UNIVERSITY OF DELAWARE
DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

CISC 650 / CPEG 651 / ELEG 651: Computer Networks II

Fall Semester, 2017 Professor: Adarsh Sethi

PROGRAMMING PROJECT 2
Some Helpful Hints

Below are some helpful hints and suggestions for the Programming Project 2:

1. Look at the "Helpful Hints" document for Programming Project 1, since information provided there is not being repeated here.

2. All programs (clients and servers) for this project must be run on the *cisc650* VM. This is necessary if you are to discover and communicate with programs written by other students in the class.

3. For this project, each message (as described in the project specification) should be transmitted as a single message using only one invocation of the *send* command. It is not permissible to send each field of the message in separate *send* commands. Also, you must conform to the message format exactly as described. If you don't, your programs will not inter-operate with those of the other students.

4. Since the client needs to communicate with both a TCP server and a UDP server, you will need to have two client sockets, a TCP socket and a UDP socket. Similarly, you may need separate structures for the TCP and UDP server addresses.

5. The project specification requires the client to maintain a *ClientInfo* file in which information received from servers is stored. When the client selects a new port number to try and discover if there are servers running at this port, it should first check if this port number already appears in the *ClientInfo* file. If it does, that port number should be skipped, and the client should proceed to the next port number.

6. In maintaining the *ClientInfo* and *ServerInfo* files, make sure that the information in these files is preserved (a) after each new record is written to the files, and (b) between different executions of the client/server. Thus, each time a new record is written to any of the files, you should close the file and then later open it again when needed. Also, after the client is finished scanning all the ports, it terminates. When you run the client again at a later time, make sure it does not overwrite or destroy the existing *ClientInfo* file. On the other hand, the server is expected to run continuously until you kill it, perhaps for a period of a few days. But if the server is killed inadvertently for some reason, then you may have to re-start it. At this point, make sure the existing *ServerInfo* file is not overwritten or destroyed.

7. To run the TCP server in the background, use the command:

```
./tcpserver&
```

   where *tcpserver* may be the name of your server file. Note the ampersand at the end. This will only work if the server does not print anything to its standard output. You may then start the UDP server in the background in the same manner. The advantage of running the servers in the background is that you can now terminate the window from which the servers were started and the servers will continue running.

8. After you start a server in the background, you will see the job number of your server execution displayed, for instance:

```
[1] 2682
```

   Here the job number is 2682. To terminate the server execution, use the command:

```
kill 2682
```

9. If you have closed the window from which the server was started, but later want to terminate the server, open a new window and give the kill command if you remember the job number.

10. If you don't remember the job number, you can find it from a new window with the command *ps x*. This will display a list of all running processes owned by you, irrespective of the terminal window in which they were started. You can then note the job number of the process that you wish to kill and then give the kill command as above. In fact, it is a good idea to periodically check for all running processes owned by you and kill any that you don't need any more.