CptS/EE 555
Project #1 – Socket Programming

Instructor: Adam Hahn
Due: 02/8/2016 at 12:10 pm

Deliverable:

Submit the code you developed for this project and include any information required for me to compile and run it. Please email both the client and server to ahahn@eecs.wsu.edu before the assigned due date. Please follow these guidelines when submitting:

-Put your name on the assignment.
-Begin you email subject with: `[CPTS/EE 555]`
-Include the client, server, and output file

Test System:

Run your assignment in the `mininet` platform. Your system will run on a basic topology with two hosts, `h1 (IP: 10.0.0.1)`, and `h2 (IP: 10.0.0.2)`, which communicate through a switch, `s1`.

1. Open three different terminal windows.
2. Start `mininet` in terminal #1 with the following command:

```
$ sudo mn -c
$ sudo mn --mac --switch ovsk  --controller remote
```

3. Now, start the POX SDN controller in a terminal #2 utilizing the custom controller program, `proj1_555.py`. Download the controller and copy into the correct directory:

```
$ wget http://eecs.wsu.edu/~ahahn/555/proj1/proj1_555.py
$ mv proj1_555.py  ~/pox/pox/misc/
```

Move to the "pox" directory and start the POX controller

```
$ cd pox
$ ./pox.py log.level --DEBUG misc.proj1_555
```

Verify that the controller connects to `mininet`, you should see the following output message:

```
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
```

4. Download the template programs (`client_udp.c, server_udp.c`) and test file (`tux.txt`) into the virtual machine.

```
$ wget http://eecs.wsu.edu/~ahahn/555/proj1/server_udp.c
$ wget http://eecs.wsu.edu/~ahahn/555/proj1/client_udp.c
$ wget http://eecs.wsu.edu/~ahahn/555/proj1/tux.txt
```

5. Compile the template programs and verify that the environment works correctly. Compile the code in the terminal #3 with the following commands:

```
$ gcc –o client_udp client_udp.c
$ gcc –o server_udp server_udp.c
```

6. Now go back to terminal #1 and run the compiled client and server code

```
mininet> h2 ./server_udp output.txt &
mininet> h1 ./client_udp 10.0.0.2 tux.txt
```

7. In terminal 3, view the newly created `ouput.txt` file. You should see an ASCII penguin similar to that in `tux.txt`, except with many lines missing.

Assignment:

In class we discussed methods to perform reliable communication over unreliable channels. The project will explore how to utilize C sockets to implement a reliable communication over a simulated unreliable link. Implement the *stop-and-wait* protocol as discussed in class (or Section 1.3.1 in Marsic) in C to reliably send a file between a client and server using UDP.

The template code will unreliably transmit a file from the client to the server. You will need to add acknowledge packets from the server to the clients, which are also subject to reliability issues.

You may want to implement sockets that only block for a short period of time, therefore allowing you to resend a lost data frames. To do so, please utilize the `setsockopt()` function which enabled you to set the amount of time the socket will block for during a `recvfrom()` call.

```
struct timeval tv;
tv.tv_sec = 1;
tv.tv_usec = 0;

…

if (setsockopt(s,SOL_SOCKET, SO_RCVTIMEO,&tv,sizeof(tv)) < 0) {
      perror("PError");
```

```
        }
```