EE 450
Lab2: Socket Programming

Summer 2018 Nazarian

Student ID: _____ Name: _____
Assigned: June 1st
Due: Wednesday, June 13th, at 11:59pm.
Maximum points: 100
Late submissions will be accepted only in the first two days after deadline with a maximum penalty of 15% per day: For each day, submissions between 12 and 1am: 2%, 1 and 2am: 4%, 2 and 3am: 8% and after 3am: 15%.

- **Lab2 is based on individual work. No collaboration is allowed. We may pick some students in random to demonstrate their design and simulations. Please refer to the syllabus for a summary of AI policies (including the penalties for any violation.) If you have any doubts about what is allowed or prohibited in this course, please contact the instructor.**

A. Problem Description

The Dynamic Host Configuration Protocol (**DHCP**) allows a host to obtain (be allocated) an IP address automatically.
DHCP is a client-server protocol, i.e., one runs on client machine and another on server. The DHCP employs a connectionless service model, using the User Datagram Protocol (UDP). DHCP operations fall into four phases: discovery, offer, request, and acknowledgement as shown in Fig .1.

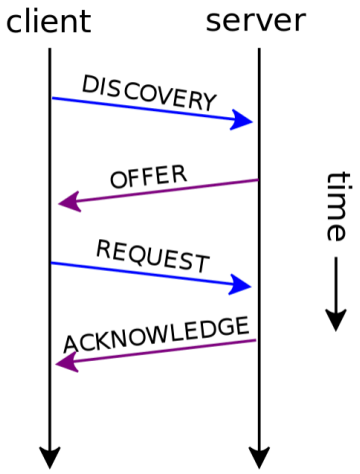

Fig .1

In this lab you will a simplified DHCP-like protocol that sends messages over the reliable TCP. You need to create a DHCP server and a client that will communicate over TCP socket.
To simplify the protocol, our version of DHCP messages include only the transaction ID, and the IP address. Also, all phases are unicast-based.
The protocol contains four phases as follows:

1) Discovery: The client generates a random 8-bit number as transaction ID (range 0-255) and sends to the DHCP server.
2) Offer: The server reads the transaction ID and then generates an IPv4 address based on the transaction ID and sends them back to the client. The IPv4 address is a 32-bit number. The IP address can be randomly generated in your code. Also the transaction ID is a randomly generated 8-bit number. The format should be in human-readable notations, such as 172.16.254.1.
3) Request: The client sends a randomly generated transaction ID and also echoes the IP address to the server that means the client wants it.
4) ACK: Server sends an ACK and echoes the IP address to the client that means its client's IP from now on. It also send a randomly generated transaction ID.

Both DHCP server and the client should read the messages and print them to the screen.

Assume that the server is running at port number of 3300 + xxx (last digits of your USC ID). Note if your USC ID number if 0123-4567-89, the static TCP port number of server will be 3300 + 789 = 4089.
You can use gethostbyname() or getaddrinfo() to obtain the IP address of nunki.usc.edu or the local host (127.0.0.0).

B. Requirement

1. You must write your program either in C or C++ on UNIX. You will be writing two different pieces of code, namely **Server.c** or **Server.cpp** and **Client.c** or **Client.cpp.**
2. Test your server and client on nunki (nunki.usc.edu) by using two different terminals.
3. You are not allowed to run and test your code on any other USC Sun machines (e.g. aludra.usc.edu). This is a policy strictly enforced by ITS and we must abide by that.
4. No MS-Windows programs will be accepted.
5. The Processes are started in this order: Server, and then Client. You are allowed to use delays in your code if you think necessary.
6. You are allowed to use blocks of code from Beej's socket programming tutorial (Beej's guide to network programming) in your lab.
7. When you run your code, if you get the message "port already in use" or "address already in use", please first check to see if you have a zombie process

(from past logins or previous runs of code that are still not terminated and hold the port busy). If you do not have such zombie processes or if you still get this message after terminating all zombie processes, try changing the static TCP port number corresponding to this error message (all port numbers below 1024 are reserved and must not be used). If you have to change the port number, please do mention it in your README file.

C. Programming languages and compilers:

You must use only C/C++ on UNIX as well as UNIX Socket programming commands and functions. Here are the pointers for Beej's Guide to C Programming and Network Programming (socket programming):
http://www.beej.us/guide/bgnet/
(If you are new to socket programming please do study this tutorial carefully as soon as possible and before starting the project)
http://www.beej.us/guide/bgc/
Once you run xwin and open an ssh connection to nunki.usc.edu, you can use a unix text editor like emacs or vi to type your code and then use compilers such as g++ (for C++) and gcc (for C) that are already installed on nunki to compile your code. You must use the following commands and switches to compile yourfile.c or yourfile.cpp. It will make an executable by the name of "yourfileoutput".
gcc -o yourfileoutput yourfile.c -lsocket -lnsl -lresolv
g++ -o yourfileoutput yourfile.cpp -lsocket -lnsl–lresolv
Do NOT forget the mandatory naming conventions mentioned before!
Also inside your code you need to include these header files in addition to any other header file you think you may need:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/wait.h>
```

D. Submission Rules

1. You should submit a zip file that contains all your code files and report as well as a readme.txt file. The name of the zip file should follow this pattern:

Firstname_Lastname_lab2.zip. For example for Mohammad Mirza the zip file should be named: Mohammad_Mirza_lab1.zip

2. Along with your code files, include a **readme.txt** with the following information:
   - **i.** Your full name and student ID
   - **ii.** Compilation steps to run your programs (Optional: include the makefile to compile your code)
   - **iii.** Additional info, if you think necessary.