Department of
Computer Science and Engineering

Title: Distributed Database Management and Implementation of Iterative and Recursive Queries of DNS Records

Computer Networking Lab

CSE 312

Green University of Bangladesh

1 Objective(s)

- To gather knowledge about distributed database and Domain Name System (DNS).
- To gather knowledge on how to manage distributed databases and iterative & recursive queries of DNS records.
- To implement iterative and recursive queries of DNS records in Java.

2 Problem analysis

DNS is a global system for translating IP addresses to human-readable domain names. When a user tries to access a web address like "example.com", their web browser or application performs a DNS Query against a DNS server, supplying the hostname. The DNS server takes the hostname and resolves it into a numeric IP address, which the web browser can connect to.
A component called a DNS Resolver is responsible for checking if the hostname is available in local cache, and if not, contacts a series of DNS Name Servers, until eventually it receives the IP of the service the user is trying to reach, and returns it to the browser or application. This usually takes less than a second.

3 Distributed Database Management

A distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network. In a distributed database system, data is stored and managed in a distributed manner, allowing for scalability, fault tolerance, and improved performance.In general, distributed databases include the following features:

- **Location independency:** Data is independently stored at multiple sites and managed by independent Distributed database management systems (DDBMS).
- **Network linking:** All distributed databases in a collection are linked by a network and communicate with each other.
- **Distributed query processing:** Distributed query processing is the procedure of answering queries (which means mainly read operations on large data sets) in a distributed environment.
- **Hardware independent:** The different sites where data is stored are hardware-independent. There is no physical contact between these Distributed Database In Dbms which is accomplished often through virtualization.
- **Distributed transaction management:** Distributed Database In Dbms provides a consistent distribution through commit protocols, distributed recovery methods, and distributed concurrency control techniques in case of many transaction failures.

4 Iterative and Recursive Queries of DNS Records

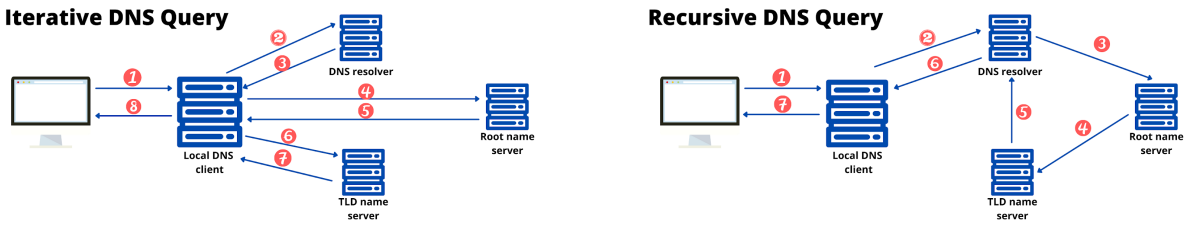
The goal of DNS (Domain Name System) is to resolve a fully qualified domain name (FQDN) to an IP address and the process is called name resolution.

A recursive DNS lookup is where one DNS server communicates with several other DNS servers to hunt down an IP address and return it to the client. This is in contrast to an iterative DNS query, where the client communicates directly with each DNS server involved in the lookup.

Let's take an example to see both of these in action — Suppose you want to go to www.google.com, you type it in the URL field in your browser, → the browser then checks if there is any previous name resolution for this address — first in your browser's cache memory, and if not found, it checks in a text file called hosts C:/Windows/System32/drivers/etc/hosts

Now assume that there is no record in either of these locations, the computer then talks with the local DNS server asking for the IP address of www.google.com. This is the recursive query in which your computer(DNS client) talks with the local DNS server.

If the local DNS server doesn't have a record for name resolution for the site, it talks with the other higher-level DNS servers in the network which can provide a referral for the IP address. This is the iterative query.



(a) (b)

Figure 1: (a) Iterative queries of DNS records (b) Recursive queries of DNS records

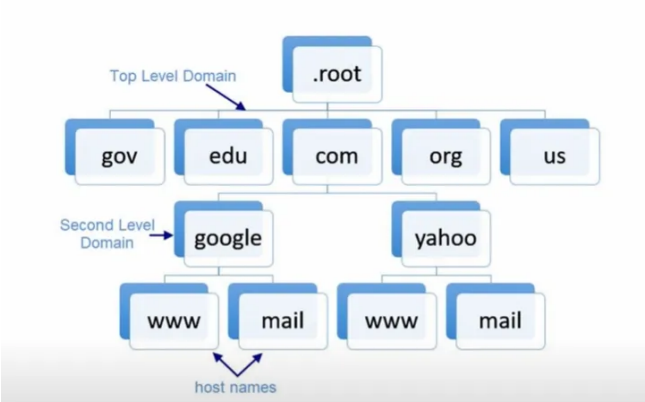


Figure 2: Hierarchical structure of domains

As a DNS server finds the name resolved for the IP address we requested, it replies the client with the IP address and the browser stores the information in memory and the next time you look for www.google.com you can avoid the whole process because you have it in memory.

5 Implementation in Java

First, let's create a simple Java class for a DNS record:

```
1  public class DNSRecord {
2      private String domain;
3      private String ipAddress;
4
5      public DNSRecord(String domain, String ipAddress) {
6          this.domain = domain;
7          this.ipAddress = ipAddress;
8      }
9
10     public String getDomain() {
11         return domain;
12     }
13
14     public String getIpAddress() {
15         return ipAddress;
16     }
17 }
```

The first program will demonstrate an iterative DNS query, and the other will demonstrate a recursive DNS query. Now, let's create a class for the iterative DNS query:

```
1  import java.util.ArrayList;
2  import java.util.List;
```

```
3
4  public class IterativeDNSQuery {
5      private List<DNSRecord> dnsRecords;
6
7      public IterativeDNSQuery() {
8          dnsRecords = new ArrayList<>();
9          dnsRecords.add(new DNSRecord("example.com", "93.184.216.34"));
10         dnsRecords.add(new DNSRecord("example.org", "2606:2800:220:1:248:1893:25
               c8:1946"));
11     }
12
13     public String findIpAddress(String domain) {
14         for (DNSRecord record : dnsRecords) {
15             if (record.getDomain().equals(domain)) {
16                 return record.getIpAddress();
17             }
18         }
19         return "Not found";
20     }
21
22     public static void main(String[] args) {
23         IterativeDNSQuery query = new IterativeDNSQuery();
24         System.out.println("example.com: " + query.findIpAddress("example.com"))
               ;
25         System.out.println("example.org: " + query.findIpAddress("example.org"))
               ;
26     }
27 }
```

The output of the above program:

```
1  example.com: 93.184.216.34
2  example.org: 2606:2800:220:1:248:1893:25c8:1946
```

And finally, let's create a class for the recursive DNS query:

```
1  public class RecursiveDNSQuery {
2      private DNSRecord[] dnsRecords;
3
4      public RecursiveDNSQuery() {
5          dnsRecords = new DNSRecord[2];
6          dnsRecords[0] = new DNSRecord("example.com", "93.184.216.34");
7          dnsRecords[1] = new DNSRecord("example.org", "
               2606:2800:220:1:248:1893:25c8:1946");
8      }
9
10     public String findIpAddress(String domain, int index) {
11         if (index >= dnsRecords.length) {
12             return "Not found";
13         }
14         if (dnsRecords[index].getDomain().equals(domain)) {
15             return dnsRecords[index].getIpAddress();
16         }
17         return findIpAddress(domain, index + 1);
18     }
19
20     public static void main(String[] args) {
21         RecursiveDNSQuery query = new RecursiveDNSQuery();
22         System.out.println("example.com: " + query.findIpAddress("example.com",
               0));
```

```
23          System.out.println("example.org: " + query.findIpAddress("example.org",
                0));
24      }
25  }
```

The output of the above program:

```
1  example.com: 93.184.216.34
2  example.org: 2606:2800:220:1:248:1893:25c8:1946
```

6 Discussion & Conclusion

From this experiments we gather knowledge about distributed Database Management, Iterative and Recursive Queries of DNS Records. Iterative and Recursive Queries of DNS Records are also implemented in java.

7 Lab Tasks

- Implement iterative and recursive DNS record queries using Breadth-First Search (BFS) algorithm.

8 Lab Exercise (Submit as a report)

- Implement the non-recursive DNS resolution methods.

9 Policy

Copying from the internet, classmates, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.