



University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 4:

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

Submitted By:

1. Syed Mumtahir Mahmud, Roll: 50
2. Nazira Jesmin Lina, Roll: 55

Submitted On :

February 16, 2023

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

1 Introduction

A distributed database management system (DDDBS) is a type of database management system that is spread across multiple computers, interconnected through a network. This type of database system allows multiple users to access and manage the same data, as well as providing data redundancy and increased data availability.

When it comes to implementing iterative and recursive queries of DNS records, it is important to understand how the Domain Name System (DNS) works. DNS is a hierarchical, distributed database that maps domain names to IP addresses, allowing for the resolution of human-readable domain names into IP addresses that can be used by computers to communicate with each other.

Iterative queries in DNS involve a series of successive requests to different DNS servers until the desired information is returned. This is in contrast to recursive queries, where a single DNS server is responsible for fulfilling the entire request, including making any necessary subsequent requests to other DNS servers.

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.

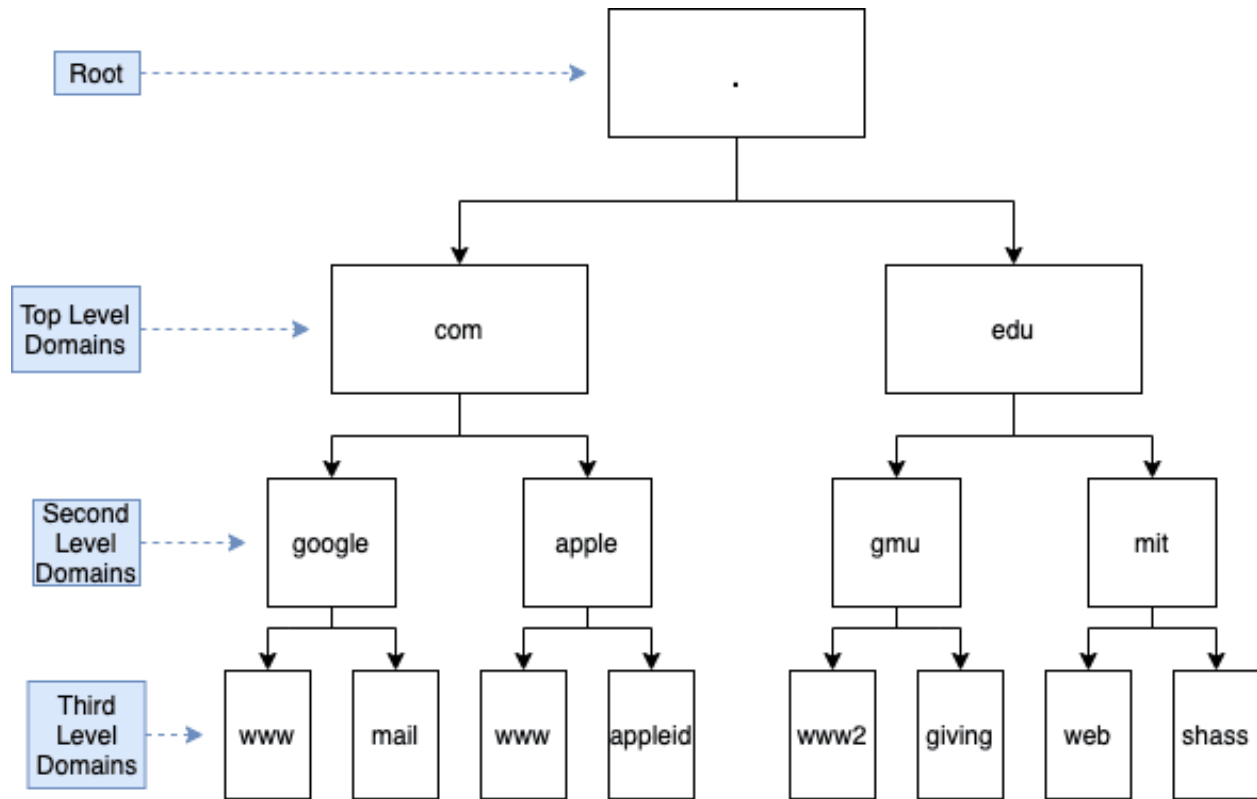
1.1 Objectives

- **Create distributed database management system:** To provide a centralized and efficient way to manage and access data that is stored on multiple computers or nodes within a network.
- **Implementing iterative and recursive queries of DNS records :** To provide a fast and reliable way to resolve domain names into IP addresses.

2 Theory

The Domain Name System (DNS) is a system for mapping domain names (e.g., example.com) to IP addresses (e.g., 192.0.2.1) and vice versa. It serves as a decentralized, hierarchical database for translating human-readable domain names into the numerical IP addresses that computers use to communicate with each other over the Internet.

A DNS query begins when a user types a domain name into a web browser or clicks on a hyperlink. The user's computer sends a request to a DNS resolver, which is usually provided by the user's Internet service provider (ISP). The resolver then forwards the request to a series of DNS servers, starting with the root DNS servers and working its way down the hierarchy until it finds the server that is authoritative for the domain name in question. The authoritative server returns the IP address corresponding to the domain name, which the resolver then passes back to the user's computer.



Graphic created by Blake Khan (blakekhan.com)

Figure 1: Hiercharchy of DNS

When we visit a domain such as google.com, our computer follows a series of steps to convert the human-readable web address into a machine-readable IP address. IP address retrieval process is as follows:

1. **Search in DNS cache:** When we ask our computer to resolve a hostname, the first place our computer looks is in its local DNS cache. If our computer doesn't already know the answer, it needs to perform a DNS query to find out.
2. **Request to ISP's DNS servers:** If the information is not stored locally, our computer queries our ISP's DNS servers.
3. **Request to root nameservers:** If those servers don't have the answer, they query the root nameservers. A nameserver is an always running computer that resolves queries about domain names, such as IP addresses.
4. **Request to Top Level Domain nameservers:** The root nameservers will look at the first part of our request, reading from right to left and direct our query to the Top-Level Domain (TLD) nameservers for .com (for google.com). Each TLD, such as .com, .org, and .bd, have their own set of nameservers, which act like a receptionist for each TLD.
5. **Request to authoritative DNS servers:** The TLD nameservers review the next part of our request and direct our query to the nameservers responsible for this specific domain. These authoritative nameservers are responsible for knowing all the information about a specific domain, which are stored in DNS records.

6. **Retrieve the record:** ISP's DNS server retrieves the record for google.com from the authoritative nameservers and stores the record in its local cache. If anyone else requests the host record for google.com, the ISP's servers will already have the answer and will not need to go through the lookup process again. All records have an expiration time. After a while, the server will need to ask for a new copy of the record to make sure the information doesn't become out-of- date.
7. **Receive the answer:** Armed with the answer, ISP's server returns the record back to our computer. Computer stores the record in its cache, reads the IP address from the record, then passes this information to browser. The browser then opens a connection to the webserver and receives the website.

3 Methodology

1. We create a thread for each DNS server.
2. We create a tree of DNS servers with a root node.
3. Each server is provided with the reference to its parent server and child servers.
4. The server awaits for request from client
5. After receiving request, server looks for the requested domain in its current location and the child (if any). if not found, it will forward the request to the parent.
6. Parent does similar search, if not found, it sends the request to its parents.
7. If the domain can't be found in any of the DNS servers, then client will be prompted with an error message.
8. If the domain is found in any of the servers, they will return to the requesting client. (iteratively/recursively)

4 Experimental Result

Some Snapshots of the Client and Server Side queries can be seen in the following figures:

4.1 Part 1: Setting up the DNS server

4.1.1 Server :

Server takes a request from the client. If client wants to know the IP address of a Domain name, it sends the Header and the IP address of the requested domain name. The Header format is similar for both types of messages. The information is held up in up to five different sections of DNS message format. The query message is having two sections- header and question records.

```
python -u "/Users/syedmumtahnimahmud/Desktop/Class/networking/NETLAB4/setting_up_dns/aserver.py"
(base) syedmumtahnimahmud@Syeds-MacBook-Pro setting_up_dns % python -u "/Users/syedmumtahnimahmu
d/Desktop/Cla
ss/networking/NETLAB4/setting_up_dns/aserver.py"
[STARTING] Server is starting
[LISTENING] Server is listening on :4487.
[RECEIVED MESSAGE] cse.du.ac.bd. AAAA from ('127.0.0.1', 64770).[ACTIVE CONNECTIONS] 1

cse.du.ac.bd.
AAAA
hi
[RECEIVED MESSAGE] www.cse.du.ac.bd. CNAME from ('127.0.0.1', 56382).[ACTIVE CONNECTIONS] 1

www.cse.du.ac.bd.
CNAME
hi
```

Figure 2: Content of server for Part 1

The response message consists of five sections:

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

DNS message format is 12 bytes.

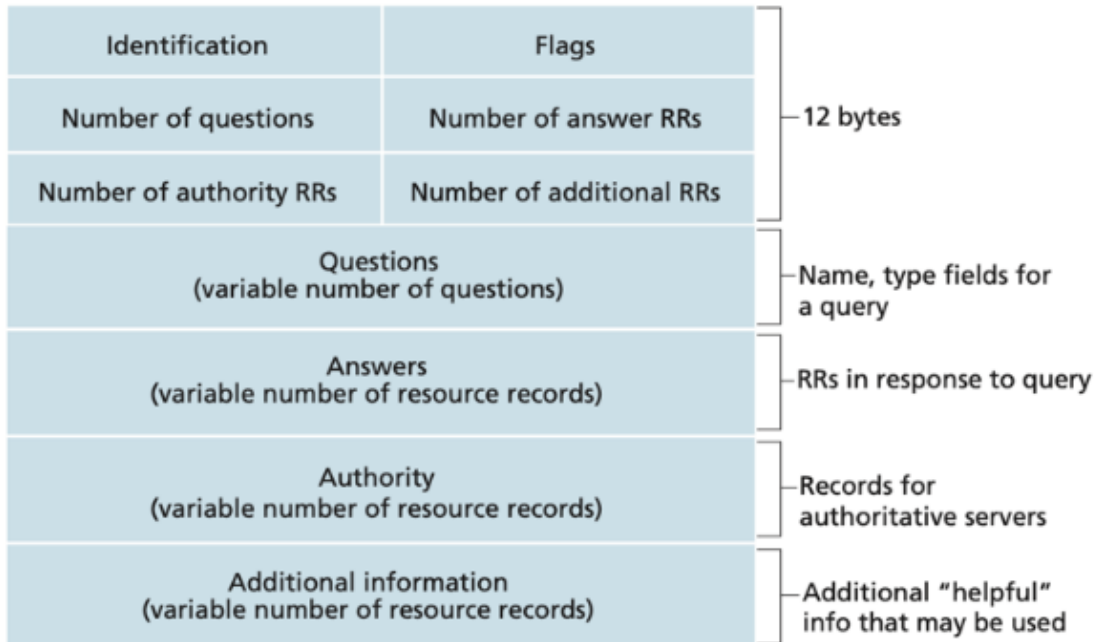


Figure 3: DNS message Format

4.1.2 Client :

The client can request for IP address for any domain name. After accepting the request by server the header and IP will show on the client's screen.

```

• (base) syedmumtahirmahmud@Syeds-MacBook-Pro setting_up_dns % python3 Client1.py
Enter a message to send to the server: cse.du.ac.bd. A
In bytes:
b'2\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00cse.du.ac.bd. 192.0.2.3 A 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'cse.du.ac.bd. 192.0.2.3 A 86400'}
• (base) syedmumtahirmahmud@Syeds-MacBook-Pro setting_up_dns % python3 Client1.py
Enter a message to send to the server: cse.du.ac.bd. AAAA
In bytes:
b'2\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00cse.du.ac.bd. 2001:db8::3 AAAA 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'cse.du.ac.bd. 2001:db8::3 AAAA 86400'}
○ (base) syedmumtahirmahmud@Syeds-MacBook-Pro setting_up_dns %

```

Figure 4: Content of Client for Task 1

4.2 Part 2: Iterative DNS resolution

Iterative DNS resolution, also known as iterative querying or iterative lookup, is a type of DNS resolution process used by DNS clients to query DNS servers for information about a domain name.

In iterative DNS resolution, the client sends a query to a DNS server and the server responds with the best answer it can provide. If the server has the information the client requested, it returns it in the response. If the server does not have the information, it returns a referral to another DNS server that may have the information.

The client then sends a query to the next DNS server, and the process continues until the requested information is found or until there are no more referrals. This process is called iterative because the client iterates through a series of queries and responses until it gets the answer it is looking for.

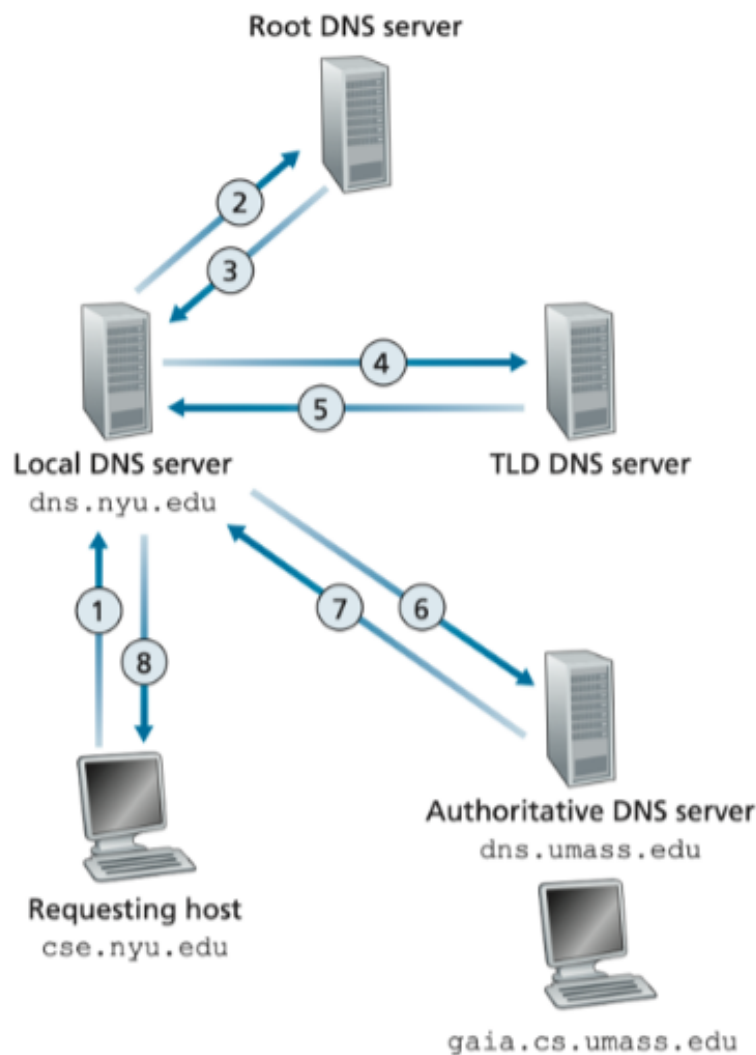


Figure 5: Iterative DNS resolution

4.2.1 Server :

An Iterative DNS Resolution Server is a type of Domain Name System (DNS) server that works by sending a query to a series of servers until the correct IP address of a domain name is found. The server starts by sending the query to a root DNS server, which provides a referral to the top-level domain (TLD) DNS server. The TLD DNS server then provides a referral to the authoritative DNS server for the domain in question, which responds with the IP address.

- **Root Server :** When a client sends a DNS query to resolve a domain name, the query first reaches the local DNS resolver. If the local resolver does not have the requested information in its cache, it sends the query to the root server.

The root server responds to the query with a referral to the top-level domain (TLD) server responsible for the TLD associated with the domain name being resolved.

```
python -u "/Users/syedmumtahirmahmud/Desktop/Class/networking/NETLAB4/Iterative/rootserver.py"
source /Users/syedmumtahirmahmud/Desktop/Class/networking/NETLAB4/Iterative/chcking/bin/activate
(base) syedmumtahirmahmud@Syeds-MacBook-Pro Iterative % python -u "/Users/syedmumtahirmahmud/Desktop/Class/networking/NETLAB4/Iterative/rootserver.py"
[STARTING] ROOT Server is starting
[LISTENING] ROOT Server is listening on :4487.
[RECEIVED MESSAGE] www.yahoo.com from ('127.0.0.1', 54944).
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.cse.du.ac.bd from ('127.0.0.1', 59492).
[ACTIVE CONNECTIONS] 0
[]
```

Figure 6: Content of Root Server for Task 2

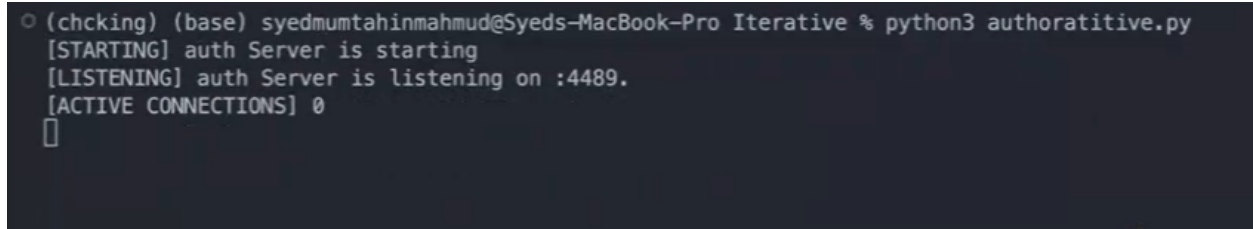
- **TLD Server:** In an iterative DNS resolution process, the top-level domain (TLD) server plays a crucial role. Its main function is to provide a referral to the authoritative name server for the domain name being resolved.

```
(chcking) (base) syedmumtahirmahmud@Syeds-MacBook-Pro Iterative % python3 tld.py
[STARTING] TLD Server is starting
[LISTENING] TLD Server is listening on :4488.
[RECEIVED MESSAGE] www.yahoo.com from ('127.0.0.1', 54944).
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.cse.du.ac.bd from ('127.0.0.1', 59492).
[ACTIVE CONNECTIONS] 0
[]
```

Figure 7: Content of TLD Server for Task 2

When a client sends a DNS query to resolve a domain name, the query first reaches the local DNS resolver. If the local resolver does not have the requested information in its cache, it sends the query to the root server. The root server then provides a referral to the TLD server responsible for the TLD associated with the domain name being resolved.

- **Authoritative Server:** The authoritative server in the iterative DNS resolution process is responsible for providing the final and most accurate answer to a DNS query. This server holds the authoritative information for a specific domain and is the source of truth for all information related to that domain.



```

○ (chcking) (base) syedmumtahnimahmud@Syeds-MacBook-Pro Iterative % python3 authoratitive.py
[STARTING] auth Server is starting
[LISTENING] auth Server is listening on :4489.
[ACTIVE CONNECTIONS] 0
□

```

Figure 8: Content of Authoritative Server for Task 2

When a client computer performs a DNS query, it may first consult its local cache or a local DNS resolver, which will then make a series of queries to other DNS servers to ultimately determine the IP address associated with a domain name. If the local resolver does not have the information stored, it will send a request to a root DNS server, which will provide a referral to a top-level domain (TLD) server. The TLD server will then refer the resolver to the authoritative server for the specific domain in question.

4.2.2 Client :

The client is responsible for sending the DNS query packet to the local DNS resolver, which acts as the first stop in the resolution process. The client is also responsible for receiving the response from the root server, which provides a referral to the appropriate TLD server, and processing this response to determine the next step in the resolution process.

4.2.3 Advantages:

- Iterative DNS resolution can be faster than recursive DNS resolution because the client can immediately start querying other servers as soon as it receives a referral.
- With iterative DNS resolution, the client has more control over the resolution process, as it can choose which server to query next based on the referral information provided by the previous server.
- Iterative DNS resolution is more scalable than recursive resolution because it distributes the load among many servers instead of relying on a single server to resolve all queries.

4.2.4 Disdvantages:

- Iterative DNS resolution can be more complex and requires more technical knowledge to use, as the client needs to know which servers to query and how to interpret the referral information.
- In some cases, iterative DNS resolution can result in higher latency, as the client may need to query multiple servers before getting a response.
- Iterative DNS resolution can also have a higher risk of errors, as there is a greater chance of a referral being incorrect or outdated, or of a server being unavailable or misconfigured.

```

● (base) syedmumtahinmahmud@Syeds-MacBook-Pro Iterative % python3 client.py
Enter an address: www.yahoo.com

Before Decoding
b'2\x00\x00\x00\x00\x01\x00\x00\x00\x00www.yahoo.com 4488 NS 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.yahoo.com 4488 NS 86400'}
www.yahoo.com 4488 NS 86400
Connecting to port 4488

Before Decoding
b'2\x00\x00\x00\x00\x01\x00\x00\x00\x00www.yahoo.com 4489 NS 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.yahoo.com 4489 NS 86400'}
www.yahoo.com 4489 NS 86400
Connecting to port 4489

Before Decoding
b'2\x00\x00\x00\x00\x01\x00\x00\x00\x00www.yahoo.com 1.2.3.9999 A 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.yahoo.com 1.2.3.9999 A 86400'}
www.yahoo.com 1.2.3.9999 A 86400
○ (base) syedmumtahinmahmud@Syeds-MacBook-Pro Iterative % 

```

Figure 9: Content of Client for Task 2

4.3 Part 3: Recursive DNS resolution

Recursive DNS resolution is a type of DNS resolution process used by DNS servers to resolve domain names into IP addresses. In recursive resolution, the DNS server fully resolves the query on behalf of the client by iterating through a series of queries and responses until it finds the answer.

Here is how recursive DNS resolution works:

The client sends a query to the DNS server with the domain name it wants to resolve. If the DNS server has the answer in its cache, it returns the response to the client. If the DNS server does not have the answer in its cache, it sends a query to another DNS server. The second DNS server checks its cache and returns the response to the first DNS server if it has the answer. If it does not have the answer, it sends a query to a third DNS server. The process repeats until the answer is found. The DNS server returns the response to the client and stores it in its cache for future requests.

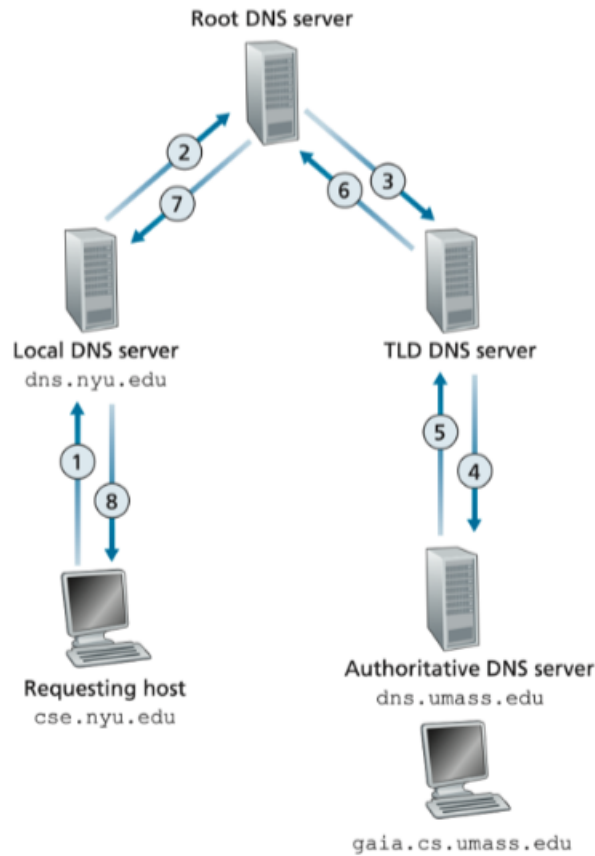


Figure 10: Recursive DNS resolution

Recursive DNS resolution is used by many DNS servers, including Internet Service Provider (ISP) DNS servers and public DNS resolvers like Google DNS and OpenDNS. It is often contrasted with iterative DNS resolution, which involves the client querying servers and receiving referrals until it gets the answer it is looking for.

4.3.1 Server :

In recursive DNS resolution, a DNS server is responsible for resolving domain names into IP addresses by querying other DNS servers. When a client requests a domain name resolution, the recursive DNS server starts the resolution process by sending a query to the root server to obtain the IP address of the top-level domain server responsible for the domain name being requested.

The recursive DNS server then queries the top-level domain (TLD) server to obtain the IP address of the authoritative name server for the domain being requested. The authoritative name server is responsible for maintaining the DNS records for the domain name being requested. The recursive DNS server sends a query to the authoritative name server to obtain the IP address associated with the domain name.

Once the recursive DNS server receives the IP address from the authoritative name server, it caches the information in its local memory to expedite future queries for the same domain name. The recursive DNS server then sends the IP address back to the client that initiated the request.

- **Root Server:** In recursive DNS resolution, the DNS server that is performing the resolution

starts by sending a query to a root server to find out which TLD server is responsible for the domain name being queried. The root server responds with a referral that contains the IP address of the appropriate TLD server, which the DNS server then queries to find the IP address of the domain name being resolved.

```
python -u "/Users/syedmumtahirmahmud/Desktop/Class/networking/NETLAB4/Recursive/rootserver.py"
(base) syedmumtahirmahmud@Syeds-MacBook-Pro Recursive % python -u "/Users/syedmumtahirmahmud/Desktop/Class/networking/NETLAB4/Recursive/rootserver.py"
[STARTING] ROOT Server is starting
[LISTENING] ROOT Server is listening on :4487.
[RECEIVED MESSAGE] www.google.com from ('127.0.0.1', 55366).
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.cse.du.ac.bd from ('127.0.0.1', 54778).
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.yahoo.com from ('127.0.0.1', 63606).
[ACTIVE CONNECTIONS] 0
█
```

Figure 11: Content of Recursive Root Server

- **TLD Server:** In recursive DNS resolution, when a DNS server receives a query for a domain name, it first sends a request to one of the 13 root servers to find out which TLD server is responsible for the domain name being queried. The root server responds with a referral that contains the IP address of the TLD server responsible for the TLD in the domain name being queried.

Once the DNS server has the IP address of the TLD server, it sends a query to that server to get the IP address of the domain name being resolved. The TLD server responds with a referral that contains the IP address of the authoritative name server for the domain being resolved. The DNS server then queries the authoritative name server to get the IP address of the domain name being resolved, and returns that IP address to the original requester.

```
(base) syedmumtahirmahmud@Syeds-MacBook-Pro Recursive % python3 tld.py
[STARTING] TLD Server is starting
[LISTENING] TLD Server is listening on :4488.
[RECEIVED MESSAGE] www.cse.du.ac.bd from ('127.0.0.1', 4487).
sending
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.yahoo.com from ('127.0.0.1', 4487).
[ACTIVE CONNECTIONS] 0
█
```

Figure 12: Content of Recursive TLD Server

- **Authoritative Server:** In recursive DNS resolution, the DNS server sends a query to the authoritative name server for the domain being resolved, requesting the IP address associated with the domain name. The authoritative name server responds with the IP address for the domain name, which the DNS server then returns to the original requester.

Authoritative name servers are essential for the DNS system to function properly, as they are responsible for maintaining the most up-to-date DNS records for domain names. Without

authoritative name servers, DNS resolution would not be possible, and users would not be able to access websites and other Internet services by name.

```
○ (base) syedmumtahirmahmud@Syeds-MacBook-Pro Recursive % python3 tld.py
[STARTING] TLD Server is starting
[LISTENING] TLD Server is listening on :4488.
[RECEIVED MESSAGE] www.cse.du.ac.bd from ('127.0.0.1', 4487).
sending
[ACTIVE CONNECTIONS] 0
[RECEIVED MESSAGE] www.yahoo.com from ('127.0.0.1', 4487).
[ACTIVE CONNECTIONS] 0
█
```

Figure 13: Content of Recursive Authoritative Server

4.3.2 Client :

In recursive DNS resolution, the client is the device or application that initiates a DNS query to resolve a domain name into an IP address. The client can be any device or application that needs to communicate with a server or service over the Internet.

When a client requests a domain name resolution, it sends a query to a recursive DNS server to resolve the domain name. The recursive DNS server then performs the DNS resolution process on behalf of the client, querying various DNS servers in a hierarchical fashion until it obtains the IP address for the requested domain name.

The client is typically unaware of the details of the DNS resolution process, as this is handled by the recursive DNS server. Once the recursive DNS server obtains the IP address for the requested domain name, it sends the information back to the client, allowing the client to connect to the intended destination.

```
● (base) syedmumtahirmahmud@Syeds-MacBook-Pro Recursive % python3 client.py
Enter an address: www.yahoo.com

Before Decoding
b'2\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00www.yahoo.com 1.2.3.9999'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.yahoo.com 1.2.3.9999'}
www.yahoo.com 1.2.3.9999
○ (base) syedmumtahirmahmud@Syeds-MacBook-Pro Recursive % █
```

Figure 14: Content of Client for Task 3

4.3.3 Advantages:

- Recursive DNS resolution is a simple process that requires little technical knowledge or input from the client.

- With recursive DNS resolution, the client only needs to send a single query to the DNS server, and the server handles the rest of the resolution process.
- Recursive DNS resolution can be more reliable than iterative resolution because the DNS server handles the entire resolution process, including error checking and retrying failed queries.

4.3.4 Disadvantages:

- Recursive DNS resolution can be slower than iterative DNS resolution because the DNS server has to handle the entire resolution process, which can involve multiple queries and responses.
- Because the DNS server handles the entire resolution process, recursive resolution can result in higher network traffic, as the DNS server may need to query multiple other servers to resolve a single domain name.
- Recursive DNS resolution can also pose a security risk if the DNS server is compromised or controlled by an attacker, as the attacker can potentially intercept or modify the DNS responses sent to the client.

4.4 Part 4: Extending the System

DNS caching is the process by which a DNS server stores previously resolved domain name-to-IP address mappings in memory, so that it can quickly respond to subsequent queries for the same domain name. When a DNS server receives a query for a domain name, it first checks its cache to see if it has a record of the domain name's IP address. If the DNS server has a record of the domain name in its cache, it returns the IP address to the requester without needing to perform a full DNS resolution.

DNS caching is beneficial because it can significantly reduce the time and network resources required to resolve DNS queries, especially for frequently accessed domain names. By caching DNS records, a DNS server can quickly respond to queries for commonly accessed domain names without needing to query other DNS servers, resulting in faster response times and lower network traffic.

```

○ (base) syedmumtahirmahmud@Syeds-MacBook-Pro cache server % python3 cached.py
Enter an address or enter 'data' to see the cahed data: www.yahoo.com

Before Decoding
b'2\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00www.yahoo.com 100.20.89.7 A 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.yahoo.com 100.20.89.7 A 86400'}
hiwww.yahoo.com 100.20.89.7 A 86400
Enter an address or enter 'data' to see the cahed data: www.google.com

Before Decoding
b'2\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00www.google.com 100.20.8.1 A 86400'

After Decoding
{(50, 0, 0, 1, 0, 0)} {'www.google.com 100.20.8.1 A 86400'}
hiwww.google.com 100.20.8.1 A 86400
Enter an address or enter 'data' to see the cahed data: data

_____Cached Data_____

www.yahoo.com 100.20.89.7
www.google.com 100.20.8.1

_____End_____

Enter an address or enter 'data' to see the cahed data: █

```

Figure 15: Content of DNS Caching

References

- [1] *General DNS Reference Information — BIND 9 documentation*. [Online; accessed 2023-02-16].
- [2] What is DNS? – Introduction to DNS - AWS. <https://aws.amazon.com/route53/what-is-dns/>. [Online; accessed 2023-02-16].
- [3] James Kurose and Keith Ross. *Computer networking: A top-down approach, global edition*. Pearson Higher Ed, oct 23 2018. [Online; accessed 2023-02-16].
- [4] Staff Writer. What's the difference between ANAME, AAAA, A, and CNAME records? <https://constellix.com/news/dns-record-types>, sep 27 2021. [Online; accessed 2023-02-16].