CSci4211: Introduction to Computer Networks
Programming Assignment I
Due Date: Friday February 23, 2018 at 11:15p.m.

January 31, 2018

1 Your Task

Domain Name Service is a service utilized by application protocols, such as HTTP, SMTP and FTP to translate host names to IP addresses. As shown in class, DNS runs over UDP and uses port 53. In this assignment; however, you are asked to simulate a DNS server using TCP and utilizing sockets. Your server will respond to client queries mapping host names to IP addresses.

2 Details

Following are details for sending/receiving a DNS query and the steps taken at both the client side and the server side.

2.1 Client

We have written the client program for you. Please refer to class web page for client code in Python (2/3) and Java.

- A client wishing to resolve a host name to an IP address will connect to the server at the port specified in the server program. (We used port 5001).
- For each DNS request, client opens a socket to server to query its desired host names as needed, providing host name for the query.
- The format of the query will have the host name "www.yahoo.com" or "google.com" (of course for stdin input, you need to follow that with a carriage return).
- Holding the socket open, the client will wait for the server to respond with a corresponding IP address (or an error message, or "Host Not Found").
- By receiving a 'q' from input, terminate the clients program
- Upon quitting from the program (by pressing 'q' or 'Q'), all the sockets must be closed. Remember, the server should still be up to service other potential clients.

2.2 Server

This is the program you will need to write. Please refer to the class web page for suggested skeleton and partial code. Note though, if you choose, you can write your own server code.

- The server will be listening to any incoming connections from any client (on the same port, e.g. 5001). Therefore, threading is utilized to handle multiple concurrent connections.

- Upon receiving a query from the client, the server will first lookup its local cache (which could be a file). If the server finds a corresponding entry in the cache, it will be sent to the client directly. If the server cannot find one, it will look up the DNS system using API (e.g. *gethostbyname*).
- When the host name is resolved by the DNS system, it should be added to the local DNS cache as well. This will avoid constantly contacting the DNS system for previously resolved queries.
- The response message will use the format of "Source:<hostname>:<IP address>". For example "Local DNS:google.com: 216.58.216.68". Since this is a simulation of DNS, you can implement your local cache using files. Your file should have the format "<host>:<IP Address>". (Note: for the provided implementation, the file is named as "DNS_Mapping.txt". Please refer to the class web page for a sample file).
- One hostname may have multiple IP addresses in the file. For example, "Local DNS:<hostname>:<IP address_1>:<IP address_2>". You should select one to send to the client (randomly).
- If the host name is neither resolved by the local cache nor by the DNS system, a response from the server should be forwarded to the client in the format "Host not found" and should also be cached. The server should also communicate to the client any formatting issues with a message, such as "Invalid format"
- If the server needs to be brought down, an "exit" should be entered at the server console, which will close all sockets and terminate.

3 What to submit

You should upload the project files to the Moodle site under Submissions section. Your submission (in a tar or zip format) should include the followings:

- The server source code (in one of the languages: Python, and Java).
- A session (using "script" or any other programs) showing a sample of the queries and their resolutions. (Please refer to the class web page for a sample).
- A brief readme.txt (less than 200 words) stating any compilation script you used, and any details that we should be aware of in order to compile and run your program. (Please try to run your program on CSE-Lab machines before submitting)

  This should also include two or more paragraphs explaining what each segment of the server code does. This is not a line-by-line comments, but rather the logic of the server code and what each segment does, such as at what point is the query cached on the local dns and why? Why does the server needs multiple 2 sockets and why are they setup differently? What is the point of threading and how we handle multiple threads?

4 Tips

- Start early to avoid last-minute issues.
- Refer to the google group for any questions and answers regarding the assignment.
- Do not try to learn a new language, utilize knowledge of languages you are familiar with.
- Save and compile your work frequently.
- Try to make your program simple, commented and easy to read.
- Bonus: Instead of selecting the address randomly, you could use Ping value to pick up a closer server.