Computer Networks COL 334/672

Application Layer: DNS

Sem 1, 2025-26

Recap: Application Layer

- HTTP
- Email
- **DNS** or Domain Name System
- P2P
- Video streaming

DNS: Domain Name System

DHCP : Application -layer protocol

- Humans understand names (google.com), Naming
- Internet hosts, routers understand IP address (12.123.12.12) Addressing
- *Q:* how to map between IP address and name, and vice versa ?

Two questions:

- How to design the database?
- How to retrieve the IP for a given domain name?

Domain Name System (DNS):

- *phone book* of the Internet
- *application-layer protocol:* hosts, DNS servers communicate to *resolve* names (address/name translation)
  - *note:* core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

End-to-end design principle

How would you design the DNS database?

- **Primary goal**: a database of domain to IP mappings

{a single machine storing all mappings?

- What are other design constraints/challenges?

↳ Scalable : # of queries to the table & size of the table

↳ Fast (Performance)

↳ Secure

AWS outage

↳ Resilience

↳ Dynamic

Approach 1: Centralized DNS

- single point of failure
- traffic volume
- maintenance
- ..

Replicate the database across N servers

↳ ① Scalability & Performance issues

② Expensive to maintain & update

Alternate

Split the database and store on multipple servers

Distributed and Hierarchical System

split the database?

- Q: On what basis to decentralize?

Hierarchical domain name space

E.g. cse.iitd.ac.in - IP mapping

① Split it alphabetically

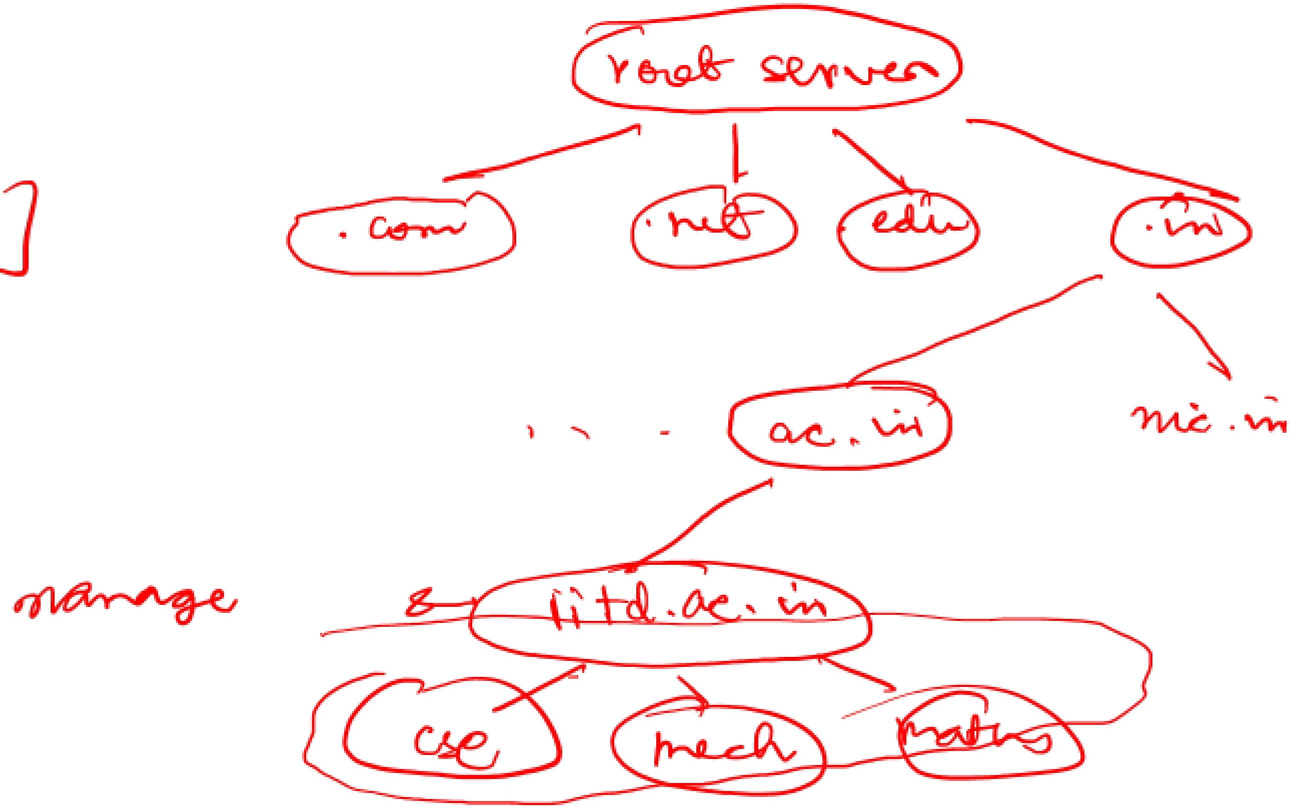
iitd.ac.in

cse.iitd.ac.in

→ should be on the same server

(Autonomy engrained into this)

② Split based on the domain name structure



Reverse DNS lookup

IP → Domains

Decentralized and distributed system

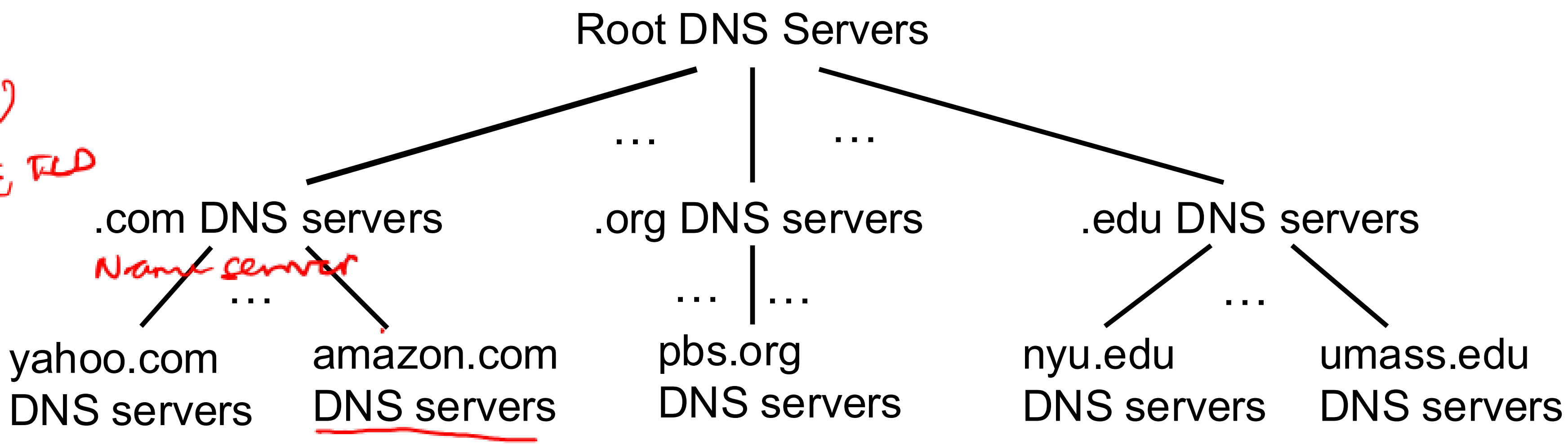
(col334.com , IPaddress)

- Q: On what basis to decentralize? Hierarchical domain name space
- Partition domain name hierarchy into zones managed by some authority → DNS registrars
  - E.g., ICANN is responsible for storing information about top-level domains
- Each zone corresponds to a name server

Name servers

(gTLD)

generic TLD



Root DNS Servers

. . .

. . .

.com DNS servers

Name server

.org DNS servers

.edu DNS servers

. . .

. . .

. . .

. . .

yahoo.com DNS servers

amazon.com DNS servers

pbs.org DNS servers

nyu.edu DNS servers

umass.edu DNS servers

can be managed by amazon

Root

Top Level Domain

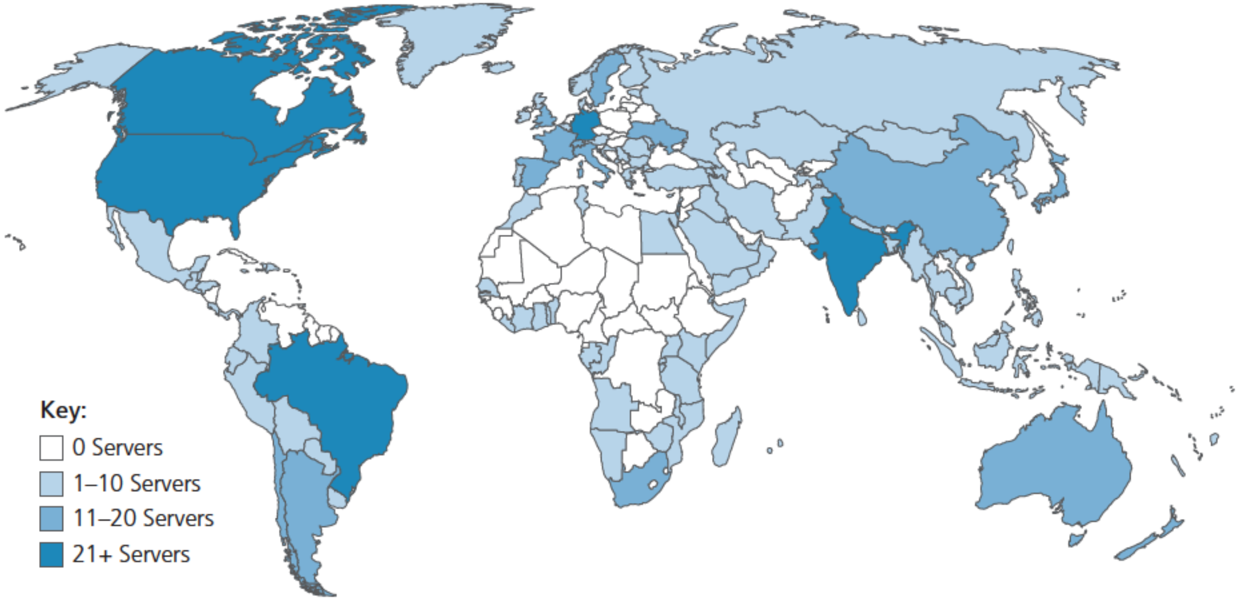
Authoritative

- Name servers are replicated and may be geographically distributed for reliability

DNS: root name servers

- official, contact-of-last-resort by name servers that can not resolve name
- *incredibly important* Internet function
  - Internet couldn't function without it!
  - DNSSEC – provides security (authentication, message integrity)
- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

13 logical root name "servers" worldwide each "server" replicated many times (~200 servers in US)



DNS records

DNS: distributed database storing resource records (RR)

RR format: (name, value, type, ttl) → used for caching

type=A

- name is hostname
- value is IP address

type=NS

- name is domain (e.g., foo.com)
- value is hostname of authoritative name server for this domain ns.foo.com

type=CNAME

- name is alias name for some “canonical” (the real) name
- www.ibm.com is really servereast.backup2.ibm.com
- value is canonical name

type=MX

- value is name of SMTP mail server associated with name

AAAA type record - IPv6

Example

(Top-level domain)

→ glue record

Root Name server

① .in , dns.xyz.in , NS, TTL

② dns.xyz.in , 112.112.112.112, A,

③ .com , dns.xyz.com , NS

④ dns.xyz.com , 113.113.113.113 , A

.edu

.net .

(who has authority
where is it?)

IN name server

① .ernet.in , dns.ernet.in
NS , TTL

② dns.ernet.in , x.y.z.w, A

ns.cloudflare.com

IP_x → IP_y

example.com

ns.example.com

cd334.com

COM Name server

① ~~example~~.com , ns.example.co
NS , TTL

② ns.example.com , IP_x , A ,TTL

① example.com , ns.cloudflare.com, NS,
TTL

② cd334.com , ns.cloudflare.com NS
TTL

How to do name resolution?

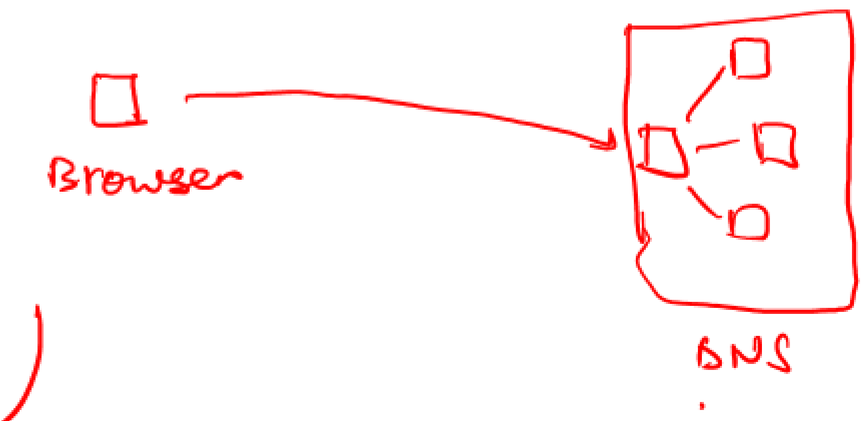
23 bit IP address

How To Do DNS Resolution?

UDP, TCP (TCP handshake)

- What is the transport protocol?

use UDP (Reliability can be implemented in application layer)



- Does the browser directly query the root server?

↓ Caching is very useful

Aggregation
↓
DNS resolver

Local DNS name servers

→ DHCP → gateway IP
→ DNS resolver

Public DNS resolver : 8.8.8.8 → Google
1.1.1.1 → cloudflare

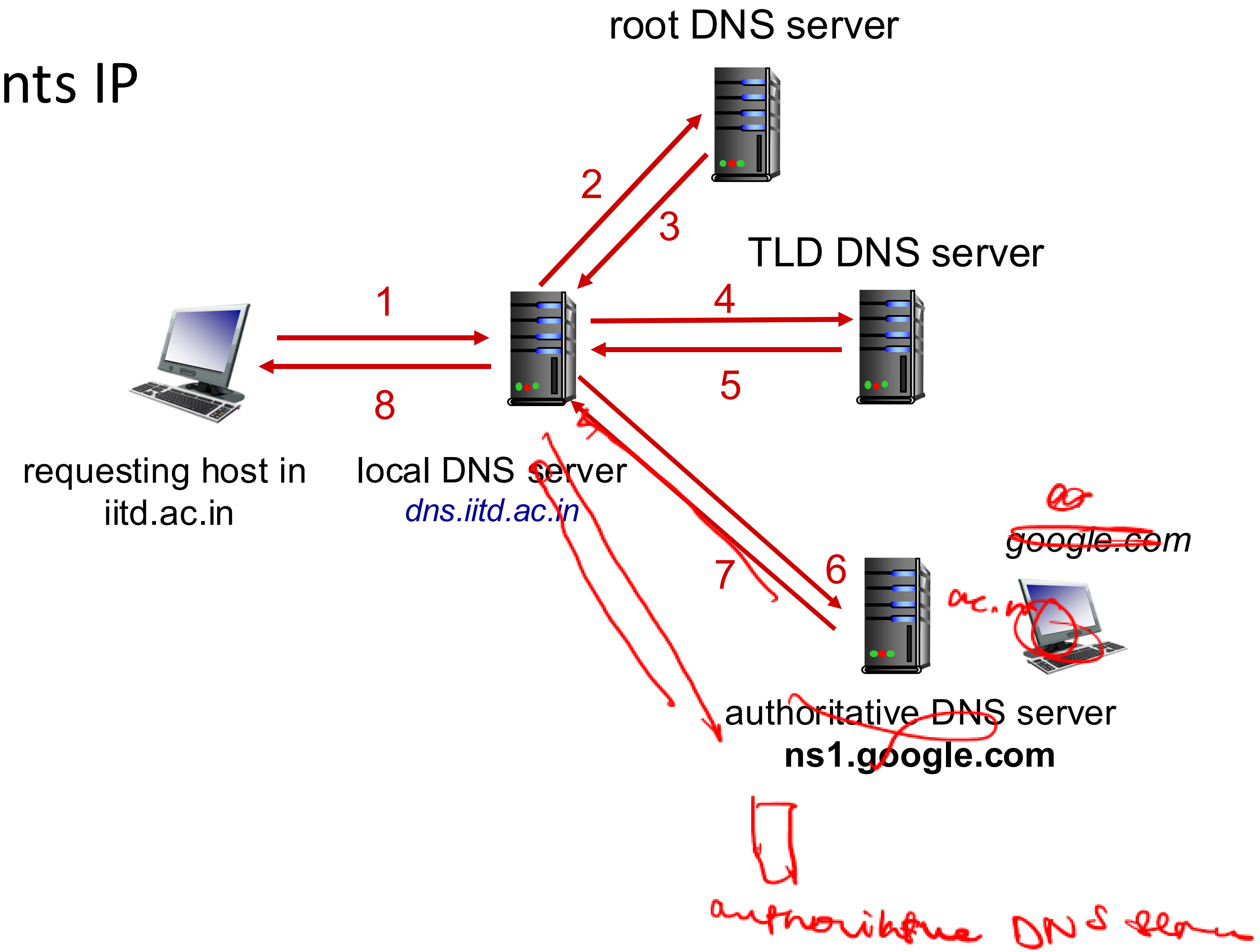
- when host makes DNS query, it is sent to its *local* DNS server
  - Local DNS server returns reply, answering:
    - from its local cache of recent name-to-address translation pairs (possibly out of date!)
    - forwarding request into DNS hierarchy for resolution
  - each ISP has local DNS name server; to find yours:
    - MacOS: `% scutil --dns`
    - Windows: `>ipconfig /all`
- local DNS server doesn't strictly belong to hierarchy

DNS name resolution: iterated query

Example: host at iitd.ac.in wants IP address for google.com

Iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



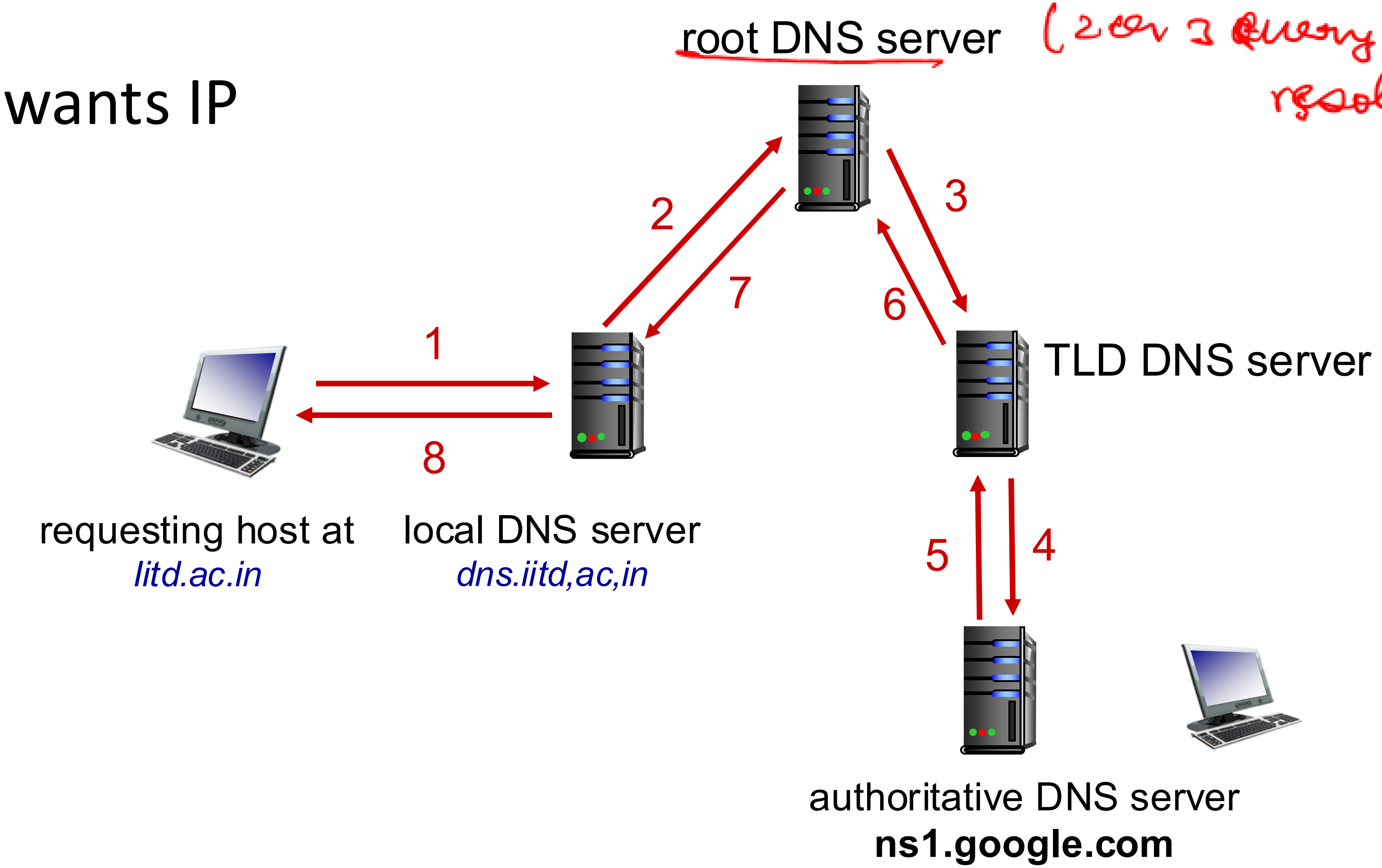
iitd -ac- in

DNS name resolution: recursive query

Example: host at iitd.ac.in wants IP address for google.com

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



(2 or 3 query resolution)

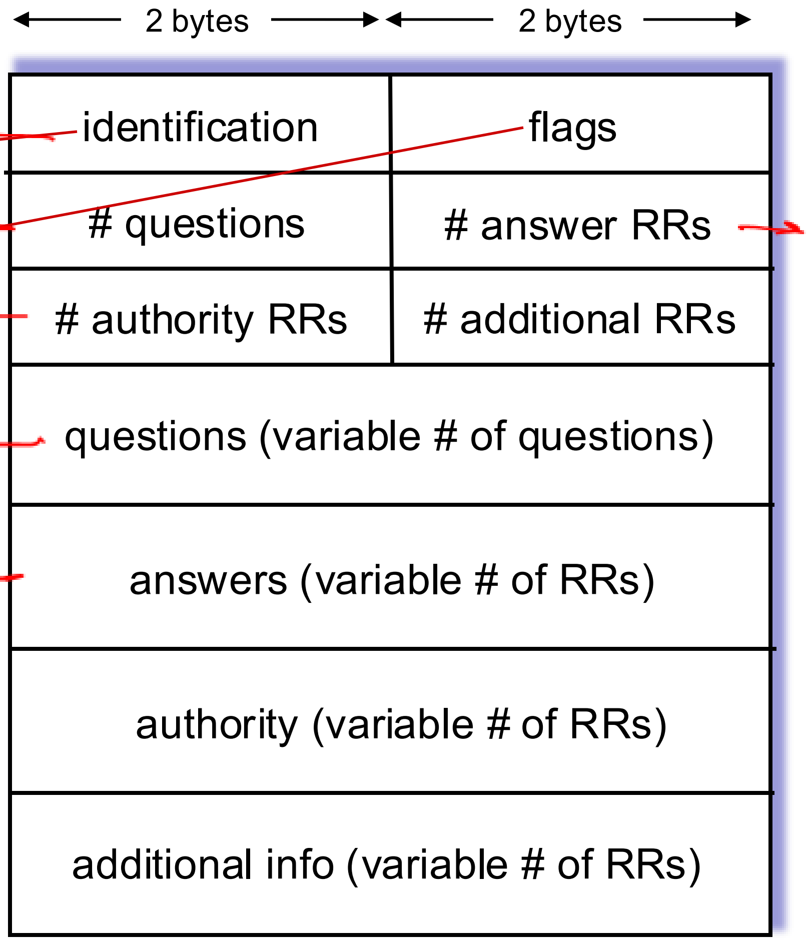
DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

message header:

- identification: 16 bit # for query, reply to query uses same #
- iitd flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

iitd.ac.in



2 bytes

2 bytes

identification | flags

# questions | # answer RRs

# authority RRs | # additional RRs

questions (variable # of questions)

answers (variable # of RRs)

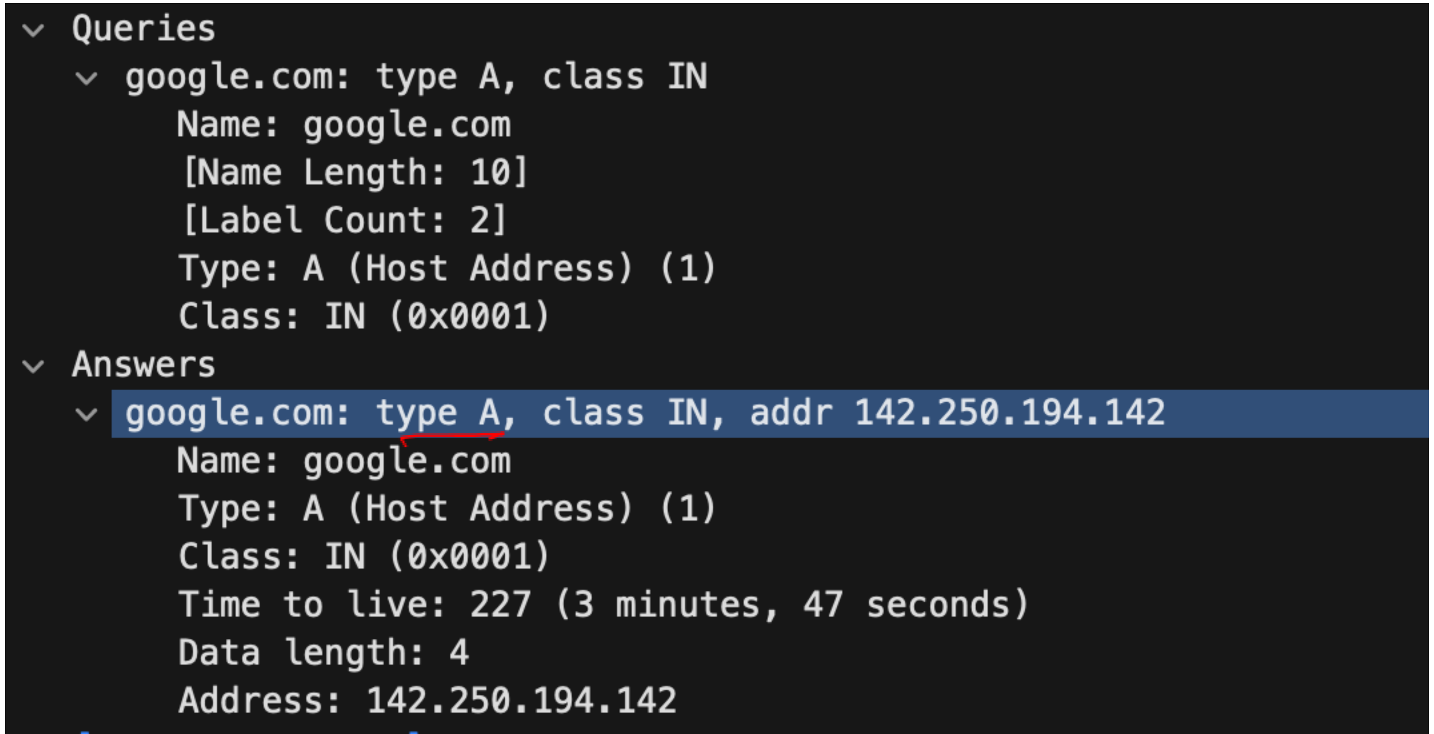
authority (variable # of RRs)

additional info (variable # of RRs)

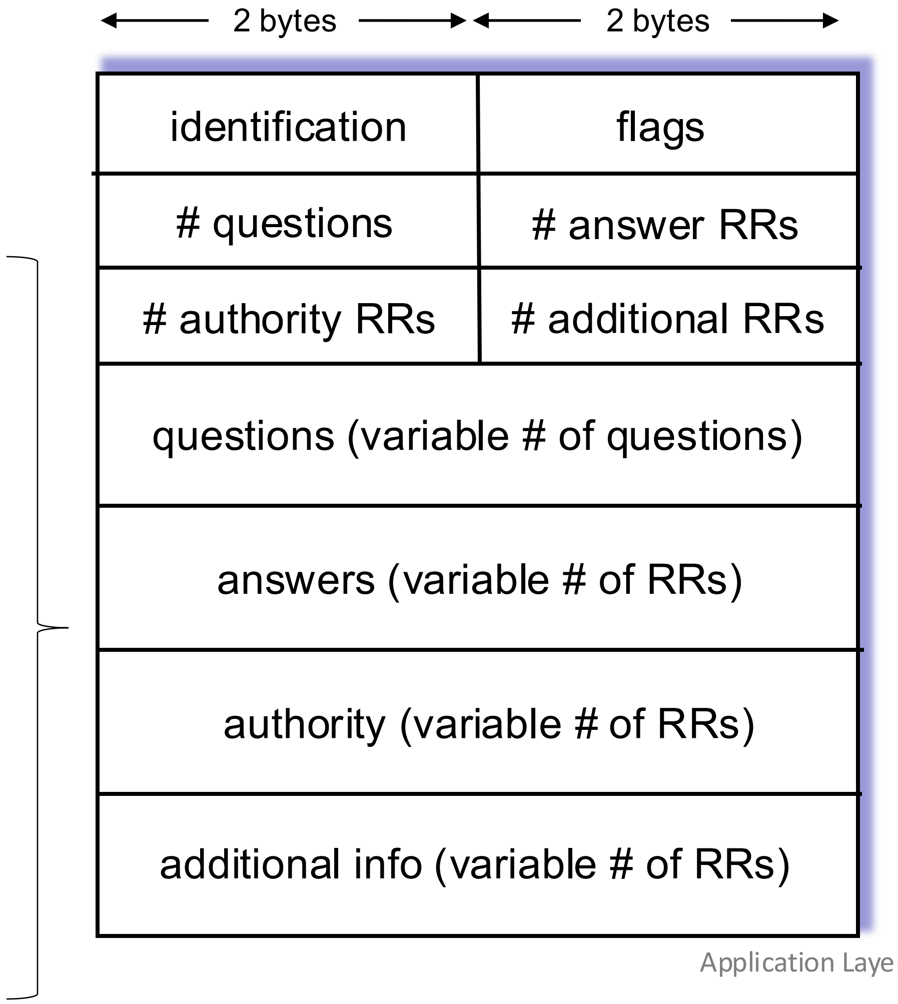
DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

dig google.com



```
⌄ Queries
  ⌄ google.com: type A, class IN
      Name: google.com
      [Name Length: 10]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
⌄ Answers
  ⌄ google.com: type A, class IN, addr 142.250.194.142
      Name: google.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 227 (3 minutes, 47 seconds)
      Data length: 4
      Address: 142.250.194.142
```



2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

Caching DNS Information

- once (any) name server learns mapping, it *caches* mapping, and i*mmediately* returns a cached mapping in response to a query
    - caching improves response time
    - cache entries timeout (disappear) after some time (TTL)
    - TLD servers typically cached in local name servers
- cached entries may be *out-of-date*
    - if named host changes IP address, may not be known Internet-wide until all TTLs expire!
    - *best-effort name-to-address translation!*

Getting your info into the DNS

example: new startup “Network Utopia”

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

DNS observations

xyz. github.io Nameserver likely

↳ Managed by Github

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass

Spoofing attacks

- intercept DNS queries, returning bogus replies
  - DNS cache poisoning
  - RFC 4033: DNSSEC authentication services

Centralization of DNS

- Name servers hosted by third-party (e.g., cloudflare, amazon)
  - Why?
  - Single point of failure?