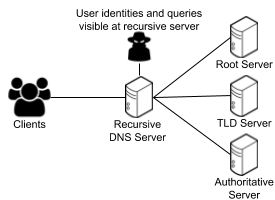
**Delving into DNS -**

Basically each website has an IP address which is a set of numbers or in the newer regime, a set of alphanumerics. Now, being impossible to know the IP’s of all the websites for a human, a DNS - Domain Name System does the job. It is basically a server involved in the regulation process.

The process of resolving a domain name to an IP-

1. What happens is that whenever a user enters a particular website name (not it’s IP), then it has to be converted to an IP for the network to understand as to which page is being requested for. As a first step, the user’s system’s cache memory is scanned to see if it knows the IP of the website being requested for.
2. If not, the user’s system requests a resolver for the IP. The resolver then scans it’s cache memory to see if it knows the IP and send it back to the user’s system.
3. If not, it then requests the root server for the IP which is the top of the DNS hierarchy. The root server then directs the resolver to the corresponding TLD server to look for the IP.
4. TLD servers are the top level domains that store the information of a particular top level domain such as .com.
5. Each TLD has their own name servers that store information of that domain called as the Authoritative name servers.
6. The Authoritative name servers have the information regarding the IP of the requested domain and they send the IP back to the user to show it the desired webpage.

The mapping of URLs varies from server to server in the sense that it can be simple enough or get more and more complex in order to increase efficiency. Like, some servers break up the URL into hierarchical paths and search for the parent ones and restrict the scope of its location and hence ultimately checks the existence of the requested URL and return the page as desired. For others, it may be actually simple enough that a direct table sort of a structure is maintained for the URLs and the corresponding pages.



**IP - Internet Protocols -**

**TCP - Transmission Control Protocol**

It is a connection based mechanism i.e. once a connection is established, data can be sent bi-directionally. It serves as a reliable method to send and receive data. But being a connection based model, which has to spend time in recovery too makes it a bit slower but at the same time safe enough.

**UDP - User Datagram Protocol**

It is a connection less mechanism i.e. messages and data are sent in chunks. There is no proper connection established between the two terminals of the internet. Since it doesn't attempt error recovery, therefore it is quite faster as compared to it’s TCP counterpart. Error recovery basically means that once an error is encountered, then that particular packet of information is lost, no attempt is made to recover the information.

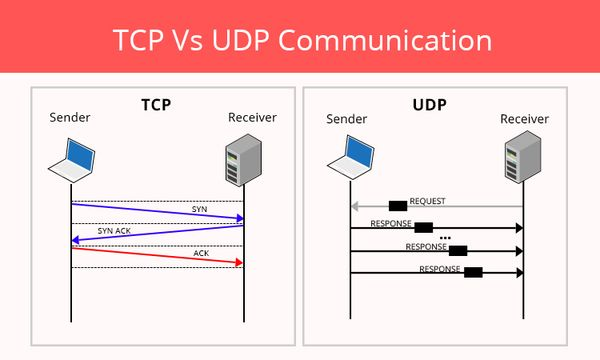
Basically whenever data is requested, the data is split up into chunks of data known as Datagrams/packets. These packets are then sent over the internet from source to the destination but they also include a header file that contains the IP of the destination computer. Now, these packets pass through gateways that are intermediate computers that match the IP in the header with the address range it handles, if not then they are passed on to other gateways and ultimately reach the destination. So, this basically is the hopping of data between routers when a request is made to the server.

**TCP in Web Browsing -**

What basically happens is that when a URL is entered, it initiates a TCP connection with the server. Now, after the server acknowledges the TCP connection being established, the browser then sends HTTP requests to the server to retrieve content which is sent by the server to the user in the form of information packets. Web Browsing is always based on TCP.

**UDP in live streaming -**

UDP being fast is used widely in live streaming and broadcasting where fast transfer of information is required. The reason why it is not used in web browsing is that it doesn’t authenticate the data sent and also it doesn’t encrypt the data, so it is easy for this data to be read by others on the transit.



**SERVERS -**

Servers are basically host computers that send the desired data to its clients as and when required. It basically is a HOST - CLIENT relationship. So, in very general terms, servers are basically systems that contain a lot of information that can be requested by the users over a network. This means that any computer can potentially be a server, even a personal laptop can be a server. In general, the SERVER - CLIENT relation is one of a request-response but at times it may be a relation that is publish-subscribe relation. There is an initial request-response step after which the client is sent information that matches with the subscribed data without confirmation of the client.

Now, what it therefore means to set up a server is basically establishing a computer system that stores data that is to be shared over a network and requested by its clients (the users).

The handling of requests -

Now, what actually happens is the union of the DNS server system and the TCP connection.

1. The user enters a URL. If not an IP, then a DNS server is contacted, which resolves it into an IP address.
2. The user’s system then initiates a TCP connection to the IP of the host server and sends a HTTP GET request.
3. The server then acknowledges this request of a connection and generates a HTTP response which is then sent back using the TCP connection.
4. The browser gets the appropriate response and then the connection is closed and the browser displays the required data.

**NGINX -**

It is an open-source web server software that was developed with an aim to handle HTTP requests but now it serves a lot of other features.

It is widely used in busy websites like e-commerce, search engines and cloud storage due to it’s faster speed than the conventional Apache. There is a major difference in it’s architectural design in the sense that conventionally each request is handled as a single thread. But NGINX handles it quite differently.

There is a master process under which are several worker processes which further have multiple worker connections under them. Unlike single thread mechanism, in this case similar requests are handled together by a worker connection which passes the data to the master process via the worker process. Each worker connection can handle multiple requests at the same time making the server very fast as compared to the conventional ones.

NGINX is said to follow an asynchronous, event-driven approach handling multiple requests in a single thread.

**127.0.0.1**

This is a loopback address i.e. it points to the same machine as the end-user. It is used to treat one’s own machine as a remote machine. It is a local host IP, unlike the IP’s used over a network to share information, it is recognised differently by the TCP connections as a loop-back address. Whenever an IP address is entered in a web browser, it is connected to a server over the internet but 127.0.0.1 is not sent over the internet because it is virtually talking to one’s own machine.This is done via a virtual interface that is created by the operating system.

This localhost is widely used by developers and network managers to test their program before actually uploading it on a server.

Using this we can achieve what is required in the problem statement.

I can host the webpage on the localhost of my machine as a server. Using the private IP address on my friend’s device connected by some network, he can view the web page by pulling a request to my localhost server.

What I tried was using python, I hosted the webpage on an IP address of my localhost and then viewed the html page on my phone by entering the private IP of my laptop connected over my mobile hotspot.