The Application Layer: Sockets, DNS

CS 352, Lecture 3

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

App-layer protocol

- Types of messages exchanged,
  - e.g., request, response
- Message format:
  - Syntax :what fields in messages & how fields are delineated
  - Semantics: meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

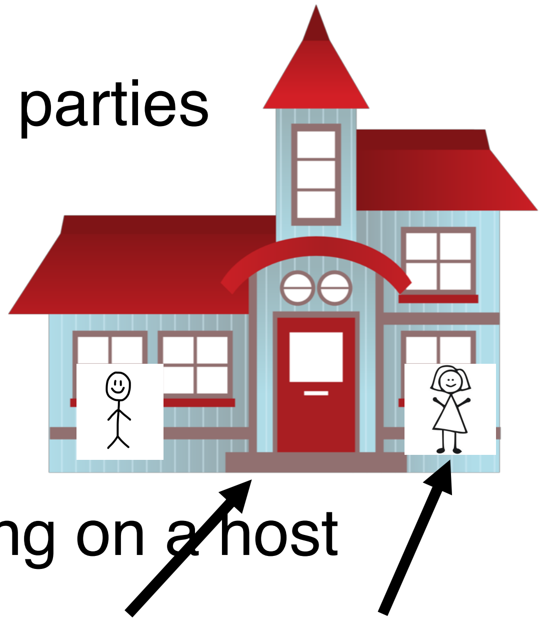
- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:

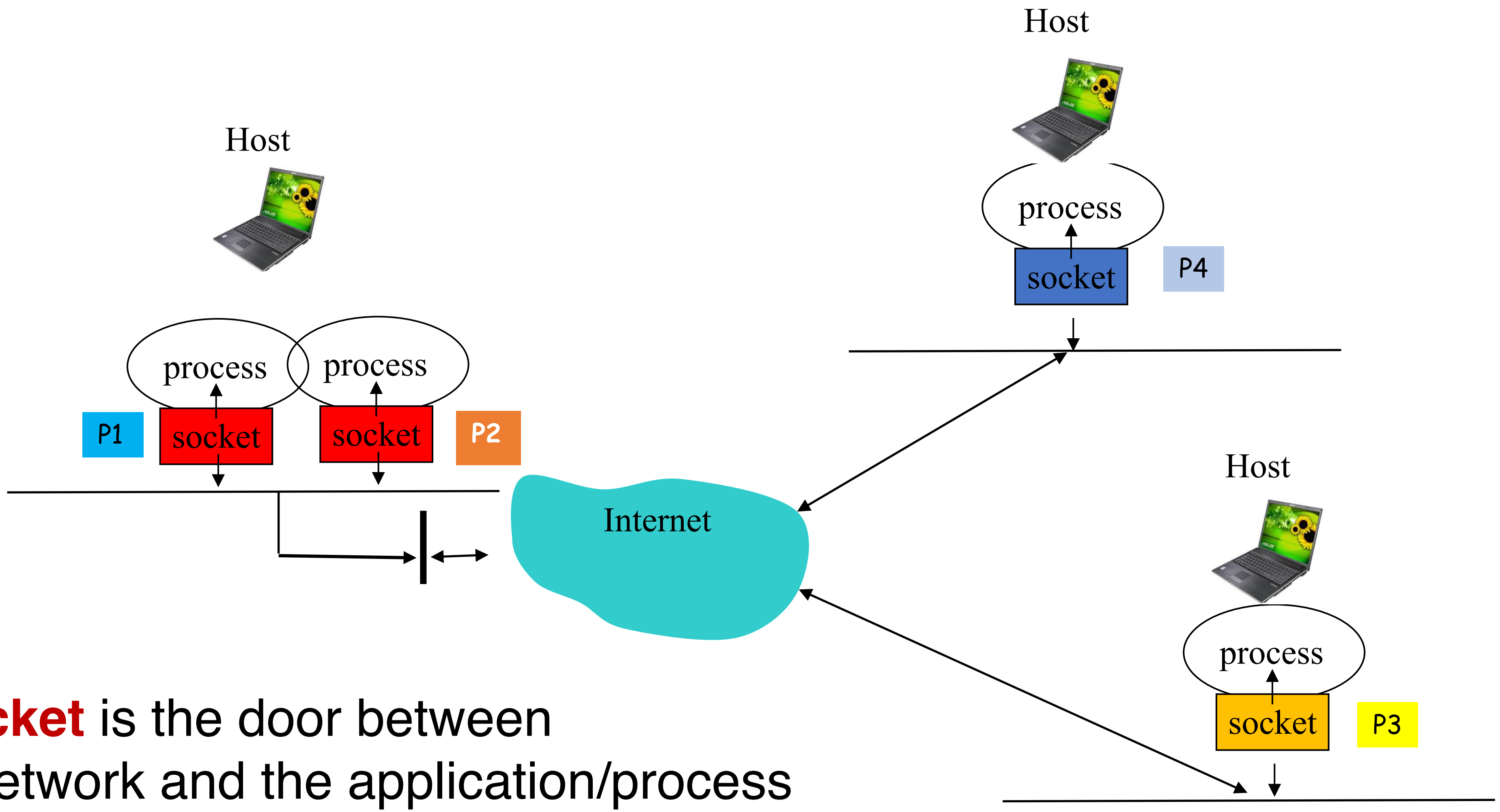
- e.g., Skype, Microsoft Exchange

Application “addresses”

- We usually think of an application executing on a single host.
- However, applications can reside on, say, 2 different hosts connected by a network
- In order to communicate, need to identify the communicating parties
- Telephone network: phone number (10 digits)
- Computer network: IP address
  - IPv4 (32 bits) 128.6.24.78
  - IPv6 (128 bits) 2001:4000:A000:C000:6000:B001:412A:8000
- Suppose there is more than one networked program executing on a host
  - In addition to host address, we need one more address
  - “Which Program to talk to?”
- Another identity for an application: port #

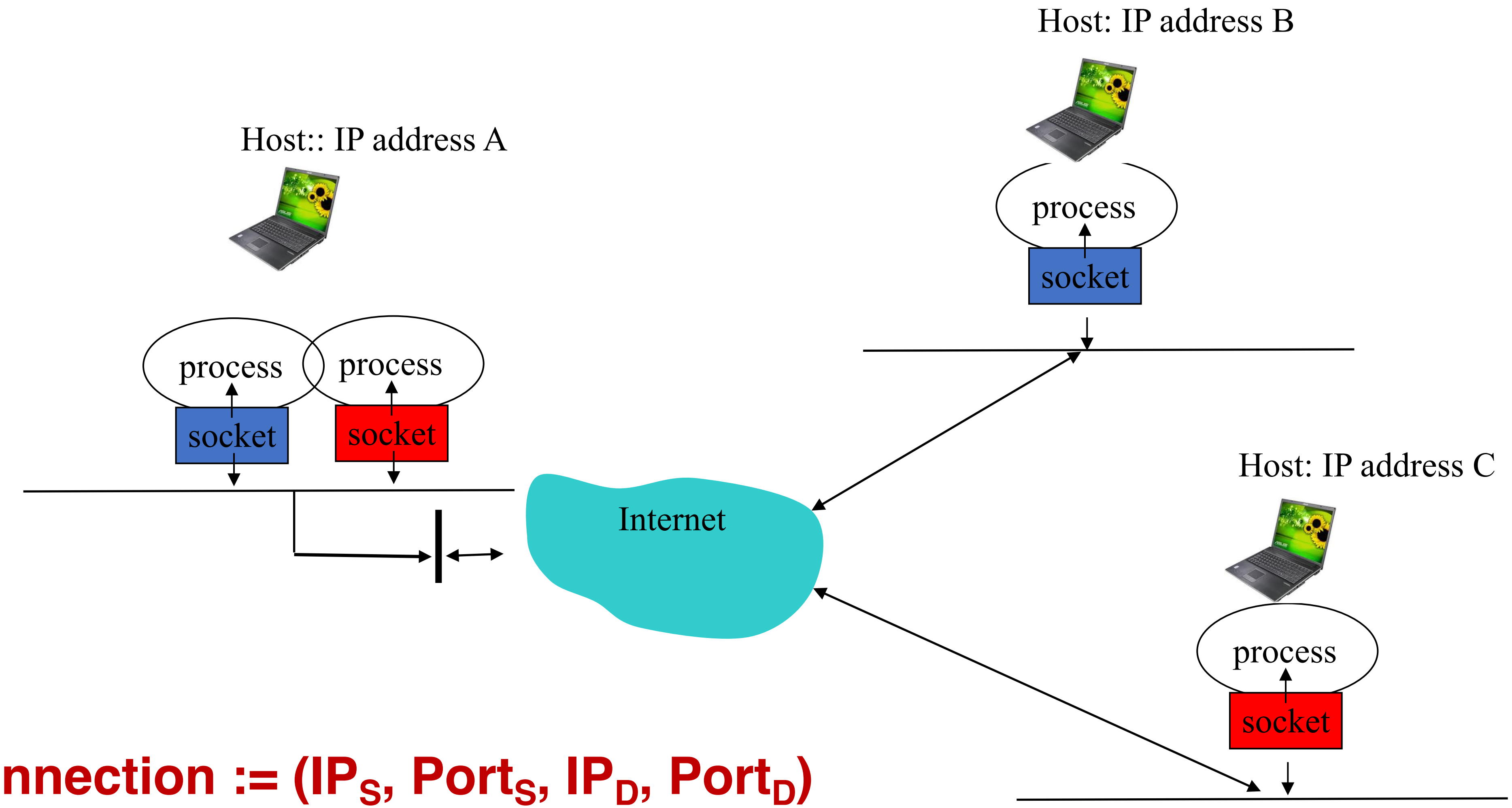


IP address & port number



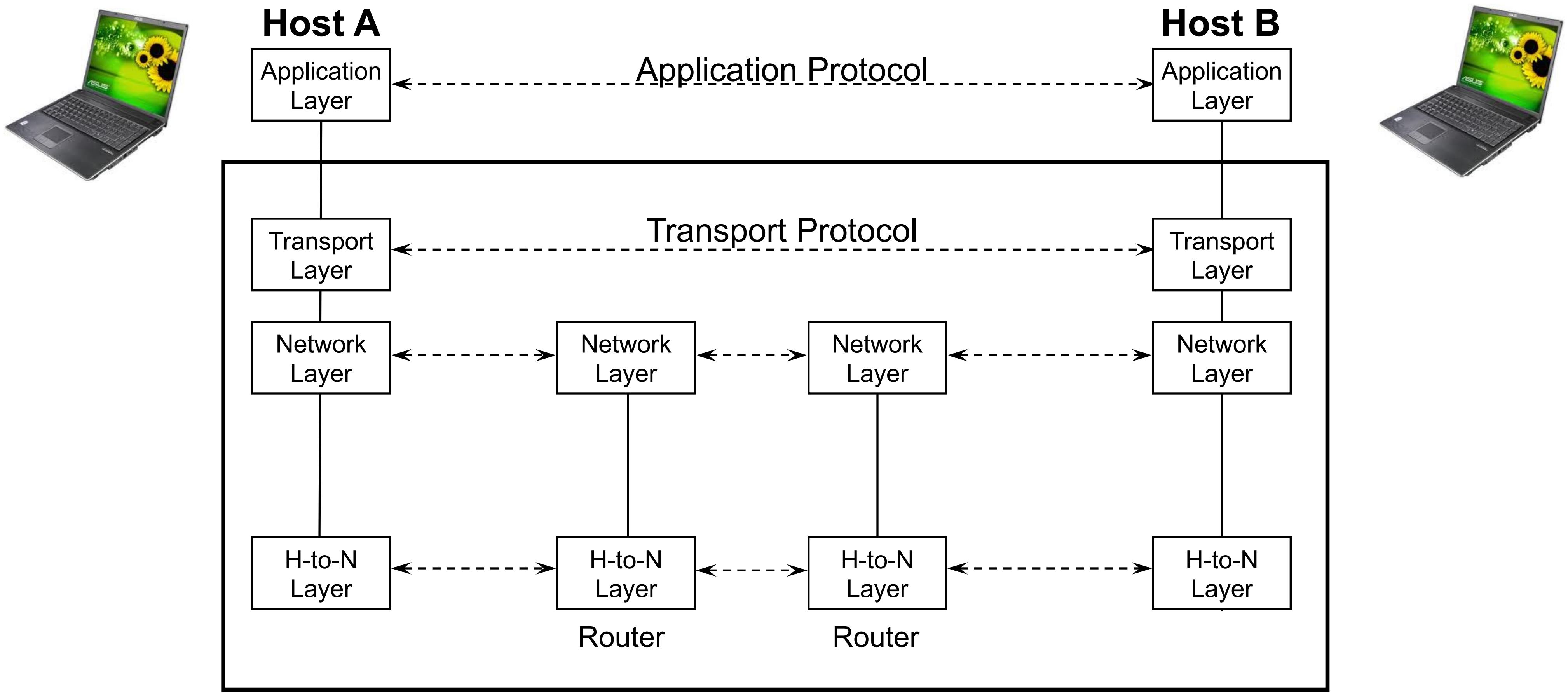
A **socket** is the door between OS/network and the application/process

A network connection is a 4-tuple

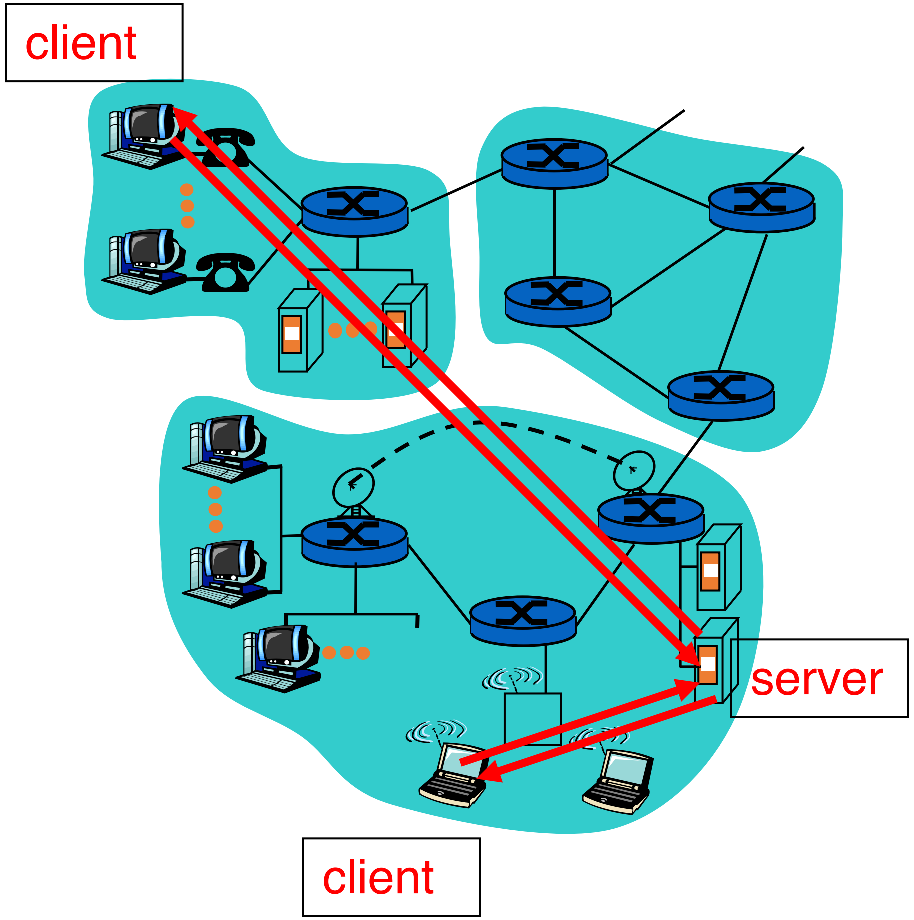


Connection := (IP_S, $Port_S$, IP_D, $Port_D$)

Recall: Services provided by lower layers



Client-server architecture



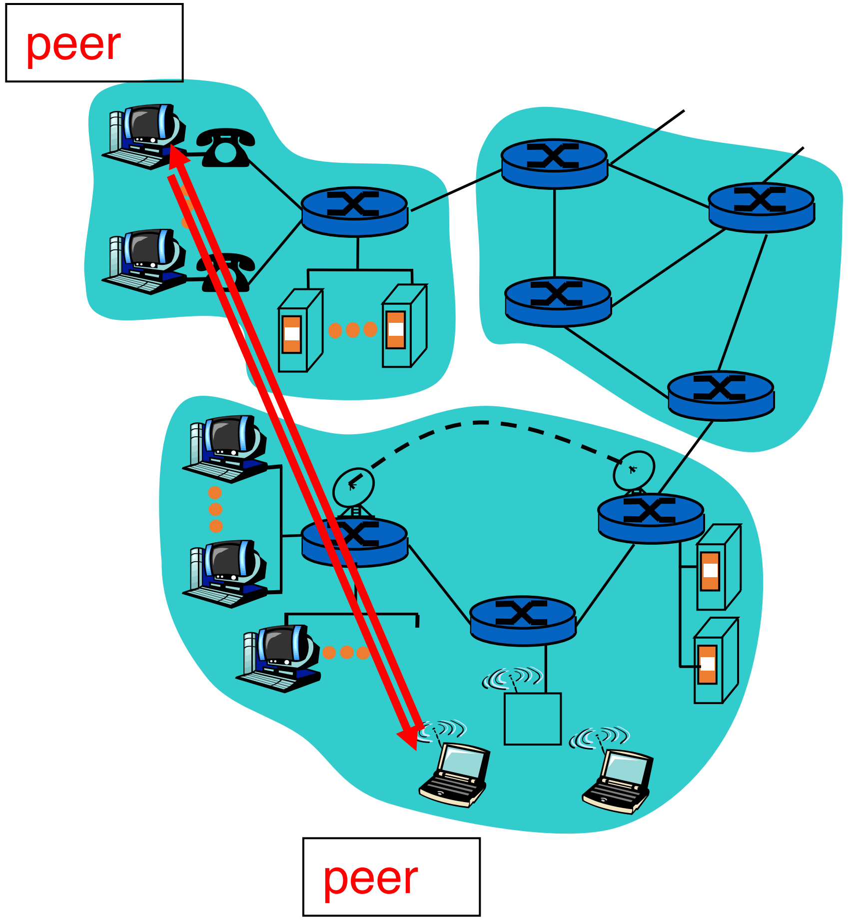
Server:

- always-on host
- permanent IP address
- server farms ("data centers") for scaling

Clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Peer-to-peer (P2P) architecture



- Peers:
  - Intermittently connected hosts
  - Directly talking to each other
- Little to no reliance on always-up servers
  - Examples: BitTorrent, Skype
- Today, many applications use a hybrid model
  - Example: Skype "supernodes"

Domain Name System (DNS)

“You have my name. Can you lookup my number?”

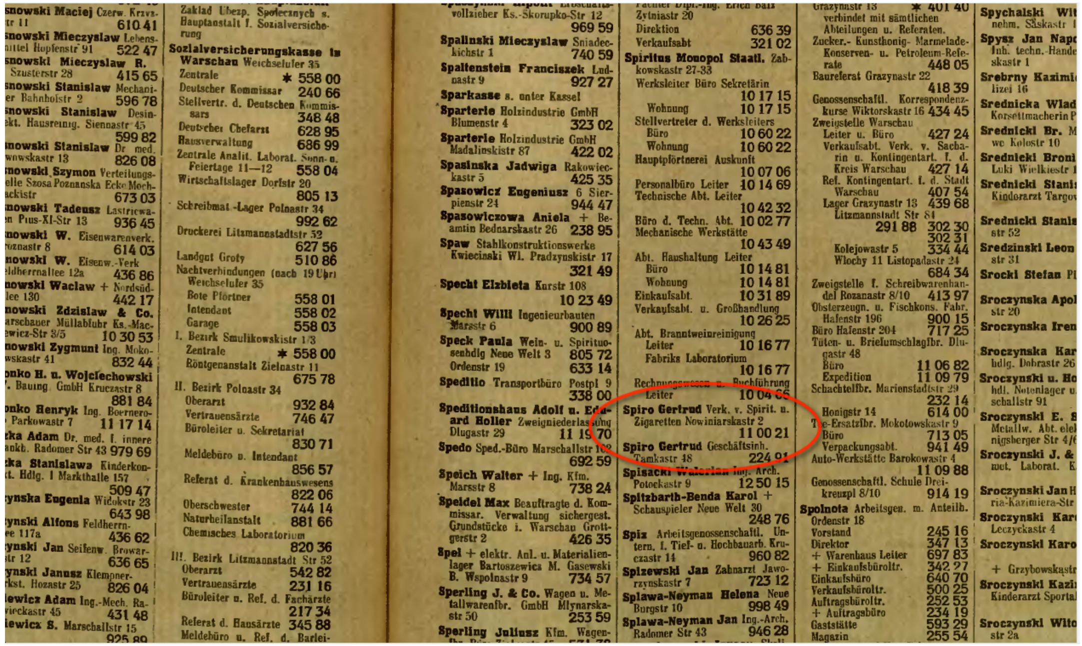
Domain Name System (DNS)

- Problem statement:
  - Average brain can easily remember 7 digits for a few names
  - On average, IP addresses have 12 digits
  - We need an easier way to remember IP addresses

- Solution:
  - Use alphanumeric names to refer to hosts
  - Just as a contacts or telephone directory (white pages)
  - Add a service (called DNS) to map between alphanumeric host names and binary IP addresses
  - We call this **Address Resolution**

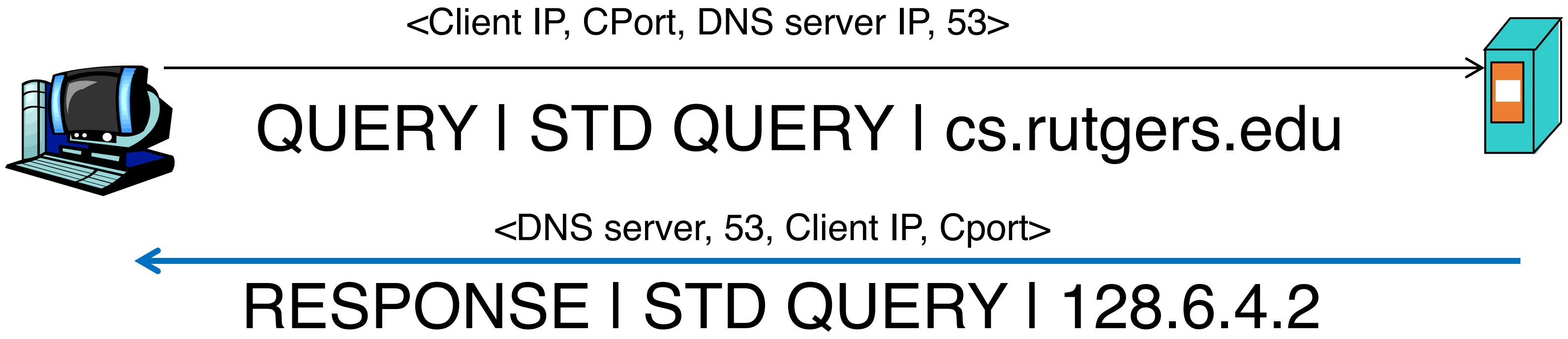
Simple DNS

- Idea (1): What if every host has a local directory?
- `/etc/hosts.txt`
  - How things worked in the early days of the Internet!
- What if hosts moved around? How do you keep this up to date?



Simple DNS

DOMAIN NAME	IP ADDRESS
www.yahoo.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



- Idea (2): Implement a server that looks up a table
  - Simple, but does not scale
- Every new host needs to be entered in this table
- Performance?
- Failure?

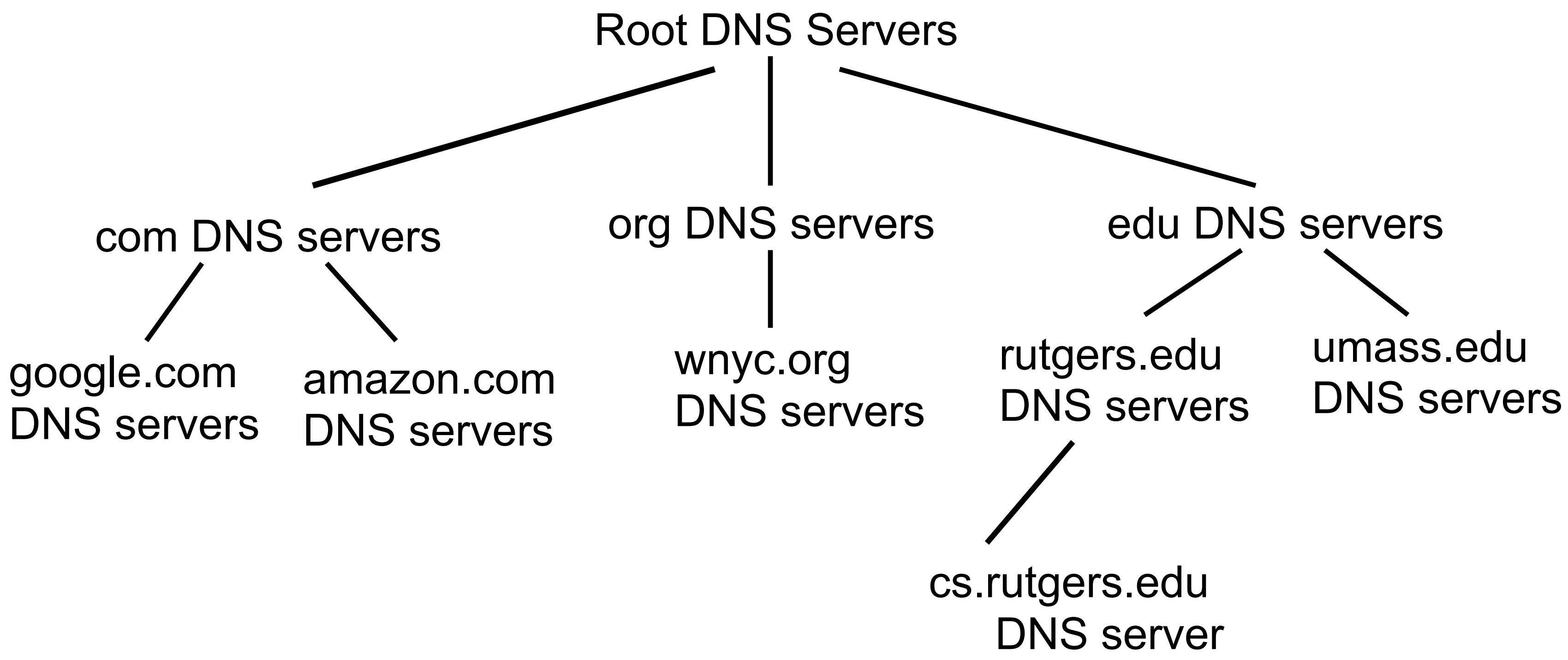
DNS design

A centralized DNS design (ex: single server) is problematic.

- single point of failure
- traffic volume
- distant centralized database
- security
- maintenance

It doesn’t *scale* to the requirements of the Internet.

Distributed and hierarchical database



RFC 1034

DNS protocol

- Client and Server
- Client connects to Port 53
- DNS server address should be known
    - Either manually configured or automatically (more on this to come…)
- Two types of messages
    - Queries
    - Responses
- Type of Query (OPCODE) methods
    - Standard query (0x0)
        - Request domain name for a given IP address
    - Updates (0x5)
        - Provide a binding of IP address to domain name
- Each type has a common message format that follows the header

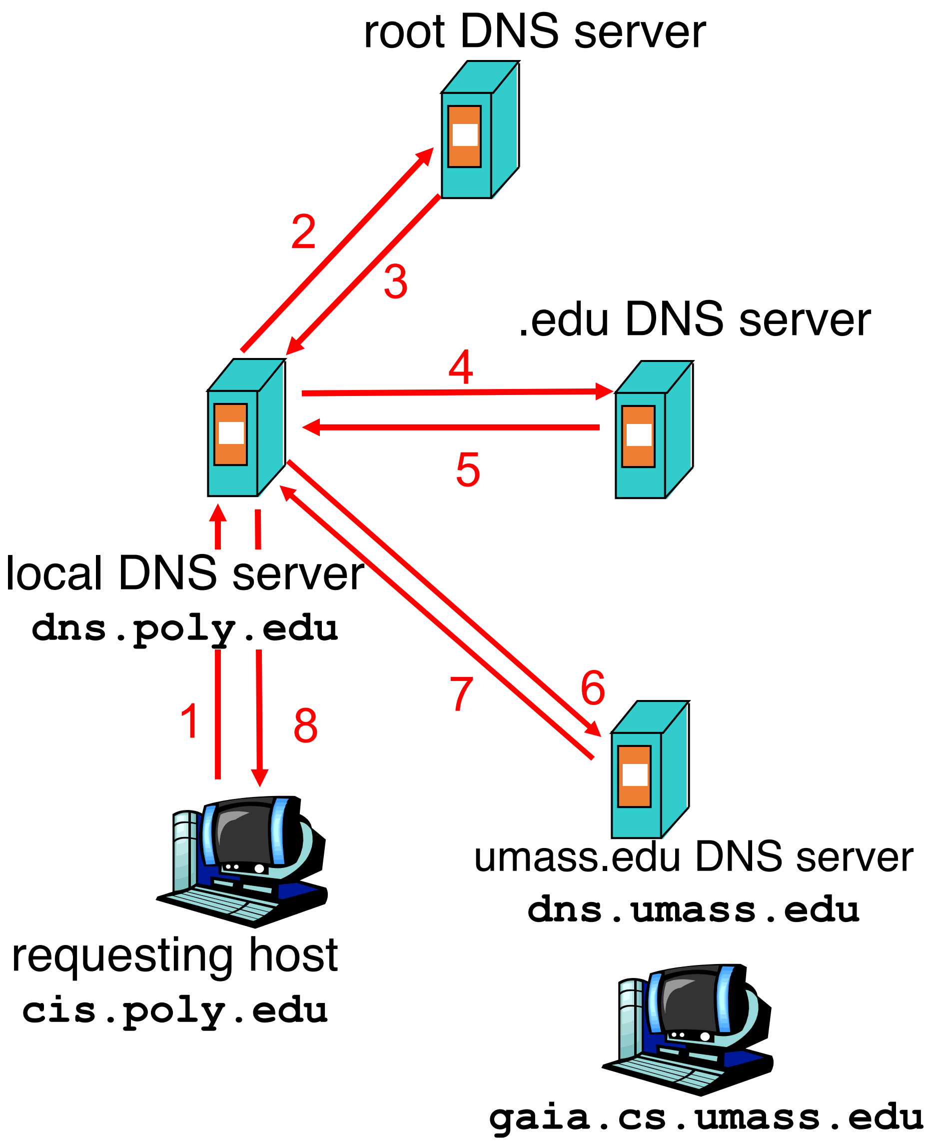
DNS Protocol

- 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 tree toward the host until it reaches a name server with the correct mapping

Example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

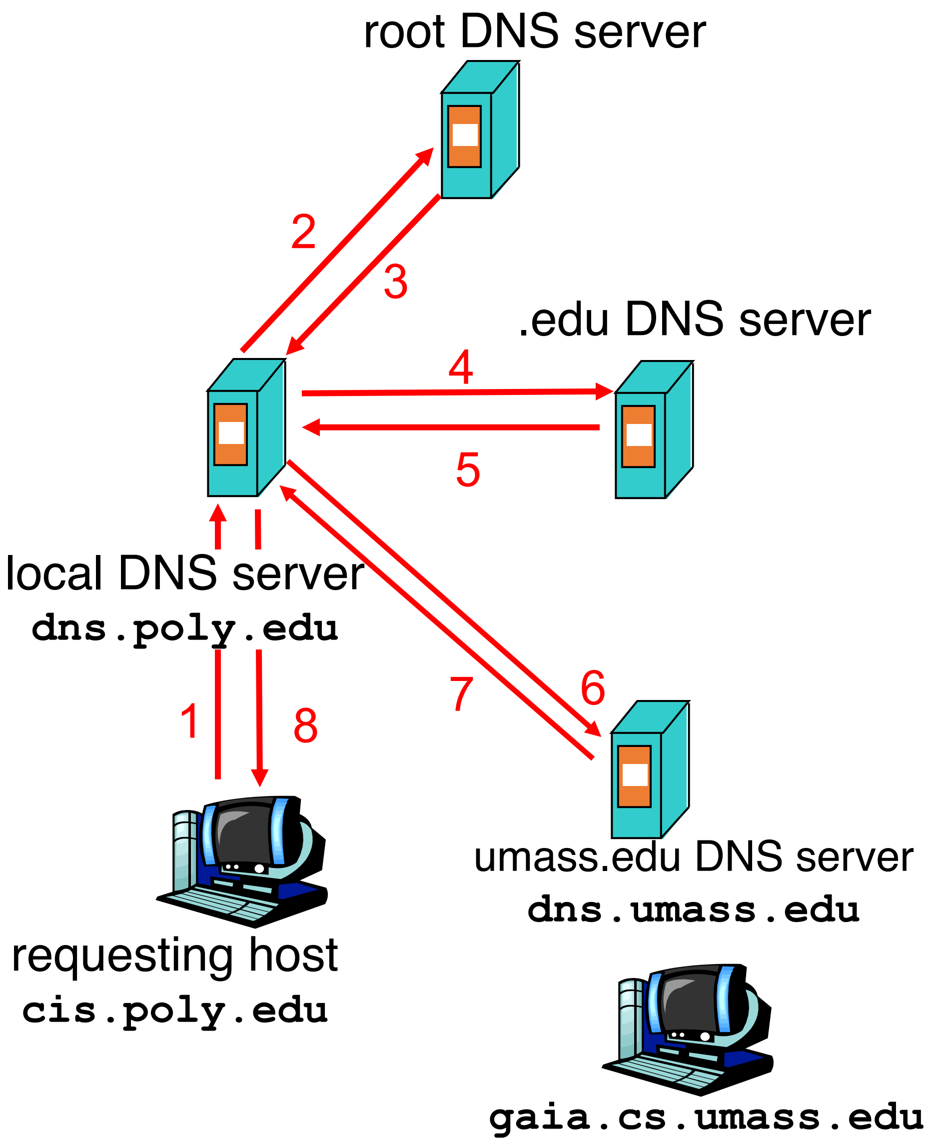
- Local DNS server
- Root DNS server
- TLD DNS server
- Authoritative DNS server



Query type

Iterated query:

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



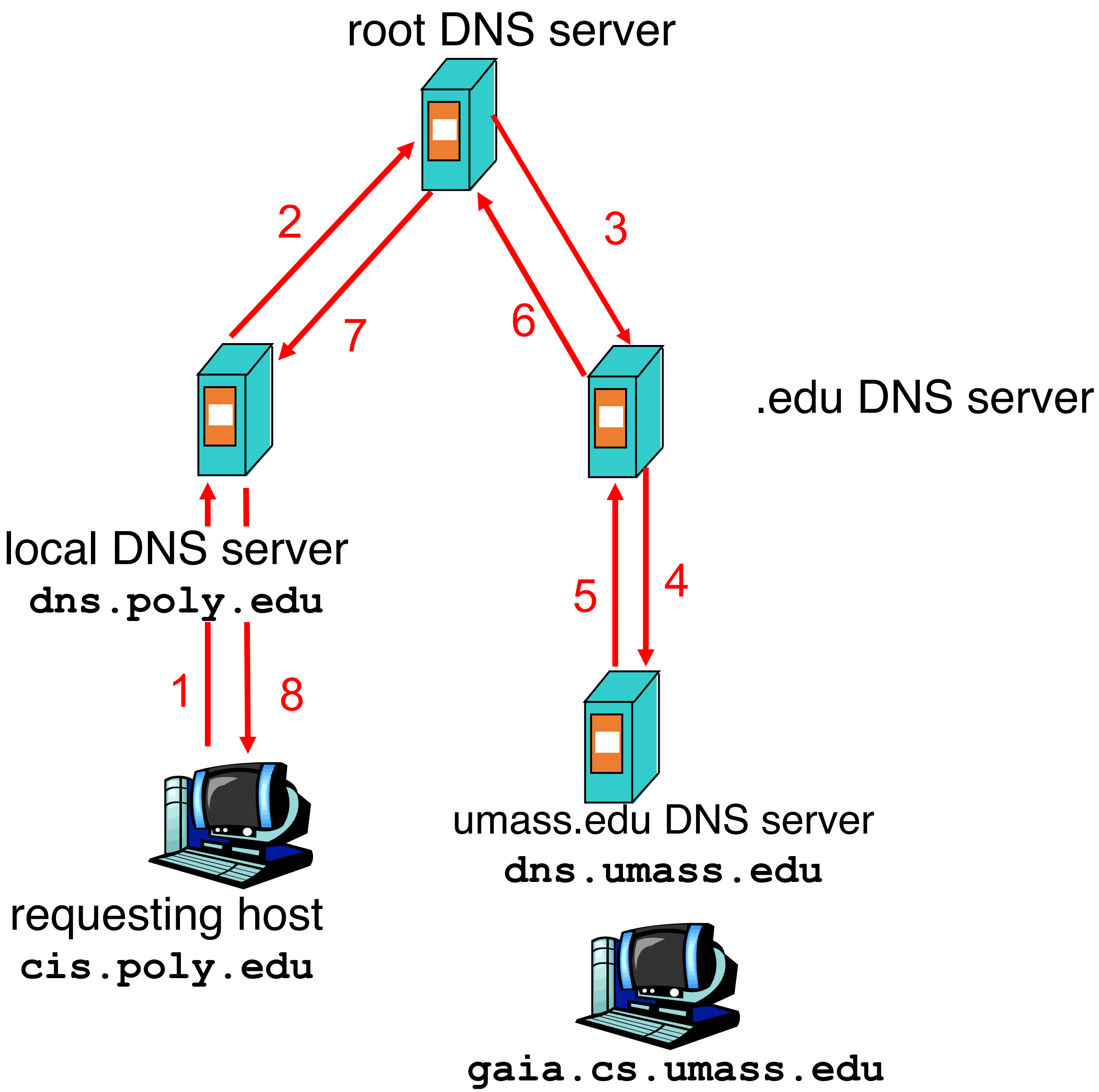
Query type

Recursive query:

- Puts burden of name resolution on the contacted name server

Problem: think about the root DNS server.

- Must it answer every DNS query?

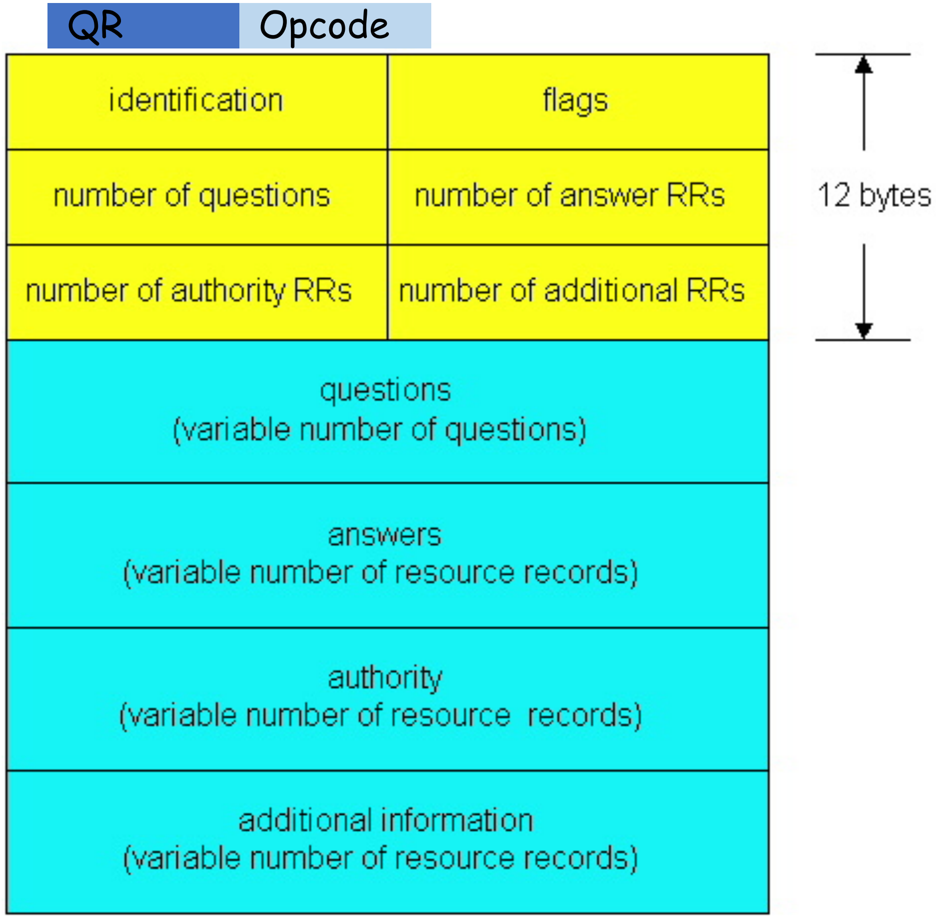


DNS caching and updating records

- 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

DNS protocol messages

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, messages

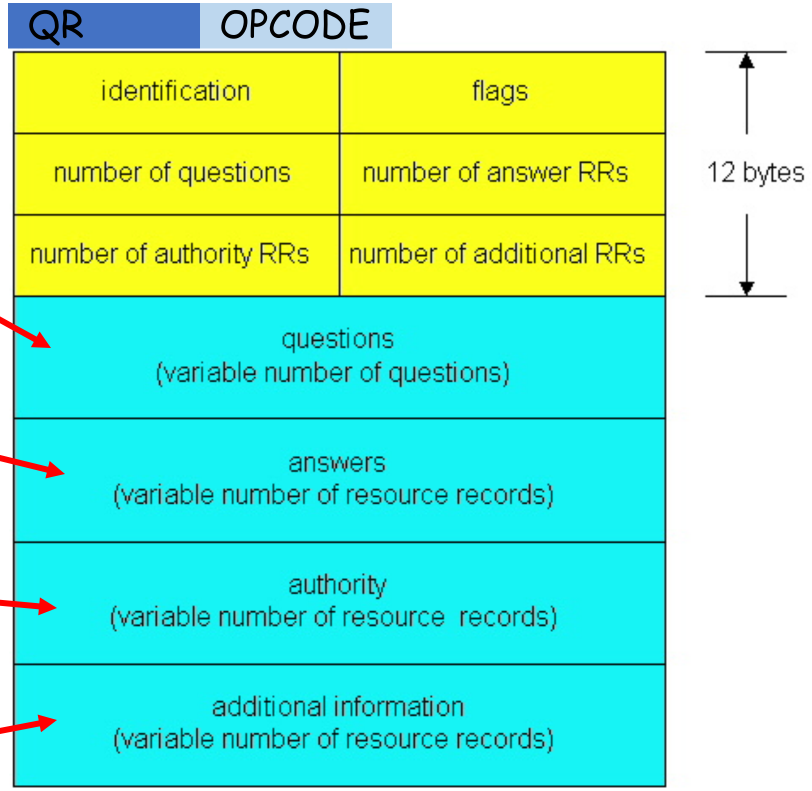
Name, type fields for a query

RRs in response to query

records for authoritative servers

Information about nameserver

additional "helpful" info that may be used



DNS records

DNS: distributed db storing resource records (RR)

RR format: **(name, type, class, ttl, addr)**

Type=A

- **name** is hostname
- **value** is IP 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

  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 summary

DNS service:

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

- Caching
- Hierarchical structure for scaling
- Multiple layers of indirection

Bootstrapping DNS

- How does a host contact the name server if all it has is the name and no IP address?
- IP address of at least 1 nameserver must be given a priori
    - or with another protocol (DHCP, bootp)

- File `/etc/resolv.conf` in unix
- `Start -> settings-> control panel-> network ->TCP/IP -> properties` in windows

Themes

- Request/response nature of these protocols
- How Messages are structured
    - HTTP, SMTP, FTP - simple ASCII protocols
- Caching
- Name Lookup
    - Division of concerns (e.g. zones)
    - Hierarchy structure