DNS

The Domain Name System

Today in CS

- 31 years ago, John Backus passed away

- Won the Nati
- Won the ACM

- Led the team
- He's the B in

```
<postal-address
<name-part
<personal-part
<street-address
<zip-part
<opt-suffix-part
<opt-apt-num> ::= <apt-num> | ""
art> <name-part>
```

```
HTTP-date    = rfc1123-date | rfc850-date | asctime-date
rfc1123-date = wkday "," SP date1 SP time SP "GMT"
rfc850-date  = weekday "," SP date2 SP time SP "GMT"
asctime-date = wkday SP date3 SP time SP 4DIGIT
date1        = 2DIGIT SP month SP 4DIGIT
               ; day month year (e.g., 02 Jun 1982)
date2        = 2DIGIT "-" month "-" 2DIGIT
               ; day-month-year (e.g., 02-Jun-82)
date3        = month SP ( 2DIGIT | ( SP 1DIGIT ))
               ; month day (e.g., Jun  2)
time         = 2DIGIT ":" 2DIGIT ":" 2DIGIT
               ; 00:00:00 - 23:59:59
wkday        = "Mon" | "Tue" | "Wed"
             | "Thu" | "Fri" | "Sat" | "Sun"
weekday      = "Monday" | "Tuesday" | "Wednesday"
             | "Thursday" | "Friday" | "Saturday" | "Sunday"
month        = "Jan" | "Feb" | "Mar" | "Apr"
             | "May" | "Jun" | "Jul" | "Aug"
             | "Sep" | "Oct" | "Nov" | "Dec"
```

(BNF examples from Wikipedia and RFC 2616)

Where are we?

- Foundations / principles
  - e.g., Packet switching, end-to-end
- Domain structure of the Internet and…
  - routing within domains
  - routing between domains
- Deep dive on IP and TCP
  - What packets are actually composed of (at L3 and L4)
  - How to make an unreliable network (look) reliable
- Today we start looking at things which are more *user-facing*

The Domain Name System (DNS)

- Overview
  - Introduction
  - Name lookup
- Digging into the Details
  - API, servers, and protocol
  - A and NS records
  - How domain names are born
- More DNS
  - Record types and use cases
- A DNS case study
  - Minecraft and SRV records
- Availability, Scalability, and Performance
  - AKA four ways to add more servers
- DNS skepticism
  - Did we name the right thing?
  - Does this thing work right?
  - Is your privacy safe?
  - Does DNS matter?

Thinking back...

- Three early “killer apps” of the Internet (or its precursor, the ARPANET)
    - Remote terminal
        - From local machine, log in to a machine somewhere else (like ssh)
        - `telnet <remote host>`
    - File transfer
        - From local machine, transfer files to/from a remote machine
        - `ftp <remote host>`
    - Email
        - Send/receive messages to/from user of a remote machine
        - `mail <user>@<remote host>`
- .. but numerical host addresses are not so nice for humans!
    - Nobody wants to type **`telnet 46.0.0.10`** !

Numerical addresses vs. humans

- Solution: create an “address book” of host names and their addresses
- Maintained by Elizabeth Jocelyn "Jake" Feinler at the Network Information Center (NIC) at SRI
  - If you wanted a hostname, phone Jake Feinler, she’d enter it in database

- Originally human-readable

Address	Hostname	Computer	Status/System
001	UCLA-NMC	PDP-11/45	User 1/1/74/ANTS
65	UCLA-CCn	IBM 360/91	Server
129	UCLA-CCBS	PDP-10	limited Server
2	SRI-ARC	PDP-10	dedicated Server/TENEX, NLS
66	SRI-AI	PDP-10	limited Server/TENEX
130	SU-DSL	(VDH)->PDP11/20	User 1/74/ANTS
3	UCSB-MOD75	IBM 360/75	Server/OLS
67	SCRL	(VDH)->PDP11/45	User/ANTS
4	UTAH-10	PDP-10	limited Server/TENEX
132	UTAH-TIP		TIP
5	BBN-11X	PDP-11	Peripheral processor for #69
69	BBN-TENEX	PDP-10	Server/TENEX
133	BBN-TENEXB	PDP-10	limited Server/TENEX
6	MIT-MULTICS	H-6180	Server till 12/17/73/Multics
70	MIT-DMS	PDP-10	Server/ITS
134	MIT-AI	PDP-10	Server/ITS
198	MIT-ML	PDP-10	Server/ITS
7	RAND-RCC	IBM 370/158	User
8	SDC-LAB	IBM 370/145	limited Server
9	HARV-10	PDP-10	Server
73	HARV-1	PDP-1	User
137	HARV-11	PDP-11	User
10	LL-67	IBM 360/67	limited Server
74	LL-TX2	TX-2	Server
138	LL-TSP	TSP	User
11	SU-AI	PDP-10	Server/SAIL
12	ILL-CAC	PDP-11/20	User/ANTS
76	ILL-NTS	PDP-11/50	User/ANTS
140	UNIVAC	UNIVAC 1616	1/15/74
13	CASE-10	PDP-10	Server/TENEX
14	CMU-10B	PDP-10	Server
78	CMU-10A	PDP-10	Server
15	I4-TENEX	PDP-10	limited Server/TENEX
79	I4-TENEX	PDP-11	Peripheral processor for #15
16	AMES-67	IBM 360/67	Server
144	AMES-TIP		TIP
208	AMES-11	PDP-11/45	User 12/73
145	MITRE-TIP		TIP
146	RADC-TIP		TIP
19	NBS-ICST	PDP-11/45	User/ANTS
147	NBS-TIP		TIP

Address	Hostname	Computer	Status/System
148	ETAC-TIP		TIP
21	LLL-RISOS	PDP-11/45	User/RATS
22	ISI-SPEECH11	PDP-11/45	User 1/74
86	USC-ISI	PDP-10	Server/TENEX
150	ISI-DEVTENEX	PDP-10	User 1/74/TENEX
23	USC-44	IBM 360/44	Server
151	USC-TIP		TIP
152	GWC-TIP		TIP
153	DOCB-TIP		TIP
26	SDAC-44	IBM 360/44	User
154	SDAC-TIP		TIP
28	ARPA-DMS	PDP-15	User
156	ARPA-TIP		TIP
29	BRL	PDP-11/40	User/ANTS
158	BBN-TESTIP		TIP
31	CCA-TENEX	PDP-10	dedicated Server/TENEX
95	LL-LANTS	PDP-11/40	User 2/74/ANTS
159	CCA-TIP		TIP
32	PARC-MAXC	(Nova)->MAXC	limited Server/TENEX
96	PARC-VTS	Nova 800	User
160	PARC-11	PDP-11	User 1/74
33	FNWC	CDC 6500	2/74
161	FNWC-TIP		TIP
98	UCB	PDP-11/45	User 1/74
35	UCSD-CC	B6700	Server
36	HAWAII-ALOHA	HP 2100	12/73
100	HAWAII- 500	BCC 500	1/74
164	ALOHA-TIP		TIP
165	RML-TIP		TIP
40	BBN-NCC	H-316	User
168	NCC-TIP		TIP
232	BBN-1D	PDP-1	User
169	NORSAR-TIP		TIP
42	UKICS-360	IBM 360/195	limited Server
170	UKICS-TIP		TIP
43	OFFICE-1	PDP-10	dedicated Server/TENEX, NLS
171	TYMSHARE-TIP		TIP
44	MIT-MULTICS	H-6180	Server 12/17/73/Multics
174	RUTGERS-TIP		TIP
175	WPAFB-TIP		TIP

Numerical addresses vs. humans

- Solution: create an “address book” of host names and their addresses
- Maintained by Elizabeth Jocelyn "Jake" Feinler at the Network Information Center (NIC) at SRI
  - If you wanted a hostname, phone Jake Feinler, she’d enter it in database

- Originally human-readable

- Eventually a standardized format (“<NETINFO>HOSTS.TXT” aka “hosts.txt”)
  - Machines could consume this directly
  - Everyone periodically uses FTP to fetch HOSTS.TXT from the NIC

```
HOST : 10.0.0.1 : UCLA-CS,UCLA-CECS : VAX-11/750 : LOCUS : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.16 : AMES-TSS,AMES-67,AMES : IBM-360/67 : TSS/360 : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.22 : ISI-SPEECH11 : PDP-11/45 : EPOS : TCP/TFTP :
HOST : 10.0.0.23 : USC-ECLB,ECLB : DEC-1090B : TOPS20 : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.26 : PENTAGON-TAC : H-316 : TAC : TCP :
HOST : 10.0.0.27 : USC-ISID,ISID : DEC-2060T : TOPS20 : TCP/TELNET,TCP/SMTP,TCP/FTP,TCP/TFTP,TCP/FINGER :
HOST : 10.0.0.32 : PARC-MAXC,PARC : MAXC : TENEX : TCP/FTP,TCP/SMTP,TCP/TELNET :
HOST : 10.0.0.34 : LBL-NMM,NMM : VAX-11/780 : VMS : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.37, 128.10.0.1 : PURDUE,PURDUE-CS,PURDUE-TCP,PURDUE-PVAX,PVAX : VAX-11/780 : UNIX : TCP/FTP,TCP/TELNET,T
HOST : 10.0.0.62 : UTEXAS-11 : PDP-11/70 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.68 : USGS1-MULTICS,RESTON,REST : H-60/68 : MULTICS : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.70 : USGS3-MULTICS,MENLO : H-6880 : MULTICS : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.73 : SRI-NIC,NI[illegible]FOONLY-F3 : TENEX : TCP/TELNET,TCP/SMTP,TCP/TIME,TCP/FTP,NCP/FTP,NCP/TELNET :;Reclama
HOST : 10.0.0.78 : UCB-ARPA [illegible]/780 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,UDP :
HOST : 10.0.0.87 : SANDIA,SNL [illegible]EC-2060T : TOPS20 : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.90 : LANL : VAX-11/750 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.0.0.91 : WASHINGTON,UDUB,UW-WARD : DEC-2060 : TOPS20 : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.1.0.1 : UCLA-CCN,CCN : IBM-370/3033 : OS/MVS : TCP/TELNET,TCP/FTP,TCP/SMTP,NCP/TELNET,NCP/FTP :;Reclama
HOST : 10.1.0.6 : MIT-DMS,DMS : DEC-1040 : ITS : TCP/TELNET,TCP/FTP,TCP/SMTP,TCP/FINGER :
HOST : 10.1.0.14 : CMU-CS-A,CMU-10A,CMUA : DEC-1080 : TOPS10 : TCP/TELNET,TCP/FTP,TCP/SMTP,TCP/FINGER,ICMP,NCP :;Recla
HOST : 10.1.0.94, 192.5.2.3 : UWISC,CSNET-SH,CSNETB,CSNET,WISCONSIN : VAX-11/750 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP
HOST : 10.2.0.9 : YALE : VAX-11/750 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.2.0.58 : RUTGERS,RUTGERS-20,RUTGERS-10,RU-RED : DEC-2060T : TOPS20 : TCP/TELNET,TCP/FTP,TCP/SMTP,TCP/FINGER
HOST : 10.2.0.78 : UCB-VAX,BERKELEY,UCB-C70 : VAX-11/750 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP,UDP :
HOST : 10.3.0.14 : CMU-CS-C,CMU-20C,CMUC : DEC-2060 : TOPS20 : TCP/TELNET,TCP/FTP,TCP/SMTP,TCP/FINGER,ICMP :
HOST : 10.3.0.24 : WHARTON-10,WHARTON : PLURIBUS : VDA : NCP/TELNET,NCP/FTP,TCP/FTP :
HOST : 10.3.0.96 : CORNELL : VAX-11/780 : UNIX : TCP/TELNET,TCP/FTP,TCP/SMTP :
HOST : 10.5.0.53 : MARTIN,MMC : PDP-11/45 : RSX : TCP/TELNET,TCP/FTP :;Reclama
. . .
```

Berkeley also had its own network now!

```
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
HOST : 46.0.0.11 : UCBKIM : VAX-11/780 : UN
HOST : 46.0.0.12 : UCBCALDER : VAX-11/750
HOST : 46.0.0.13 : UCBDALI : VAX-11/750 :
HOST : 46.0.0.14 : UCBMATISSE : VAX-11/750
HOST : 46.0.0.15 : UCBMEDEA : VAX-11/750 :
HOST : 46.0.0.19 : UCBINGRES : VAX-11/780
```

You type:

`$ telnet UCBVAX`

- UCBVAX looked up in hosts file
- opens connection to 46.0.0.10
- much nicer than **`telnet 46.0.0.10`**!

Class A network! Like a /8 !
By 1986, these were all 128.32.0.x instead (Class B — Berkeley still has this /16)

Numerical addresses vs. humans

- Solution: create an “address book” of host names and their addresses
- Maintained by Elizabeth Jocelyn "Jake" Feinler at the Network Information Center (NIC) at SRI
  - If you wanted a hostname, phone Jake Feinler, she’d enter it in database
- Originally human-readable
- Eventually a standardized format (“<NETINFO>HOSTS.TXT” aka “hosts.txt”)
  - Machines could consume this directly
  - Everyone periodically uses FTP to fetch HOSTS.TXT from the NIC
- But this wasn’t ideal…

Numerical addresses vs. humans

- Increasing amount of work for Jake Feinler and her team!
- Increasing amount of data transfer!
    - As networks grows (more hosts)
        - file size increases
        - number of hosts fetching it increases
        - frequency with which you fetch to remain up to date increases
        - .. absolute best case is that this is quadratic!
    - .. were starting to be a *lot* more hosts (e.g., due to rise of workstations)
- Longer transfers more likely to fail; may end up with partial hosts file!

- In short:
    - Centralized administration was burdensome and counter to “open” trend
    - Centralized distribution of (increasingly) large file was bad news

The Domain Name System

- DNS developed to confront the problems being faced
    - Developed by Paul Mockapetris; RFC in 1983
    - He was given the task of pulling several proposals into a final one…
        - .. just developed his own one instead!
        - .. without much change, we still use it today

The Domain Name System: Goals

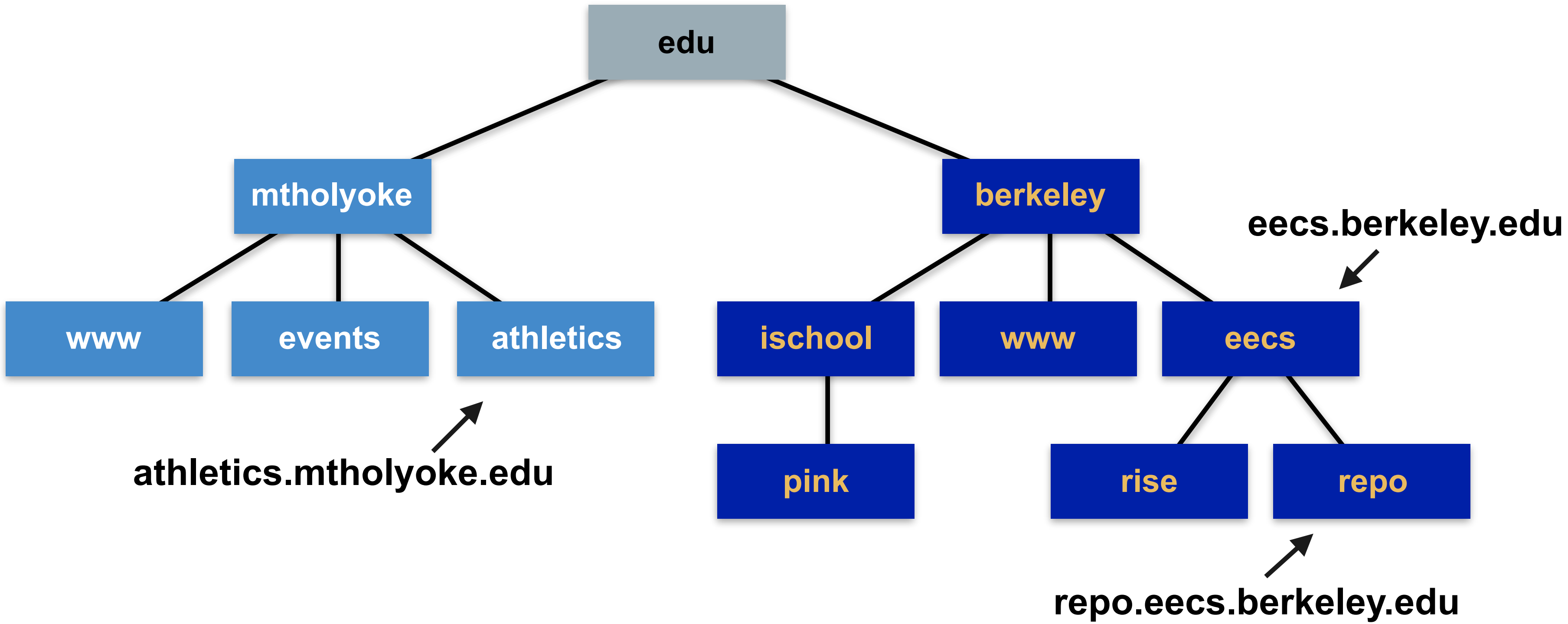
- Primary purpose: map from human-friendly names to IP addresses

- Deal with scale!
    - Many hosts/names
    - Many name/address lookups
    - Many updates (can't have bottleneck at NIC)
- Be highly available
    - No single point of failure (what if the NIC's FTP server was down?)
- Perform well
    - Lots of communication starts with a name lookup!

- How do you solve these problems?
    - ~~Hierarchy!~~
    - Three intertwined *hierarchies*!

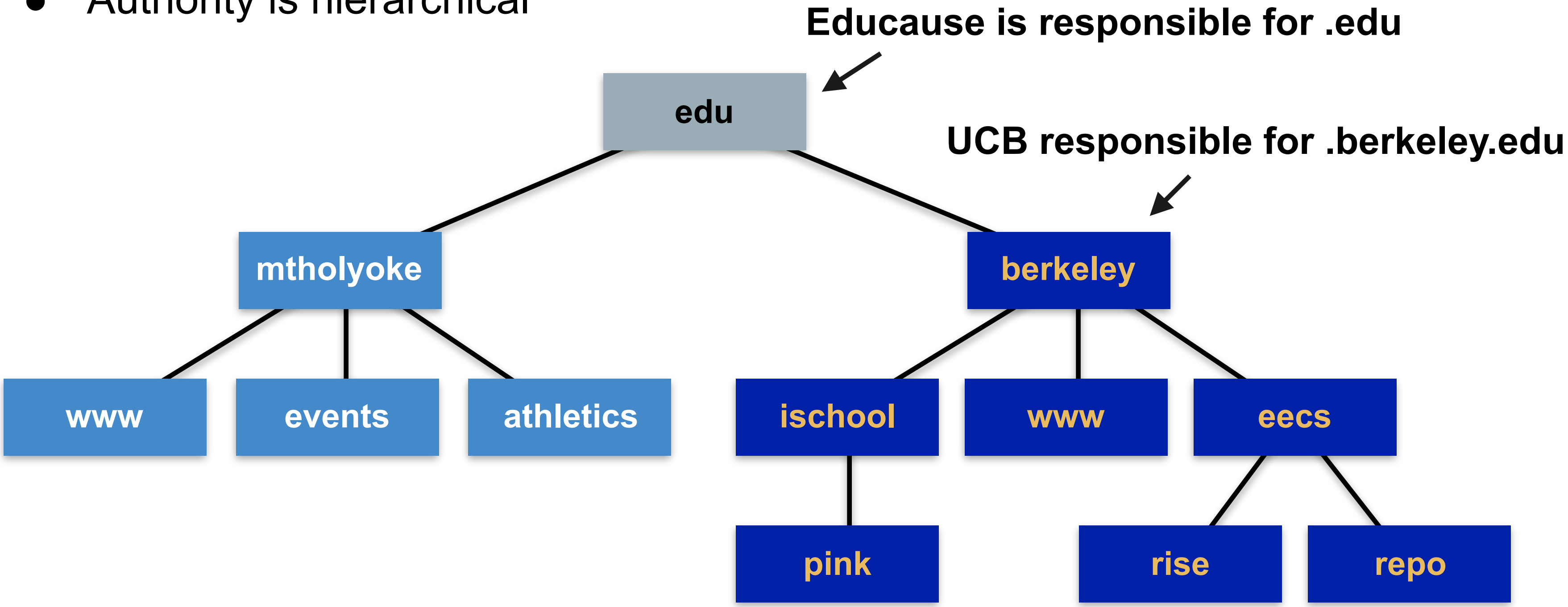
The Domain Name System: Hierarchies

- Names are hierarchical



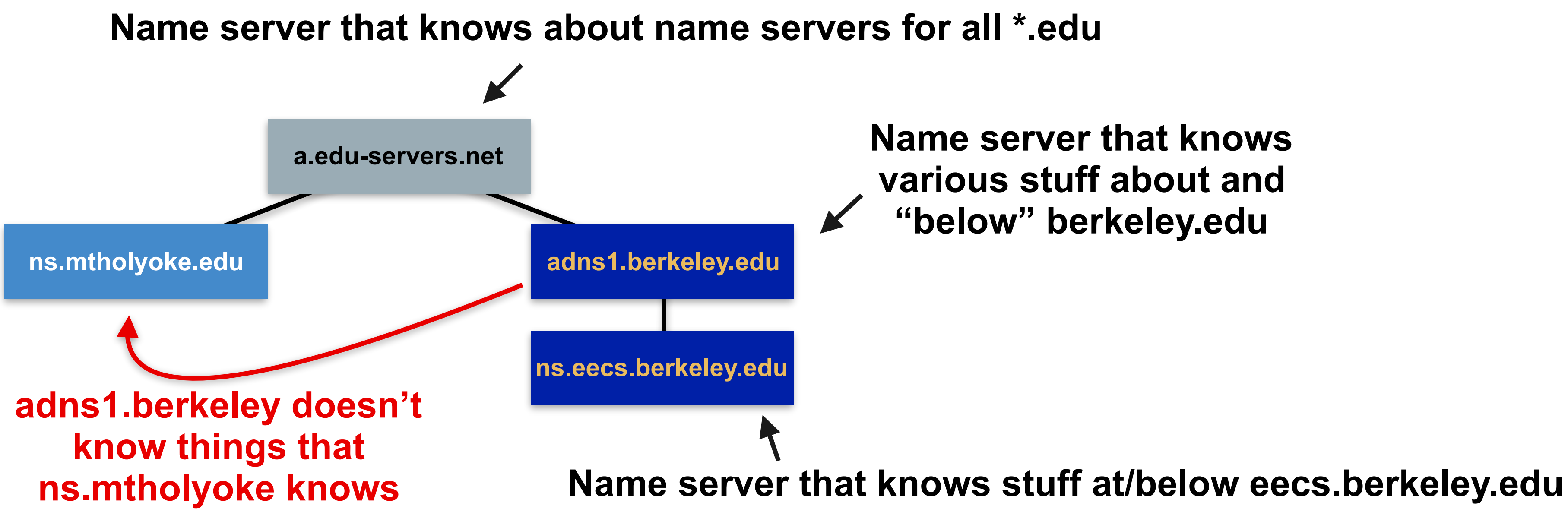
The Domain Name System: Hierarchies

- Authority is hierarchical

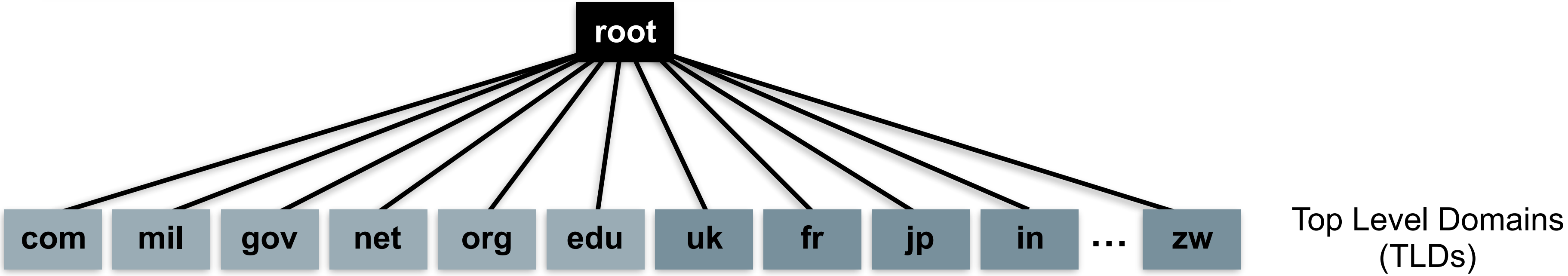


The Domain Name System: Hierarchies

- Infrastructure is hierarchical
  - Infrastructure is not just a single server that knows all the names
  - It's a *hierarchy* of *name servers* which know parts of the hierarchy

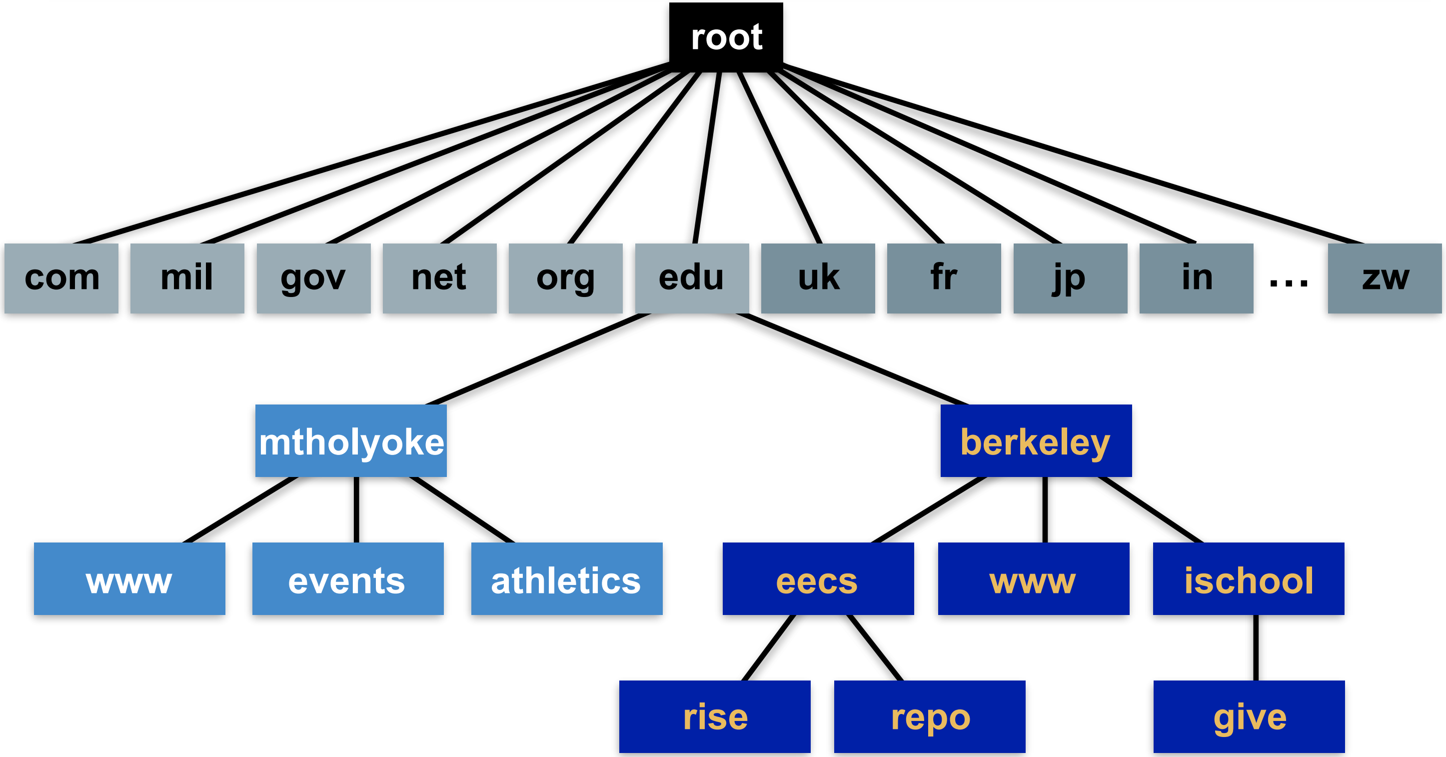


DNS: Bigger Picture



- DNS root
  - Controlled by ICANN
- Top Level Domains (TLDs)
  - Controlled by Educause (.edu), Verisign (.net, .com), AFNIC (.fr), US Government (.gov), etc., etc. (1,515 as of March 2020)

DNS: Bigger Picture



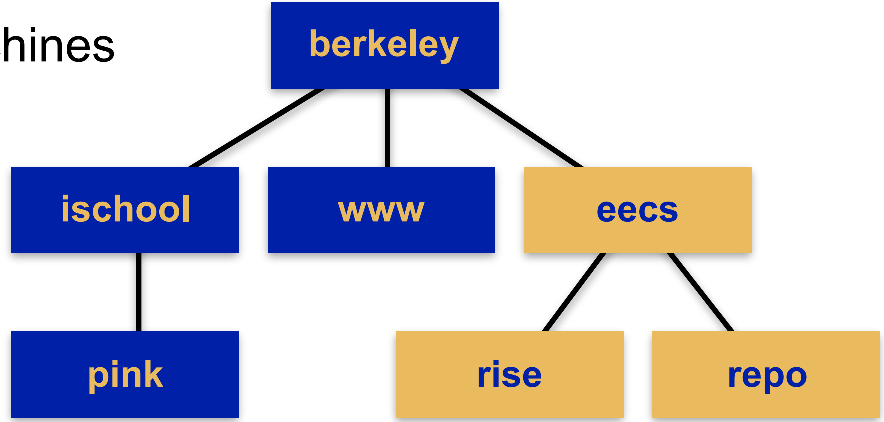
Top Level Domains (TLDs)

nth-level domains

Controlled by many, many organizations!

DNS: Zones, Authority, Delegation

- A *zone* corresponds to an administrative authority responsible for contiguous portion of hierarchy
- UCB controls *.berkeley.edu and *.ischool.berkeley.edu
- EECS controls *.eecs.berkeley.edu
- .. you have choice of whether/where to delegate authority of children
- .. means EECS doesn't need to coordinate with main campus IT to name EECS machines



DNS: Name lookup

- “Iterative” resolution process:
  - Start with root name server
  - Ask for the name you want
  - If it has an answer — you’re done!
  - If not, it will direct you to next name server to ask

DNS: Name resolution

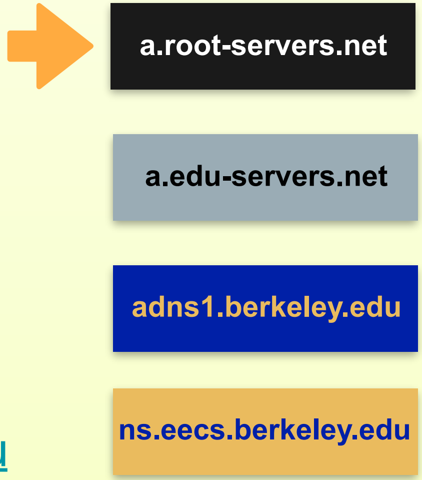
Example: Let's look up (or *resolve*) **repo.eecs.berkeley.edu**

1. Ask a.root-servers.net for repo.eecs.berkeley.edu
2. It won't know, but it will tell you who to ask: a.edu-servers.net

3. Ask a.edu-servers.net for repo.eecs.berkeley.edu
4. It won't know, but it will tell you who to ask: adns1.berkeley.edu

5. Ask adns1.berkeley.edu for repo.eecs.berkeley.edu
6. It won't know, but it will tell you who to ask: ns.eecs.berkeley.edu

7. Ask ns.eecs.berkeley.edu for repo.eecs.berkeley.edu
8. It will tell you: 128.32.138.46 !



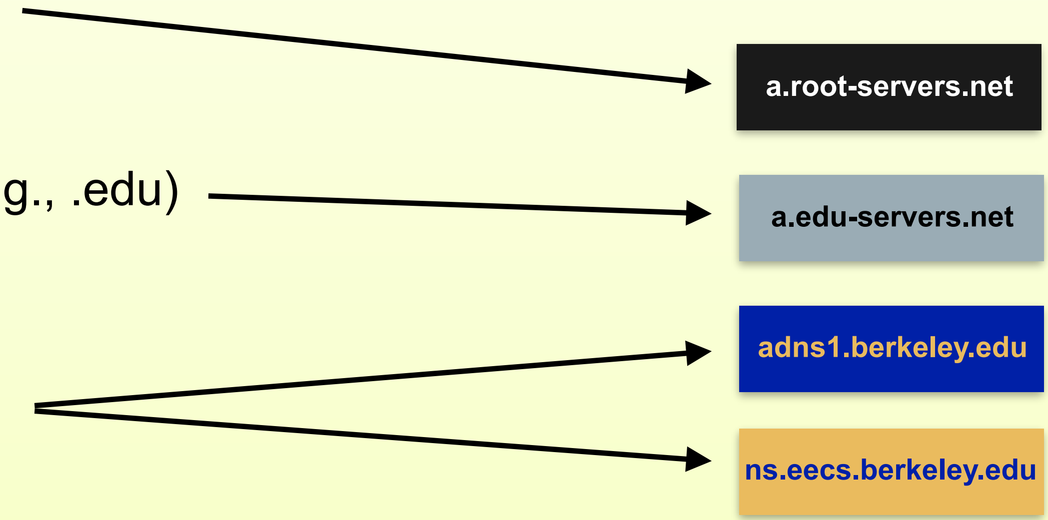
DNS Sidenote: Classes of name servers

Root server
Knows about all the TLD servers

TLD server
Knows about a particular TLD (e.g., .edu)

Authoritative servers
Know about stuff in their zone
Actually do name to IP mapping!

Can be operated by an organization itself (e.g., UCB), or by a service provider



DNS: Name lookup

- "Iterative" resolution process:
  - Start with root name server
  - Ask for the name you want
  - If it has an answer — you're done!
  - If not, it will direct you to next name server to ask

DNS: Name lookup

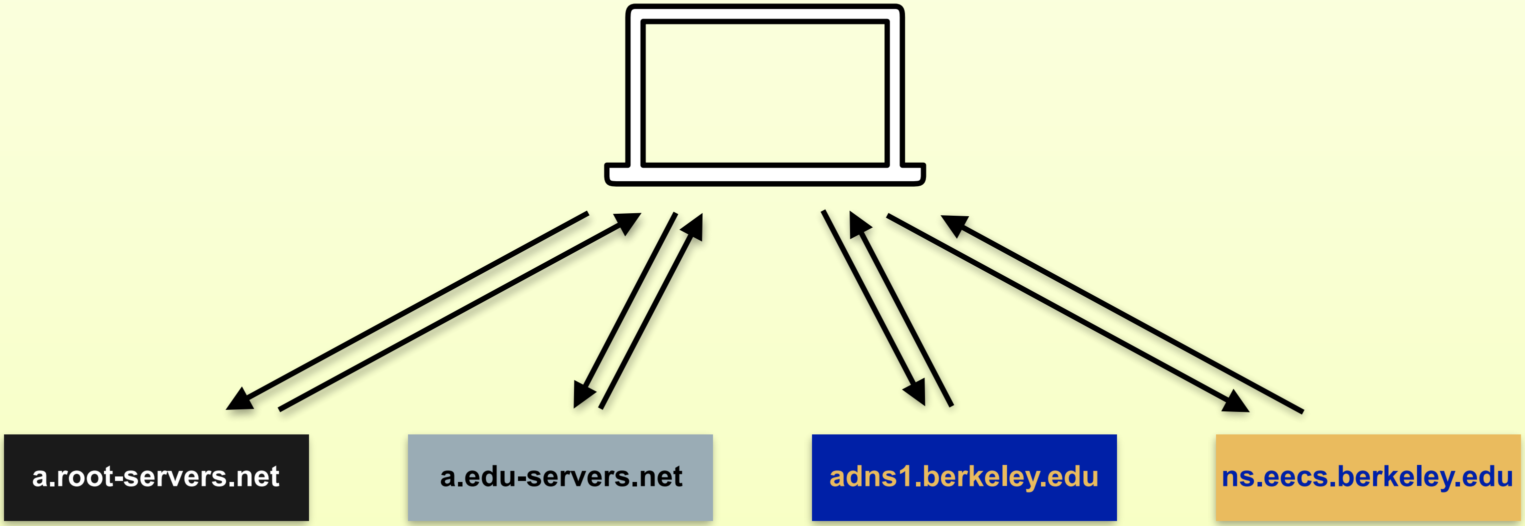
- “Iterative” resolution process:
  - Start with root name server
  - Ask for the name you want
  - If it has an answer — you’re done!
  - If not, it will direct you to next name server to ask

- Three important questions here:
  - 1) Who actually does this multi-step lookup process?

DNS: Name lookup

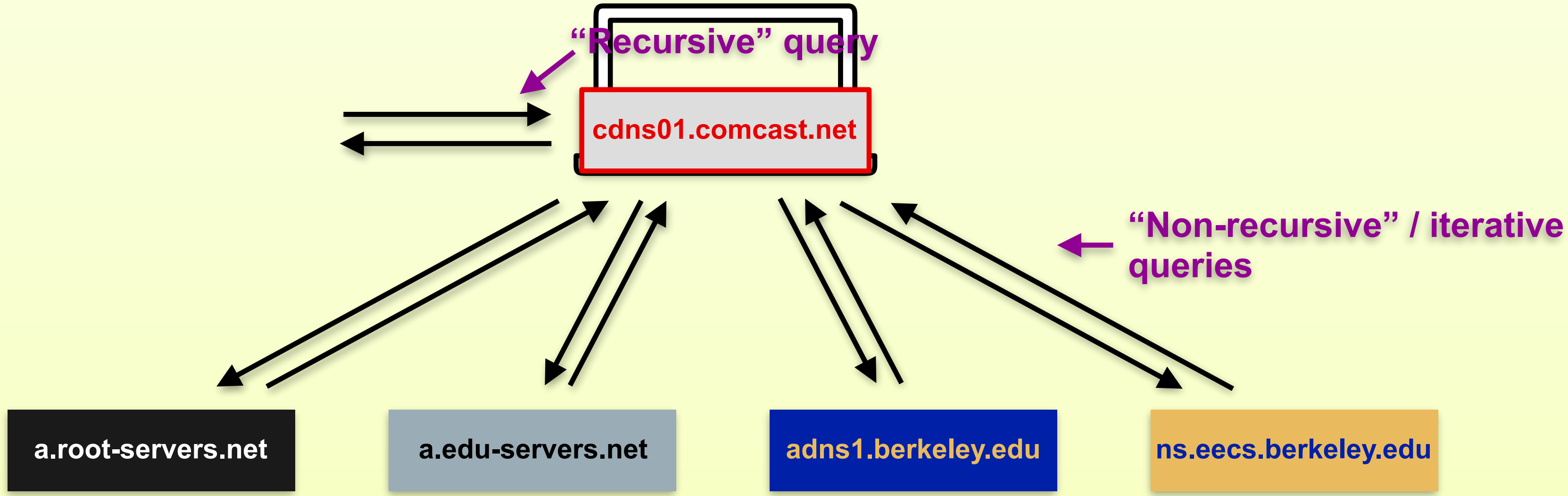
- Who actually does this multi-step lookup process?

- Originally, likely that host did it directly



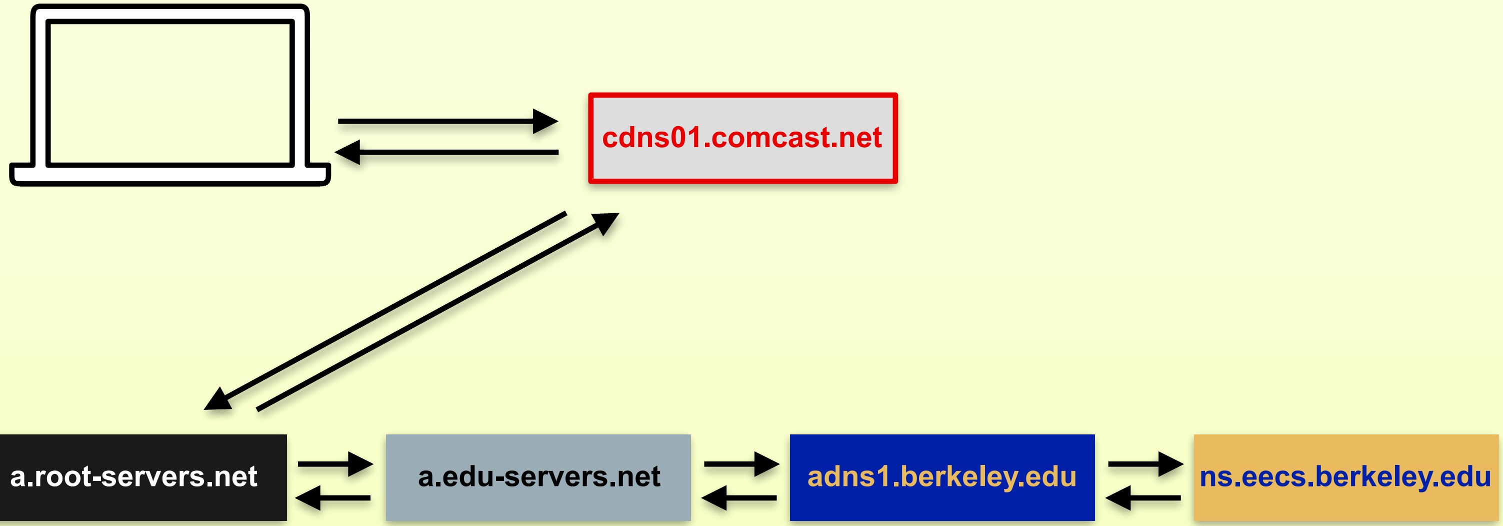
DNS: Name lookup

- Who actually does this multi-step lookup process?
- Today, usually done by a *resolving name server*



DNS: Name lookup

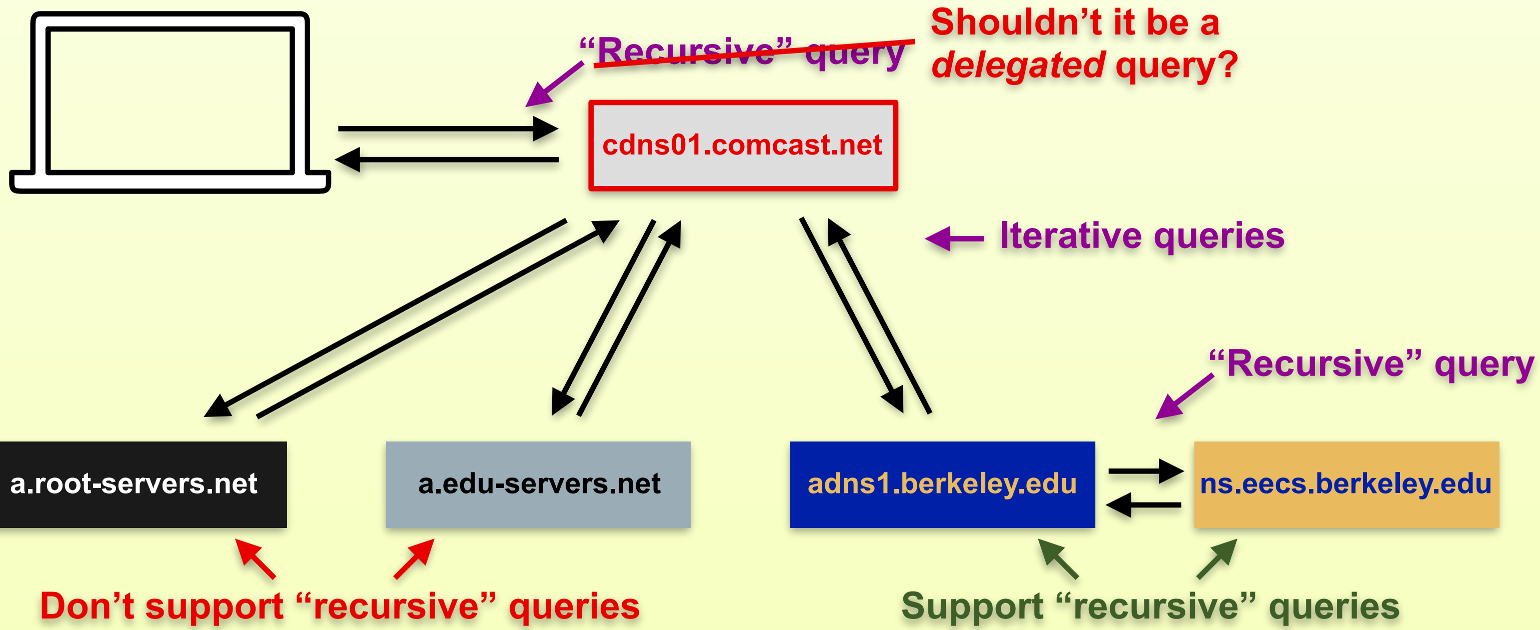
- Who actually does this multi-step lookup process?
- Today, usually done by a *resolving name server*



These servers don't support "recursive" queries — it's harder and they're busy!

DNS: Name lookup

- Who actually does this multi-step lookup process?
- Today, usually done by a *resolving name server*



DNS: Name lookup

- When a server gets a request for a normal/non-recursive query:
  - If server knows the answer — return answer!
  - If not — return reference to next server to query

- When a server gets a request for a recursive query:
  - If server knows the answer — return answer!
  - In theory, *could* perform recursive query on "next" server ← **Truly recursive**
  - More likely this server does the "iterative" process itself ← **Not really recursive?**
  - Even more likely: return an error saying you don't support "recursion"

  - .. usually only specialized resolving servers support these queries
    - Often provided by your ISP
    - Generally aren't authoritative for any domain (don't have specific name-to-IP mappings that they're responsible for)

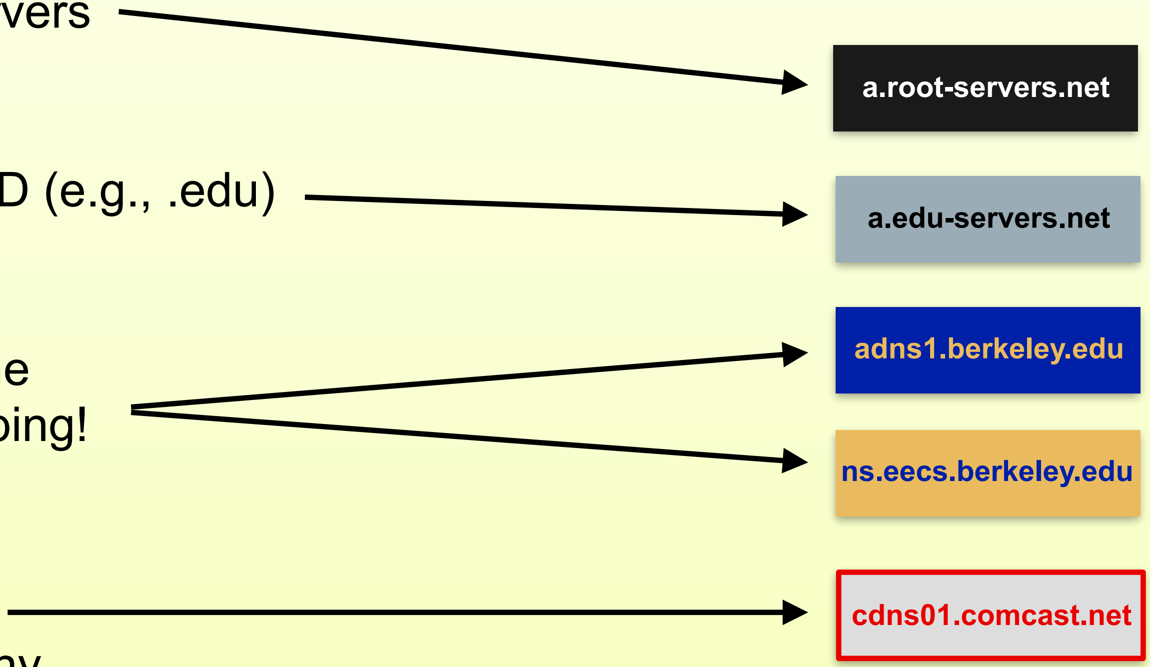
DNS Sidenote: Classes of name servers

Root server
Knows about all the TLD servers

TLD server
Knows about a particular TLD (e.g., .edu)

Authoritative servers
Know about stuff in their zone
Actually do name to IP mapping!

Resolving DNS servers
Just for delegating lookups
Not really part of the hierarchy



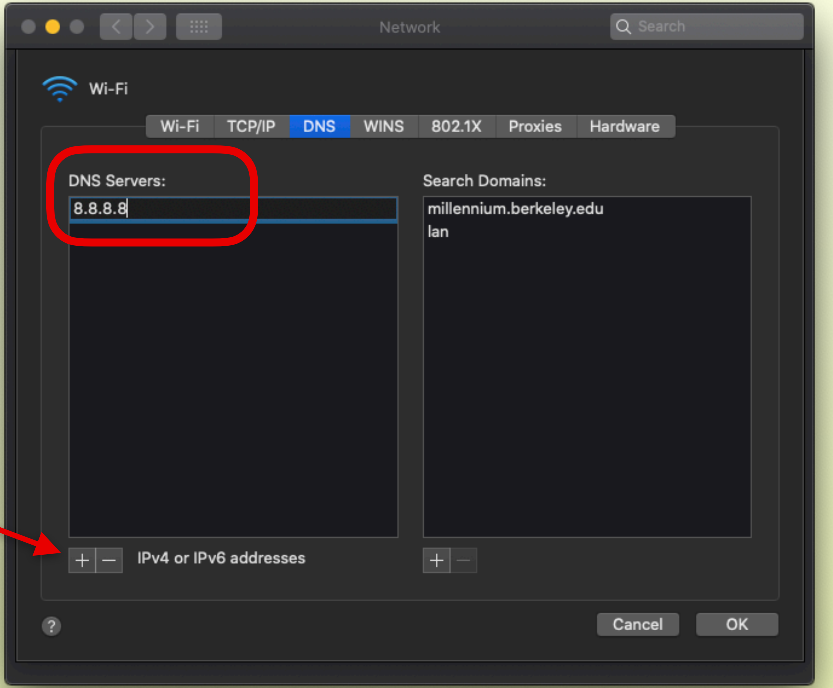
DNS: Name lookup

- "Iterative" resolution process:
  - Start with root name server
  - Ask for the name you want
  - If it has an answer — you're done!
  - If not, it will direct you to next name server to ask

- Three important questions here:
  - 1) Who actually does this multi-step lookup process?
    - A host can do it, but probably delegates it to a *resolving DNS server*
  - 2) How do I know the address of my resolving DNS server?

DNS: Name lookup

- How do you know the address of your resolving DNS server?
- Possibly: Configure it manually
- More likely: DHCP
  Dynamic Host Configuration Protocol
  (We'll cover later)
- Note: You can have more than one!
  Depending on OS/config, may cycle through them, or switch to later one if earlier one fails, or…



DNS: Name lookup

- “Iterative” resolution process:
  - Start with root name server
  - Ask for the name you want
  - If it has an answer — you’re done!
  - If not, it will direct you to next name server to ask

- Three important questions here:
  - 1) Who actually does this multi-step lookup process?
    - A host can do it, but probably delegates it to a *resolving DNS server*
  - 2) How do I know the address of my resolving DNS server?
    - Could be manual, but probably via DHCP (more later)
  - 3) How does anyone know the address of the root DNS server?!
    - First, must come clean about a fib…

DNS: Availability (Preview)

- I’ve been acting like there’s one root name server, like Berkeley had one name server, and so on
- This is already somewhat resilient to failure…
  - Berkeley’s name server could go down; would not affect UCLA
- But actually, every zone always has *at least* two name servers (replicas)

  By IETF decree, more or less
  - The main Berkeley zone has at least:

    `adns1.berkeley.edu` - 128.32.136.3

    `adns2.berkeley.edu` - 128.32.136.14

    `adns3.berkeley.edu` - 192.107.102.142
  - .edu has 13 (“a” through “m” `.edu-servers.net`)
  - root also has 13 (“a” through “m” `.root-servers.net`)
    - .. actually, more. We’ll come back to this.

DNS: Name lookup

- “Iterative” resolution process:
    - Start with root name server
    - Ask for the name you want
    - If it has an answer — you’re done!
    - If not, it will direct you to next name server to ask

- Three important questions here:
    - 1) Who actually does this multi-step lookup process?
        - A host can do it, but probably delegates it to a *resolving DNS server*
    - 2) How do I know the address of my resolving DNS server?
        - Could be manual, but probably via DHCP (more later)
    - 3) How does anyone know the address of ~~the~~ a root DNS server?!

DNS: Name lookup

- How do you know the address of a root name server?!

- You know it’s named a.root-servers.net or b.root-servers.net, etc., but…
  - Where do you look that up to find the IP address?!
  - Bit of a chicken and egg problem here

- Multiple ways, but a decent solution isn’t too complicated…
  - Program that does name resolution ships with root server IP addresses (possibly hard coded, possibly in default config file)
  - Try query those pre-configured addresses until you find one that works
  - Ask it for an up-to-date list
    - Called a *priming* query
  - Works as long as at least one of the pre-configured ones still works

DNS: Name lookup

- "Iterative" resolution process:
    - Start with root name server
    - Ask for the name you want
    - If it has an answer — you're done!
    - If not, it will direct you to next name server to ask

- Three important questions here:
    - 1) Who actually does this multi-step lookup process?
        - A host can do it, but probably delegates it to a *resolving DNS server*
    - 2) How do I know the address of my resolving DNS server?
        - Could be manual, but probably via DHCP (more later)
    - 3) How does anyone know the address of a root DNS server?!
        - Preconfigured addresses; use those to get updated ones (*priming*)

DNS: Name lookup

- A final note on lookup…

- Remember that HOSTS.TXT file from the pre-DNS world?

- Legacy of it remains today on many systems…
  - `/etc/hosts` on Unix-like systems (macOS, Linux, …)
  - `C:\Windows\System32\Drivers\etc\hosts` on Windows

  - .. but there's usually not much in it!

DNS

Digging into the Details

DNS: Digging into the details

- APIs
- Servers
- Protocol
- How a domain is born

DNS: API, servers, and protocol

- The usual API functions:

    - `result = gethostbyname("example.com");`
        - Very old; deprecated for many years
        - Wildly common in real code anyway
        - Limited to IPv4
    - `error = getaddrinfo("example.com", NULL, NULL, &result);`
        - Modern
        - Not limited to IPv4

    - Available in C on Unix-like systems and Windows
    - Available in Python `socket` module

- These usually just make a request to the configured resolving DNS server

DNS: API, servers, and protocol

- Gold standard DNS server: BIND
  - First DNS server for Unix
  - Written by four Berkeley grad students in 1983 (same year as DNS RFC)
  - Berkeley Internet Name Domain Server
    - .. why not Berkeley Internet Name Daemon?!

The Berkeley Internet Name Domain Server

Douglas B. Terry, Mark Painter, David W. Riggle, and Songnian Zhou

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

ABSTRACT

The Berkeley Internet Name Domain (BIND) Server allows a

Sidenote: Daemons

- Many network server processes are called *daemons*
  - The main program of BIND is called “named” (name daemon)
  - SSH server is “sshd”
- Not strictly just network servers
  - Sometimes referred to as “background” or “non-interactive” processes
  - Sort of misleading — a name server is certainly interactive!
  - .. but not generally run/used directly from command line
  - Roughly: A daemon is a “server process”
  - Generally long lived
  - Generally communicated with via some sort of IPC or network
  - Equivalent programs in Windows world usually called *services*
- Why “daemon”?
  - Goes back at least as far as Descartes…

DNS: API, servers, and protocol

- Gold standard DNS server: BIND
  - First DNS server for Unix
  - Written by four Berkeley grad students in 1983 (same year as DNS RFC)
  - Berkeley Internet Name Domain Server
  - Perhaps it should not be surprising…
    - .. berkeley.edu is the oldest .edu domain on the Internet!

DNS: Protocol

- Client/Server design
  - Client is often a user host; could be another server (e.g., recursive query)
  - Client sends query
  - Server replies with response

- Server typically listens on well-known UDP port 53

- Why UDP?
  - Saves RTT for TCP connection establishment
  - TCP requires servers to keep state per connection… *lots* of connections
  - No real need for ordered stream abstraction; a single packet is often fine

- But wait… UDP is not reliable! What if packets are dropped?
  - Simple timeout/retry mechanism
  - Varies from OS to OS, etc. (but can be fairly slow)

DNS: Protocol

- Some DNS servers also use TCP port 53
  - Not usually used for normal queries
  - Primarily used for "zone transfers" (replicating name database)
    - This is much more data than a normal query!
    - Three-way handshake likely negligible; reliability/ordering important
- We'll talk about some more variants of the protocol later…

DNS: Protocol

- All messages share the same basic format

See text for more details on message format (or RFC 1035)

- Messages may be:
  - Query (“QR” bit in header is 0)
  - Response (“QR" bit in header is 1)
- Queries may *theoretically* be of several different types
  - **IQUERY** obsoleted in 2002 (RFC 3425)
    - "has not been generally implemented and has usually been operationally disabled where it has been implemented.”
  - **STATUS** never really defined
    - *Proposed* standard in 2001 (DNS was 18 years old by this time)
  - **QUERY** is used for basically everything

- “RD” bit in header is *recursion desired* — requests “recursive” lookup

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

- Many types!

- Remembering primary goal of DNS (map human-friendly names to IP addrs)… The two types we need for that are:

  - **A** records (address)
  - **NS** records (name server)

- We'll talk about other types later…

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

- Name associated with the record

- For **A** records, this is a hostname of interest, e.g., www.google.com

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

- Value associated with the record

- For **A** records, this is the IPv4 address associated with name

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

- How long (in seconds) the record is valid for

- May omit this going forward

- We'll come back to it later

DNS: Protocol

- The actual data stored in the DNS is held in *resource records* (RRs)
- Essentially a tuple: `(type, name, value, ttl, class)`

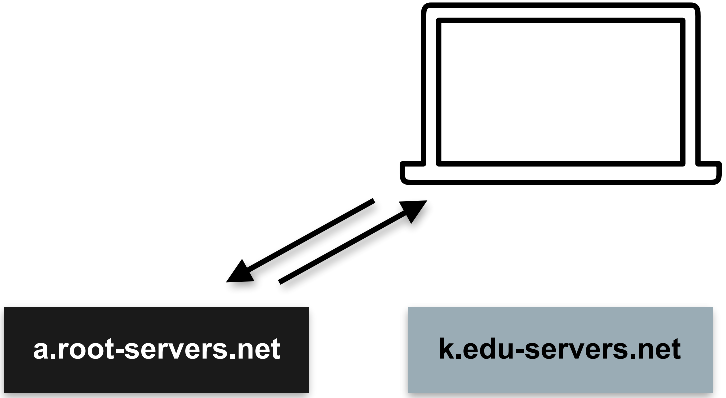
- DNS can be used for network types besides the Internet
    - *class* field specifies what network type

- Don't think this was ever used much (class=Internet almost always)

- We'll ignore it

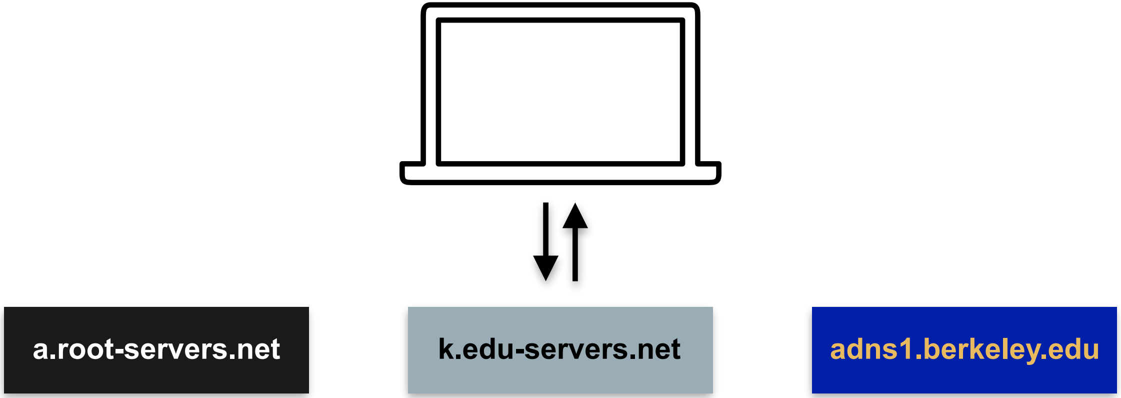
DNS: Protocol example

- Query root server requesting A record for ischool.berkeley.edu
- It sends back (NS, edu, k.edu-servers.net), (NS, edu, l.edu-servers.net), …
  - Not what we asked for, but tells us our next step!
- Also sends (A, k.edu-servers.net, `192.52.178.30`), …
  - "Additional" record(s)
  - It's probably what we would have asked for next!



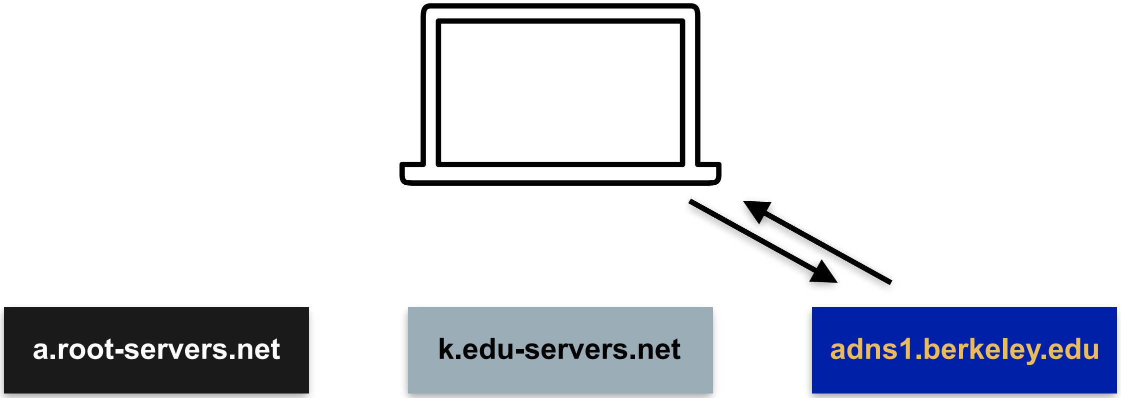
DNS: Protocol example

- Query k.edu-servers.net requesting A record for ischool.berkeley.edu
- It sends back (NS, berkeley.edu, adns1.berkeley.edu), …
- Also sends (A, adns1.berkeley.edu, `128.32.136.3`), …



DNS: Protocol example

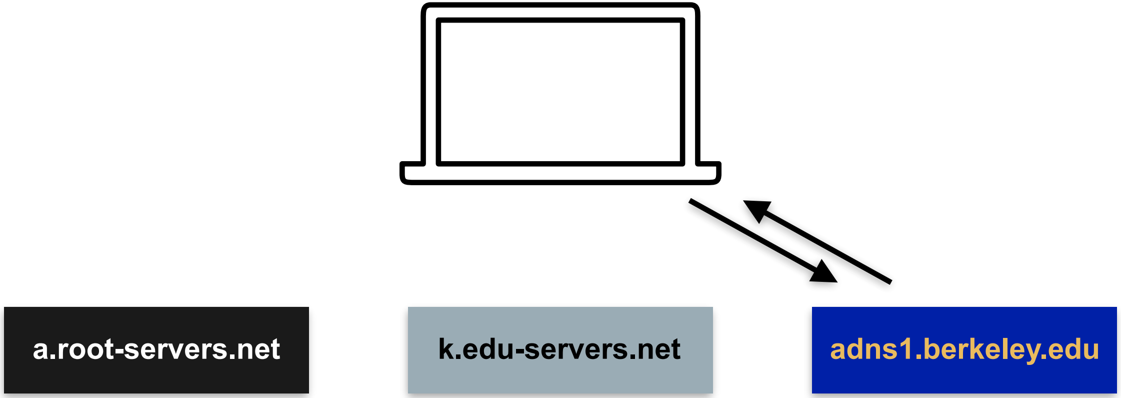
- Query adns1.berkeley.edu requesting A record for ischool.berkeley.edu
- It sends back (A, ischool.berkeley.edu, `128.32.78.26`)
    - That's what we wanted!



DNS: Protocol example

- Query adns1.berkeley.edu requesting A record for ischool.berkeley.edu
- It sends back (A, ischool.berkeley.edu, `128.32.78.26`, 10800)
    - That's what we wanted!

Can keep using this IP address for 10800 seconds (3 hours)



Moving on…

DNS: How is a domain name created?

- Example: you just created company Example Industries
- You get a block of IP addresses from your ISP
- e.g., 192.0.2.0/25

- Register example.com with *registrar* (e.g., GoDaddy)
  - Probably less than $15/year
- Run two authoritative name servers for your domain (or have someone run them for you)
- Give your name server addresses to your registrar
- Registrar inserts pairs of records for them into TLD name servers, e.g.:
  - (NS, example.com, ns1.example.com)
  - (A, ns1.example.com, 192.0.2.6)

- Store resource records in your servers!
  - e.g., type A record for www.example.com - (A, www.example.com, 192.0.2.1)
  - Costs you basically nothing to create any subdomains you want

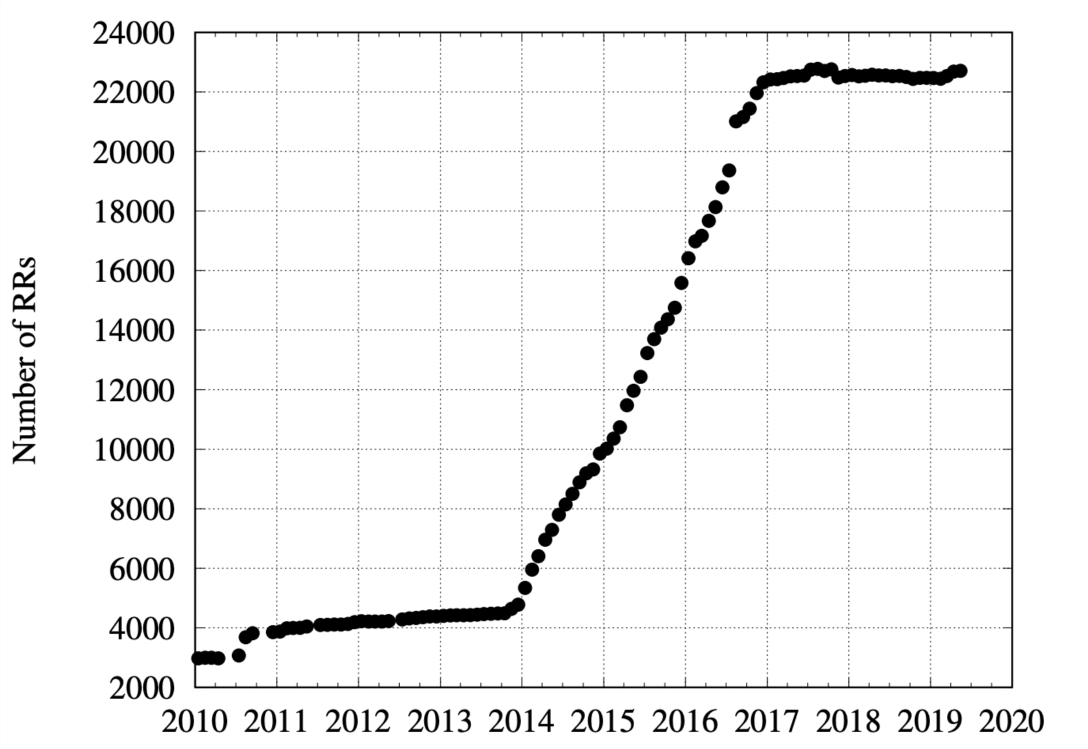
DNS: How is a domain name created?

- What if I want my own top level domain?
  - I want to be murphy@awesome.cs168 !

- Talk to ICANN…
  - Get your own for the low, low price of about $185,000?

  - (If we all chip in, it's only about $370 per person.)
  - (Just saying.)

Number of records at root over time

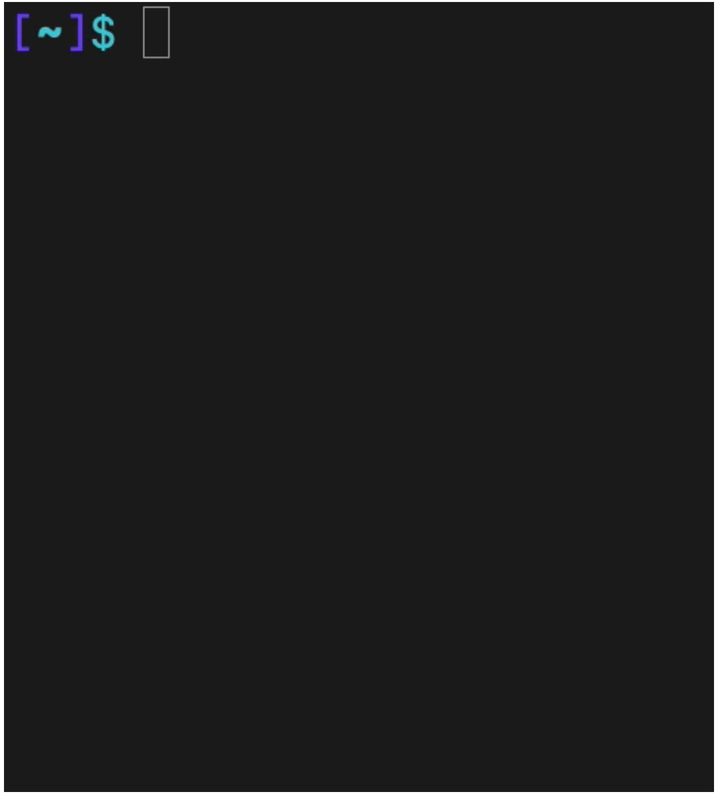


[Data from Allman 2019]

DNS: Beyond the Basics

More DNS: Multiple A records

- There might be more than one A record with the same name!
- Server returns multiple A records
- Shuffles the order
- Allows coarse-grained load balancing
  - .. different users look up yahoo.com
  - .. get different IP addresses
  - .. contact different servers
- Allows simple resiliency
  - .. if first one doesn't work, try next



More DNS: IPv6

- Everything we've looked at so far used IPv4 addresses
- Want IPv6?
  - Ask for an **AAAA** record

```
[~]$ dig +short www.google.com A
172.217.0.36

[~]$ dig +short www.google.com AAAA
2607:f8b0:4005:808::2004
```

More DNS: Reverse lookups

- What if I have an address, e.g., `138.110.1.200` ?
  - What's its hostname?
- **PTR** record
  - Value is an associated hostname
  - Name is:
    - Dot-quad IP address ***listed backwards***
      - `138.110.1.200` → `200.1.110.138`
    - Followed by `.in-addr.arpa`

Similar mechanism for IPv6 using **ip6.arpa**

```
[~]$ dig +short 200.1.110.138.in-addr.arpa PTR
ns.mtholyoke.edu.
```

More DNS: Name aliasing

- CNAME record
  - “Canonical name”
  - Allows you to define an alias for another name

```
[~]$ dig www.berkeley.edu | clean | head -1
www.berkeley.edu. 185 IN  CNAME   www-production-1113102805.us-west-2.elb.amazonaws.com.
```

- www.berkeley.edu name translates to an amazonws.com name
  - (Because Berkeley’s main website is hosted by Amazon)
- Next step would be to look up the A record for the amazonws.com name
  - (Actually, server included it — the “**`head -1`**” hid it)
- Similar DNAME record maps a whole subtree:
  - WHATEVER.foo.com → WHATEVER.bar.com

More DNS: Email

- Send an email to murphy@berkeley.edu and I get it… at google.com?
- How? Why?
  - berkeley.edu is hosted by Amazon, not Google!
- Even in past, mail server was often separate machine, e.g., mail.berkeley.edu
  - Nobody wants to address messages to murphy@mail.berkeley.edu!
- Email servers look up **MX** record (*mail exchanger*) of recipient domain
  - This tells the mail server(s) to use for mail to that domain

```
[~]$ dig berkeley.edu MX | clean
berkeley.edu.     219 IN  MX  1 aspmx.l.google.com.
berkeley.edu.     219 IN  MX  5 alt2.aspmx.l.google.com.
berkeley.edu.     219 IN  MX  5 alt1.aspmx.l.google.com.
berkeley.edu.     219 IN  MX  10 alt3.aspmx.l.google.com.
berkeley.edu.     219 IN  MX  10 alt4.aspmx.l.google.com.
```

More DNS: TXT records

- **TXT** records were originally meant for human-readable information

- These days, often used for things like *site verification*

```
[~]$ dig +short berkeley.edu TXT | grep veri
"adobe-idp-site-verification=a113c870-3c49-4b4a-b3a4-31e1cf1860cb"
"ZOOM_verify_RirbP7N1QWC3Zzm02oL4Cw"
"google-site-verification=fL93jj-VPnl_5wdFDh26YshzKVPraWAurHaBCu-k-Xw"
"google-site-verification=loQrJWyMsMB249uINb-AsRGTWVoLdTc44Td3aMGn-NE"
```

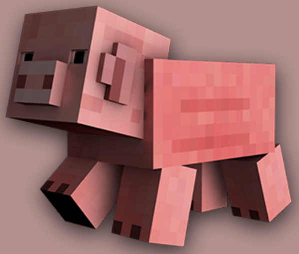
- Adobe, Zoom, Google, Facebook, etc. give you a magic value
- You put it in TXT record on your domain — proves you have control over domain
- Unlocks capabilities on other site
  - Google will show how often your site shows up in results
  - Facebook lets you edit how shared links to your site appear

More DNS: SRV records

- MX was this special-purpose redirection for email
- What about other services?
    - Do they all need their own special record types?
    - Seems silly
    - **SRV** record solves similar problem for arbitrary services

- Record na ... .here
    - Recor
        - “t
        - P
        - (S

See Minecraft case study video
if you want to see how this is used
in a real-world scenario.



- Easy to add more services — just create more SRV records

More DNS: Simple Indirection

- Remember, in an ideal world, your IP address is topologically meaningful!
  - e.g., it's a sub-allocation of your provider's address range

- What if you have a popular website, www.example.com...
  - The server is at `203.0.113.4`…
  - And you switch providers…
  - And new provider gives you `198.51.100.88` ?

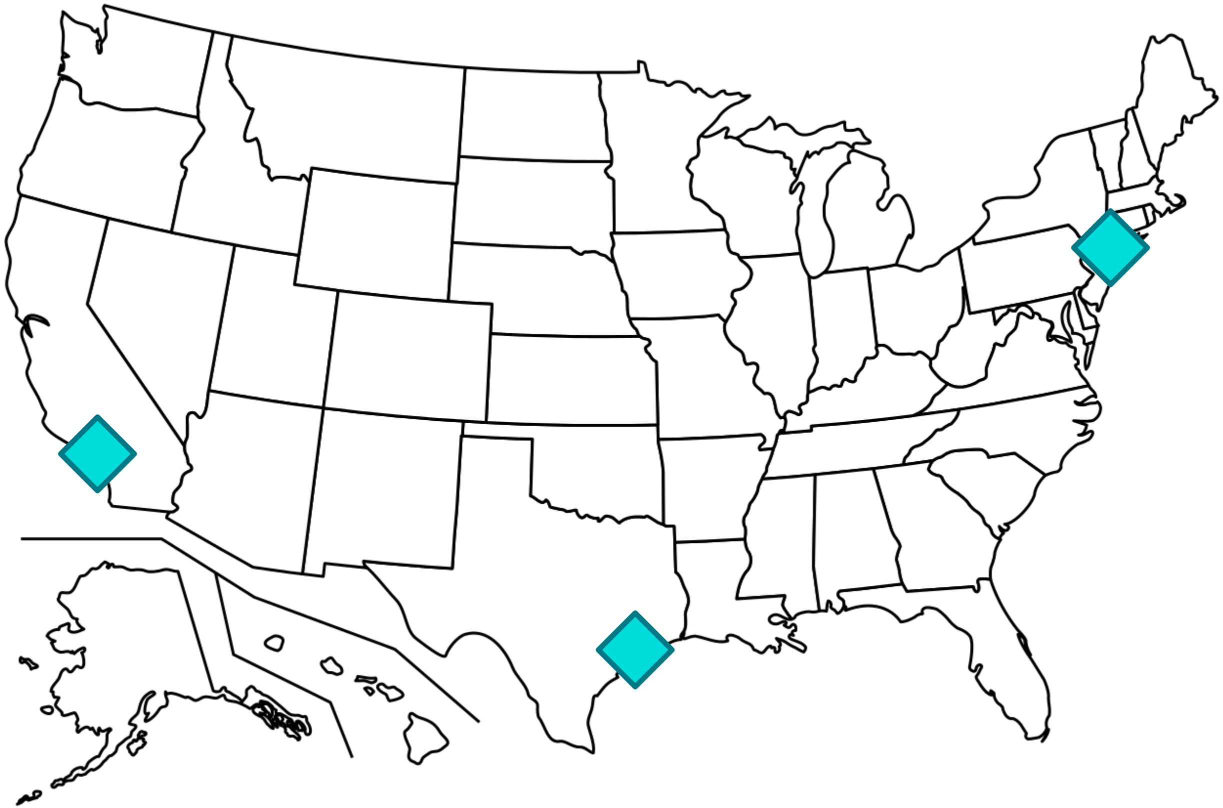
- .. just update www.example.com's A record to `198.51.100.88`
  - .. few people likely to notice

More DNS: Intelligent indirection

- You stream video from three servers across the United States
- Three different IP addresses

- Smart DNS server looks at IP address of client…
- Does GeoIP lookup…
- Selects closest server!

- Saves money/latency



More DNS: Summary

- We’ve looked at a lot of things DNS can do!

- Coarse server load spreading and resiliency (via multiple A records)
- Name-to-IPv6-Address mapping (via AAAA records)
- IP-Address-to-Name (reverse) mapping (via PTR records)
- Alias names (via CNAME record)
- Nice email addresses (via MX records)
- Site verification (via TXT records)
- General name-to-service mapping (via SRV records)

- DNS as an indirection layer

- .. and there are many more record types and DNS tricks!

Attributions

Minecraft PNG, Creative Commons 4.0 BY-NC

http://pngimg.com/download/59250

Blank US map borders.svg, Public Domain

https://commons.wikimedia.org/wiki/File:Blank_US_map_borders.svg