CS348 Notes
DNS (Application Layer)
Video Numbers: 30

OjMaha

I have prepared these notes by watching the videos from Networks Playlist. The following notes may be asynchronous and irrelevant to what Prof. Vinay teaches in class (cuz I do not pay attention during lectures lol). Further, these notes might not cover *everything* as explained in the video lectures. Consider these to be a supplemental read :). If you find any errors, do notify me so they can be edited.

DNS (Domain Name System)

Part of APPL. layer

Layer 3 doesn't understand URLs like www.google.com. → human-friendly

DNS does the job of converting them into IP addresses. → not human-friendly

↳ should be robust & non-hackable else might return wrong IP.

Idea:

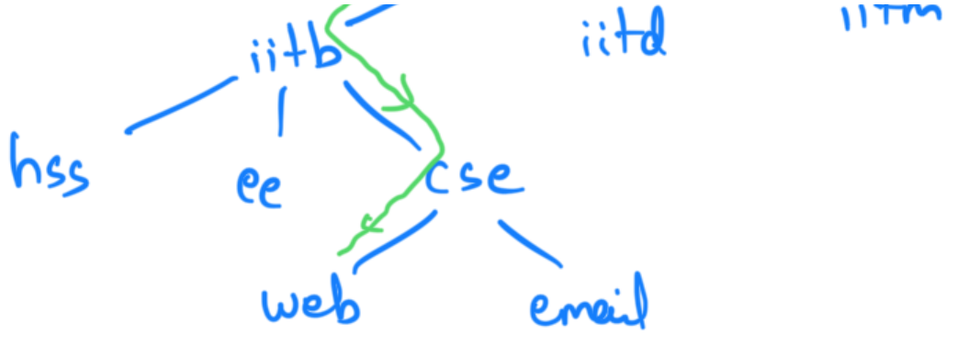
Server has well-known IP address and does all the job of sending info.

Problem 1: single pt. of failure.

Problem 2: Overload due to too many requests

We can use a somewhat centralised system.

The URL www.cse.iitb.ac.in can be resolved as follows:



Each of these have their own DNS server.

First the resolution is done via Root server. Root server doesn't have IP addresses of all DNS servers. But it has atleast the IP address of the DNS server 'in'. Then the DNS server at in gives IP address of the next DNS server 'ac' and so on.

DNS is a hierarchical system.

Root Server: → should be robust to attacks else whole internet down.

Well provisioned bandwidth and CPU. Else DoS attack.

↳ denial of service

flood it with so much fake traffic that it ← is no longer accessible.

We want multiple root servers s.t. jic someone cuts cable or smth then

↳ Redundancy.

ded.

We have 13 root servers. A . root-servers.net
⋮
M . root-servers.net

DNS pkt size of 512 bytes has space for 13 addresses at most.

That is why only 13 root servers.

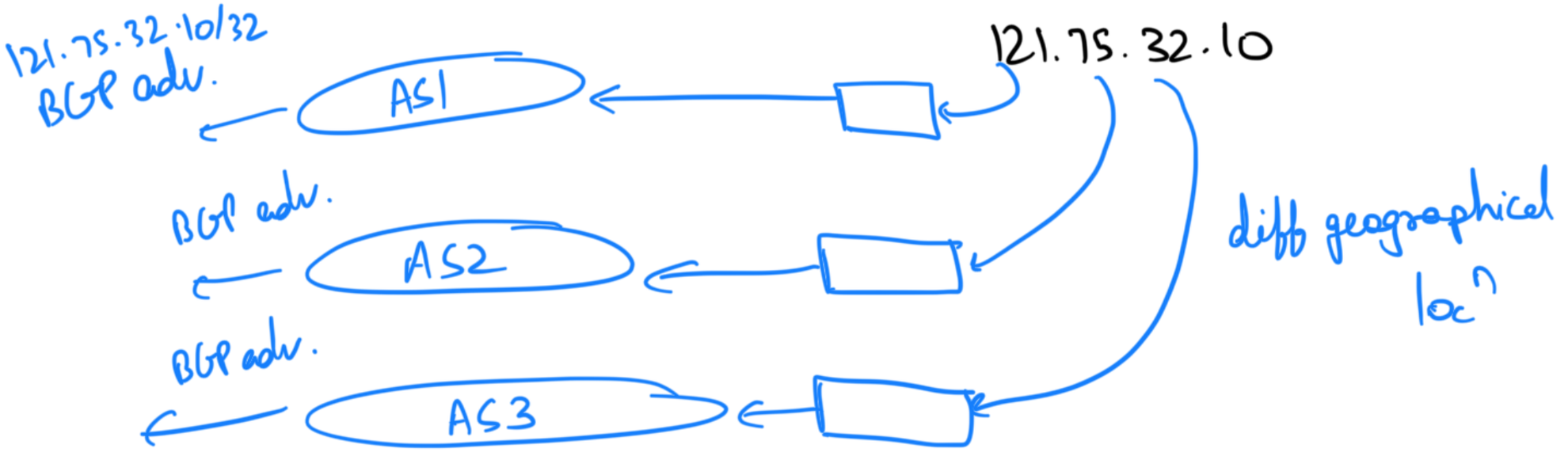
Use Anycast for more redundancy.

↓

sends out a packet s.t. it doesn't care which server it reaches in the end as long as the server is able to provide.

What we can do is have multiple machines the same IP addr. (???)

Say you have A.root-servers.net. with IP 121.75.32.10.
And this server is connected to multiple geographically distributed physical machines with the same IP.



Thus, a user might end up on any one of these paths depending on what AS it is a part of. (They are identical purpose-wise but this system allows robustness)
↳ the paths

There are 13 such root servers. Each of them connected to various physical machines distributed worldwide. Thus, it is hard to bring down the root-level DNS.

Resource Record: (RR)

< Name, Value , Type , Class , TTL >

Mapping

↓

How value shd be interpreted.

Type	Value
A	IP address
NS	Name Server (host running DNS service in domain corresponding to the "name")
CName	alias of "Name"; canonical name of host specified in "Name"
MX	name of host running mail server in domain specified in "Name"

eg:-

Name: www.google.com Value: IP addr of google server. Type: A. → cuz it's IP addr

We need a TTL cuz it's possible that the name shifts to another IP Address.

eg:- Root server has: < edu, a3.nstld.com, NS>

→ name of DNS server of "edu" domain.

: < a3.nstld.com , 192.5.6.32 , A >

↳ IP of "name"

Server a_3.nstld.com has

< princeton.edu, dns.princeton.edu, NS >

< dns.princeton.edu, 128.112.12.95, A >

Then server dns.princeton.edu has more dns entries with IP..

< www.cs.princeton.edu , coreweb.cs.princeton.edu , CName >

i.e. hosting the same server under a diff. name.

< www.cs.princeton.edu, mail.cs... , MX >

< mail... , 128.15... , A >

Hence, not all info is stored at the Root. Though the root contains info of DNS server of atleast one hierarchy lower.

Look Up Example:

DNS runs on UDP (port 53) We need DNS to be fast that's why.

Get local DNS server by manually configuring or DHCP.

Say the local DNS server has no entry except Root. It forwards the query using Anycast. The query reaches one of the 13 root servers. Suppose the root server doesn't have more info. Then it just sends back a_3.nstld.com; IPu_1 (returns all Resource (RR) Records corresponding to u_1). The local DNS server caches this info for time specified in TTL. Then the server queries for the server in RR. Then next layer server info is retrieved. This happens recursively till we finally retrieve the webpage. Then the final IP address travels through the layers and reaches the local DNS server. The local DNS server then provides the client with the IP address.

Note that all the info is cached till TTL. So if further queries result in the same few initial hierarchies lookup is fast.