- Assignment - 11

Problem Definition: Write a program for DNS lookup. Given an IP address input, it shouldreturn URL and vice versa**.**

Objective:

- ➢ Toget the host name and IP address.
- ➢ Map the host name with IP address and Vice-versa

Learning Objectives:

- ➢ Understand what is Domain Name System and DNS lookup working.
- ➢ Understand what is DNSStructure and Hierarchy.

New Concepts:

- ➢ Name Server and Domain NameSystem.
- ➢ DNS lookup, Zone

Theory:

Need for DNS:

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.When the Internet was small, mapping was done using a host file. The host file had only two columns: name and address. Every host could store the host file on its disk and update it periodically from a master host file. When a program or a user wanted to map a name to an

Address, the host consulted the host file and found the mapping.

Today, however, it is impossible to have one single host file to relate every address with a name and vice versa. The host file would be too large to store in every host. In addition, it would be impossible to update all the host files every time there is a change. One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet. Another solution, the one used today, is to divide this huge amount of informationinto smaller parts and store each part on a different computer. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used bythe Domain Name System (DNS).

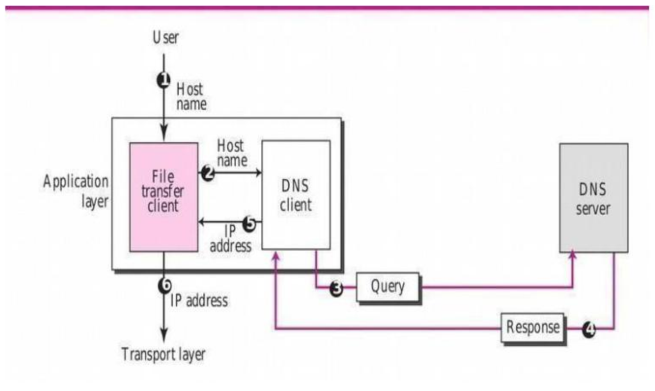


Figure 1: How TCP/IP uses a DNS client and a DNS server to map a nameto an address; the reverse mapping is similar

In Figure 1, a user wants to use a file transfer client to access the corresponding file transfer serverrunning on a remote host. The user knows only the file transfer server name, such as forouzan.com.However, theTCP/IP suite needs the IP address of the file transfer server to make the connection.

The following six steps map the host name to an IP address.

1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.
3. We know that each computer, after being booted, knows the address of one DNS server.
4. The DNS server responds with the IP address of the desired file transfer server.
5. The DNS client passes the IP address to the file transfer server.
6. The file transfer client now uses the received IP address to access the file transfer server.

NAME SPACE:

To be unambiguous, the names assigned to machines must be carefully selected from a name

space with complete control over the binding between the names and IP

addresses. In otherwords, the names must be unique because the addresses are

unique.

A name space that maps each address to a unique name can be organized in

two ways: flat orhierarchical.

Flat Name Space:

In a flat name space, a name is assigned to an address. A name in this space is a sequence ofcharacters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical Name Space:

In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of theorganization.

Domain Name Space:

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels:level 0 (root) to level 127 (see Figure 2).

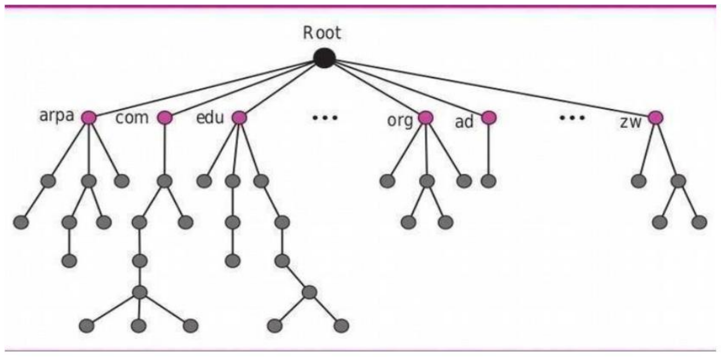


Figure 2: Domain Name Space

Label:

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domainnames.

Domain Name:

Each node in the tree has a domain name. A full domain name is a sequence of labels separated bydots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.

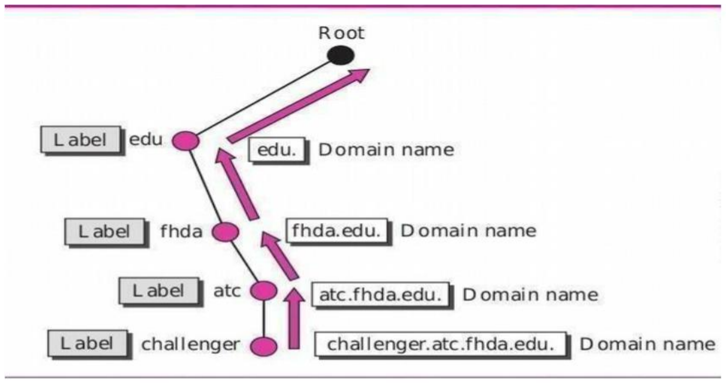


Figure 3 shows some domain names.

Domain:
A domain is a subtree of the domain name space. The name of the domain is the name of the nodeat the top of the subtree. Figure 4 shows some domains. Note that a domain may itself be dividedinto domains (or subdomains as they are sometimes called).

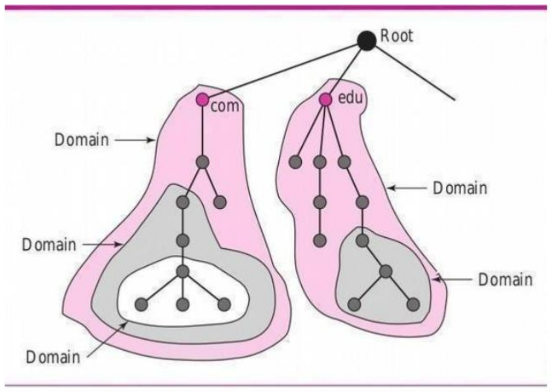


Figure 4 Domain

RESOLUTION:
Mapping a name to an address or an address to a name is called name-address resolution.

Resolver:

DNS is designed as a client-server application. A host that needs to map an address to a name ora name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise,it either refers the resolver to other servers or asks other servers to provide the
Information. After the resolver receives the mapping, it interprets the response to see if it is areal resolution or an error, and finally delivers the result to the process that requested it.

Mapping Names to Addresses

Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the country domains to find the mapping. If the domain name is from the generic domains section, the resolver receives a domainname such as “chal.atc.fhda.edu.” The query is sent by the resolver to the local

DNS server for resolution.

If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly. If the domain name is from the country domains section, the resolver receives a domain name such as “ch.fhda.cu.ca.us.” The procedure is the same. Mapping Addresses to Names a client can send an IP address to a server to be mapped to a domain name. As mentioned before, this is called a PTR query. To answer queries of this kind, DNS uses the inverse domain. However, in the request, the IP address is reversed and two labels, in-addr and arpa, are appended to create a domain acceptable by the inverse domain section. For example, if the resolver receives the IP address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is “121.45.34.132.in-addr.arpa.”, which is received by the local DNS and resolved.

Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved,the response travels back until it finally reaches the requesting client.

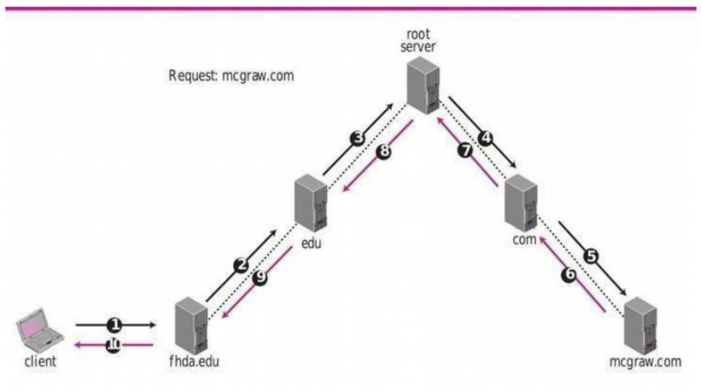


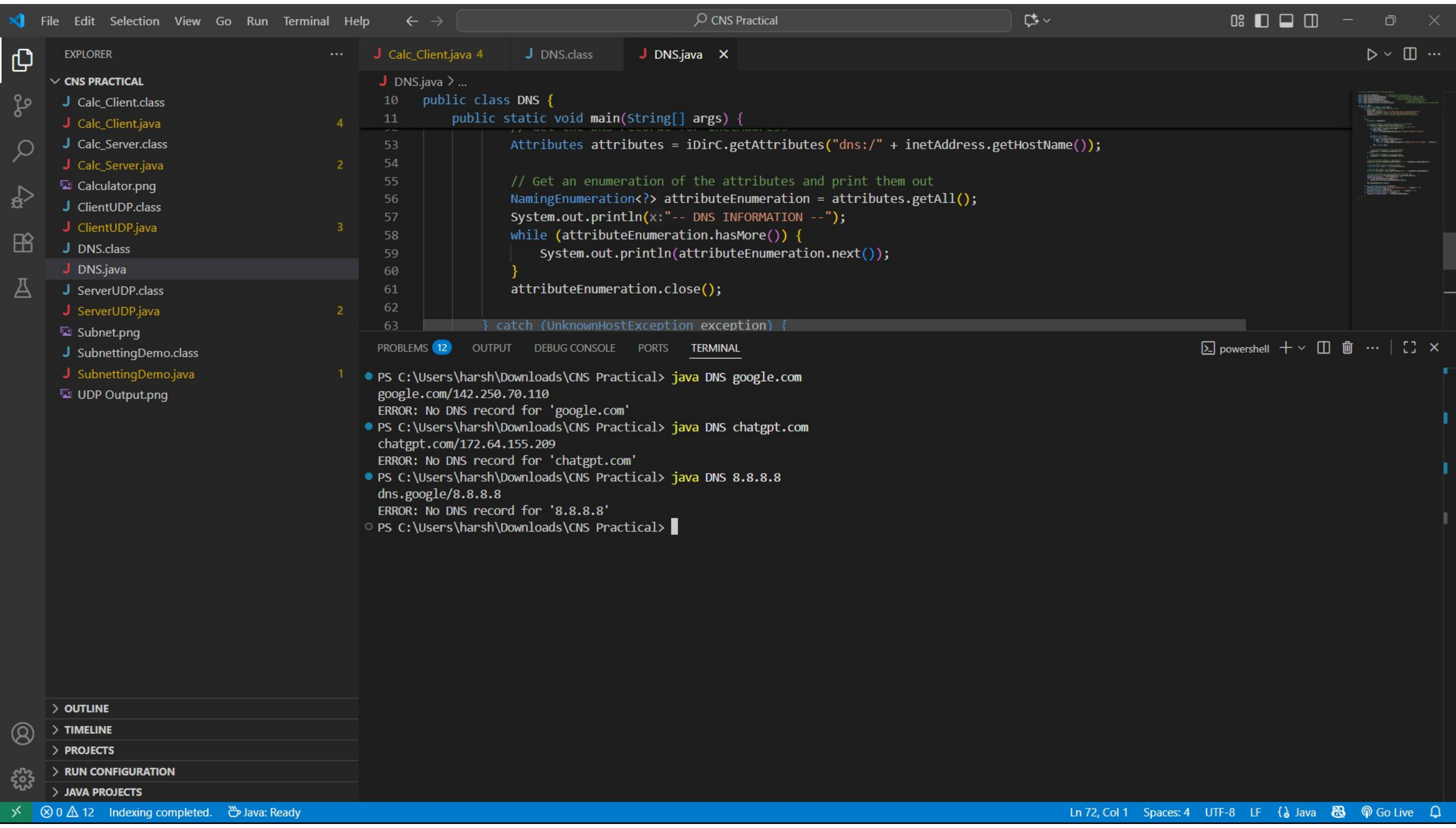
Figure 5: Recursive resolution.

Algorithm:

1. Give the input as website address e.g.www.google.com.
2. Get address, if first character is a digit then assume is an address
3. Convert address from string representation to byte array
4. Get Internet Address of this host address
5. Get Internet Address of this host name
6. Showthe Internet Address as name/address
7. Print Host Name and HostAddress
8. Get the default initial DirectoryContext
9. Get the DNS records forAddress
10. Get an enumeration of the attributesand

print them out# javac DNSLookup.java

 #java DNSLookup
google.com
google.com/64.233.167.99
-- DNS INFORMATION --
A: 64.233.187.99, 72.14.207.99, 64.233.167.99
NS: ns4.google.com., ns1.google.com., ns2.google.com.,
ns3.google.com. MX: 10 smtp4.google.com., 10
smtp1.google.com., 10 smtp2.google.com., 10



```java
public class DNS {
    public static void main(String[] args) {
            Attributes attributes = iDirC.getAttributes("dns:/" + inetAddress.getHostName());

            // Get an enumeration of the attributes and print them out
            NamingEnumeration<?> attributeEnumeration = attributes.getAll();
            System.out.println(x:"-- DNS INFORMATION --");
            while (attributeEnumeration.hasMore()) {
                System.out.println(attributeEnumeration.next());
            }
            attributeEnumeration.close();

        } catch (UnknownHostException exception) {
```

```
PS C:\Users\harsh\Downloads\CNS Practical> java DNS google.com
google.com/142.250.70.110
ERROR: No DNS record for 'google.com'
PS C:\Users\harsh\Downloads\CNS Practical> java DNS chatgpt.com
chatgpt.com/172.64.155.209
ERROR: No DNS record for 'chatgpt.com'
PS C:\Users\harsh\Downloads\CNS Practical> java DNS 8.8.8.8
dns.google/8.8.8.8
ERROR: No DNS record for '8.8.8.8'
PS C:\Users\harsh\Downloads\CNS Practical>
```