**Assignment No.**

**Title:** Implementation of DNS lookup

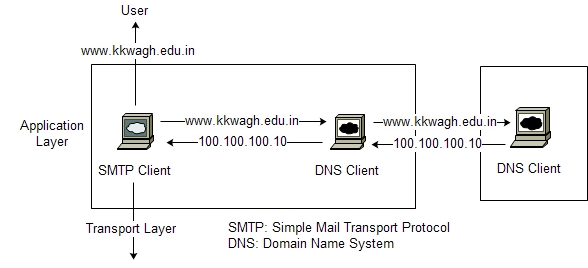
**Problem Statement:** Write a program for DNS lookup. Given an IP address input, it should return URL and vice-versa.

**Prerequisites:** Java/Python

**Objectives:** To know the working of DNS

**Theory: Domain Name Service** (**DNS)**

The protocol provides a distributed name resolution service is called Domain Name Service (DNS). The numerical IP addresses are not possible to remembering, so the computers running Domain Name Service provide the facilities like name lookup, changing a computer's name into the IP addresses computers need to communicate. Domain Name Service (DNS) is the technique used to translate the name of a computer into computer's IP address. This service is provided by server running DNS software such Microsoft DNS or another DNS server application. Using DNS web browser can translate an Internet domain name such as www.google.com, into an IP address where the website is located. This translation is provided by DNS servers using lookup. When the lookup take a name and return an address, it is referred to as forward DNS lookup. Domain Name Service can also translate IP addresses into domain names which is called reverse DNS lookup.



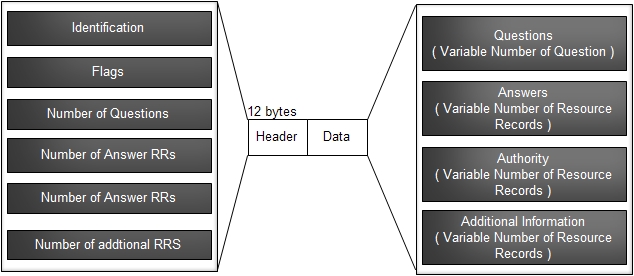
**Figure** DNS Architecture

Figure shows an example of how a DNS client/server architecture of DNS. A DNS server is requested with fully Qualified Domain Name (FQDN) and responds back with IP address apped. DNS uses UDP port number 53.

**DNS Message Format:**

There are the two types of DNS messages, request and reply messages have the same format

, as shown in below Figure.

****

**Figure** Request and Reply message Format of DNS

**Header**

Request and reply messages have the same header format and some fields set to zero for the request messages. The header is 12 bytes, and its format is shown figure. There are different subfields of header as follow,

* **Identification:** It is used to match a client reply to request. The client uses a different identification number each time it sends a request. The server duplicates this number in the corresponding response.
* **Flag:** There are a different flags are available in the flag field like query/reply flag indicates whether the message is a query (0) or a reply (1), authoritative flag is set in a reply message to indicate server is authoritative or not .
* **Number of question:** Thissubfield contains the number of queries in the question section of the message
* **Number of answer records:** This subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message.
* **Number of authoritative records:** This subfield contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message.
* **Number of additional records:** This subfield contains the number additional records in the additional section of a response message. Its value is zero in the query message.
* **Question section:** This section contains theinformation about the query that is being fired. It consists of two subfield name field and type field. It is present only on query messages. The name field contains the name that is being queried and type field that indicates the type of question being asked.
* **Answer Section:** This is a section consisting of one or more resource records. It is present only on response messages. It includes the answer from the server to the client.
* **Authority section:** This section contains records of other authoritative servers. It is present only on response messages.
* **Additional section:** This section contains records like hostname of a mail server associated with the alias name. It is present only on response messages.

**Name Space**

The names of system should be unique because the addresses are unique. The entity used to maps each address to a unique name it is called name space. Name space can be divided into two ways: flat name space and hierarchical name space.

**Flat Name Space**

In this name space system a name is assigned to an address. The name is nothing but a sequence of characters in space and there is no structure used to assign a name. The main disadvantage of a flat name space is that we cannot use in a large system such as the Internet because it should be centralized to avoid ambiguity and duplication. There is also a second disadvantage is memory space because single system cannot manage such a heavy load.

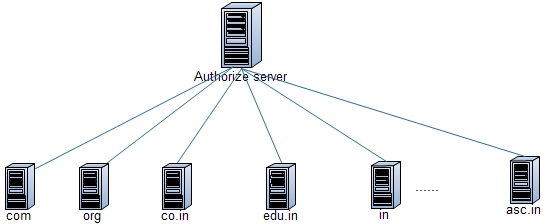
**Hierarchical Name Space**

In this name space system name space is organized hierarchically, means name is made of several parts. The first part defines the type of the organization, the second part defines the name of an organization, and the third part defines departments in the organization, and so on. In this type of name space can allow name space can be decentralized. The first two parts assigns by a central authority and responsibility of the rest of the name can be given to the organization

Itself. The organization can add remaining part as per their requirement. The organization need not worry that the name duplication they assigned to host. Because, even if part of an address is the same, the whole address is different. For example, assume two colleges have assigned common name *kkw*. The first college is given a name by the central authority such as *abc.edu,* the second college is given the name *xyz.edu* the end result is two distinguishable names: *kkw.abc.edu and kkw.xyz.edu.*

**Domain Name Space**

The domain name space was designed using a hierarchical name space. In this structure names are defined in an inverted-tree structure with the root at the top.

****

**Figure** Domain Name Space

* **Label**

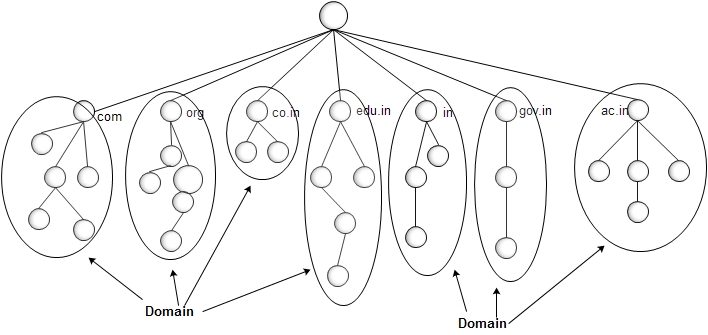
It is an inverted-tree consists of number of nodes and has assigned a label, which is a string with a maximum of 63 characters. The root label is a null string. DNS designed such a way that has different labels, which guarantees the uniqueness of the domain names.

* **Domain Name**

Domain name has assigned to each node in the tree. A full domain name is nothing but a sequence of labels which is separated by dots (.). The domain names are always read from the bottom node up to the root. The last label of a domain name is the label of the root. This means that a full name or FQDN is always ends in a null label, which means the last character is a dot because the null string is nothing. Figure shows some domain names.

* **Domain**

A domain is a subpart or subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Following figure shows some domains. Domain can be divided into domains it is called subdomains.

****

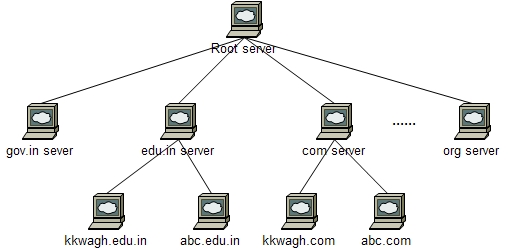
**Figure** Domain

**Distribution of name space**

Distribution of name space is a process of storing domain name space among many computers. However, it is very inconvenient and also unreliable to store just one computer. It is not able to handle all requests by single system. It is might create failure in data in access. The solution to these problems is to distribute the data or information among many computers called DNS servers.

**Hierarchy of Name Servers**

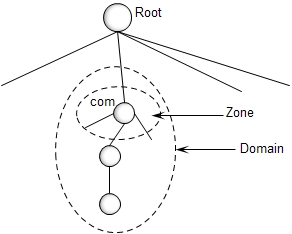
Hierarchy of name space means divide the whole space into many domains based on the first level. The root is a first node to create many domains. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domain. i.e. subdomains. The hierarchy of servers is same way that we have a hierarchy of names.



**Figure** Hierarchy of Name Server

* **Zone**

As we know space divides among many domains, means the hierarchy of servers divides many servers. It is not possible to store whole information or records on single server. What a server is responsible or has authority over is called zone. A zone file is a computer file in ASCII text format that contains all resource records for a given domain. If a server does not divide the domain into smaller domains, then the *domain* and the *zone* are same.

****

**Figure** Zone

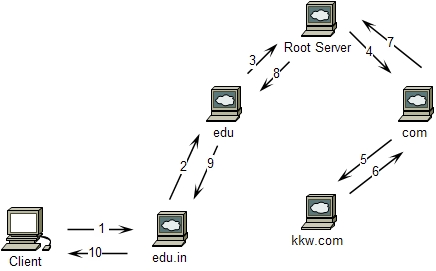
**DNS Resolution**

For mapping the IP addresses to name or name to IP address we require a process, resolution is the process of converting name into an IP address or an IP address back into a name. This process always starts with asking the root servers which name server to ask about the domain name that is being resolved. The root name servers will refer you to another name server which will refer you to another name server and so on. Finally, we get the name server which is authoritative for the domain name. The authoritative server will look in its zone files and respond with an answer.

DNS uses a client/server application that map an address to a name or a name to an address calls a DNS client called a resolver. The resolution process uses the resolver for mapping the name and address entity. There are two types of resolution,

1. **Recursive Resolution**

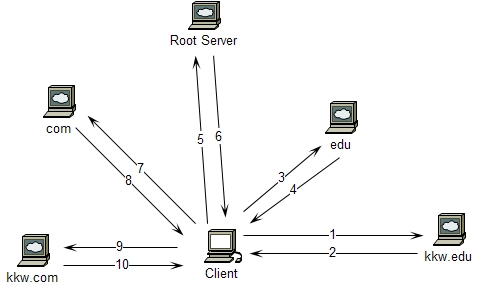
The client (resolver) can ask query to the name server, means that the resolver expects the server to send the final answer. If the server is the authority for the domain name, it checks its records and responds. If the server is not the authority, it passes the request to another server usually the upper node and waits for the response. If the upper node is the authority, it responds; otherwise, it passes the query to yet another server. When the query is finally resolved, the response comes back until it finally reaches the requesting client. This is called recursive resolution and is shown in Figure.



**Figure** Recursive Resolution

1. **Iterative Resolution**

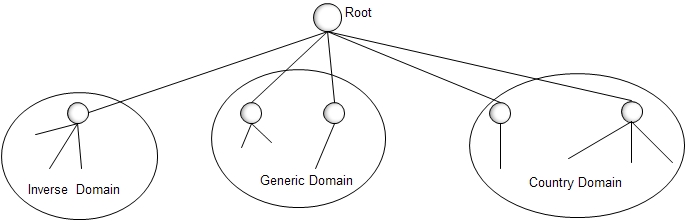
If the client does not possible to ask the query recursively, in this condition mapping can be done iteratively.



**Figure** Iterative Resolution

If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client has to send query to second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Once again the client must repeat the query to the third server. This process is called iterative resolution because the client repeats the same query to multiple servers and is shown in Figure. **DNS in the Internet**

DNS is a protocol uses in Internet, as we know the domain name space also called tree is divided into different domains. In Internet domains are classify into generic domains, country domains and inverse domain as shown in Figure.

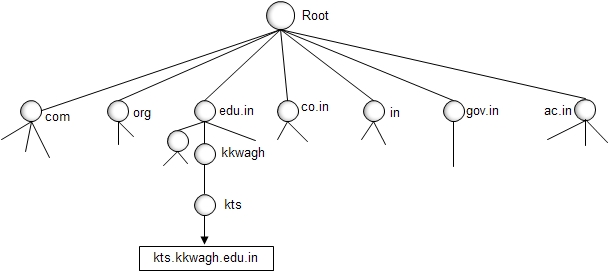
****

**Figure** Different types of DNS in internet

Following are the different types o DNS in internet are given,

* 1. **Generic Domains**

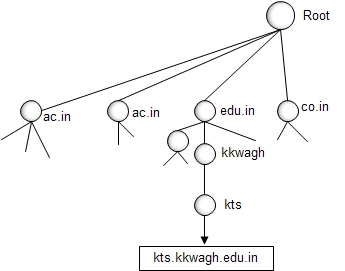
The generic domainsdefine the domain according to their generic behavior. Each node in the tree defines a domain,which consists of domain name space database. The generic domains section allows 14 possible labels. These labels describe the organization types like .com, .net, .edu, .gov, etc.

****

**Figure** Generic Domains

* 1. **Country Domains**

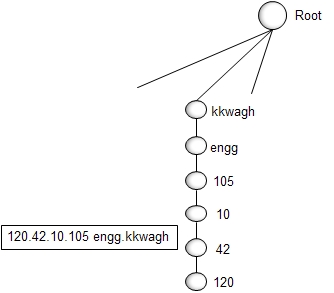
The country domains define the domain according to the two-character country abbreviations (e.g., us for United States, in for india ). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.)

****

**Figure** Country Domains

* 1. **Inverse Domains**

The inverse domain is used to map an address to a name. To handle a inverse query, the inverse domain is added to the domain name space.



**Figure** Inverse Domains

**Conclusion:** Thus we have studies working of DNS

**Program**

import socket

choose=True

while choose:

print("\nMenu\n(1)Get Host by Name \n(2)Get Host by Address\n(3)Quit")

choose=raw\_input("Enter the choice ")

if choose=="1":

addr1 = socket.gethostbyname('google.com')

print(addr1)

elif choose=="2":

addr3=socket.gethostbyaddr("216.58.199.142")

print(addr3)

elif choose=="3":

exit()

else:

print("Invalid choice, please choose again")

print("\n")