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: Meherun Farzana

Roll No : 05

Name: Mohd. Jamal Uddin Mallick

Roll No : 07

Submitted On :

February 15, 2024

Submitted To :

Dr. Md. Abdur Razzaque

Dr. Md. Mamun Or Rashid

Dr. Muhammad Ibrahim

Redwan Ahmed Rizvee

1 Introduction

The Domain Name System (DNS) is a vital component of the Internet, as it provides a way to map human-readable domain names to numerical IP addresses. DNS is a distributed database implemented in a hierarchy of name servers, which communicate with each other using application layer protocols. DNS queries can be performed in two ways: recursively or iteratively. In a recursive query, a client sends a request to a local name server, which then contacts other name servers on behalf of the client until it obtains the answer or an error. In an iterative query, a client sends a request to a local name server, which responds with either the answer or a referral to another name server, and the client repeats the process until it obtains the answer or an error.

The purpose of this lab report is to explore the concepts of distributed database management and the implementation of iterative and recursive queries of DNS records. Implementing iterative and recursive queries in a DDDBS can involve complex programming, as it requires coordination between multiple servers and the handling of multiple requests and responses. However, by utilizing a well-designed database architecture, it is possible to provide efficient and reliable resolution of DNS records in a distributed environment. By the end of this lab report, we will have a better understanding of how DNS works and how it affects the performance and reliability of the Internet.

1.1 Objectives

- The lab aims to simulate how DNS works and compare two types of DNS queries
- The client asks for the IP address of a domain name
- The name servers use DNS to find and send back the IP address if the domain name exists

2 Theory

2.1 IP address retrieval procedure

Our computer goes through several processes when we visit a domain like google.com to translate the machine-readable IP address from the human-readable web address. The following is the IP address retrieval procedure:

1. **Perform a DNS cache search:** The first location our computer searches when we ask it to resolve a hostname is in its local DNS storage. Our computer must run a DNS query to obtain the answer if it isn't already aware of it.
2. **Request to ISP DNS servers:** If the data is not locally cached, our computer makes a DNS server query to our ISP.
3. **Request to root nameservers:** These servers make a query to the root nameservers if they are unable to provide the solution. A nameserver is a machine that is always on and answers questions regarding IP addresses and domain names.

4. **Request to Top Level Domain nameservers:** The request to the Top-Level Domain (TLD) nameservers is handled by the root nameservers, who will read the first half of the request from right to left and forward it to the TLD nameservers for the.com domain (google.com). Every Top Level Domain (TLD), like.com,.org, and.bd, has its own collection of nameservers that serve as each TLD's front desk.

5. **Request to authoritative DNS servers:** After examining the following section of our request, the TLD nameservers forward our inquiry to the nameservers in charge of this particular domain. The information included in DNS records about a particular domain is known to these authoritative nameservers.

6. **Retrieve the record:** The DNS server of the ISP obtains the google.com record from the authoritative nameservers and saves it locally in its cache. It will be unnecessary for the ISP's servers to do the lookup procedure if someone else asks for the host record for google.com because they already have the answer. Every record has a time limit on it. To ensure that the data doesn't get outdated, the server will need to request a fresh copy of the record after some time.

7. **Get the response:** The ISP's server sends the record back to our machine after obtaining the solution. The IP address is retrieved from the record by the computer, which caches it before sending it to the browser. After that, the web browser establishes a connection with the webserver and loads the webpage.

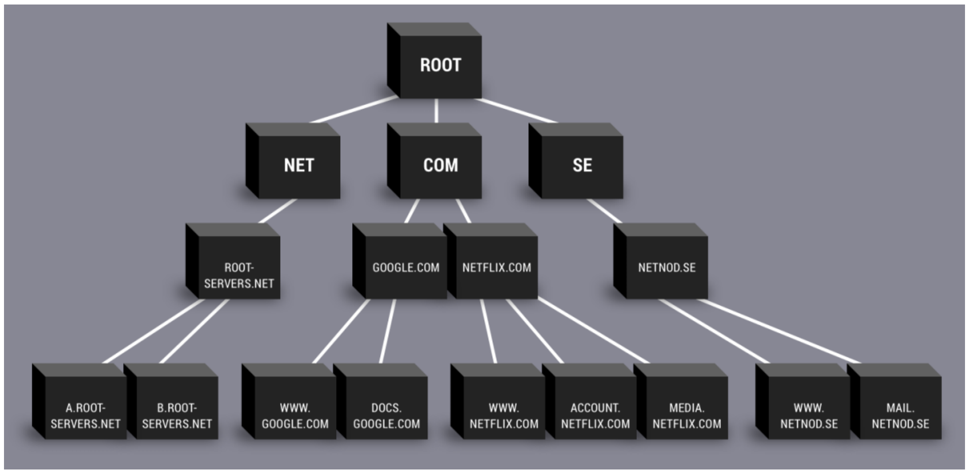


Figure 1: DNS Hierarchy

2.2 DNS Message Format

DNS eliminates the need for you to memorize lengthy numerical sequences to communicate with devices on the Internet. Two kinds of DNS messages are used to exchange data between a client and a server: message query and message of response. For both kinds of messages, the format is the same. Up to five distinct DNS message parts can contain the information.

The header and question records are the two sections that make up the query message.

The response message consists of five sections:

- Header
- Question
- Records
- Answer records
- Authoritative records
- Additional records

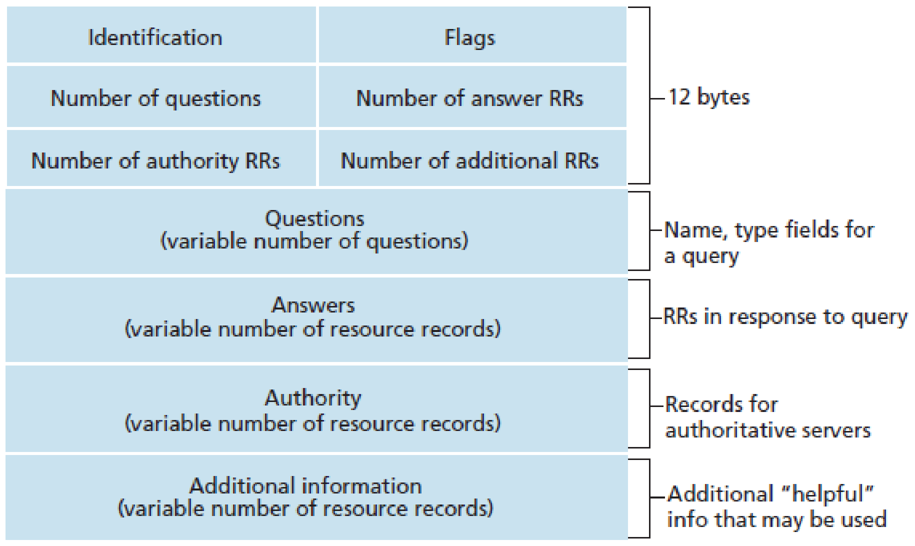

Identification
Flags
Number of questions
Number of answer RRs
Number of authority RRs
Number of additional RRs
12 bytes
Questions
(variable number of questions)
Name, type fields for a query
Answers
(variable number of resource records)
RRs in response to query
Authority
(variable number of resource records)
Records for authoritative servers
Additional information
(variable number of resource records)
Additional "helpful" info that may be used

Figure 2: DNS Message Format

3 Methodology

1. For every DNS server, a thread is created.
2. We establish a root node in a tree of DNS servers.
3. A reference to the parent server and child servers is given to each server.
4. The server is waiting for a client request.
5. The server searches for the specified domain and any children (if any) in its current location after receiving the request. If the request cannot be located, it will be forwarded to the parent.

6. The parent conducts a comparable search and notifies its parents if the request is not discovered.

7. The client will receive an error message if none of the DNS servers can locate the domain.

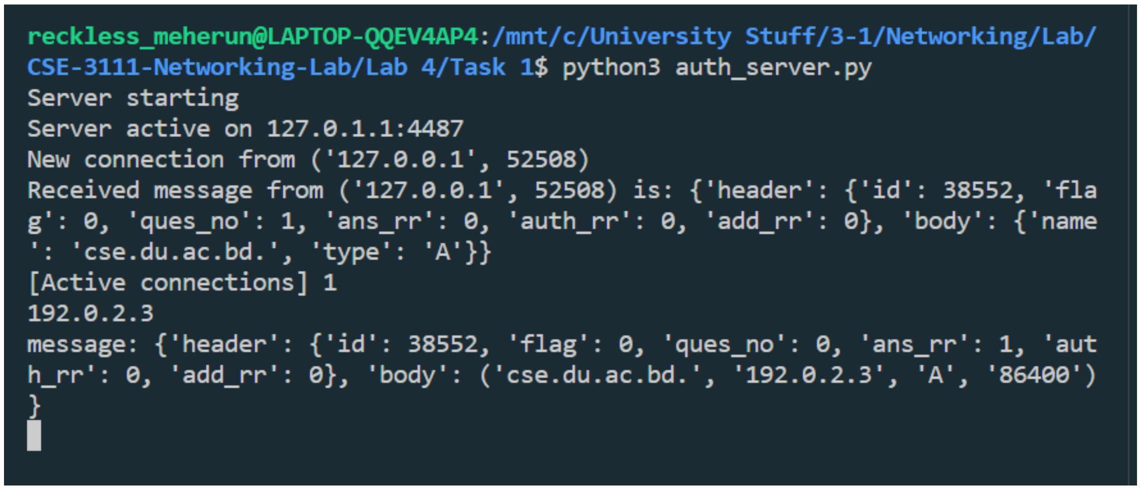
8. Should the domain be discovered on any server, the requesting client will receive a response. (Recursively/iteratively)

4 Experimental Result

4.1 Part 1: Setting up the DNS server

4.1.1 Server

The client submits a request to the server. The header and the IP address of the requested domain name are sent by the client if it wishes to know the IP address of a domain name. There is a comparable Header format for both kinds of messages. Up to five distinct DNS message parts can contain the information. The header and question records are the two sections that make up the query message.

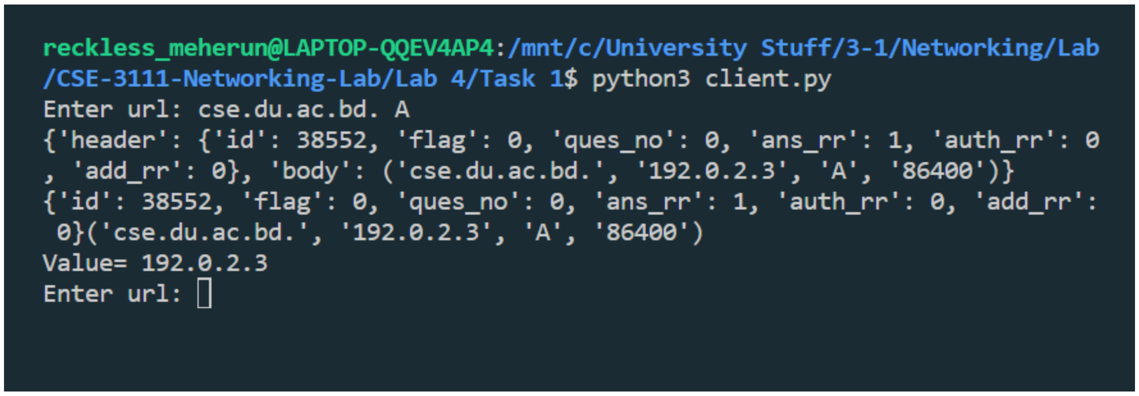


```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/University Stuff/3-1/Networking/Lab/
CSE-3111-Networking-Lab/Lab 4/Task 1$ python3 auth_server.py
Server starting
Server active on 127.0.1.1:4487
New connection from ('127.0.0.1', 52508)
Received message from ('127.0.0.1', 52508) is: {'header': {'id': 38552, 'fla
g': 0, 'ques_no': 1, 'ans_rr': 0, 'auth_rr': 0, 'add_rr': 0}, 'body': {'name
': 'cse.du.ac.bd.', 'type': 'A'}}
[Active connections] 1
192.0.2.3
message: {'header': {'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'aut
h_rr': 0, 'add_rr': 0}, 'body': ('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')
}
```

Figure 3: Server Side of Part 1

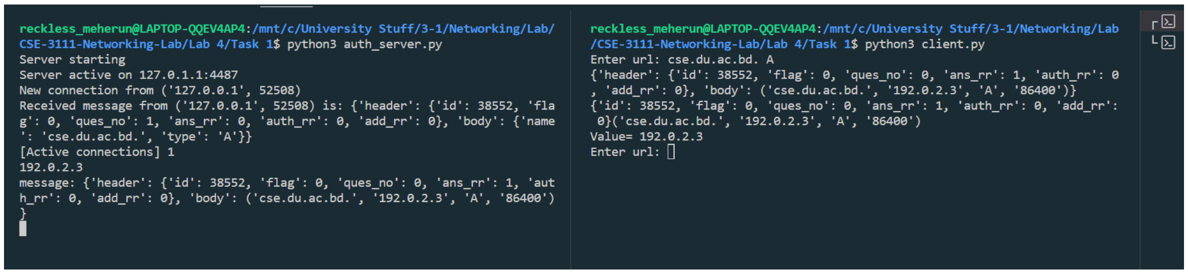
4.1.2 Client

Any domain name can be requested for an IP address by the customer. The client's screen will display the header and IP once the server has accepted the request.



```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/University Stuff/3-1/Networking/Lab
/CSE-3111-Networking-Lab/Lab 4/Task 1$ python3 client.py
Enter url: cse.du.ac.bd. A
{'header': {'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'auth_rr': 0
, 'add_rr': 0}, 'body': ('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')}
{'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'add_rr':
 0}('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')
Value= 192.0.2.3
Enter url: 
```

Figure 4: Client Side of Part 1



```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/University Stuff/3-1/Networking/Lab/
CSE-3111-Networking-Lab/Lab 4/Task 1$ python3 auth_server.py
Server starting
Server active on 127.0.1.1:4487
New connection from ('127.0.0.1', 52508)
Received message from ('127.0.0.1', 52508) is: {'header': {'id': 38552, 'fla
g': 0, 'ques_no': 1, 'ans_rr': 0, 'auth_rr': 0, 'add_rr': 0}, 'body': {'name
': 'cse.du.ac.bd.', 'type': 'A'}}
[Active connections] 1
192.0.2.3
message: {'header': {'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'aut
h_rr': 0, 'add_rr': 0}, 'body': ('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')
}
```

```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/University Stuff/3-1/Networking/Lab
/CSE-3111-Networking-Lab/Lab 4/Task 1$ python3 client.py
Enter url: cse.du.ac.bd. A
{'header': {'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'auth_rr': 0
, 'add_rr': 0}, 'body': ('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')}
{'id': 38552, 'flag': 0, 'ques_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'add_rr':
 0}('cse.du.ac.bd.', '192.0.2.3', 'A', '86400')
Value= 192.0.2.3
Enter url: 
```

Figure 5: Both Sides of Part 1

4.2 Part 2: Iterative DNS resolution

Iterative DNS resolution is the process of finding the IP address of a domain name by asking different name servers in a sequence. The name server that receives the query either responds with the IP address if it knows it, or with a referral to another name server that is closer to the answer. The query is then repeated until the IP address is found or an error occurs. For example, if you want to find the IP address of google.com, you can use an iterative DNS query as follows:

- The client can request the IP address of google.com from your local name server.
- The answer is unknown to the local name server, but it is aware of the name server for the.com top-level domain (TLD). The local server is referred to the root server.
- The local server requests the IP address of google.com from the.com name (root) server.
- The root server for the google.com domain is known to the.com name server, but it is unaware of the answer. The local server is referred to as that by the root server.
- The local server requests the IP address of google.com from the google.com authoritative server.

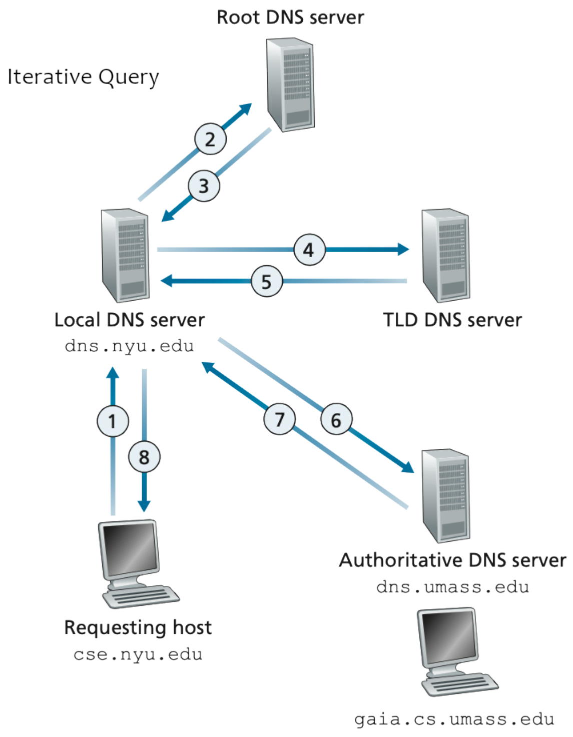
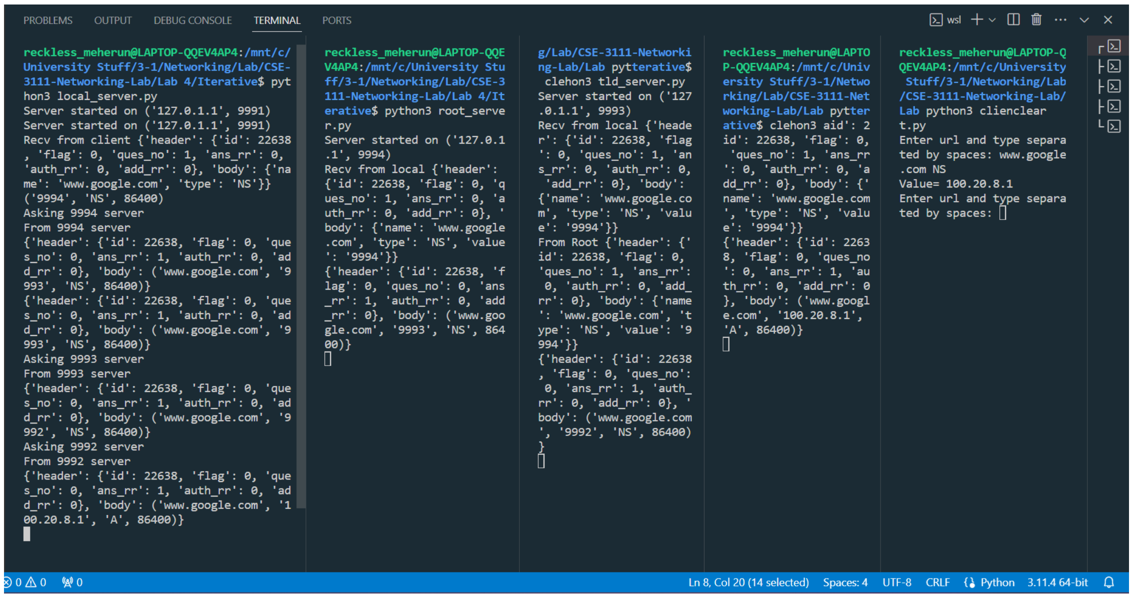


Figure 6: Iterative Approach Model

- Knowing the answer, the authoritative server replies to the local server with the google.com IP address.
- Finally the local server replies back to the client with the IP address of www.google.com.



```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/
University Stuff/3-1/Networking/Lab/CSE-
3111-Networking-Lab/Lab 4/Iterative$ pyt
hon3 local_server.py
Server started on ('127.0.1.1', 9991)
Server started on ('127.0.1.1', 9991)
Recv from client {'header': {'id': 22638
, 'flag': 0, 'ques_no': 1, 'ans_rr': 0,
'auth_rr': 0, 'add_rr': 0}, 'body': {'na
me': 'www.google.com', 'type': 'NS'}}
('9994', 'NS', 86400)
Asking 9994 server
From 9994 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
993', 'NS', 86400)}
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
993', 'NS', 86400)}
Asking 9993 server
From 9993 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
992', 'NS', 86400)}
Asking 9992 server
From 9992 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '1
00.20.8.1', 'A', 86400)}
```

```
reckless_meherun@LAPTOP-QQE
V4AP4:/mnt/c/University Stu
ff/3-1/Networking/Lab/CSE-3
111-Networking-Lab/Lab 4/It
erative$ python3 root_serve
r.py
Server started on ('127.0.1
.1', 9994)
Recv from local {'header':
{'id': 22638, 'flag': 0, 'q
ues_no': 1, 'ans_rr': 0, 'a
uth_rr': 0, 'add_rr': 0}, '
body': {'name': 'www.google
.com', 'type': 'NS', 'value
': '9994'}}
{'header': {'id': 22638, 'f
lag': 0, 'ques_no': 0, 'ans
_rr': 1, 'auth_rr': 0, 'add
_rr': 0}, 'body': ('www.goo
gle.com', '9993', 'NS', 864
00)}
```

```
g/Lab/CSE-3111-Networki
ng-Lab/Lab pytterative$
 clehon3 tld_server.py
Server started on ('127
.0.1.1', 9993)
Recv from local {'heade
r': {'id': 22638, 'flag
': 0, 'ques_no': 1, 'an
s_rr': 0, 'auth_rr': 0,
 'add_rr': 0}, 'body':
{'name': 'www.google.co
m', 'type': 'NS', 'valu
e': '9994'}}
From Root {'header': {'
id': 22638, 'flag': 0,
'ques_no': 1, 'ans_rr':
 0, 'auth_rr': 0, 'add_
rr': 0}, 'body': {'name
': 'www.google.com', 't
ype': 'NS', 'value': '9
994'}}
{'header': {'id': 22638
, 'flag': 0, 'ques_no':
 0, 'ans_rr': 1, 'auth_
rr': 0, 'add_rr': 0}, '
body': ('www.google.com
', '9992', 'NS', 86400)
}
```

```
reckless_meherun@LAPTO
P-QQEV4AP4:/mnt/c/Univ
ersity Stuff/3-1/Netwo
rking/Lab/CSE-3111-Net
working-Lab/Lab pytter
ative$ clehon3 aid': 2
id': 22638, 'flag': 0,
 'ques_no': 1, 'ans_rr
': 0, 'auth_rr': 0, 'a
dd_rr': 0}, 'body': {'
name': 'www.google.com
', 'type': 'NS', 'valu
e': '9994'}}
{'header': {'id': 2263
8, 'flag': 0, 'ques_no
': 0, 'ans_rr': 1, 'au
th_rr': 0, 'add_rr': 0
}, 'body': ('www.googl
e.com', '100.20.8.1',
'A', 86400)}
```

```
reckless_meherun@LAPTOP-Q
QEV4AP4:/mnt/c/University
 Stuff/3-1/Networking/Lab
/CSE-3111-Networking-Lab/
Lab python3 clienclear
t.py
Enter url and type separa
ted by spaces: www.google
.com NS
Value= 100.20.8.1
Enter url and type separa
ted by spaces:
```

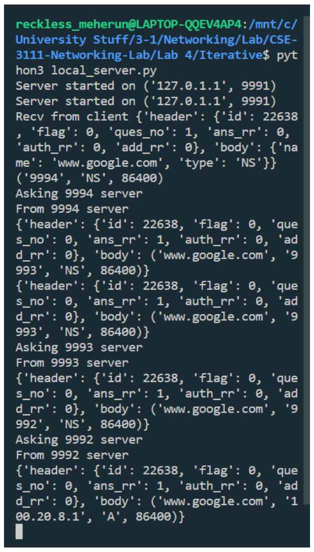
```
⊗ 0 ⚠ 0   0                    Ln 8, Col 20 (14 selected)   Spaces: 4   UTF-8   CRLF   {} Python   3.11.4 64-bit
```

Figure 7: All the servers and the client of Part 2

4.2.1 Server

There are 4 levels of servers used here. They are described below:

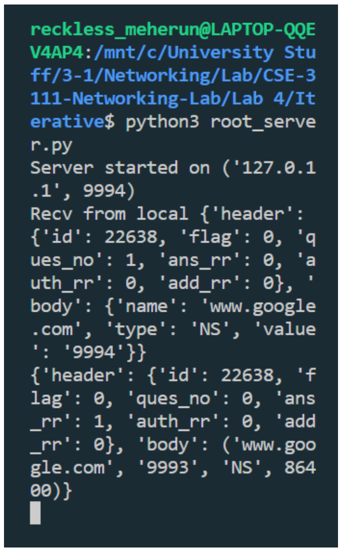
- **Local Server:** In DNS iterative resolution, the local server is the name server that receives the query from the client and contacts other name servers to find the answer. The local server does not perform recursive queries on behalf of the client, but instead returns either the IP address of the requested domain name or a referral to another name server that is closer to the answer. The local server also caches the responses it receives from other name servers so that it can answer future queries faster. The local server acts as an intermediary between the client and the DNS hierarchy, reducing the load on the root and top-level domain name servers.



```
reckless_meherun@LAPTOP-QQEV4AP4:/mnt/c/
University Stuff/3-1/Networking/Lab/CSE-
3111-Networking-Lab/Lab 4/Iterative$ pyt
hon3 local_server.py
Server started on ('127.0.1.1', 9991)
Server started on ('127.0.1.1', 9991)
Recv from client {'header': {'id': 22638
, 'flag': 0, 'ques_no': 1, 'ans_rr': 0,
'auth_rr': 0, 'add_rr': 0}, 'body': {'na
me': 'www.google.com', 'type': 'NS'}}
('9994', 'NS', 86400)
Asking 9994 server
From 9994 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
993', 'NS', 86400)}
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
993', 'NS', 86400)}
Asking 9993 server
From 9993 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '9
992', 'NS', 86400)}
Asking 9992 server
From 9992 server
{'header': {'id': 22638, 'flag': 0, 'que
s_no': 0, 'ans_rr': 1, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': ('www.google.com', '1
00.20.8.1', 'A', 86400)}
```

Figure 8: Local Server of Iterative Approach

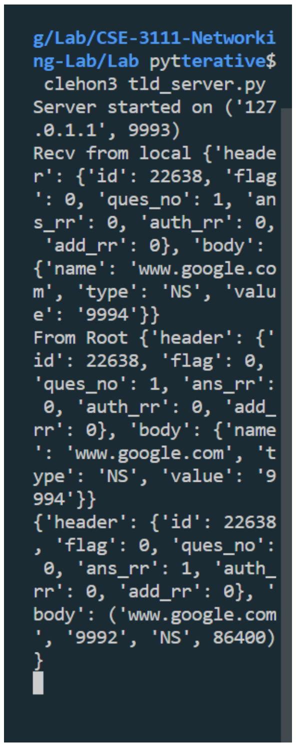
- **Root Server:** The root server is the first-name server that the resolver contacts and it responds with a referral to a TLD server based on the domain extension (such as .com, .net, etc.).



```
reckless_meherun@LAPTOP-QQE
V4AP4:/mnt/c/University Stu
ff/3-1/Networking/Lab/CSE-3
111-Networking-Lab/Lab 4/It
erative$ python3 root_serve
r.py
Server started on ('127.0.1
.1', 9994)
Recv from local {'header':
{'id': 22638, 'flag': 0, 'q
ues_no': 1, 'ans_rr': 0, 'a
uth_rr': 0, 'add_rr': 0}, '
body': {'name': 'www.google
.com', 'type': 'NS', 'value
': '9994'}}
{'header': {'id': 22638, 'f
lag': 0, 'ques_no': 0, 'ans
_rr': 1, 'auth_rr': 0, 'add
_rr': 0}, 'body': ('www.goo
gle.com', '9993', 'NS', 864
00)}
```

Figure 9: Root Server of Iterative Approach

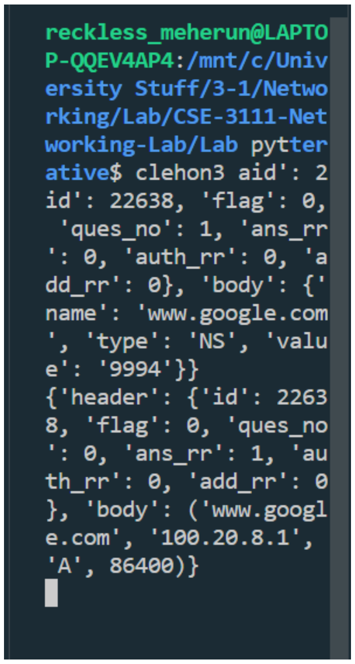
- **TLD (Top Level Domain) Server:** The TLD server is the second name server that a resolver contacts, and it responds with a referral to an authoritative server based on the domain name (such as google.com, amazon.com, etc.).



```
g/Lab/CSE-3111-Networki
ng-Lab/Lab pytterative$
 clehon3 tld_server.py
Server started on ('127
.0.1.1', 9993)
Recv from local {'heade
r': {'id': 22638, 'flag
': 0, 'ques_no': 1, 'an
s_rr': 0, 'auth_rr': 0,
 'add_rr': 0}, 'body':
{'name': 'www.google.co
m', 'type': 'NS', 'valu
e': '9994'}}
From Root {'header': {'
id': 22638, 'flag': 0,
'ques_no': 1, 'ans_rr':
 0, 'auth_rr': 0, 'add_
rr': 0}, 'body': {'name
': 'www.google.com', 't
ype': 'NS', 'value': '9
994'}}
{'header': {'id': 22638
, 'flag': 0, 'ques_no':
 0, 'ans_rr': 1, 'auth_
rr': 0, 'add_rr': 0}, '
body': ('www.google.com
', '9992', 'NS', 86400)
}
```

Figure 10: TLD Server of Iterative Approach

- **Authoritative Server:** When it comes to a particular domain name, the DNS server is in charge of giving authoritative responses. If a DNS resolver's cache does not have the information it needs to answer a domain name query, it will submit the inquiry to a DNS server. After that, this server—which is usually a recursive resolver—asks the authoritative DNS server in charge of the concerned domain. The most current and accurate data regarding the IP addresses, nameservers, and other DNS records associated with a domain can be found on the authoritative server. After receiving a response from the authoritative server, the recursive resolver might store the data for later use and send the response back to the original requester. Because it provides authoritative answers for domain names that are questioned, the authoritative server serves as the final arbiter in the DNS resolution process.

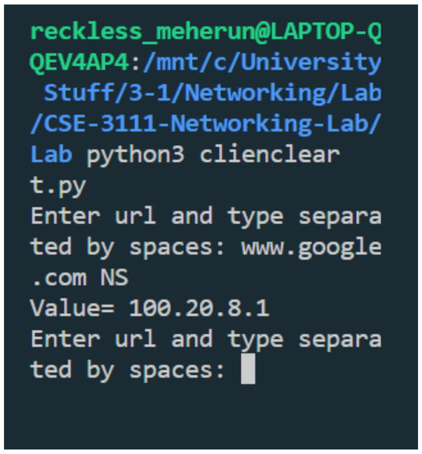


```
reckless_meherun@LAPTO
P-QQEV4AP4:/mnt/c/Univ
ersity Stuff/3-1/Netwo
rking/Lab/CSE-3111-Net
working-Lab/Lab pytter
ative$ clehon3 aid': 2
id': 22638, 'flag': 0,
 'ques_no': 1, 'ans_rr
': 0, 'auth_rr': 0, 'a
dd_rr': 0}, 'body': {'
name': 'www.google.com
', 'type': 'NS', 'valu
e': '9994'}}
{'header': {'id': 2263
8, 'flag': 0, 'ques_no
': 0, 'ans_rr': 1, 'au
th_rr': 0, 'add_rr': 0
}, 'body': ('www.googl
e.com', '100.20.8.1',
'A', 86400)}
```

Figure 11: Authoritative Server of Iterative Approach

4.2.2 Client

The client sends a DNS query packet to the local DNS server, which is the initial step of the resolution process. The client is also in charge of digesting the response it receives from the root server, which refers to the relevant TLD server, and figuring out what to do next in the resolution process.



```
reckless_meherun@LAPTOP-Q
QEV4AP4:/mnt/c/University
 Stuff/3-1/Networking/Lab
/CSE-3111-Networking-Lab/
Lab python3 clienclear
t.py
Enter url and type separa
ted by spaces: www.google
.com NS
Value= 100.20.8.1
Enter url and type separa
ted by spaces:
```

Figure 12: Client of Iterative Approach

4.2.3 Advantages

- **Reduced Security Risks:** Because iterative DNS searches rely on steps and trust only authoritative DNS servers, they are less susceptible to DNS assaults such as cache poisoning.
- **Better Control:** Because users can select which authoritative DNS servers to query, clients have greater influence over the DNS resolution process.
- **Enhanced Reliability:** Because clients can retry searches with alternative authoritative servers, iterative DNS can be more tolerant to network outages and disturbances.

4.2.4 Disadvantages

- **Complexity:** Compared to recursive DNS, iterative DNS requires a more complicated client-side implementation since clients must manage several steps in the resolution process.
- **Increased Latency:** Compared to recursive DNS, iterative DNS can cause more latency because it requires more steps and may ask many authoritative servers, particularly when resolving complicated queries.
- **Resource Intensive:** Because iterative DNS inquiries handle numerous queries and DNS response handling, they may require more network resources and computing overhead on the client side.
- **DNSSEC Implementation:** Iterative DNS makes it more important to implement DNSSEC (Domain Name System Security Extensions) to guarantee the integrity and validity of DNS answers.
- **Reduced Attack Surface:** By restricting trust to authoritative DNS servers and minimizing reliance on potentially unreliable recursive servers, iterative DNS can lessen the attack surface compared to recursive DNS, even though it doesn't eliminate security threats.

4.3 Part 3: Recursive DNS resolution

Recursive DNS resolution refers to how a DNS server answers requests for which it is not authoritative by utilizing the hierarchy of zones and delegations. A list of names and IP addresses known as root hints is included in some DNS server setups, allowing the servers to query the DNS root servers. Upon receiving a query from a client, a DNS server looks up the answer first in its zones and cache. If it is unable to locate the solution, it queries one of the root servers. In response, the root server points users to a top-level domain (TLD) server that is closer to the solution, like.com or.net. The TLD server then receives a query from the DNS server and refers users to an authoritative server for that domain, such as google.com or amazon.com. After that, the DNS server queries the authoritative server, which replies with the requested domain name's IP address or, if the domain name is not registered, an error. After that, the DNS server sends the client either the error or the response. Until the DNS server determines the solution or an error, this process is repeated. With recursive DNS resolution, the client has less work to do because the DNS server handles all of the client's queries. Still, the DNS server has to submit more requests and wait for answers, which adds to its workload. If the DNS server is improperly configured, recursive DNS resolution may also provide security issues like cache poisoning or denial-of-service attacks.

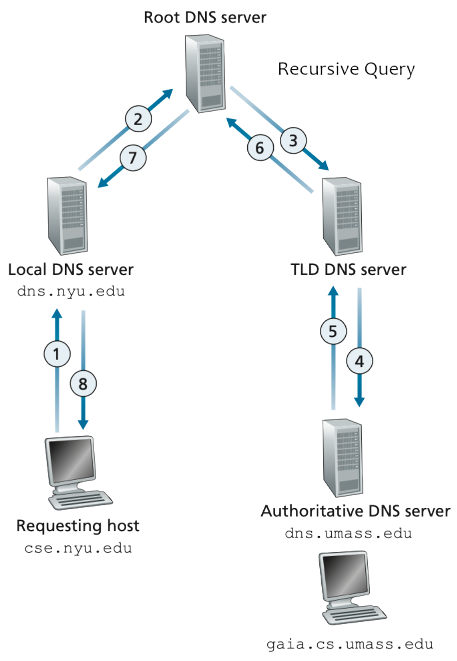
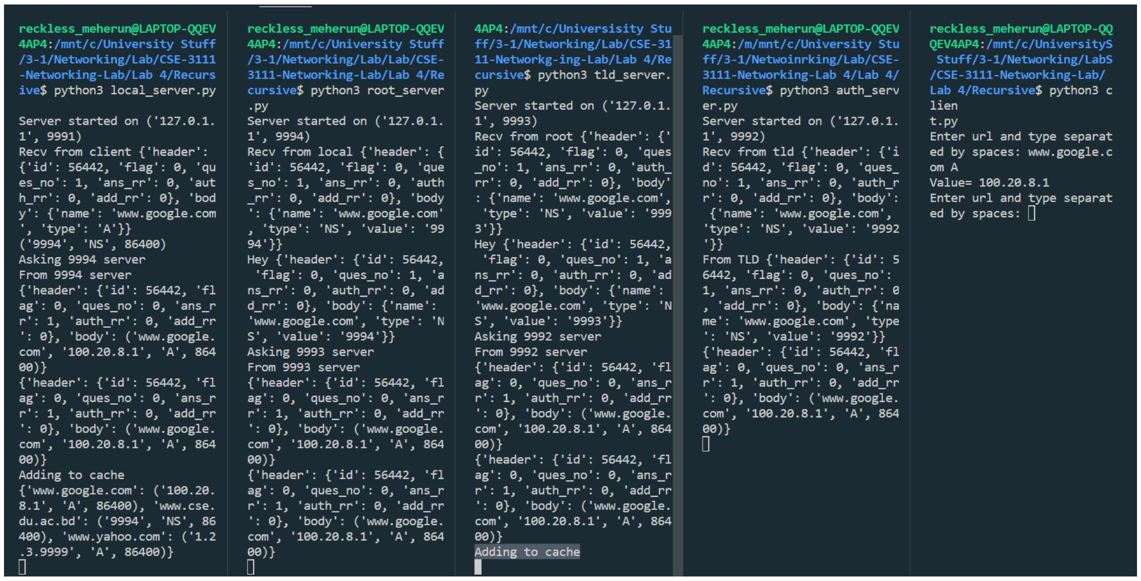


Figure 13: Recursive Approach Model



```
reckless_meherun@LAPTOP-QQEV
4AP4:/mnt/c/University Stuff
/3-1/Networking/Lab/CSE-3111
-Networking-Lab/Lab 4/Recurs
ive$ python3 local_server.py

Server started on ('127.0.1.
1', 9991)
Recv from client {'header':
{'id': 56442, 'flag': 0, 'qu
es_no': 1, 'ans_rr': 0, 'aut
h_rr': 0, 'add_rr': 0}, 'bod
y': {'name': 'www.google.com
', 'type': 'A'}}
('9994', 'NS', 86400)
Asking 9994 server
From 9994 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
Adding to cache
{'www.google.com': ('100.20.
8.1', 'A', 86400), 'www.cse.
du.ac.bd': ('9994', 'NS', 86
400), 'www.yahoo.com': ('1.2
.3.9999', 'A', 86400)}
▯
```

```
reckless_meherun@LAPTOP-QQEV
4AP4:/mnt/c/University Stuff
/3-1/Networking/Lab/Lab/CSE-
3111-Networking-Lab/Lab 4/Re
cursive$ python3 root_server
.py
Server started on ('127.0.1.
1', 9994)
Recv from local {'header': {
'id': 56442, 'flag': 0, 'que
s_no': 1, 'ans_rr': 0, 'auth
_rr': 0, 'add_rr': 0}, 'body
': {'name': 'www.google.com'
, 'type': 'NS', 'value': '99
94'}}
Hey {'header': {'id': 56442,
 'flag': 0, 'ques_no': 1, 'a
ns_rr': 0, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': {'name':
'www.google.com', 'type': 'N
S', 'value': '9994'}}
Asking 9993 server
From 9993 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
▯
```

```
4AP4:/mnt/c/Universisity Stu
ff/3-1/Networking/Lab/CSE-31
11-Networkg-ing-Lab/Lab 4/Re
cursive$ python3 tld_server.
py
Server started on ('127.0.1.
1', 9993)
Recv from root {'header': {'
id': 56442, 'flag': 0, 'ques
_no': 1, 'ans_rr': 0, 'auth_
rr': 0, 'add_rr': 0}, 'body'
: {'name': 'www.google.com',
 'type': 'NS', 'value': '999
3'}}
Hey {'header': {'id': 56442,
 'flag': 0, 'ques_no': 1, 'a
ns_rr': 0, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': {'name':
'www.google.com', 'type': 'N
S', 'value': '9993'}}
Asking 9992 server
From 9992 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
Adding to cache
█
```

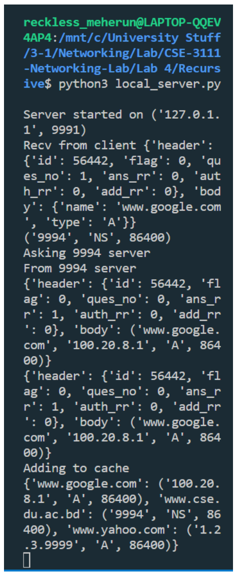
```
reckless_meherun@LAPTOP-QQEV
4AP4:/m/mnt/c/University Stu
ff/3-1/Netwoinrking/Lab/CSE-
3111-Networking-Lab 4/Lab 4/
Recursive$ python3 auth_serv
er.py
Server started on ('127.0.1.
1', 9992)
Recv from tld {'header': {'i
d': 56442, 'flag': 0, 'ques_
no': 1, 'ans_rr': 0, 'auth_r
r': 0, 'add_rr': 0}, 'body':
 {'name': 'www.google.com',
'type': 'NS', 'value': '9992
'}}
From TLD {'header': {'id': 5
6442, 'flag': 0, 'ques_no':
1, 'ans_rr': 0, 'auth_rr': 0
, 'add_rr': 0}, 'body': {'na
me': 'www.google.com', 'type
': 'NS', 'value': '9992'}}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
▯
```

```
reckless_meherun@LAPTOP-QQ
QEV4AP4:/mnt/c/UniversityS
 Stuff/3-1/Networking/LabS
/CSE-3111-Networking-Lab/
Lab 4/Recursive$ python3 c
lien
t.py
Enter url and type separat
ed by spaces: www.google.c
om A
Value= 100.20.8.1
Enter url and type separat
ed by spaces: ▯
```

Figure 14: All the Servers and the Client of Recursive Approach

4.3.1 Server

- **Local DNS Server:** As a proxy, the local DNS server answers all of the client's queries and delivers either an error or the whole response. In order to respond to queries from users more quickly in the future, the local DNS server additionally caches the answers it receives from other name servers. The client's effort is lessened by the local DNS server as it only needs to send one query and wait for one answer. But because the local DNS server must send and receive many queries and responses, it also needs to work harder. If the local DNS server is not set up correctly, recursive DNS resolution can potentially result in security problems like cache poisoning or denial-of-service attacks.

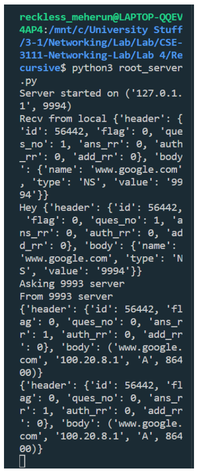


```
reckless_meherun@LAPTOP-QQEV
4AP4:/mnt/c/University Stuff
/3-1/Networking/Lab/CSE-3111
-Networking-Lab/Lab 4/Recurs
ive$ python3 local_server.py

Server started on ('127.0.1.
1', 9991)
Recv from client {'header':
{'id': 56442, 'flag': 0, 'qu
es_no': 1, 'ans_rr': 0, 'aut
h_rr': 0, 'add_rr': 0}, 'bod
y': {'name': 'www.google.com
', 'type': 'A'}}
('9994', 'NS', 86400)
Asking 9994 server
From 9994 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
Adding to cache
{'www.google.com': ('100.20.
8.1', 'A', 86400), 'www.cse.
du.ac.bd': ('9994', 'NS', 86
400), 'www.yahoo.com': ('1.2
.3.9999', 'A', 86400)}
```

Figure 15: Local Server of Recursive Approach

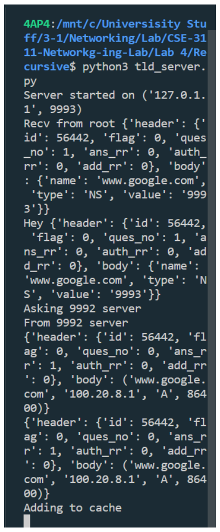
- **Root Server:** Top-level domains (TLDs) like.com,.net,.org, and so forth are stored with information about them kept on a root server. When a local DNS server receives a query from a client, it first contacts a root server in the DNS recursive resolution process. In response, the root server points to a TLD server that is more likely to have the solution. After establishing a connection with the TLD server, the local DNS server keeps trying until it either discovers the IP address of the requested domain name or encounters an error.



```
reckless_meherun@LAPTOP-QQEV
4AP4:/mnt/c/University Stuff
/3-1/Networking/Lab/Lab/CSE-
3111-Networking-Lab/Lab 4/Re
cursive$ python3 root_server
.py
Server started on ('127.0.1.
1', 9994)
Recv from local {'header': {
'id': 56442, 'flag': 0, 'que
s_no': 1, 'ans_rr': 0, 'auth
_rr': 0, 'add_rr': 0}, 'body
': {'name': 'www.google.com'
, 'type': 'NS', 'value': '99
94'}}
Hey {'header': {'id': 56442,
 'flag': 0, 'ques_no': 1, 'a
ns_rr': 0, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': {'name':
'www.google.com', 'type': 'N
S', 'value': '9994'}}
Asking 9993 server
From 9993 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
```

Figure 16: Root Server of Recursive Approach

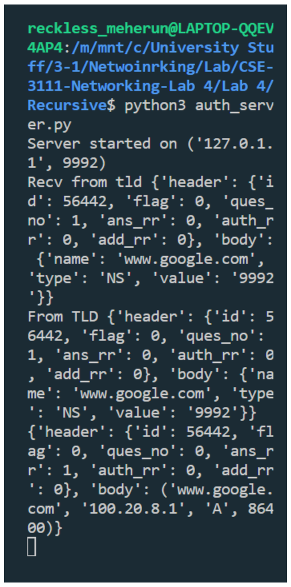
- **TLD Server:** A TLD server is the second-name server that the local DNS server contacts during DNS recursive resolution following a referral from a root server. In response, the TLD server points users to an authoritative server—such as google.com or amazon.com—that is in charge of that particular domain name. After establishing a connection with the authoritative server, the local DNS server keeps trying until it either discovers the IP address of the requested domain name or encounters an error.



```
4AP4:/mnt/c/Universisity Stu
ff/3-1/Networking/Lab/CSE-31
11-Networkg-ing-Lab/Lab 4/Re
cursive$ python3 tld_server.
py
Server started on ('127.0.1.
1', 9993)
Recv from root {'header': {'
id': 56442, 'flag': 0, 'ques
_no': 1, 'ans_rr': 0, 'auth_
rr': 0, 'add_rr': 0}, 'body'
: {'name': 'www.google.com',
 'type': 'NS', 'value': '999
3'}}
Hey {'header': {'id': 56442,
 'flag': 0, 'ques_no': 1, 'a
ns_rr': 0, 'auth_rr': 0, 'ad
d_rr': 0}, 'body': {'name':
'www.google.com', 'type': 'N
S', 'value': '9993'}}
Asking 9992 server
From 9992 server
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
Adding to cache
```

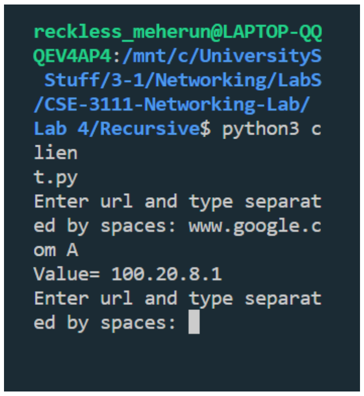
Figure 17: TLD Server of Recursive Approach

- **Authoritative Server:** Recursive DNS resolution entails the DNS server contacting the domain's authoritative name server to obtain the IP address associated with the domain name. Once the DNS server receives the answer from the authoritative name server, it forwards the domain name's IP address back to the original requester. As authoritative name servers are responsible for maintaining the most recent DNS entries for domain names, they are essential to the DNS system's functionality. Without authoritative name servers, it would be impossible to resolve DNS queries or use names to access websites and other Internet services.



```
reckless_meherun@LAPTOP-QQEV
4AP4:/m/mnt/c/University Stu
ff/3-1/Netwoinrking/Lab/CSE-
3111-Networking-Lab 4/Lab 4/
Recursive$ python3 auth_serv
er.py
Server started on ('127.0.1.
1', 9992)
Recv from tld {'header': {'i
d': 56442, 'flag': 0, 'ques_
no': 1, 'ans_rr': 0, 'auth_r
r': 0, 'add_rr': 0}, 'body':
 {'name': 'www.google.com',
'type': 'NS', 'value': '9992
'}}
From TLD {'header': {'id': 5
6442, 'flag': 0, 'ques_no':
1, 'ans_rr': 0, 'auth_rr': 0
, 'add_rr': 0}, 'body': {'na
me': 'www.google.com', 'type
': 'NS', 'value': '9992'}}
{'header': {'id': 56442, 'fl
ag': 0, 'ques_no': 0, 'ans_r
r': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.
com', '100.20.8.1', 'A', 864
00)}
```

Figure 18: Authoritative Server of Recursive Approach



```
reckless_meherun@LAPTOP-QQ
QEV4AP4:/mnt/c/UniversityS
 Stuff/3-1/Networking/LabS
/CSE-3111-Networking-Lab/
Lab 4/Recursive$ python3 c
lien
t.py
Enter url and type separat
ed by spaces: www.google.c
om A
Value= 100.20.8.1
Enter url and type separat
ed by spaces:
```

Figure 19: Client of Recursive Approach

4.3.2 Client

The term "client" refers to any software or device that needs to interact online with a server or service. A client sends a query to a recursive DNS server asking for domain name resolution; the server responds with the domain name. The recursive DNS server then hierarchically searches additional DNS servers on behalf of the client until it locates the IP address linked to the requested domain name. The client is typically unaware of the details of the DNS resolution process because it is managed by the recursive DNS server. The recursive DNS server tells the client and allows it to connect to the specified site after it has obtained the IP address for the requested domain name.

4.3.3 Advantages

- **Convenience:**Recursive DNS offers a straightforward resolution process by answering the whole query on the client's behalf.
- **Efficiency:** By caching answers, recursive DNS minimizes the time and resources required for recurrent domain searches.
- **User Privacy:** To improve user privacy, recursive DNS servers have the option to encrypt their conversations.
- **Ease of Configuration:** Usually, minimum client configuration is needed for recursive DNS.

4.3.4 Disadvantages

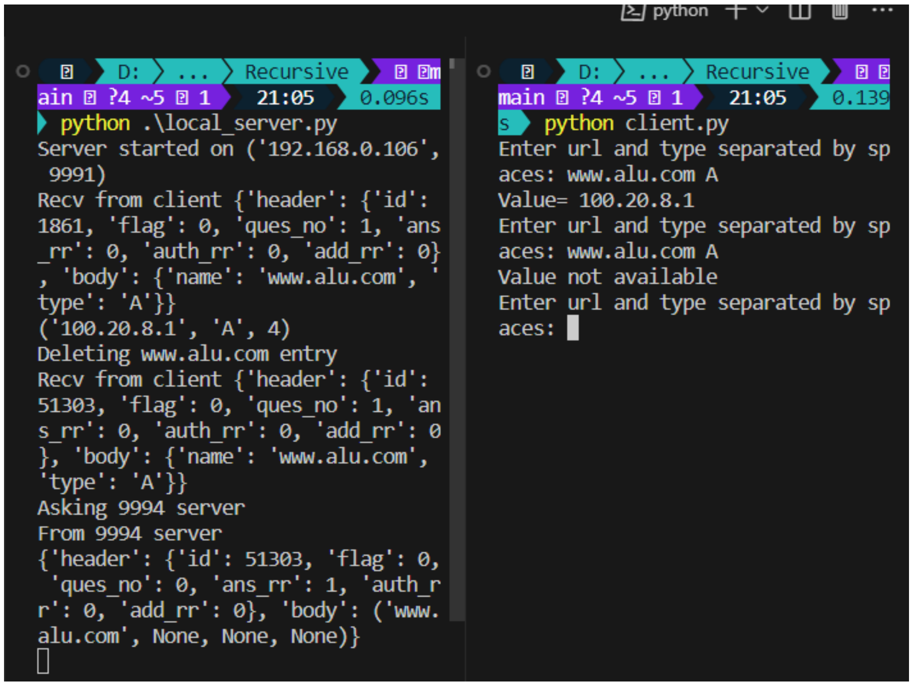
- **Potential for Amplification Attacks:** DNS amplification attacks, in which the attacker impersonates the source IP address and sends a tiny DNS query to the server, which subsequently delivers a bigger response to the victim, can be used against recursive DNS servers.
- **Dependence on Trust:** It can be troublesome if the recursive DNS server is compromised or malicious because clients need to trust it to deliver secure and correct responses.
- **Potential for DNS Spoofing:** DNS spoofing attacks, in which a hacker presents fictitious DNS information to reroute users to malicious websites, can affect recursive DNS requests.
- **Cache Poisoning:** Cache poisoning attacks, in which malicious material is injected into the cache to cause subsequent clients to receive wrong DNS resolutions, can affect recursive DNS servers.
- **Man-in-the-Middle Attacks:** Recursive DNS searches sometimes involve several DNS servers, which increases the possibility of attackers intercepting and changing DNS answers, which could result in data tampering or redirection.

4.4 Part 4: Extending the System

4.4.1 Deleting resource record based on TTL value

Resource records in the Domain Name System (DNS) hold details on domain names and related information such IP addresses, mail server addresses, and other kinds of records. A Time-to-Live (TTL) value is assigned to each resource record, indicating the duration for which DNS resolvers or servers may cache the data. When a resource record's TTL expires, it is deleted based on its value, which removes the record from storage or the cache.

To implement this, a thread is running to decrease the **Time To Live (TTL)** by 1 second. Whenever the TTL gets 0, the record is being deleted. Thus a record can be deleted based on TTL value.



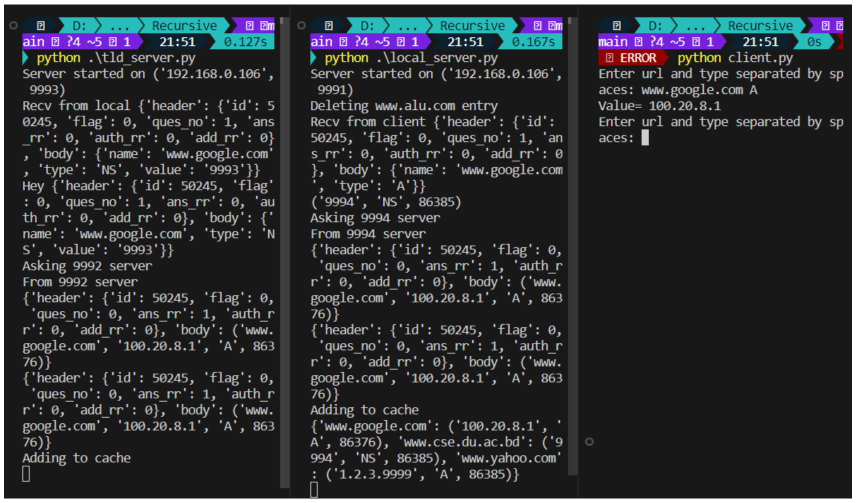
```
python .\local_server.py
Server started on ('192.168.0.106',
 9991)
Recv from client {'header': {'id':
1861, 'flag': 0, 'ques_no': 1, 'ans
_rr': 0, 'auth_rr': 0, 'add_rr': 0}
, 'body': {'name': 'www.alu.com', '
type': 'A'}}
('100.20.8.1', 'A', 4)
Deleting www.alu.com entry
Recv from client {'header': {'id':
51303, 'flag': 0, 'ques_no': 1, 'an
s_rr': 0, 'auth_rr': 0, 'add_rr': 0
}, 'body': {'name': 'www.alu.com',
'type': 'A'}}
Asking 9994 server
From 9994 server
{'header': {'id': 51303, 'flag': 0,
 'ques_no': 0, 'ans_rr': 1, 'auth_r
r': 0, 'add_rr': 0}, 'body': ('www.
alu.com', None, None, None)}
```

```
python client.py
Enter url and type separated by sp
aces: www.alu.com A
Value= 100.20.8.1
Enter url and type separated by sp
aces: www.alu.com A
Value not available
Enter url and type separated by sp
aces:
```

Figure 20: Deleting resource record based on TTL value'

4.4.2 DNS caching in local and TLD servers

Keeping domain name-to-IP address mappings that have already been resolved in a DNS server's memory is known as DNS caching. This enables the server to reply to subsequent queries for the same domain name promptly. When a DNS server receives a query for a domain name, it first checks its cache to see if it has the IP address linked to the domain name. If such a record is already in the cache, the DNS server provides the requester with the IP address immediately, saving them the trouble of going through the full DNS resolution process. Since DNS caching can drastically cut down on the time and network resources needed to respond to DNS queries, it offers several advantages, especially for domain names that are often accessed. By caching DNS records, a DNS server can respond to queries for commonly visited domain names more rapidly than by contacting other DNS servers. This leads to decreased network traffic and quicker response times.



```
D: ... Recursive
ain ?4 ~5 1   21:51   0.127s
python .\tld_server.py
Server started on ('192.168.0.106',
 9993)
Recv from local {'header': {'id': 5
0245, 'flag': 0, 'ques_no': 1, 'ans
_rr': 0, 'auth_rr': 0, 'add_rr': 0}
, 'body': {'name': 'www.google.com'
, 'type': 'NS', 'value': '9993'}}
Hey {'header': {'id': 50245, 'flag'
: 0, 'ques_no': 1, 'ans_rr': 0, 'au
th_rr': 0, 'add_rr': 0}, 'body': {'
name': 'www.google.com', 'type': 'N
S', 'value': '9993'}}
Asking 9992 server
From 9992 server
{'header': {'id': 50245, 'flag': 0,
 'ques_no': 0, 'ans_rr': 1, 'auth_r
r': 0, 'add_rr': 0}, 'body': ('www.
google.com', '100.20.8.1', 'A', 863
76)}
{'header': {'id': 50245, 'flag': 0,
 'ques_no': 0, 'ans_rr': 1, 'auth_r
r': 0, 'add_rr': 0}, 'body': ('www.
google.com', '100.20.8.1', 'A', 863
76)}
Adding to cache
```

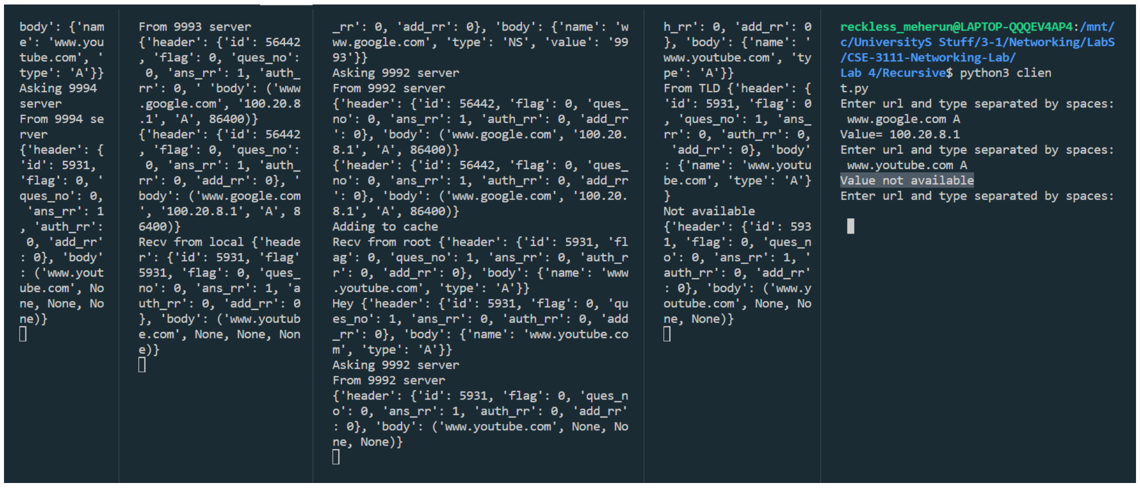
```
D: ... Recursive
ain ?4 ~5 1   21:51   0.167s
python .\local_server.py
Server started on ('192.168.0.106',
 9991)
Deleting www.alu.com entry
Recv from client {'header': {'id':
50245, 'flag': 0, 'ques_no': 1, 'an
s_rr': 0, 'auth_rr': 0, 'add_rr': 0
}, 'body': {'name': 'www.google.com
', 'type': 'A'}}
('9994', 'NS', 86385)
Asking 9994 server
From 9994 server
{'header': {'id': 50245, 'flag': 0,
 'ques_no': 0, 'ans_rr': 1, 'auth_r
r': 0, 'add_rr': 0}, 'body': ('www.
google.com', '100.20.8.1', 'A', 863
76)}
{'header': {'id': 50245, 'flag': 0,
 'ques_no': 0, 'ans_rr': 1, 'auth_r
r': 0, 'add_rr': 0}, 'body': ('www.
google.com', '100.20.8.1', 'A', 863
76)}
Adding to cache
{'www.google.com': ('100.20.8.1', '
A', 86376), 'www.cse.du.ac.bd': ('9
994', 'NS', 86385), 'www.yahoo.com'
: ('1.2.3.9999', 'A', 86385)}
```

```
D: ... Recursive
main ?4 ~5 1   21:51   0s
ERROR  python client.py
Enter url and type separated by sp
aces: www.google.com A
Value= 100.20.8.1
Enter url and type separated by sp
aces:
```

Figure 21: After caching in TLD server, the address newly added addresses are shown in the local server

4.4.3 Test failure of a DNS server process

If the client requests the IP address of a hostname that is not available to any server, the client gets a "Value not available" message. This is considered the 'Test Failure' of a DNS server process.



```
body': {'nam
e': 'www.you
tube.com', '
type': 'A'}}
Asking 9994
server
From 9994 se
rver
{'header': {
'id': 5931,
'flag': 0, '
ques_no': 0,
 'ans_rr': 1
, 'auth_rr':
 0, 'add_rr'
: 0}, 'body'
: ('www.yout
ube.com', No
ne, None, No
ne)}
```

```
From 9993 server
{'header': {'id': 56442
, 'flag': 0, 'ques_no':
 0, 'ans_rr': 1, 'auth_
rr': 0, ' 'body': ('www
.google.com', '100.20.8
.1', 'A', 86400)}
{'header': {'id': 56442
, 'flag': 0, 'ques_no':
 0, 'ans_rr': 1, 'auth_
rr': 0, 'add_rr': 0}, '
body': ('www.google.com
', '100.20.8.1', 'A', 8
6400)}
Recv from local {'heade
r': {'id': 5931, 'flag'
5931, 'flag': 0, 'ques_
no': 0, 'ans_rr': 1, 'a
uth_rr': 0, 'add_rr': 0
}, 'body': ('www.youtub
e.com', None, None, Non
e)}
```

```
_rr': 0, 'add_rr': 0}, 'body': {'name': 'w
ww.google.com', 'type': 'NS', 'value': '99
93'}}
Asking 9992 server
From 9992 server
{'header': {'id': 56442, 'flag': 0, 'ques_
no': 0, 'ans_rr': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.com', '100.20.
8.1', 'A', 86400)}
{'header': {'id': 56442, 'flag': 0, 'ques_
no': 0, 'ans_rr': 1, 'auth_rr': 0, 'add_rr
': 0}, 'body': ('www.google.com', '100.20.
8.1', 'A', 86400)}
Adding to cache
Recv from root {'header': {'id': 5931, 'fl
ag': 0, 'ques_no': 1, 'ans_rr': 0, 'auth_r
r': 0, 'add_rr': 0}, 'body': {'name': 'www
.youtube.com', 'type': 'A'}}
Hey {'header': {'id': 5931, 'flag': 0, 'qu
es_no': 1, 'ans_rr': 0, 'auth_rr': 0, 'add
_rr': 0}, 'body': {'name': 'www.youtube.co
m', 'type': 'A'}}
Asking 9992 server
From 9992 server
{'header': {'id': 5931, 'flag': 0, 'ques_n
o': 0, 'ans_rr': 1, 'auth_rr': 0, 'add_rr'
: 0}, 'body': ('www.youtube.com', None, No
ne, None)}
```

```
h_rr': 0, 'add_rr': 0
}, 'body': {'name': '
www.youtube.com', 'ty
pe': 'A'}}
From TLD {'header': {
'id': 5931, 'flag': 0
, 'ques_no': 1, 'ans_
rr': 0, 'auth_rr': 0,
 'add_rr': 0}, 'body'
: {'name': 'www.youtu
be.com', 'type': 'A'}
}
Not available
{'header': {'id': 593
1, 'flag': 0, 'ques_n
o': 0, 'ans_rr': 1, '
auth_rr': 0, 'add_rr'
: 0}, 'body': ('www.y
outube.com', None, No
ne, None)}
```

```
reckless_meherun@LAPTOP-QQQEV4AP4:/mnt/
c/UniversityS Stuff/3-1/Networking/LabS
/CSE-3111-Networking-Lab/
Lab 4/Recursive$ python3 clien
t.py
Enter url and type separated by spaces:
 www.google.com A
Value= 100.20.8.1
Enter url and type separated by spaces:
 www.youtube.com A
Value not available
Enter url and type separated by spaces:
```

Figure 22: Test failed for 'www.youtube.com'

References

[1] Internet Systems Consortium. General dns reference information.

[2] Constellix. Dns record types. dec 19 2019.

[3] Jim Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Pearson; 8th edition (2020), 2020.

[4] Amazon Web Services. What is dns?