College of Science
& Engineering

COMP2881 Operating Systems, DE
COMP9881 Operating Systems GE
Semester 1, 2020

CN Workshop06: Working with IP/UDP

Due: (check submission time on FLO)

Lab 6 is comprised of 2 tasks. This workshop contributes 10% to your final grade. Following completion of the checkpoints, a copy of the appropriate, makefile, source code and/or output for each task should be compiled into a single text file and uploaded to the relevant submission box on FLO for moderation. Speak to your demonstrator if you are unsure how to do this. While discussion of programming tasks with fellow students in labs is welcome and encouraged, please be mindful of the University's **Academic Integrity** policy and refrain from simply copying another student's code (even if they agree!). You will gain a much greater understanding of the material if you take the time to work through your own solutions, and you will be expected to be able to explain your code before being awarded checkpoints. Remember, the demonstrators are there to help if you get stuck.

Objectives

Upon completing this lab, you should be able to:
- Demonstrate an understanding of Computer Network concepts; IP addresses, sockets, simplex vs duplex connections, network protocols
- Demonstrate you are able to develop programs in the UNIX environment in C
- Demonstrate you are able to articulate user program design choices in terms of; security, efficiency, performance, or other relevant requirements

Learning out comes;
- LO1: Understand computer network terminology and topologies, functions and architectures of computer networks
- LO2: Demonstrate the design and implementation of simple process and network level programs
- LO3: Demonstrate an understanding of the various layers which make up networks and how they are used to form complex, reliable and secure communication mediums
- LO4: Understand and be able to articulate the structure and function of the major types of networks in common use
- LO5: Understand aspects of the internet, including various associated protocols IP, UDP, TCP, IP addressing, header fields and the operation of servers / clients
- LO6: Appreciate the real-time nature of networked devices and communications and demonstrate an understanding of current and emerging methods for secure information transfer

Preparation

Read through the entire workshop before you begin.

You may need to use tcpdump and/or wireshark to inspect the packets in flight over the wire/air.

Down load the example IP/UDP source code to use as a starting point;

> **https://github.com/amineamanzou/UDP-TCP-File-Transfer**

Compile the UDP client and server programs to make sure they work in your environment. As ports 0-1024 need root user access, select a port number above 1024, so a regular user can run a network service to listen. Check `/etc/services` for unused ports.

Download the CTAP source code an use the `ctap.h` file in your client program;

> **https://github.com/jhunt/ctap**

Read through the `README.md` file and make note of functions available.

Explore generating random digits with the `rand()` function;

> **https://www.tutorialspoint.com/c_standard_library/c_function_rand.htm**

Explore using `sprintf()` to print the characters into a string;

> **https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm**

Read documentation as appropriate;

> **http://cnp3book.info.ucl.ac.be/2nd/cnp3bis.pdf**
> **https://beej.us/guide/bgnet/pdf/bgnet_A4_2.pdf**

Task 1

Modify the provided IP/UDP server code to provide a service like ping. The purpose of this service is the echo back the data provided over a datagram service. Document each step in your implementation, marks will be allocated to informative commentary on what the code is doing.

The server should;

- Use a port number provided by an argument to `getopt()`, check it is between 10000-59999
- Create a socket on the provided port number
- Listen on the socket
- Wait to accept incoming requests
- Respond by echoing back the data received in a single datagram
- Close the connection to the client
- Sleep for one second
- Start the process again

This program should be run on the command line and listen on the localhost IP or the IP of the available network interface. The server should print out the details of the IP and port number when it starts.

Example;

```
$ udpserver -p 23456
Listening on IP: 127.0.0.1 and port: 23456
Connection: “1”, data received: “28190359”
Connection: “2”, data received: “11738806”
```

Task 2

Modify the provided IP/UDP client code to provide a test program to test the above server. While test are supposed to be self documenting, please use meaningful messages in your tests.

- Create appropriate unit tests using CTAP to;
  - TEST ok: Get the start time
  - TEST ok: Create a client connection on a port and IP address, stored in global variables
  - TEST note: Display pay load, eight random characters
  - TEST ok: Send a request to the server with payload
  - TEST ok: Wait for the response from the server
  - TEST ok: Check the response payload from the server matches the eight characters sent
  - TEST note: Display response pay load
  - TEST ok: Get the finish time
  - TEST note: Calculate and display the start, finish and time taken to complete the network connection
  - TEST ok: Check that the time taken is less than one second

The goal here is to write a program that checks that the server is responding to the request and that it returns data that matches the data we send to it from our datagram. The client should be run on the command line in a separate terminal and provide meaningful output for each test.

For more information on see;

https://beej.us/guide/bgnet/pdf/bgnet_A4_2.pdf