Lecture 14 (Applications 1)

DNS

CS 168, Fall 2026 @ UC Berkeley

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

What is DNS For?

Lecture 14, CS 168, Fall 2026

DNS

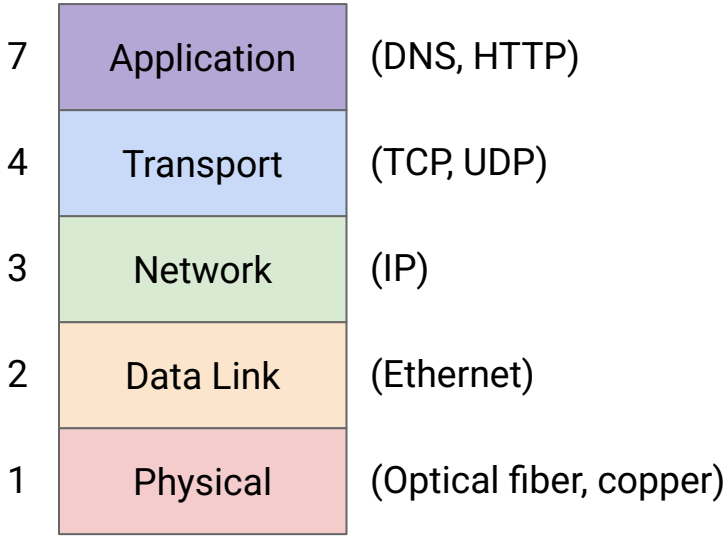
- **What is DNS For?**
- Design
- Implementation
- Scaling
- Other Uses

Application Layer

Lots of applications are built on the Internet infrastructure we've seen.

We'll focus on some common ones:

- DNS (today).
- HTTP (next time).



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.

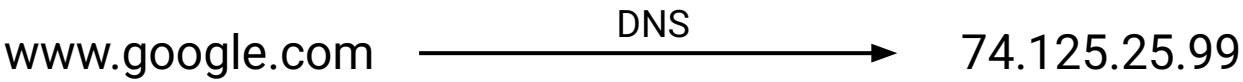
- Example: www.google.com.
- Not useful for machines: Contains no relevant routing information.
- Useful for humans: Meaningful words and phrases, easy to remember.

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.



Brief History of DNS

On the Internet and ARPANET (its predecessor), three main applications.

Remote terminal (`telnet 34.8.12.0`):

- Connect to someone else's machine remotely. SSH is a modern successor.

File transfer (`ftp 74.1.254.1`):

- Copy files across the network.

Email (`mail alice@88.1.24.0`):

- Send a message to another user.

Remembering the remote host address is difficult for humans.

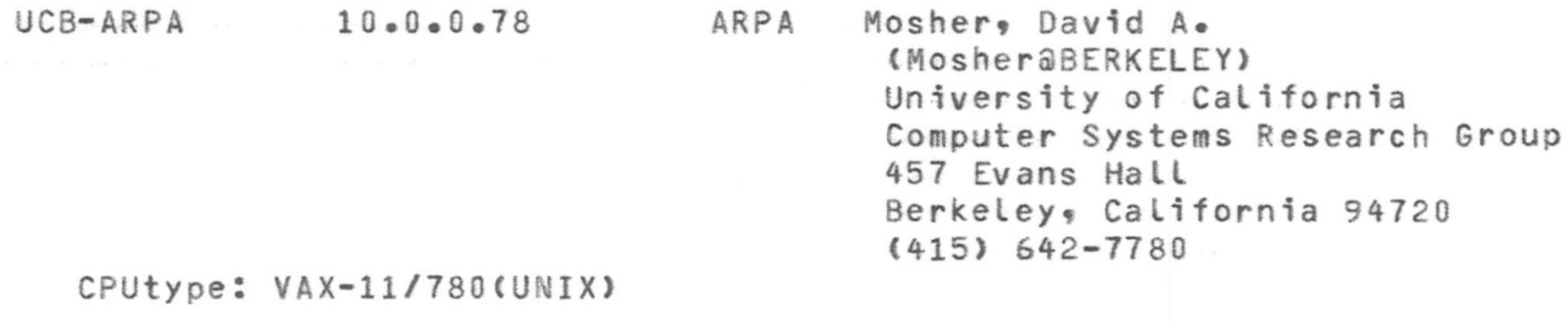
Brief History of DNS

Original design: Computers had a `hosts.txt` file that mapped names to IP addresses.

- Example: `ucb-arpa` maps to 10.0.0.78.
- Using IP address: `mail mosher@10.0.0.78`.
- Using hostname: `mail mosher@ucb-arpa`.

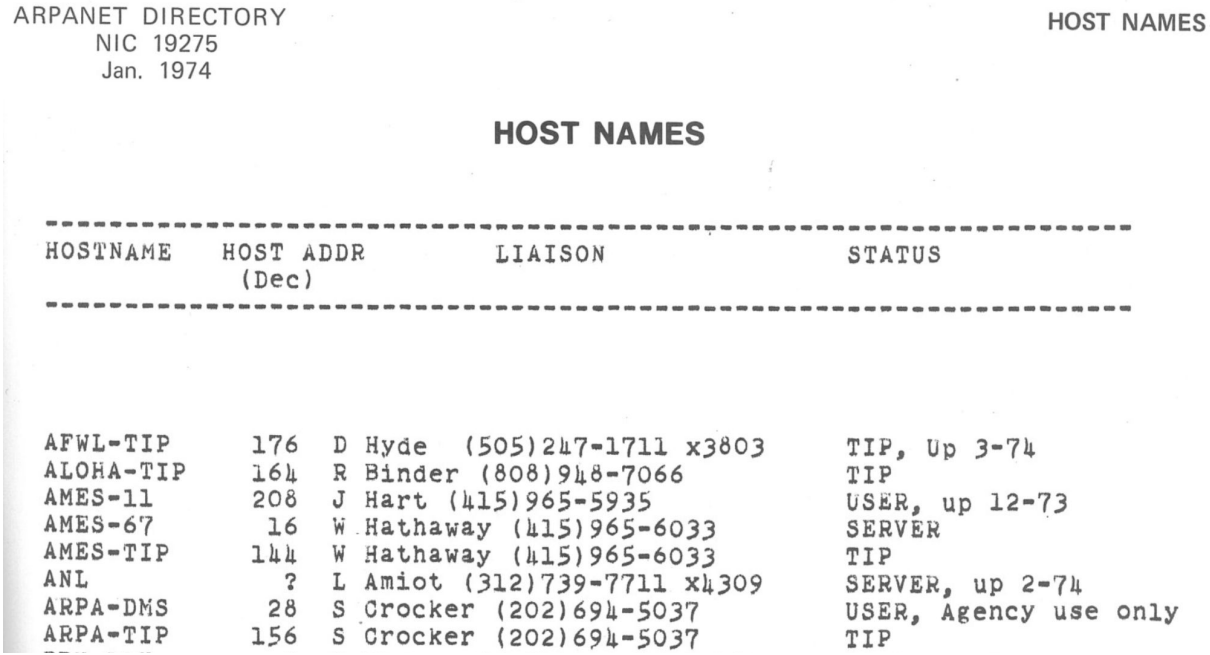
Originally maintained by Elizabeth Jocelyn "Jake" Feinler.

- The file was photocopied and exchanged between users.



Brief History of DNS

Originally, `hosts.txt` was human-readable.



ARPANET DIRECTORY
NIC 19275
Jan. 1974

HOST NAMES

HOST NAMES

HOSTNAME	HOST ADDR (Dec)	LIAISON	STATUS
AFWL-TIP	176	D Hyde (505)247-1711 x3803	TIP, Up 3-74
ALOHA-TIP	164	R Binder (808)948-7066	TIP
AMES-11	208	J Hart (415)965-5935	USER, up 12-73
AMES-67	16	W Hathaway (415)965-6033	SERVER
AMES-TIP	144	W Hathaway (415)965-6033	TIP
ANL	?	L Amiot (312)739-7711 x4309	SERVER, up 2-74
ARPA-DMS	28	S Crocker (202)694-5037	USER, Agency use only
ARPA-TIP	156	S Crocker (202)694-5037	TIP

"I remembered that back then we simply xeroxed the hosts.txt file and put it into the Arpanet Directory, so I copied that." –Elizabeth Feiner

Brief History of DNS

Problems with `hosts.txt`:

- Everyone needs to use the same file.
  - If you type `mail mosher@ucb-arpa` on different computers, you should be emailing the same person.
- Users might have an outdated file.

> It seems about time to put an end to the absurd situation where each site on the network must maintain a different, generally out-of-date, host list.
> –RFC 606 (1973)

Brief History of DNS

First step: Make the file machine-readable.

- Can use FTP to transfer the `hosts.txt` file between computers.

Scaling is still an issue as the Internet grew.

- Could end up with a partial file if the download breaks halfway through.

```
NET :  44.0.0.0  : AMPRNET  :
NET :  45.0.0.0  : C3-PR    :
NET :  46.0.0.0  : UCB-ETHER    :
NET :  47.0.0.0  : SAC-PR-TEMP :

HOST : 46.0.0.4  : UCBARPA  : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,UDP :
HOST : 46.0.0.5  : UCBCAD   : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,UDP :
HOST : 46.0.0.6  : UCBERNIE : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,UDP :
HOST : 46.0.0.7  : UCBMONET : VAX-11/750 : UNIX : TCP/TELNET,TCP/FTP,UDP :
HOST : 46.0.0.9  : UCBESVAX : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,UDP :
HOST : 46.0.0.10 : UCBVAX   : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,UDP :
```

Check out RFC608 and this 1983 hosts file.

Brief History of DNS

DNS was first proposed in RFC882 (1983).

We use this system (with some modifications – but not many) today.

There's a huge amount of DNS history here at UC Berkeley!

- First DNS server written for Unix was BIND.
- berkeley.edu is the oldest .edu domain!

The Berkeley Internet Name Domain Server	Link
Douglas B. Terry, Mark Painter, David W. Riggle, Songnian Zhou (UC Berkeley)	*1984*

Goals of DNS

DNS must be **scalable**.

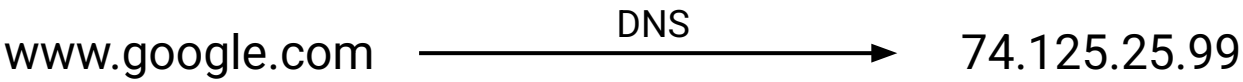
- Many hosts/names. Many lookups. Many updates.

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, Fall 2026

DNS

- What is DNS For?
- **Design**
- Implementation
- 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?

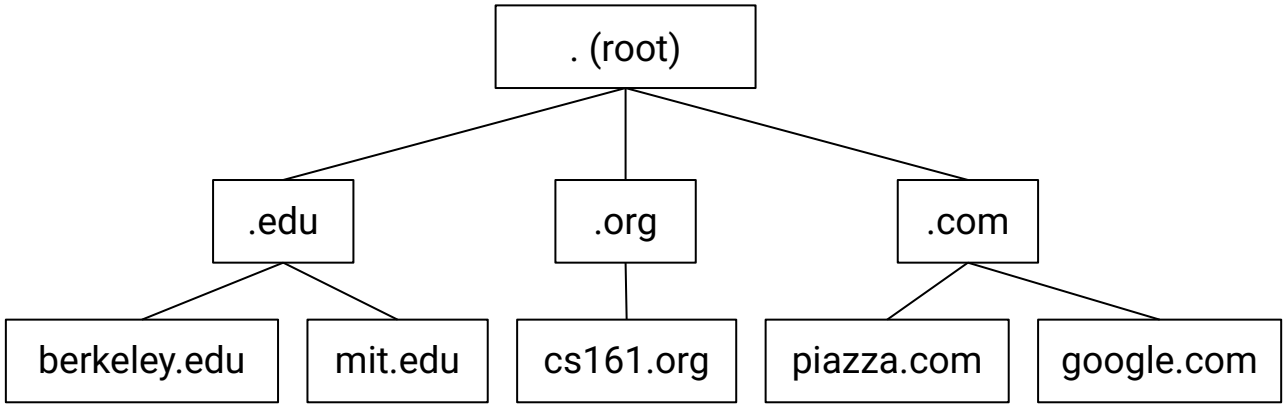
DNS Name Server Hierarchy

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.

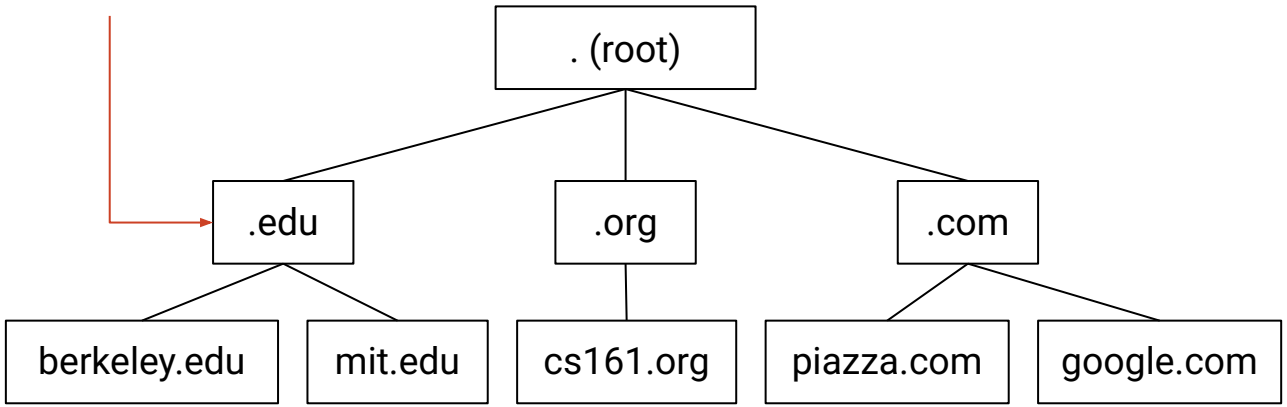
- Intuition: Name servers will direct you down the tree until you receive the answer to your query.



DNS Name Server Hierarchy

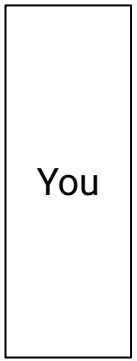
Each box is a name server. The label represents which queries the name server is responsible for answering.

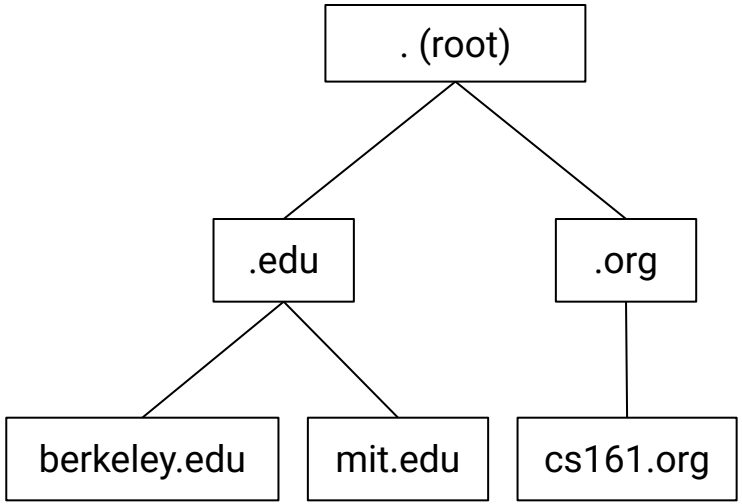
This name server is responsible for .edu queries like eecs.berkeley.edu, but not a query like mail.google.com.



Steps of a DNS Lookup

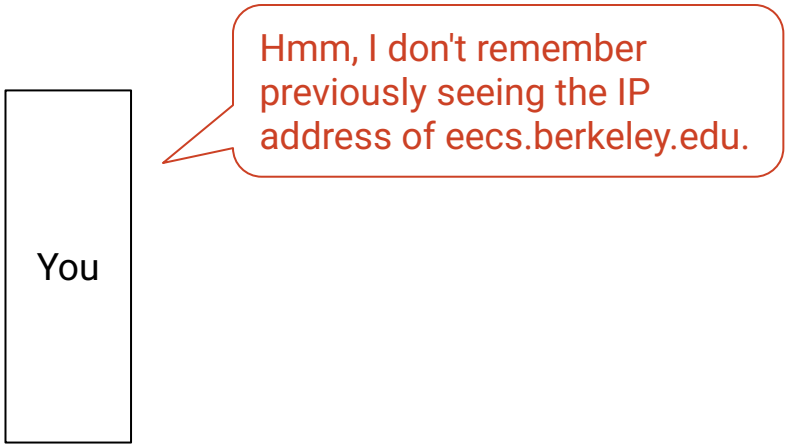
Let's walk through a DNS query for the IP address of eecs.berkeley.edu.

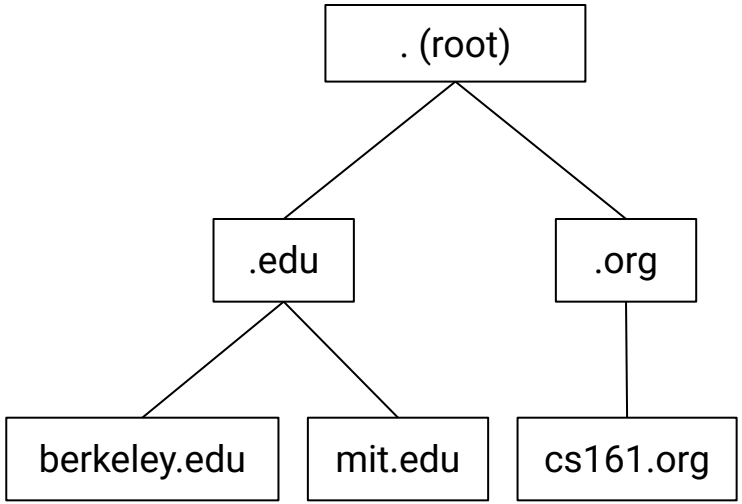




Steps of a DNS Lookup

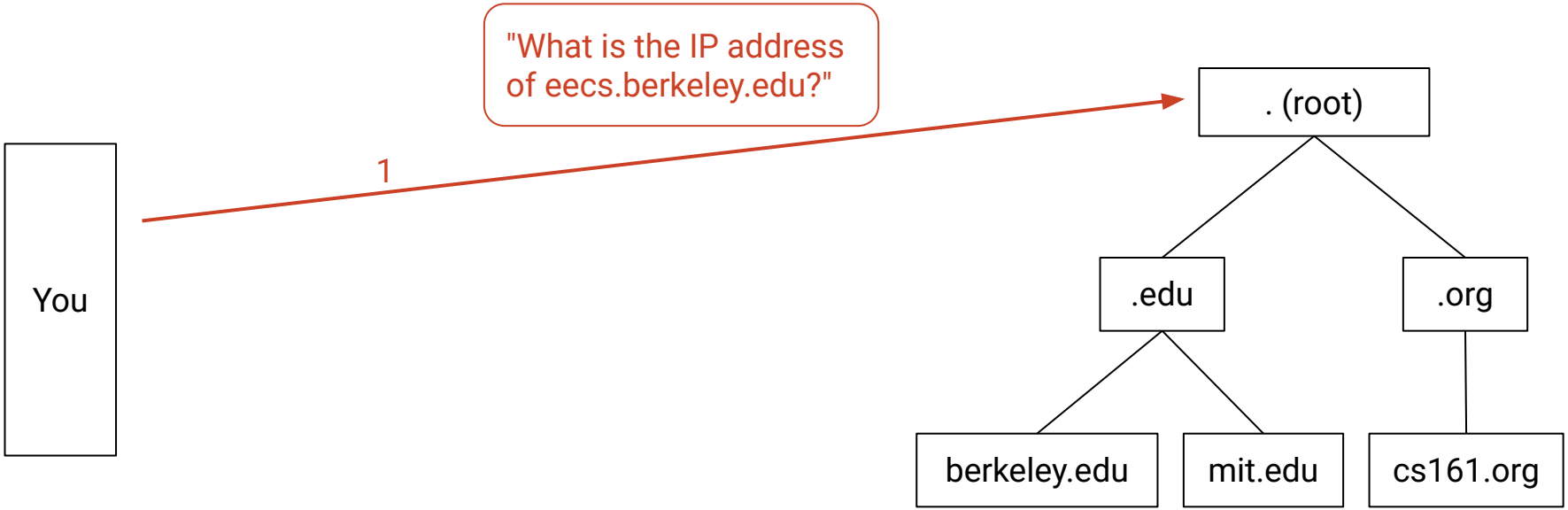
First, you check your cache to see if you already know the answer to your query.





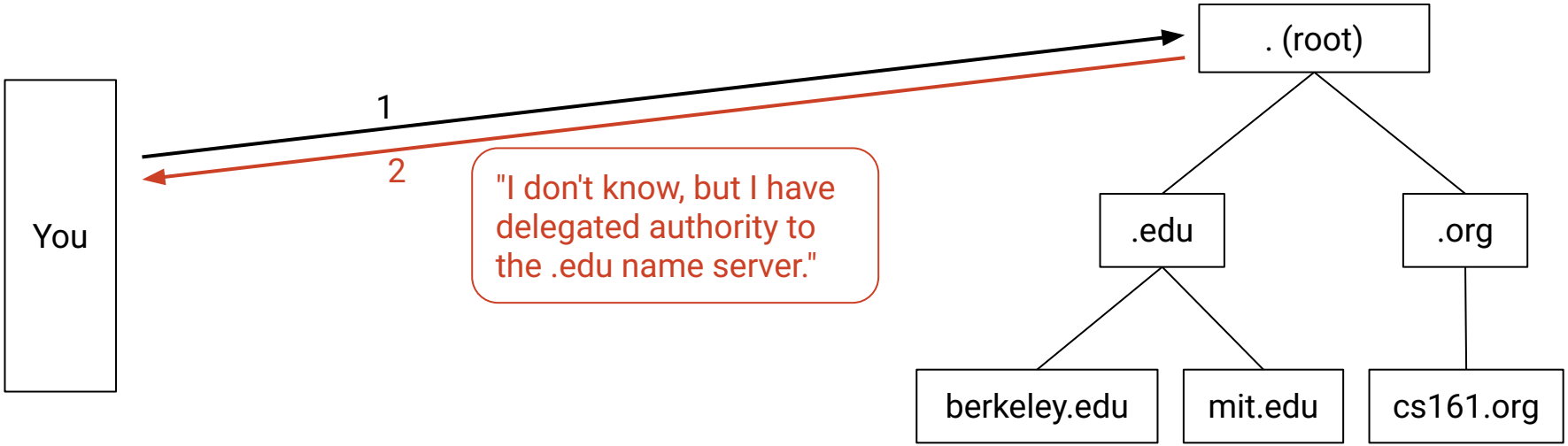
Steps of a DNS Lookup

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



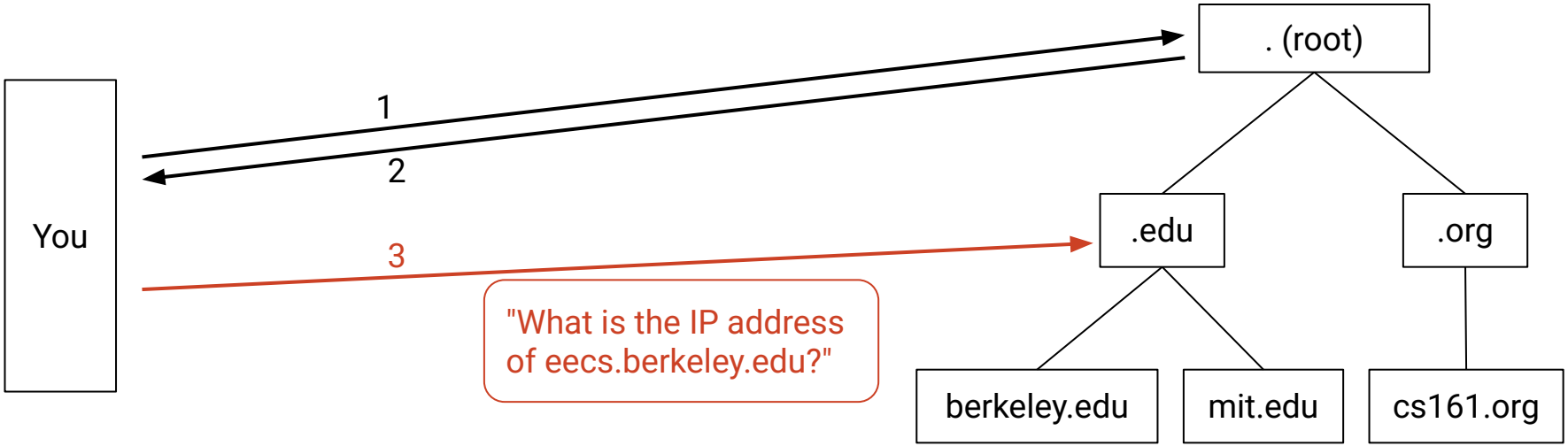
Steps of a DNS Lookup

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



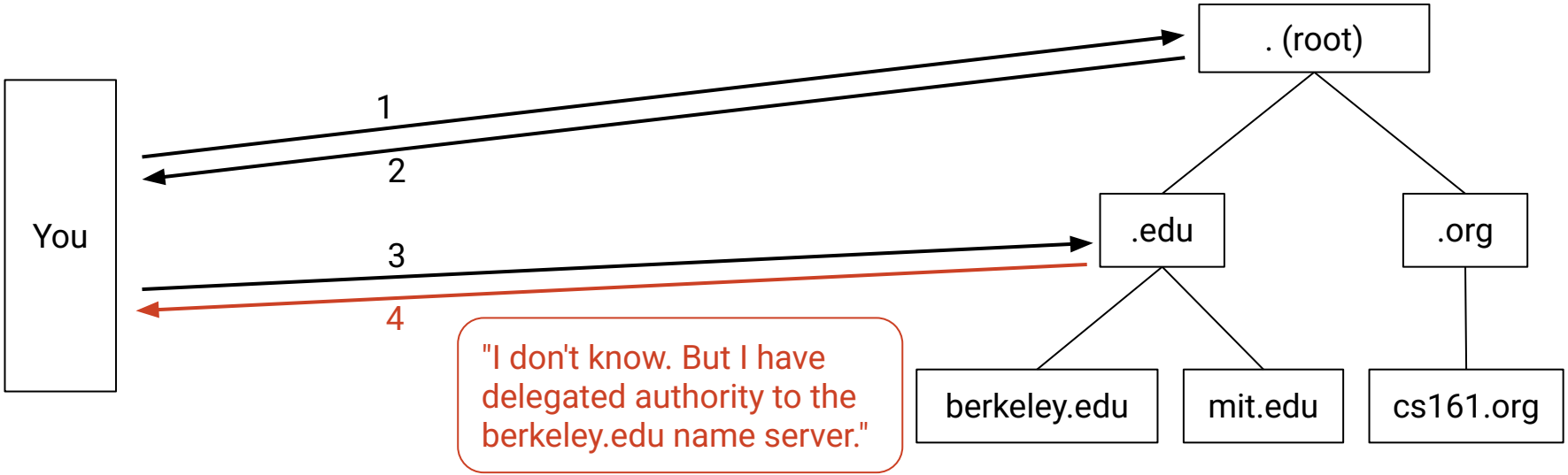
Steps of a DNS Lookup

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



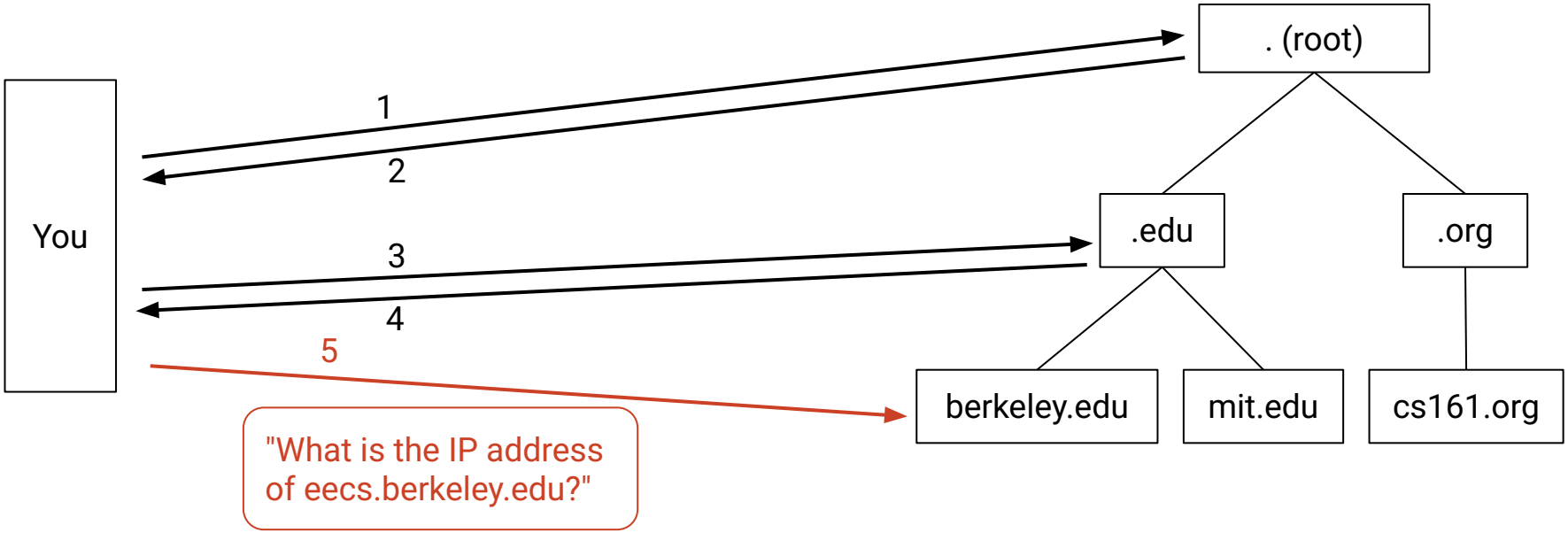
Steps of a DNS Lookup

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



Steps of a DNS Lookup

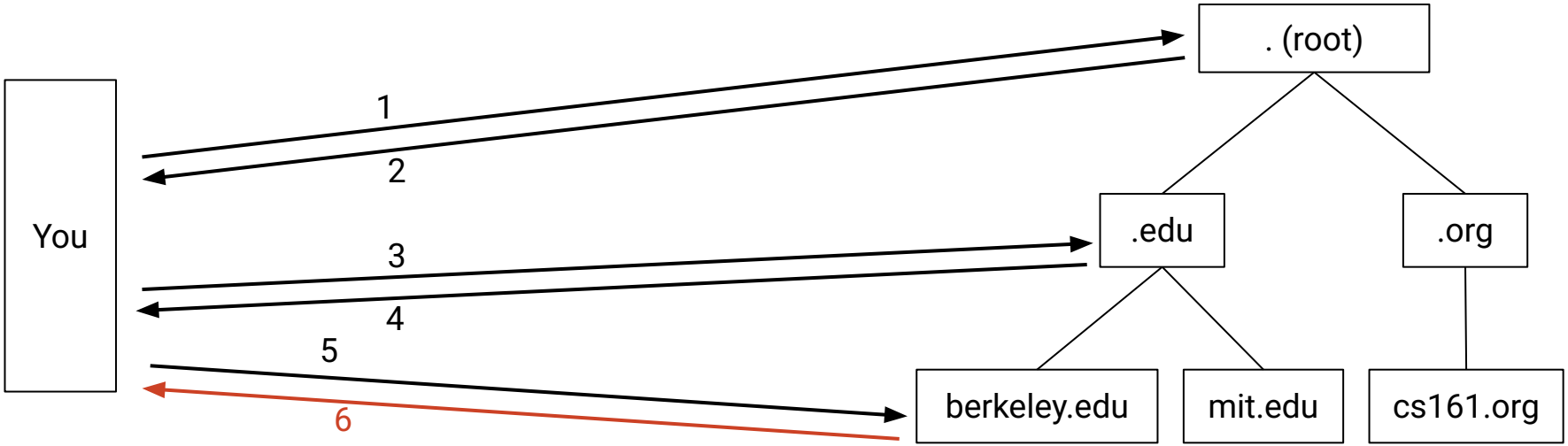
Next, you ask the same question, but now to the berkeley.edu name server.



Steps of a DNS Lookup

The berkeley.edu name server responds with the answer.

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



Stub Resolvers and Recursive Resolvers

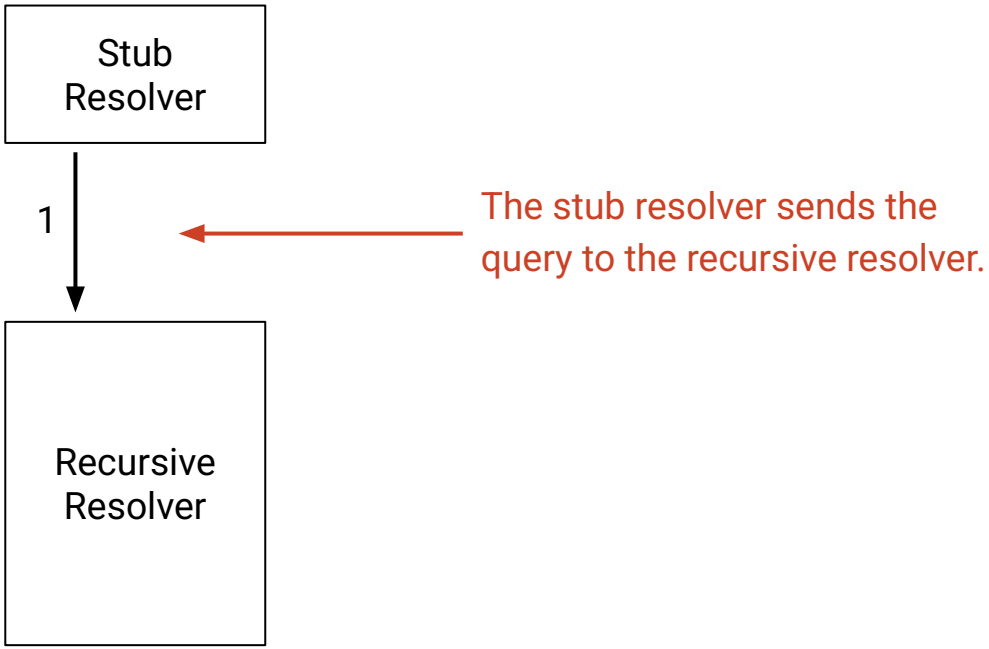
In practice, your computer usually tells another resolver to perform the query for you.

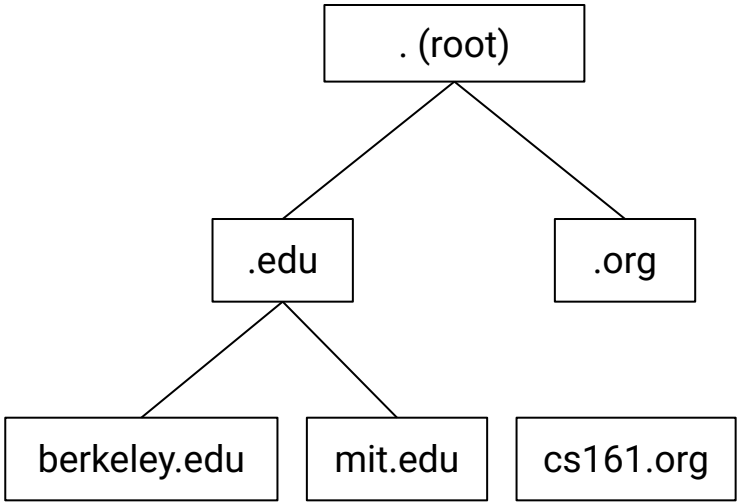
Stub resolver: The resolver on your computer.

- Only contacts the recursive resolver and receives the answer.

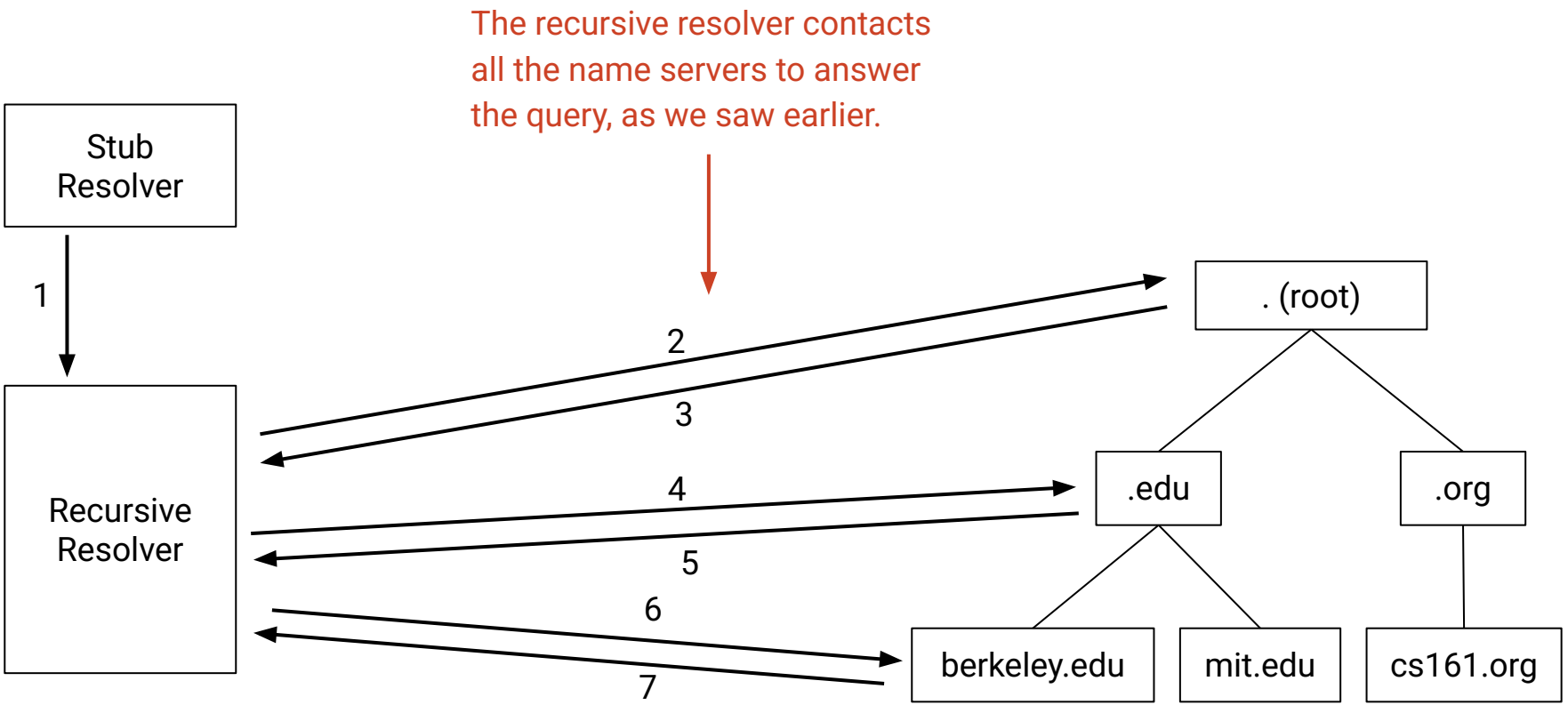
Recursive resolver: The resolver that makes the actual DNS queries.

Steps of a DNS Lookup (With Recursive Resolver)

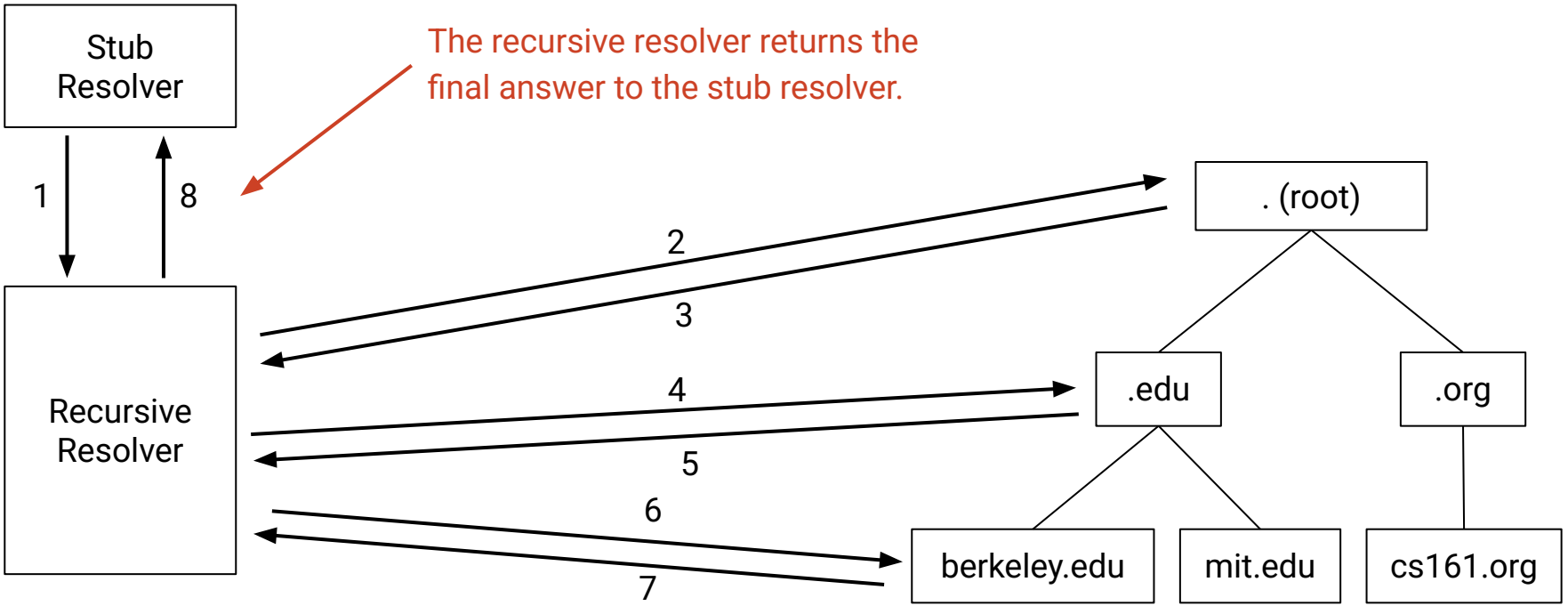




Steps of a DNS Lookup (With Recursive Resolver)



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).

Recursive Resolver Failures

What happens if a resolver breaks?

- Anyone relying on 1.1.1.1 to resolve DNS queries cannot access webpages.

Resolvers can be a single point of failure!

- Recently 1.1.1.1 stopped responding.
- *Maybe* a BPG issue?

BGP Origin Hijack: 107469

Details of the BGP hijack event

Hijacker ASN: AS4755 - Tata Communications Limited (IN)
Victim ASNs: AS13335 - Cloudflare, Inc. (US)
Prefixes: 1.1.1.0/24
Messages: 4
Earliest message time: 07/14/2025, 21:54
Latest message time: 07/14/2025, 22:01
Peers observed: 2%
Confidence: High
Tags: RPKI Invalid | IRR Invalid | IRR Old Origin Valid | RPKI Old Origin Valid | Provider

Issues with 1.1.1.1 public resolver

Incident Report for Cloudflare

Resolved	This incident has been resolved. Posted 6 hours ago. Jul 14, 2025 - 23:20 UTC
Monitoring	A fix has been implemented and we are monitoring the results. Posted 7 hours ago. Jul 14, 2025 - 22:59 UTC
Update	Gateway DNS and Authoritative DNS are not impacted. Posted 7 hours ago. Jul 14, 2025 - 22:19 UTC
Identified	The issue has been identified and a fix is being implemented. Posted 7 hours ago. Jul 14, 2025 - 22:17 UTC
Investigating	Cloudflare is aware of, and investigating, an issue which potentially impacts multiple users that use 1.1.1.1 public resolver and Gateway plain text DNS. Further detail will be provided as more information becomes available. Posted 8 hours ago. Jul 14, 2025 - 22:13 UTC

DNS Implementation

Lecture 14, CS 168, Fall 2026

DNS

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

APIs for Developers

What does DNS look like from a developer perspective?

Relatively simple, common APIs that are available in almost all languages.

Example API in C:

- `result = gethostbyname("foo.com")`
  - Limited to IPv4.
  - Deprecated, but still used.
- `error = getaddrinfo("example.com", NULL, NULL, &result)`
  - Replaces the deprecated method.
  - Supports more than IPv4.

Functions usually just make requests to the OS's configured resolving DNS server.

- All the complexities of DNS hidden from the end developer.

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.

DNS Packet Format: UDP Header

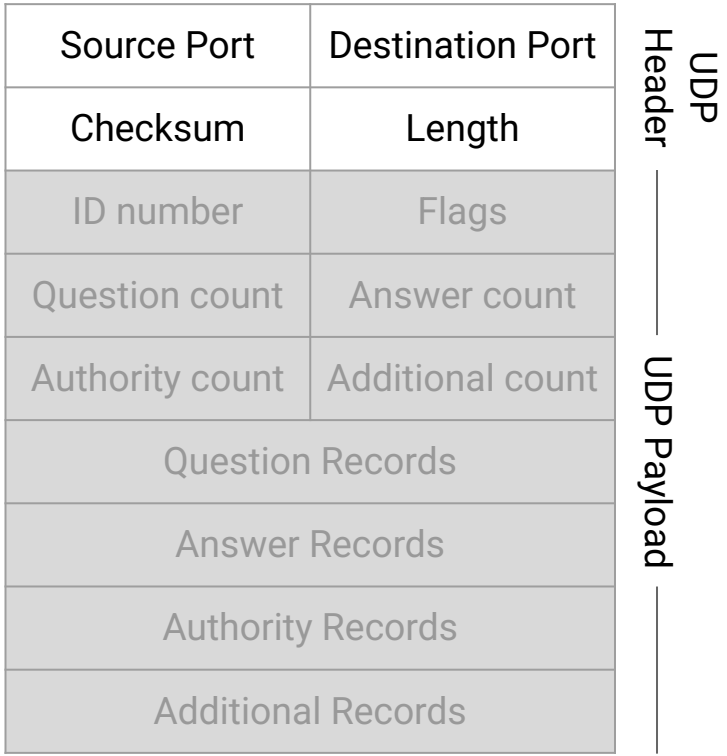
Source port (16 bits): Chosen by the client.

Destination port (16 bits): Usually 53.

- Servers typically listen for queries on a well-known UDP port (53).

Checksum on UDP payload.

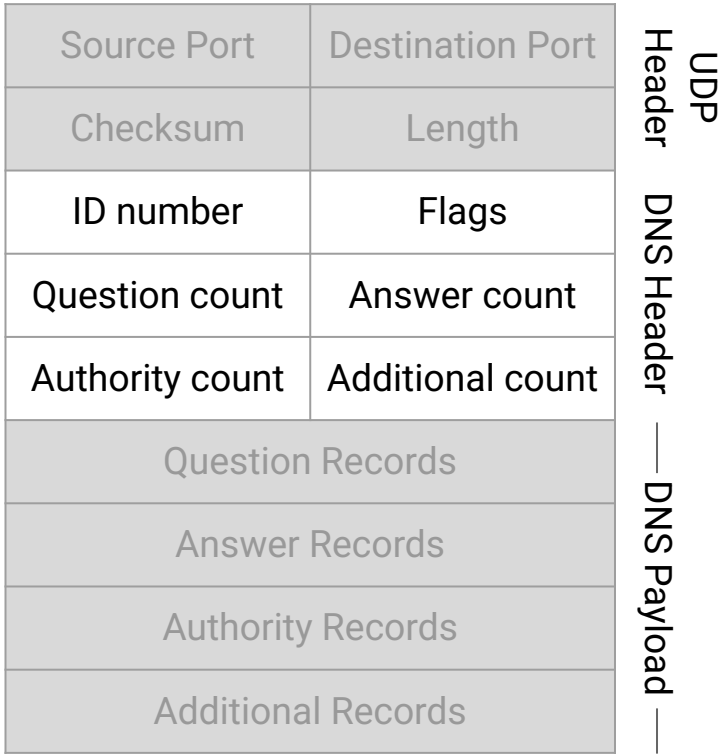
Length of UDP payload.



DNS Packet Format: DNS Header

ID number (16 bits): Used to associate queries with responses.

- Client picks an ID number in the query.
- Name server uses the same ID number in the response.
- Should be randomly chosen for security purposes (not discussed further).

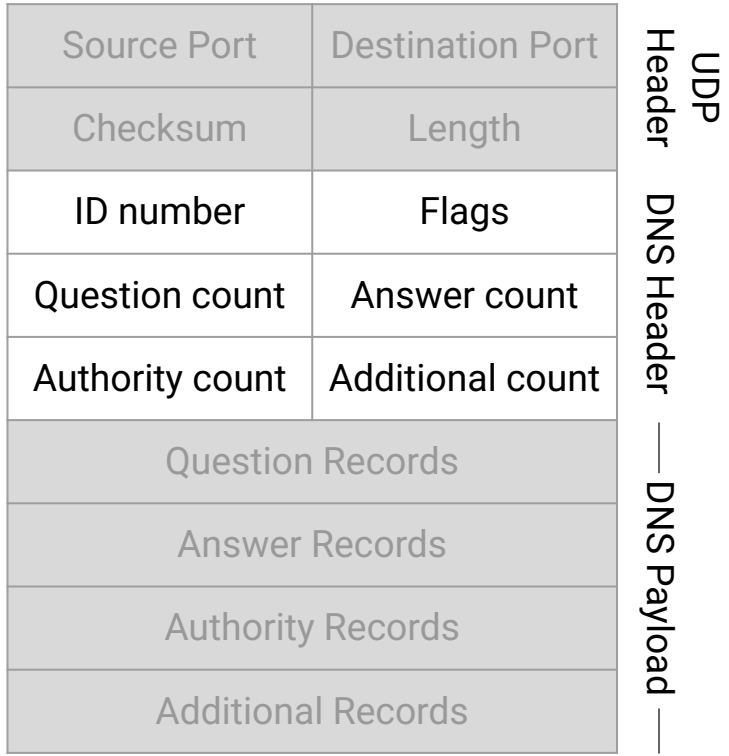


DNS Packet Format: DNS Header

Flags:

- QR bit: 0 in queries, 1 in responses.
- RD bit: 1 if we want the recursive resolver to do the lookup for us, and 0 otherwise.
- In theory, there are different query types.
  - IQUERY: Obsolete.
  - STATUS: Not really defined.
  - QUERY: Used for basically everything.

Counts: The number of records of each type in the DNS payload.



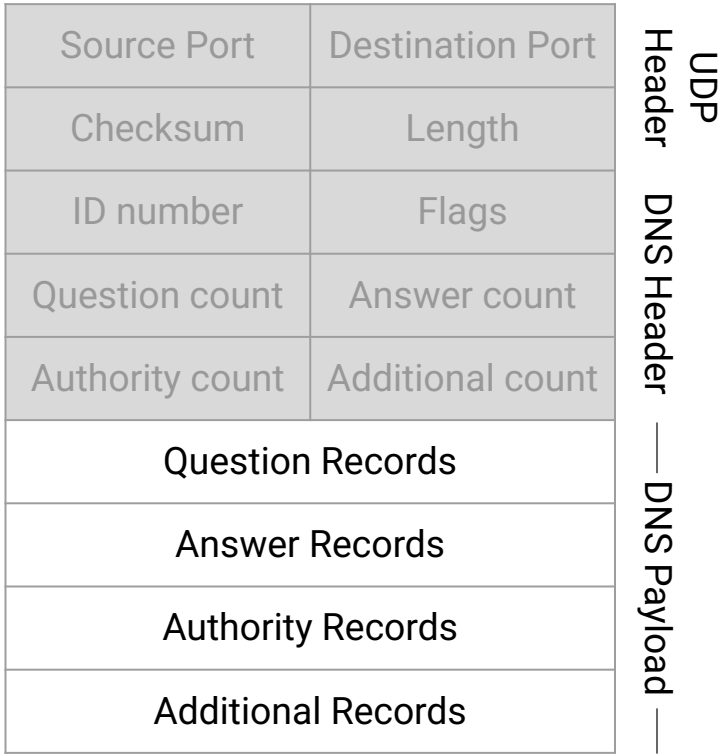
DNS Packet Format: DNS Payload

The DNS payload contains a variable number of **resource records (RRs)**.

Each RR is a name-value pair.

RRs are sorted into four sections:

- Question section.
- Answer section.
- Authority section.
- Additional section.



DNS Record Format

Each record is a name-value pair with a type.

- **A (answer) type records**: Maps a domain name to an IPv4 address.
- **AAAA type records**: Maps a domain name to an IPv6 address.
- **NS (name server) type records**: Designates another DNS server to handle a domain.
- Other types exist, but these are the ones you need to know for now.

Each record also contains some metadata.

- **Time to live (TTL)**: How long the record can be cached.
- **Class**: We'll ignore this. Basically always set to IN (Internet).

DNS Record Sections

Question section: What is being asked.

- Included in both requests and responses.
- Usually an A type record with the domain being looked up.

Answer section: A direct response to the question.

- Empty in requests.
- Used if the name server responds with the answer.
- Usually an A type record with the IP address of the domain being looked up.

DNS Record Sections

Authority section: A delegation of authority for the question.

- Empty in requests.
- Used to direct the resolver to the next name server.
- Usually an NS type record with the zone and domain of the child name server.

Additional section: Additional information to help with the response, sometimes called glue records.

- Empty in requests.
- Provides helpful, non-authoritative records for domains.
- Usually an A type record with the domain and IP address of the child name server (since the NS record provides the child name server as a domain).

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4
```

You can try this at home! Use the **dig** utility in your terminal, and remember to set the **+norecurse** flag so you can traverse the name server hierarchy yourself.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4
```

We are performing a DNS lookup for the IP address of **eecs.berkeley.edu**.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4
```

DNS queries always start with a request to the root name server. The IP address of the root name server is usually hard-coded into recursive resolvers.

Operators can also configure a *root hints file* with the root addresses, like this: https://www.internic.net/domain/named.root

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A     192.5.6.30
b.edu-servers.net.  172800   IN   A     192.33.14.30
c.edu-servers.net.  172800   IN   A     192.26.92.30
...
```

Here's the DNS response from the root name server.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A     192.5.6.30
b.edu-servers.net.  172800   IN   A     192.33.14.30
c.edu-servers.net.  172800   IN   A     192.26.92.30
...
```

Here's the DNS header.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A     192.5.6.30
b.edu-servers.net.  172800   IN   A     192.33.14.30
c.edu-servers.net.  172800   IN   A     192.26.92.30
...
```

Here's the 16-bit ID number in the DNS header.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN    A

;; AUTHORITY SECTION:
edu.                  172800    IN    NS    a.edu-servers.net.
edu.                  172800    IN    NS    b.edu-servers.net.
edu.                  172800    IN    NS    c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.    172800    IN    A     192.5.6.30
b.edu-servers.net.    172800    IN    A     192.33.14.30
c.edu-servers.net.    172800    IN    A     192.26.92.30
...
```

Here are the record counts in the DNS header.

Here are the flags in the DNS header.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A     192.5.6.30
b.edu-servers.net.  172800   IN   A     192.33.14.30
c.edu-servers.net.  172800   IN   A     192.26.92.30
...
```

Here's the DNS payload. It's a collection of resource records (one per line), sorted into four sections.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN    A

;; AUTHORITY SECTION:
edu.                    172800    IN    NS    a.edu-servers.net.
edu.                    172800    IN    NS    b.edu-servers.net.
edu.                    172800    IN    NS    c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.      172800    IN    A      192.5.6.30
b.edu-servers.net.      172800    IN    A      192.33.14.30
c.edu-servers.net.      172800    IN    A      192.26.92.30
...
```

Here's the question section. The name is **eecs.berkeley.edu**, the type is A, and the value is blank. It shows that we are looking for the IP address of **eecs.berkeley.edu**.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN    A

;; AUTHORITY SECTION:
edu.                    172800    IN    NS    a.edu-servers.net.
edu.                    172800    IN    NS    b.edu-servers.net.
edu.                    172800    IN    NS    c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.      172800    IN    A      192.5.6.30
b.edu-servers.net.      172800    IN    A      192.33.14.30
c.edu-servers.net.      172800    IN    A      192.26.92.30
...
```

The answer section is blank, because the root name server did not return the answer we're looking for.

We can confirm this by checking the header, which says there are 0 records in the answer section.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN    A

;; AUTHORITY SECTION:
edu.                    172800    IN    NS    a.edu-servers.net.
edu.                    172800    IN    NS    b.edu-servers.net.
edu.                    172800    IN    NS    c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.      172800    IN    A      192.5.6.30
b.edu-servers.net.      172800    IN    A      192.33.14.30
c.edu-servers.net.      172800    IN    A      192.26.92.30
...
```

The authority and additional sections tell the resolver where to look next.

Note that there are multiple **.edu** name servers for redundancy.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A     192.5.6.30
b.edu-servers.net.  172800   IN   A     192.33.14.30
c.edu-servers.net.  172800   IN   A     192.26.92.30
...
```

For redundancy, there are usually several name servers for each zone. Any of them will usually work. Let's pick the first one.

This NS record says that **a.edu-servers.net** is a **.edu** name server.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @198.41.0.4

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26114
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN   A

;; AUTHORITY SECTION:
edu.                172800   IN   NS   a.edu-servers.net.
edu.                172800   IN   NS   b.edu-servers.net.
edu.                172800   IN   NS   c.edu-servers.net.
...

;; ADDITIONAL SECTION:
a.edu-servers.net.  172800   IN   A    192.5.6.30
b.edu-servers.net.  172800   IN   A    192.33.14.30
c.edu-servers.net.  172800   IN   A    192.26.92.30
...
```

This A record helpfully tells us the IP address of the next name server we mean to contact.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @192.5.6.30
```

Next, we query the **.edu** name server. We know the IP address of the **.edu** name server because the root name server gave the information to us.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @192.5.6.30

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36257
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN    A

;; AUTHORITY SECTION:
berkeley.edu.          172800    IN    NS    adns1.berkeley.edu.
berkeley.edu.          172800    IN    NS    adns2.berkeley.edu.
berkeley.edu.          172800    IN    NS    adns3.berkeley.edu.

;; ADDITIONAL SECTION:
adns1.berkeley.edu.    172800    IN    A     128.32.136.3
adns2.berkeley.edu.    172800    IN    A     128.32.136.14
adns3.berkeley.edu.    172800    IN    A     192.107.102.142
...
```

The answer section is blank again. The authority and additional section tell us to query a **berkeley.edu** name server, and provide us with the IP address of the next name server.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @128.32.136.3
```

Next, we query the **berkeley.edu** name server for the IP address of **eecs.berkeley.edu**. We know the IP address of the **berkeley.edu** name server because the root name server gave the information to us.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @128.32.136.3

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52788
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN    A

;; ANSWER SECTION:
eecs.berkeley.edu.   86400    IN    A    23.185.0.1
```

The answer section has one A type record. It tells us that the IP address of **eecs.berkeley.edu** is **23.185.0.1**.

DNS Lookup Walkthrough

```
$ dig +norecurse eecs.berkeley.edu @128.32.136.3

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52788
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;eecs.berkeley.edu.          IN    A

;; ANSWER SECTION:
eecs.berkeley.edu.  86400   IN    A    23.185.0.1
```


Here's the time-to-live (TTL) field in the record. It tells us that we can cache this answer for 86,400 seconds (24 hours).

We finished our original query! Let's try a few other queries.

DNS Lookup Walkthrough

Let's try asking for a different domain.

```
$ dig +norecurse repo.eecs.berkeley.edu @128.32.136.3

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25192
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;repo.eecs.berkeley.edu.          IN    A

;; ANSWER SECTION:
repo.eecs.berkeley.edu.   21600   IN    CNAME   repo-2.eecs.berkeley.edu

repo-2.eecs.berkeley.edu 21600   IN    A       128.32.138.46
```

New record type: CNAME tells us that repo.eecs.berkeley.edu is actually an alias (alternate name) for repo-2.eecs.berkeley.edu.

Then, the A record tells us the IP address for the true name, repo-2.eecs.berkeley.edu.

DNS Lookup Walkthrough

Let's try specifically requesting the IPv6 address.

```
$ dig +norecurse repo.eecs.berkeley.edu AAAA @128.32.136.3

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25192
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;repo.eecs.berkeley.edu.        IN    AAAA

;; AUTHORITY SECTION:
eecs.berkeley.edu. 1971          IN    SOA    ns.eecs.berkeley.edu,
dns.eecs.berkeley.edu, 100012225 10887 3600 604800 86400
```

The answer section is blank, because this domain does not have an IPv6 address.

We got some information about who's operating the zone.

Scaling DNS

Lecture 14, CS 168, Fall 2026

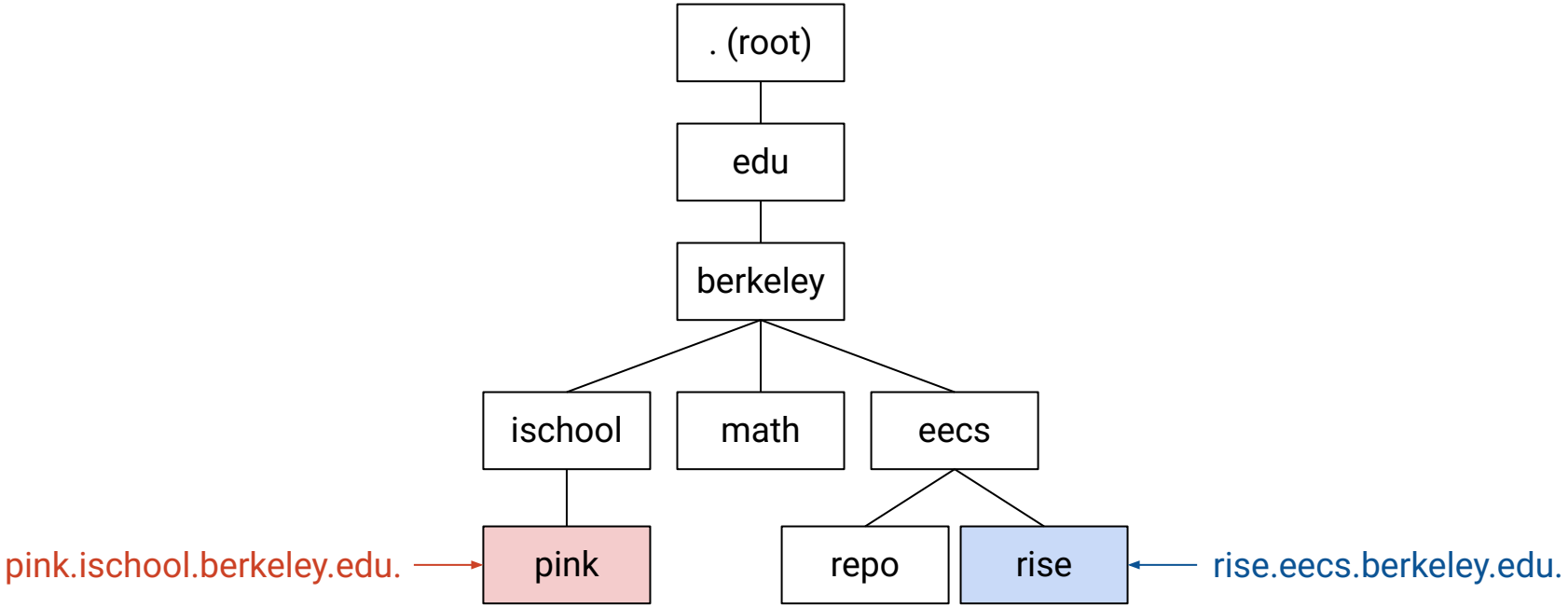
DNS

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

DNS Hierarchy

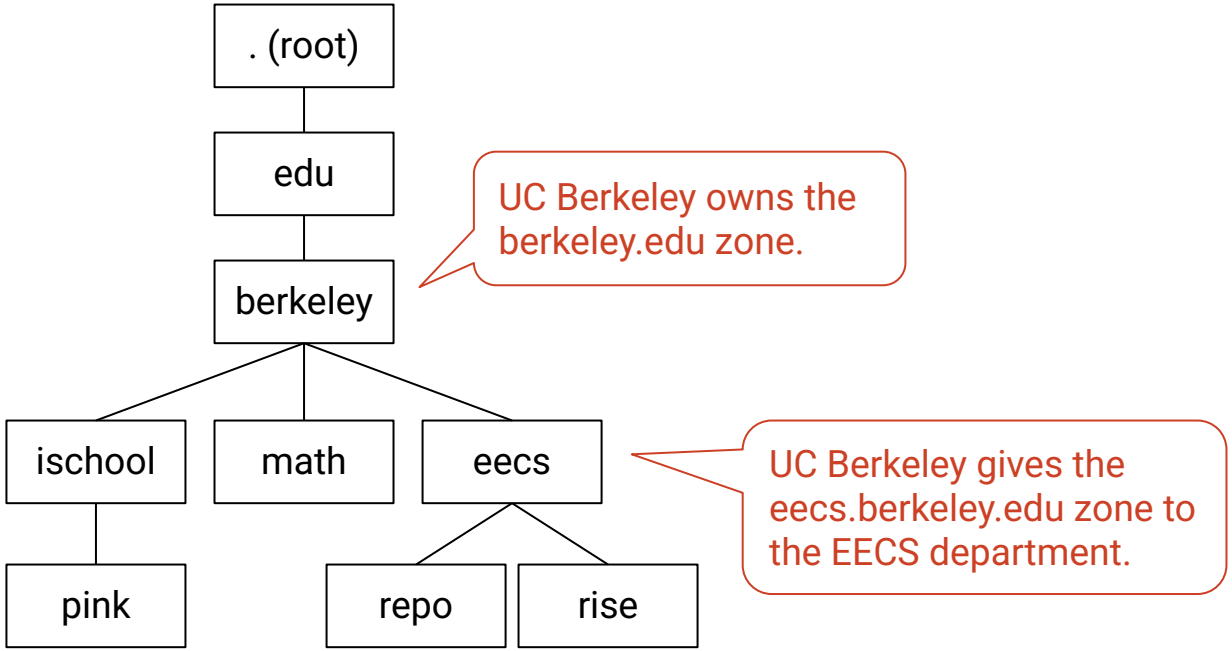
The DNS tree represents 3 forms of hierarchy.

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



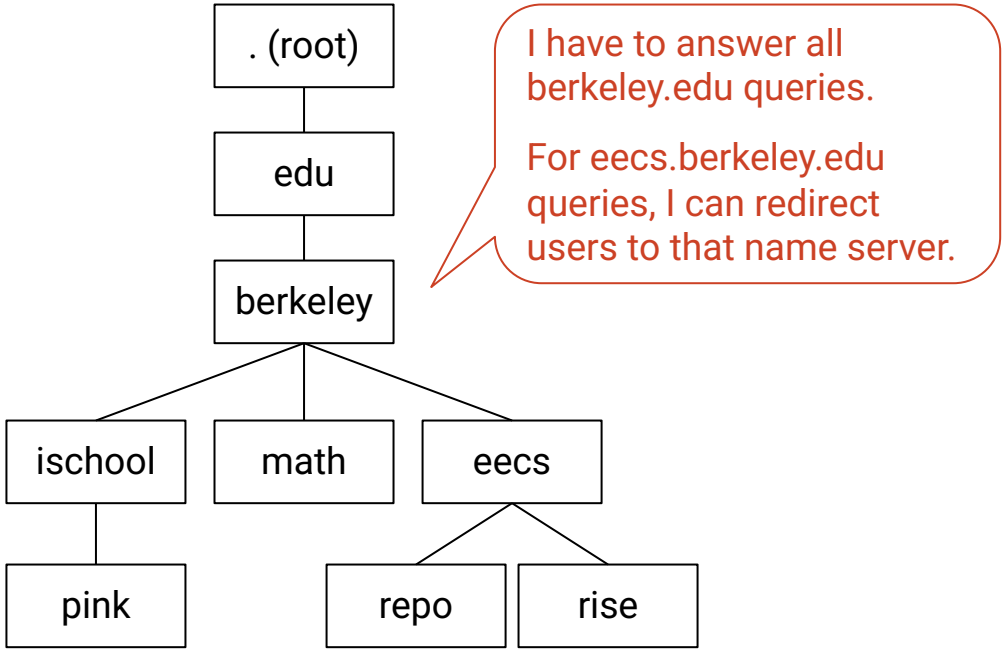
DNS Hierarchy

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.

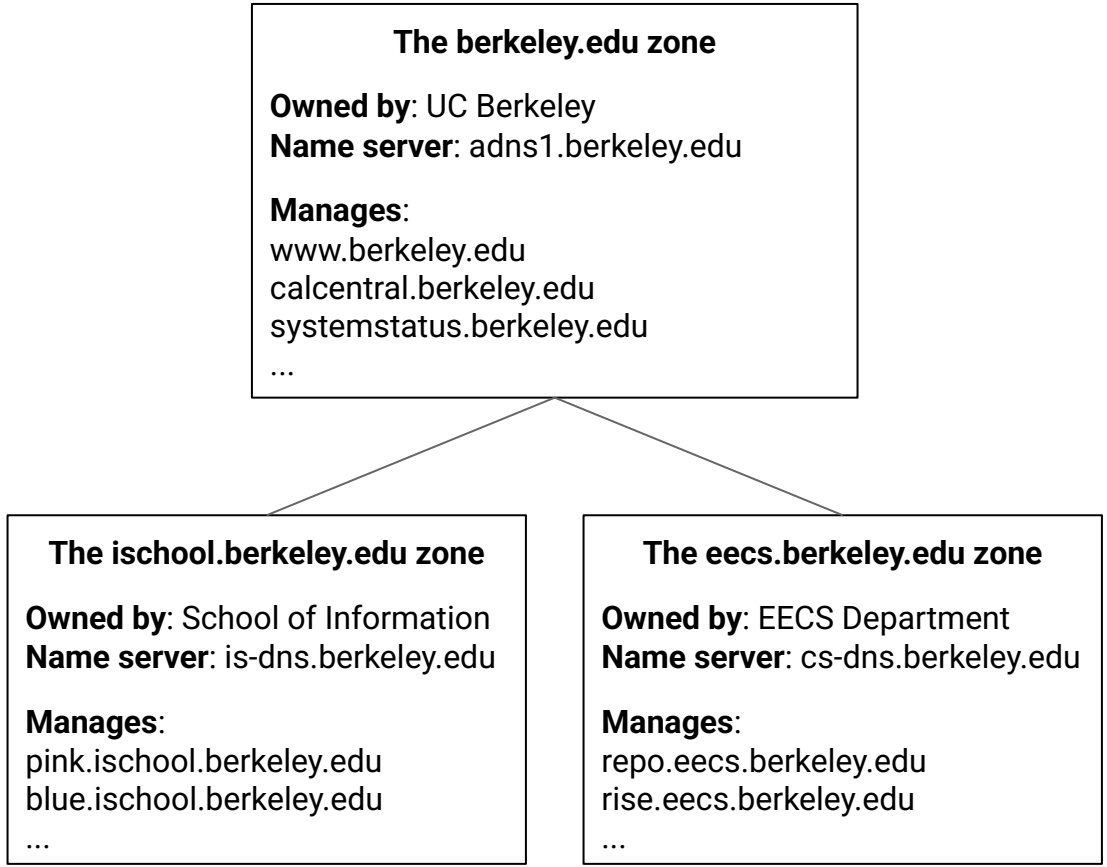


DNS Hierarchy

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.

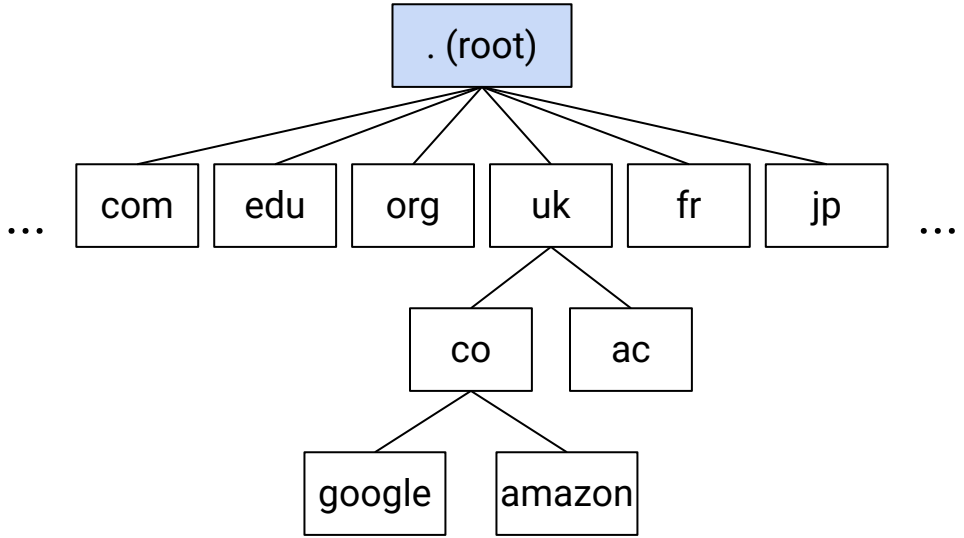
Zones help DNS scale.

- Different administrative authorities are responsible for different parts of the hierarchy.
- Example: Educause (.edu zone operators) don't have to know about what's happening in the berkeley.edu zone (authority delegated to UC Berkeley).

Administrative Authorities

The DNS root is controlled by ICANN.

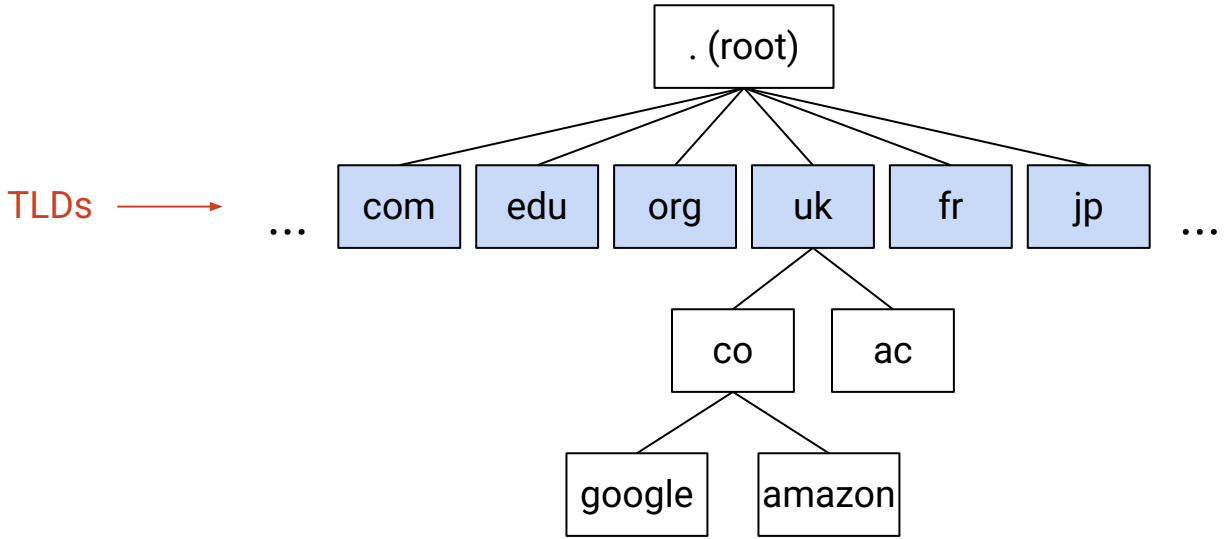
- Internet Corporation for Assigned Names and Numbers.



Administrative Authorities

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

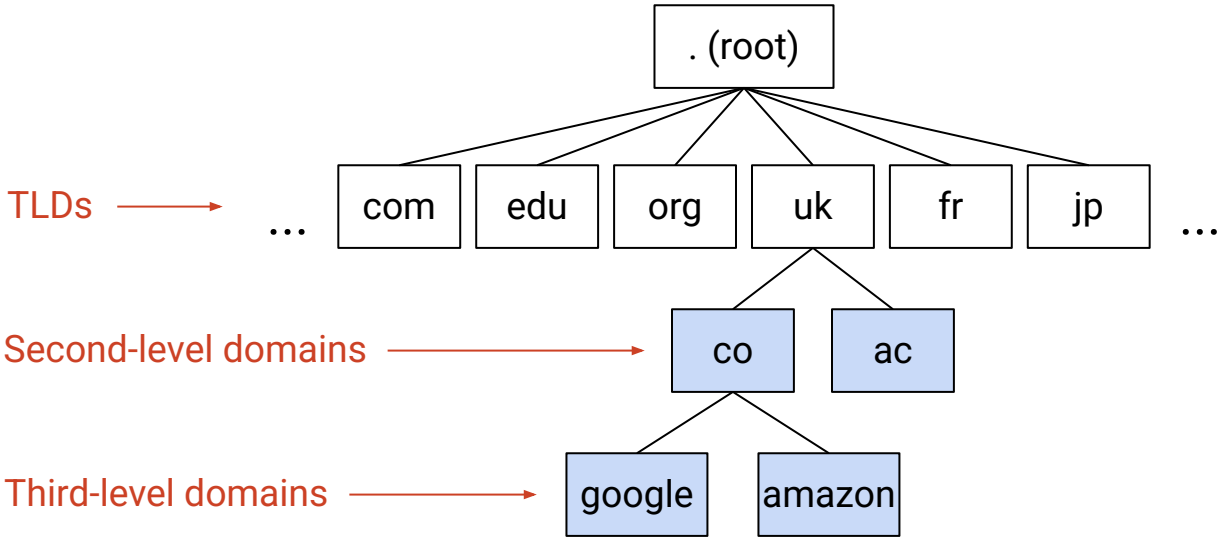
- Historically, relatively few, based on purpose (**org**anization, **edu**cation, **com**mercial, etc.), and country (UK, France, Japan, etc.).
- More recently, many more weird ones (.travel, .pizza, etc.). Over 1500 TLDs today!



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.

Zones "must" have 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.

"Must" = it's standard practice,
and many registrars require it.

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, Fall 2026

DNS

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

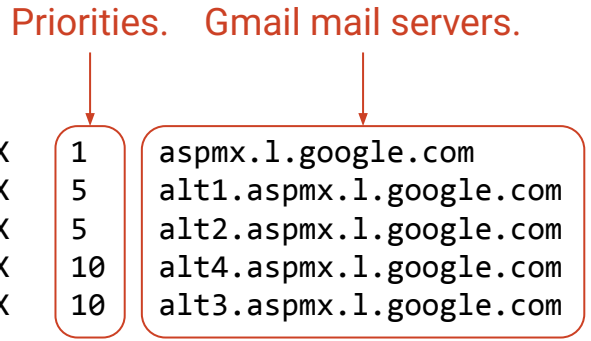
Other Uses of DNS: Email

DNS can store and serve more than just domain-to-IP mappings. Example: Email.

- To send email to evanbot@berkeley.edu, we need to know where to send packets.
- **MX** type records map a domain to a mail server.
  - Historically: The mail server ran on a user's machine.
  - Today: Email services (e.g. Gmail) run mail servers, which receive your mail and let you access your mail.
- MX records contain a priority in the value.
  - If you receive multiple MX records for a domain, try lowest number first.

```
$ dig eecs.berkeley.edu MX

;; ANSWER SECTION:
eecs.berkeley.edu     10549     IN     MX     1     aspmx.l.google.com
eecs.berkeley.edu     10549     IN     MX     5     alt1.aspmx.l.google.com
eecs.berkeley.edu     10549     IN     MX     5     alt2.aspmx.l.google.com
eecs.berkeley.edu     10549     IN     MX     10    alt4.aspmx.l.google.com
eecs.berkeley.edu     10549     IN     MX     10    alt3.aspmx.l.google.com
```



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.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, and organizes data in records of various types.
- DNS extends beyond address resolution into service resolution (e.g. email).
- DNS can be used for load-balancing.