Lecture 14 (Applications 1)

DNS

CSC 175, Spring 2026

Slides credit: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao, Nicholas Ngai, Nicholas Weaver

What is DNS For?

Lecture 14, CS 168, Spring 2026

DNS

- **What is DNS For?**
- Design
- Scaling
- Other Uses

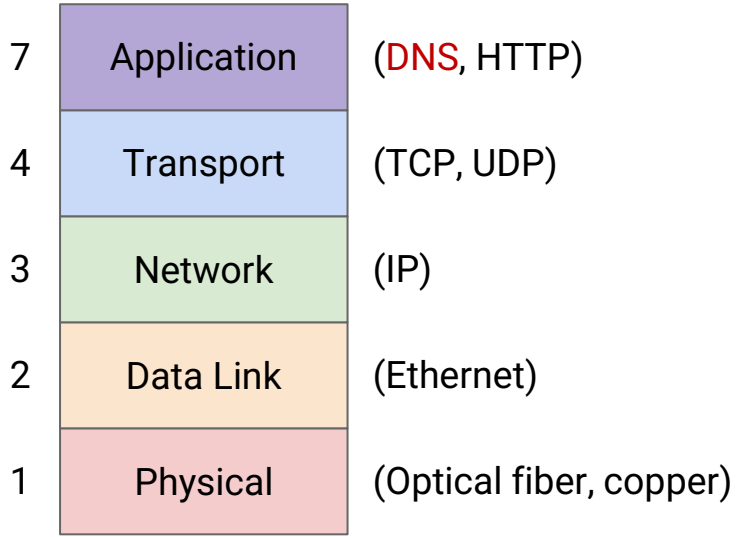
Domain Names

Recall: Computers are addressed by IP address.

- Example: 74.125.25.99.
- Useful for machines: Helps making routing scalable.
- Not useful for humans: Numbers not meaningful to humans. Hard to remember.

Alternative addressing system: Human-readable domain names.

- Example: www.google.com.
- Not useful for machines: Contains no relevant routing information.
- Useful for humans: Meaningful words and phrases, easy to remember.

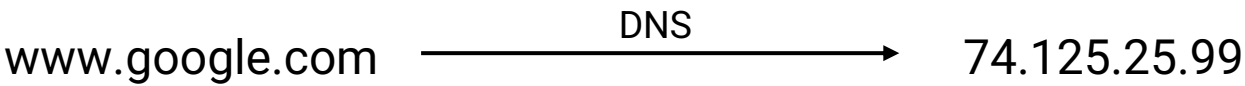


DNS: Definition

DNS (Domain Name System): An Internet protocol for translating human-readable domain names to IP addresses.

Usage:

- You want to send a packet to a certain domain (e.g. you type a domain into your browser).
- Your computer performs a **DNS lookup** to translate the domain name to an IP address.
- Your computer sends the packet to the corresponding IP address.



Goals of DNS

DNS must be **scalable**.

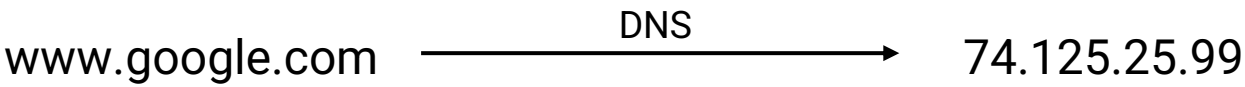
- Many hosts/names. Many lookups. Many updates.

DNS must be **highly available**.

- No single point of failure.

DNS must be **lightweight** and **fast**.

- Connections often start with a name lookup (e.g. user typing domain in browser).
- If DNS is slow, every connection is slow.



DNS Design

Lecture 14, CS 168, Spring 2026

DNS

- What is DNS For?
- **Design**
- Scaling
- Other Uses

DNS Name Servers

Name server: A server on the Internet responsible for answering DNS requests.

- Name servers have domain names and IP addresses too.
- Example: There's a name server with name a.edu-servers.net and IP 192.5.6.30.

DNS is a client-server, request-response protocol.

- To perform a DNS lookup, you (the client) send a DNS query:
  "What is the IP address of www.google.com?"
- The name server sends a DNS response with the answer:
  "The IP address of www.google.com is 74.125.25.99."

Issues:

- One name server can't handle every DNS request from the entire Internet.
- If there are many name servers, how do you know which one to contact?

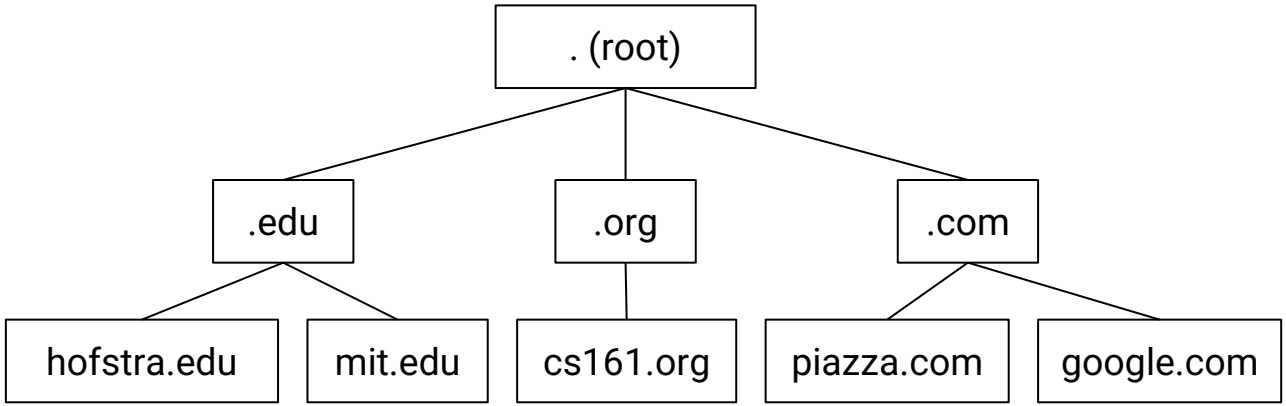
DNS Name Server Hierarchy

Idea #1: If one name server doesn't know the answer to your query, the name server can direct you to another name server.

- Analogy: If I don't know the answer to your question, I will direct you to a friend who can help.

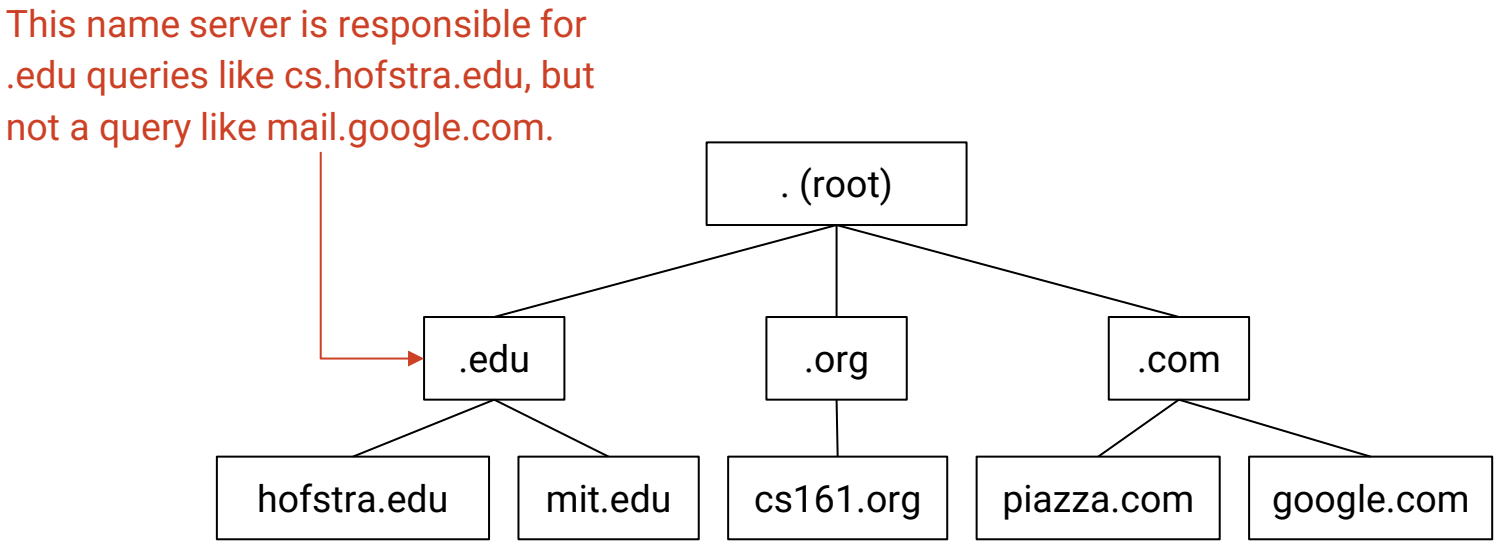
Idea #2: Arrange the name servers in a tree hierarchy.

- Intuition: Name servers will direct you down the tree until you receive the answer to your query.



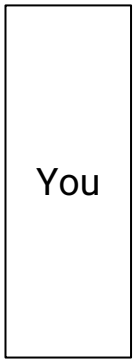
DNS Name Server Hierarchy

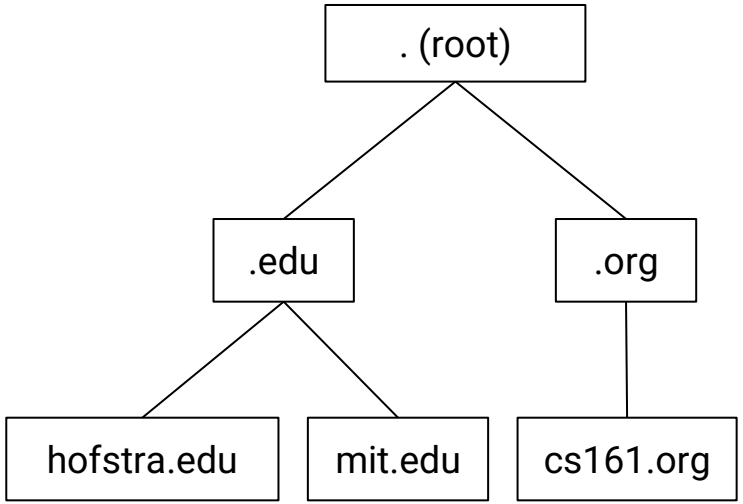
Each box is a name server. The label represents which queries the name server is responsible for answering.



Steps of a DNS Lookup

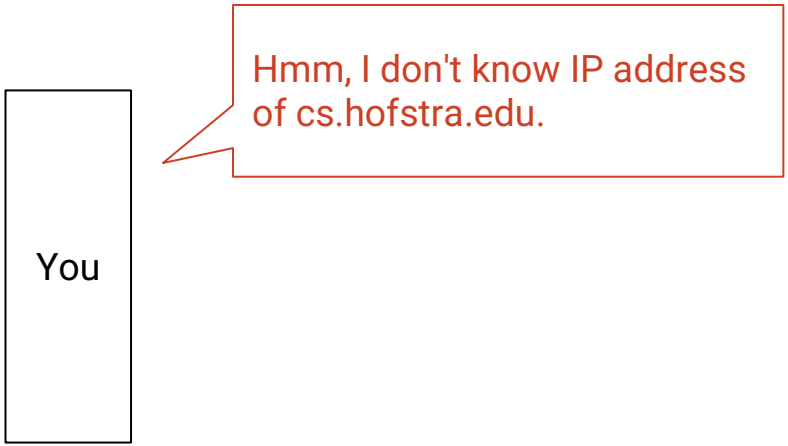
Let's walk through a DNS query for the IP address of cs.hofstra.edu.

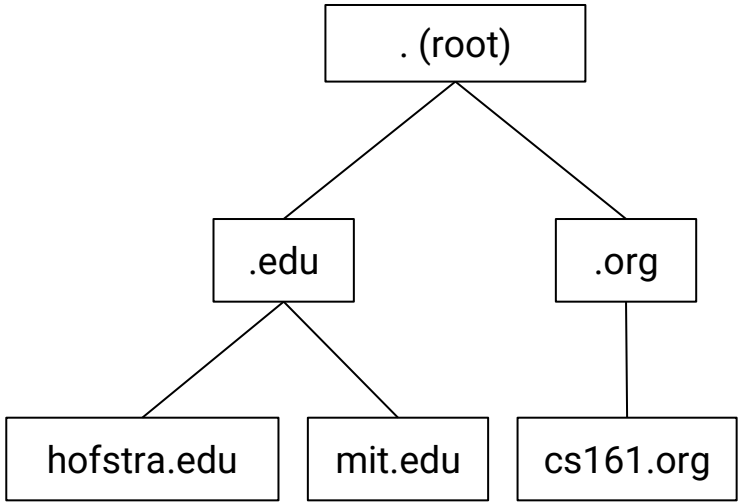




Steps of a DNS Lookup

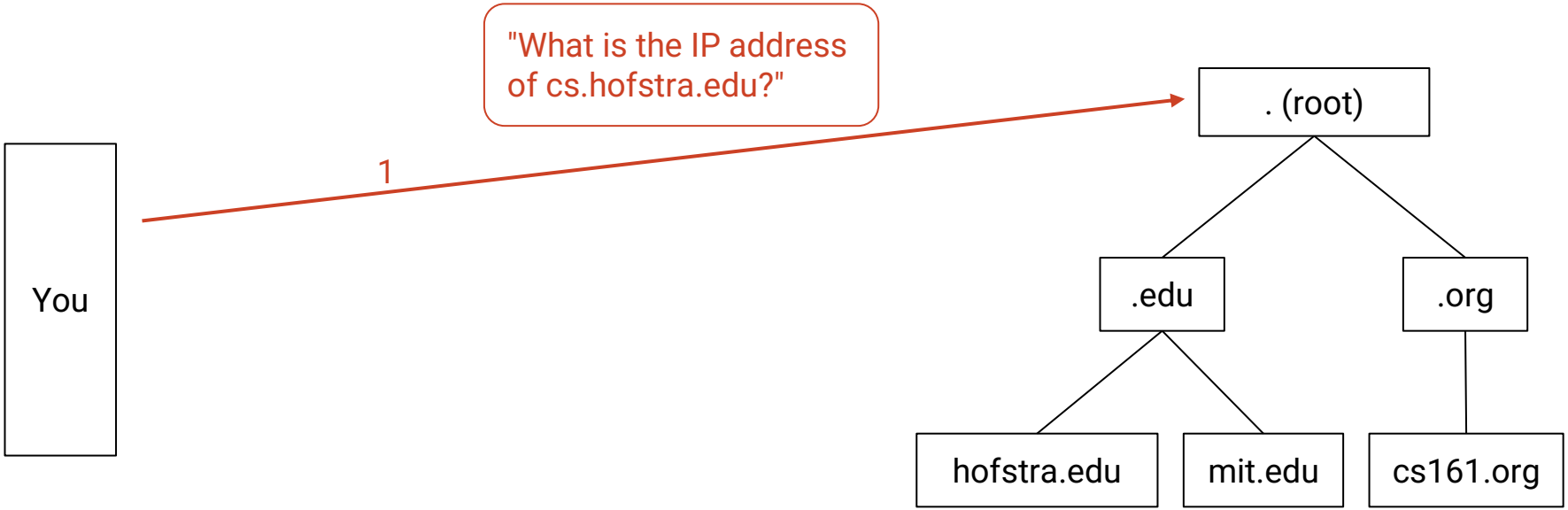
First, you check your cache to see if you already know the answer to your query.





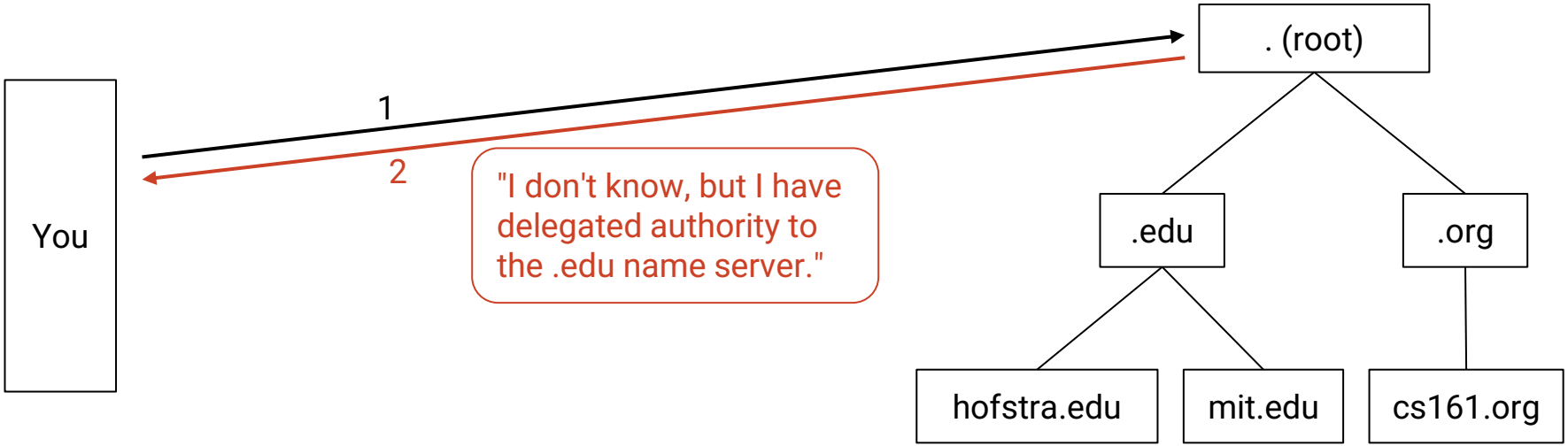
Steps of a DNS Lookup

DNS queries always start with a request to the root name server, which is responsible for all requests.



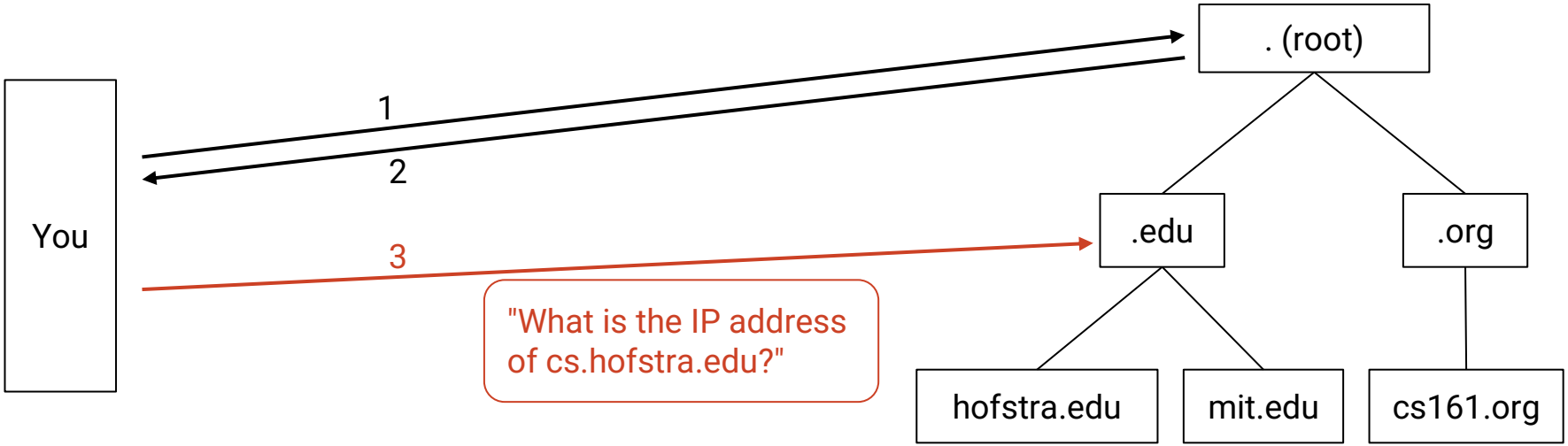
Steps of a DNS Lookup

The root name server responds by directing you to the correct child name server (in this case, the .edu name server).



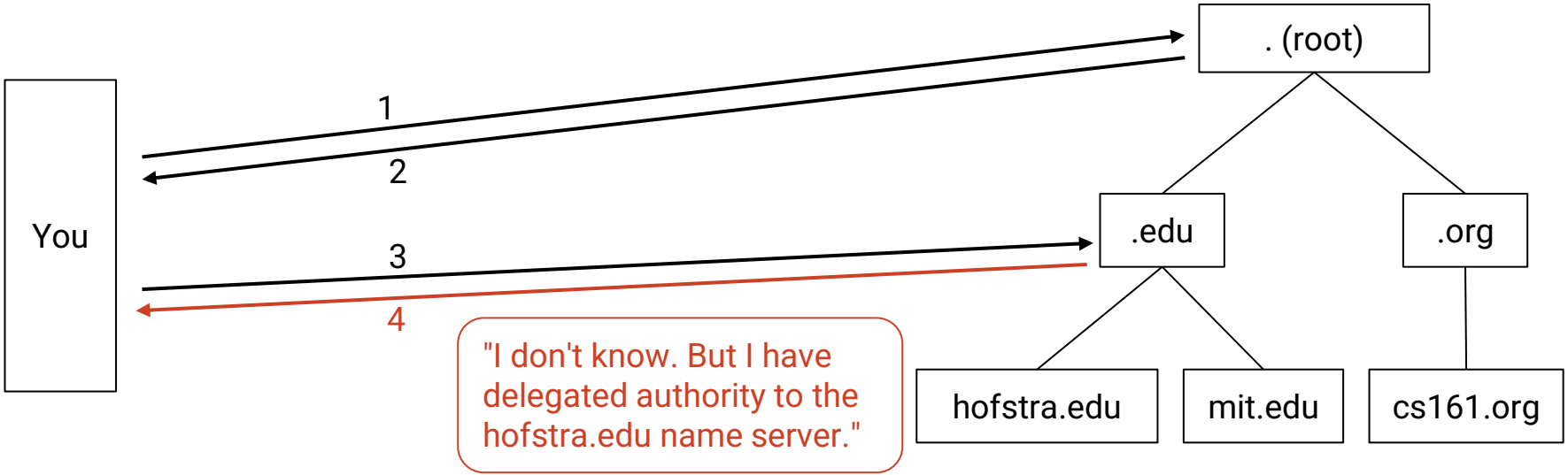
Steps of a DNS Lookup

Next, you ask the same question, but now to the .edu name server.



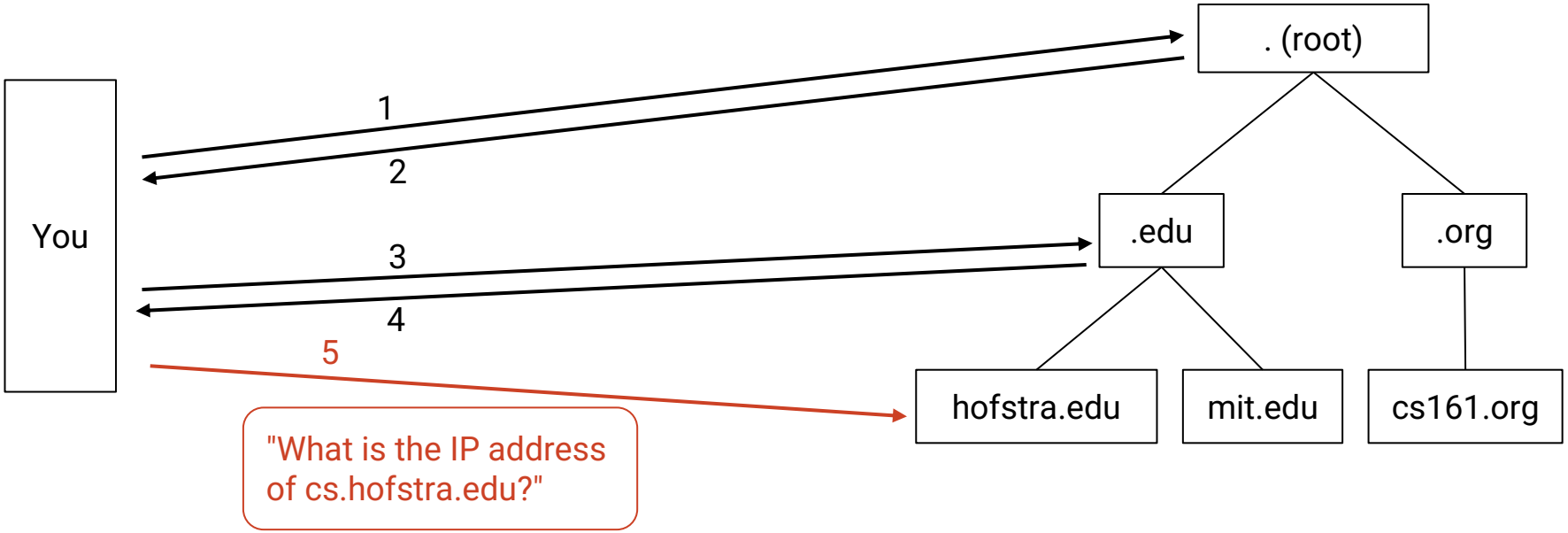
Steps of a DNS Lookup

The .edu name server redirects you to the hofstra.edu name server.



Steps of a DNS Lookup

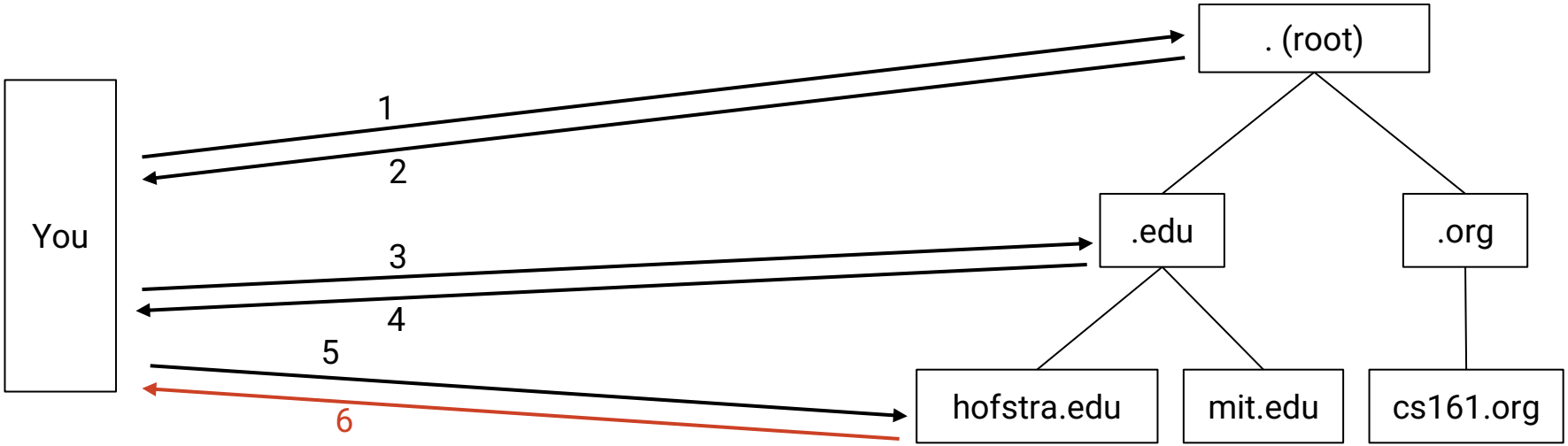
Next, you ask the same question, but now to the hofstra.edu name server.



Steps of a DNS Lookup

The hofstra.edu name server responds with the answer.

You can cache the answer to avoid having to ask again later.



Stub Resolvers and Recursive Resolvers

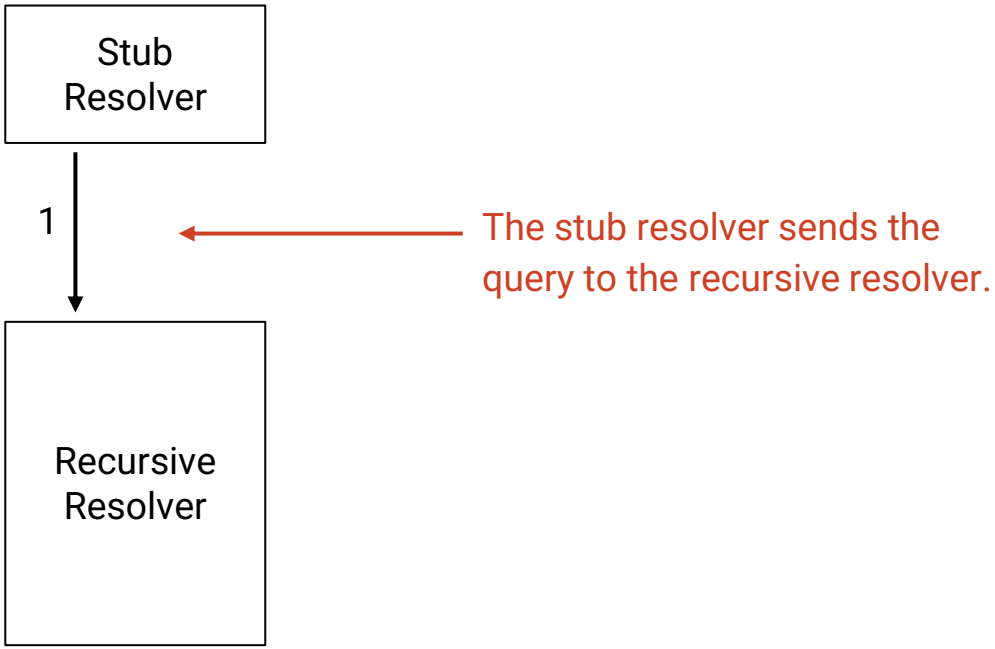
In practice, your computer usually tells another resolver to perform the query for you.

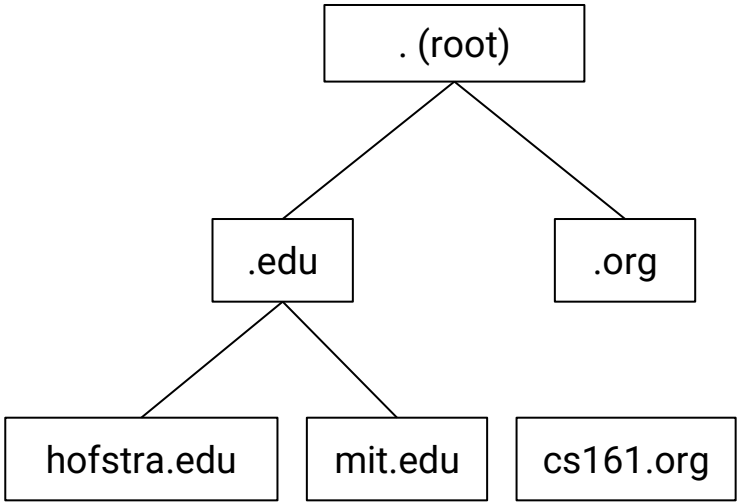
Stub resolver: The resolver on your computer.

- Only contacts the recursive resolver and receives the answer.

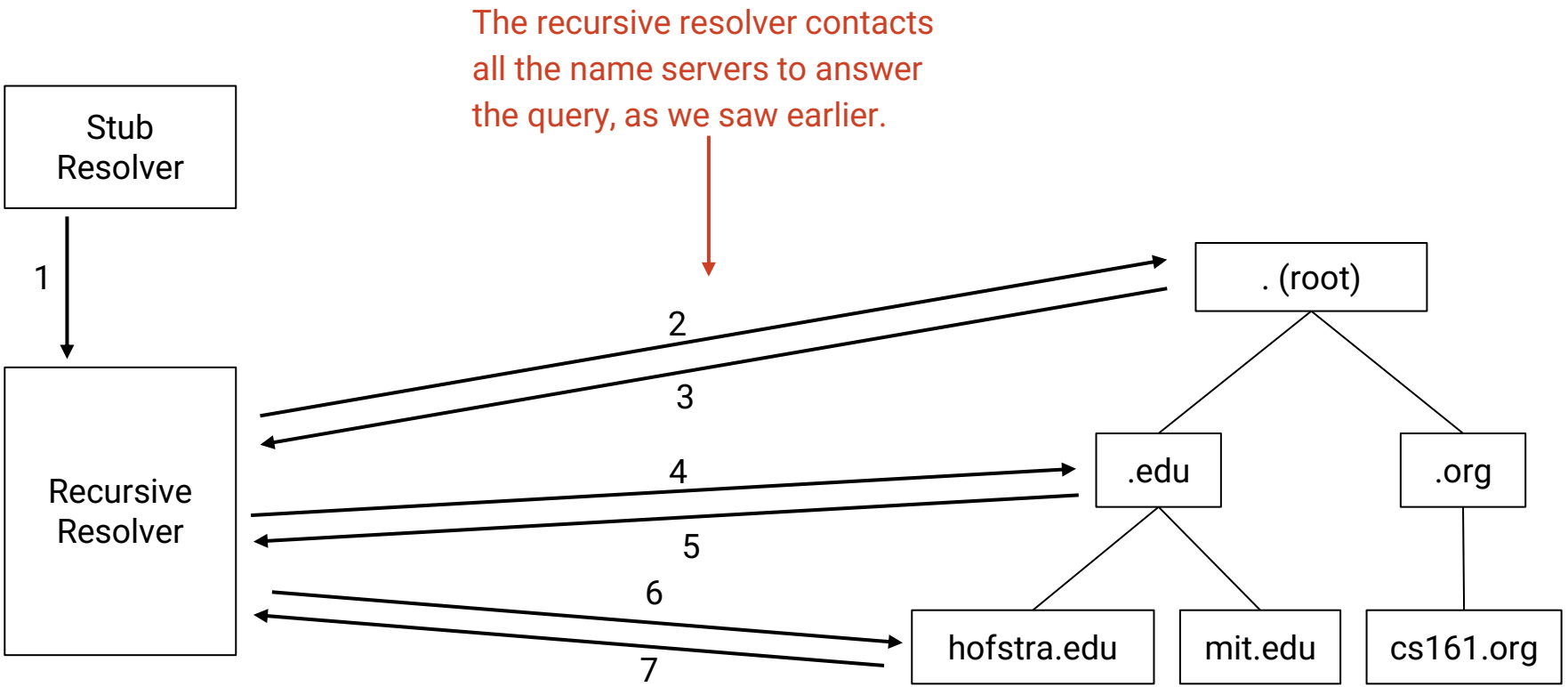
Recursive resolver: The resolver that makes the actual DNS queries.

Steps of a DNS Lookup (With Recursive Resolver)

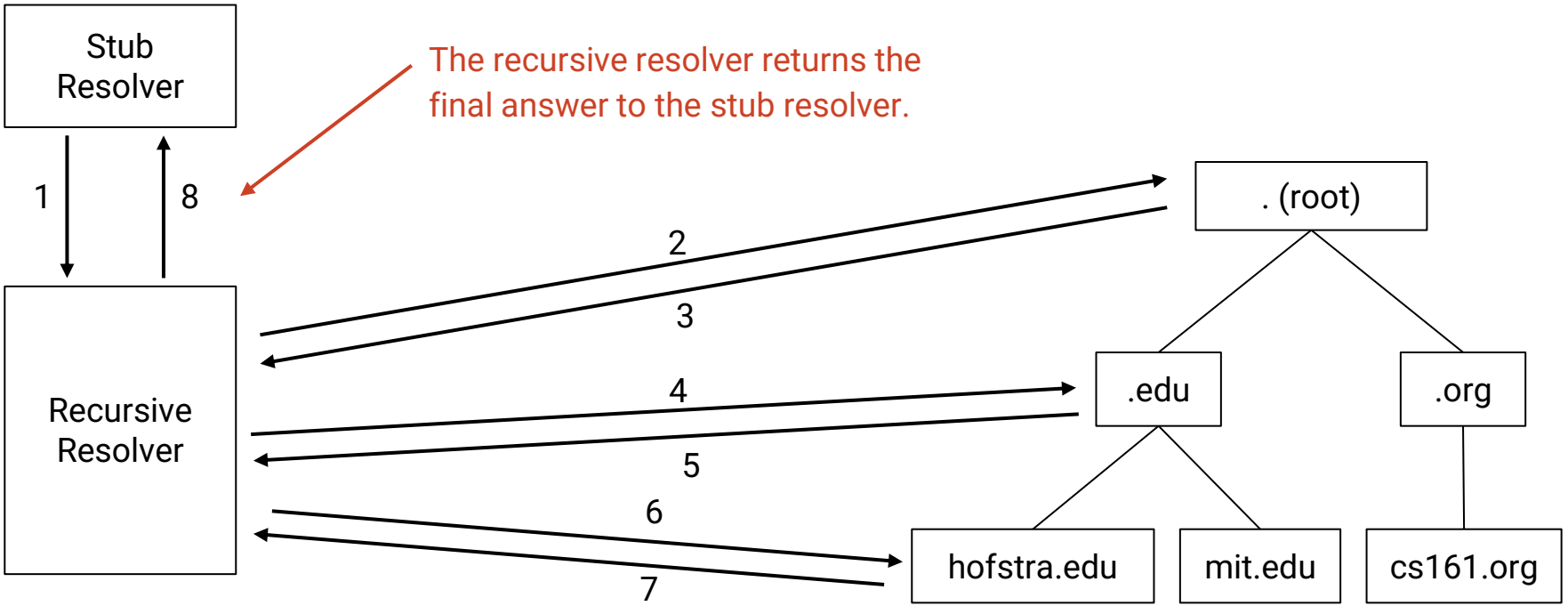




Steps of a DNS Lookup (With Recursive Resolver)



Steps of a DNS Lookup (With Recursive Resolver)



Recursive Resolvers

How do we know where the recursive resolver is?

- When you first join the network, somebody can tell you a resolver address.
- You can use a well-known resolver with a memorable IP address.
  - 1.1.1.1 (operated by Cloudflare).
  - 8.8.8.8 (operated by Google).

Why are recursive resolvers useful?

- The recursive resolver can build a larger cache from multiple users' requests.
  - If you query for www.google.com for the first time, the recursive resolver might have the answer cached from somebody else's query.
- Reduces load on name servers.
  - Recursive resolver asks the name server for www.google.com once, and can serve the answer to many users.

Recursive resolvers usually run by ISPs or application providers (Google, Cloudflare).

DNS Uses UDP

Recall UDP vs. TCP:

- UDP: Unreliable, but faster.
- TCP: Reliable, but slower.

DNS should be lightweight and fast, so it uses UDP.

- No 3-way handshake needed.
- No need for servers to maintain connection state.
- Most queries/responses fit in a single UDP packet (no bytestream needed).

DNS Uses UDP

What if a packet is dropped?

- Set a timer, and retry on timeout.
- Varies on different OSes, but can be fairly slow.
  - This is why resolvers need to be highly-available.

What if a message doesn't fit in a UDP packet?

- Generally never happens on typical requests and responses.
- Larger messages can be sent over TCP.

Recent advances in DNS use a secure transport protocol, with encryption.

- TCP and UDP are vulnerable to attackers.

Scaling DNS

Lecture 14, CS 168, Spring 2026

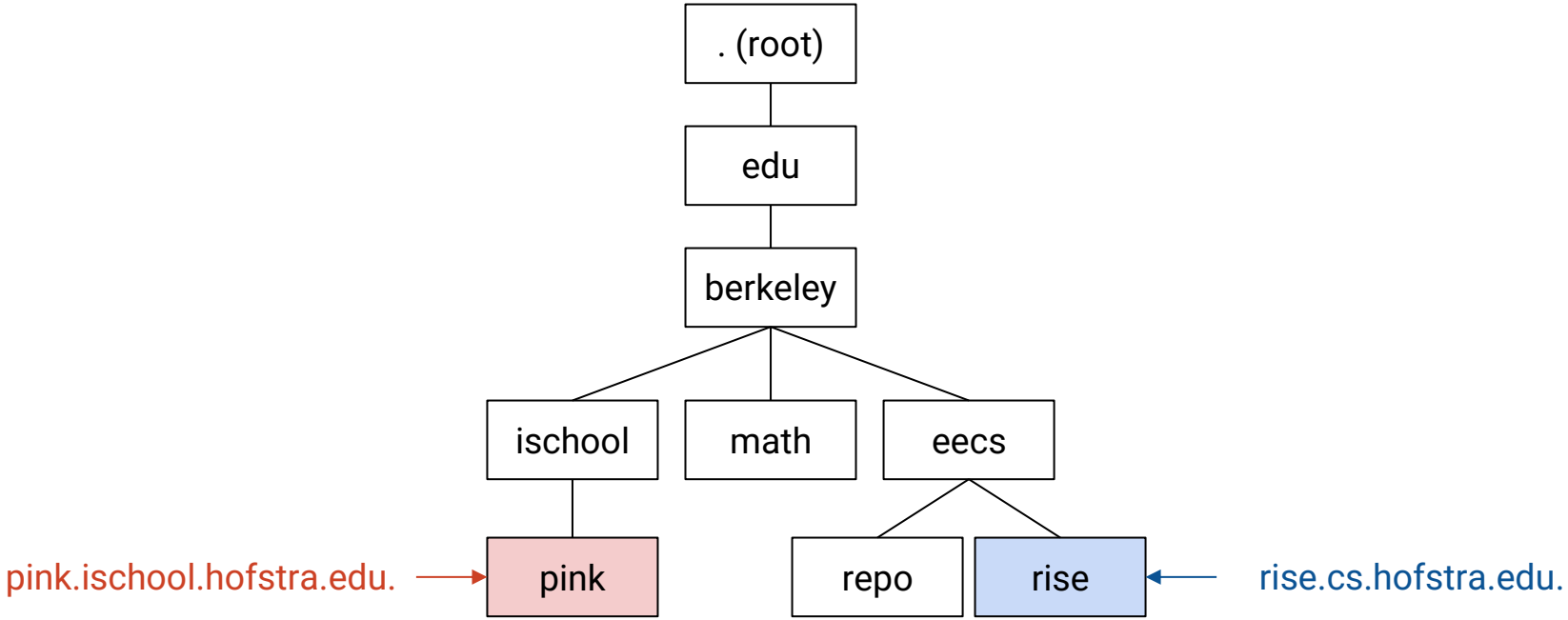
DNS

- What is DNS For?
- Design
- **Scaling**
- Other Uses

DNS Hierarchy

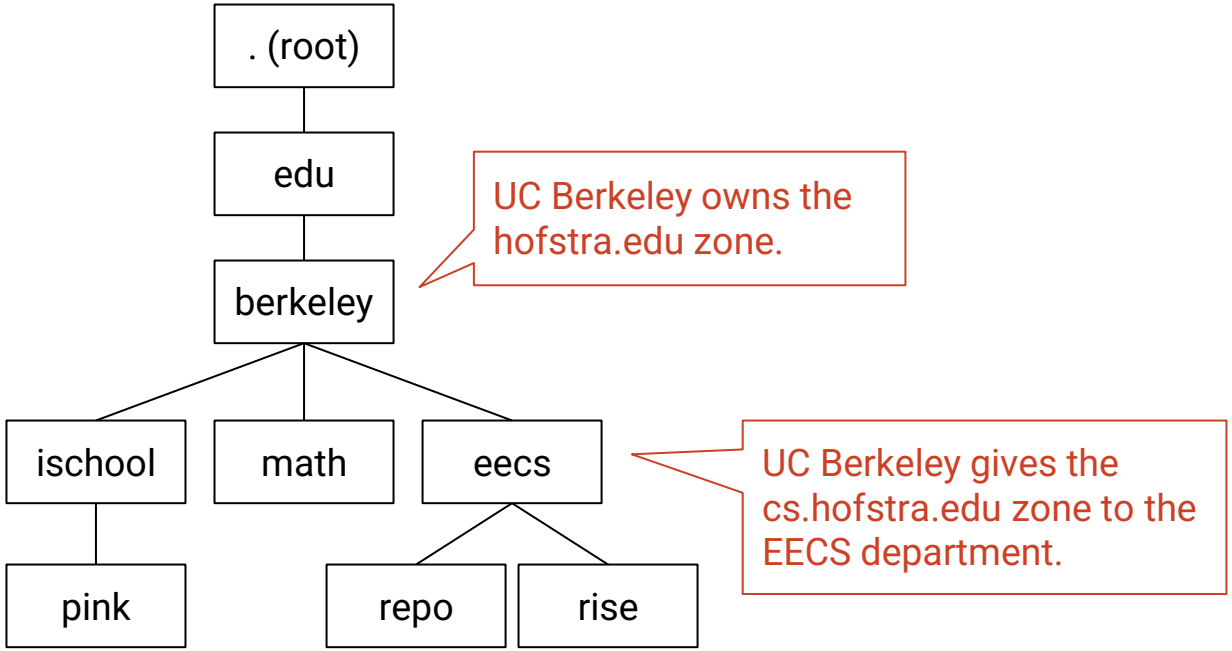
The DNS tree represents 3 forms of hierarchy.

Names are hierarchical. This is why our domain names are words separated by dots.



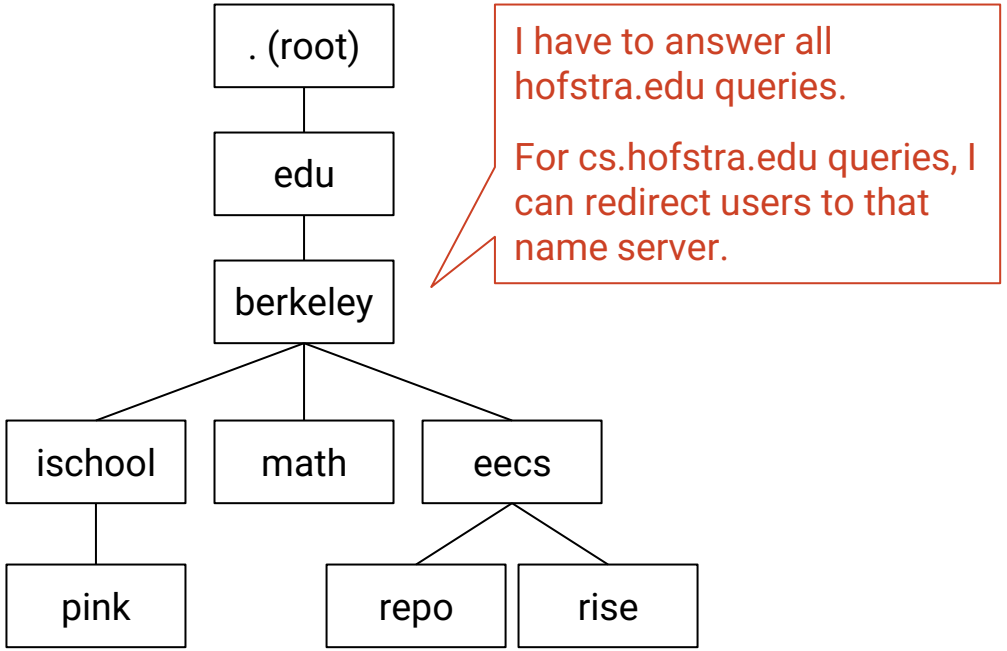
DNS Hierarchy

Authority is hierarchical. Each organization manages a zone, and can delegate parts of the zone to other organizations.



DNS Hierarchy

Infrastructure is hierarchical. Each name server only needs to know about a subset of domains.

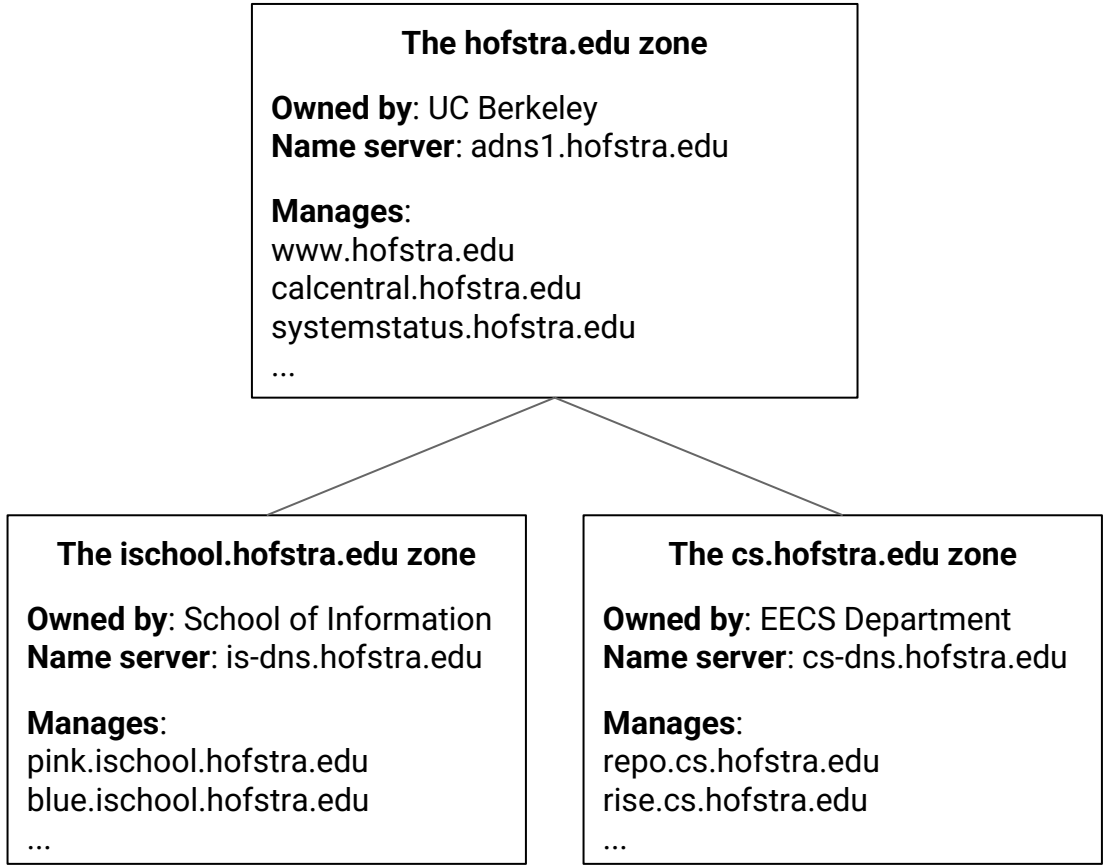


DNS Hierarchy

Each node in the tree represents a **zone** of domains.

An organization managing a zone can **delegate** part of its zone to somebody else.

Each zone has an **authoritative name server** that knows about the domains in that zone.



DNS Hierarchy

A **zone** corresponds to an administrative authority response for some part of the hierarchy.

- A zone is **authoritative** for how names within that part of the hierarchy are controlled.
- You can choose to **delegate** authority for part of the zone to somebody else.

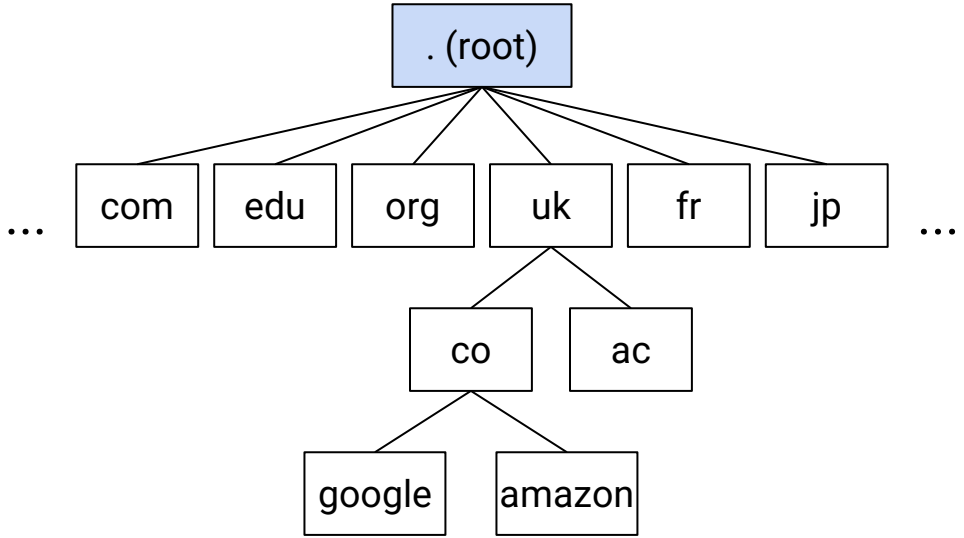
Zones help DNS scale.

- Different administrative authorities are responsible for different parts of the hierarchy.

Administrative Authorities

The DNS root is controlled by ICANN.

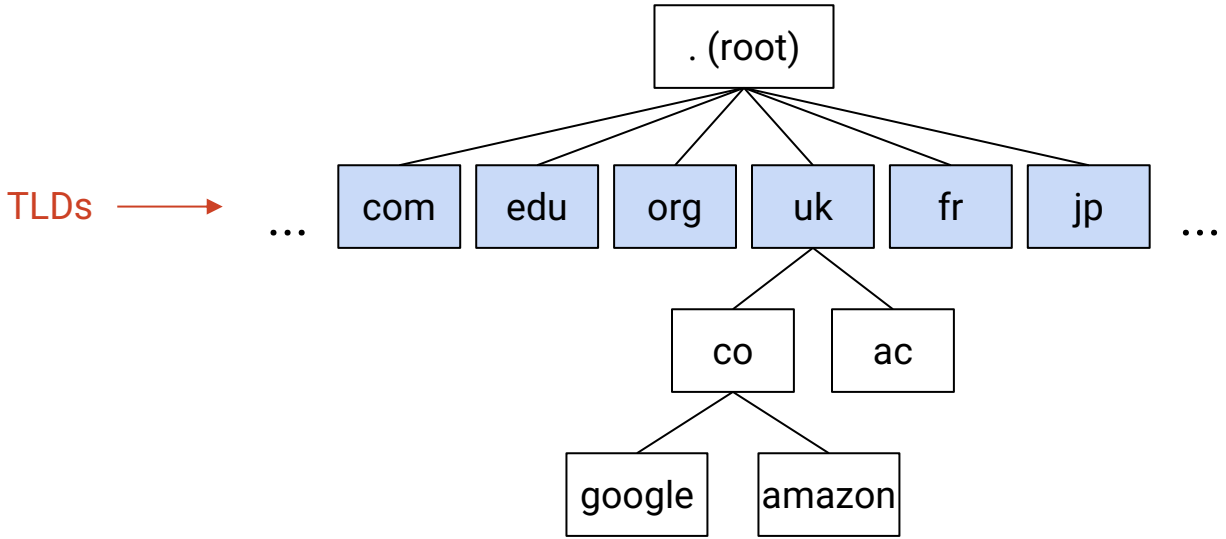
- Internet Corporation for Assigned Names and Numbers.



Administrative Authorities

Top-level domains (TLDs) are the zones directly below root.

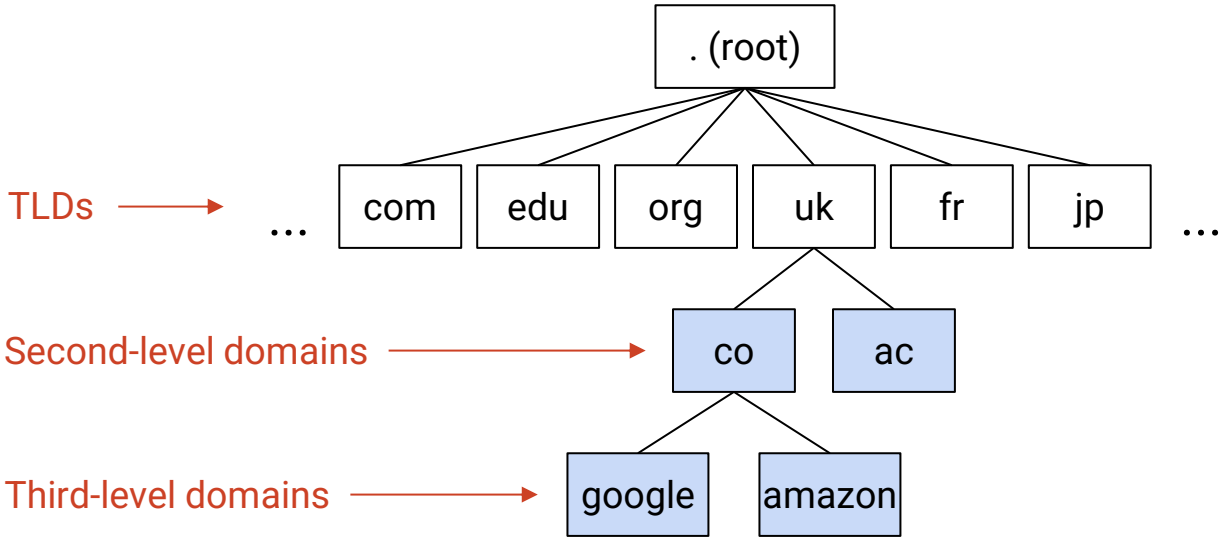
- Historically, relatively few, based on purpose (**org**anization, **edu**cation, **com**mercial, etc.), and country (UK, France, Japan, etc.).
- More recently, many more weird ones (.travel, .pizza, etc.). Over 1500 TLDs today!



Administrative Authorities

Second-level domains, third-level domains, etc. are below TLDs.

- Each TLD can decide its own structure.
- Example: The .uk zone delegated .co.uk (commercial) and .ac.uk (academic).



Administrative Authorities

Users can lease specific domains from **domain registries** (e.g. Verisign).

- If you want to host your website on example.com, you have to reserve the domain for a recurring (e.g. monthly) fee.
- Then, the authoritative .com name servers add a record mapping example.com to your IP address.

Organizations (e.g. Google, Amazon) can operate their own zones, with their own name servers.

- The organization has to tell its parent about its name server(s).
- Then, the parent will redirect queries to the organization.

Zone Availability

Recall: Each zone has several name servers for redundancy.

Each zone has two authoritative name servers.

- This ensures availability of that zone.

The name servers work in a primary/secondary model.

- The **primary** name server manages and updates the actual records.
- The **secondary** name server uses a read-only copy of the primary server's records.
- The primary server periodically transfers its records to the secondary servers.
    - Transfer can be large! This is where DNS might use TCP instead of UDP.

Root Zone Availability with Anycast

It would be really bad if the root servers were unavailable.

- Someone with an empty cache would be unable to make any lookups.

Root servers use a trick called anycast to ensure high availability.

- There are only 13 root-server domains (a.root-servers.net to m.root-servers.net).
- But, there are actually thousands of root servers.

Anycast: Use the same IP address for many servers on the Internet.

- Thousands of servers all claim to be k.root-servers.net with address 193.0.14.129 in routing protocols.
- You might hear advertisements from many different servers. You can pick any route (e.g. the shortest one), and you'll reach one of the root servers.

Anycast can also be used for other highly-available services (e.g. the 8.8.8.8 resolver).

Root Zone Availability with Anycast

All of these servers are advertising themselves as k.root-servers.net, with IP 193.0.14.129.

Lots of cooperation between network operators to keep root servers highly available.

Which instance am I actually using?

```
$ dig +short +norec @k.root-servers.net hostname.bind chaos txt
"ns1.us-mia.k.ripe.net"
```

"us-mia" is probably Miami, Florida.

Other Uses of DNS

Lecture 14, CS 168, Spring 2026

DNS

- What is DNS For?
- Design
- Scaling
- **Other Uses**

DNS for Load-Balancing

Multiple servers with different IP addresses can all host the same application.

- Useful for popular services (e.g. YouTube, Twitter).
- The name server can return multiple IP addresses for a single domain.
  - No precedence between these different A records.
  - Client can pick any of them. We often pick the first one.
  - Server shuffles the order in the response.
- Provides coarse-grained load-balancing across different servers.
- Provides simple resiliency. If one IP is down, pick another.

```
;; ANSWER SECTION:
microsoft.com.                  1999                    IN          A
            20.112.250.133
microsoft.com.                  1999                    IN          A
            20.231.239.246
microsoft.com.                  1999                    IN          A
            20.76.201.171
microsoft.com.                  1999                    IN          A
            20.70.246.20
microsoft.com.                  1999                    IN          A
            20.236.44.162
```

DNS for Load-Balancing

The name server can give different responses based on *where* the query came from.

- Server needs extra logic: "If the query is from X, reply with Y."

How do we determine which response to give?

- Address of the recursive resolver.
  - Most clients don't directly query name servers.
- Address of the client.
  - Requires extension to DNS to carry client address.
- Geographical location of the client.
  - Requires database to map IP addresses to physical locations, e.g. MaxMind.

From California:

```
$ dig google.com +short
142.251.46.238
```

From Oregon:

```
$ dig google.com +short
74.125.135.113
```

DNS for Load-Balancing

Challenges with geographical load-balancing:

- Ideally, we want to direct the client to the "nearest" server.
- We don't know the how "far" the user is from the server.
  - We don't know the geographic distance.
  - We don't know the network distance (e.g. number of hops).
- We don't know the performance between the user and the server.
  - What's the bandwidth and latency of the links between user and server?
- Some guessing is involved.
- Some proprietary logic is required at the name server.

DNS for Load-Balancing

Load-balancing experiment:

- From San Francisco, ask for the IPv6 address of www.youtube.com.
- Look up the name corresponding to that specific IP address.
- `sfo03s25-in-x0e.1e100.net`. "SFO" = San Francisco...pretty close!

Another load-balancing experiment:

- From Oregon, ask for the IPv6 address of www.youtube.com.
- San Francisco → IP address from San Francisco request = 20 ms RTT.
- San Francisco → IP address from Oregon request = 35 ms RTT.

Mapping logic gave us a way to map a client to a better-performing server.

Doing a "reverse" DNS lookup from IP address to hostname:

```
$ host 2607:f8b0:4005:80d::200e
e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.0.8.0.5.0.0.4.0.b.8.f.7.0.6.2.ip6.arpa domain
name pointer sfo03s25-in-x0e.1e100.net.
```

Summary: DNS

- DNS was created to allow for name to IP address resolution.
  - Helps humans easily access things on the Internet.
- DNS is a hierarchical system.
  - Zones and name servers have a hierarchy.
  - Allows for delegation to authoritative entities.
- DNS is implemented over UDP.
- DNS extends beyond address resolution into service resolution (e.g. email).
- DNS can be used for load-balancing.