Lab 5
Sockets: Master/Slave Voting

Objective

In this lab, students will learn how to perform IP communication and synchronization in a distributed environment. This objective will be achieved by:

- Creating and using sockets for communication
- Creating a Master/Slave setup

Prelab

In the prelab you should briefly explain the functions needed to:

- create a socket
- write to a socket
- read from a socket
- bind a socket
- change socket permissions to allow broadcast

Do not forget to include the pseudo code or flow chart of the program.

Lab Procedure

In a master/slave setup, one device has control over one or more other devices. An example of this is a network of computers, where one computer (master) receives tasks and assigns them to other computers (slaves) based on certain criteria. Usually, if no master device is present, the devices use an election to determine a new master.

For this lab, each student will implement a server using their Raspberry Pi board. Each board will start with a slave status. A client program can ask all the student's boards which one of them is the master by sending the message "WHOIS". If no board replies that it is the master, the client program can then ask the boards to vote on a new master by sending the message "VOTE". To vote, each board will send a broadcast message to all of the other boards that begins with a # sign and contains the board's IP address followed by a space followed by a random number that it generated (e.g.: "# 128.206.19.20 4"). After receiving the votes, each server program must decide if it will become the master or not. This is done by comparing its own vote with the other votes received; the highest vote wins. If a tie occurs, the board with the highest IP wins. If the client program sends another WHOIS message, the board who is the new master will send a message to the client program saying that it is the master. This message should include the student's name and the board's IP (e.g. "Peter on board 128.206.19.11 is the master"). **Be sure that the string you send to the client is less than or equal to 40 characters or you will get weird characters when printing.**

The server should also disregard any invalid message.

Requirements:

1.) Each board must determine its IP dynamically (you cannot hard code the IP address in).
2.) The messages will always be character arrays of size 40.
3.) Votes must be randomly generated integers in the range [1, 10].
4.) The port used for communication should be an argument of your program, i.e., when you run your program, you should be able to provide the port number. By default, port 2000 will be used.

Hints:

A few example client and server programs are provided in the file **ece4220lab5.zip**. You can find it on Canvas under *Modules > Additional Lab Files*. The TA will give you some more details about those programs. Also included in the zip file are two binaries, **`Lab5_client`** and **`Lab5_client_WS`**. They are the same program, only compiled for the RPi and the workstation, respectively. This program is a client program that broadcasts whatever the user types (including the 'return' character when you press Enter), and prints out messages that are sent to it. This program will be used to test your server program on the post-lab due date. You should use it to test how your board responds to the different messages and the tie breaker.

To run `Lab5_client`, make sure you log into another RPi board, so you have one board running the client and another one running your server program (you can ask a lab mate that may be working in the lab if you can log in to their RPi). You can actually run many instances of your server program on different boards, to test the different cases. If you do not have access to another RPi, you can run `Lab5_client_WS` on the workstation.

You should test your server program with other students' server programs running at the same time. That is how the TA will test the day of the presentation.