

# CS 352

# The Application Layer

Lecture 3.1, Spring 2020

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

Srinivas Narayana

# 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

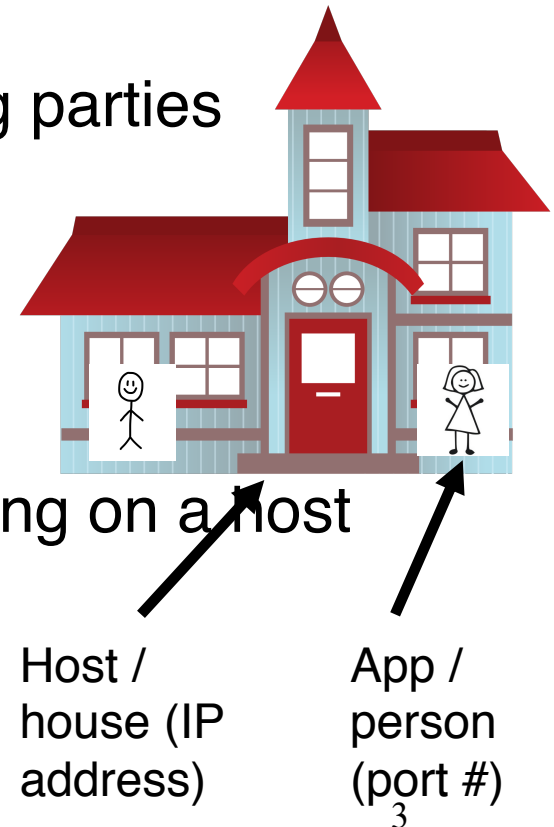
## Proprietary protocols:

- e.g., Skype

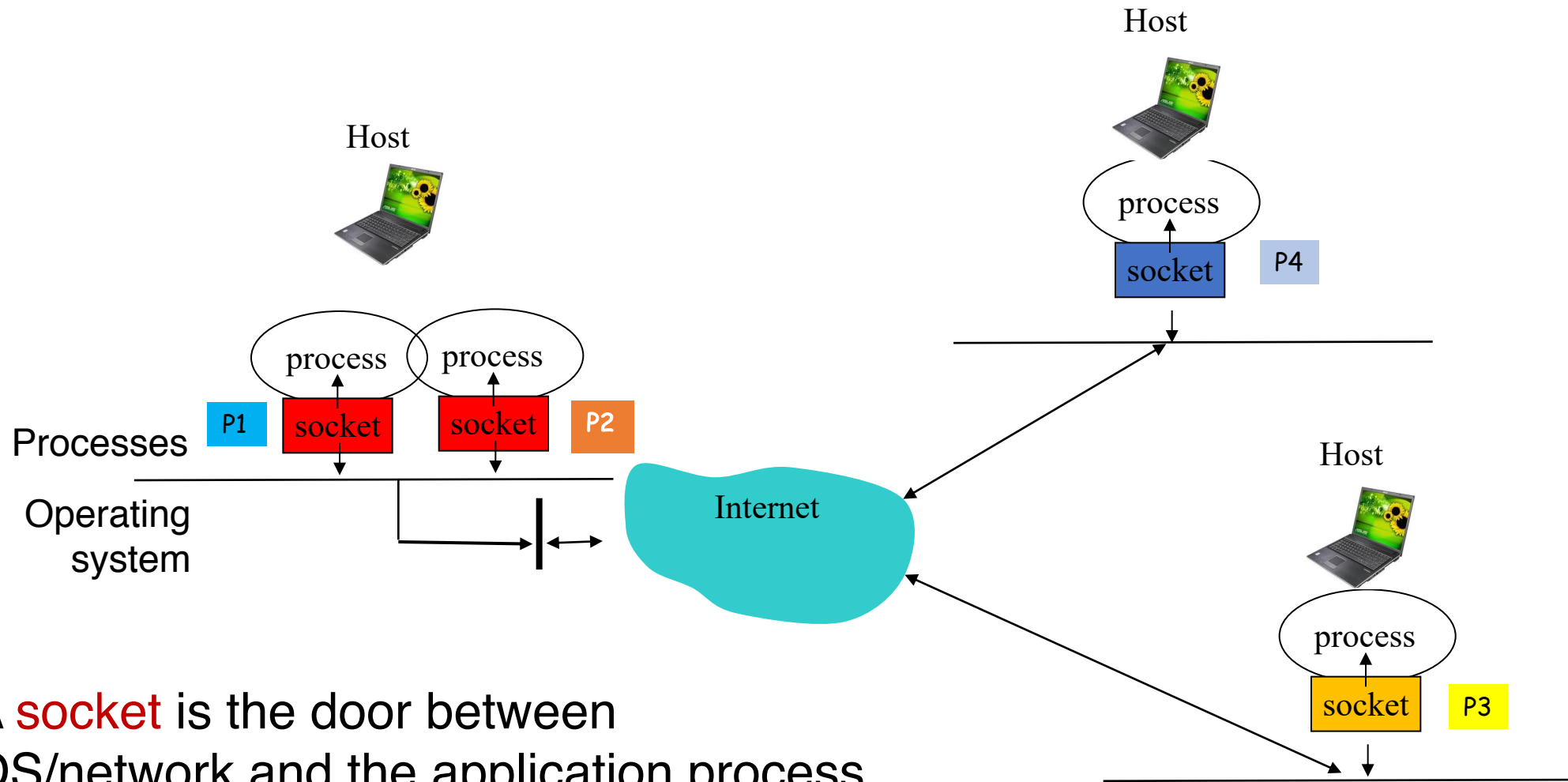


# Application Addresses

- 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**



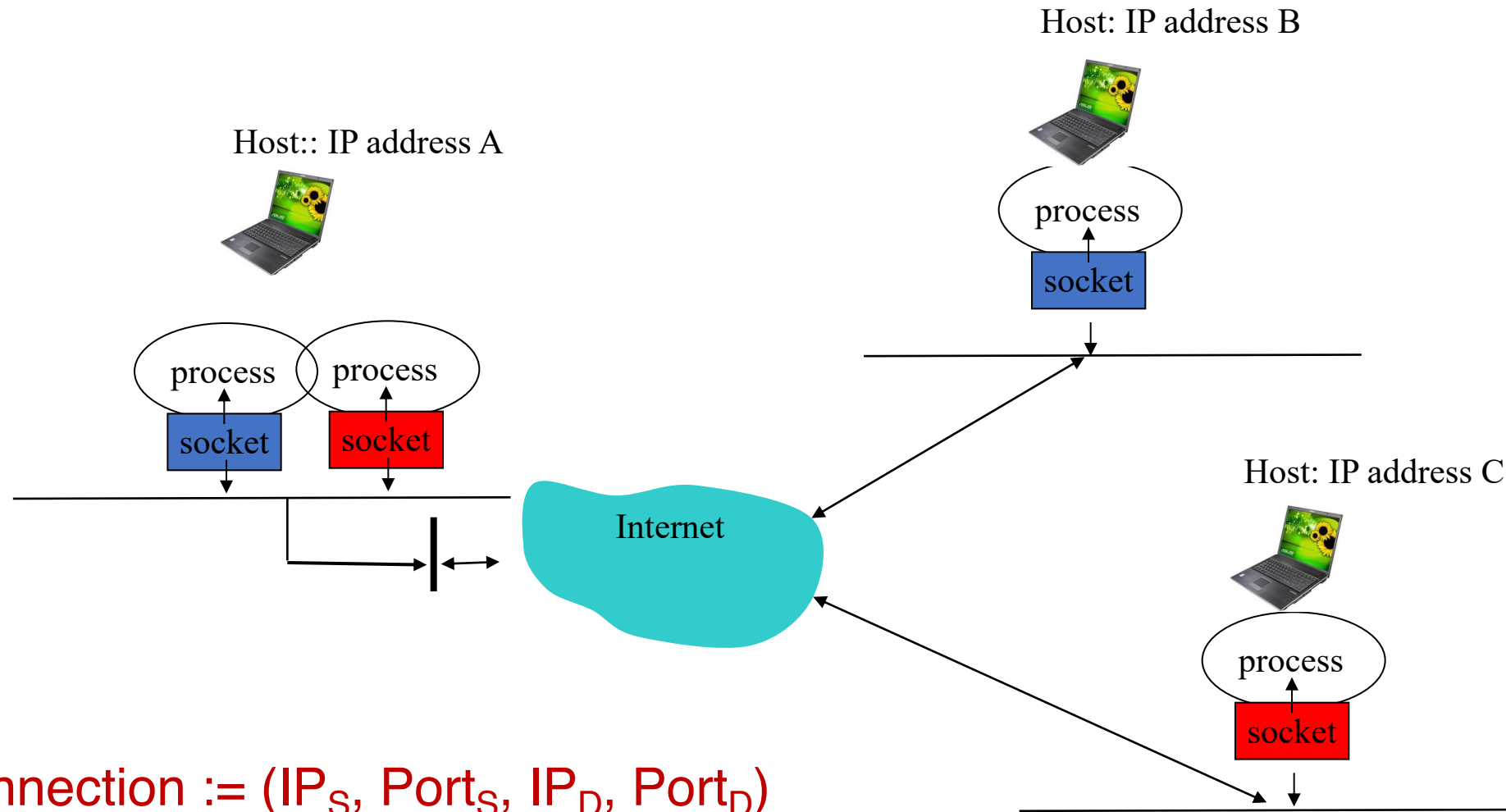
# IP address & port number



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

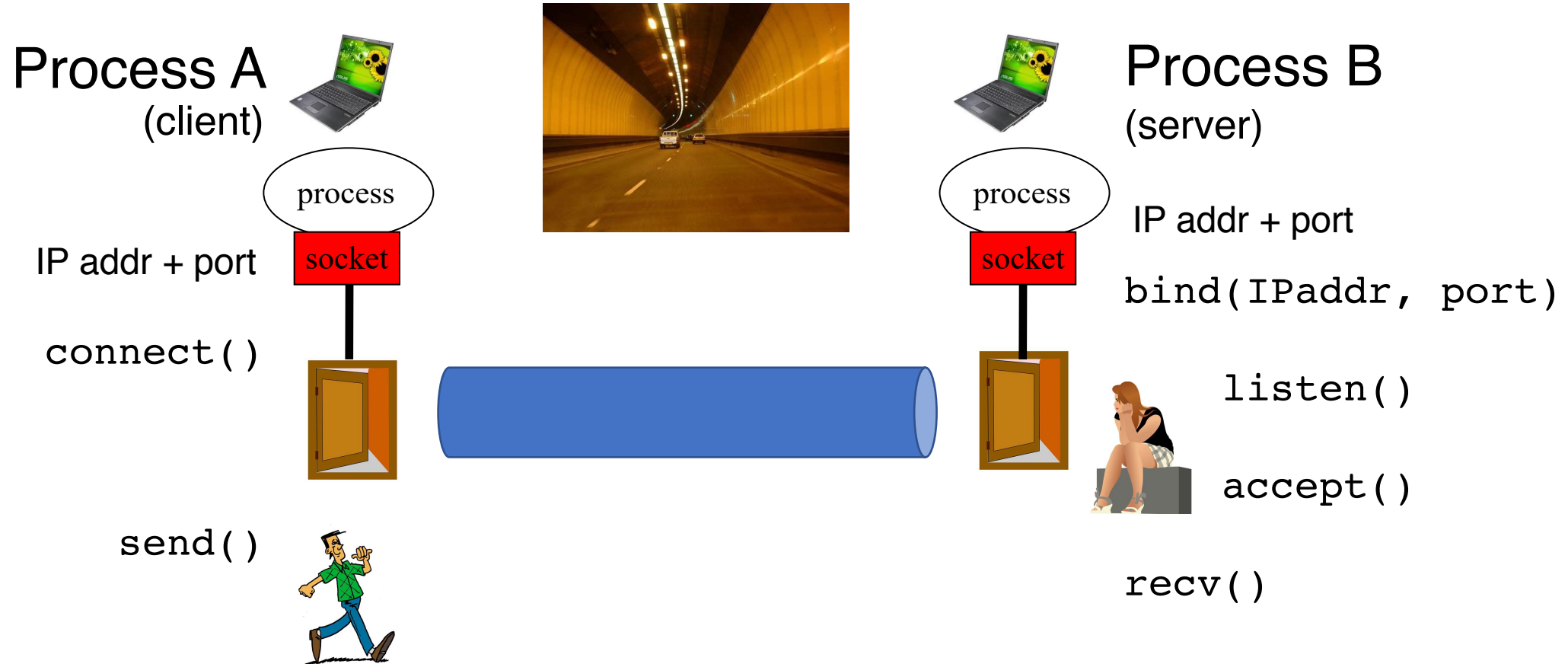
The **application's programming interface** to the network

# An app-layer connection is a 4-tuple



Connection := (IP<sub>S</sub>, Port<sub>S</sub>, IP<sub>D</sub>, Port<sub>D</sub>)  
(S = source, D = destination)

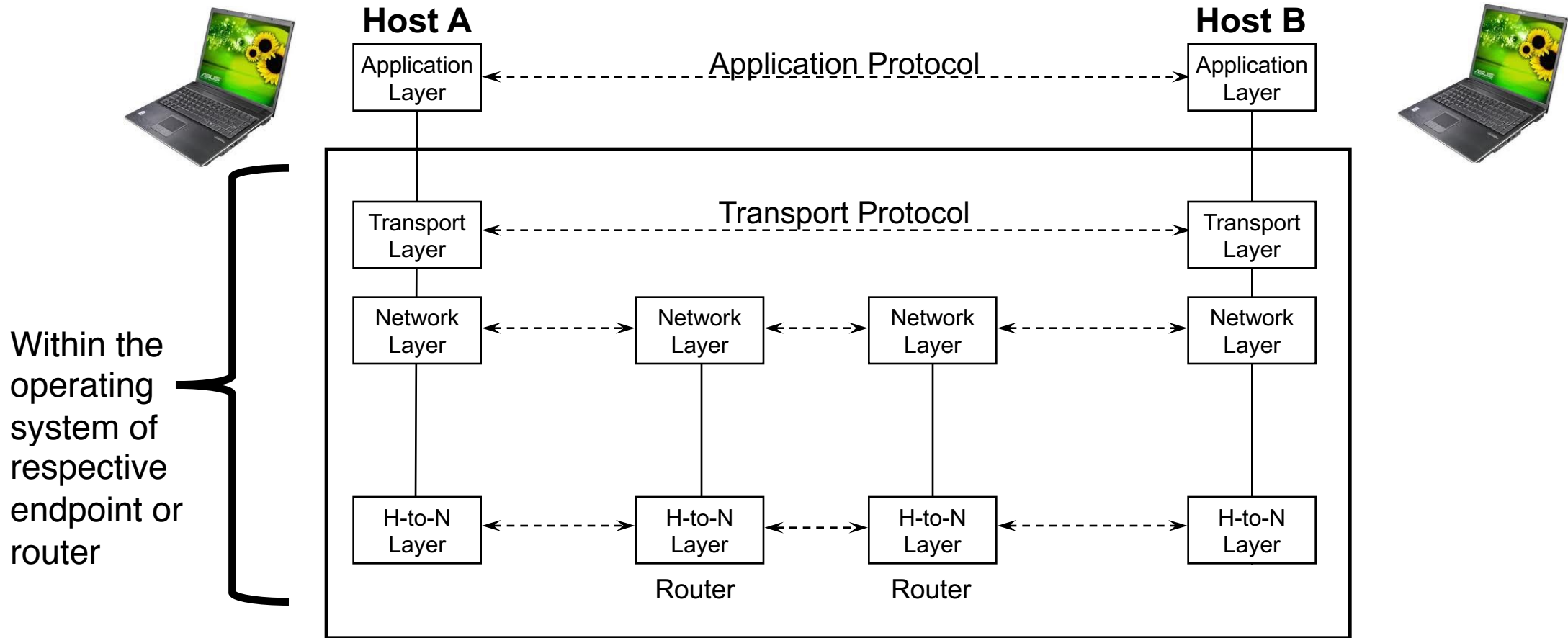
# Socket system calls



# Seeing app-layer connections

- `netstat`

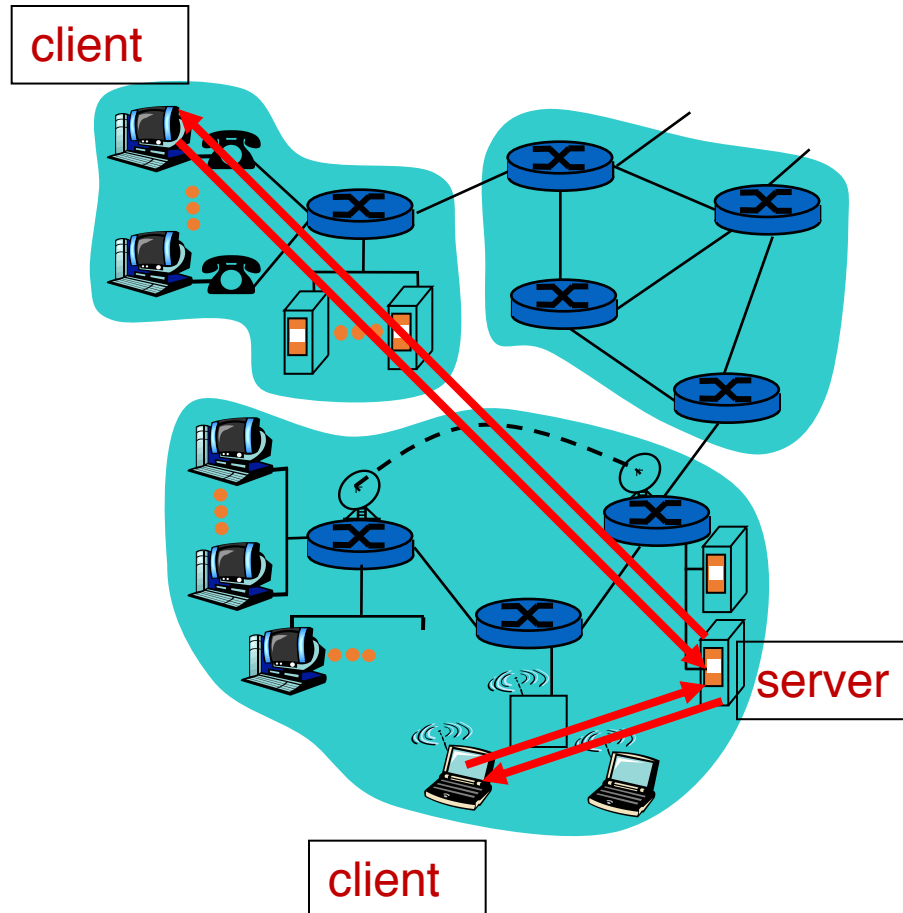
# Recall: Apps rely on services by lower layers





# Common Architectures of Applications

# Client-server architecture



## Server:

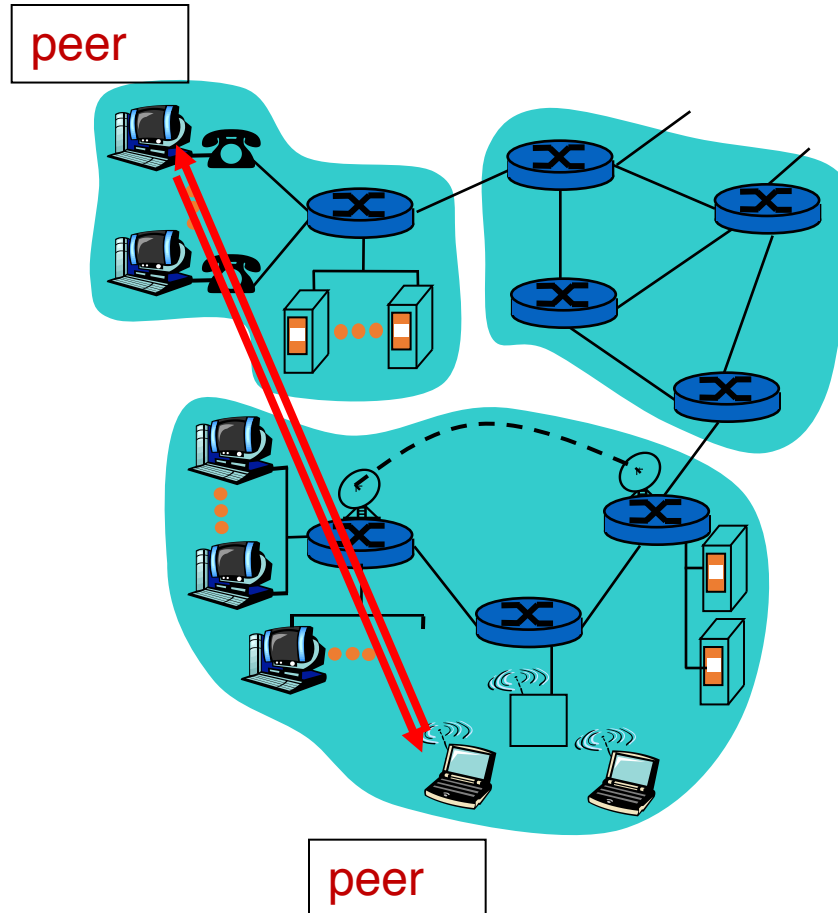
- Always-on endpoint
- Provides a “service” to the world
- Typically permanent IP address
- Compute clusters to scale to many users

## 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?”

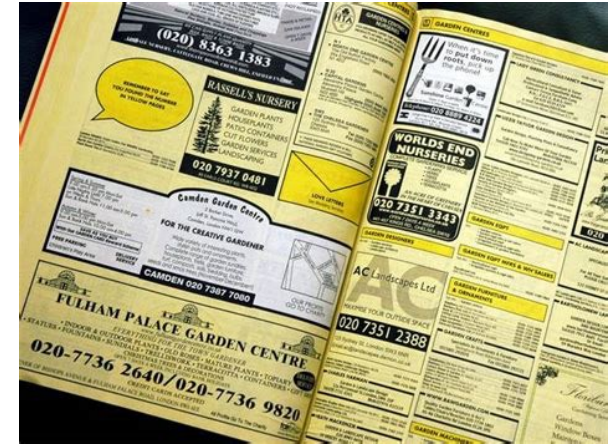
# Domain Name System (DNS)

- 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)



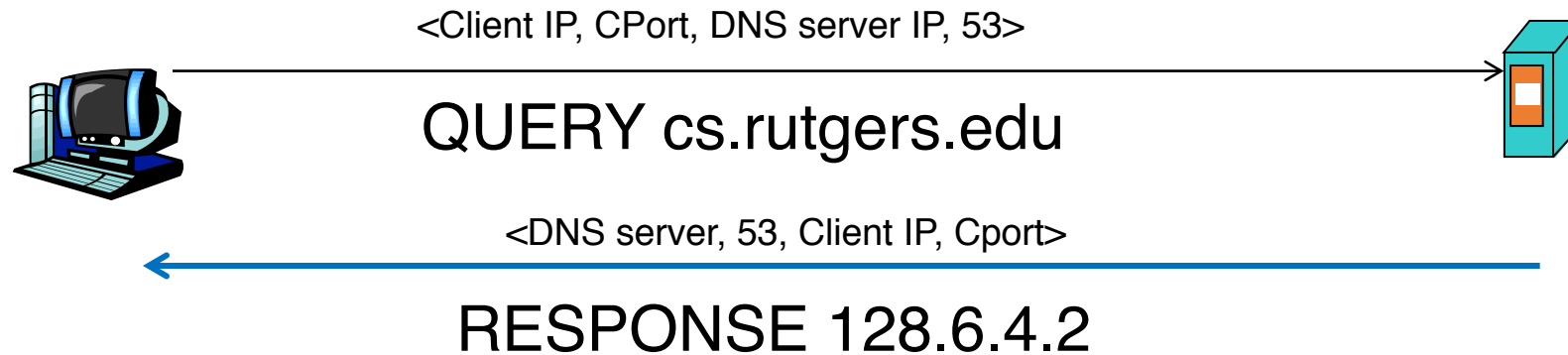
# Simple DNS

- 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?

<b>nowski Maciej</b> Gerny Kryst. tr 11 610 41 <b>nowski Mieczysław</b> Lehens- mittel Hopfenstr 91 522 47 <b>nowski Mieczysław R.</b> Snaustewer 28 415 65 <b>nowski Stanisław</b> Mechan- iker Bahnhofstr 2 596 78 <b>nowski Stanisław</b> Desin- kt. Hausrenung, Siemowstr 45 599 82 <b>nowski Stanisław</b> Dr. med. nowskastr 13 826 08 <b>nowski, Szymon</b> Verteilungs- stelle Sosna Pomska Erdo-Mech- schistr 673 03 <b>nowski Tadeusz</b> Lastreiw- n Pusz-M-Str 13 936 45 <b>nowski W.</b> Eisenwarenverk. Snaust 8 614 03 <b>nowski W.</b> Eisenw.-Verk. Hahernallee 12a 436 86 <b>nowski Wacław</b> + Nordind- ice 130 442 17 <b>nowski Zdzisław</b> & Co. arschauer Mühlabühr Ks. Mac- ewic-Str 3/5 10 30 53 <b>nowski Zygmunt</b> Ing. Miko- wast 41 832 44 <b>oko H. u. Wojciechowski</b> , Baug. GmbH Kreuzstr 8 881 84 <b>oko Henryk</b> Ing. Boernero- Parkwastr 7 11 17 14 <b>oka Adam</b> Dr. med. I. inner- nuk. Radomer Str 43 979 69 <b>oka Stanisława</b> Kinderkon- d. Bldg. I Markthalle 157 509 47 <b>ynska Eugenia</b> Widostr 25 643 98 <b>ynski Alfons</b> Feldtherm- re 117a 436 62 <b>ynski Jan</b> Seilenn. Brower- str 12 636 65 <b>ynski Janusz</b> Klemmer- Str. Hounz 28 826 04 <b>ewicz Adam</b> Ing.-Mech. Ba- reckstr 45 431 48 <b>ewicz S.</b> Marschallstr 15 925 80	<b>Zaklad Ubezpie. Społecznych u.</b> Hauptanstalt f. Sozialversiche- rung <b>Sozialversicherungskasse in</b> <b>Warschau</b> Weichselufer 33 Zentrale * 558 00 Deutscher Kommissar 240 66 Stellvertr. d. Deutsches Kommis- sars 348 48 Deutscher Chefars 628 95 Hausverwaltung 686 99 Zentrale Analit. Laborat. Sonn. u. Feiertage 11-12 558 04 Wirtschaftslager Dorfstr 20 805 13 Schreibmat.-Lager Polnast 34 992 62 Druckerei Litmannstadtstr 32 627 56 Landgut Grotz Nachtverbindungen (nach 19 Uhr) Weichselufer 35 Rote Florent 558 01 Intendant 558 02 Garage 558 03 I. Bezirk Smulikowskistr 1/3 Zentrale * 558 00 Röntgenanstalt Zielast 11 675 78 II. Bezirk Polnast 34 Oberarzt 932 84 Vertrauensärzte 746 47 Büroleiter u. Sekretariat 830 71 Meldebüro u. Intendant 856 57 Referat d. Krankenhauswesens 822 06 Überschweste 744 14 Naturheilstalt 681 66 Chemisches Laboratorium 820 36 III. Bezirk Litmannstadt Str 52 Oberarzt 542 82 Vertrauensärzte 231 16 Büroleiter u. Ref. d. Facharzt- B. Wapolskistr 217 34 Referat d. Hausärzte 345 88 Meldebüro u. Ref. d. Barlei-	<b>Spallinski Mieczysław</b> Snaude- kuchstr 1 740 59 <b>Spaltenstein Franciszek</b> Lud- nast 9 927 27 <b>Sparkasse s. unter Kassel</b> <b>Spartaria Holzindustrie</b> GmbH Blumenstr 4 323 02 <b>Spartaria Holzindustrie</b> GmbH Madalinskistr 87 422 02 <b>Spasinska Jadwiga</b> Rakowiec- kast 5 425 35 <b>Spasowicz Eugeniusz</b> 6 Sier- pienski 24 944 47 <b>Spasowiczowa Aniela</b> + Be- amin Redarskast 26 238 95 <b>Spaw</b> Stahlkonstruktionswerke Kwiecinski Wl. Pradyskiskistr 17 321 49 <b>Specht Elzbieta</b> Kurstr 108 10 23 49 <b>Specht Willi</b> Ingenieurkaut- Marstr 6 900 89 <b>Speck Paula</b> Welo. u. Spiritus- seehldg Neue Welt 8 805 72 Ordemtr 19 633 14 <b>Spedillo</b> Transportbüro Postpl 9 358 00 <b>Speditionhaus Adolf u. Ed- ard Holler</b> Zweigniederlaschub Dlugastr 29 11 19 70 <b>Spedo</b> Sped.-Büro Marschallstr 10 692 59 <b>Speich Walter</b> + Ing. Kfm. Marstr 8 739 24 <b>Speldel Max</b> Beauftragte d. Kom- missar. Verwaltung sichergestellt. Grundstücke I. Warschau Grotz- gust 2 426 35 <b>Spel</b> + elekt. Anl. u. Materialien- lager Bartoszewicz M. Gasowski B. Wapolskistr 9 734 57 <b>Sperling J. &amp; Co.</b> Wagon u. Mo- tallwarenbr. GmbH Mlynarska- str 30 253 59 <b>Sperling Juliusz</b> Kfm. Wagon- str 30 253 59	<b>Spiritus Monopol Staatl.</b> Zah- kowskast 27-38 Werkleiter Büro Sekretärin 10 17 15 10 17 15 <b>Spartaria Holzindustrie</b> GmbH Stellvertreter d. Werkseiler Leiter u. Büro 10 60 22 Wohnung 10 60 22 Hauptplorterei Auskunft 10 07 06 Eisen Warschau 427 14 Ref. Kontingentart. I. d. Stadt Warschau 407 54 Lager Grzywnastr 19 439 68 Litmannstadt Str 81 291 88 302 30 302 31 Kolejowastr 5 334 44 Wlochy 11 Listopadast 24 684 34 Zweigstelle f. Schreibwarenhan- del Rosanast 8/10 413 97 Osterezeug. u. Fischkons. Fabr. Halestr 196 900 15 Büro Halestr 264 717 25 Tulow. u. Briefumschlagfabr. Dis- kast 48 11 06 82 Büro 11 09 79 Expedition Schachtelfabr. Marionstadtstr 232 14 232 14 Hosigstr 14 614 00 1. u. Ersatzfabr. Mokotowskistr 9 Verpackungsbüro 941 49 Auto-Werkstätte Barokowskistr 4 11 09 88 Genossenschaft. Schule Drei- kreuzpl 8/10 914 19 Ordnast 18 245 16 Vorstand 247 13 Direktor 697 83 + Einkaufsbüro 242 27 Einkaufsbüro 640 70 Verkaufsbüro 500 25 Auftragbüro 252 53 + Auftragbüro 234 19 Gaststätte 593 29 Magazin 255 54	<b>Spychalski Wit</b> solim. Skakast 1 <b>Spyra Jan</b> Nap- Inh. techn. Hand- skast 1 <b>Srebrny Kazimi-</b> lusz 16 <b>Srednicka Wlad</b> Kontrollmachein F <b>Srednicki Br. M</b> we Kolost 10 <b>Srednicki Broni</b> Loki Wroklowstr 1 <b>Srednicki Stani-</b> Kindozent Targow <b>Srednicki Stanis-</b> str 52 <b>Srednicki Leon</b> str 31 <b>Srocki Stefan</b> Pl <b>Sroczyńska Apol</b> str 20 <b>Sroczyńska Iren</b> <b>Sroczyńska Kar</b> bldg. Dobrast 26 <b>Sroczyński u. He</b> bld. Nienlager u schallstr 91 <b>Sroczyński E. S</b> Metallw. Abt. eliel nigskberger Str 4/4 <b>Sroczyński J. &amp;</b> med. Laborat. E <b>Sroczyński Jan H</b> ria-Kosmiera-Str <b>Sroczyński Kar</b> Lecakast 4 <b>Sroczyński Karo</b> + Grzybowskaya <b>Sroczyński Kazi</b> Kinderarzt Sporta <b>Sroczyński Wito</b> str 2a
---	---	---	--	---

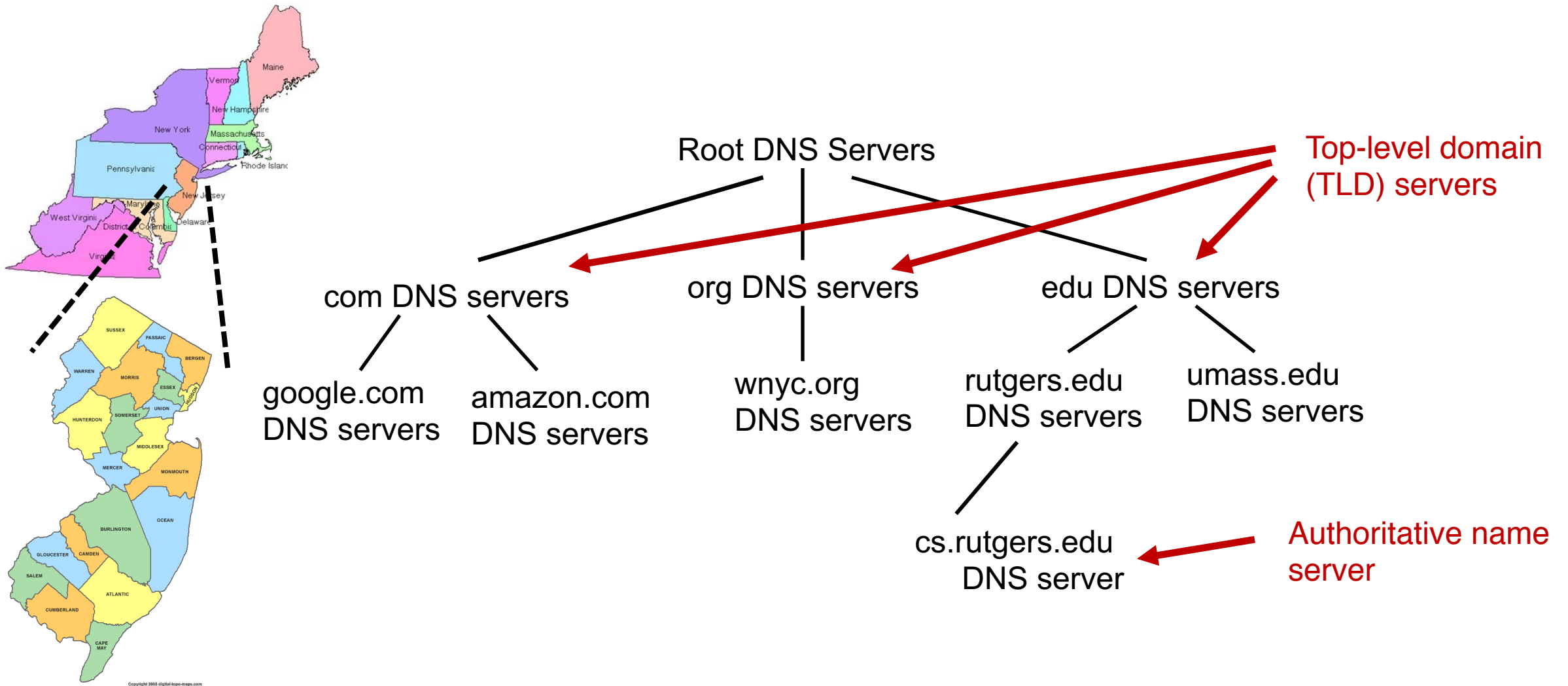
# Simple DNS

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

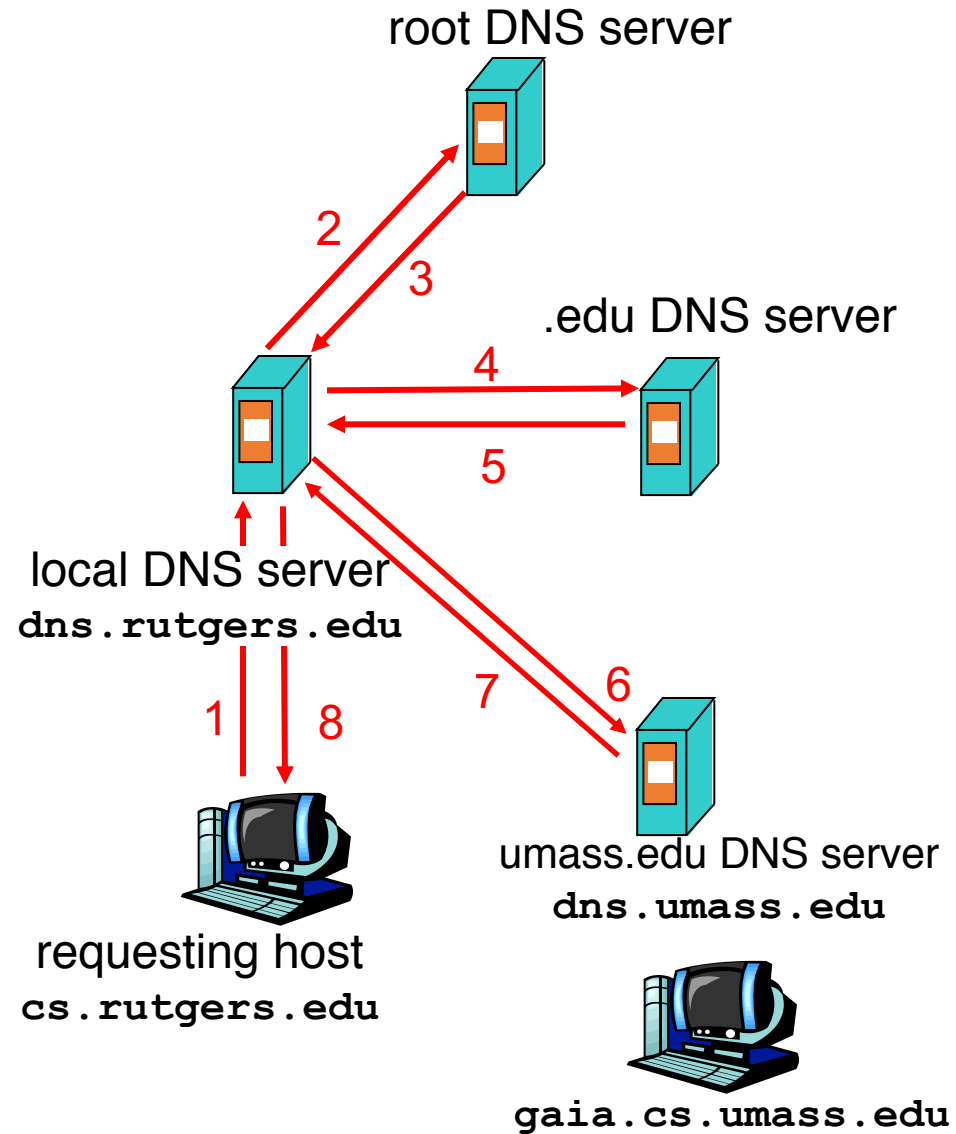
- Client and Server
- Client connects to Port 53 on server
- Assume DNS server IP known
- Two types of messages
  - Queries
  - Responses
- Type of Query (OPCODE)
  - Standard query (0x0)
    - e.g., Request IP address for a given domain name
  - Updates (0x5)
    - Provide a binding of IP address to 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

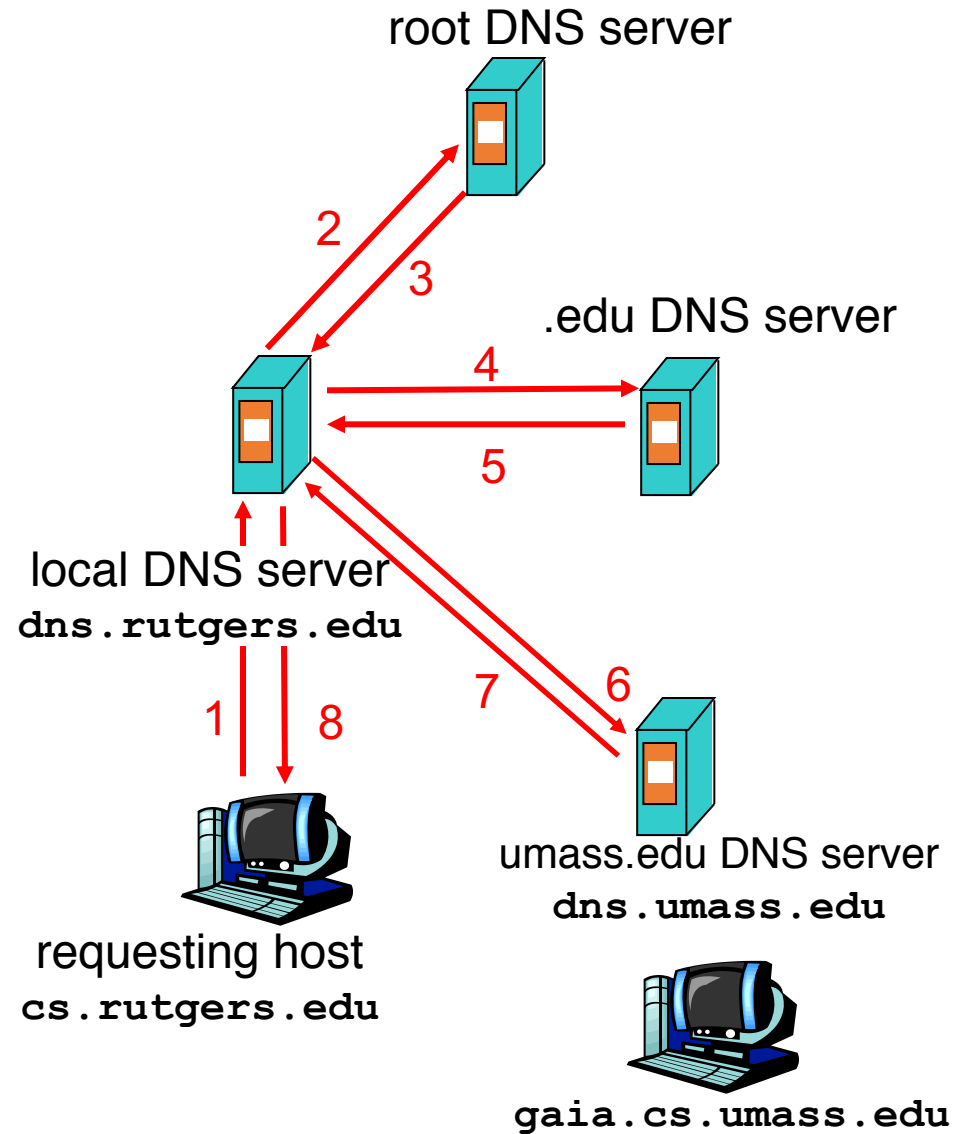
- Host at cs.rutgers.edu wants IP address for gaia.cs.umass.edu
- Local DNS server
- Root DNS server
- TLD DNS server
- **Authoritative** DNS server



# 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





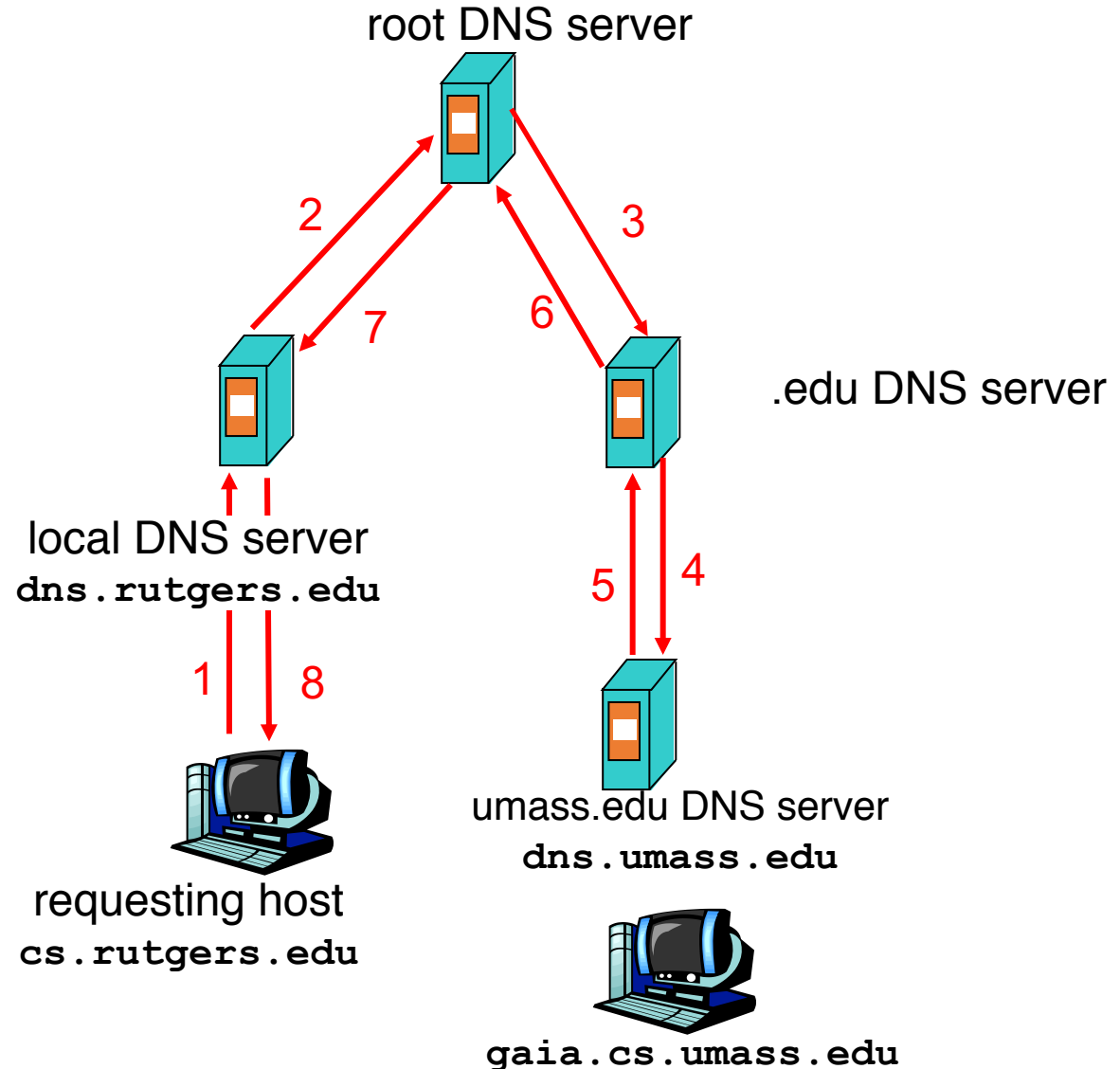
# Query type

## 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**

# 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