University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 4: Distributed Database Management, Implementation of Iterative, and Recursive Queries of DNS Records

Submitted By :

Name: Md. Emon Khan

Roll No : 30

Name: Mahmudul Hasan

Roll No : 60

Submitted To :

Dr. Md. Abdur Razzaque

Dr. Md. Mamun Or Rashid

Dr. Muhammad Ibrahim

Md. Redwan Ahmed Rizvee

Submitted On : February 18, 2024

1 Introduction

The goal of lab-4 is to investigate the Domain Name System (DNS) through the implementation and analysis of the iterative and recursive resolution methods. We simulate a DNS client and server to understand their functionalities and compare their message flow, performance, and suitability for various scenarios. By testing and evaluating the implemented system, we aim to gain practical insights into the inner workings of DNS and contribute to a deeper understanding of this fundamental internet technology.

1.1 Objectives

Some of the preliminary objectives of this lab experiment are:

- To emulate the Domain Name Service (DNS) protocol
- To implement and understand the functionality of an iterative DNS resolution
- To implement and understand the functionality of a recursive DNS resolution
- To understand the differences between the iterative and recursive DNS resolutions
- To emulate caching mechanism of DNS servers and subsequent error handling

2 Theory

The Domain Name System (DNS) is a hierarchical distributed system that translates human readable domain names like "www.example.com" into machine readable IP addresses. When a user tries to access a website, their computer follows a series of steps to resolve the domain name to an IP address:

- **Search DNS Cache:** The computer first checks its local DNS cache to see if the IP address for the domain name is already stored. If it is, the process is complete and the user is directed to the website.
- **Query ISP's DNS Servers:** If the IP address is not found in the cache, the computer queries its Internet Service Provider's (ISP) DNS

servers. These servers have a larger cache of domain names and IP addresses, and they may be able to resolve the query without needing to contact other servers.

- **Query Root Nameservers:** If the ISP's DNS servers do not have the answer, they query the root nameservers. The root nameservers are the top level of the DNS hierarchy, and they know the location of the Top-Level Domain (TLD) nameservers for all domains, e.g., .com, .org, .net etc.
- **Query TLD Nameservers:** The root nameservers direct the query to the TLD nameservers for the specific domain. These nameservers are responsible for a particular TLD, and they know the location of the authoritative nameservers for that domain.
- **Query Authoritative Nameservers:** The TLD nameservers direct the query to the authoritative nameservers for the specific domain. These nameservers are the ultimate source of information for the domain name, and they store the IP address for the domain.
- **Receive the Answer:** The authoritative nameservers send the IP address back to the ISP's DNS servers, which then cache the information and send it back to the user's computer.
- **Store the Answer:** The user's computer stores the IP address in its DNS cache for future reference.

This process is typically very fast, and users rarely notice the steps involved. However, it is important to understand how DNS works, as it is a fundamental part of the internet infrastructure.

3 Methodology

We use Python socket UDP connection to simulate a DNS server. To manipulate and handle DNS messages in the standard DNS message format we use the Python dnslib library

3.1 Setting up the DNS Server

- We create a single server that acts as an authoritative DNS server
- The server awaits a client's request

- Upon receiving the request the server looks for the requested domain
- If the domain is found the server replies with the IP
- If the domain is not found the server replies with an error message

3.2 Iterative DNS resolution

- We create a rooted tree structure of DNS servers and have then run in different threads
- The client first sends a request to the root DNS server
- If the requested domain is found, the rooted server sends the IP and the query ends
- If the request is not found, the rooted server replies with a referral to a top level domain (TLD) DNS server
- The client then sends the request to that TLD server
- If the requested domain is found, the TLD server sends the reply and the query ends
- Else the TLD server replies with a referral to some authoritative DNS server
- Then the client sends the request to that authoritative DNS server
- The authoritative DNS server then replies with an answer if the domain is found, or else an error, anyway, ending the query

3.3 Recursive DNS resolution

- We create a rooted tree structure of DNS servers and have then run in different threads
- The client sends a recursive DNS query to recursive DNS resolver
- The recursive DNS resolver will send query to the root DNS server on behalf of the client
- If the root has the answer it'll send the answer to the resolver or else forward it to a TLD server

- If the TLD server finds the domain it'll reply with the IP to the root, otherwise forward to an authoritative DNS server
- If the authoritative server finds the domain it'll reply to the TLD with the IP or an error otherwise
- The replies will be backtracked and finally be received by the client through the resolver

3.4 Extending the System

- We try extending the system using a short TTL value
- Each server deletes a record when the TTL expires
- We implement DNS caching by saving successful query results
- We test the DNS server process failure

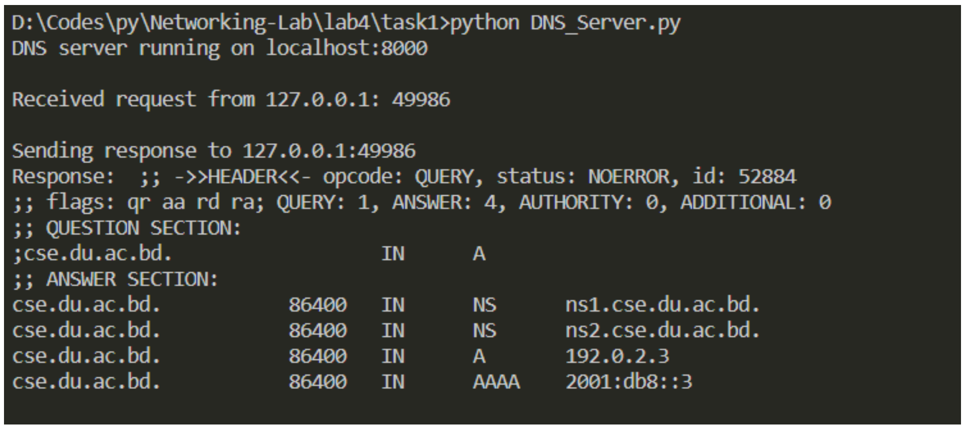
4 Experimental result

4.1 Setting up the DNS Server

In the first task we are required to test a DNS server by sending request to an authoritative server. On average, the client gets the result in roughly 0.01s. In the example below, it took 0.0089s to retrieve a response

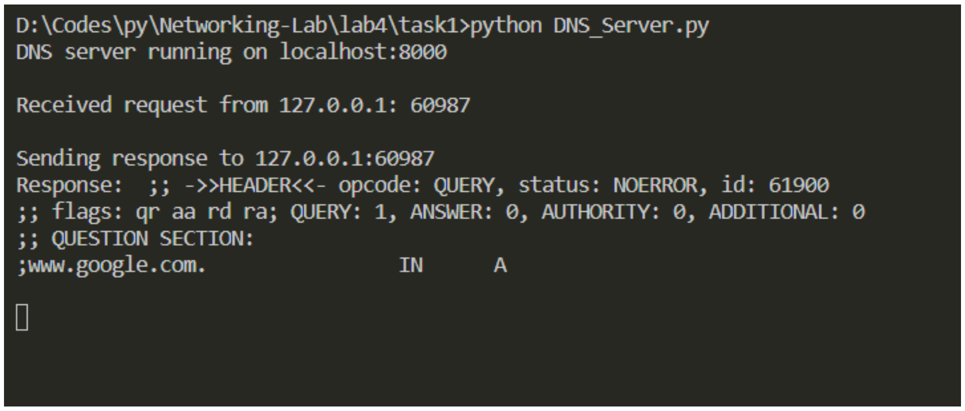
Authoritative DNS Server

The only server here acts as the authoritative server.



```
D:\Codes\py\Networking-Lab\lab4\task1>python DNS_Server.py
DNS server running on localhost:8000

Received request from 127.0.0.1: 49986

Sending response to 127.0.0.1:49986
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52884
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.bd.                IN      A
;; ANSWER SECTION:
cse.du.ac.bd.         86400   IN      NS      ns1.cse.du.ac.bd.
cse.du.ac.bd.         86400   IN      NS      ns2.cse.du.ac.bd.
cse.du.ac.bd.         86400   IN      A       192.0.2.3
cse.du.ac.bd.         86400   IN      AAAA    2001:db8::3
```

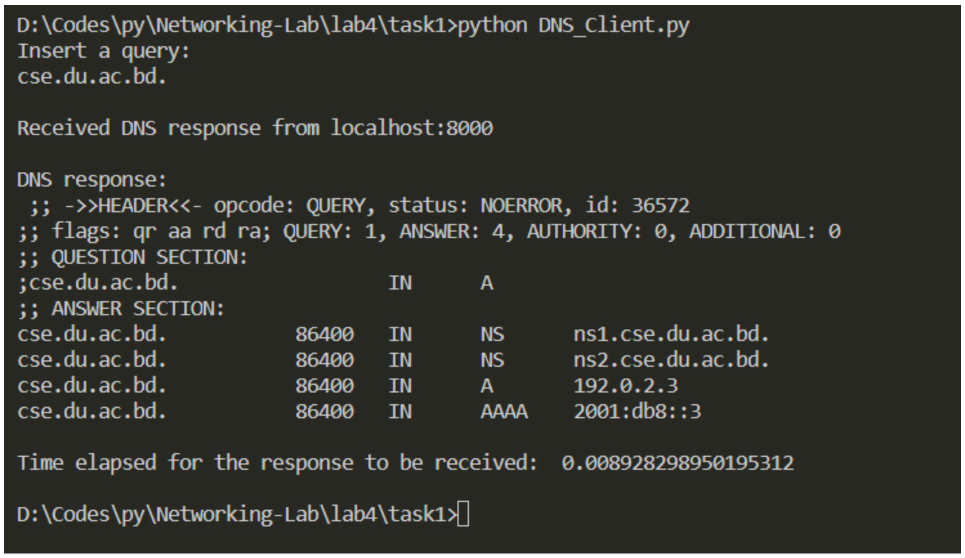
Figure 1: Authoritative DNS Server Status on Successful Answer



```
D:\Codes\py\Networking-Lab\lab4\task1>python DNS_Server.py
DNS server running on localhost:8000

Received request from 127.0.0.1: 60987

Sending response to 127.0.0.1:60987
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61900
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.google.com.               IN      A
```

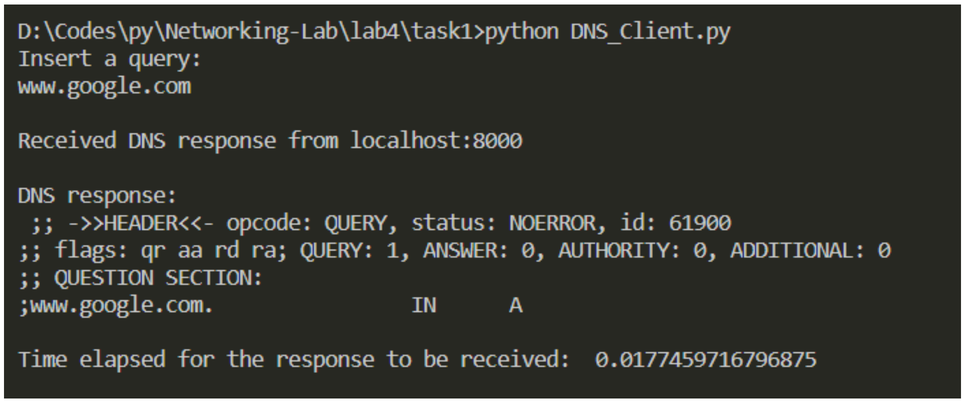
Figure 2: Authoritative DNS Server Status on No Answer

Client



```
D:\Codes\py\Networking-Lab\lab4\task1>python DNS_Client.py
Insert a query:
cse.du.ac.bd.

Received DNS response from localhost:8000

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36572
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.bd.                  IN      A
;; ANSWER SECTION:
cse.du.ac.bd.           86400   IN      NS      ns1.cse.du.ac.bd.
cse.du.ac.bd.           86400   IN      NS      ns2.cse.du.ac.bd.
cse.du.ac.bd.           86400   IN      A       192.0.2.3
cse.du.ac.bd.           86400   IN      AAAA    2001:db8::3

Time elapsed for the response to be received:  0.008928298950195312

D:\Codes\py\Networking-Lab\lab4\task1>
```

Figure 3: Client Status on Successful Answer



```
D:\Codes\py\Networking-Lab\lab4\task1>python DNS_Client.py
Insert a query:
www.google.com

Received DNS response from localhost:8000

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61900
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.google.com.                IN      A

Time elapsed for the response to be received:  0.0177459716796875
```

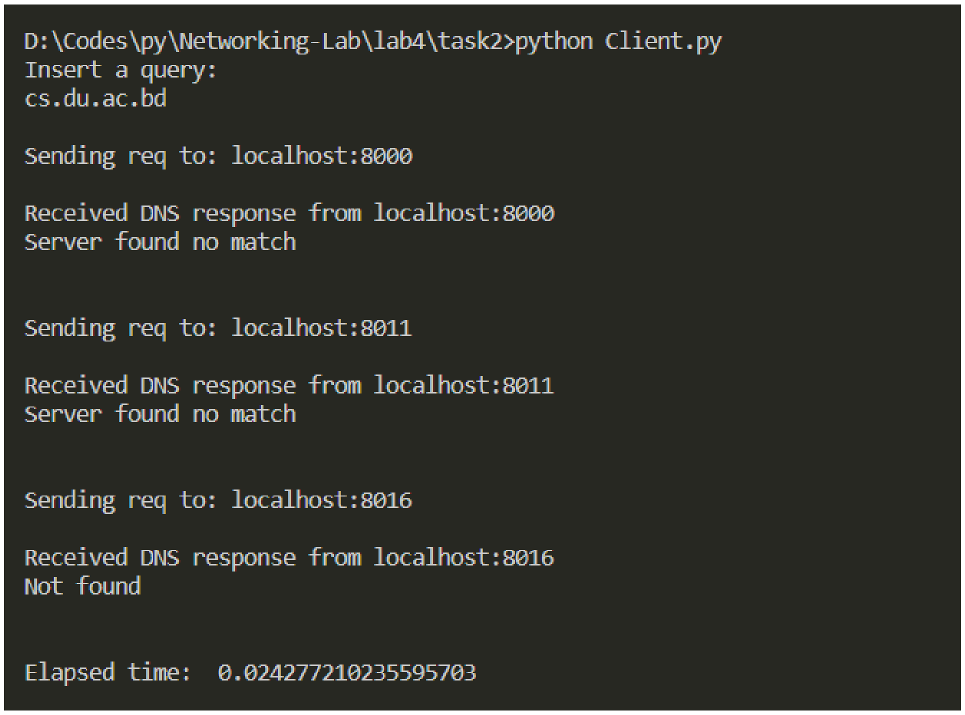
Figure 4: Client Status on No Answer

4.2 Iterative DNS resolution

Multiple servers are implemented for demonstrating the working method of an iterative DNS resolution. It seems, on average, the user gets a response in a total of 0.025s

Client

For testing we send three requests to the DNS server



```
D:\Codes\py\Networking-Lab\lab4\task2>python Client.py
Insert a query:
cs.du.ac.bd

Sending req to: localhost:8000

Received DNS response from localhost:8000
Server found no match


Sending req to: localhost:8011

Received DNS response from localhost:8011
Server found no match


Sending req to: localhost:8016

Received DNS response from localhost:8016
Not found


Elapsed time:  0.024277210235595703
```

Figure 5: A Non-Existing Domain Name Query

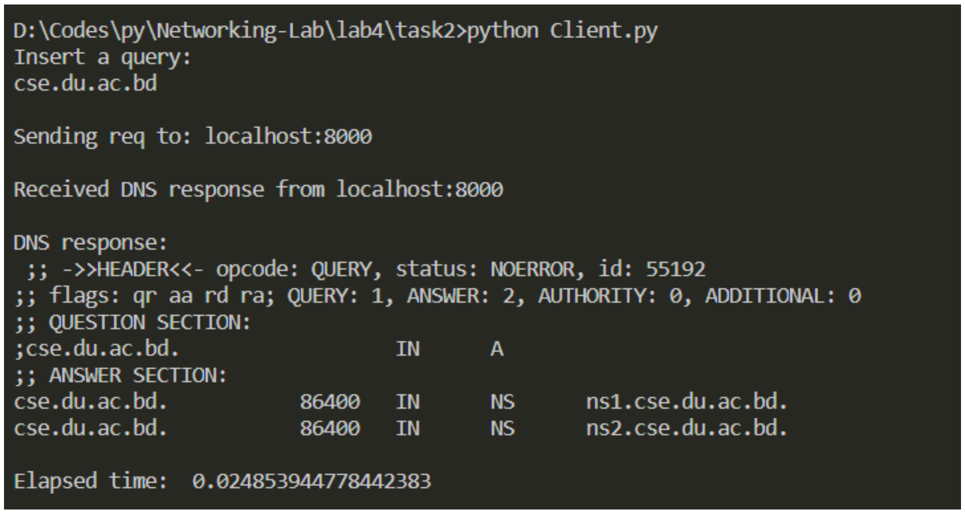
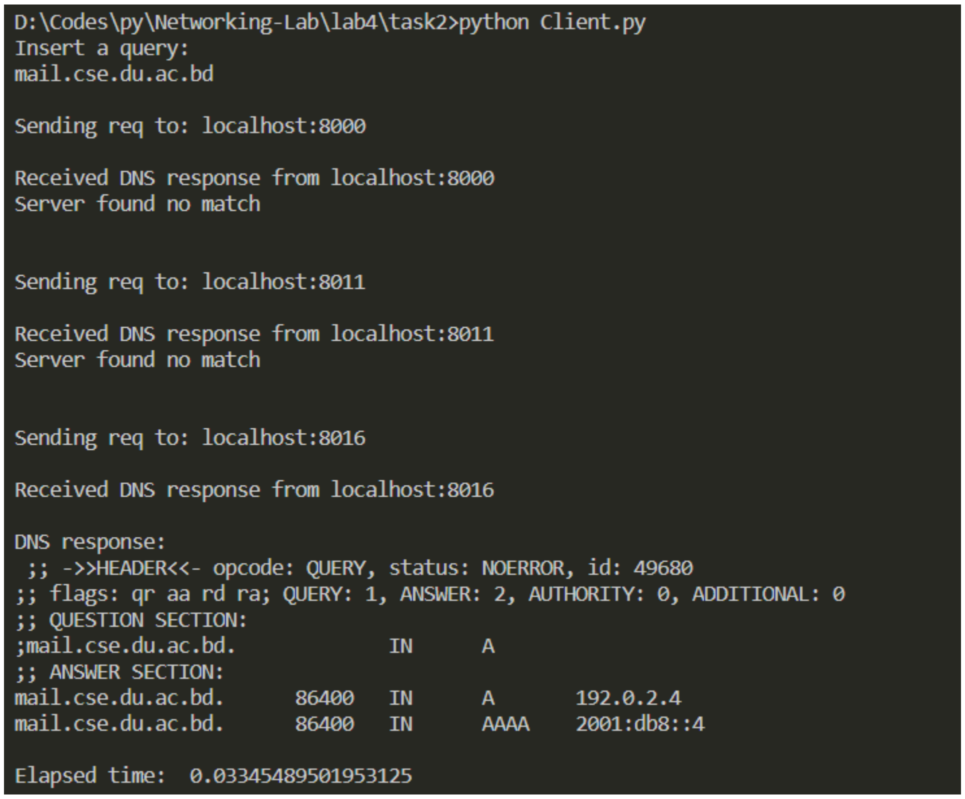

```
D:\Codes\py\Networking-Lab\lab4\task2>python Client.py
Insert a query:
cse.du.ac.bd

Sending req to: localhost:8000

Received DNS response from localhost:8000

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55192
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.bd.                  IN      A
;; ANSWER SECTION:
cse.du.ac.bd.           86400   IN      NS      ns1.cse.du.ac.bd.
cse.du.ac.bd.           86400   IN      NS      ns2.cse.du.ac.bd.

Elapsed time:  0.024853944778442383
```

Figure 6: Found in Root Server



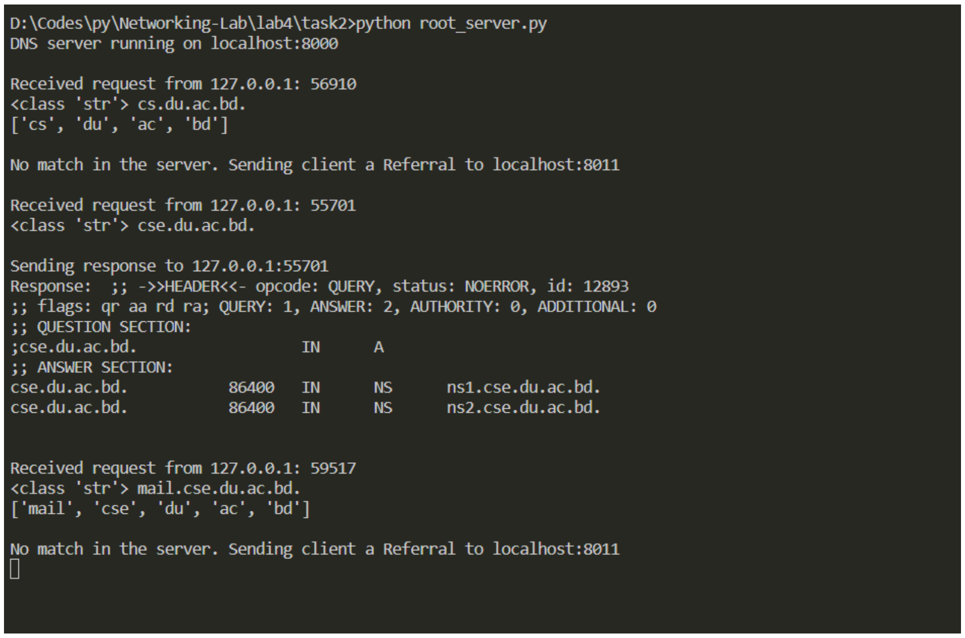
```
D:\Codes\py\Networking-Lab\lab4\task2>python Client.py
Insert a query:
mail.cse.du.ac.bd

Sending req to: localhost:8000

Received DNS response from localhost:8000
Server found no match


Sending req to: localhost:8011

Received DNS response from localhost:8011
Server found no match


Sending req to: localhost:8016

Received DNS response from localhost:8016

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49680
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;mail.cse.du.ac.bd.            IN      A
;; ANSWER SECTION:
mail.cse.du.ac.bd.      86400   IN      A       192.0.2.4
mail.cse.du.ac.bd.      86400   IN      AAAA    2001:db8::4

Elapsed time:  0.03345489501953125
```

Figure 7: Had to Go All the Way to the Authoritative Server

DNS Servers

Each server, upon receiving a request, either responds with a referral or an IP, depending on whether it has the domain or not. So the client is always making the requests iteratively

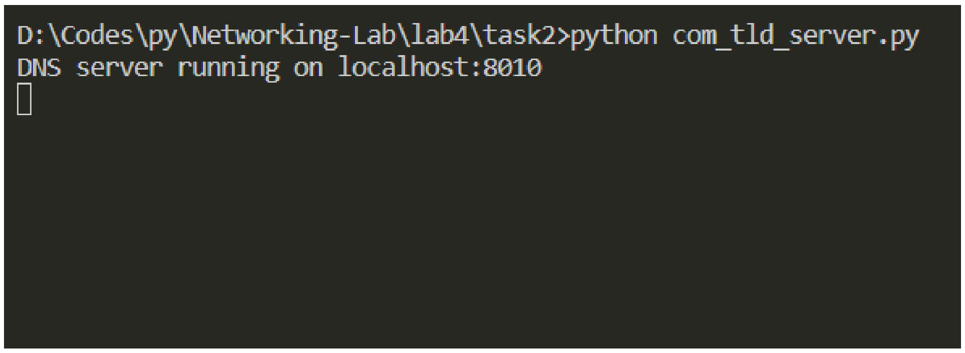
- The Root Server upon receiving the requests from the client



```
D:\Codes\py\Networking-Lab\lab4\task2>python root_server.py
DNS server running on localhost:8000

Received request from 127.0.0.1: 56910
<class 'str'> cs.du.ac.bd.
['cs', 'du', 'ac', 'bd']

No match in the server. Sending client a Referral to localhost:8011

Received request from 127.0.0.1: 55701
<class 'str'> cse.du.ac.bd.

Sending response to 127.0.0.1:55701
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12893
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.bd.                IN      A
;; ANSWER SECTION:
cse.du.ac.bd.          86400   IN      NS      ns1.cse.du.ac.bd.
cse.du.ac.bd.          86400   IN      NS      ns2.cse.du.ac.bd.


Received request from 127.0.0.1: 59517
<class 'str'> mail.cse.du.ac.bd.
['mail', 'cse', 'du', 'ac', 'bd']

No match in the server. Sending client a Referral to localhost:8011
```

Figure 8: Root Server Status after All Three Queries

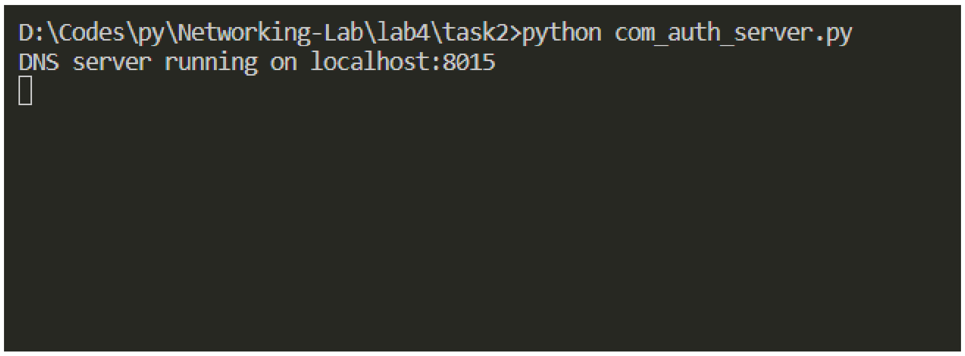
- The .com TLD server during the whole query remains idle



```
D:\Codes\py\Networking-Lab\lab4\task2>python com_tld_server.py
DNS server running on localhost:8010
```

Figure 9: TLD .com Server Status after All Three Queries

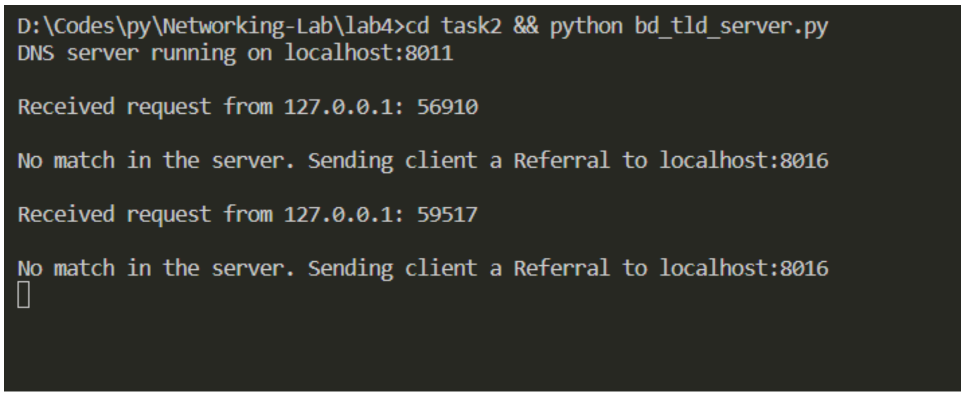
- The .com Authoritative server during the whole process also remains idle



```
D:\Codes\py\Networking-Lab\lab4\task2>python com_auth_server.py
DNS server running on localhost:8015
```

Figure 10: The .com Authoritative Server Status after All Three Queries

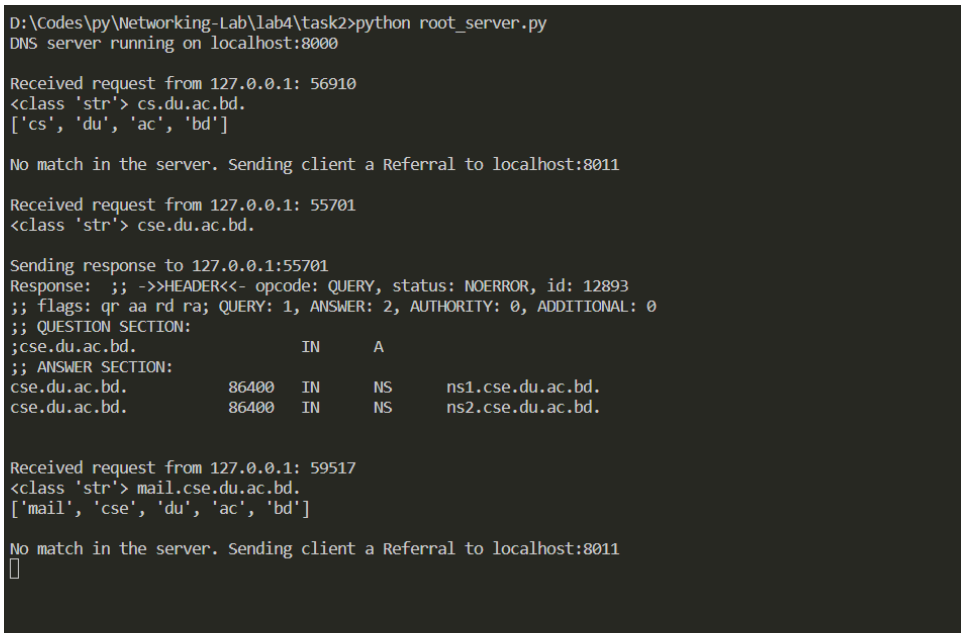
- The .bd TLD server receives the requests from client and responses



```
D:\Codes\py\Networking-Lab\lab4>cd task2 && python bd_tld_server.py
DNS server running on localhost:8011

Received request from 127.0.0.1: 56910

No match in the server. Sending client a Referral to localhost:8016

Received request from 127.0.0.1: 59517

No match in the server. Sending client a Referral to localhost:8016
```

Figure 11: The .bd TLD Server Status after All Three Queries

- The .bd Authoritative server also receives and responds accordingly to the client directly



```
D:\Codes\py\Networking-Lab\lab4\task2>python root_server.py
DNS server running on localhost:8000

Received request from 127.0.0.1: 56910
<class 'str'> cs.du.ac.bd.
['cs', 'du', 'ac', 'bd']

No match in the server. Sending client a Referral to localhost:8011

Received request from 127.0.0.1: 55701
<class 'str'> cse.du.ac.bd.

Sending response to 127.0.0.1:55701
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12893
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.bd.                  IN      A
;; ANSWER SECTION:
cse.du.ac.bd.           86400   IN      NS      ns1.cse.du.ac.bd.
cse.du.ac.bd.           86400   IN      NS      ns2.cse.du.ac.bd.


Received request from 127.0.0.1: 59517
<class 'str'> mail.cse.du.ac.bd.
['mail', 'cse', 'du', 'ac', 'bd']

No match in the server. Sending client a Referral to localhost:8011
```

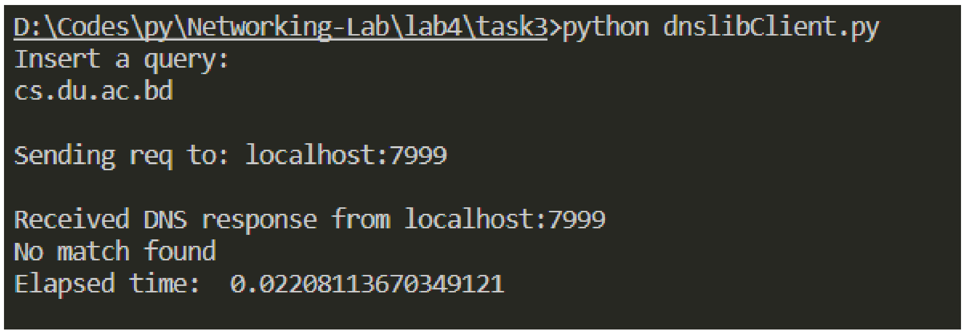
Figure 12: Root Server Status after All Three Queries

4.3 Recursive DNS resolution

In this resolution, a resolver DNS server takes on the role of querying the data from the higher order servers like the root, tld or the authoritative server. The average time it takes in this resolution for a client to receive a final response is roughly 0.015s. The client always receives from the same resolver server as there's only one resolver in this scenario

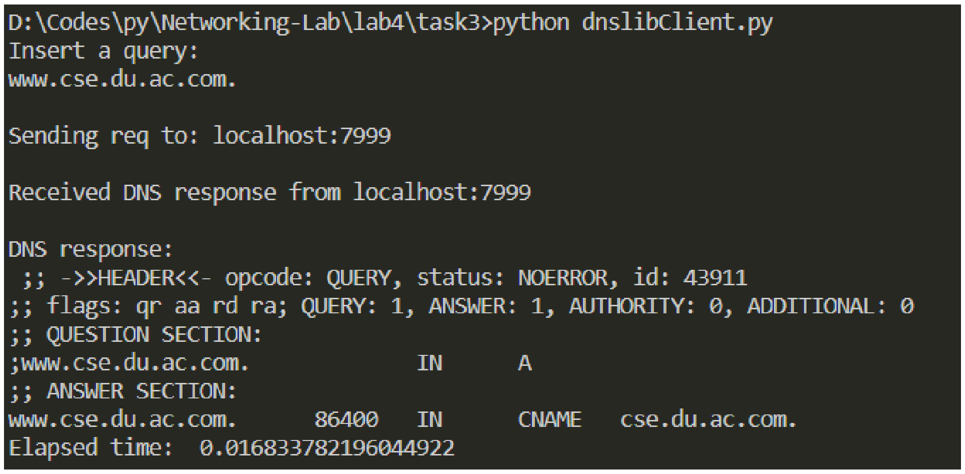
Client

For testing, the client sends three requests to the servers.



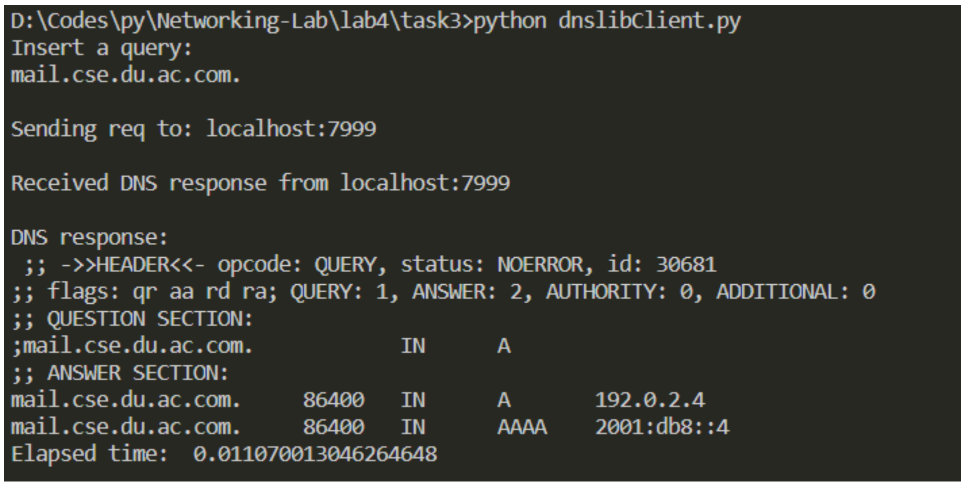
```
D:\Codes\py\Networking-Lab\lab4\task3>python dnslibClient.py
Insert a query:
cs.du.ac.bd

Sending req to: localhost:7999

Received DNS response from localhost:7999
No match found
Elapsed time:  0.02208113670349121
```

Figure 13: Client Status after Unsuccessful Query



```
D:\Codes\py\Networking-Lab\lab4\task3>python dnslibClient.py
Insert a query:
www.cse.du.ac.com.

Sending req to: localhost:7999

Received DNS response from localhost:7999

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43911
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.cse.du.ac.com.             IN      A
;; ANSWER SECTION:
www.cse.du.ac.com.      86400   IN      CNAME   cse.du.ac.com.
Elapsed time:  0.016833782196044922
```

Figure 14: Client Receives Data from TLD Server through the Same Resolver Server



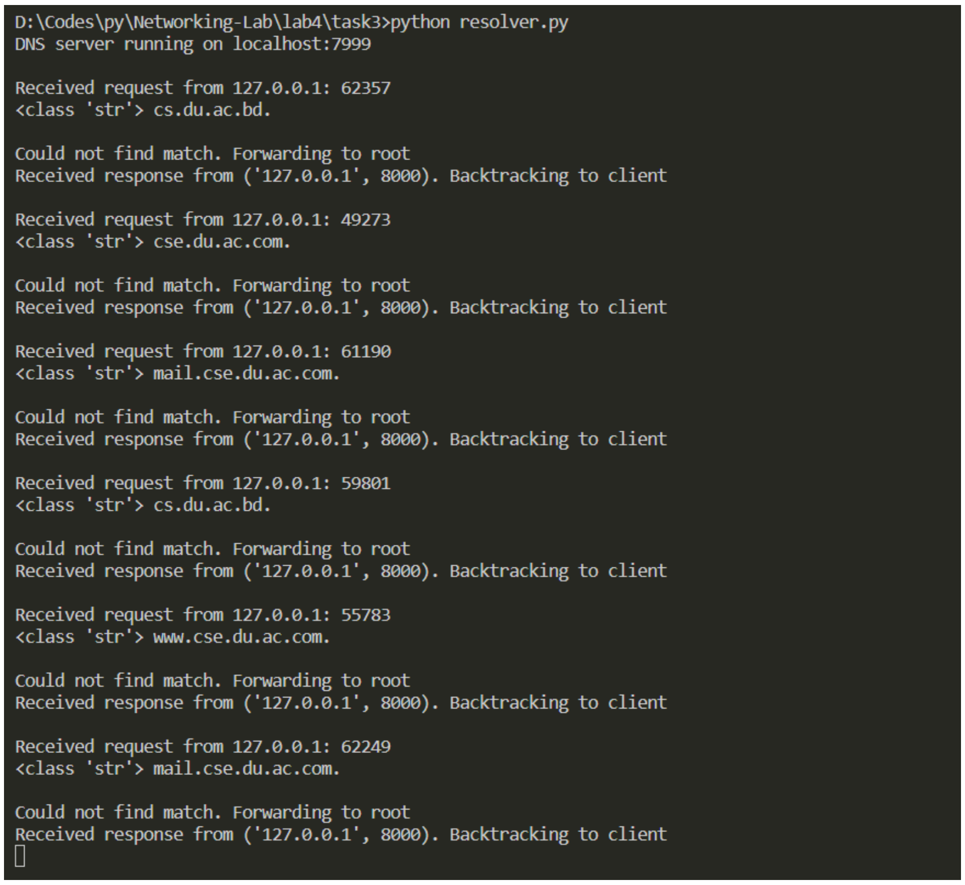
```
D:\Codes\py\Networking-Lab\lab4\task3>python dnslibClient.py
Insert a query:
mail.cse.du.ac.com.

Sending req to: localhost:7999

Received DNS response from localhost:7999

DNS response:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30681
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;mail.cse.du.ac.com.            IN      A
;; ANSWER SECTION:
mail.cse.du.ac.com.     86400   IN      A       192.0.2.4
mail.cse.du.ac.com.     86400   IN      AAAA    2001:db8::4
Elapsed time:  0.011070013046264648
```

Figure 15: Client Receives Data from Auth Server through the Same Resolver Server

DNS Servers

The recursive resolver DNS server now takes the role of making all the queries. Each server either sends an answer, or queries to another server itself and sends back the responses.

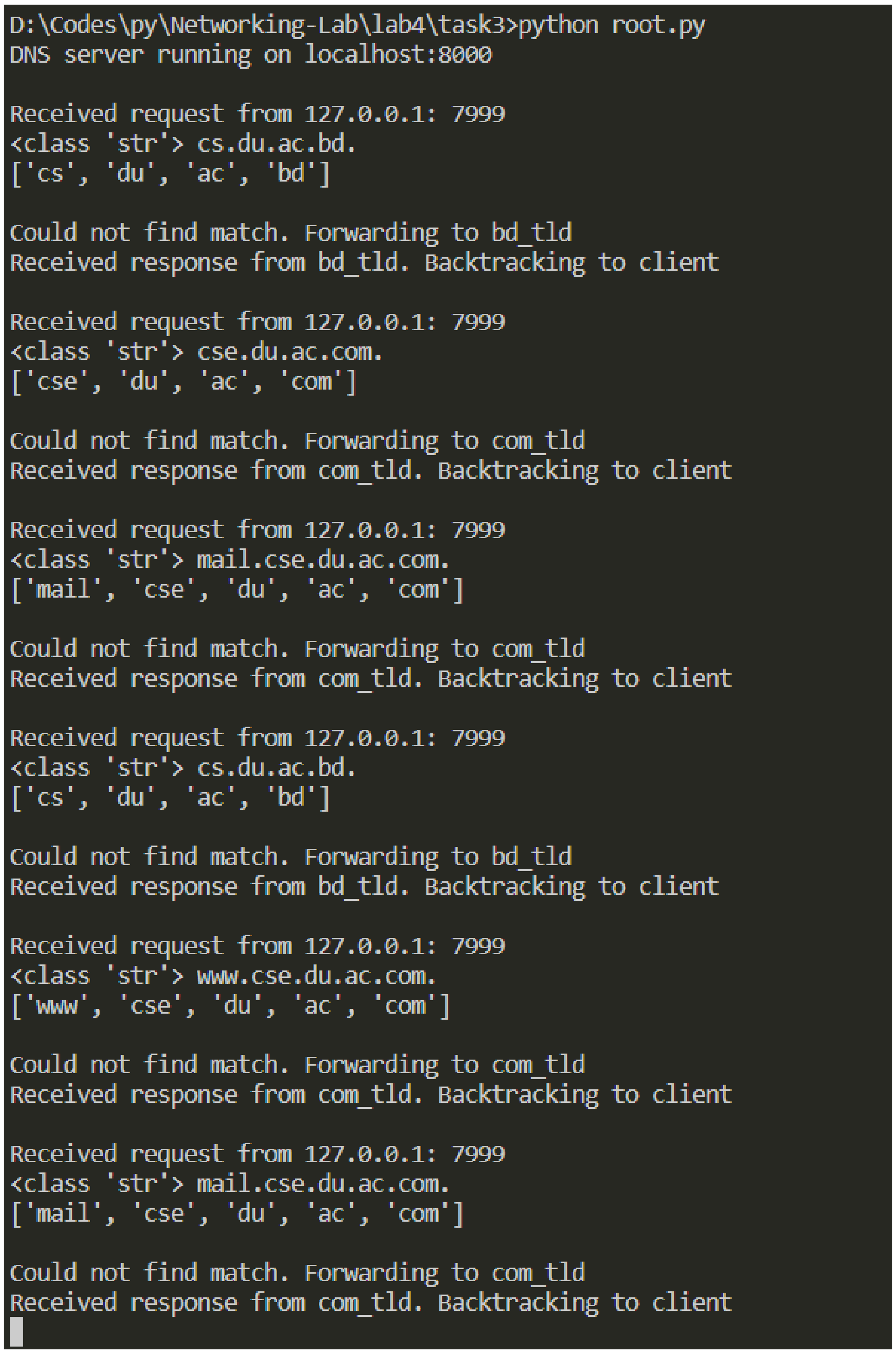
- The Recursive Resolver Server, upon receiving the requests from the client, looks in its cache, and when it doesn't find anything, it queries to the root server



```
D:\Codes\py\Networking-Lab\lab4\task3>python resolver.py
DNS server running on localhost:7999

Received request from 127.0.0.1: 62357
<class 'str'> cs.du.ac.bd.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client

Received request from 127.0.0.1: 49273
<class 'str'> cse.du.ac.com.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client

Received request from 127.0.0.1: 61190
<class 'str'> mail.cse.du.ac.com.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client

Received request from 127.0.0.1: 59801
<class 'str'> cs.du.ac.bd.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client

Received request from 127.0.0.1: 55783
<class 'str'> www.cse.du.ac.com.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client

Received request from 127.0.0.1: 62249
<class 'str'> mail.cse.du.ac.com.

Could not find match. Forwarding to root
Received response from ('127.0.0.1', 8000). Backtracking to client
```

Figure 16: Resolver Server Status after All Three Queries

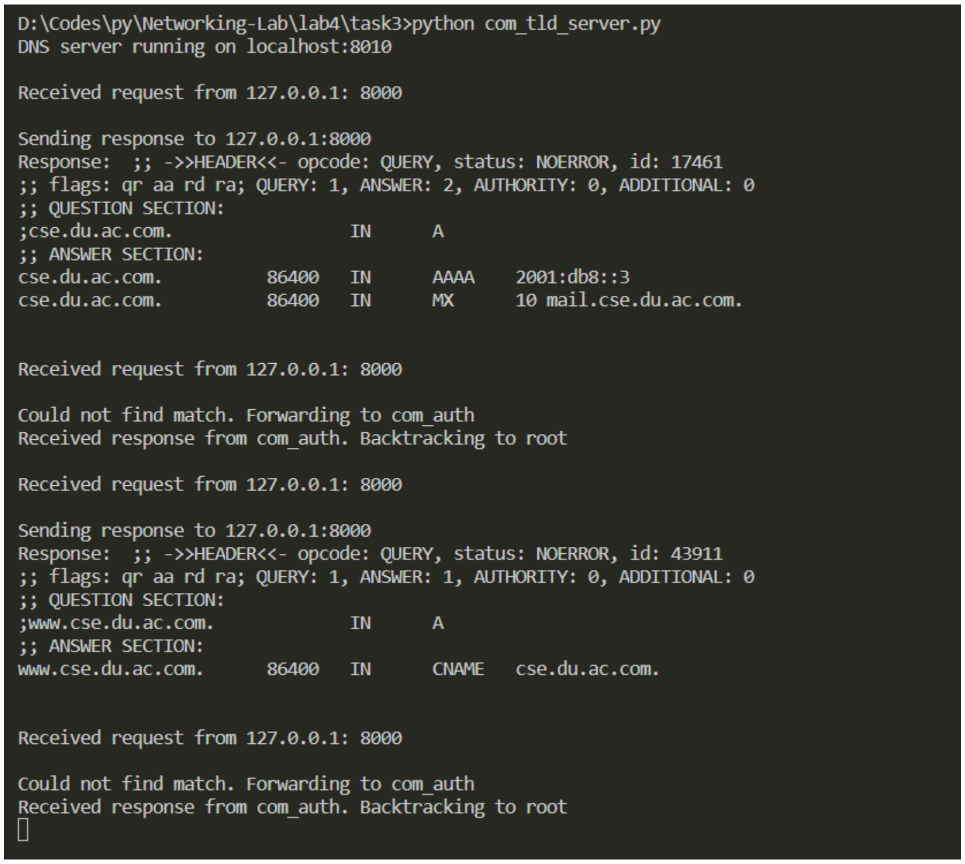
- The Root Server, upon receiving the requests from the client



```
D:\Codes\py\Networking-Lab\lab4\task3>python root.py
DNS server running on localhost:8000

Received request from 127.0.0.1: 7999
<class 'str'> cs.du.ac.bd.
['cs', 'du', 'ac', 'bd']

Could not find match. Forwarding to bd_tld
Received response from bd_tld. Backtracking to client

Received request from 127.0.0.1: 7999
<class 'str'> cse.du.ac.com.
['cse', 'du', 'ac', 'com']

Could not find match. Forwarding to com_tld
Received response from com_tld. Backtracking to client

Received request from 127.0.0.1: 7999
<class 'str'> mail.cse.du.ac.com.
['mail', 'cse', 'du', 'ac', 'com']

Could not find match. Forwarding to com_tld
Received response from com_tld. Backtracking to client

Received request from 127.0.0.1: 7999
<class 'str'> cs.du.ac.bd.
['cs', 'du', 'ac', 'bd']

Could not find match. Forwarding to bd_tld
Received response from bd_tld. Backtracking to client

Received request from 127.0.0.1: 7999
<class 'str'> www.cse.du.ac.com.
['www', 'cse', 'du', 'ac', 'com']

Could not find match. Forwarding to com_tld
Received response from com_tld. Backtracking to client

Received request from 127.0.0.1: 7999
<class 'str'> mail.cse.du.ac.com.
['mail', 'cse', 'du', 'ac', 'com']

Could not find match. Forwarding to com_tld
Received response from com_tld. Backtracking to client
```

Figure 17: Root Server Status after All Three Queries

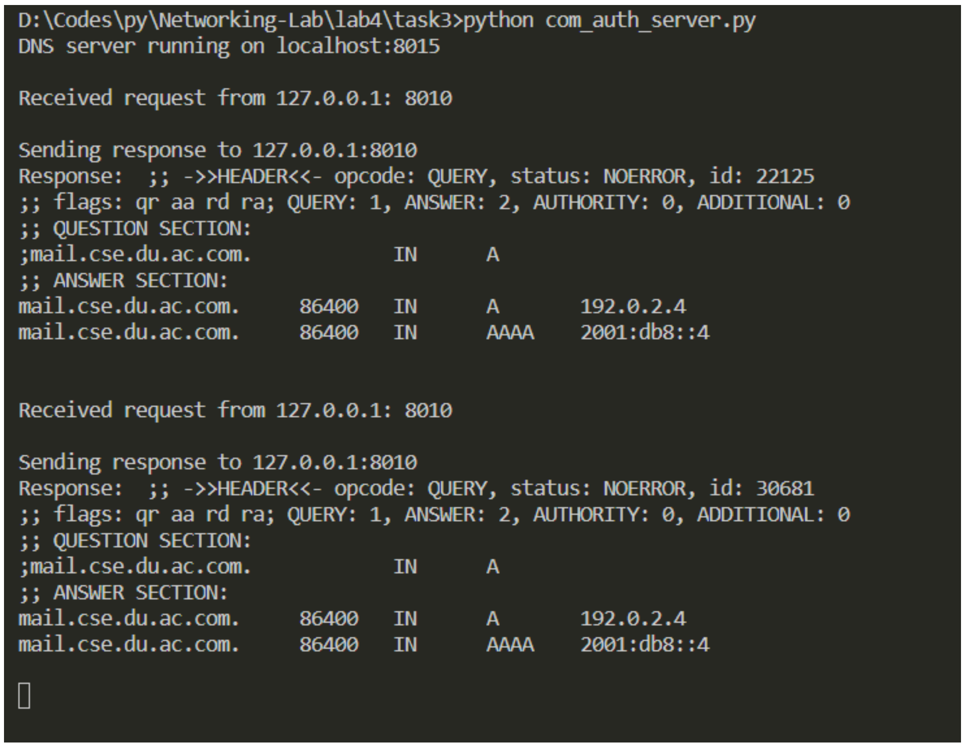
- The .com TLD server during the whole query



```
D:\Codes\py\Networking-Lab\lab4\task3>python com_tld_server.py
DNS server running on localhost:8010

Received request from 127.0.0.1: 8000

Sending response to 127.0.0.1:8000
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17461
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;cse.du.ac.com.                 IN      A
;; ANSWER SECTION:
cse.du.ac.com.          86400   IN      AAAA    2001:db8::3
cse.du.ac.com.          86400   IN      MX      10 mail.cse.du.ac.com.


Received request from 127.0.0.1: 8000

Could not find match. Forwarding to com_auth
Received response from com_auth. Backtracking to root

Received request from 127.0.0.1: 8000

Sending response to 127.0.0.1:8000
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43911
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.cse.du.ac.com.             IN      A
;; ANSWER SECTION:
www.cse.du.ac.com.      86400   IN      CNAME   cse.du.ac.com.


Received request from 127.0.0.1: 8000

Could not find match. Forwarding to com_auth
Received response from com_auth. Backtracking to root
```

Figure 18: TLD .com Server Status after All Three Queries

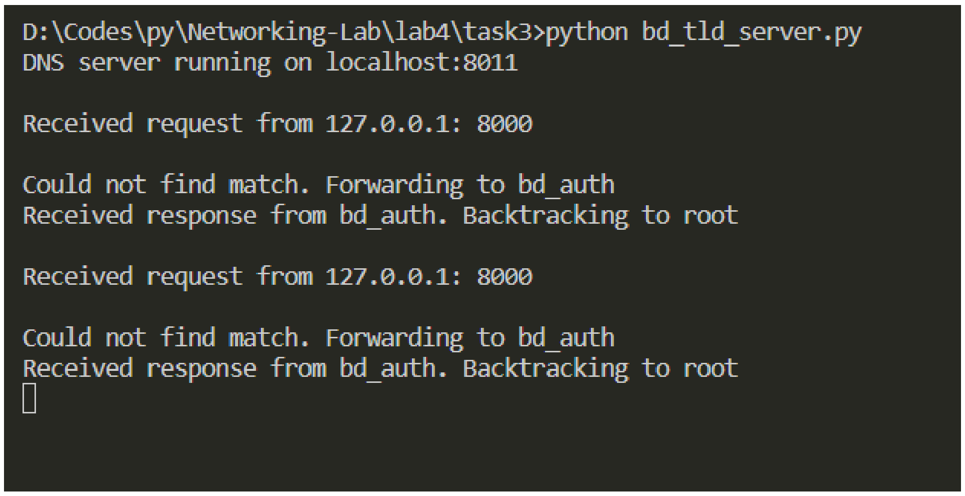
- The .com Authoritative server during the whole process



```
D:\Codes\py\Networking-Lab\lab4\task3>python com_auth_server.py
DNS server running on localhost:8015

Received request from 127.0.0.1: 8010

Sending response to 127.0.0.1:8010
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22125
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;mail.cse.du.ac.com.            IN      A
;; ANSWER SECTION:
mail.cse.du.ac.com.     86400   IN      A       192.0.2.4
mail.cse.du.ac.com.     86400   IN      AAAA    2001:db8::4


Received request from 127.0.0.1: 8010

Sending response to 127.0.0.1:8010
Response:  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30681
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;mail.cse.du.ac.com.            IN      A
;; ANSWER SECTION:
mail.cse.du.ac.com.     86400   IN      A       192.0.2.4
mail.cse.du.ac.com.     86400   IN      AAAA    2001:db8::4
```

Figure 19: The .com Authoritative Server Status after All Three Queries

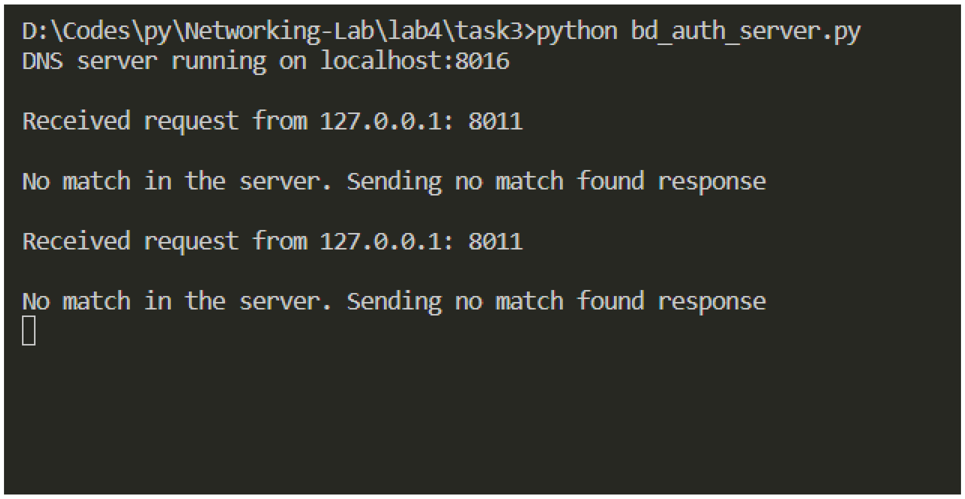
- The .bd TLD server receives an unsuccessful request from the client



```
D:\Codes\py\Networking-Lab\lab4\task3>python bd_tld_server.py
DNS server running on localhost:8011

Received request from 127.0.0.1: 8000

Could not find match. Forwarding to bd_auth
Received response from bd_auth. Backtracking to root

Received request from 127.0.0.1: 8000

Could not find match. Forwarding to bd_auth
Received response from bd_auth. Backtracking to root
```

Figure 20: The .bd TLD Server Status after All Three Queries

- The .bd Authoritative server also receives an unsuccessful query which was forwarded from the TLD server



```
D:\Codes\py\Networking-Lab\lab4\task3>python bd_auth_server.py
DNS server running on localhost:8016

Received request from 127.0.0.1: 8011

No match in the server. Sending no match found response

Received request from 127.0.0.1: 8011

No match in the server. Sending no match found response
```

Figure 21: The .bd Authoritative Server Status after All Three Queries

Conclusion

In conclusion, we can say that the recursive DNS server resolution simplifies client-side configurations. But the increased load on the recursive servers may lead to slower response. The iterative DNS resolution distributes the whole load across multiple servers. But as it's constantly sending request back to the client, it could take longer due to latency. This is what we observe when we compare the average time taken as mentioned above.

5 Experience

- We had to look up how to use the dnslib package in Python
- We had a clearer look into the DNS message format

References

[1] `https://pypi.org/project/dnslib/`

[2] `https://github.com/paulc/dnslib`

[3] `https://aws.amazon.com/route53/what-is-dns`

[4] `https://constellix.com/news/dns-record-types`