Lab8 (100 points)

Important: We will demo in class or online. For in class demos, we will go around the room. Be prepared to demo at the start of class. The submitted code will be used only to verify that you did not copy from others, to compile and re-run your program, to make sure you were indeed demonstrating your own code, and to grade for documentation of your code.

In this program we will start with Lab7. We are going to introduce several new concepts.

First, you will store messages you receive and forward (remembering the rules about forwarding from lab 8). You will use a TTL counter. When you get a message, the initial TTL will be 5. Periodically, every 20 seconds, you will resend any stored messages and decrement the TTL field. When the field reaches 0, you delete it and stop resending it. Do not store duplicate messages at the forwarding node. Duplicate message is same seq#, same toPort, same fromPort.

If you move to a new square, you will immediately resend all messages, and decrement their TTL fields. This includes messages where you are the originator and messages where you are the forwarder.

struct timeval timeout;

```
   timeout.tv_sec = 20;
   timeout.tv_usec = 0;
   rc = select (maxSD+1, &socketFDS, NULL, NULL, &timeout);
#ifdef DEBUG
   printf ("select popped rc is %d\n", rc);
#endif
   if (rc == 0){ // had a timeout!
     reSend(&messages,sd, Partners, myPort, myLoc);
     printf ("timeout!\n");
     continue;
   }
```

Optionally, if you see an ACK for a message, whether you are the originator or the forwarder, you will remove that message from the list of stored messages so that you will not re-forward it later.

Submit well-documented and well indented code along with a README file explaining how to run the program, and a makefile. Submit it using GitHub

The grading rubric is as follows:

- Program correctness and robustness (what happens if I give garbage input): 80%
- Coding style (comments, indentations, README, Makefile): 20%