Lab 3

Objectives:

- Gain insights into the operation of the DNS protocol
- Dig deep into DNS server organisation
- Get familiar with the socket programming

Prerequisites & Links:

- Week 2 & 3 Lectures
- Relevant Parts of Chapter 2 of the textbook
- Introduction to Tools of the Trade
- Basic understanding of Linux. A good resource is here but there are several other resources online.
- Several socket programming resources and sample code for all 3 programming languages are available here .
- dns-ethereal-trace-2
- index.html

Marks: **10 marks.**

- We expect the students to go through as much of the lab exercises as they can at home and come to the lab for clarifying any doubts in procedure/specifications
- If lab exercise involves diagrams or plots, you require to show them to the tutor as well

Deadline:

23:59:59 Sunday 17th March. You can submit as many times as you wish before the deadline. A later submission will override the earlier submission, so make sure you submit the correct file. Do not leave until the last moment to submit, as there may be technical, or communications error and you will not have time to rectify it.

Late Report Submission Penalty:

Late penalty will be applied as follows:

- 1 day after deadline: 20% reduction
- 2 days after deadline: 40% reduction
- 3 days after deadline: 60% reduction
- 4 or more days late: NOT accepted

Note that the above penalty is applied to your final mark for your lab report. For example, if you submit your lab work report 2 days late and your score on the lab report is 4, then your final mark will be 4 - 1.6 (40% penalty) = 2.4.

Submission Instructions:

Submit a PDF document **Lab3.pdf** with answers to Exercises. Include any supporting documents that you think are necessary. Your server should be named WebServer.c or WebServer.java or WebServer.py. Create a tar archive of all the files called **Lab3.tar**. Submit the archive using give. You can submit from a lab machine or ssh into the CSE login server. Instructions to ssh into CSE login servers are here .

If you are using Python, then please add a comment at the top of your code to indicate the correct version of Python to use for testing.

Original Work Only:

You are strongly encouraged to discuss the questions with other students in your lab. However, each student must submit his or her own work. You may need to refer to the material indicated above (particularly Tools of the Trade document) and also conduct your own research to answer the questions.

OS Compatibility:

We will test your code on CSE machines via the command line interface. Note that CSE machines support the following: **gcc version 4.9.2, Java 1.8, Python 2.7, 2.8 and 3.** It is thus imperative that you test your code on CSE machines before submitting. This is particularly important if you write your code on your own machine and use an IDE. You MUST test your code on CSE machines (via command line) before submitting. If we cannot get your code to work on CSE machines, then we can't mark it and unfortunately, you won't receive any marks.

Exercise 1: Explore DNS records (Not marked, No need to submit)

DNS servers use different record types for different purposes. For each type of DNS record, there is an associated type of DNS query. Check the following page (https://en.wikipedia.org/wiki/List_of_DNS_record_types) and find out what the following resource record types are used for:

- A
- CNAME
- MX
- NS
- PTR
- SOA

Exercise 2: Tracing DNS with Wireshark (Not marked, No need to submit)

For this particular experiment download the dns trace file: dns-etheral-trace-2 .

Step 1: Open an xterm and run Wireshark.

Step 2: Load the trace file *dns-ethereal-trace-2* by using the *File* pull down menu, choosing *Open* and selecting the appropriate trace file. This file captures the sequence of messages exchanged between a host and its default DNS server while using the *nslookup* (this tool is similar to the *dig* tool that we used earlier in the lab) utility for obtaining the canonical name (type A record) of www.mit.edu . The IP address of the default DNS server for the host is 128.238.29.22. Now filter out all non-DNS packets by typing "dns" (without quotes) in the filter field. Also click the right arrow for DNS in the packet-header detail window. Now focus on the **last two DNS messages** from the 6 listed and answer the following questions:

Question 1: What transport layer protocol is being used by the DNS messages?

Question 2: What is the source and destination port for the DNS query message and the corresponding response?

Question 3: To what IP address is the DNS query message sent? Is this the same as the default local DNS server?

Question 4: How many "questions" are contained in the DNS query message? What "Type" of DNS queries are they? Does the query message also contain any "answers"?

Question 5: Examine the DNS response message. Provide details of the contents of the "Answers", "Authority" and "Additional Information" fields. What can you infer from these?

Exercise 3: Digging into DNS

In order to answer the following questions, you will make DNS queries using some of the query types you have encountered in the above exercise. Some questions require you to make multiple DNS queries. Before you proceed, read the manpage of dig (type man dig in the terminal). Make sure you understand how you can explicitly specify the following:

- nameserver to query
- type of DNS query to make (the default query types are those you saw in exercise 1)
- performing reverse queries

Note: Include the output of all the dig commands you have used in your answers.

To send a query to a particular name server (say x.x.x.x) you should use the following command:

```
dig @x.x.x.x hostname
```

Question 1. What is the IP address of www.cecs.anu.edu.au . What type of DNS query is sent to get this answer?

Question 2. What is the canonical name for the CECS ANU web server? What is its IP address? Suggest a reason for having an alias for this server.

Question 3. What can you make of the rest of the response (i.e. the details available in the Authority and Additional sections)?

Question 4. What is the IP address of the local nameserver for your machine?

Question 5. What are the DNS nameservers for the “cecs.anu.edu.au” domain (note: the domain name is cecs.anu.edu.au and notwww.cecs.anu.edu.au)? Find out their IP addresses? What type of DNS query is sent to obtain this information?

Question 6. What is the DNS name associated with the IP address 149.171.158.109? What type of DNS query is sent to obtain this information?

Question 7. Run dig and query the CSE nameserver (129.94.242.33) for the mail servers for Yahoo! Mail (again the domain name is yahoo.com, not www.yahoo.com). Did you get an authoritative answer? Why? (HINT: Just because a response contains information in the authoritative part of the DNS response message does not mean it came from an authoritative name server. You should examine the flags in the response to determine the answer)

Question 8. Repeat the above (i.e. Question 7) but use one of the nameservers obtained in Question 5. What is the result?

Question 9. Obtain the authoritative answer for the mail servers for Yahoo! mail. What type of DNS query is sent to obtain this information?

Question 10. In this exercise you simulate the iterative DNS query process to find the IP address of your machine (e.g. lyre00.cse.unsw.edu.au). First, find the name server (query type NS) of the "." domain (root domain). Query this nameserver to find the authoritative name server for the "au." domain. Query this second server to find the authoritative nameserver for the "edu.au." domain. Now query this nameserver to find the authoritative nameserver for "unsw.edu.au". Next query the nameserver of unsw.edu.au to find the authoritative name server of cse.unsw.edu.au. Now query the nameserver of cse.unsw.edu.au to find the IP address of your host. How many DNS servers do you have to query to get the authoritative answer?

Question 11. Can one physical machine have several names and/or IP addresses associated with it?

Exercise 4: A Simple Web Server

In this exercise, you will learn the basics of TCP socket programming: how to create a socket, bind it to a specific address and port, as well as send and receive an HTTP packet. You will also learn some basics of HTTP header format. You will develop a web server that handles one HTTP request at a time. Specifically, your web server should do the following:

(i) create a connection socket when contacted by a client (browser).

(ii) receive HTTP request from this connection. Your server should only process GET request. You may assume that only GET requests will be received.

(iii) parse the request to determine the specific file being requested.

(iv) get the requested file from the server's file system.

(v) create an HTTP response message consisting of the requested file preceded by header lines.

(vi) send the response over the TCP connection to the requesting browser.

(vii) If the requested file is not present on the server, the server should send an HTTP “404 Not Found” message back to the client.

(viii) the server should listen in a loop, waiting for next request from the browser.

You don't have to deal with any other error conditions.

Your program should be called WebServer.c or WebServer.java or WebServer.py.

You should write the server so that it executes with the following command:

```
$java WebServer port (for Java)

$WebServer port (for C)

$python WebServer.py port (for Python)
```

where port is the port No your Web server will be listening on.

1. Place a simple HTML file (index.html, without any hyperlinks) in the same directory as the server program. A sample index.html file is provided here . Run the server program as indicated above. Open a web browser on the same machine. Type the following url:
http://*127.0.0.1:port/index.html* where *port* is the port number the server listens on. If you forget to include the port number, the browser will assume the default port of 80. The browser should display the content of index.html.

2. Place multiple image files (.png) in the same directory as the server program. Run the server program as indicated above. Open a web browser on the same machine.
Type *http://127.0.0.1:port/myimage.png*

where *port* is the port number the server listens on and *myimage.png* is one of the image files present in the server's directory. The browser should display the image.

3. Now try and request for an object that does not exist in the server directory, e.g.: *http://127.0.0.1:port/bio.html* . The browser should display the 404 error message.

Note that you cannot use any of the pre-made web servers available in different programming languages.