Lab 3

Objectives:

- Gain insights into the operation of DNS
- Dig deep into DNS server organisation
- Get familiar with the basic socket programming
- Preparation for programming assignments

Prerequisites and Links:

- Week 2 & 3 Lectures.
- Relevant Parts of Chapter 2 of the textbook (socket programming) and Socket Programming discussions of the course.
- Basic understanding of Linux. A good resource is here but there are several other resources online
- Several socket programming resources for all 3 programming languages are available at the bottom of the Sample Client Server Programs & Networking Resources page

Marks: 4 marks

Questions to be marked: 1) Exercise 2. Questions 7, 2) Exercise 2. Questions 10, 3) Exercise 3.

- Each lab comprises of a number of exercises. Not all the exercises for each lab are marked. Only those marked with (*) and written in bold will be marked.
- We expect the students to go through as much of the lab exercises as they can at home and come to the lab ready.
- Please attend your allocated lab and show/explain the answers of the marked exercises to your tutor.
- If lab exercise involves diagrams or plots, you require to show them to the tutor as well.
- Please make sure you **sign the marking form** once the tutor marked your lab. Signing this form implies that you agreed on the mark you received.
- There are 7 labs during this course. For each student, the 5 best performing labs will contribute to your final lab mark.

Exercise 1: Explore DNS records

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
- AAAA

Exercise 2: 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 man page 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 the previous exercise)
- performing reverse queries

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? How did you figure this out?

Question 5. What are the DNS nameservers for the “cecs.anu.edu.au” domain (note: the domain name is cecs.anu.edu.au and not www.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. Use 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) (0.5 mark)

Question 8. Repeat the above (i.e. Question 7) but use the nameserver 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? (0.75 mark)

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

(*) Exercise 3: Implementing a simple Web Server (2.75 mark)

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.

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)
```

or

```
$WebServer port (for C)
```

or

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

Place an HTML file (you can download an example from course website) in the same directory as the server program. Run the server program as indicated above. Open a web browser on the same machine. Type the following in the url: http://127.0.0.1:port/index.html window where *port* is the port number the server listens on. If you forget that the browser will assume the default port of 80. The browser should display the content of index.html.

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.

Marking guide:

Correct Loading of index page: 2 marks

Next load a page that does not exist - http://127.0.0.1:port/bio.html

Correct Display of 404 error message: 0.75 marks