ECSE 416 Experiment 1 Report

Group 5
Saima Haneef - 260744058
Nusaiba Radi - 260672575

Introduction

For this experiment, a Domain Name System (DNS) client was implemented using sockets in Java. The DNS request/response mechanism is used to map the domain name to an IP address and vice-versa by the application. The objective of this lab was to acquire a better understanding of the inner workings of the DNS process, involving the queries being sent and the responses being received.

To achieve the objective of this lab, we were expected to create a DNS Client program which would be responsible for constructing a DNS request packet and interpreting the DNS server response. This program was required to send and interpret different types of records, as well as retransmit lost queries for a fixed number of retries chosen by the user. In the case of an error, the program would also have to display an appropriate message to notify the user of what went wrong. Some examples of possible errors would involve incorrect arguments or even response messages that do not follow DNS specifications. To increase our knowledge of DNS servers, the Google DNS server and the McGill DNS servers were also compared and the results were studied.

DNS Client Program Design

Our DNS client program design consisted of one class which contained 1 main method and a few helper methods. The arguments entered were read and the packet to be sent was filled accordingly (according to the lab instructions) within the main method. A UDP packet was used to send this query to the server. Appropriate error conditions were also set to handle multiple types of errors while the arguments were read. A timeout was set (changeable by the user), which would play a role in deciding when to retry and send another query if necessary. The amount of retries to be executed is also checked in accordance with the inputted arguments or default value. When a response is received, no retry will take place and the subsequent helper methods are called to parse the response packet and print the appropriate outputs. The helper method (“readPacket”) decodes the header of the packet (to determine fields such as authoritative or non authoritative) and determine the amount of records stored in the packet. Another helper method (“packetParse”) is also called within this method to parse through the specific resource records (Answer, Authority and Additional Sections) which will in turn output the proper domain name(s) or IP address(es) for each section in addition to any other information required (ttl, etc.).

Compressed domain names were dealt with appropriately within this program as well. As mentioned previously, the response packet is parsed suitably depending on its response type. If the response type is A, no compression is necessary and the IP address is outputted as a string. However, for CNAME, NS and MX, the rData field (containing the data to be received) is parsed by first checking if the first two bytes are leading 1s (0xc0 in hex), in which case a helper method called pointerName is executed (shown in Figure 6 in the Appendix). The leading 1s indicate that an offset is required which will relocate the index to a repeated domain name. The first byte of the given domain name would indicate the amount of characters to be expected in the following label. Each subsequent byte is then stored as a character in a String and a period is added before decoding

the next label. This is done until one byte of zeros is detected to indicate the end of the domain name. In the case that two leading 1s are not detected, the index is not altered.

With regards to error handling, syntax errors as well as response errors (in the rCode for example) were detected. If arguments did not follow the correct format, the appropriate error message was displayed. If less than 2 arguments entered or more than one @ sign is entered, the system will exit and print the corresponding error. Additionally, optional parameters (such as -t,-r and -p) were checked to see if they were inputted correctly (i.e. "-tr", "t" or any other incorrectly written parameters would not be accepted). Format errors, server failures, unimplemented, unfound and refused conditions in the rCode were also detected.

Testing

To ensure that the DNSClient program functioned correctly, several tests had to be conducted. Since the program could take in different types of parameters as inputs, the tests had to be constructed carefully so as to detect any possible errors when the responses were outputted. Since three possible types of queries could be used in the program (Type A, MX and NS), it was important to make sure all 3 types were tested. Additionally, optional parameters, such as the timeout, port and amount of retries also had to be tested on. Consequently, various tests that contained combinations of the optional parameters, as well as the types of queries along with different domain names and IP addresses of the appropriate DNS servers were constructed. To make sure the program was not hard coded, tests containing domain names of different lengths were also tested on. Tests with purposely inputted mistakes were also carried out to verify the functionality of the program's error conditions. Some examples of the tests that were conducted are shown in the Figures below.

To confirm that the DNS packets were being formatted and parsed correctly, the application Wireshark was used. Wireshark was very helpful as the packet was viewable in hexadecimal format, and also displayed important details that the DNSClient program was required to output. This made it easier to understand the organization of the packets being sent and received. Figures 1 and 2 below demonstrate how both the outputs of the DNSClient program and the information received in the packets on Wireshark complied which each other, consequently demonstrating how well the DNSClient program works.

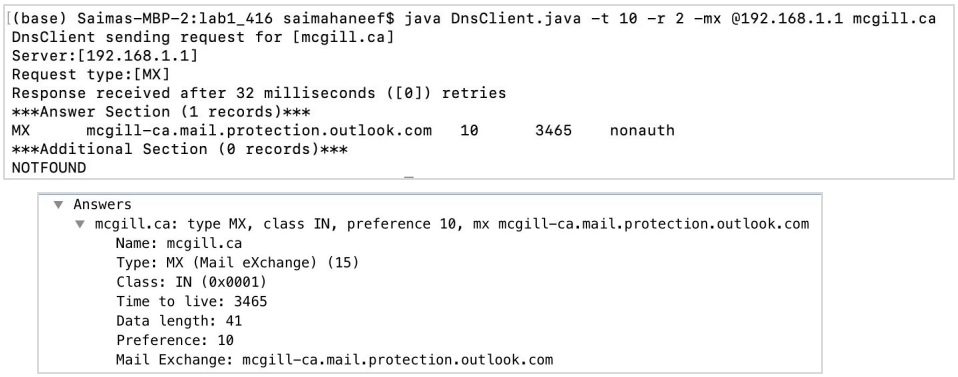


Figure 1. An example of an executed test demonstrating the response shown on the command line when the DnsClient program is run (top) and the corresponding result shown for the same response packet on Wireshark (bottom)

```
(base) Saimas-MBP-2:lab1_416 saimahaneef$ java DnsClient.java -t 5 -r 2 -ns @192.168.1.1 www.hotmail.com
DnsClient sending request for www.hotmail.com
Server: 192.168.1.1
Request type: NS
Response received after 0.056 seconds (0 retries)
***Answer Section (2 records)***
CNAME   outlook-fd-0010.live.com        2247    noauth
CNAME   a-0010.a-msedge.net     852     noauth
***Additional Section (0 records)***
```

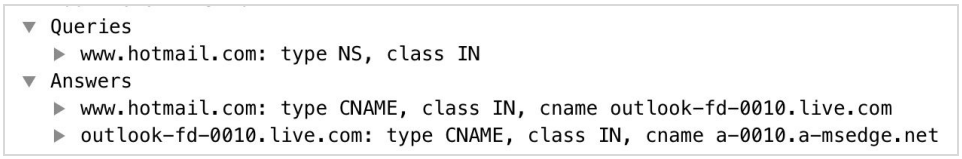


Figure 2. An example of an executed test demonstrating the response shown on the command line when the DnsClient program is run (top) and the corresponding result shown for the same response packet on Wireshark (bottom)

Since the program deals with numerous parameters, there are many error conditions that need to be considered. An example of a possible error condition is depicted below in Figure 3. Our program did not initially examine the bounds of each value in a given IP address so an extra error condition (in the form of an if statement) was simply added so as to make the user aware of an incorrectly entered IP address not within the appropriate boundaries, as shown in Figure 4.

```
(base) Saimas-MacBook-Pro-2:lab1_416 saimahaneef$ java DnsClient.java -tr 5 @132.206.85.18 www.yahoo.com
ERROR    incorrect syntax
 Proper format: [-t] [-r] [-p] [-nx || -mx] @server name
```

Figure 3. A sample test demonstrating the output on the command line when the DnsClient program is run with an incorrect input argument

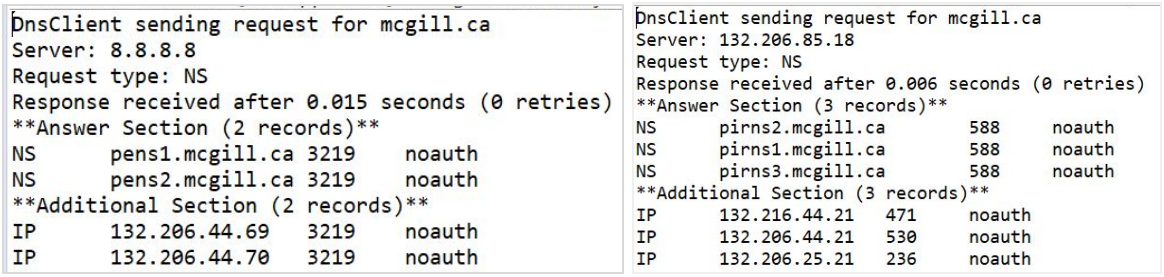
```
(base) Saimas-MBP-2:lab1_416 saimahaneef$ java DnsClient.java -t 5 -r 2 -ns @192.-168.1.1 www.hotmail.com
ERROR    Please make sure your arguments follows the correct format
```

Figure 4. A sample test demonstrating the output on the command line when the DnsClient program is run with a negative IP address

Experiment

The IP addresses of McGill's DNS servers are found to be 132.206.44.69 and 132.206.44.70 [1]. To confirm this, Google's DNS Server (8.8.8.8) was used to perform a NS query for mcgill.ca. As anticipated, the answer section of the response to this query consisted of the host names "pens1.mcgill.ca" and "pens2.mcgill.ca". When a query was sent using these host names, the IP addresses, 132.206.44.69 and 132.206.44.70, were returned. Figure 5 demonstrates the response received when a NS query for mcgill.ca was sent once more, this time with the IP addresses of McGill's DNS servers shown in the Additional section (as a result of the previous queries that were sent and are now stored in the DNS cache).

When the same query was sent using the Mcgill DNS server, more information was provided in both the Answer and Additional section of the response. It is interesting to note that slightly different IP addresses were included in the Additional section (132.216.44.21, 132.206.44.21, 132.206.25.21) as shown in Figure 5. This is probably done since the McGill domain name is accessed more frequently within the McGill campus and users are averted to different IP addresses to reduce traffic on the main authoritative name server IP addresses mentioned above.



```
DnsClient sending request for mcgill.ca
Server: 8.8.8.8
Request type: NS
Response received after 0.015 seconds (0 retries)
**Answer Section (2 records)**
NS      pens1.mcgill.ca 3219    noauth
NS      pens2.mcgill.ca 3219    noauth
**Additional Section (2 records)**
IP      132.206.44.69   3219    noauth
IP      132.206.44.70   3219    noauth
```

```
DnsClient sending request for mcgill.ca
Server: 132.206.85.18
Request type: NS
Response received after 0.006 seconds (0 retries)
**Answer Section (3 records)**
NS      pirns2.mcgill.ca        588     noauth
NS      pirns1.mcgill.ca        588     noauth
NS      pirns3.mcgill.ca        588     noauth
**Additional Section (3 records)**
IP      132.216.44.21   471     noauth
IP      132.206.44.21   530     noauth
IP      132.206.25.21   236     noauth
```

Figure 5. Performing a NS query for mcgill.ca using Google DNS (right) and using McGill DNS (left)

To have a better understanding of the different types of DNS servers, seven different website addresses were queried and the responses received from both DNS servers were compared. Table 1 and 2 below demonstrate the difference in the parameters included in the responses when www.google.com and www.facebook.com were queried. Five more tables summarizing the responses received from both DNS servers are included in the Appendix.

Based on the tables of the responses received from the Google public DNS server and the McGill DNS Server, there is a significant difference in the amount of records included in the additional section of the response. For example, in Table 1, when querying www.google.com, there were 0 records reported in the additional section using the McGill's DNS server, whereas there were 11 additional records found when using Google's public DNS server. The reasoning behind this could be a result of the fact that Google's DNS server contains a lot of information in its cache as a result of its numerous users. This information could then be used as additional records that could then be added to other similar queries, since the additional section simply stores information that is relevant to your query. Another difference between the two DNS servers dealt with the time taken to receive a response. As expected, when the McGill DNS server was used, less time was required while more time was needed to receive a response when the Google DNS server was used. This is obviously due to the proximity of the DNS servers (McGill's DNS server is closer). Another factor could also be the extra data included in the additional section of the response.

In addition, if a CNAME was included in the Answer section, the same alias was found when querying both DNS servers. However, it was also noticed that the IP addresses included in the Answer section (and the Additional section) could differ since multiple DNS servers could be used for one domain name, consequently more than one IP address could be contained in the response packets.

To test the hypothesis mentioned above, more experiments were conducted but this time with the IP address of an additional public DNS server to compare the general results. As can be seen in Table 9 (in the Appendix), when using another public DNS server (Cloudflare, IP address 1.1.1.1), the findings were more similar to the results found when using the McGill DNS server; there were substantially less results found in the additional section when compared to Google's DNS server. This could be due to the fact that Google's DNS servers are used more popularly worldwide. More experiments with different hostnames were also executed and supported our initial hypothesis, further confirming the previously stated hypothesis.

Table 1. Comparison of responses gotten for www.google.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A	A
Time taken	6 ms	15ms
IP Address / Alias	IP 172.217.0.228	IP 172.217.13.164
TTL	146 ms	81ms
AA	Non authoritative	Non authoritative
Records of Answer Section	1	1
Records of Additional Section	0	11

Table 2: Comparison of responses gotten for www.facebook.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time Taken	5ms	16ms
IP Address / Alias	CNAME star-mini.c10r.facebook.com IP 31.13.80.36	CNAME star-mini.c10r.facebook.com IP 31.13.80.36
TTL	401ms & 39ms	506ms & 8ms
AA	Non authoritative	Non authoritative
Records of Answer Section	2	2
Records of Additional Section	0	9

Discussion

After completing this experiment, we gained a better understanding of the implementation of a network application using sockets with a UDP protocol. Through this lab, we learnt about the differences between the three DNS records, the packet compositions of both queries and responses as well as specific details dealing with the inner workings of the DNS process such as packet compression taking place in the response packets.

Some of the discoveries that were made in this lab dealt with the time taken to receive a response as well as the records included in the Additional section of the response packet. Specifically, we learnt that the time taken to receive a response from a local DNS server is shorter than the time taken for a public one. Additionally, Google DNS servers provide a noticeable amount of records in the Additional section when compared to other local and public DNS servers.

The main challenges faced while designing the program involved the byte organization of the sent packet and ensuring that the appropriate indexing was followed. Furthermore, acknowledging all possible errors that could be encountered was also an obstacle since the program contained various parameters and conditions that needed to be considered.

A way to improve our implementation would be to create more classes as it would allow us to avoid duplication of the code and provide a better structure. Using advanced Java libraries would also allow us to write more concise code and implement the program in a more efficient manner. These improvements would make it easier to overcome challenges such as the byte organization of the sent and received packets. An extra script (in Python or any appropriate language) could be constructed to help make our program more robust by performing more rigorous tests.

References

[1] DNSlytics, “IP information 132.216.177.160,” *DNSlytics*. [Online]. Available: https://dnslytics.com/ip/132.216.177.160?fbclid=IwAR09yg25TyV-6QvHdYPiMQFgGl5nRGVcNMv6uKK1GRcjsJHoJVbmPWj9lIQ. [Accessed: 28-Sep-2019].

Appendix:

```
static String pointerName(byte[] byteRec, int indx) {

    String alias = "";
    //checks if there is leading 1s
    if((byteRec[indx] & 0xFF) >= 0xC0) {

        indx++;
        // pointer to the new address where an old domain name exists
        int pointer = byteRec[indx];
        //checks till zeros are reached (indicating the end of domain name)
        while(byteRec[pointer]!=0) {

            //the first byte indicating size of the first word before the period
            int size = byteRec[pointer];
            for(int i =0; i<size; i++) {
                pointer++;
                alias = alias + (char)byteRec[pointer];
            }

            indx++;
            //if its not zero, add a period after the first word and subsequent words till
            // zeros are reached
            if(byteRec[indx]!=0) {
                alias = alias + ".";
                pointer++;
            }
        }
        indx++;

    }

    return alias;
}
```

Figure 6. A helper method to deal with the packet compression

Table 3: Comparison of responses gotten for www.yahoo.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time Taken	7ms	123ms
IP Address / Alias	CNAME atsv2-fp-shed.wg1.b.yahoo.com IP 72.30.35.9 IP 72.30.35.10	CNAME atsv2-fp-shed.wg1.b.yahoo.com IP 72.30.35.10 IP 72.30.35.9
TTL	107ms & 13ms &13ms	120ms & 30ms&30ms
AA	Non authoritative	Non authoritative
Records of Answer Section	3	3
Records of Additional Section	0	8

Table 4: Comparison of responses gotten for www.twitter.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time Taken	11 ms	17ms
IP Address / Alias	CNAME twitter.com IP 104.244.42.129 IP 104.244.42.65	CNAME twitter.com IP 104.244.42.129 IP 104.244.42.1
TTL	434 ms & 267ms & 267ms	503ms & 1203ms & 1203ms
AA	Non authoritative	Non authoritative
Records of Answer Section	3	3
Records of Additional Section	0	9

Table 5: Comparison of responses gotten for www.instagram.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time taken	23 ms	118ms
IP Address / Alias	CNAME z-p42-instagram.c10r.facebook.com IP 31.13.80.174	CNAME z-p42-instagram.c10r.facebook.com IP 31.13.80.174
TTL	1858ms & 27ms	1814ms & 60ms
AA	Non authoritative	Non authoritative
Records of Answer Section	2	2
Records of Additional Section	0	9

Table 6: Comparison of responses gotten for www.youtube.com using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time taken	5 ms	16ms
IP Address / Alias	CNAME youtube-ui.l.google.com IP 172.217.164.238 IP 172.217.1.174 IP 172.217.165.14 IP 172.217.1.14 IP 172.217.164.206 IP 172.217.0.238	CNAME youtube-ui.l.google.com IP 172.217.13.110 IP 172.217.13.174 IP 172.217.13.206 IP 172.217.13.142
TTL	62243ms & 53ms (repeated)	1422ms & 73ms (repeated)
AA	Non authoritative	Non authoritative
Records of Answer Section	7	5
Records of Additional Section	0	7

Table 7: Comparison of responses gotten for www.wikipedia.org using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time taken	12 ms	17ms
IP Address / Alias	CNAME dyna.wikimedia.org IP 208.80.154.224	CNAME dyna.wikimedia.org IP 208.80.154.224
TTL	62172ms & 122ms	3529MS & 529ms
AA	Non authoritative	Non authoritative
Records of Answer Section	2	2
Records of Additional Section	0	12

Table 8: Comparison of responses gotten for www.canada.ca using two different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**
Type	A & CNAME	A & CNAME
Time taken	7ms	395ms
IP Address / Alias	CNAME www.canada.ca.edgekey.net CNAME e4073.dscb.akamaiedge.net IP 23.212.168.178	CNAME www.canada.ca.edgekey.net CNAME e4073.dscb.akamaiedge.net IP 104.88.9.233
TTL	77ms & 77ms & 5ms	300ms &300ms &20ms
AA	Non authoritative	Non authoritative
Records of Answer Section	3	3
Records of Additional Section	0	7

Table 9: Comparison of responses gotten for www.bing.com using three different servers

Response	**McGill DNS Server 132.206.85.18**	**Google public DNS Server 8.8.8.8**	**Cloudflare public DNS Server 1.1.1.1**
Type	A & CNAME	A & CNAME	A & CNAME
Time taken	17ms	722ms	20ms
IP Address / Alias	CNAME a-0001.a-afdentry.net.trafficmanager.net CNAME dual-a-0001.a-msedge.net IP 13.107.21.200 IP 204.79.197.200 21	CNAME a-0001.a-afdentry.net.trafficmanager.net CNAME dual-a-0001.a-msedge.net IP 13.107.21.200 IP 204.79.197.200	CNAME a-0001.a-afdentry.net.trafficmanager.net CNAME dual-a-0001.a-msedge.ne IP 13.107.21.200 IP 204.79.197.200
TTL	774ms &37ms & 21ms & 21ms	2805ms & 60ms & 60ms & 60ms	683ms & 58ms & 24ms & 24ms
AA	Non authoritative	Non authoritative	Non authoritative
Records of Answer Section	4	4	4
Records of Additional Section	0	6	0