The Application Layer: Sockets, DNS

Lecture 3

http://www.cs.rutgers.edu/~sn624/352-S22

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Application-layer Protocol

- Message format:
  - Syntax: what fields in messages & how fields are delineated
  - Semantics: meaning of information in fields
- Actions: when and how processes send & respond to messages

Public-domain protocols:

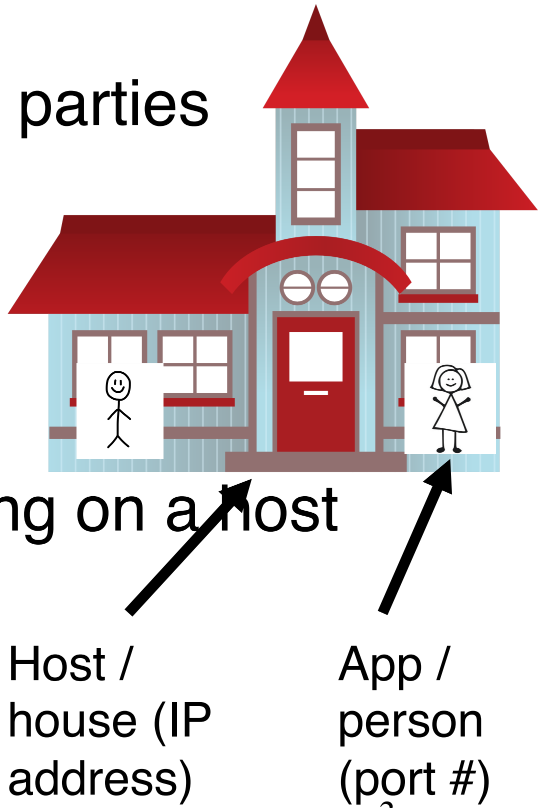
- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:

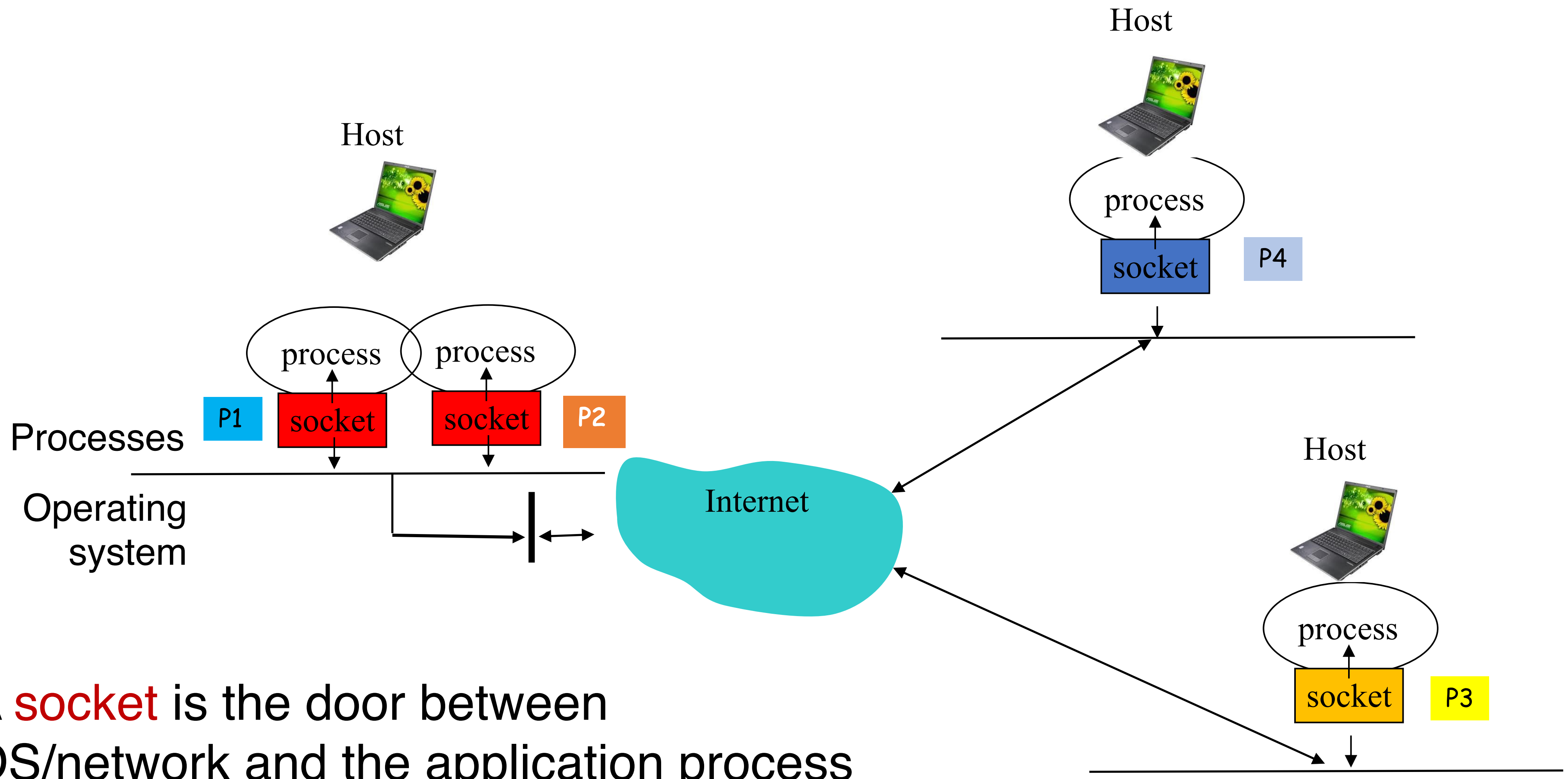
- e.g., Skype

Application Addresses

- We usually think of an application executing on a single endpoint
- However, applications can reside on, say, 2 different endpoints 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?”
- Identity for an application: port number + IP addr



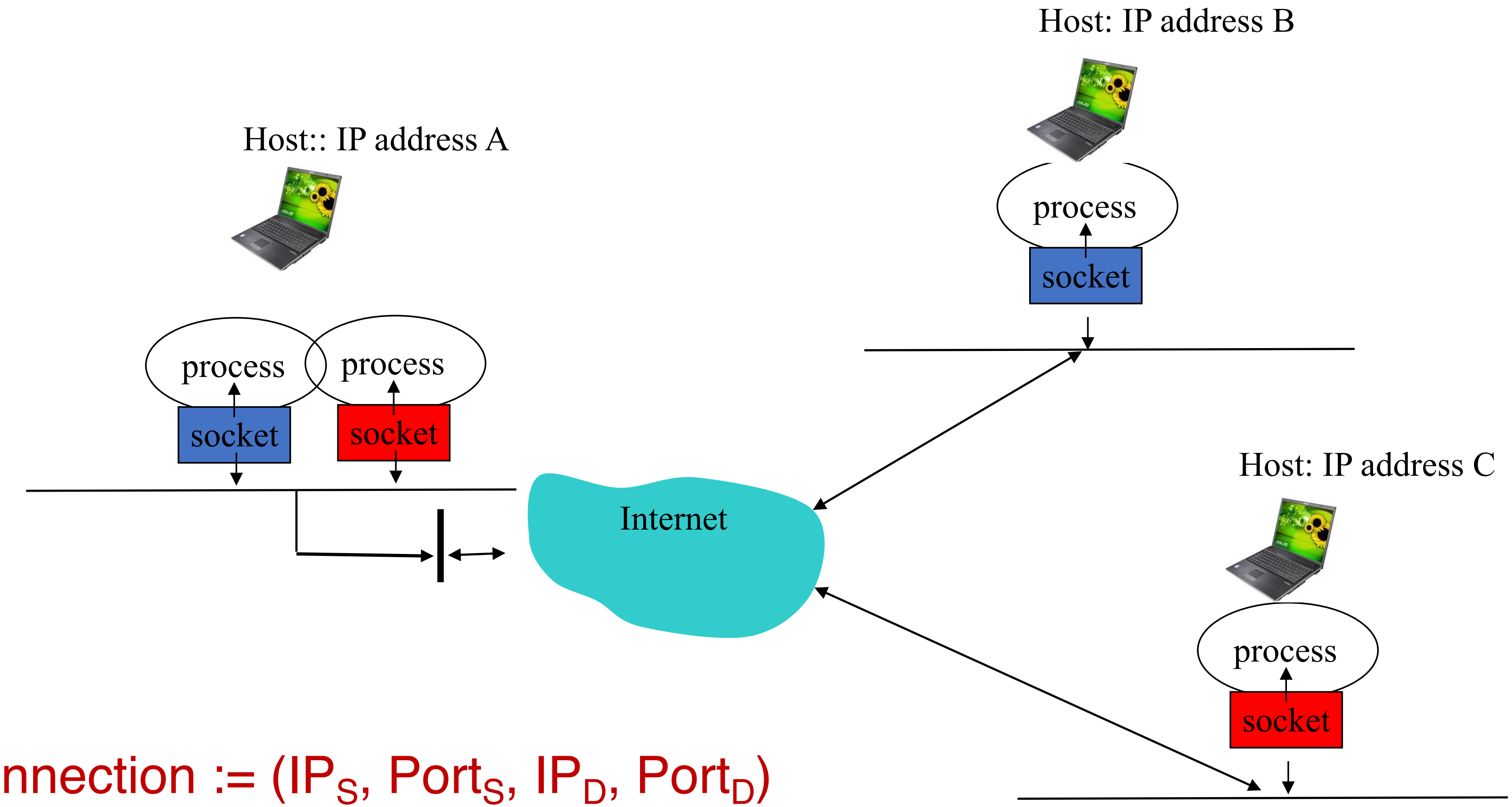
IP address & port number



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

The application's programming interface to the network

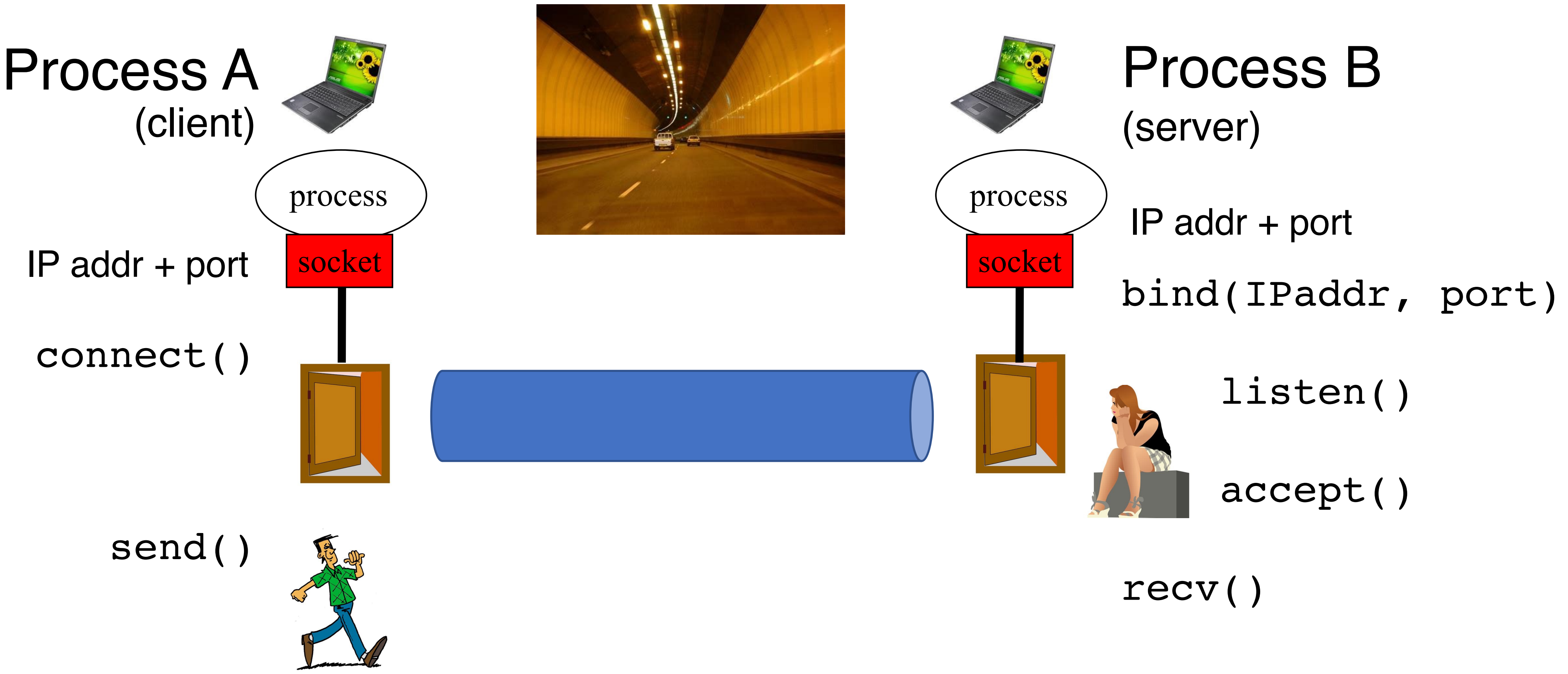
An app-layer connection is a 4-tuple



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

(S = source, D = destination)

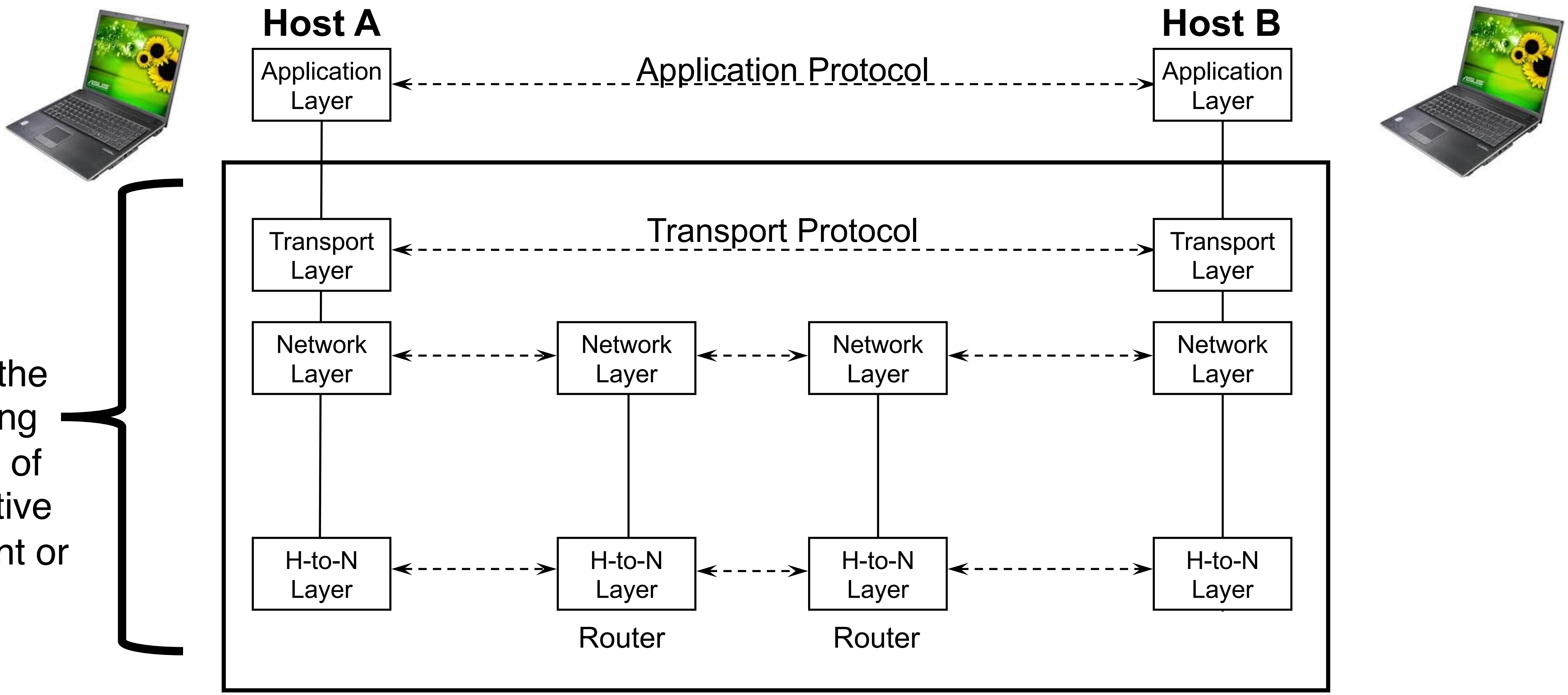
Socket system calls



Seeing app-layer connections

- `netstat`

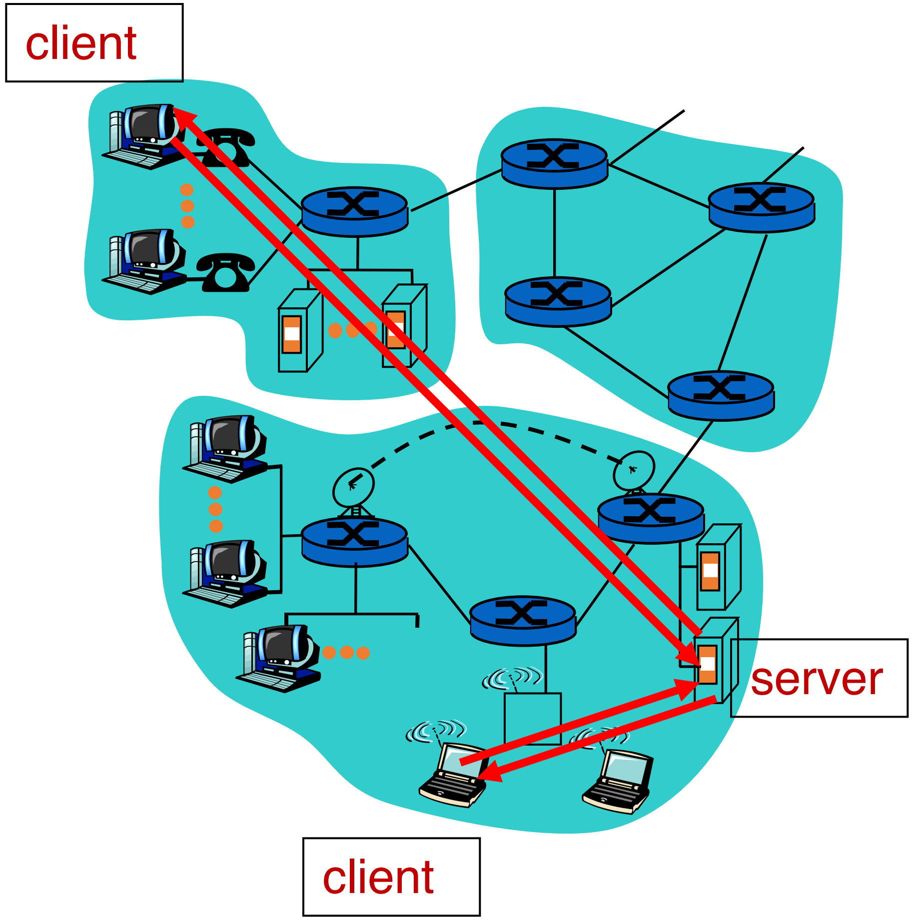
Recall: Apps rely on services by lower layers



Within the operating system of respective endpoint or router

Common Architectures of Applications

Client-server architecture



Server:

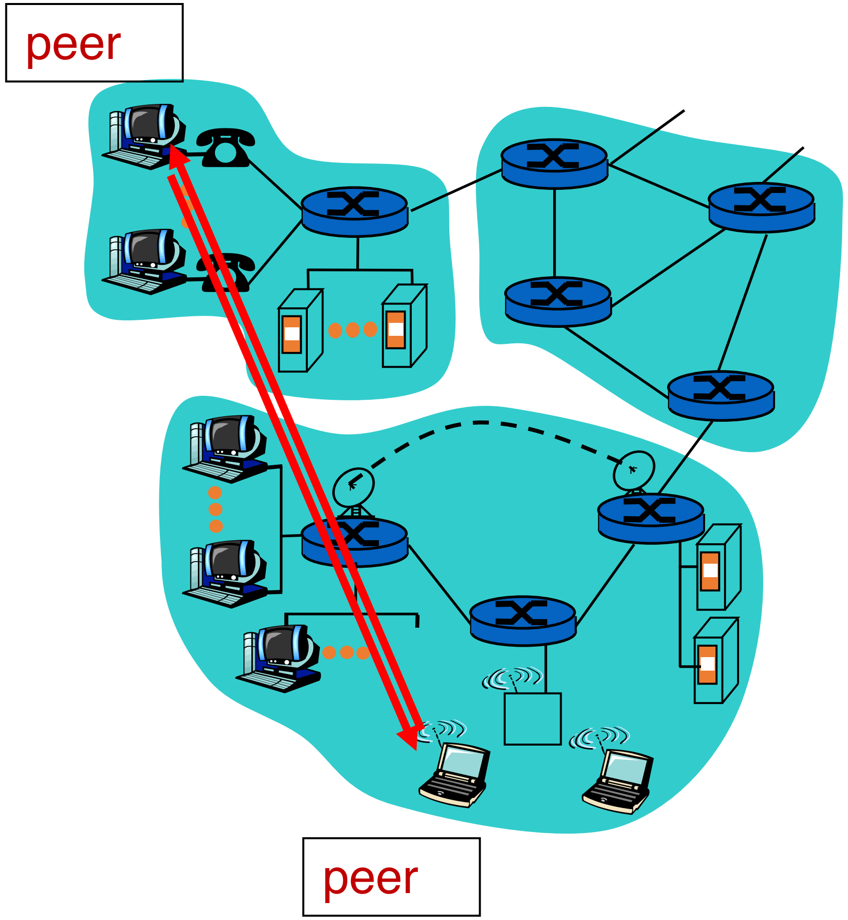
- Always-on endpoint
- Provides a “service” to the world
- Typically permanent IP address
- Compute clusters to scale to many users

Clients:

- A “customer” of the server
- May be intermittently connected
- May have dynamic IP addresses
- Typically do not communicate directly with other clients

- The web (HTTP) works this way!
- Many mobile apps too (e.g., Instagram)

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”

Going forward: A few applications

- Domain Name System
- The web: HTTP
- Mail
- Streaming video

Domain Name System

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

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

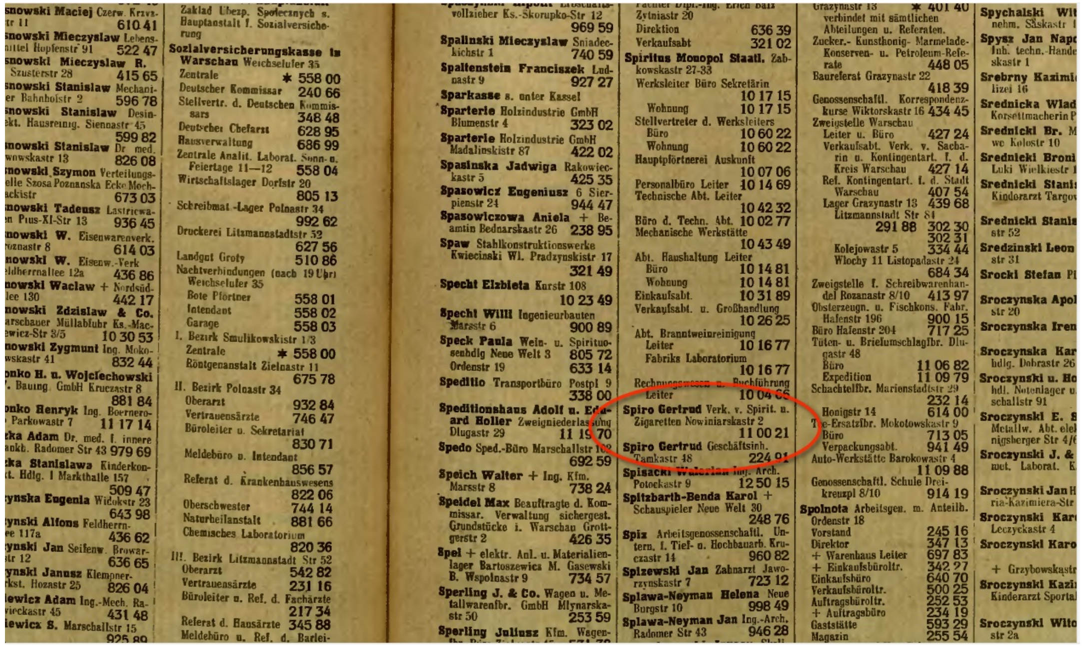
- Solution:
  - Use alphanumeric names to refer to hosts. Called host names or domain names
    - Example: cs.rutgers.edu
  - We need a directory (address book): add a service to map between alphanumeric host names and binary IP addresses
  - We call this process Address Resolution

Types of Directories

- Directories map a *name* to an *address*
- Simplistic designs
  - Central directory
  - Ask everyone (e.g., flooding)
  - Tell everyone (e.g., push to a file like /etc/hosts)
- Scalable distributed designs
  - Hierarchical namespace (e.g., Domain Name System (DNS))
  - Flat name space (e.g., Distributed Hash Table)

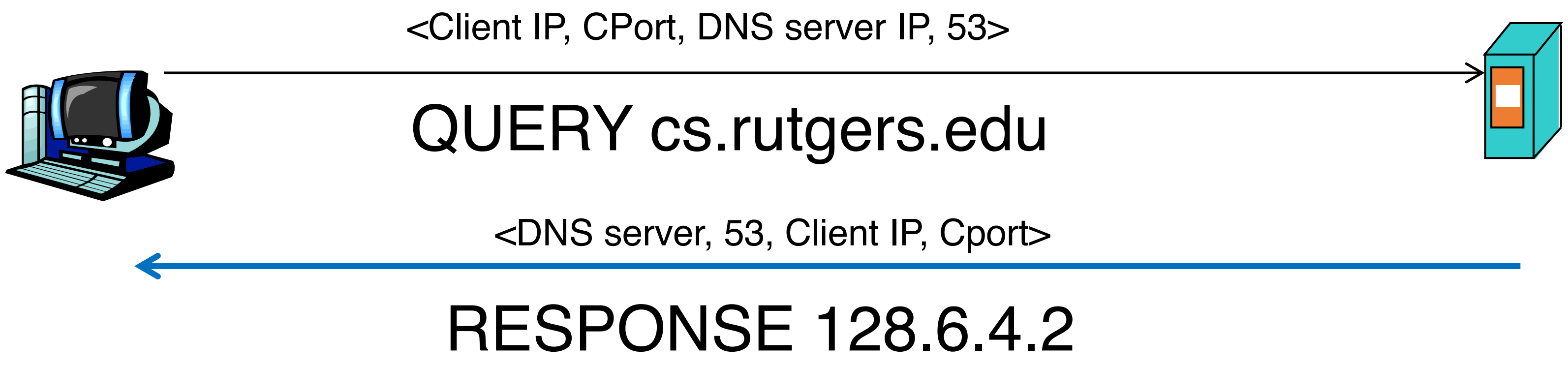
Simple DNS

- What if every endpoint has a local directory?
- `/etc/hosts.txt`
  - How things worked in the early days of the Internet!
- What if endpoints changed addresses? How do you keep this up to date?



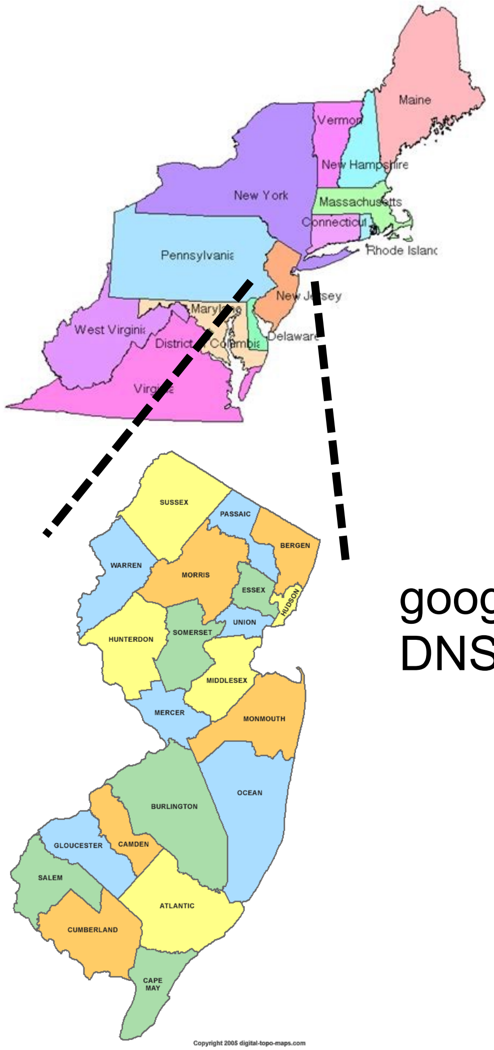
Simple DNS

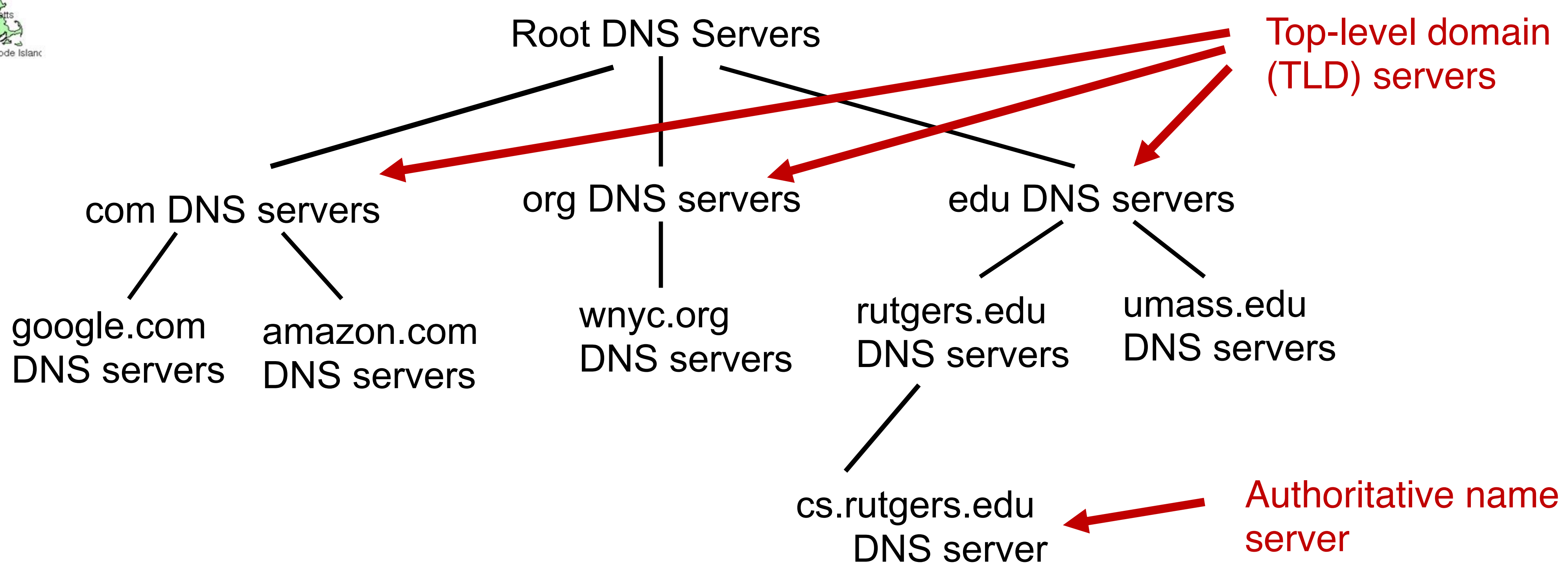
DOMAIN NAME	IP ADDRESS
spotify.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



- Key idea: Implement a server that looks up a table.
- Will this scale?
  - Every new host needs to be entered in this table
  - Performance: can the server serve billions of Internet users
  - Failure: what if the server or the database crashes?
  - How to secure this server?

Distributed and hierarchical database





RFC 1034: Distribution through hierarchy enables scaling

DNS Protocol

- Client and Server
- Client connects to Port 53 on server
- 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

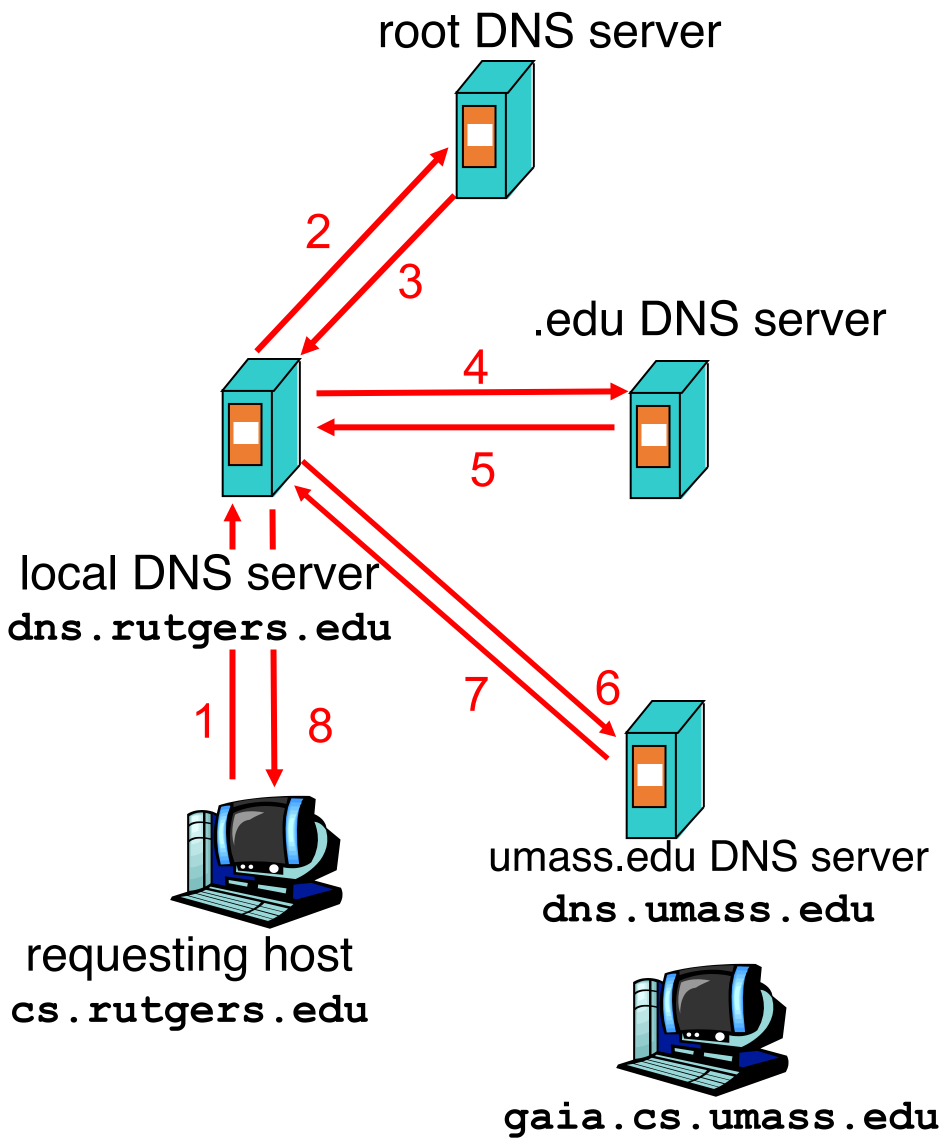
DNS Protocol

- 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 cs.rutgers.edu wants IP address for gaia.cs.umass.edu

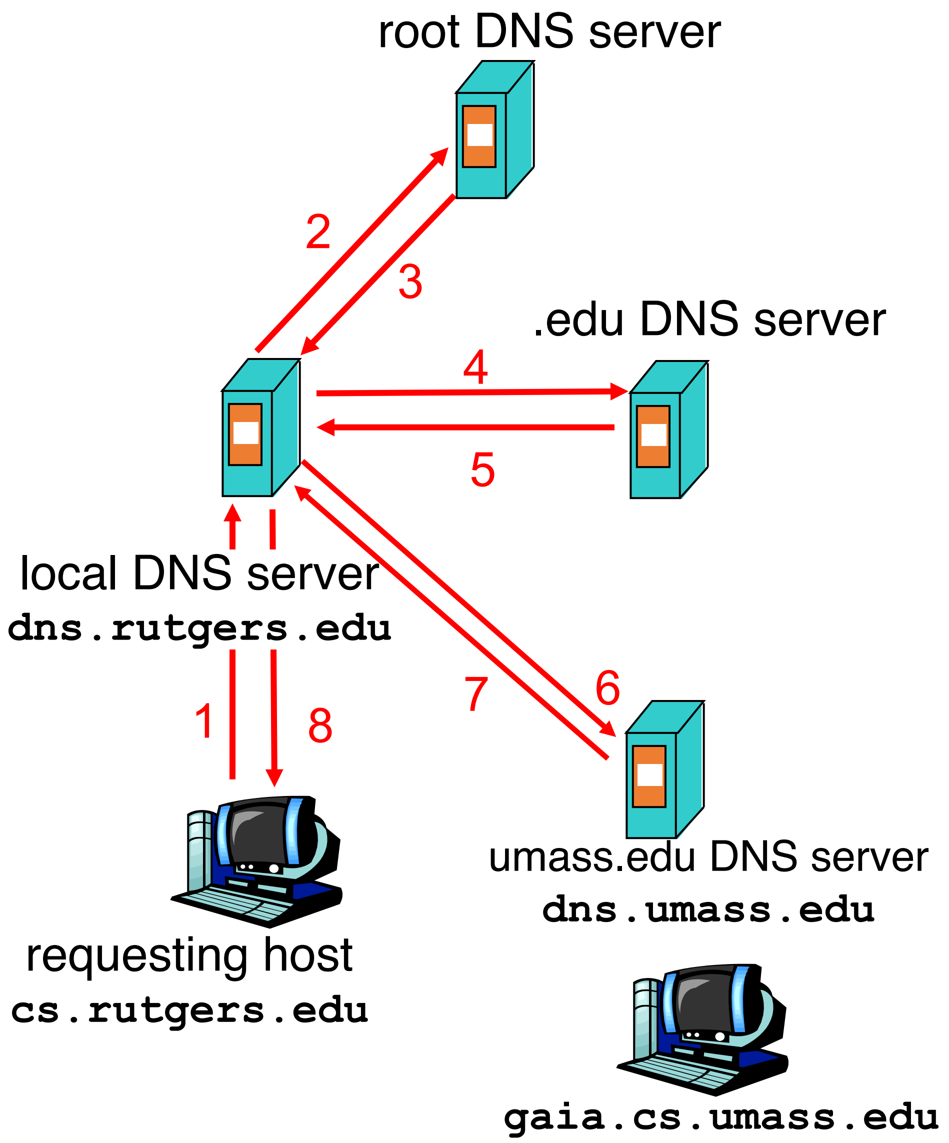
- Local DNS server
- Root DNS server
- TLD DNS server
- Authoritative DNS server



Query type

Iterative query:

- Contacted server replies with name of server to contact
- "I don't know this name, but ask this server"
- Queries are iterative for the local DNS server



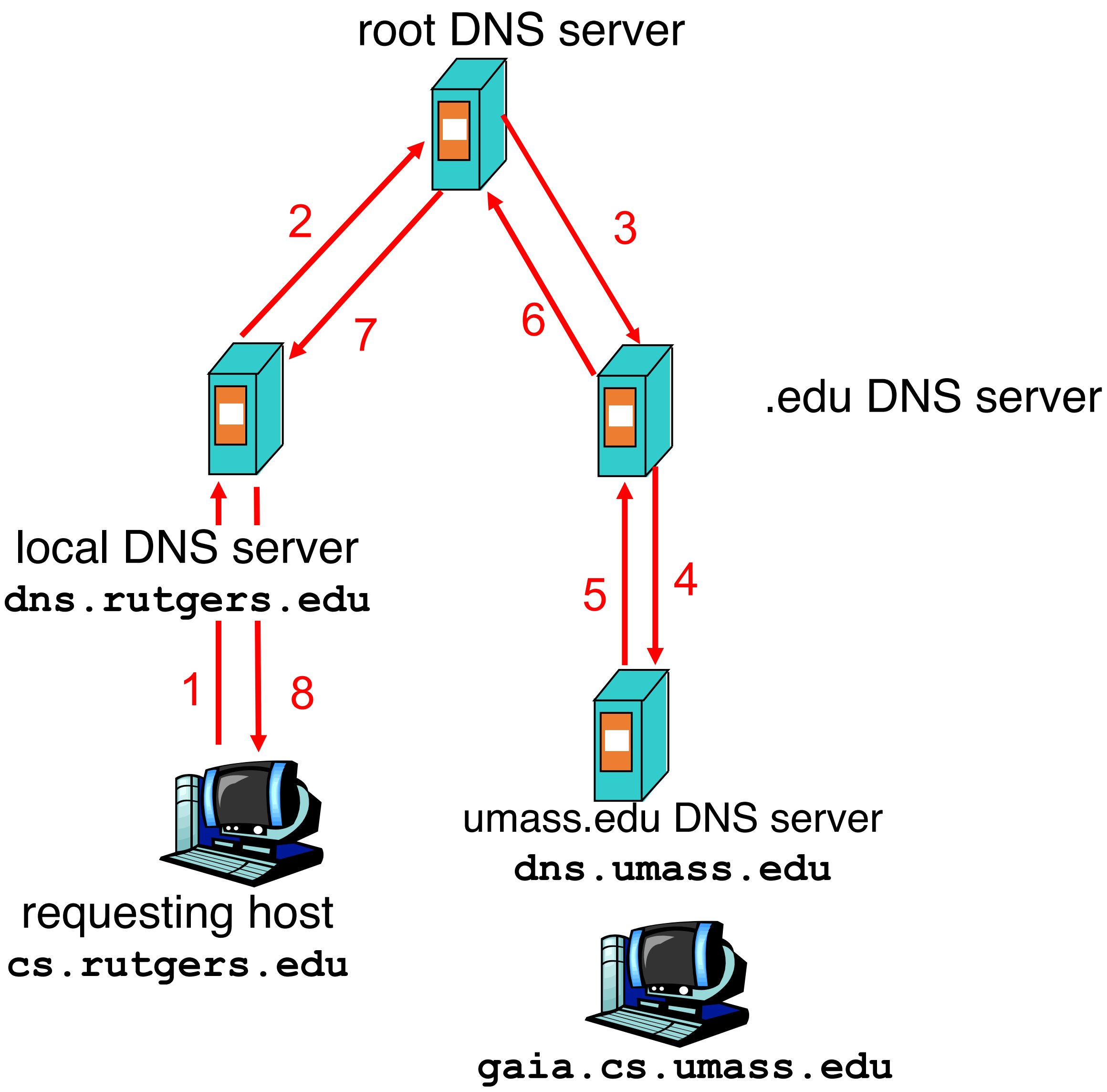
Query type

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 in action

- `dig <domain-name>`
- `dig +trace <domain-name>`
- `dig @<dns-server> <domain-name>`

DNS may seem simple, 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 (headers):
  - 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 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

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**

More complete info at https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

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>`

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

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)

Summary of DNS

- Hostname to IP address translation via a global network of servers
- Use Multiple layers of indirection
  - Hierarchically scale
  - Good performance (load distribution)
  - Resilient to local transient failure
- Additional load distribution can happen at each level (e.g., TLD server)
- Uses caching all over for better performance

- DNS can be used to implement useful primitives atop domain names:
- Example: Scaling large web services, e.g., google search
- Domain-authoritative server will return an address from a pool of IP addresses, for example from Google's server "farm"

Some themes and observations on DNS

- Request/response nature of the protocol
- How messages are structured: simple, text-based protocol
  - Similar in HTTP, SMTP, FTP
- Load distribution through hierarchy and replication
- Caching is an effective method to improve performance