"Application Layer Protocols" :⇒

"Application layer" is responsible for all the services that the internet will provide.

There are thousands of services in internet & for every service there is a corresponding "Application layer protocol" available.

Eg:⇒ for web service "HTTP" is used

"Application Layer Protocols" for GATE syllabus :⇒

for web applications:
1. "DNS" (Domain Name Service) (DN to IP).
2. "HTTP" (webpages)
3. "FTP" (files)
4. "SMTP" & "POP" (for sending emails) (for retrieving emails)
5. "ICMP" (It works at N/W layer)

1. 'DNS' ("Domain Name Service") :⇒ We know that every host is identified by an "IP-Address," but remembering a "32-bit numbers" is very very difficult.

& when we shift our "host" (web site) from one location to another (one service provider to other service provider) then IP addr will change & hence it is not reached by users who knew my earlier IP address. But if they know my "domain name" they can easily reach me using "DNS".

Reasons for using "Domain names":

(i) They are "easy to remember".
(ii) "IP address" are not static.

Even if we use "Domain name", for communication purposes with layers model TCP "domain-name" has to be somehow converted to "IP address".

For this purpose we use DNS ("Domain Name Service")

"DN" —"DNS"→ "IP"

DNS :⇒ It is a protocol used for converting "Domain Names to IP-address."

There are various type of domains :⇒

1) "Generic domain" :⇒ (.com, .edu, .mil, .org, .net)

.Com (Commercial) .mil (Military)
.edu (education) .org (Non profit organizations)
.net (for networks as well as commercial)

(ii) "Country-domain" :⇒ (.in, .uk, .us)
Every country has a domain.

(iii) "Inverse-domain" :⇒ "Given an IP-add, we can find out what is the domain name of the machine running on this "IP-address."

("IP" → "DN")

P.T.O

"DNS" can provide both the mappings from

1) ("DN → IP") 2) ("IP → DN")

In "Windows_Machine" in order to convert "DN" to "IP".

we can type/use command "NSlookup www.google.com"

"DNS" is also able to do "load-balancing". (multiple entries for google being at diff servers)

let us now see how a "DNS database" is organized :⇒

Note :⇒ "Entire database of DNS" cannot be kept on "one servers",

but "distributed in nature".

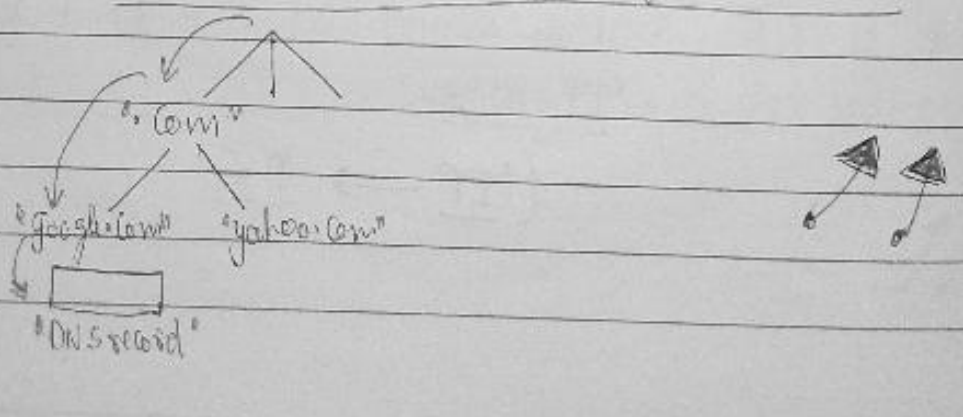
Note :⇒ for "every website" we are having a "DNS record" associated with it

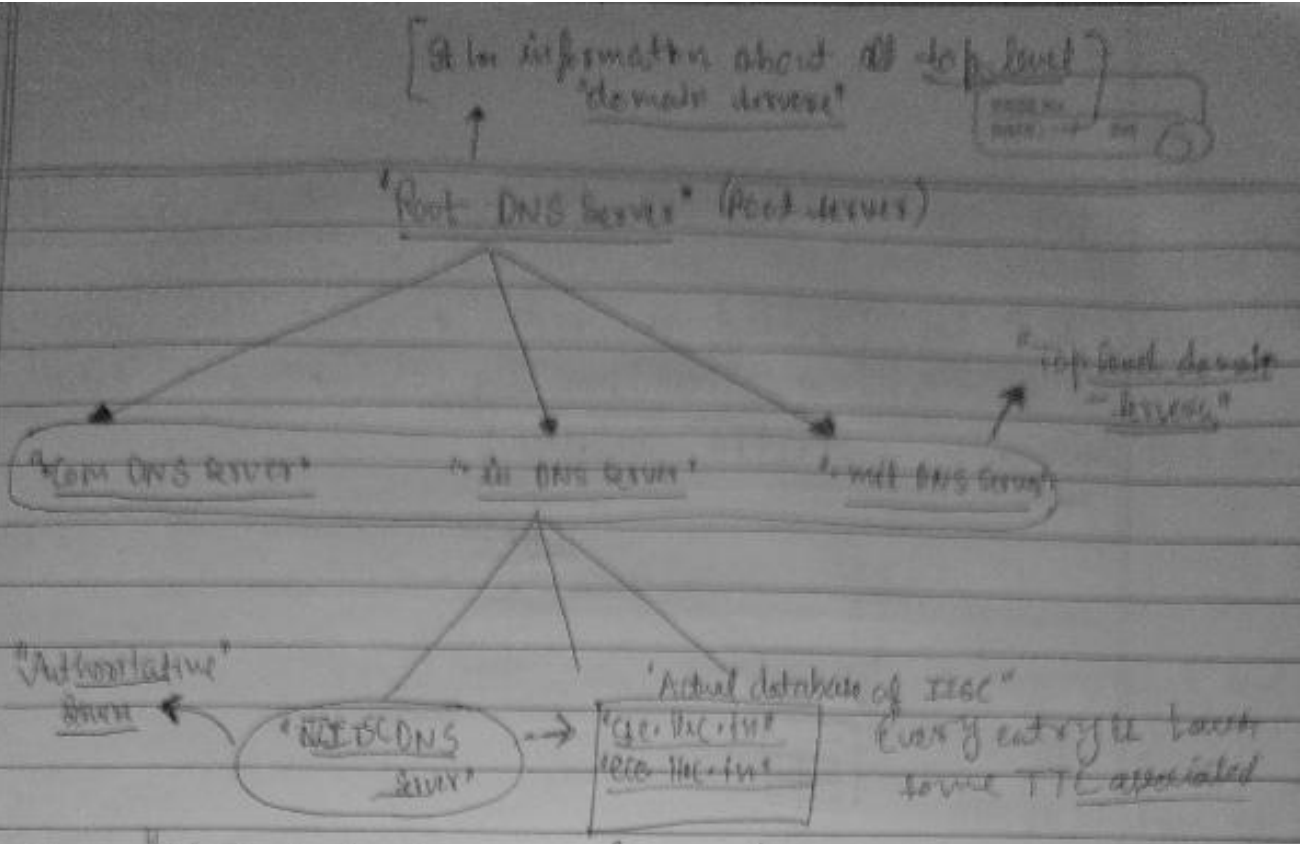
↳ "IP address"

↳ "Validity" (for how long the "IP-add" is associated with this "domain name")

Whenever we send a "DNS-request" we basically are supposed to get this "DNS-record".

So "DNS has a tree kind of structure"





These "Top level domain servers" are having something called as "Authoritative DN Servers" associated with them.

Eg:- Suppose we have a website as [iisc.in]

Under this we have

- [cse.iisc.in]
- [ece.iisc.in]

Now in order to get these entries, first we have to understand that we have to go to ".in". Then to "IISC server" & from there we can reach "these entries".

Now the problem is, what happens if the "root-server" fails.

So, if "root-servers fails" every thing will "go down". That is the reason, "IETF" has managed "13 root servers" all over the world.

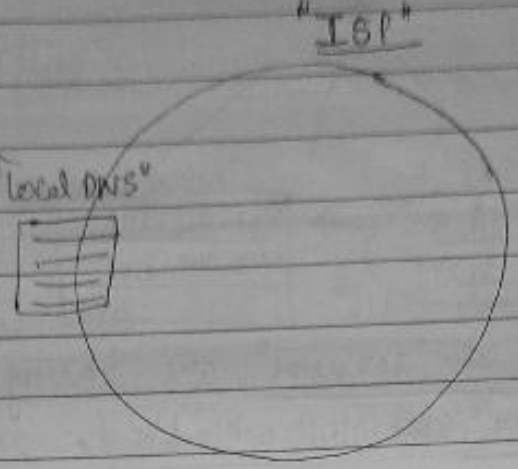
Each country will use a "root server" which is closest to it.

For "India" closest root servers is in "Tokyo".

P.T.O

We always need not to go to the "Root Servers"

ISP has provided us with a "Local DNS server" & in the "cache" of "local DNS server", we have entries for "frequently contacted websites".

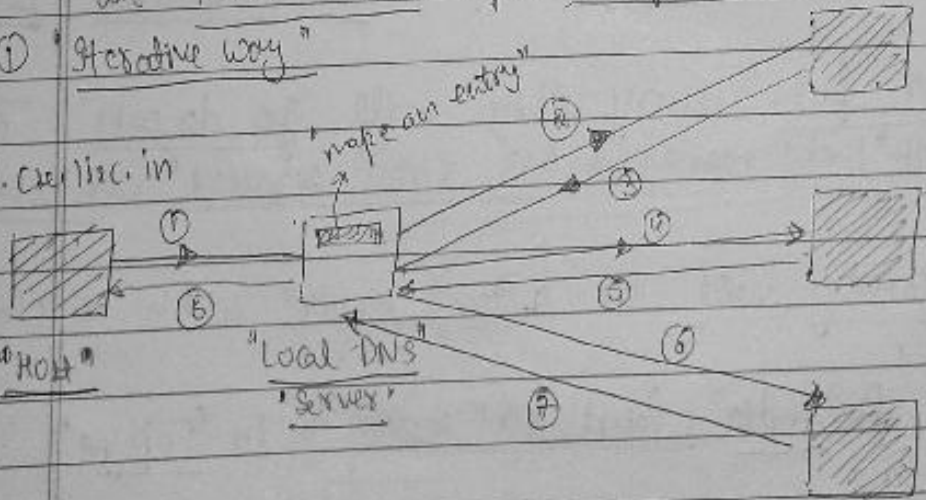


In case we ask for a "domain-name" which our "local DNS server" doesn't have any information about, then local DNS server will contact the "root servers"

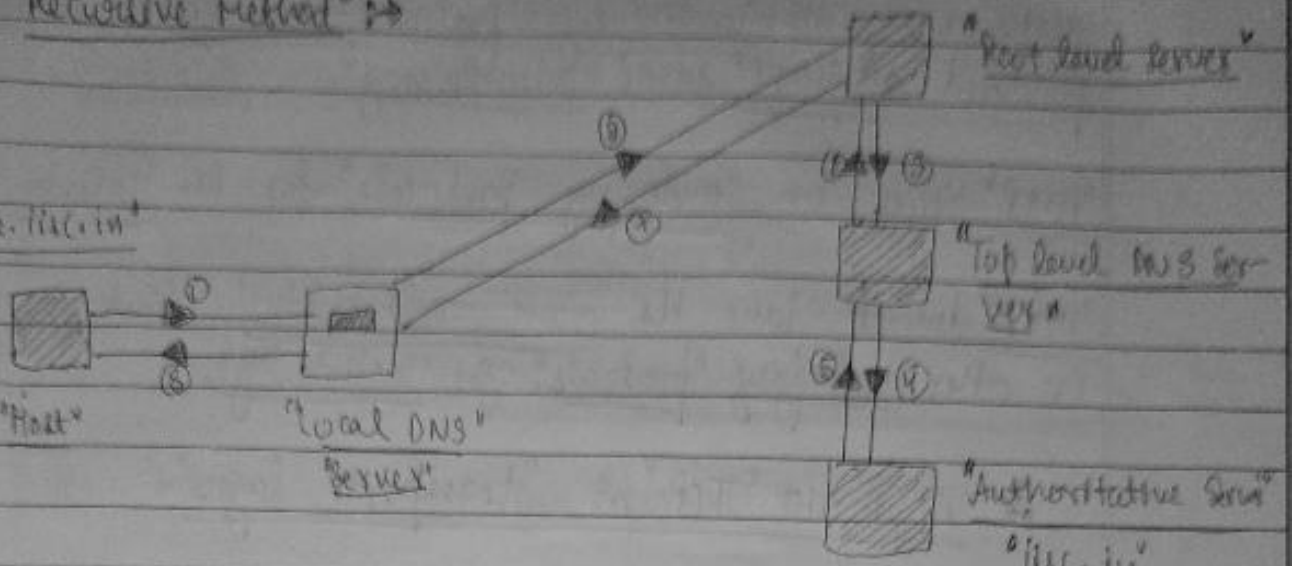
This "entire time" taken to resolve this "DNS request" is called "DNS-overhead".

→ There are two ways in which "Local DNS" can contact the "Root-Servers" for "information" ⇒

① "Iterative way"



② "Recursive Method" :⇒



All these Messages are "Req/: Reply"

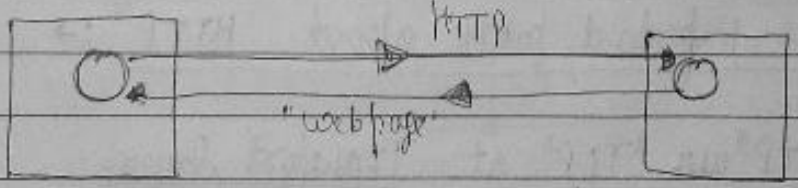
imp. ∴ "DNS" uses "UDP" at "Transport layer."

"HTTP" ("Hyper Text Transfer-Protocol") :⇒

"HTTP" works at [port no. "80"].

If we want to have a "webpage" we will contact "webserver" at "port no. 80".

Now the "webserver" will send the "webpage" we are looking for :⇒



"HTTP" wants reliability.

P.T.O

Whatever be the "underlying protocol" support should be provided to "HTTP" in a "reliable way".

"HTTP" relies on "underlying protocols" for the "reliability".

"HTTP" doesn't have its "own reliability". But it relies on other underlying "protocols" for "reliability".

∴ "HTTP uses "TCP" at "transport layer"

Note:⇒ "Everything is taken" care by "TCP" at "transport layer".

∴ if we are using "TCP" then "reliability is provided" to "HTTP".

"HTTP is an "inband protocol", which means both "data" & "command" goes in one connection only.

There is no distinction made between "Command" & "data". Both "data" & "command" will go & fill in one "buffer".

⇒ "Commands" will not get any "priority".

Note:⇒ "HTTP" is a "stateless-protocol" which means it will not keep "record" of the "users".

"Some important points about HTTP" :⇒

1⇒ "HTTP" uses "TCP" at Transport layer.
2⇒ "HTTP" is an "inband" protocol.
3⇒ "HTTP" is a stateless protocol.

Note :⇒ "Server" will pop a "small packet" called "cookie" on the client side, which will keep track of "user-activities".

Two popular versions of "HTTP".

1⇒ "HTTP 1.0", It use "non-persistent connection"

2⇒ "HTTP 1.1", It use "persistent-connection".

↳ "Bandwidth" is high, "Connection" is opened for a "long time".

*. "Both of them are currently being used."

⇒ "Some popular methods used in "HTTP" are as":⇒

1⇒ "Head" 2⇒ "Get" 3⇒ "Post" 4⇒ "Put" 5⇒ "Delete" 6⇒ "Trace" 7⇒ "Options" 8⇒ "Connect".

1⇒ "Head" :⇒ This "method" is used to get only "metadata" about a "webpage".

2⇒ "Get" :⇒ 3⇒ "Post" :⇒ ("filling-forms")

4⇒ "Put" :⇒ To "upload" an "object" on to the "server".

5⇒ "Delete" :⇒ Used to delete an "object".

6⇒ "Trace" :⇒ To trace out what are the "servers" the "data" is coming from.

"P "T.O"

3) 'options' :- to find out whether "optional-fields" are provided by the 'server' or not.

4) "Connect" :- It is used by "https" (used in "TCP Tunneling")

"FTP" (File Transfer Protocol) :-

It is used to "transfer the files"

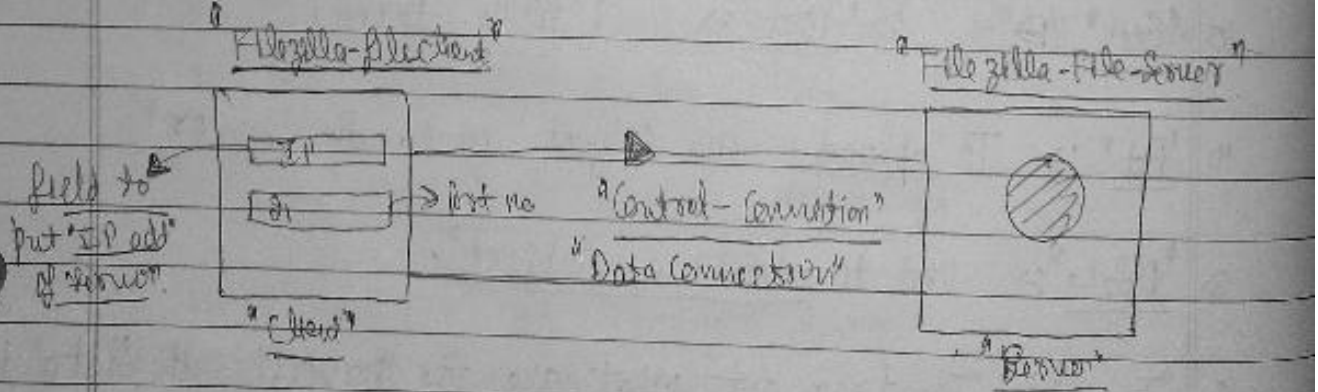
"Some Graphical tools to transfer files in Internet"

- "Tectia"
- "Filezilla"

let us suppose we want to "download file" from our friend Computer".

Our "friend" should initially install "filezilla" "file server" on his "Computer".

& on our system we should install "filezilla" "file client".



Now in order to download files, we should know about

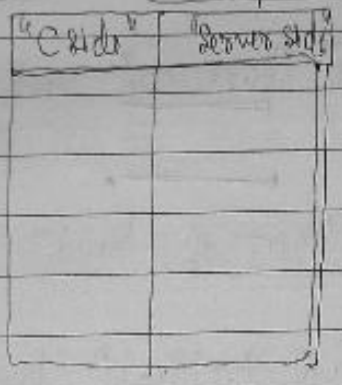
"public IP-address" of the "server".

Now once we open the "filezilla file-client", it asks us about the "current IP address" of the server.

Then it asks us about "port no." & we press "FTP" say at [port no. "21"]

Once this "connection" ("FTP connection") gets established

In our "window":→ "Two portions will appear":-



Now we can browse all the files on the server computer in such a way as we are browsing them on our local machine.

This "FTP-connection" b/w "Client & Server" is called as "Control-connection", as we can even send "Command signals" through this "connection".

We can even move files b/w "Client & Server Computer".

When we start "dragging the files", a "new connection" called "data connection" will be established b/w "client" & the "server".

"P.T.O"

Note :- Once we start "uploading" or "downloading" files "from client to server" or "server to client"

for each such transfer a "new connection" will be established & once "transfer is complete" the "data connection" is terminated.

∴ "Data-connection" is "Non-persistent"

whereas

"control connection" is "persistent"

for "Control-Connection" "port no." used is ["21"].

&

for "Data-Connection" "port no." used is ["20"].

"FTP" is called "OUT of band" protocol"

It is called "Out of band protocol" beoz both "control-signals" & "data-signals" are going out in "different connection"

Since "FTP requires reliability" ∴ the "underlying protocols" are supposed to provide "support for reliability" to FTP (file transfer protocol).

∴ "FTP" uses "TCP" at underlying "Transport layer"

& "FTP" is a "stateful protocol" Hence it will "create logs" for all the "user activities"

"SMTP & POP" ("Simple Mail Transfer Protocol" & "Post-office Protocol")

"Email is a file" so why don't we use "FTP" for "emails"

The reason is if have to transfer a file using "FTP" then both "Server & file client" should be online.

But when we are sending an "email" the "expectation" is that the receiver may not be available at that "time".

"Email" is the best thing because it doesn't "disturb anyone".

So we can send "email" at any time & the receiver may not be bothered at the time we are "sending an email".

Note :- "Gmail & hotmails" are "web based emails".

Note :- "Hotmail" is the first "email service" provided to general public, but it gives very less space.

Note :- "Email server" is actually "hosted on the webservers"

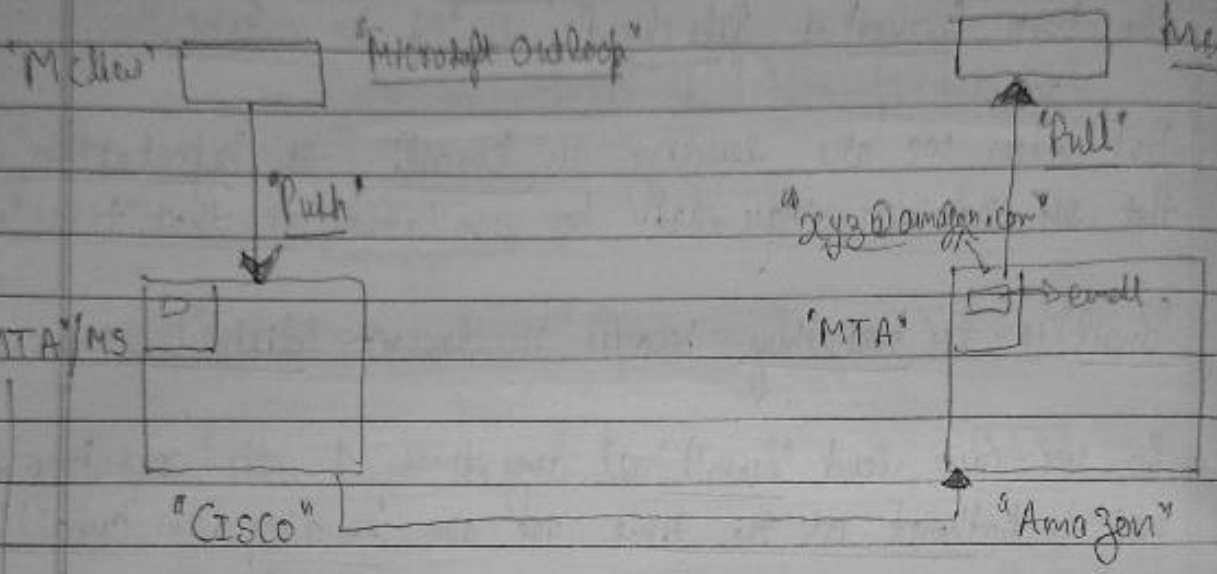
Now at address resolution, when we contact "DNS", we should even ask for this. (whether we want location of "email servers" or "web-servers").

Now let us see how this is "implemented"

:-

Assume that there is one company "Cisco" & other company "amazon".

Now employee at "Cisco" wants to send a message to another employee at "Amazon".



"MTA" ("Mail Transfer Agent") / MServer

Before sending an "email", its "first requirement" is to connect to an "email server" using an "email-client".

↳ This "email client" is generally "Microsoft outlook".

& then we open / login in "Microsoft outlook" compose the "email" & then we "push" it on to the "MTA / MS"

Let us suppose that the "destination - address" is "xyz@amazon.com"

↳ This is the "account name" to which we want to send the message / email.

Now "CISCO-server" will try to resolve the address (xyz@amazon.com).

↳ The "address" can be resolved by asking "DNS" (Domain Name Server) about it.

Now using "IP-address" of the "email-server" at "Amazon" we are going to push the "email / message"

& on the way there can be "many nos" of "mail-transfer agents" or "routers" in b/w. "CISCO & Amazon"

Now email send from "CISCO" will go & sit in "MTA" at Amazon"

Now "xyz" can now log into his "email-account" using "email-client" & then he can "pull" the email from the "Amazon's MTA".

Note :⇒ for Pushing we use "SMTP"

& for Pulling we use "POP"

Back in 1990's we can send only "text messages via email"

but now we can send even "videos", "images" via email".

"Non-text" → "Text"

"Text" → "Non-text"

"MIME" is used for this conversions.

So, when we send "Non-text Msgs", they first have to be converted to "text".

P.T.O

PAGE No.
DATE : / 201

and at "Receiver"

"Text → Nontext"

So, when we have to send any "videos", or "images" via, emails.

↳ we will use "MIME"

"MIME" ("Multipurpose "Internet" Mail Extension")