CS 43: Computer Networks

06:Distributed Systems and DNS

September 24, 2024

SWARTHMORE COLLEGE

Reading Quiz

Last class

- Inter-process communication using message passing
- How send and recv buffers work
- Concurrency

Today

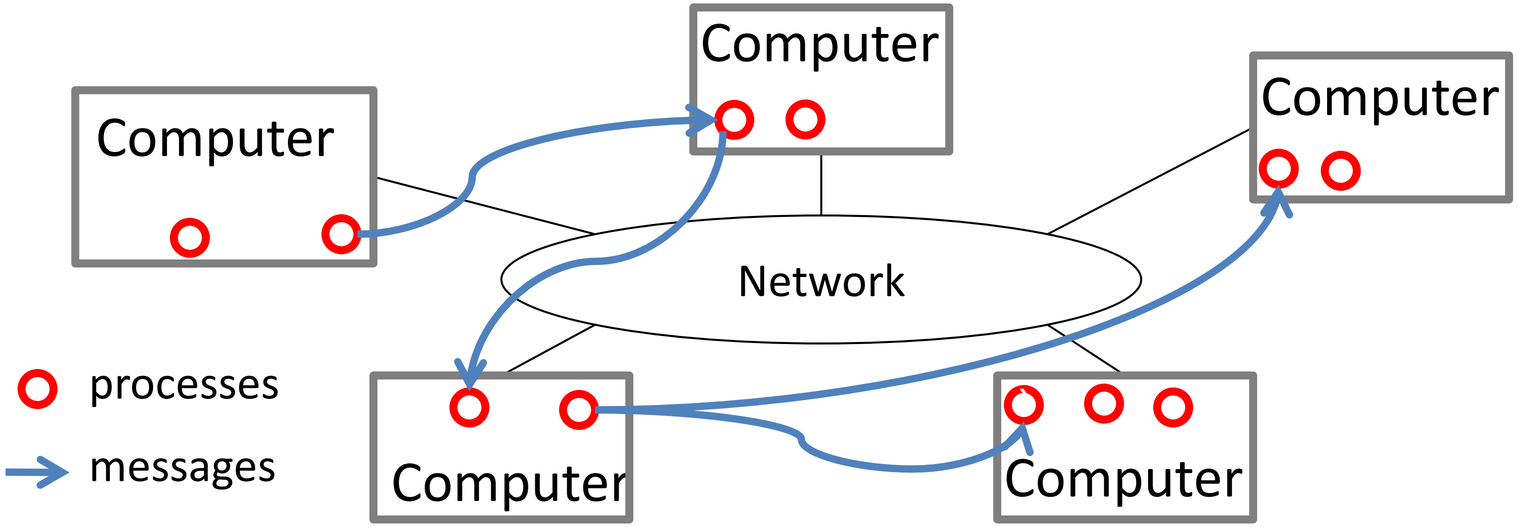
- Application-layer communication paradigms:
  - Client-Server
  - Peer-to-peer architecture
- Distributed network applications: Sources of complexity
- DNS

Where we are

Application: the application (e.g., the Web, Email)
Transport: end-to-end connections, reliability
Network: routing
Link (data-link): framing, error detection
Physical: 1's and 0's/bits across a medium (copper, the air, fiber)

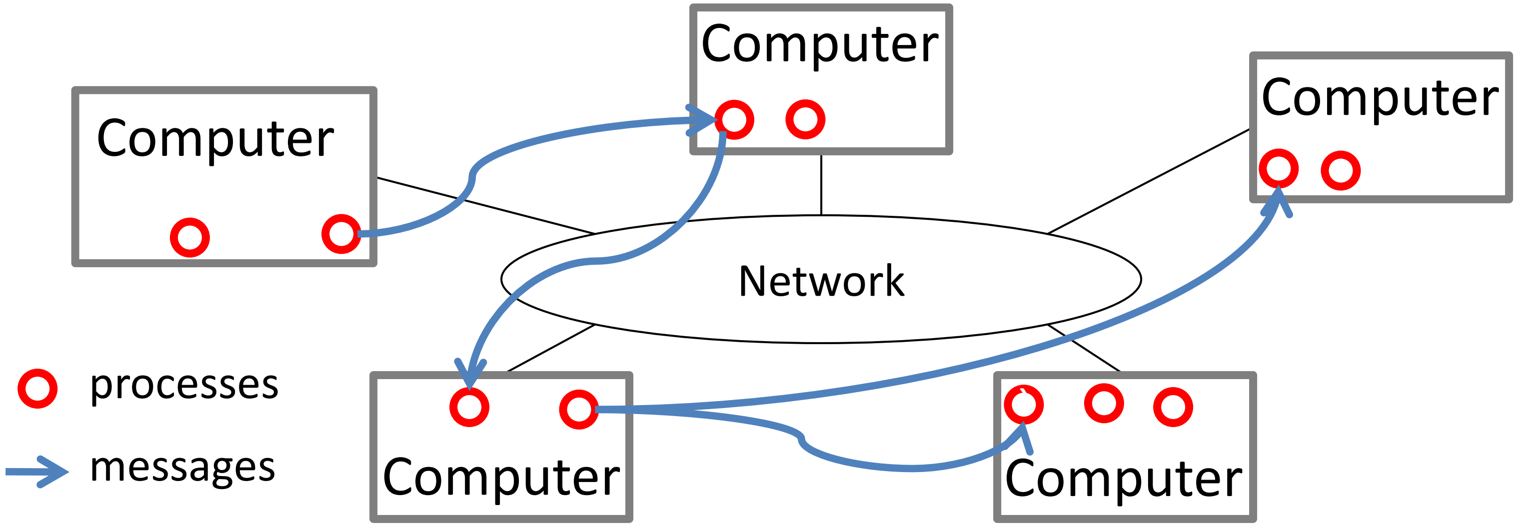
Distributed Network Applications

What is a distributed application?



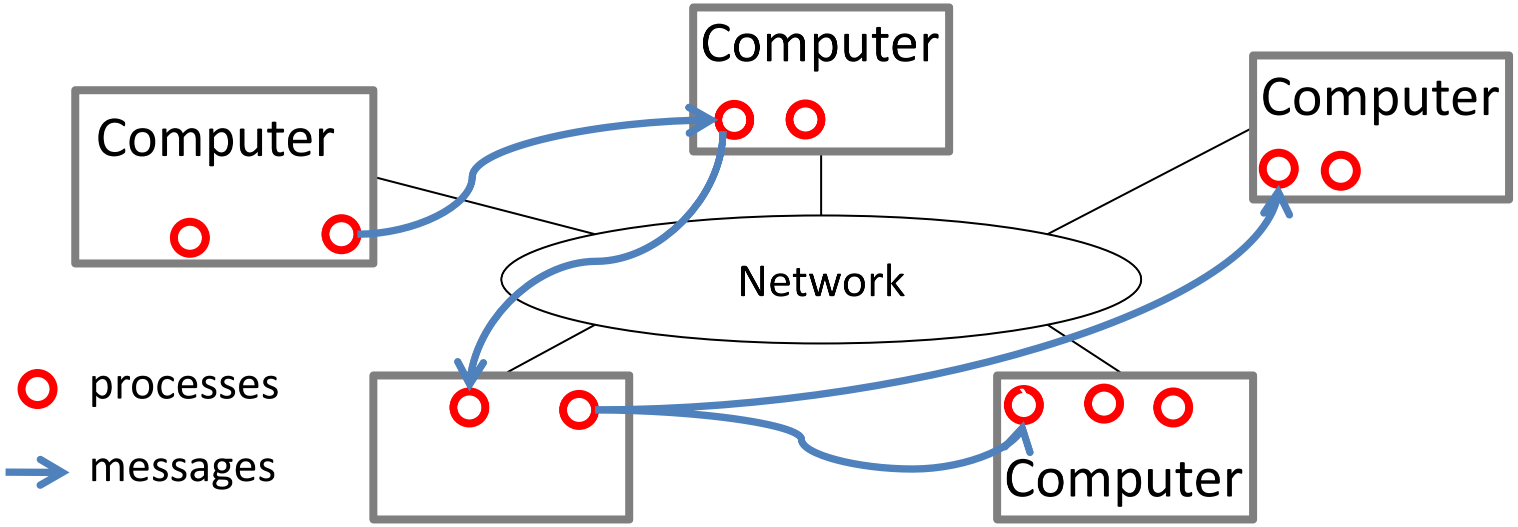
- Cooperating processes in a computer network
- Varying degrees of integration
  - Loose: email, web browsing
  - Medium: chat, Skype, remote execution, remote file systems
  - Tight: process migration, distributed file systems

Distributed Systems: Advantages



- Speed: parallelism, less contention
- Reliability: redundancy, fault tolerance (NSPF)
- Scalability: incremental growth, economy of scale
- Geographic distribution: low latency, reliability

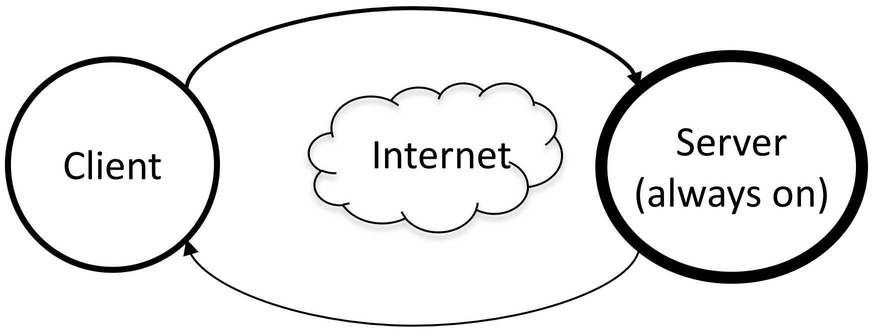
Distributed Systems: Disadvantages



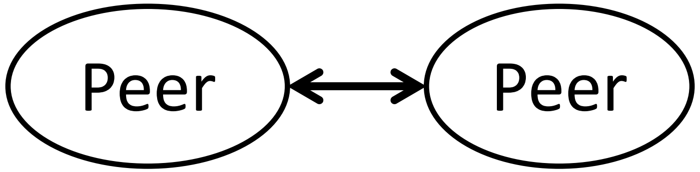
- Fundamental problems of decentralized control
  - State uncertainty: no shared memory or clock
  - Action uncertainty: mutually conflicting decisions
- Distributed algorithms are complex

Designating roles to an endpoint

Client-server architecture



Peer-to-peer architecture



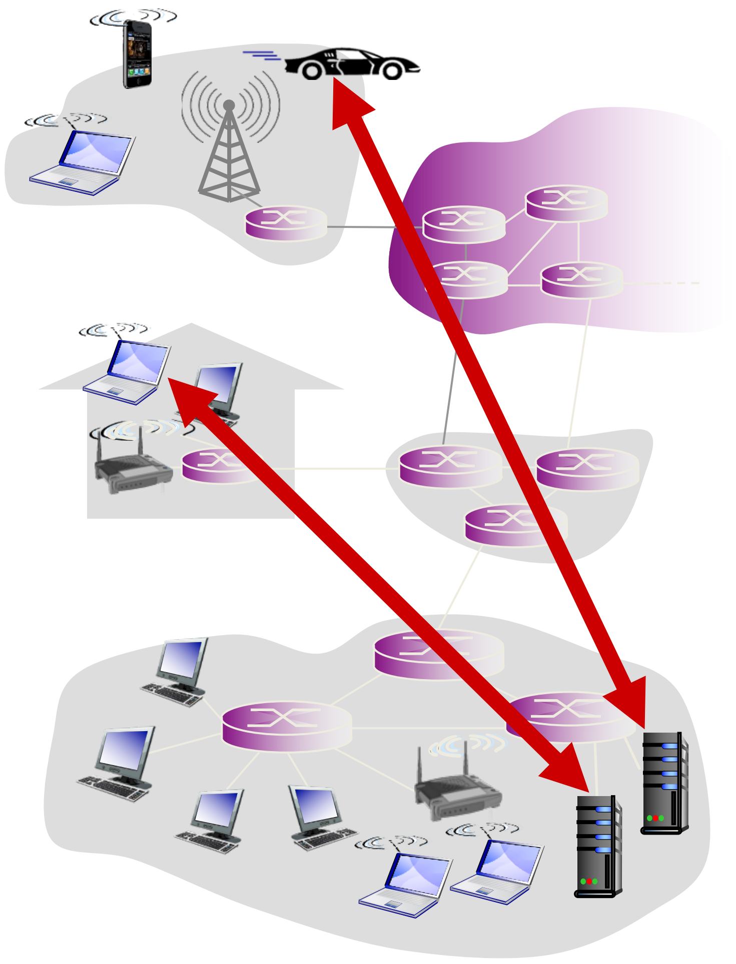
Client-Server Architecture

server:

- always-on host
- permanent IP address
- data centers for scaling

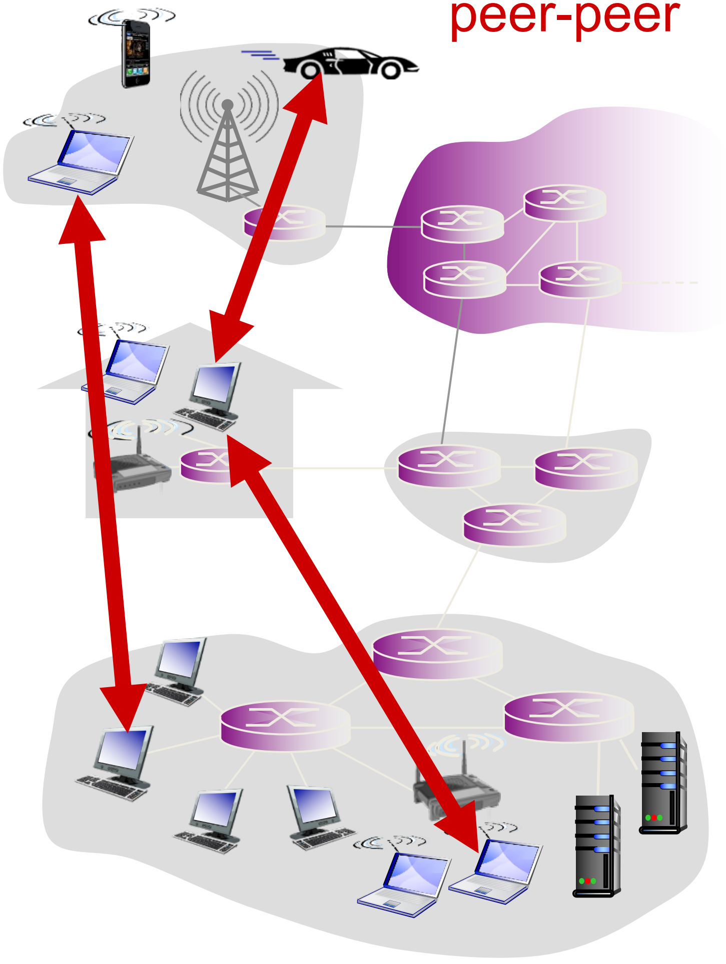
clients:

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



Peer-to-Peer Architecture

- no always-on server
- A peer talks directly with another peer
  - Symmetric responsibility (unlike client/server)
- peers request service from other peers, provide service in return to other peers
  - self scalability – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



In a peer-to-peer architecture, are there clients and servers?

A. Yes

B. No

If one machine can process requests at a rate of X per second, how quickly can two machines process requests?

A. Slower than one machine (<X)
B. The same speed (X)
C. Faster than one machine, but not double (X-2X)
D. Twice as fast (2X)
E. More than twice as fast(>2X)

On a single system…

- You have a number of components
  - CPU
  - Memory
  - Disk
  - Power supply

- If any of these go wrong, you're (usually) toast.

On multiple systems…

- New classes of failures (**partial failures**).
  - A link might fail
  - One (of many) processes might fail
  - The network might be partitioned

On multiple systems…

- New classes of failures (**partial failures**).
  - A link might fail
  - One (of many) processes might fail
  - The network might be partitioned

Introduces major complexity!

If a process sends a message, can it tell the difference between a slow link and a delivery failure?

A. Yes

B. No

What should we do to handle a partial failure? Under what circumstances, or what types of distributed applications?

A. If one process fails or becomes unreachable, switch to a spare.
B. Pause or shut down the application until all connectivity and processes are available.
C. Allow the application to keep running, even if not all processes can communicate.
D. Handle the failure in some other way.

Desirable Properties

- Consistency
  – Nodes agree on the distributed system's state
- Availability
  – The system is able and willing to process requests
- Partition tolerance
  – The system is robust to network (dis)connectivity

The CAP Theorem

- **C**onsistency
  - Nodes agree on the distributed system's state
- **A**vailability
  - The system is able and willing to process requests
- **P**artition tolerance
  - The system is robust to network (dis)connectivity
- Choose Two
- "CAP prohibits only a tiny part of the design space: perfect availability and consistency in the presence of partitions, which are rare."*

* Brewer, Eric. "CAP twelve years later: How the" rules" have changed." Computer 45.2 (2012): 23-29.

Event Ordering

- It’s very useful if all nodes can agree on the order of events in a distributed system
- For example: Two users trying to update a shared file across two replicas

If two events occur (digitally or in the "real world"), can we always tell which happened first?

A. Yes

B. No

Event Ordering

- It’s very useful if all nodes can agree on the order of events in a distributed system

- For example: Two users trying to update a shared file across two replicas

- “Time, Clocks, and the Ordering of Events in a Distributed System” by Leslie Lamport (1978)
  – Establishes causal orderings
  – Cited > 8000 times

Causal Consistency Example

- Suppose we have the following scenario:
  - Sally posts to Facebook, “Bill is missing!”
  - (Bill is at a friend’s house, sees message, calls mom)
  - Sally posts new message, “False alarm, he’s fine”
  - Sally’s friend James posts, “What a relief!”

Causal Consistency Example

- Suppose we have the following scenario:
  - Sally posts to Facebook, “Bill is missing!”
  - Sally’s friend James posts, “What a relief!
- NOT causally consistent:
  - Third user, Henry, sees only:
  - Sally posts to Facebook, “Bill is missing!”
  - Sally’s friend James posts, “What a relief!”

Causal Consistency Example

- Suppose we have the following scenario:
  1. Sally posts to Facebook, “Bill is missing!” (Bill is at a friend’s house, sees message, calls mom)
  2. Sally posts new message, “False alarm, he’s fine”
  3. Sally’s friend James posts, “What a relief!”

- Causally consistent version:
  – Because James had seen Sally’s second post (which caused his response), Henry must also see it prior to seeing James’s.

Summary

- Client-server vs. peer-to-peer models
- Distributed systems are hard to build!
  - Partial failures
  - Ordering of events
- Take CS 87 for more details!

Today

- Identifiers and addressing

- Domain Name System
  - Telephone directory of the Internet
  - Protocol format
  - Caching: Load balancing
  - Security Challenges

DNS: Domain Name System

People: many identifiers:

- name, swat ID, SSN, passport #

Internet hosts (endpoints), routers (devices inside a n/w):

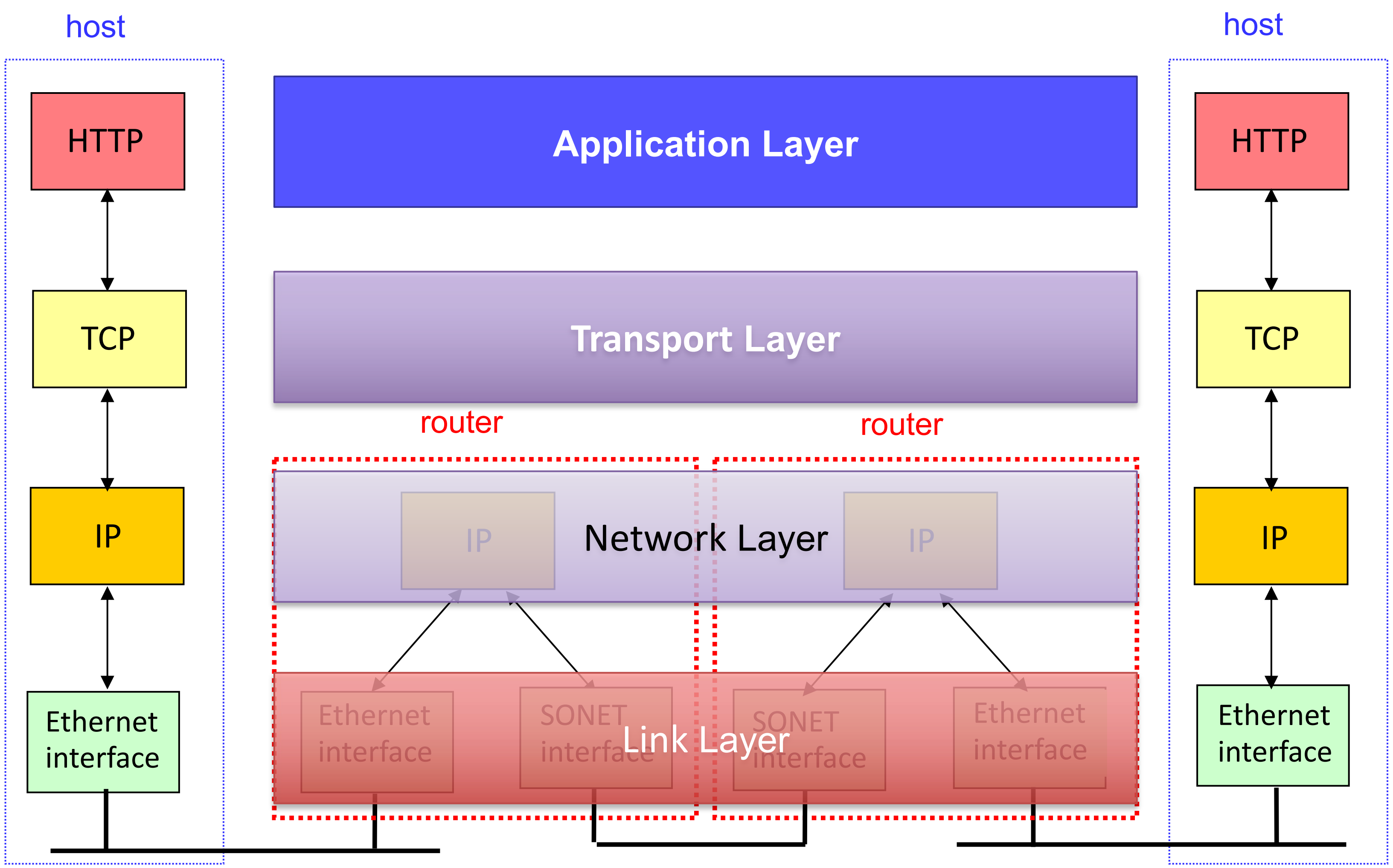
- "name", e.g., www.google.com - used by humans
- IP address (32 bit) - used for addressing packets

How do we map between IP address and name, and vice versa ?

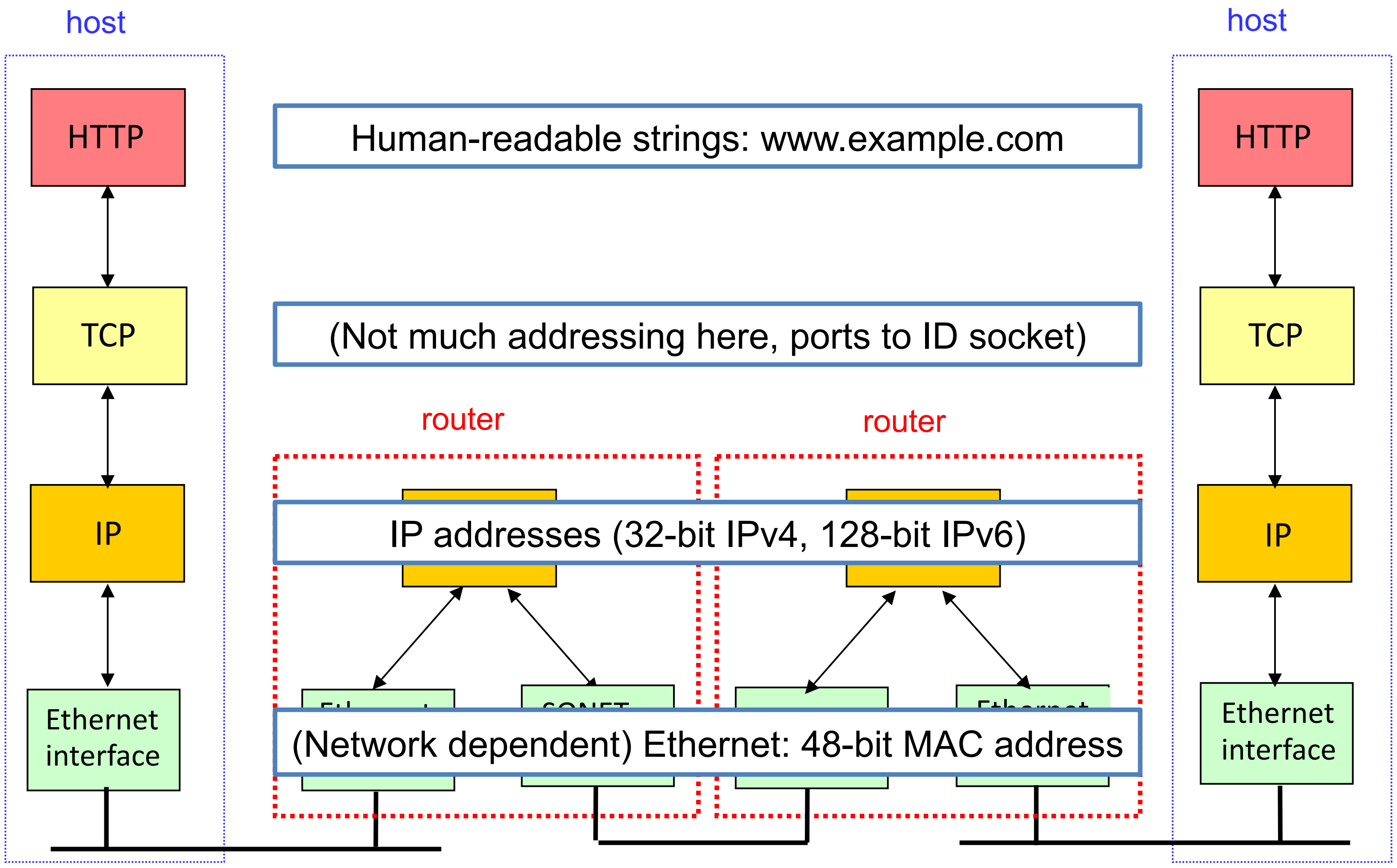
DNS: Application Layer Protocol

- distributed database
  - implemented in hierarchy of many name servers.
- application-layer protocol:
  - hosts communicate to name servers
  - resolve names → addresses
- *note: core Internet function, implemented as application-layer protocol*

Where


host
HTTP
TCP
IP
Ethernet interface
Application Layer
Transport Layer
router
router
IP
Network Layer
IP
Ethernet interface
SONET interface
Link Layer
SONET interface
Ethernet interface
host
HTTP
TCP
IP
Ethernet interface

Recall: TCP/IP Protocol Stack



DNS: domain name system

- distributed database implemented in hierarchy of many name servers.
- application-layer protocol: hosts, name servers communicate to resolve names → addresses
  - *note: core Internet function, implemented as application-layer protocol*
  - *complexity at network's "edge"*

Why do we need to map names to IP addresses? Why not route on names at the network layer?

A. Domain names are hierarchical, so we can route on domain names too.
B. Domain names are variable length, vs IP are fixed length, some changes will be required to switch.
C. With domain names we wouldn’t know where to route to geographically.
D. Some other reason.

Why do we need to map names to IP addresses? Why not route on names at the network layer?

A. Domain names are hierarchical, so we can route on domain names too (Named Data Networking Efforts).

B. Domain names are variable length, vs IP are fixed length, some changes will be required to switch.

C. With domain names we wouldn't know where to route to geographically (mostly true).

D. Some other reason.

Identifiers

- Host name (e.g., www.swarthmore.edu)
  - Used by humans to specify host of interest
  - Unique, selected by host administrator
  - Hierarchical, variable-length string of alphanumeric characters
- IP address (e.g., 130.58.68.164)
  - Used by routers to forward packets
  - Unique, topologically meaningful locator
  - Hierarchical namespace of 32 bits

Mapping Between Identifiers

- Domain Name System (DNS)
  - Given a host name, provide the IP address
  - Given an IP address, provide the host name

What's the biggest challenge for DNS?

A. It's old.

B. The fact that the Internet is global.

C. The fact that DNS is now critical infrastructure.

D. The sheer number of name lookups happening at any given time.

E. How and when the name -> IP address mapping should change.

What's the biggest challenge for DNS?

A. It's old.

B. The fact that the Internet is global.

C. The fact that DNS is now critical infrastructure.

D. The sheer number of name lookups happening at any given time.

E. How and when the name -> IP address mapping should change.

In the old days…

- Pre-1982, everyone downloads a "hosts.txt" file from SRI
- Pre-1998, Jon Postel, researcher at USC, runs the Internet Assigned Numbers Authority (IANA)
  - RFCs 882 & 883 in 1983
  - RFCs 1034 & 1035 in 1987

Emailed 8/12 root DNS servers, asked change to his authority. They did.

http://www.wired.com/wiredenterprise/2012/10/joe-postel/

Since 1998…

- Control of Internet Assigned Numbers Authority (IANA) transferred to Internet Corporation for Assigned Names and Numbers (ICANN)
  - ICANN is a private non-profit (formerly) blessed by US DOC
  - Global advisory committee for dealing with international issues
  - 2000's: Many efforts for UN control, US resisted
  - 2016: ICANN no longer partnered with DOC

Who should control DNS?

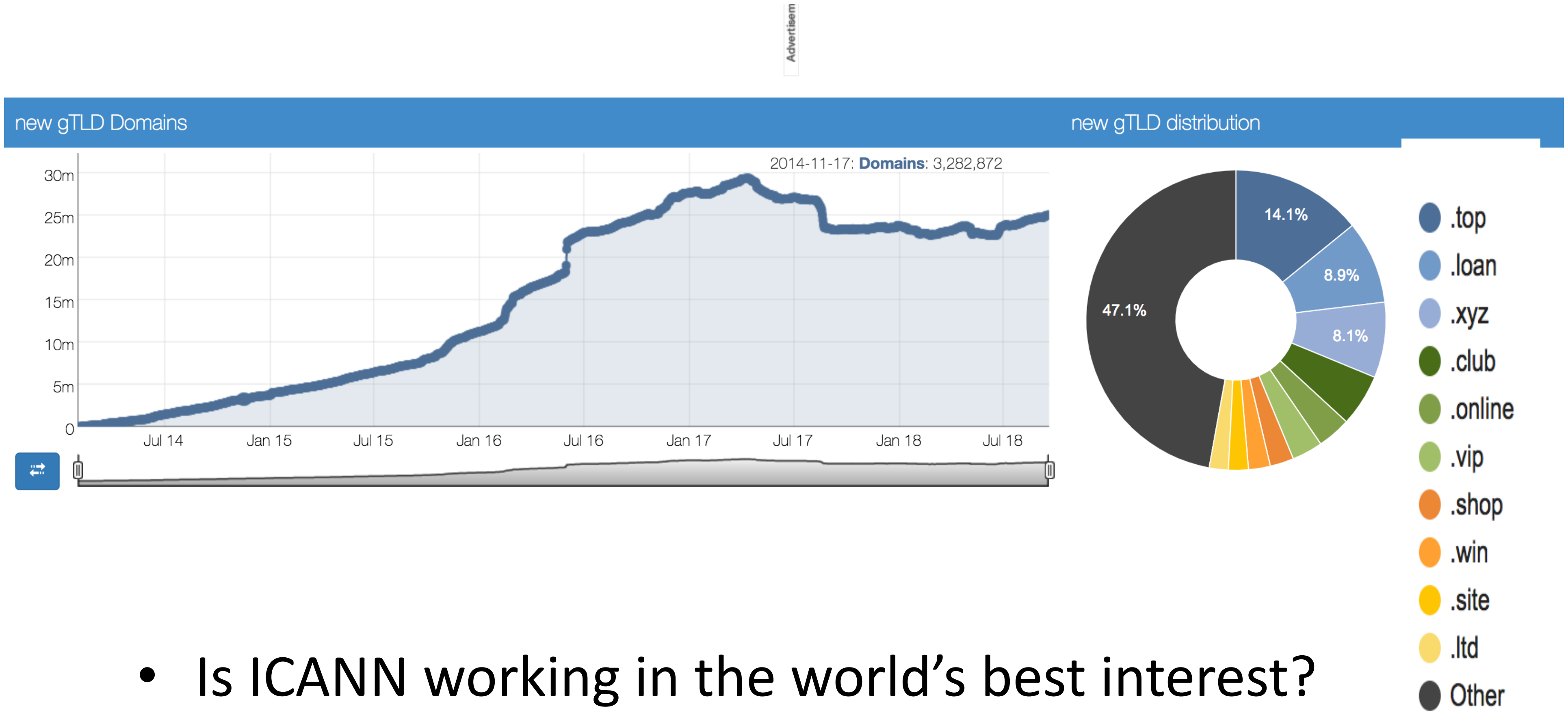
A. US government

B. UN / International government

C. Private corporation

D. Someone else

Recent Controversy



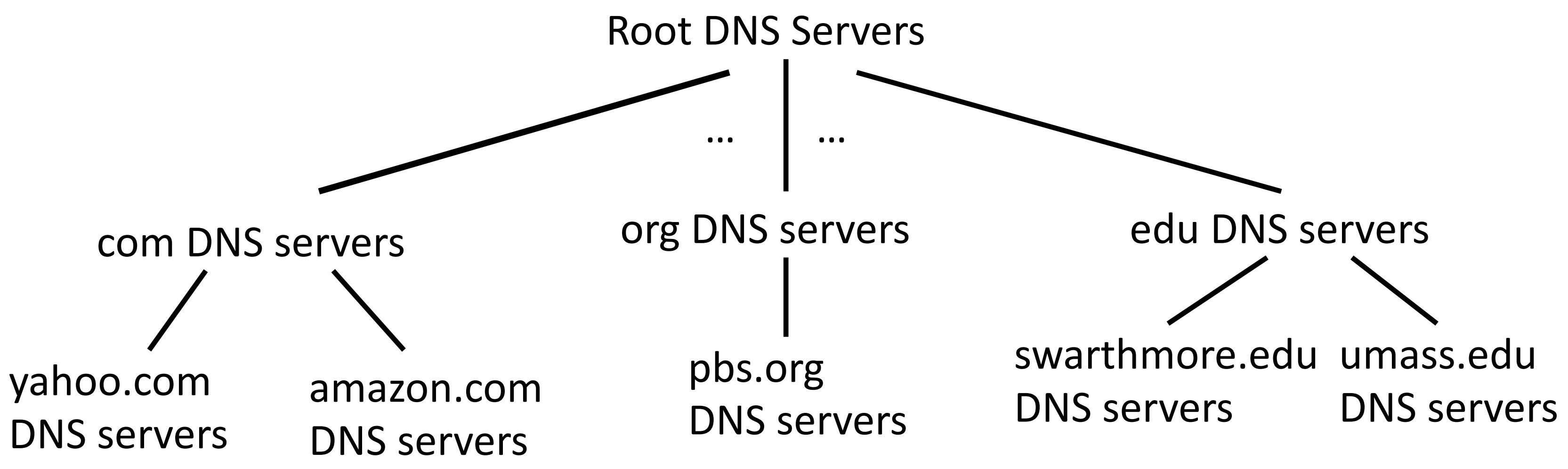
- Is ICANN working in the world's best interest?

- New "top level domains" added, for auction

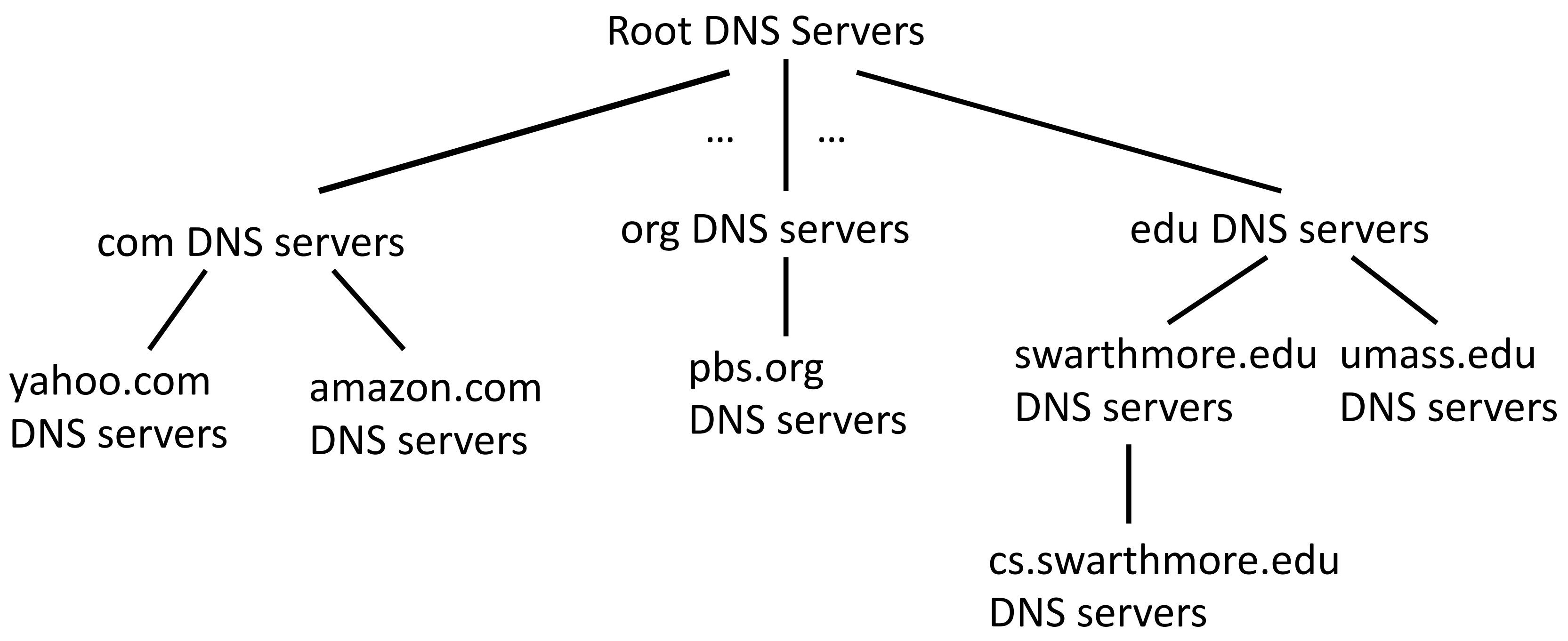
DNS Services

- DNS is an application-layer protocol. E2E design!
- It provides:
  - Hostname to IP address translation
  - Host aliasing (canonical and alias names)
  - Mail server aliasing
  - Load distribution (one name may resolve to multiple IP addresses)
  - Lots of other stuff that you might use a directory service to find. (Wikipedia: List of DNS record types)

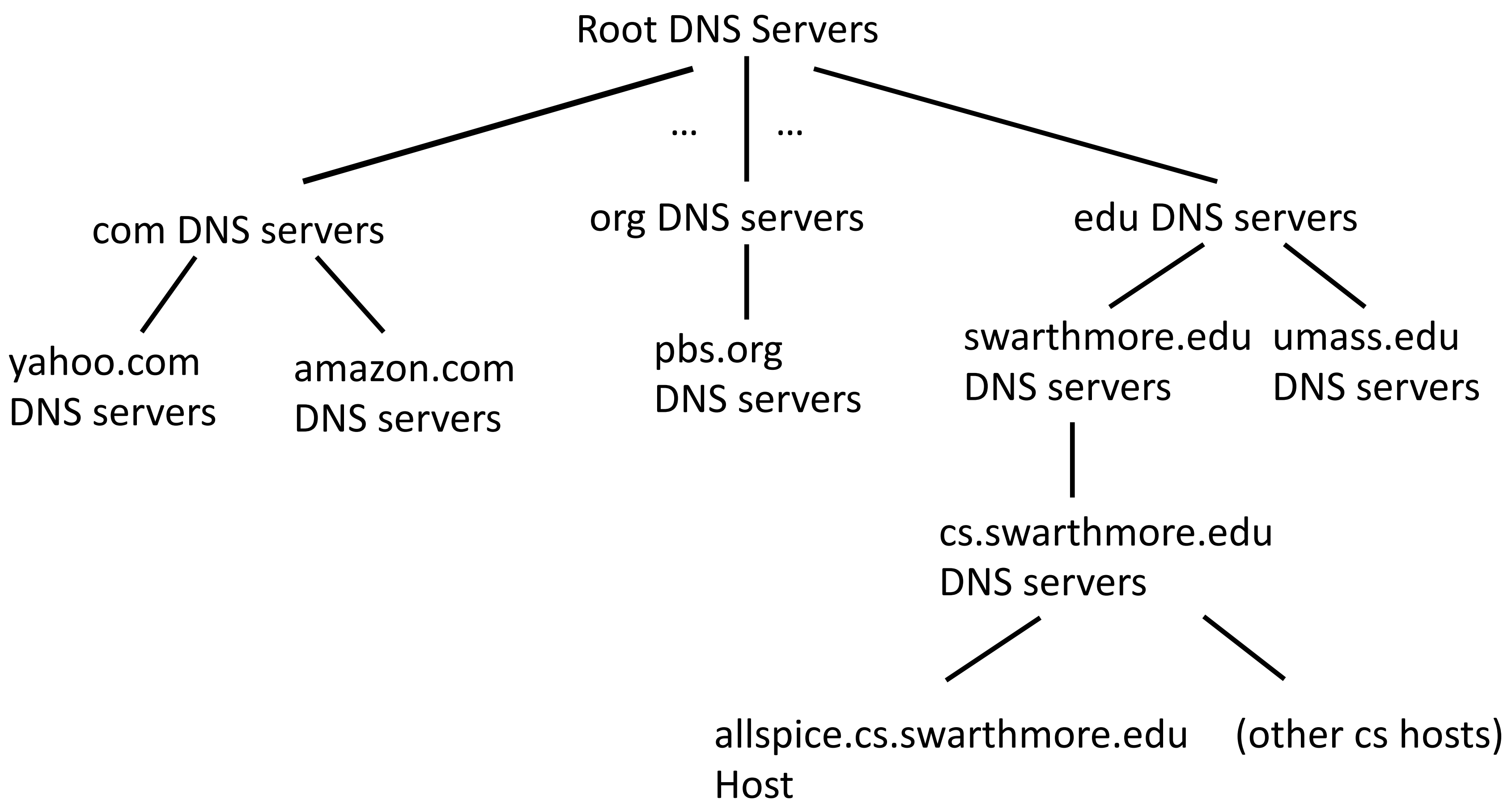
DNS: a distributed, hierarchical database



DNS: a distributed, hierarchical database



DNS: a distributed, hierarchical database



- allspice.cs.swarthmore.edu.

Nameless root,
Usually implied.

Domain Name System (DNS)

- Distributed administrative control
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers

- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers

- Performing the translations
  - Local DNS servers
  - Resolver software

Why do we structure DNS like this? Which of these helps the most? Drawbacks?

A. It divides up responsibility among parties.

B. It improves performance of the system.

C. It reduces the size of the state that a server needs to store.

D. Some other reason.

Why do we structure DNS like this? Which of these helps the most? Drawbacks?

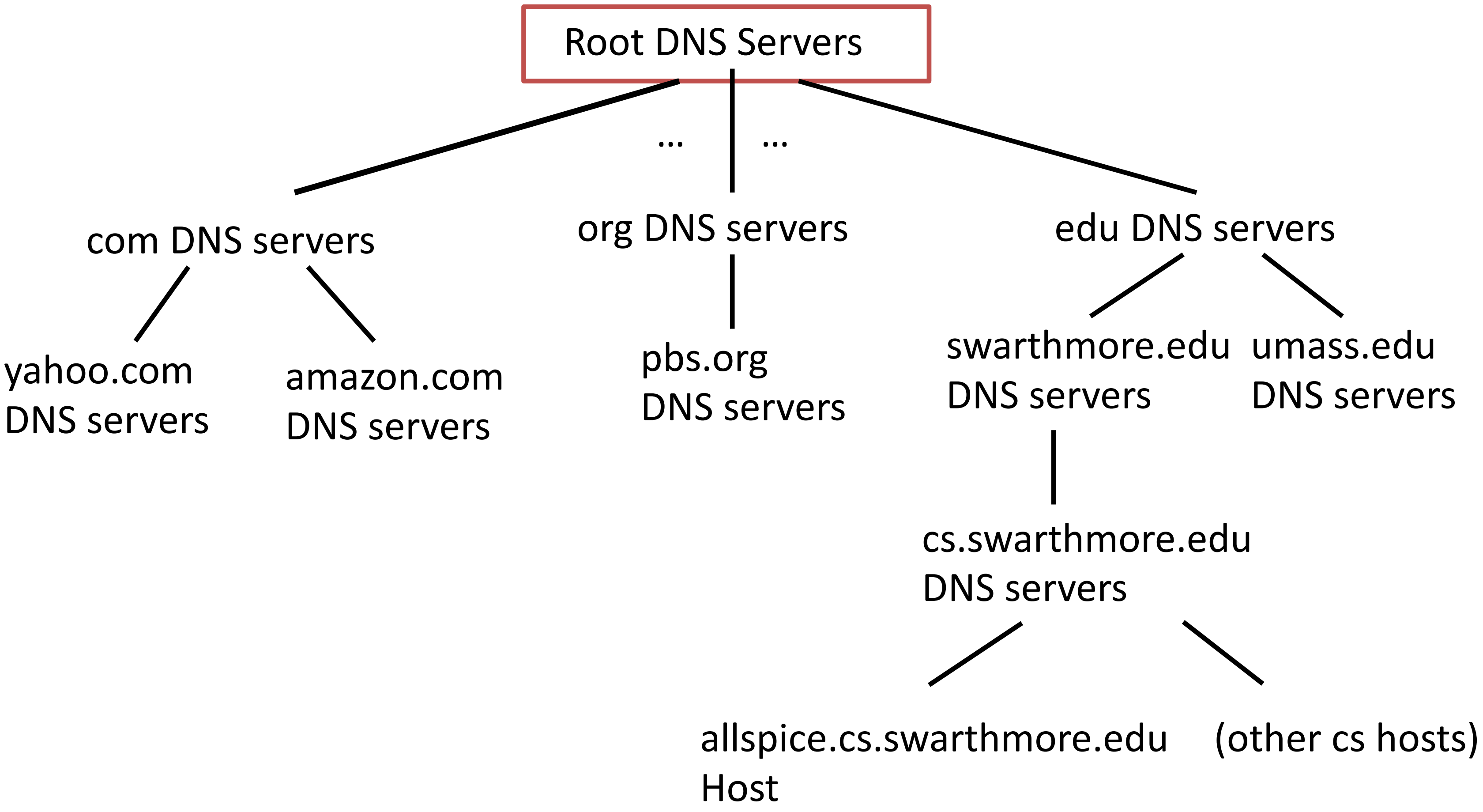
A. It divides up responsibility among parties.

B. It improves performance of the system overall but individual end hosts (assuming no caching) have a look-up overhead of traversing the hierarchy .

C. It reduces the size of the state that a server needs to store.

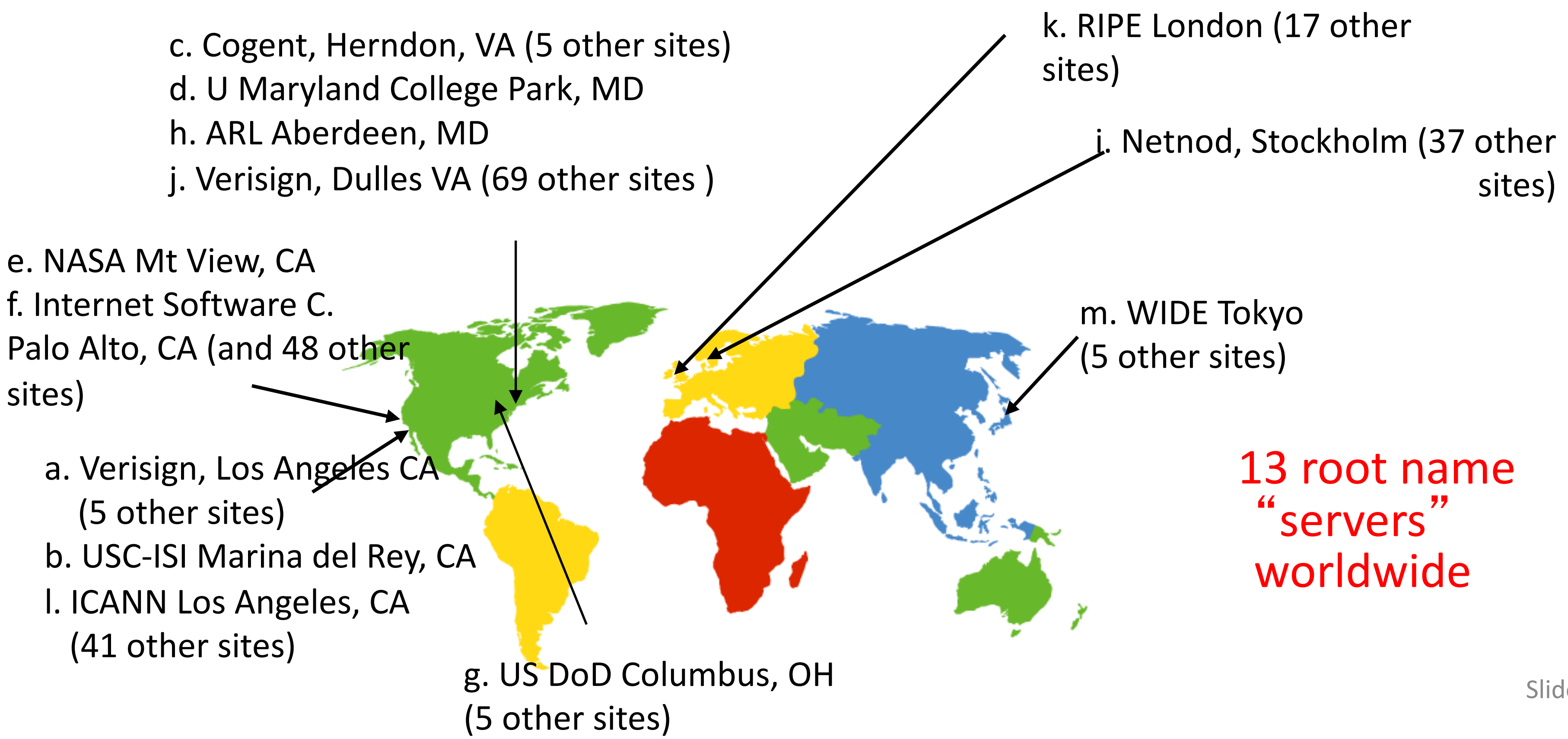
D. Some other reason.

DNS: a distributed, hierarchical database



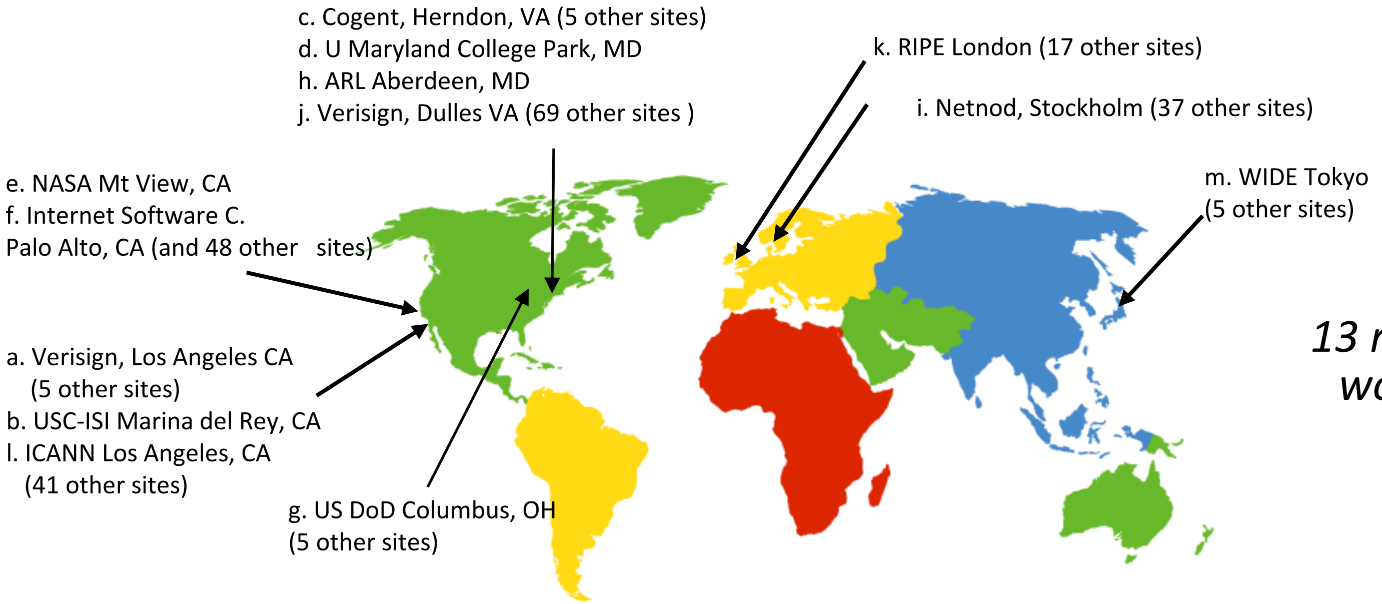
DNS: Root Name Servers

- Root name server:
  - Knows how to find top-level domains (.com, .edu, .gov, etc.)
  - How often does the location of a TLD change?



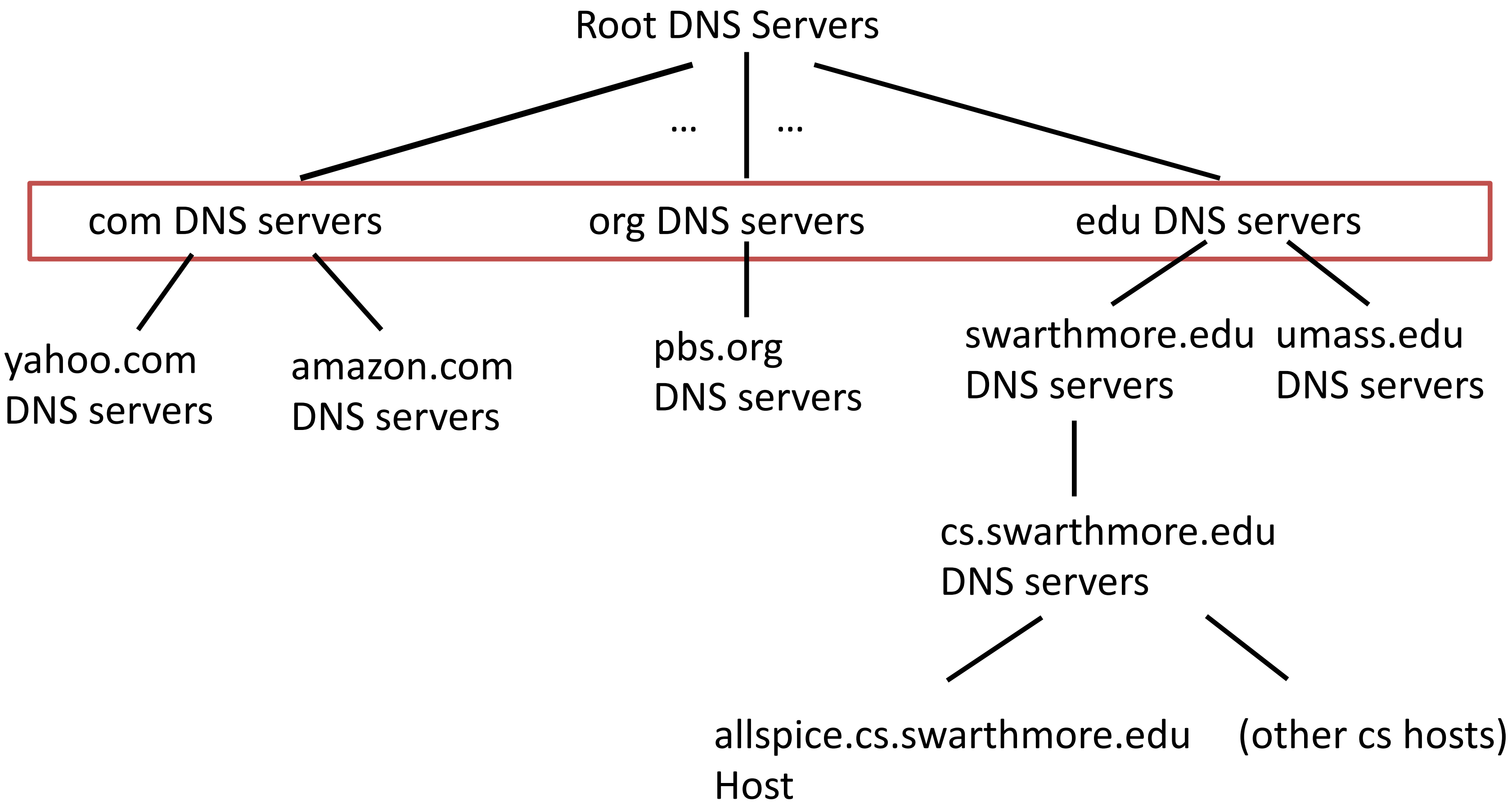
DNS: Root Name Servers

- Root name server:
  - Knows how to find top-level domains (.com, .edu, .gov, etc.)
  - How often does the location of a TLD change?
  - approx. 400 total root servers
  - Significant amount of traffic is not legitimate



13 root name "servers" worldwide

DNS: a distributed, hierarchical database

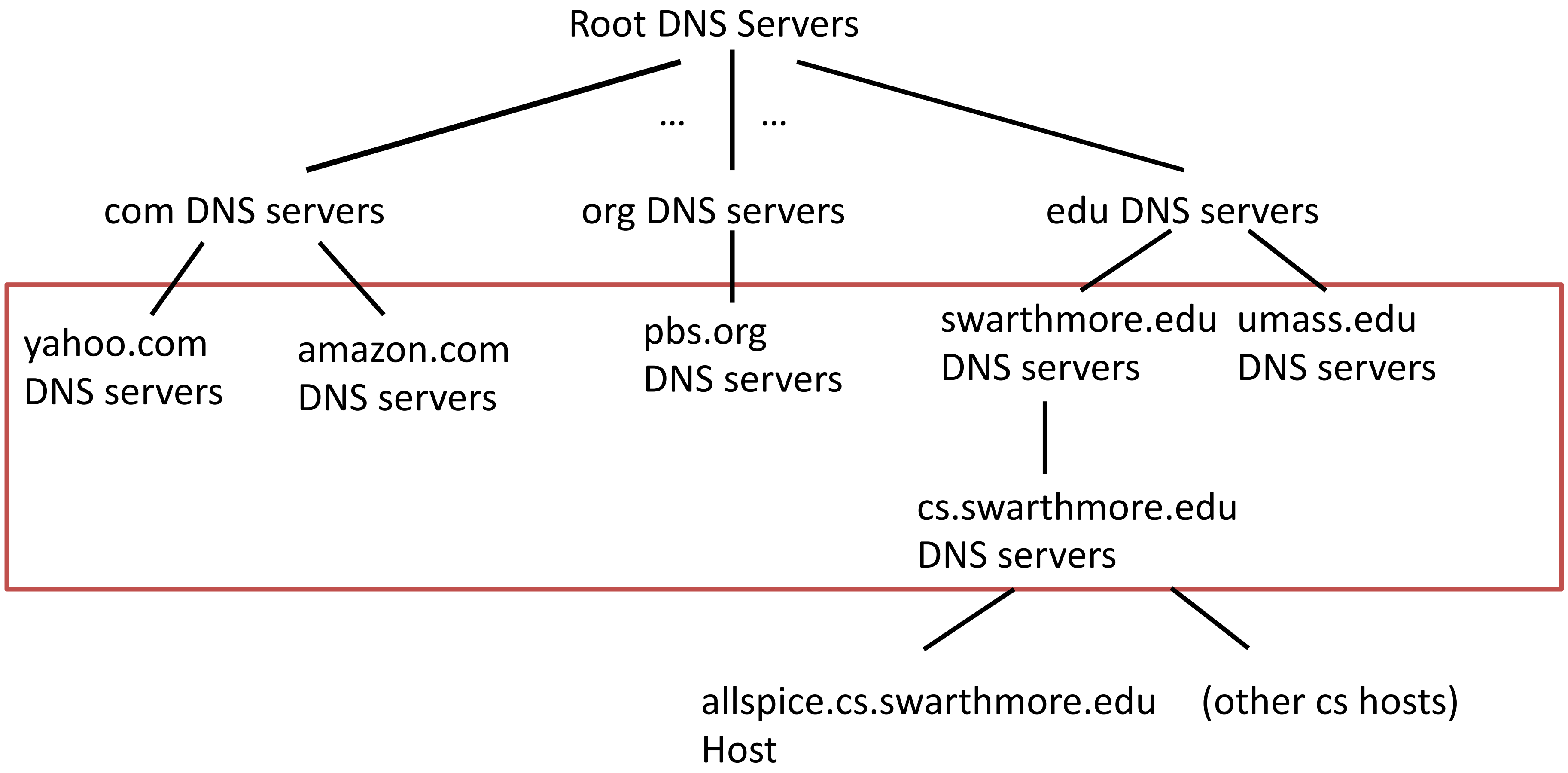


Top Level Domains

Top-level domain (TLD) servers:

- Responsible for com, org, net, edu, gov, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, de, ca, jp, etc.
- Verisign maintains servers for .com and .net TLD
- Educause for .edu TLD (Verisign actually runs backend)
- Others managed by corresponding entity (e.g., local governments or companies)

DNS: a distributed, hierarchical database



Authoritative Servers

Authoritative DNS servers:

- Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- Can be maintained by organization or service provider, easily changing entries
- Often, but not always, acts as organization's local name server (for responding to look-ups)

Resolution Process

- End host wants to look up a name, who should it contact?
  - It could traverse the hierarchy, starting at a root
  - More efficient for ISP to provide a local server

- ISP's local server for handling queries not necessarily a part of the pictured hierarchy

Local DNS Name Server

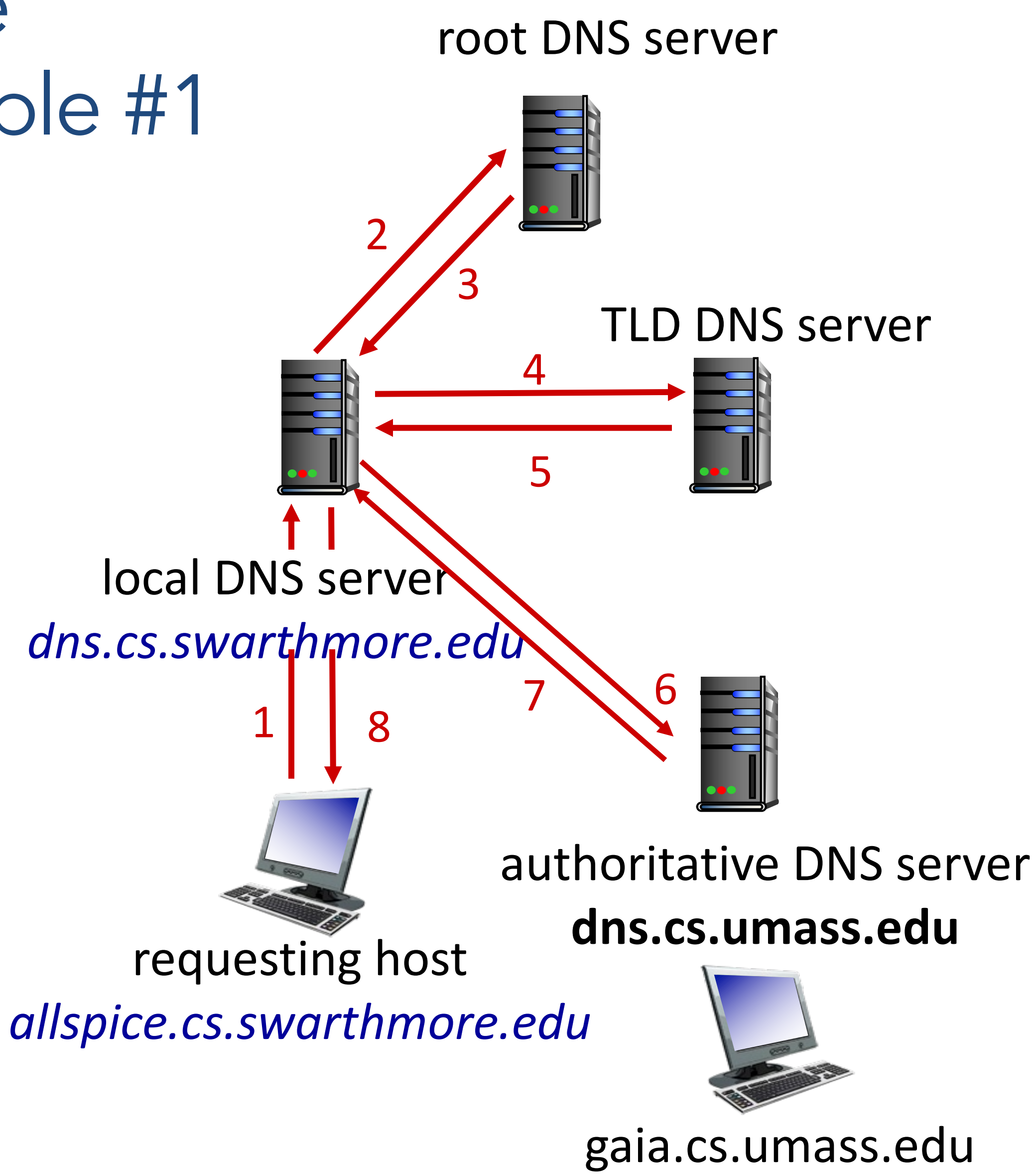
- Each ISP (residential ISP, company, university) has (at least) one
  - also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution example #1

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

iterative query:

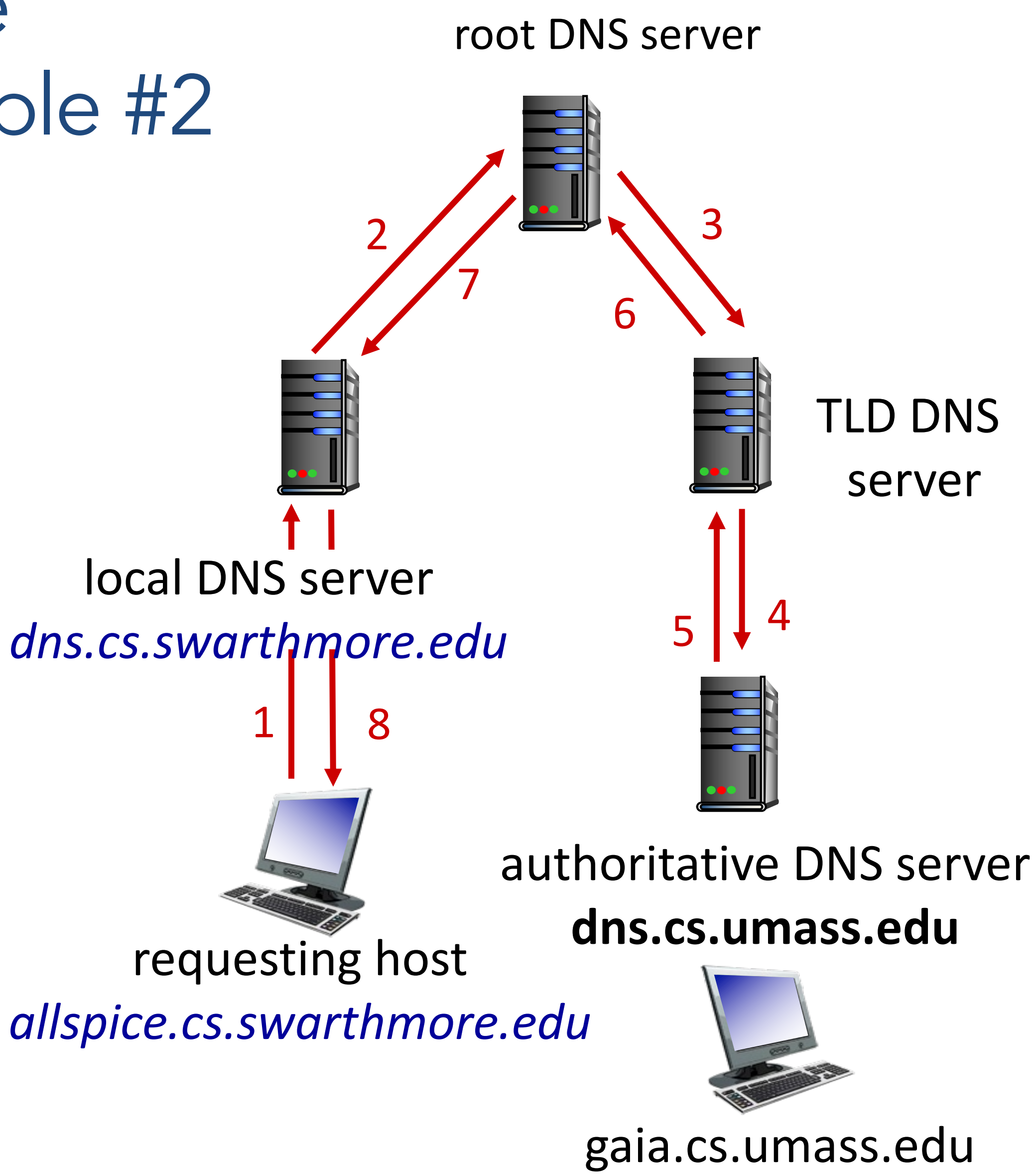
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example #2

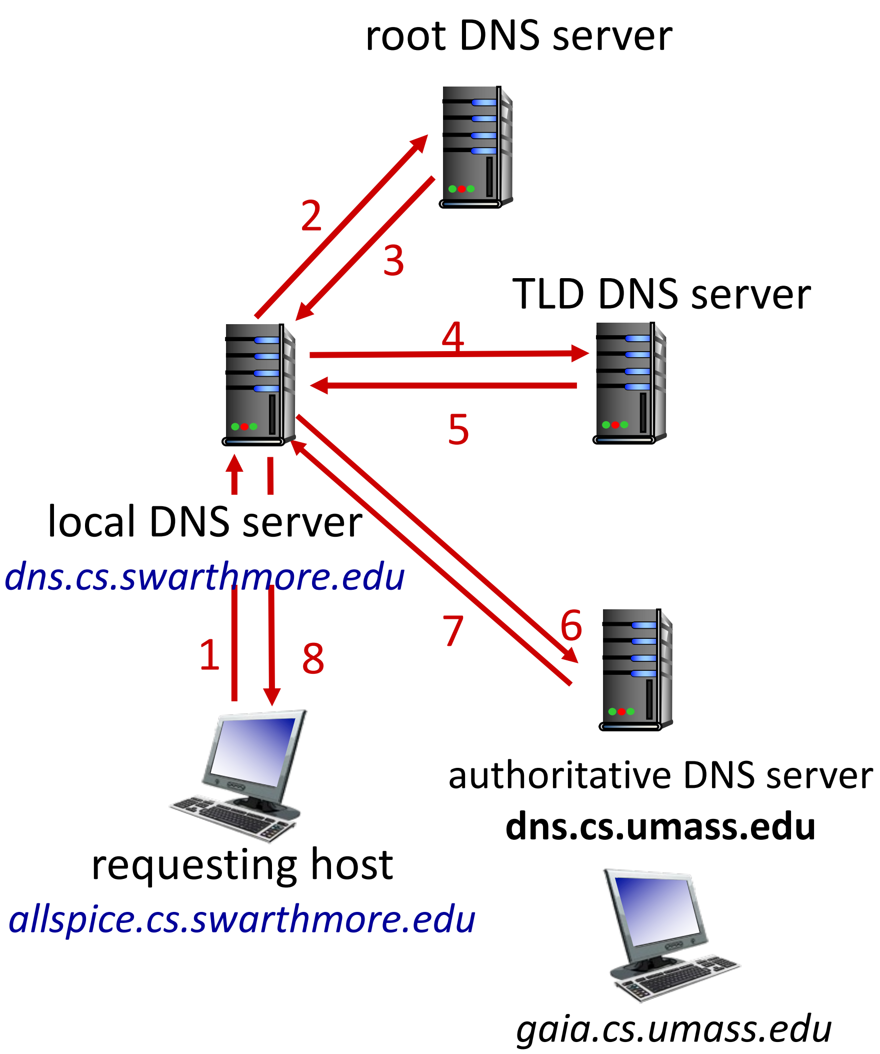
recursive query:

- each server asks the next one, in a chain

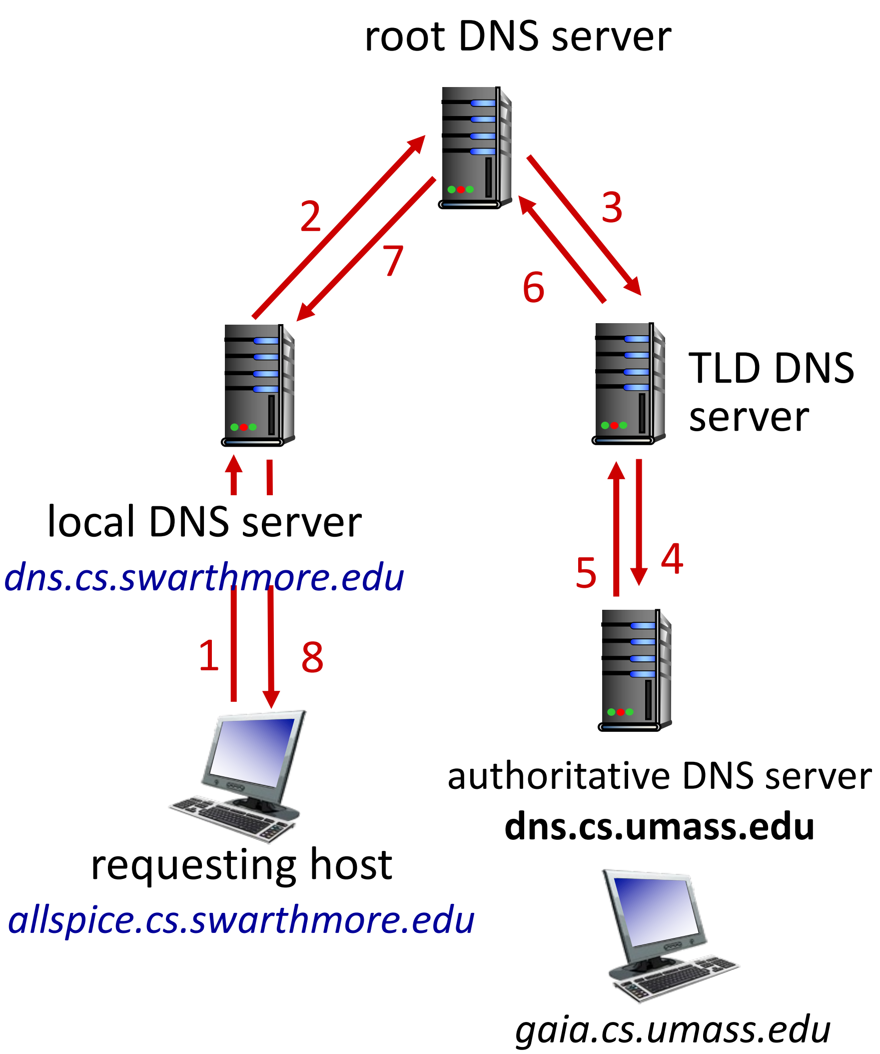


Which would you use? Why?

A. Iterative



B. Recursive



Example: iterative query using dig()

```
dig . ns

dig +norec demo.cs.swarthmore.edu @a.root-servers.net

dig +norec demo.cs.swarthmore.edu @a.edu-servers.net

dig +norec demo.cs.swarthmore.edu @ibext.its.swarthmore.edu

    demo.cs.swarthmore.edu.  259200 IN  A   130.58.68.26
```

Caching

- Once (any) name server learns a mapping, it caches mapping
  - cache entries timeout (disappear) after some time (TTL: time to live)
  - TLD servers typically cached in local name servers
  - Thus root name servers not often (legitimately) visited

Caching

- Once (any) name server learns a mapping, it caches mapping
  - cache entries timeout (disappear) after some time (TTL: time to live)
  - TLD servers typically cached in local name servers.
  - Root name servers not often (legitimately) visited
- (+) Subsequent requests need not burden DNS
- (-) Cached entries may be out-of-date (best effort!)
  - If host's name or IP address changes, it may not be known Internet-wide until all TTLs expire

The TTL value should be…

A. Short, to make sure that changes are accurately reflected

B. Long, to avoid re-queries of higher-level DNS servers

C. Something else