use the reverse channel to confirm the flow control.

Write a second version of the Part A implementation which will allow the user to input a text message of some sufficient length (256 chars max). The sender will break the message into 8 character frames. Each frame will contain a header in the following format [#], where # is the sequence number of the frame. When R receives a frame, it will look at the sequence number and determine if it is ready to accept more data. For our purpose, R will receive up to 5 frames before an ACK is generated.

Consider flow control with go back N, in our case let us assume a sequence from 0-4 of the frames which are sent. Thus the sliding window system can support up to 5 frames at any one time. S can send 5 consecutive frames before it must wait for a response from R. Once R receives 5 frames it will respond to S with [#] indicating said frames have been received, and it awaits more.

R will assemble the complete message sent by S and display it to the user.

**Part C**

Extend Part B to include error checking. For our purposes let us assume that when a frame is received at R it has a 10% chance of being corrupt. After R has receive 5 frames, it will either respond back with [#] indicating to S everything is good or {#} indicating frame # us corrupt and must be resent. Since we are implementing go back N, this implies that the offending frame and all subsequent ones may need to be resent. As well, any additional capacity in the window can be used to send new frames.

## Submission

1. Completed and properly documented source code for R and S for all 3 parts.
2. Hard copy to be submitted to the assignment drop box across from J333. Include a completed coverpage.
3. Electronic copy to be submitted: <submit c3p01a> on sandcastle.

db