University of Dhaka

CSE:3111 Computer Networking Lab

Lab Report 4

Distributed Database Management, Implementation of Iterative, and Recursive Queries of DNS Records.

Shariful Islam Rayhan
Roll: 41
Md Sakib Ur Rahman
Roll: 37

Submitted to:
Dr. Md. Abdur Razzaque
Dr. Muhammad Ibrahim
Dr. Md. Redwan Ahmed Rizvee
Dr. Md. Mamun Or Rashid

Submitted on: 2024-02-15

1 Introduction

In the vast landscape of the internet, every website or webpage possesses a unique identity address, known as an IP address. However, these numerical identities are cumbersome to remember. Instead, we rely on Uniform Resource Locators (URLs), which are more human-friendly and easier to recall. Behind the scenes, the Domain Name System (DNS) serves as the backbone, seamlessly translating these user-friendly URLs into corresponding IP addresses. This pivotal role of the DNS system enables our browsers to effortlessly connect to websites and webpages, enhancing the accessibility and usability of the internet.

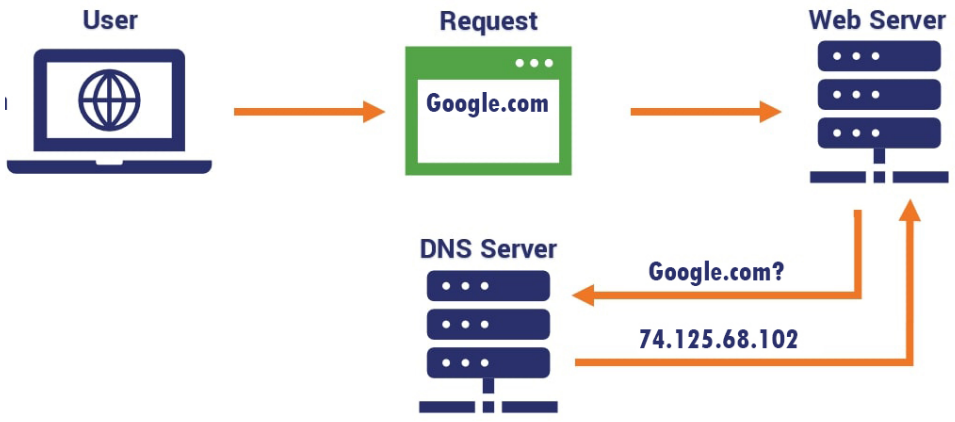


Figure 1: DNS system to convert a url to ip

1.1 Objectives

- Emulate the Domain Name Service (DNS) protocol within a controlled laboratory environment.
- Investigate the differences between iterative and recursive DNS resolution methods. clients to request the IP address of desired domains.
- Utilize the nameserver hierarchy to facilitate DNS resolution processes.
- Verify the validity of domain names and return corresponding IP addresses to clients.
- Gain practical insights into DNS operations, including resolution and translation mechanisms.
- Enhance understanding of distributed database management principles in networked environments through DNS emulation.

2 Theory

In the journey of accessing a domain such as google.com, our computer undergoes a meticulously orchestrated process to translate the user-friendly web address into a machine-readable IP address. Initially, our computer consults its local DNS cache, seeking a swift resolution of the hostname. Should the necessary information be absent from the cache, a DNS query is initiated, reaching out to our Internet Service Provider's (ISP) DNS servers for assistance. In the event of a cache miss at the ISP level, the query progresses to the root nameservers, which serve as the foundational layer of the DNS hierarchy. These nameservers meticulously direct the query to the appropriate Top-Level Domain (TLD) nameservers, such as .com, .org, or .bd, based on the domain extension. Subsequently, the TLD nameservers, acting as authoritative gatekeepers, route the query to the specific authoritative DNS servers responsible for the queried domain. These authoritative servers harbor comprehensive information about the domain, stored within DNS records. Upon retrieval of the requisite record from the authoritative nameservers, the ISP's DNS server caches this information locally, facilitating expedited resolutions for subsequent requests. Ultimately, armed with the IP address gleaned from the DNS resolution process, our computer establishes a connection with the webserver, enabling the retrieval of the desired website content through the browser.

3 Methodology

3.1 Task 1:Setting up the DNS server

In this step,we just make the authorivite name server that basically provide the ip for first time if there use cache and a client.

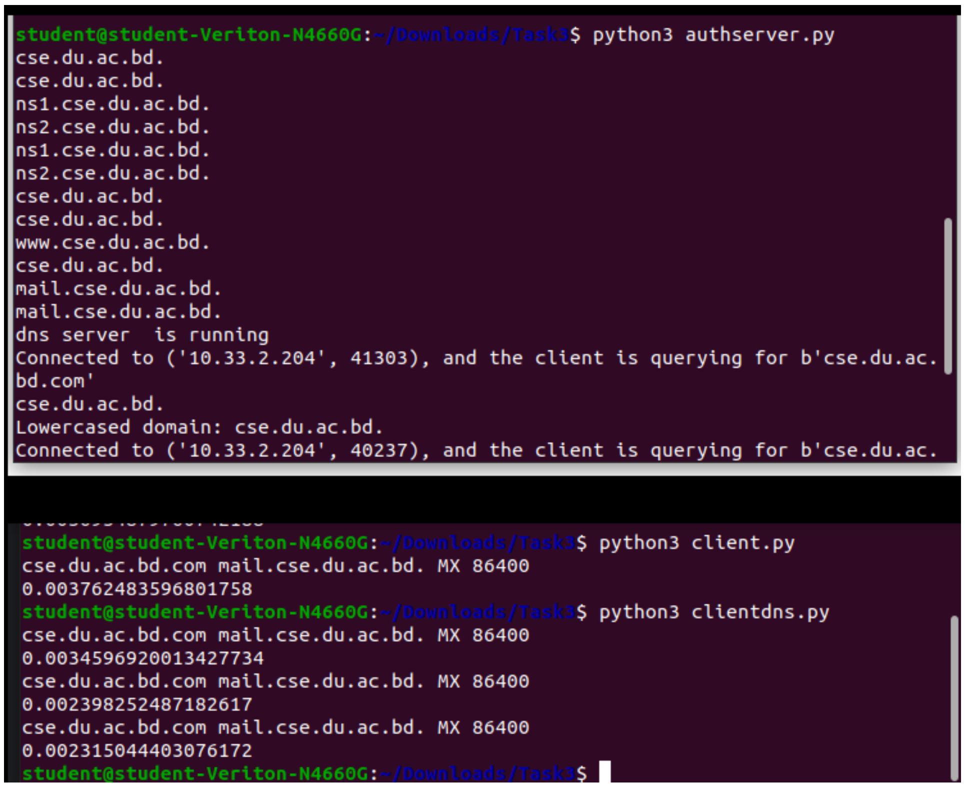
3.1.1 Server

```python
import socket
import threading
import os
import struct


def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to lowercase
            print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    flag="Auth server is working"
    server.sendto(flag.encode(),c_addr)
    print(f'Connected to {c_addr}, and the client is querying for {msg}')

    domain_name = msg.decode().lower()
    my_string =domain_name
    modified_string = my_string[:-3]
    print(modified_string)
    domain_name=modified_string


    # Decode the message from bytes to string
    print("Lowercased domain:", domain_name)
    if domain_name in dns_record:
        records = dns_record[domain_name][-1]  # Get the last record for the domain
        # Construct the response by joining the last record's components

        result = domain_name+"com"+" " + " ".join(map(str, records))
    else:
        result = "Not found. You have to register this domain name."

    server.sendto(result.encode(), c_addr)  # Send the response to the client


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record = load_file("dns_records.txt")
    server.bind(('10.33.2.203', 9997))

    print("dns server  is running")

    while True:
        msg, c_addr = server.recvfrom(1024)
        th = threading.Thread(target=handle_dns_query_iterative, args=(msg, c_addr, server))
        th.start()


if __name__ == '__main__':
    main()
```

3.1.2 Client

```python
import socket
import  time


def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr = (("10.33.2.204", 9993))

    for i in range(0,3):
        t1 = time.time()
        domain = "cse.du.ac.bd.com"
        client.sendto(domain.encode(), server_addr)

        msg, _ = client.recvfrom(1024)
        print(msg.decode())

        msg, _ = client.recvfrom(1024)
        print(msg.decode())
        t2 = time.time()
        print(t2 - t1)


if __name__ == "__main__":
    main()
```

3.1.3 Experimental Result for task 1



```
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 authserver.py
cse.du.ac.bd.
cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
cse.du.ac.bd.
cse.du.ac.bd.
www.cse.du.ac.bd.
cse.du.ac.bd.
mail.cse.du.ac.bd.
mail.cse.du.ac.bd.
dns server  is running
Connected to ('10.33.2.204', 41303), and the client is querying for b'cse.du.ac.
bd.com'
cse.du.ac.bd.
Lowercased domain: cse.du.ac.bd.
Connected to ('10.33.2.204', 40237), and the client is querying for b'cse.du.ac.
```

```
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 client.py
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.003762483596801758
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 clientdns.py
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.0034596920013427734
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.002398252487182617
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.002315044403076172
student@student-Veriton-N4660G:~/Downloads/Task3$
```

Figure 2: Authoritive Dns and client is running

3.2 Task 2: Iterative DNS resolution

To implement this iterative approach,we make 5 files.They are client ,local dns server,root server,tld server and authorive server.Every time we use the same authoritive server and client server.so, we don't mention it in future.

3.2.1 Local DNS Server

```python
import socket
import threading
import os
import struct

def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to lowercase
           # print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):




    print(f'Connected to {c_addr}, and the client is querying for {msg}')

    print("locan dns cannot find the ip..so it start the request for root")


    #connected with root
   # kisu=input("press enter for access root")
    client=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr=(("10.33.2.203",9195))

    client.sendto(msg,server_addr)

    # flag,_=client.recvfrom(1024)
    # print(flag.decode())


    msg1,_=client.recvfrom(1024)

    address=msg1.decode()
    address=address.split()
    ip=address[0]
    port=int(address[1])
    #connected to tld

    client_tld=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr_tld=((ip,port))

    client_tld.sendto(msg,server_addr_tld)

    msg2,_=client_tld.recvfrom(1024)

    #connect to auth
    msg2=msg2.decode()

    print(f"ip of auth {msg2}")

    msg2=msg2.split()
    ip1=msg2[0]

    port1=int(msg2[1])

    client_auth=socket.socket(socket.AF_INET, socket.SOCK_DGRAM
)
    server_addr_auth=((ip1,port1))

    client_auth.sendto(msg,server_addr_auth)

    msg3,_=client_auth.recvfrom(1024)







    server.sendto(msg3, c_addr)








    # domain_name = msg.decode().lower()  # Decode the message
from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last
 record for the domain
    # # Construct the response by joining the last record's
components

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain
 name."
```

```python

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.2.204',9993))

    print("local_dns server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)

       th=threading.Thread(target=handle_dns_query_iterative,
    args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.2.2 Root DNS Server

```python
import socket
import threading
import os
import struct
def handlelocaldns(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying
    for {msg}')
    msgs=msg.decode()
    dom=msgs.split('.')
    kisu=dom[-1]
    print(kisu)
    #aaa=input("please enter to open tld server")
    #client=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    address="10.33.2.204 9996"
    server.sendto(address.encode(),c_addr)



    # msg,_=client.recvfrom(1024)
    # print(msg.decode())
    #
    # msg,_=client.recvfrom(1024)
    # server.sendto(msg,c_addr)
def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    server.bind(('10.33.2.203',9195))

    print("root server  is running")

    while True:
        msg,c_addr =server.recvfrom(1024)
        flag="Root server working"
        #server.sendto(flag.encode(),c_addr)
        handlelocaldns(msg,c_addr,server)


if __name__=='__main__':
    main()
```

3.2.3 TLD Name Server for .com

```python
import socket
import threading
import os
import struct

```

```
def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to lowercase
           # print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying for {msg}')



    #print("locan dns cannot find the ip..so it start the request for root")

    #kisu=input("press enter for access autorivite")

    add="10.33.2.203 9997"
    server.sendto(add.encode(),c_addr)














    # domain_name = msg.decode().lower()  # Decode the message from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last record for the domain
    # # Construct the response by joining the last record's components
```

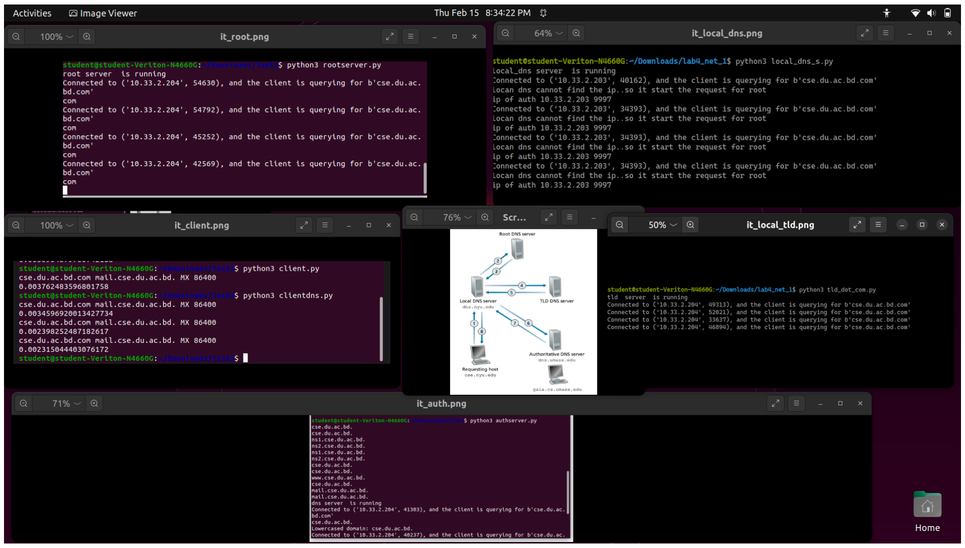
```python

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain name."

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.2.204',9996))

    print("tld  server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)
       th=threading.Thread(target=handle_dns_query_iterative, args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.2.4 TLD Name Server for .org

```python
import socket
import threading
import os
import struct

def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to lowercase
           # print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying for {msg}')

    #print("locan dns cannot find the ip..so it start the request for root")

    kisu=input("press enter for access autorivite")
    client=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr=(("10.33.3.38",9997))

    client.sendto(msg,server_addr)


    msg,_=client.recvfrom(1024)



    server.sendto(msg, c_addr)








    # domain_name = msg.decode().lower()  # Decode the message from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last record for the domain
    # # Construct the response by joining the last record's components

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain name."

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.3.20',9999))

    print("tld  server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)
       th=threading.Thread(target=handle_dns_query_iterative,args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.2.5 Authoritive Name Server is same like before

3.2.6 Experimental Result for task 2



```
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 rootserver.py
root server  is running
Connected to ('10.33.2.204', 54630), and the client is querying for b'cse.du.ac.
bd.com'
com
Connected to ('10.33.2.204', 54792), and the client is querying for b'cse.du.ac.
bd.com'
com
Connected to ('10.33.2.204', 45252), and the client is querying for b'cse.du.ac.
bd.com'
com
Connected to ('10.33.2.204', 42569), and the client is querying for b'cse.du.ac.
bd.com'
com
```

```
student@student-Veriton-N4660G:~/Downloads/lab4_net_1$ python3 local_dns_s.py
local_dns server  is running
Connected to ('10.33.2.203', 40162), and the client is querying for b'cse.du.ac.bd.com'
locan dns cannot find the ip..so it start the request for root
ip of auth 10.33.2.203 9997
Connected to ('10.33.2.203', 34393), and the client is querying for b'cse.du.ac.bd.com'
locan dns cannot find the ip..so it start the request for root
ip of auth 10.33.2.203 9997
Connected to ('10.33.2.203', 34393), and the client is querying for b'cse.du.ac.bd.com'
locan dns cannot find the ip..so it start the request for root
ip of auth 10.33.2.203 9997
Connected to ('10.33.2.203', 34393), and the client is querying for b'cse.du.ac.bd.com'
locan dns cannot find the ip..so it start the request for root
ip of auth 10.33.2.203 9997
```

```
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 client.py
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.003762483596801758
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 clientdns.py
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.003459692001342773 4
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.002398252487182617
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.002315044403076172
student@student-Veriton-N4660G:~/Downloads/Task3$
```

```
student@student-Veriton-N4660G:~/Downloads/lab4_net_1$ python3 tld_dot_com.py
tld  server  is running
Connected to ('10.33.2.204', 49313), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.204', 52021), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.204', 33637), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.204', 46894), and the client is querying for b'cse.du.ac.bd.com'
```

```
student@student-Veriton-N4660G:~/Downloads/Task3$ python3 authserver.py
cse.du.ac.bd.
cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
cse.du.ac.bd.
cse.du.ac.bd.
www.cse.du.ac.bd.
cse.du.ac.bd.
mail.cse.du.ac.bd.
mail.cse.du.ac.bd.
dns server  is running
Connected to ('10.33.2.204', 41303), and the client is querying for b'cse.du.ac.
bd.com'
cse.du.ac.bd.
Lowercased domain: cse.du.ac.bd.
Connected to ('10.33.2.204', 40237), and the client is querying for b'cse.du.ac.
```

Figure 3: Iterative Dns resolution

3.3 Task 3: Recursive DNS resolution

To implement this approach,we make 5 files.They are client ,local dns server,root server,tld server and authorive server.Every time we use the same authoritive server and client server.so, we don't mention it in future.

3.3.1 Local DNS Server

```python
import socket
import threading
import os
import struct

def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to
   lowercase
```

```python
            # print(name)
             if name not in dns_record:
                 dns_record[name] = []  # Initialize list if domain name doesn't exist
             dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):


    flag="true..local dns process to find your data from root"
    server.sendto(flag.encode(), c_addr)

    print(f'Connected to {c_addr}, and the client is querying for {msg}')

    print("locan dns cannot find the ip..so it start the request for root")

   # kisu=input("press enter for access root")
    client=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr=(("10.33.2.203",9195))

    client.sendto(msg,server_addr)

    flag,_=client.recvfrom(1024)
    print(flag.decode())


    msg,_=client.recvfrom(1024)



    server.sendto(msg, c_addr)









    # domain_name = msg.decode().lower()  # Decode the message from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last record for the domain
    # # Construct the response by joining the last record's
    components

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain
     name."

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.2.204',9993))

    print("local_dns server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)

       th=threading.Thread(target=handle_dns_query_iterative,
    args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.3.2 Root DNS Server

```python
import socket
import threading
import os
import struct
def handlelocaldns(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying for {msg}')
    msgs=msg.decode()
    dom=msgs.split('.')
    kisu=dom[-1]
    print(kisu)
    #aaa=input("please enter to open tld server")
    client=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    server_addr=(("10.33.2.204",9996))

    client.sendto(msg,server_addr)
    msg,_=client.recvfrom(1024)
    print(msg.decode())

    msg,_=client.recvfrom(1024)
    server.sendto(msg,c_addr)
def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    server.bind(('10.33.2.203',9195))

    print("root server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)
       flag="Root server working"
       server.sendto(flag.encode(),c_addr)
       handlelocaldns(msg,c_addr,server)


if __name__=='__main__':
    main()
```

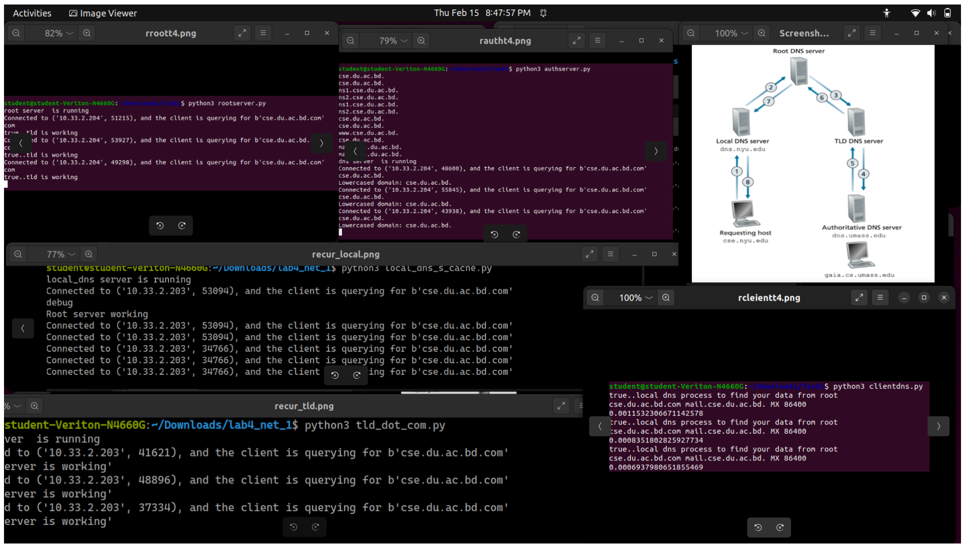
3.3.3 TLD Name Server for .com

```python
import socket
import threading
import os
import struct

def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to
lowercase
          # print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if
domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying
for {msg}')


    flag="true..tld is working"
    server.sendto(flag.encode(), c_addr)
    #print("locan dns cannot find the ip..so it start the
request for root")

    #kisu=input("press enter for access autorivite")
    client=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr=(("10.33.2.203",9997))

    client.sendto(msg,server_addr)
    msg,_=client.recvfrom(1024)
    print(msg)

    msg,_=client.recvfrom(1024)



    server.sendto(msg, c_addr)









    # domain_name = msg.decode().lower()  # Decode the message
from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last
```

```
     record for the domain
    # # Construct the response by joining the last record's
    components

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain
     name."

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.2.204',9996))

    print("tld  server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)
       th=threading.Thread(target=handle_dns_query_iterative,
    args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.3.4 Authoritive Name Server for .Same like before

3.3.5 Experimental Result for task 3



```
student@student-Veriton-N4660G:~/Downloads/Task1$ python3 rootserver.py
root server  is running
Connected to ('10.33.2.204', 51215), and the client is querying for b'cse.du.ac.bd.com'
com
true..tld is working
Connected to ('10.33.2.204', 53927), and the client is querying for b'cse.du.ac.bd.com'
com
true..tld is working
Connected to ('10.33.2.204', 49298), and the client is querying for b'cse.du.ac.bd.com'
com
true..tld is working
```

```
student@student-Veriton-N4660G:~/Downloads/Task1$ python3 authserver.py
cse.du.ac.bd.
cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
ns1.cse.du.ac.bd.
ns2.cse.du.ac.bd.
cse.du.ac.bd.
cse.du.ac.bd.
www.cse.du.ac.bd.
cse.du.ac.bd.
mail.cse.du.ac.bd.
mail.cse.du.ac.bd.
dns server  is running
Connected to ('10.33.2.204', 48600), and the client is querying for b'cse.du.ac.bd.com'
cse.du.ac.bd.
Lowercased domain: cse.du.ac.bd.
Connected to ('10.33.2.204', 55845), and the client is querying for b'cse.du.ac.bd.com'
cse.du.ac.bd.
Lowercased domain: cse.du.ac.bd.
Connected to ('10.33.2.204', 43938), and the client is querying for b'cse.du.ac.bd.com'
cse.du.ac.bd.
Lowercased domain: cse.du.ac.bd.
```

```
student@student-Veriton-N4660G:~/Downloads/lab4_net_1$ python3 local_dns_s_cache.py
local_dns server is running
Connected to ('10.33.2.203', 53094), and the client is querying for b'cse.du.ac.bd.com'
debug
Root server working
Connected to ('10.33.2.203', 53094), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.203', 53094), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.203', 34766), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.203', 34766), and the client is querying for b'cse.du.ac.bd.com'
Connected to ('10.33.2.203', 34766), and the client is querying for b'cse.du.ac.bd.com'
```

```
student-Veriton-N4660G:~/Downloads/lab4_net_1$ python3 tld_dot_com.py
ver  is running
d to ('10.33.2.203', 41621), and the client is querying for b'cse.du.ac.bd.com'
erver is working'
d to ('10.33.2.203', 48896), and the client is querying for b'cse.du.ac.bd.com'
erver is working'
d to ('10.33.2.203', 37334), and the client is querying for b'cse.du.ac.bd.com'
erver is working'
```

```
student@student-Veriton-N4660G:~/Downloads/Task1$ python3 clientdns.py
true..local dns process to find your data from root
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.0011532306671142578
true..local dns process to find your data from root
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.0008351802825927734
true..local dns process to find your data from root
cse.du.ac.bd.com mail.cse.du.ac.bd. MX 86400
0.0006937980651855469
```

Figure 4: Recursive Dns resolution

3.4 Task 4: Extending the System

Here ,we just modify the authorivite server so that it can delete the same domain name that is big than its min ttl value.And we update the local and TLD server,actually we update the local server for dns caching.

3.4.1 Authoritive DNS Server

```python
import socket
import threading
import os
import struct


def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to lowercase
            print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    flag = "Auth server is working"
    server.sendto(flag.encode(), c_addr)
    print(f'Connected to {c_addr}, and the client is querying for {msg}')

    domain_name = msg.decode().lower()
    modified_domain = domain_name[:-3]
    print(modified_domain)

    # Decode the message from bytes to string
    print("Lowercased domain:", modified_domain)

    if modified_domain in dns_record:
        # Fetch all records for the domain
        records = dns_record[modified_domain]

        # Find the record with the minimum TTL
        min_ttl = float('inf')  # Initialize with infinity
        min_ttl_record = None

        for record in records:
            if record[2] < min_ttl:
                min_ttl = record[2]
                min_ttl_record = record

        # Construct the response by joining the components of the record with minimum TTL
        result = modified_domain + ".com" + " " + " ".join(map(str, min_ttl_record))

        # Update text file by removing records with TTL greater than min_ttl
        dns_record[modified_domain] = [min_ttl_record]  # Keep only the record with minimum TTL
        with open("ndns_records.txt", "r+") as file:
            lines = file.readlines()
            file.seek(0)
            for line in lines:
                parts = line.strip().split()
                if parts[0].lower() == modified_domain and int(parts[-1]) > min_ttl:
                    continue  # Skip writing this line if TTL is greater than min_ttl
                file.write(line)
            file.truncate()  # Remove any extra lines at the end of the file
    else:
        result = "Not found. You have to register this domain name."

    server.sendto(result.encode(), c_addr)  # Send the response to the client


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record = load_file("ndns_records.txt")
    server.bind(('10.33.2.203', 9997))

    print("dns server  is running")

    while True:
        msg, c_addr = server.recvfrom(1024)
        th = threading.Thread(target=handle_dns_query_iterative, args=(msg, c_addr, server))
        th.start()


if __name__ == '__main__':
    main()
```

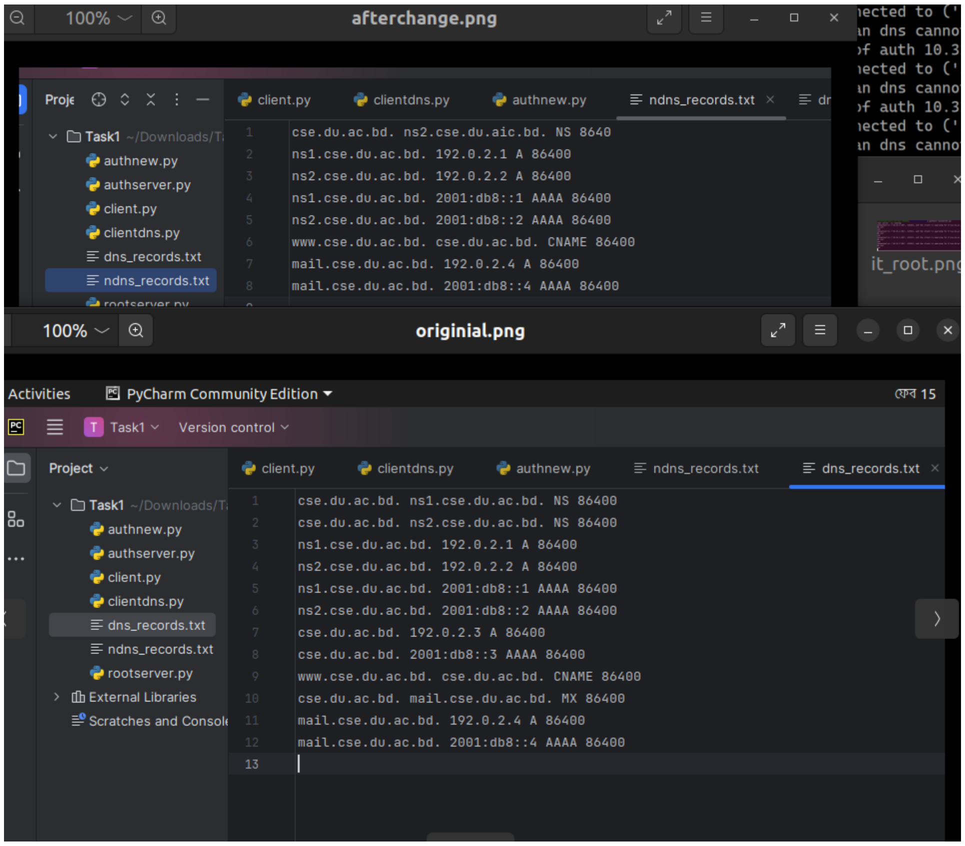
3.4.2 Local DNS Cache Server

```python
import socket
import threading
import os
import struct

cache = []

def handle_dns_query_iterative(msg, c_addr, server):
    flag = "true..local dns process to find your data from root
    "
    server.sendto(flag.encode(), c_addr)
    print(f'Connected to {c_addr}, and the client is querying
    for {msg}')
    #print("local dns cannot find the ip, so it starts the
    request for root")
    debug="debug"
    for item in cache:
        parts = item.split()  # Splitting the string by space
        debug=parts[0]
        if parts[0] == msg.decode():
            server.sendto(item.encode(), c_addr)

            return
    print(debug)
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr = ("10.33.2.203", 9195)

    client.sendto(msg, server_addr)

    flag, _ = client.recvfrom(1024)
    print(flag.decode())

    msg, _ = client.recvfrom(1024)
    server.sendto(msg, c_addr)

    cache.append(msg.decode())

def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    global dns_record

    server.bind(('10.33.2.204', 9993))
    print("local_dns server is running")

    while True:
        msg, c_addr = server.recvfrom(1024)
        th = threading.Thread(target=handle_dns_query_iterative
    , args=(msg, c_addr, server))
        th.start()

if __name__ == '__main__':
    main()
```

3.4.3 TLD Name Cache Server for .com

```
import socket
import threading
import os
import struct

def load_file(filename):
    dns_record = {}
    with open(filename, "r") as file:
        for line in file:
            name, value, r_type, ttl = line.strip().split()
            name = name.lower()  # Convert domain name to
    lowercase
           # print(name)
            if name not in dns_record:
                dns_record[name] = []  # Initialize list if
    domain name doesn't exist
            dns_record[name].append((value, r_type, int(ttl)))
    return dns_record


def handle_dns_query_iterative(msg, c_addr, server):
    print(f'Connected to {c_addr}, and the client is querying
    for {msg}')


    flag="true..tld is working"
    server.sendto(flag.encode(), c_addr)
    #print("locan dns cannot find the ip..so it start the
    request for root")

    #kisu=input("press enter for access autorivite")
    client=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_addr=(("10.33.2.203",9997))

    client.sendto(msg,server_addr)
    msg,_=client.recvfrom(1024)
    print(msg)

    msg,_=client.recvfrom(1024)



    server.sendto(msg, c_addr)









    # domain_name = msg.decode().lower()  # Decode the message
    from bytes to string
    # print("Lowercased domain:", domain_name)
    # if domain_name in dns_record:
    #     records = dns_record[domain_name][-1]  # Get the last
     record for the domain
    # # Construct the response by joining the last record's
    components

    #     result = domain_name+" ".join(map(str, records))
    # else:
    #     result = "Not found. You have to register this domain
     name."

      # Send the response to the client


def main():
    server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    global dns_record
    dns_record=load_file("dns_records.txt")
    server.bind(('10.33.2.204',9996))

    print("tld  server  is running")

    while True:
       msg,c_addr =server.recvfrom(1024)
       th=threading.Thread(target=handle_dns_query_iterative,
    args=(msg,c_addr,server))
       th.start()



if __name__=='__main__':
    main()
```

3.4.4 Experimental Result for task 4



```
cse.du.ac.bd. ns2.cse.du.aic.bd. NS 8640
ns1.cse.du.ac.bd. 192.0.2.1 A 86400
ns2.cse.du.ac.bd. 192.0.2.2 A 86400
ns1.cse.du.ac.bd. 2001:db8::1 AAAA 86400
ns2.cse.du.ac.bd. 2001:db8::2 AAAA 86400
www.cse.du.ac.bd. cse.du.ac.bd. CNAME 86400
mail.cse.du.ac.bd. 192.0.2.4 A 86400
mail.cse.du.ac.bd. 2001:db8::4 AAAA 86400
```

```
cse.du.ac.bd. ns1.cse.du.ac.bd. NS 86400
cse.du.ac.bd. ns2.cse.du.ac.bd. NS 86400
ns1.cse.du.ac.bd. 192.0.2.1 A 86400
ns2.cse.du.ac.bd. 192.0.2.2 A 86400
ns1.cse.du.ac.bd. 2001:db8::1 AAAA 86400
ns2.cse.du.ac.bd. 2001:db8::2 AAAA 86400
cse.du.ac.bd. 192.0.2.3 A 86400
cse.du.ac.bd. 2001:db8::3 AAAA 86400
www.cse.du.ac.bd. cse.du.ac.bd. CNAME 86400
cse.du.ac.bd. mail.cse.du.ac.bd. MX 86400
mail.cse.du.ac.bd. 192.0.2.4 A 86400
mail.cse.du.ac.bd. 2001:db8::4 AAAA 86400
```

Figure 5: Update authoritive server using ttl value

4 Experience

1. The browser first checks its local DNS cache to determine if the IP address for google.com is already stored, aiming for swift resolution.
2. n case the ISP's DNS servers do not possess the required information, the query is escalated to the root nameservers.
3. TLD nameservers, acting as gatekeepers, direct the query to the authoritative DNS servers responsible for google.com.
4. It was nice to implement server and client and make sense

References

[1] HTTP : https://constellix.com/news/dns-record-types

[2] DNS Basic: https://aws.amazon.com/route53/what-is-dns

[3] SendingGetandPost:https://www.digitalocean.com/community/tutorials/java-httpurlconnection-example-java-http-request-get-post