CS3205 – Computer Networks
Prof. Krishna Sivalingam, Dept. of CSE, IIT Madras
Jan. – May 2023
Lab 2: Simple DNS Server
Due Date: **March 4, 2023, 11PM**. On-Line Submission via Moodle
NO LATE SUBMISSIONS ALLOWED.

This is an individual assignment – please refer to the honor code regarding assignment work; any violation of this will result in 'U' grade, and other penalties.

1 Assignment description

The objective of this assignment is to implement a simple hierarchical Domain Name Server (DNS) system, with a client and a set of relevant DNS servers. The objective of the assignment is to familiarize yourself with writing a network server and a client.

Reference: [KR]: Kurose and Ross's "Computer Networking", Eighth Edition.

There are two major components to the software: (i) the client, and (ii) the DNS servers. The communication will be based on a UDP socket. The main program entities are:

- **Client (C):** This will provide a particular server's name (e.g. ftp.x.edu) to the **Name Resolver** that will return its server's IP address, if known.
- **Name Resolver (NR) or Local DNS Server:** This will take as input from the **Client** and return an output as mentioned above.

  The NR will first contact the **Root DNS Server** and then launch a query with the relevant DNS servers to find the IP address requested. Refer to Fig. 2.18 of KR's book, for more details. In a typical system, all DNS servers will listen on a known port (e.g. 53). In our implementation, the DNS servers will listen on a user-defined non-privileged port.

  The server will print to a file called *NR.output*, each query and the response sent.
- **Root DNS Server (RDS):** RDS will return to NR, the corresponding Top-Level Domain (TLD) Server's IP address and port number.

  The server will print to *RDS.output*, each query and the response sent.
- **TLD DNS Servers (TDS):** There are multiple DNS servers, one per top-level domain, e.g. *.com, .edu, .gov, .in*.

  A TDS will return to NR, the corresponding Authoritative DNS Server's IP address and port number.

  The server will print to *TDS.output*, each query and the response sent.
- **Authoritative DNS Servers (ADS):** There are multiple ADS servers, one per organization, e.g. *cnn.com, uncc.edu, iitmadras.in*. The ADS will return to NR, the requested server's IP address if a match is found and a suitable error message, otherwise.

  The server will print to *ADS.output*, each query and the response sent.

In this assignment, you will implement the client and the set of servers mentioned above. The user will interact with the client in a loop until the user types "bye": the user will enter a server's name; the client with query the NR and print the matching IP address (if found) received from the NR. If no matching record is found, the client will print an appropriate error message.

The sequence of messages sent and received, in order to perform the DNS name-to-address mapping, is presented in Fig. 2.18 of the book.

2 Command-Line invocation

```
% ./lab2-cs20b888 startportnum inputfile
```

The order of command-line arguments is fixed as given above. When your program is invoked, the necessary servers will be created to run in separate child processes listening to the respective port numbers.

The parent process will run the **Client** program and interact with the user. When the user terminates interaction with the client, the client should kill all the server processes that it created and then exit.

For this assignment, we will assign there are only 2 TDS servers and 3 ADS servers per Top-level Domain.

Port Numbers: Let K denote `startportnum` (assume that $K > 1024$). The servers' port numbers are as follows:

- ▷ NR: 1 Server (port $K + 53$)
- ▷ RDS: 1 Server (port $K + 54$)
- ▷ TDS: 2 servers (port $K + 55$ for .com, $K + 56$ for .edu)
- ▷ ADS: 6 servers (3 per TLD) with port numbers $K + 57$, $K + 58$ and so on.

Input File Format:

```
BEGIN_DATA
NR IPAddress
RDS IPAddress
TDS_com IPAddress
TDS_edu IPAddress
ADS1 IPAddress
...
ADS6 IPAddress
List_of_ADS1
www.abank.com 202.15.5.1
smtp.abank.com 202.15.5.2
ftp.abank.com 202.15.5.3
sshd1.abank.com 202.15.5.4
vpn.abank.com 202.15.5.5
```

```
List_of_ADS2
..
List_of_ADS3
..
List_of_ADS4
www.green.edu 192.13.5.1
smtp.green.edu 192.13.5.2
ftp.green.edu 192.13.5.3
sshd1.green.edu 192.13.5.4
vpn.green.edu 192.13.5.5
List_of_ADS5
..
List_of_ADS6
..
END_DATA
```

Note that there are exactly 5 DNS address entries per ADS. The main program launches all these 10 servers (on different ports if on the same system; or use virtual IP addresses on the same system which might require root access; or run the servers on different machines).

Please test your program on your own input files. During grading, the TAs will be testing using a different set of (correctly formatted) input files.

3 Sample Session

Assume that you have created the files lab2-cs20b888.c and the corresponding executables in your LAB2 directory.

```
% ./lab2-cs20b888 10000 inpfile
Enter Server Name: ftp.green.edu
DNS Mapping: 192.13.5.3
Enter Server Name: web.alpha.com
No DNS Record Found
Enter Server Name: bye
All Server Processes are killed. Exiting.
%
```

The text in blue is the user input.

4 What to Submit

Name your assignment directory as LAB2 (Note: ALL UPPERCASE)

Once you are ready to submit, change directory to the directory above LAB2, and tar all files in the directory with the command:

```
% tar czf XYmnZabc-Lab4.tgz LAB2
```

where XYmnZabc refers to your roll number.

The directory should contain the following files:

- ▷ All Source Files: Each file MUST contain a header with the following information:

```
// NAME:
// Roll Number:
// Course: CS3205 Jan. 2023 semester
// Lab number: 2
// Date of submission: <fill>
// I confirm that the source file is entirely written by me without
// resorting to any dishonest means.
// Website(s) that I used for basic socket programming code are:
// URL(s): <fill>
```

- ▷ Makefile
  Typing command *'make'* at the UNIX command prompt, should generate all the required executables.
- ▷ A Script file obtained by running UNIX command *script* which will record the way you have finally tested your program.
- ▷ a README file containing what port number to use, and instructions to compile, run and test your program.
- ▷ a COMMENTS file which describes your experience with the assignment, suggestions for change, and anything else you may wish to say regarding this assignment. This is your opportunity for feedback, and will be very helpful.

Make sure that all files have been correctly tar-red with the appropriate command. Then, submit the *tgz* file via Moodle.

5 Grading

- ▷ Correctly written and executing code: 85 points
- ▷ Viva Voce: 15 points
- ▷ No Script File: -10 points
- ▷ Incomplete Compilation: -10 points
- ▷ NO README: -5 points
- ▷ NO COMMENTS: -5 points

6 Miscellaneous

1. WARNING ABOUT ACADEMIC DISHONESTY: Do not share or discuss your work with anyone else. The work YOU submit SHOULD be the result of YOUR efforts. Any violation of this policy will result in an automatic ZERO on the assignment, a 'U' in the course, and other academic penalties.
2. Ask questions EARLY. Do not wait until the week before. This assignment is quite time-consuming.
3. Questions/Doubts MUST be raised in class or by email on or before Feb. 27, 2023, 11pm.
4. Implement the solutions, step by step. Trying to write the entire program in one shot, and compiling the program will lead to frustration, more than anything else.