

# CS 352

# Name Resolution

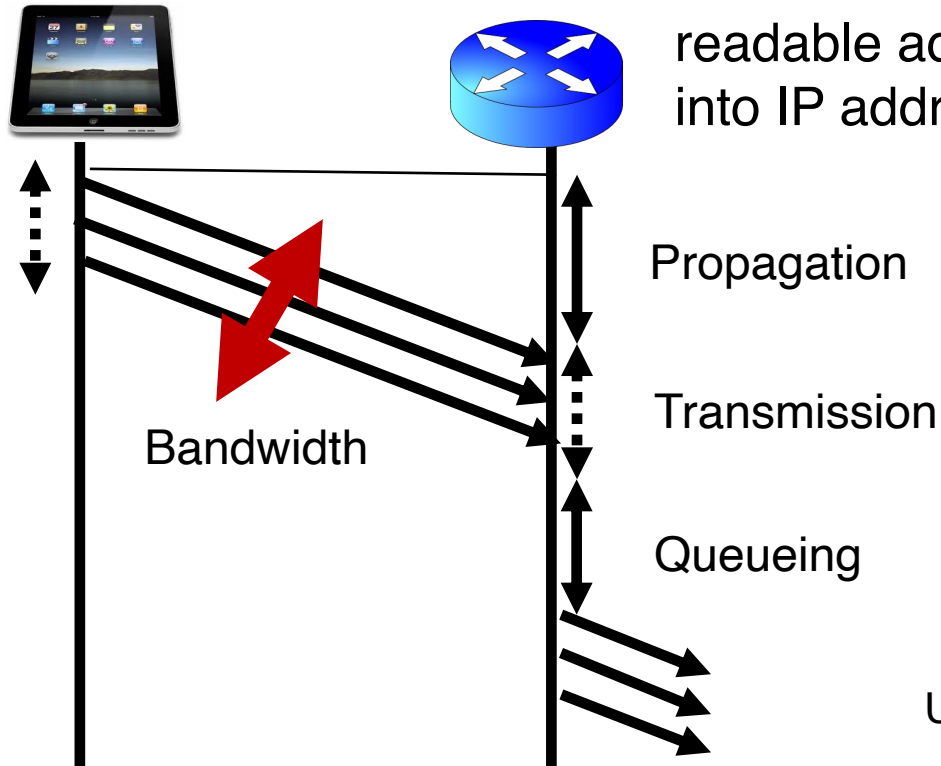
Lecture 4

<http://www.cs.rutgers.edu/~sn624/352-F22>

Srinivas Narayana

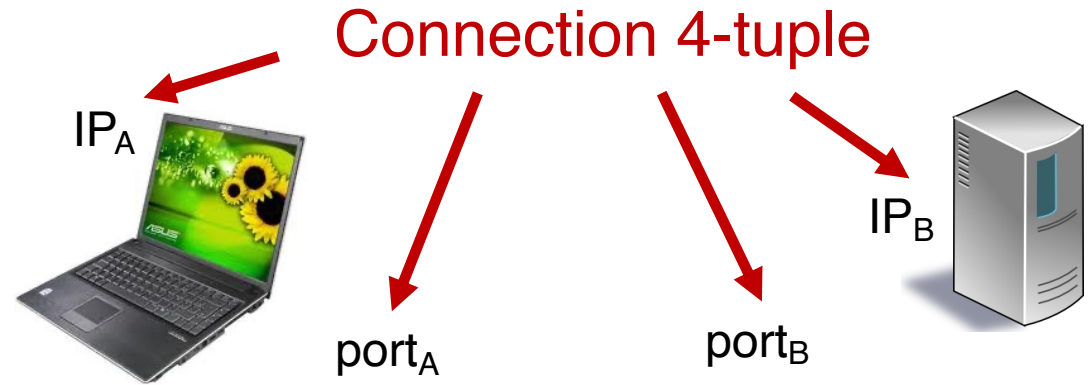
# Review of concepts

**DNS:** turn human-readable addresses into IP addresses.



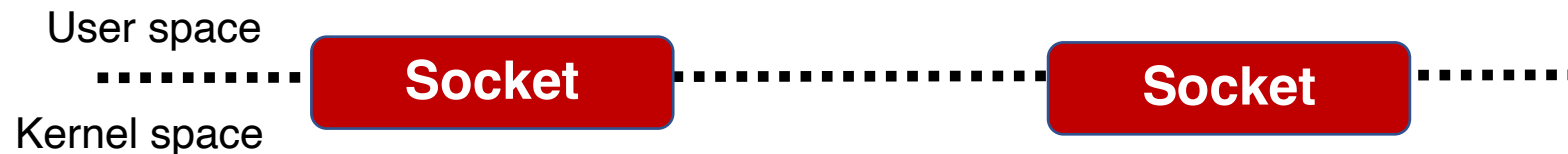
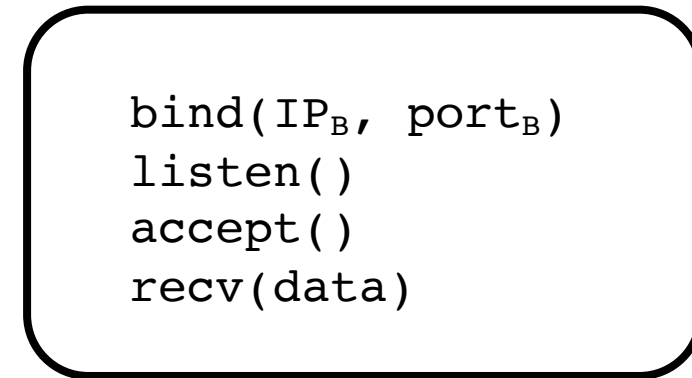
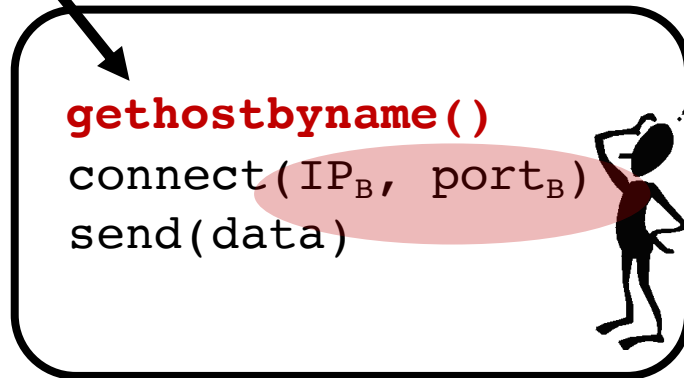
Client-server architecture

Peer to peer architecture



Application process

Application process



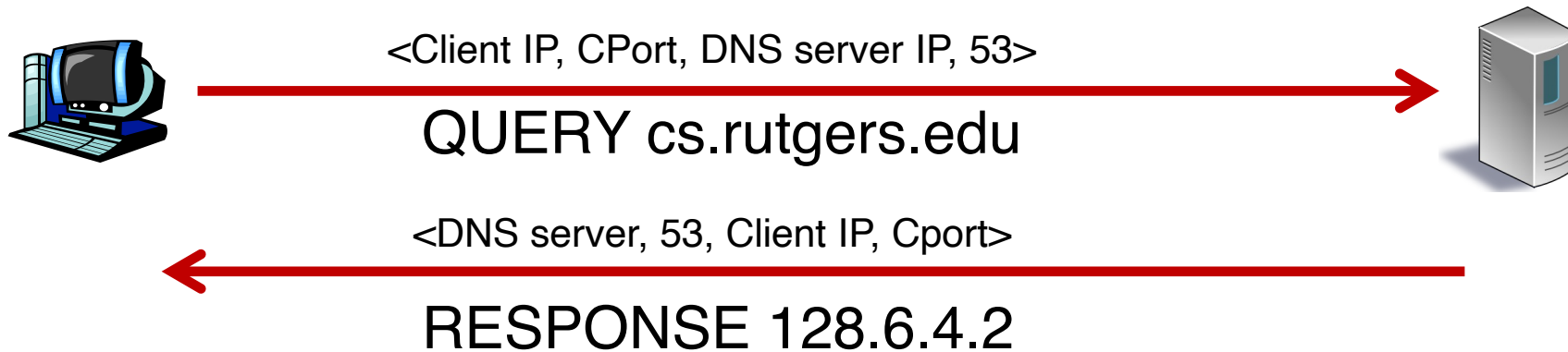
OS network stack layers

OS network stack layers



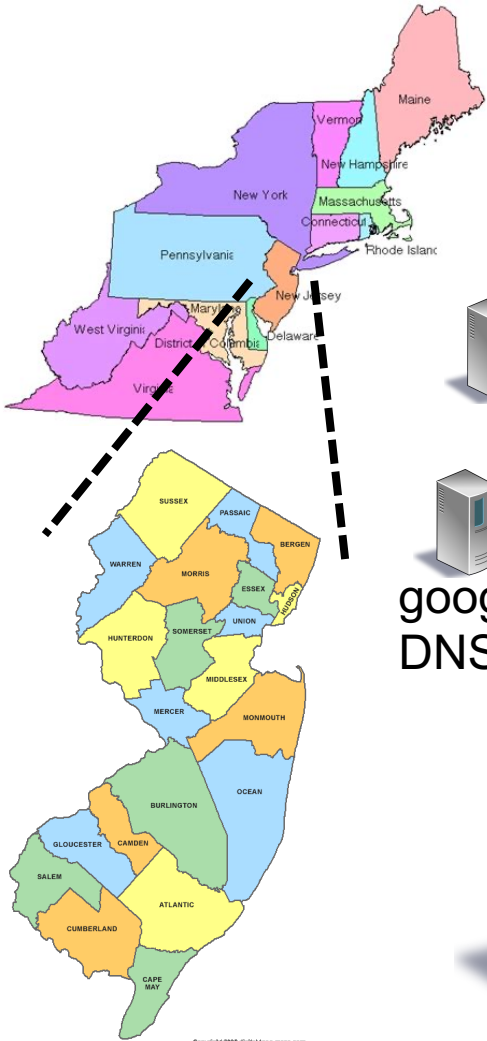
# Simple DNS

| DOMAIN NAME       | IP ADDRESS     |
|-------------------|----------------|
| spotify.com       | 98.138.253.109 |
| cs.rutgers.edu    | 128.6.4.2      |
| www.google.com    | 74.125.225.243 |
| www.princeton.edu | 128.112.132.86 |

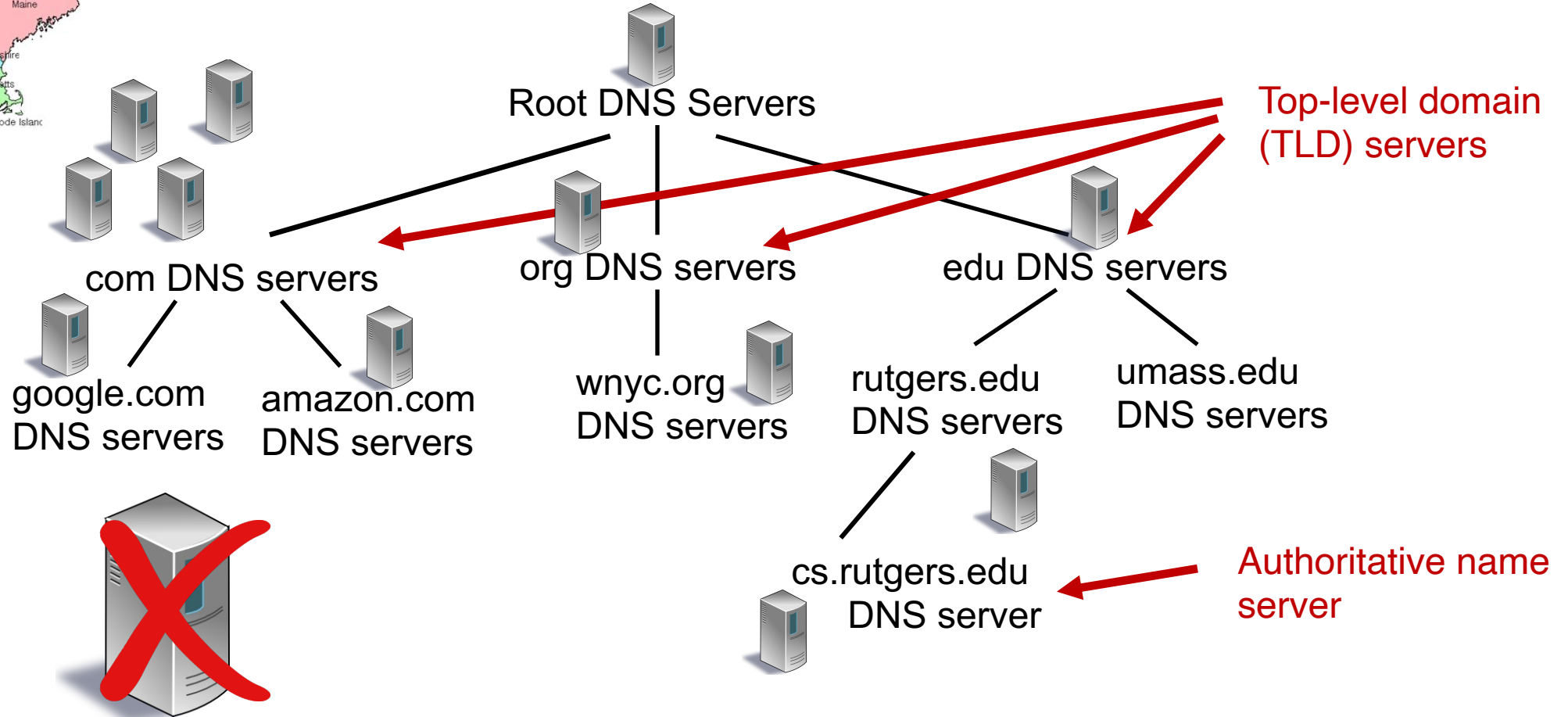


- Key idea: Implement a server that looks up a table.
- Will this scale?
  - Every new (changed) host needs to be (re)entered in this table
  - Performance: can the server serve billions of Internet users?
  - Failure: what if the server or the database crashes?
  - Security: What if someone “takes over” this server?

# Distributed and hierarchical database



RFC 1034



Hierarchy

Replication

# DNS Protocol

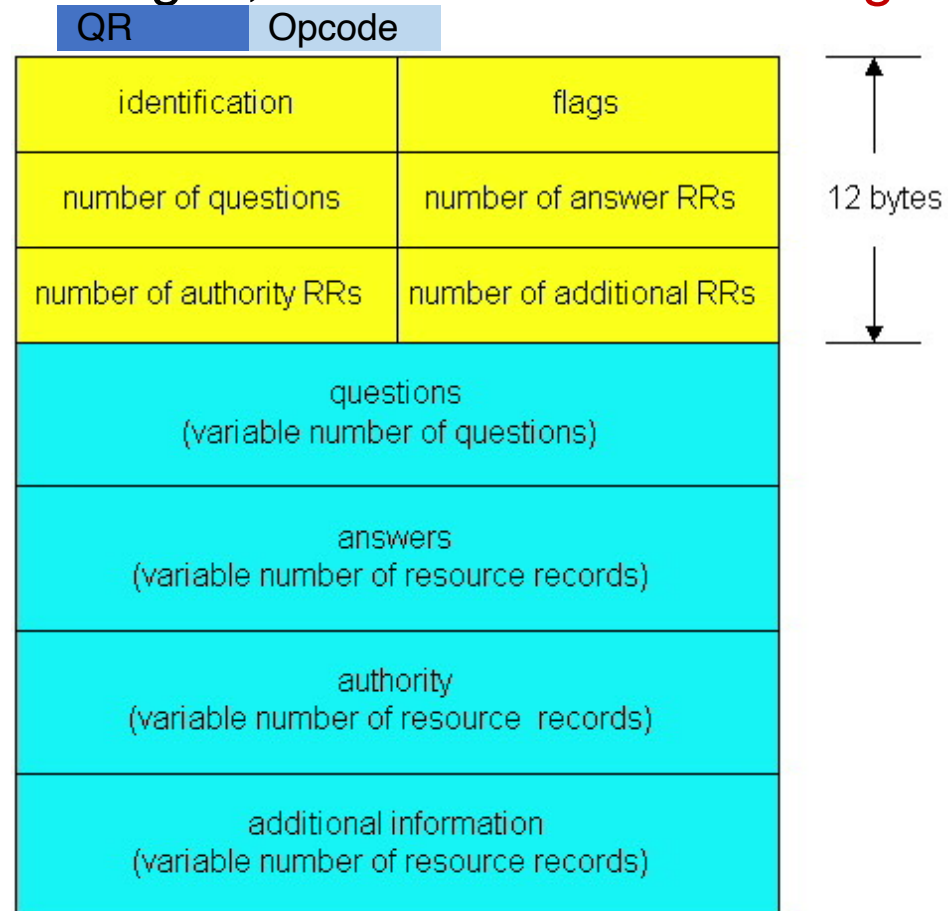
- Client-server application
- Client connects to (known) port 53 on server
- For now, assume DNS server IP known
- Two types of messages
  - Queries
  - Responses
- Type of Query (OPCODE)
  - Standard query (0x0)
    - e.g., Request IP address for a given domain name
  - Updates (0x5)
    - Provide a binding of IP address to domain name
- Each type has a common message format that follows the header

# DNS protocol: Message format

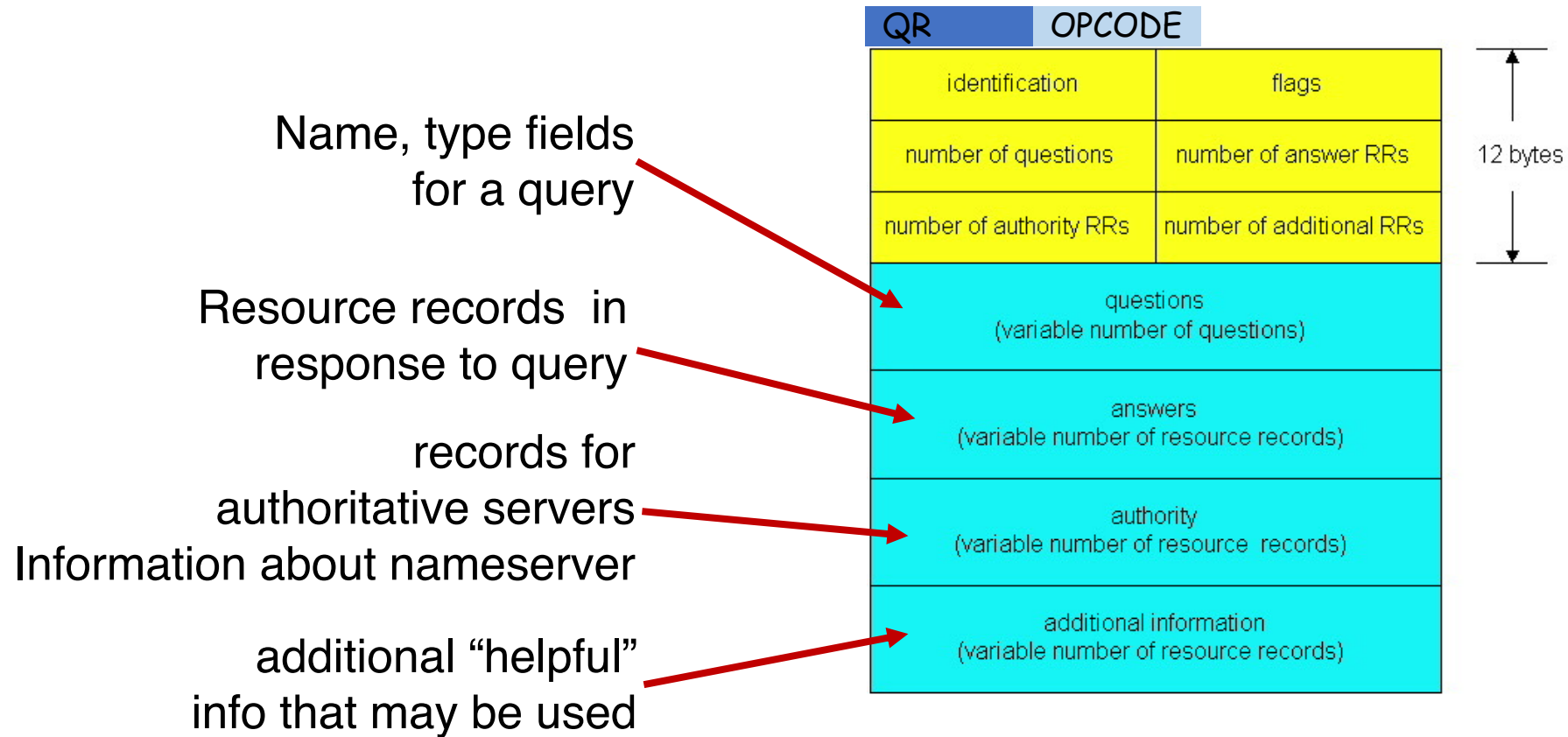
DNS protocol: query and reply messages, both with same message format

## Message header

- QR = 0 for Query, 1 for response
- Opcode= 0 standard
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - Authoritative answer
  - recursion desired
  - recursion available
  - reply is authoritative



# DNS protocol: Message format



# DNS Protocol: Actions

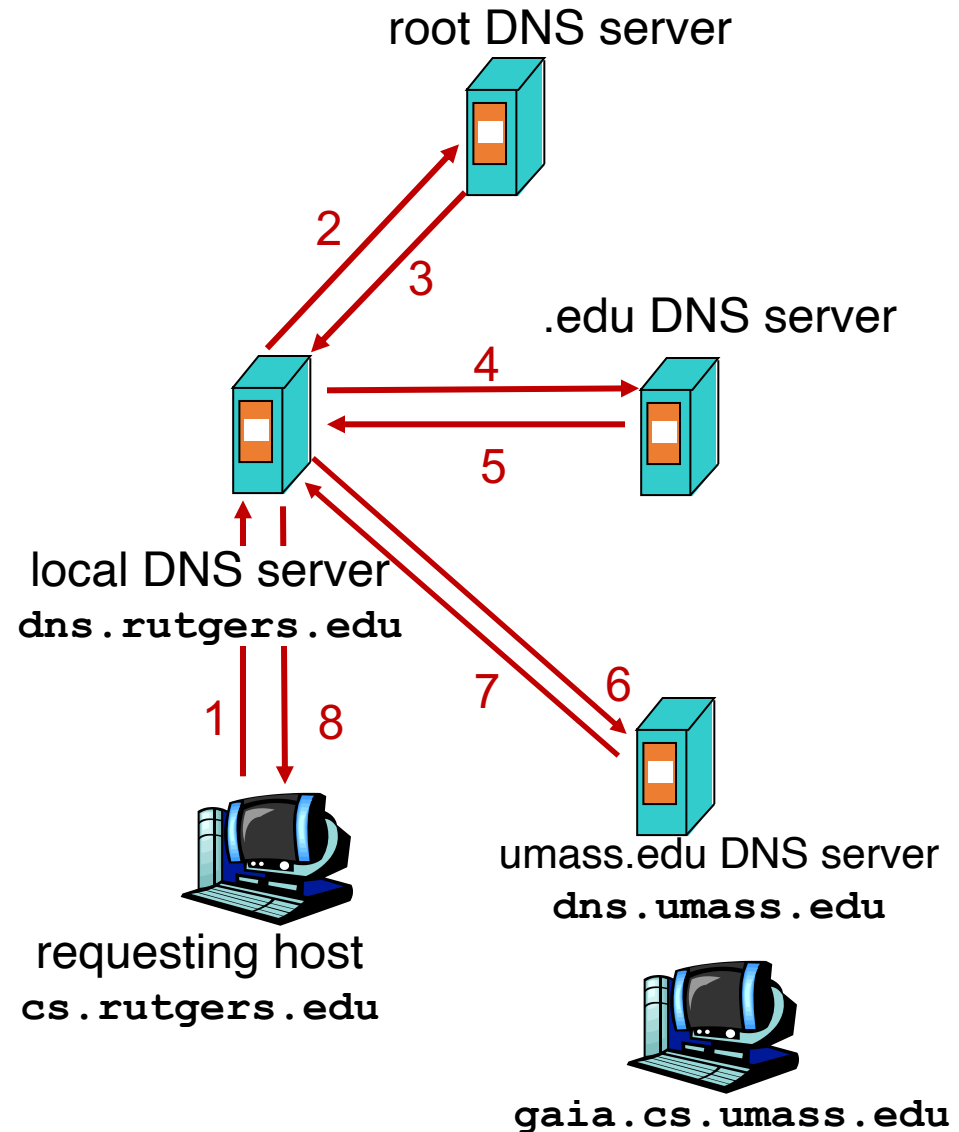
- When client wants to know an IP address for a host name
  - Client sends a DNS query to the “local” name server in its network
  - If name server contains the mapping, it returns the IP address to the client
  - Otherwise, the name server forwards the request to the root name server
  - The request works its way down the DNS hierarchy until it reaches a name server with a mapping for the requested name





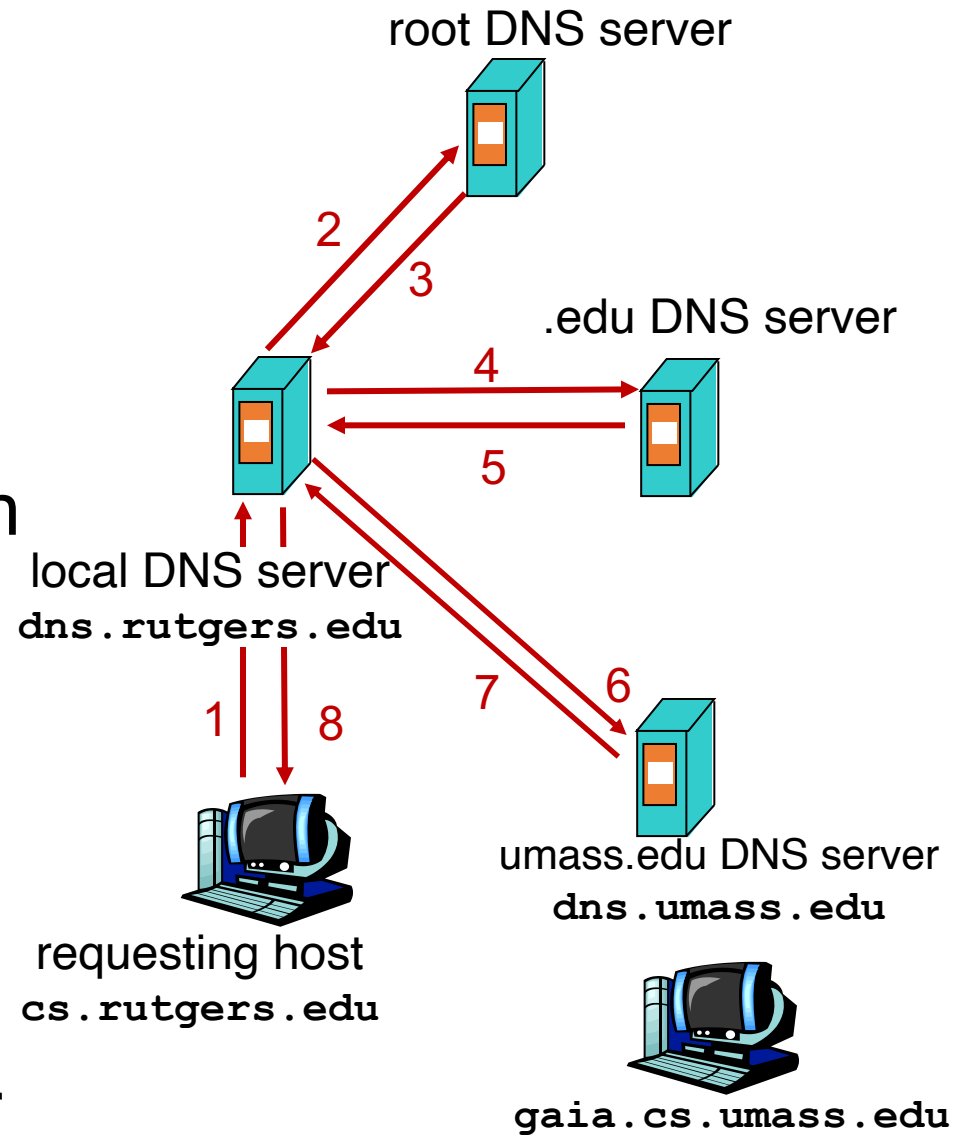
# Example

- Host at `cs.rutgers.edu` wants IP address for `gaia.cs.umass.edu`
- Local DNS server
- Root DNS server
- TLD DNS server
- **Authoritative** DNS server



# Query type

- Iterative query
- Contacted server replies with name of server to contact
- “I don’t know this name, but ask this other server”
- Queries are iterative from POV of the local DNS server



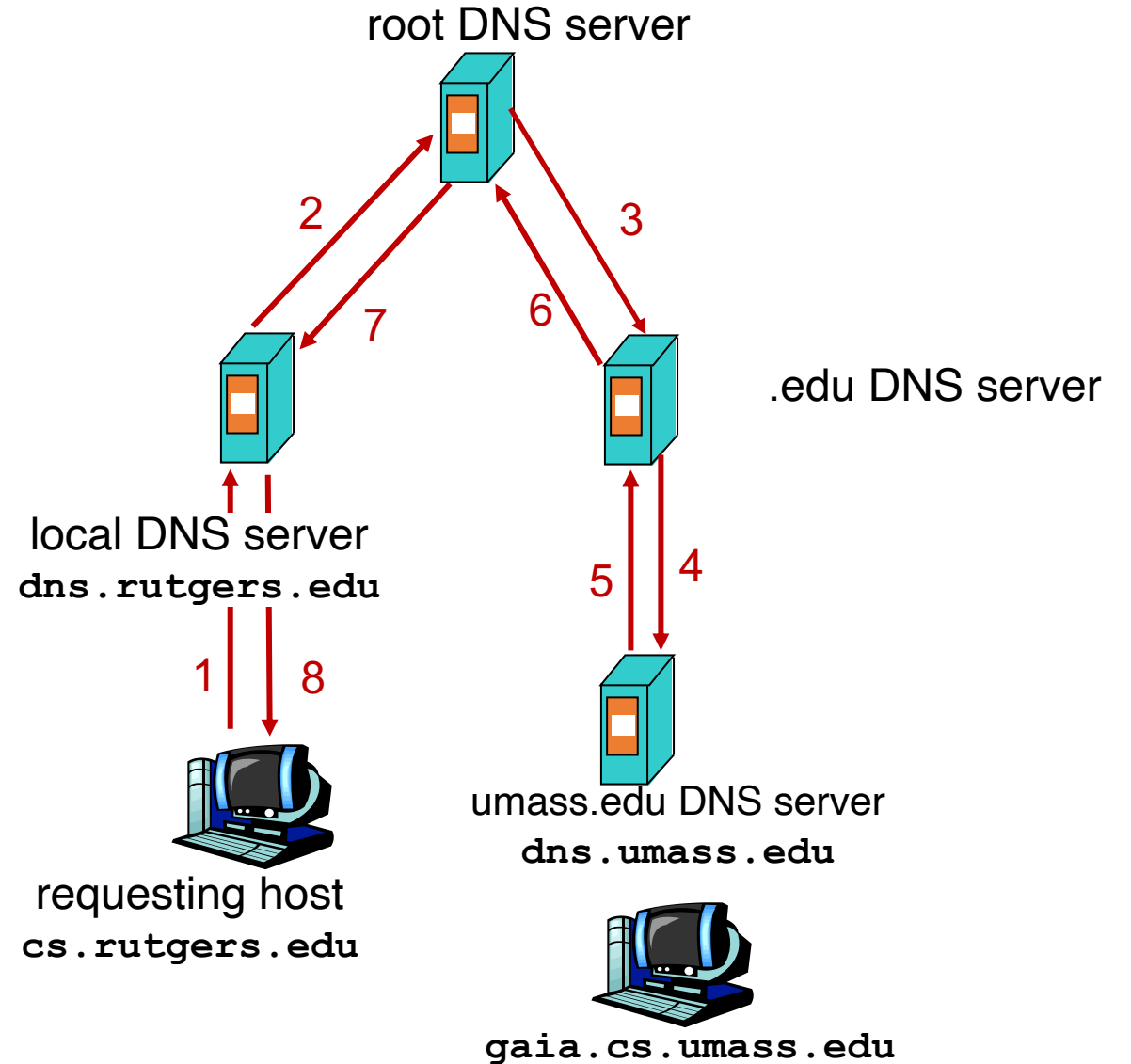
# Query type

## Recursive query:

- Puts burden of name resolution on the contacted (e.g., root) name server
- Query to root DNS server is recursive from POV of local

Problem: think about load on the root DNS server.

- Must it answer every DNS query?



# DNS caching

- Once (any) name server learns a name to IP address mapping, it *caches* the mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
  - In practice, root name servers aren't visited often!
- **Caching is pervasive in DNS**

# DNS in action

- `dig <domain-name>`
- `dig +trace <domain-name>`
- `dig @<dns-server> <domain-name>`
  
- Don't just watch; try it!

# Bootstrapping DNS

- How does a host contact the name server if all it has is the domain name and no (name server) IP address?
- IP address of at least 1 nameserver (usually, a local resolver) must be known a priori
- The name server may be bootstrapped “statically”, e.g.,
  - File `/etc/resolv.conf` in unix
  - Start -> settings-> control panel-> network ->TCP/IP -> properties in windows
- ... or with another protocol!
  - **DHCP**: Dynamic Host Configuration Protocol (more on this later)

# DNS may seem “basic”, low level, but ...

## *Gone in Minutes, Out for Hours: Outage Shakes Facebook*

Akamai DNS outage knocks many major websites and services offline: PSN, Steam, Fidelity, more [U]

### **Overloaded Azure DNS Servers to Blame For Microsoft Outage**

April 5, 2021

POSTED ON OCTOBER 5, 2021 TO [NETWORKING & TRAFFIC](#)

## More details about the October 4 outage

# DNS Resource Records



# DNS is a distributed database

- DNS stores **resource records (RRs)**
- (Incomplete) message format for each resource record (RR):
  - Class, type, name, value, TTL
- You can read all the gory details of the message format at <https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml>

# DNS records

## Type=A

- ❖ **name** is hostname
- ❖ **value** is IPv4 address

## Type=AAAA

- ❖ **name** is hostname
- ❖ **value** is IPv6 address

- Type=NS

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

## Type=CNAME

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

## Type=MX

- ❖ **value** is name of mailserver associated with **name**

# DNS record example

RRs in response  
to query



| NAME    | Design.cs.rutgers.edu |
|---------|-----------------------|
| TYPE    | A                     |
| CLASS   | IN                    |
| TTL     | 1 day(86400)          |
| ADDRESS | 192.26.92.30          |

records for  
authoritative  
servers  
Information about  
nameserver



| NAME    | Cs.rutgers.edu      |
|---------|---------------------|
| TYPE    | NS                  |
| CLASS   | IN                  |
| TTL     | 1 day(86400)        |
| NSDNAME | Ns-lcsr.rutgers.edu |

DNS serves as a general repository of information for the Internet!

# DNS record types

- `dig -t <type> <domain-name>`

# Summary of DNS

- Hostname to IP address translation via a global network of servers
- Embodies several scaling principles
  - Partition through a hierarchy to silo query load
  - Replication to scale out at each level of hierarchy
  - Caching to reduce query load
- Once you have a reliable DB, can implement many useful things on top!
- Example 1: Scaling large web services, e.g., google search, by redirecting different clients to different servers (IP addresses)
  - Reliability, load balancing, performance optimization
- Example 2: Associating certificates, keys (security info) with domain names
  - <https://www.rfc-editor.org/rfc/rfc8162.html>
  - <https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/00/>