

Problem #2 UDP Sockets

UDP Server (60 points)

Write a console program that takes as a **command line argument** the **port number** on which the UDP Server will receive messages. The UDP server collects parts of a larger **composite message** from the clients. The UDP server collects these message parts from different clients and assembles them in order, based on the `lSeqNum`. The UDP server keeps a running list of all clients that have sent a message to it and will broadcast the composite message to all clients when commanded. The UDP server needs to be able to receive data from clients without blocking the main application thread. The program needs to respond to user input while handling socket communications at the same time.

The program should continuously prompt the user for commands to execute like so:

Please enter command: 0

Please enter command: 1

Please enter command: 2

Composite Msg: *current composite message*

The data being sent back and forth will use the following packet structure:

```
struct udpMessage
{
    unsigned char  nVersion;
    unsigned char  nType;
    unsigned short nMsgLen;
    unsigned long  lSeqNum;
    char chMsg[1000];
};
```

The UDP server needs to have the following functionality from the **command prompt**:

- If **command** == 0 the server immediately sends to all clients the current composite message and clears out the composite message.
- If **command** == 1 the server immediately clears out the composite message.
- If **command** == 2 the server immediately displays to the console the composite message but takes no other actions.

The UDP server needs to have the following functionality when **receiving messages**:

- If **nVersion** is not equal to 1 then the message is ignored.

- If **nType == 0** the composite message is immediately cleared and anything in the chMsg buffer is ignored.
- If **nType == 1** the composite message is immediately cleared and the message in chMsg is used as the start of a new composite message
- If **nType == 2** the message in chMsg is added to the composite message based on its **lSeqNum**
- If **nType == 3** the server immediately sends to all clients the current composite message and clears out the composite message.

If the composite message becomes larger than 1000 then the composite message (up to 1000) is immediately set out to all clients and any remaining characters are used to start a new composite message with a `lSeqNum = 0`.

UDP Client (40 points)

In order to test your server, write a console program that takes as a **command line argument** the **IP Address** and **port number** of the server as shown below:

./a.out localhost 51717

The program should prompt the user for inputs and display any messages received.

Here are example user inputs:

Please enter command: v # the user enters a "v", a space, and then a version number. This version number is now used in all new messages.

Please enter command: t # # message string the user enters a "t", a space, and then a **type** number, and then a **sequence** number, followed by the **message** (if any). Be sure you are able to handle the spaces in the message.

Please enter command: q the user enters a "q" causes the socket to be closed and the program to terminate.

Any messages received should be displayed as followed:

Received Msg Type: 1, Seq: 33, Msg: *message received*