CS 43: Computer Networks

DNS and Email

October 01, 2024

SWARTHMORE COLLEGE

Reading Quiz

Julia Evans
@b0rk

everything inside a DNS packet

I literally mean everything, I copied this verbatim from a real DNS request using Wireshark.
(DNS packets are binary but we're showing a human-readable representation here)

Let's look at the actual data being sent during a DNS query:

request

```
Query ID:  0x05a8
Flags:     0x1000
Questions:             1
Answer records:        0
Authority records:     0
Additional records:    0
Question:
   Name:   example.com
   Type:   A
   Class:  IN
```

- Query ID: randomly generated
- Flags: these flags just mean "this is a request"
- A is for IPv4 Address. other types: MX, CNAME, AAAA, etc
- IN stands for "INternet"

response

```
Query ID:  0x05a8
Flags:     0x8580
Questions:             1
Answer records:        1
Authority records:     0
Additional records:    0
Question:
   Name:   example.com
   Type:   A
   Class:  IN
Answer records:
   Name:     example.com
   Type:     A
   Class:    IN
   TTL:      86400
   Content:  93.184.216.34
Authority records:
   (empty)
Additional records:
   (empty)
```

- Query ID: matches request ID
- Flags: the response code is encoded in the last 4 bits of these flags. The 3 main response codes are:
  - → NOERROR (success!)
  - → NXDOMAIN (doesn't exist!)
  - → SERVFAIL (error!)
- Question (Name, Type, Class): copied from request
- Answer Name: domain names aren't case sensitive
- Content: the IP we asked for
- Authority records / Additional records: page 12 ("NS records") talks more about these 2 sections

Where we are

Application: the application (e.g., HTTP, DNS)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

Today

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

Goals of DNS

A wide-area distributed database

Possibly biggest such database in the world!

Goals

- Scalability; decentralized maintenance
- Robustness
- Global scope
- Names mean the same thing everywhere
- Distributed updates/queries
- Good performance

DNS: Application Layer Protocol

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

Uses of DNS

Hostname to IP address translation

- Reverse lookup: IP address to hostname translation

Host name aliasing: other DNS names for a host

- Alias hostnames point to canonical hostname

Email: look up domain’s mail server by domain name

Different DNS Mappings

- **1-1 mapping between domain name and IP addr**
  - www.cs.cornell.edu maps to 132.236.207.20
- **Multiple domain names maps to the same IP addr**
  - eecs.mit.edu and cs.mit.edu both map to 18.62.1.6
- **Single domain name maps to multiple IP addrs**
  - aol.com and www.aol.com map to multiple IP addrs
- **Some valid domain names don’t map to any IP addr**
  - cmcl.cs.cmu.edu

DNS Services

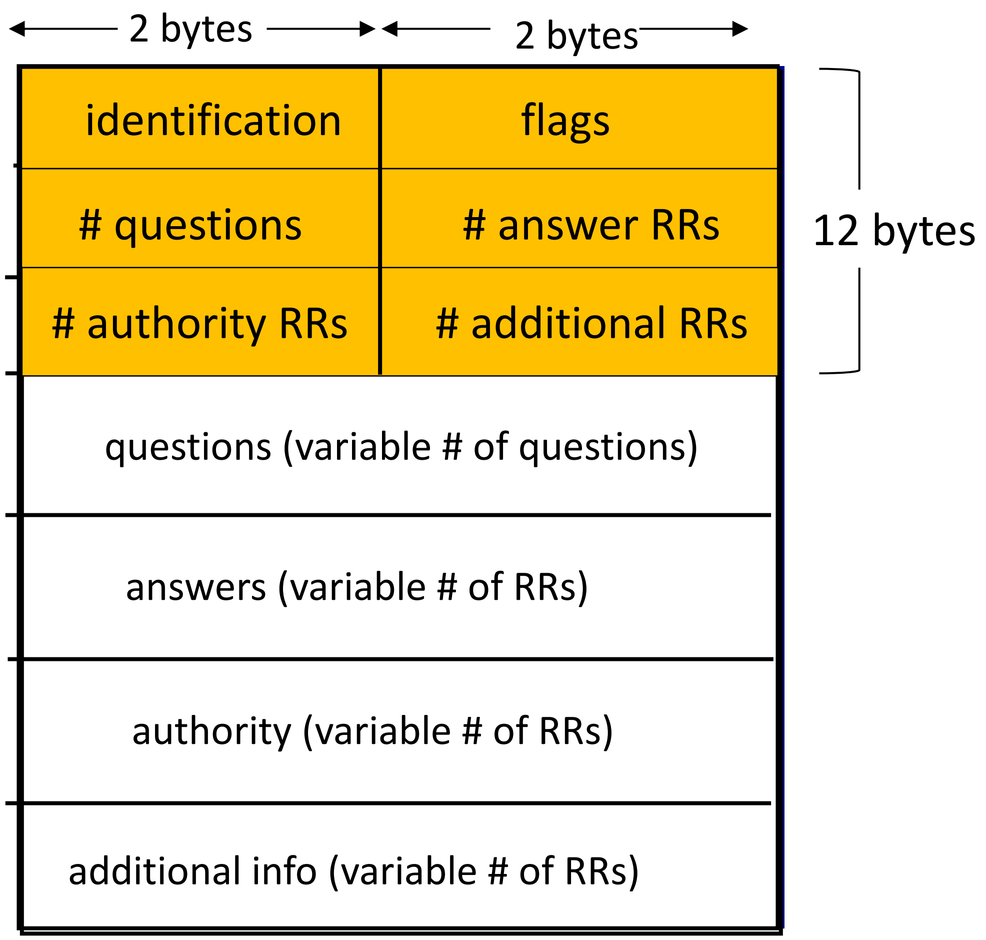
- DNS is an application-layer protocol. E2E design!
- It provides:
  - Hostname to IP address translation
  - Host aliasing (canonical and alias names)
  - Mail server aliasing
  - Load distribution (one name may resolve to multiple IP addresses)
  - Lots of other stuff that you might use a directory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

DNS protocol, messages

- query and reply messages, both with same message format

Message header

- identification: 16 bit id for query, reply to query uses same id.
- flags: recursion, query/reply
- # Resource Records to follow



DNS protocol, messages

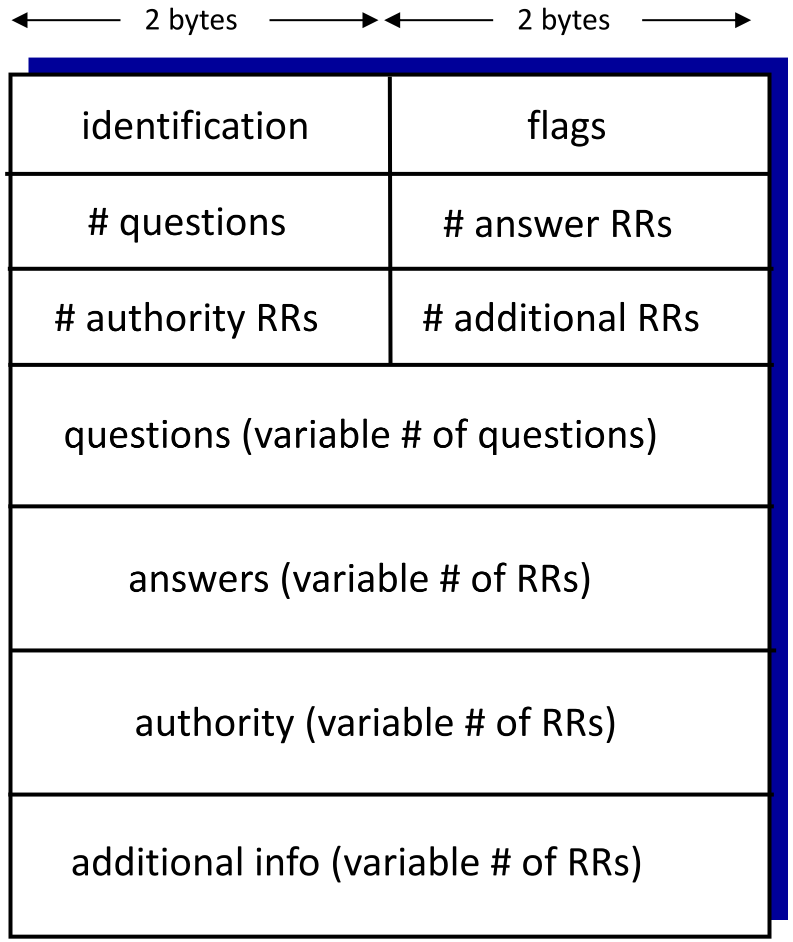
- query and reply messages, both with same message format!

Binary Protocol!

- Delimiters: pre-defined lengths/field
- Names: <len><name>

Sent via UDP (User Datagram Protocol)

- No connection established
- Not reliable!



DNS Records

DNS*:* distributed DB storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A

- **name** is hostname
- **value** is IP address

type=NS

- **name** is domain (e.g., foo.com)
- **value** is hostname of authoritative name server for this domain

type=CNAME

- **name** is alias name for some "canonical" (the real) name
- **www.ibm.com** is really
- servereast.backup2.ibm.com
- **value** is canonical name

type=MX

- **value** is name of mailserver associated with name

DNS Types

RR format: `(name, value, type, ttl)`

- Type = A / AAAA
  - Name = domain name
  - Value = IP address
  - A is IPv4, AAAA is IPv6

Query
Name: cs.swarthmore.edu
Type: A

Resp.
Name: cs.swarthmore.edu
Value: 130.58.68.9

- Type = NS
  - Name = partial domain
  - Value = name of DNS server for this domain
  - "Go send your query to this other server"

Query
Name: cs.swarthmore.edu
Type: NS

Resp.
Name: cs.swarthmore.edu
Value: 130.58.68.9

DNS Types, Continued

RR format: `(name, value, type, ttl)`

- Type = CNAME
  - Name = hostname
  - Value = canonical hostname
  - Useful for aliasing
  - CDNs use this

Query	Name: foo.mysite.com Type: CNAME
Resp.	Name: foo.mysite.com Value: bar.mysite.com

- Type = MX
  - Name = domain in email address
  - Value = canonical name of mail server

Query	Name: cs.umass.edu Type: MX
Resp.	Name: cs.umass.edu Value: barramail.cs.umass.edu.

Domain Name System (DNS)

- Distributed administrative control
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translations
  - Local DNS servers
  - Resolver software

Domain Name System (DNS)

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

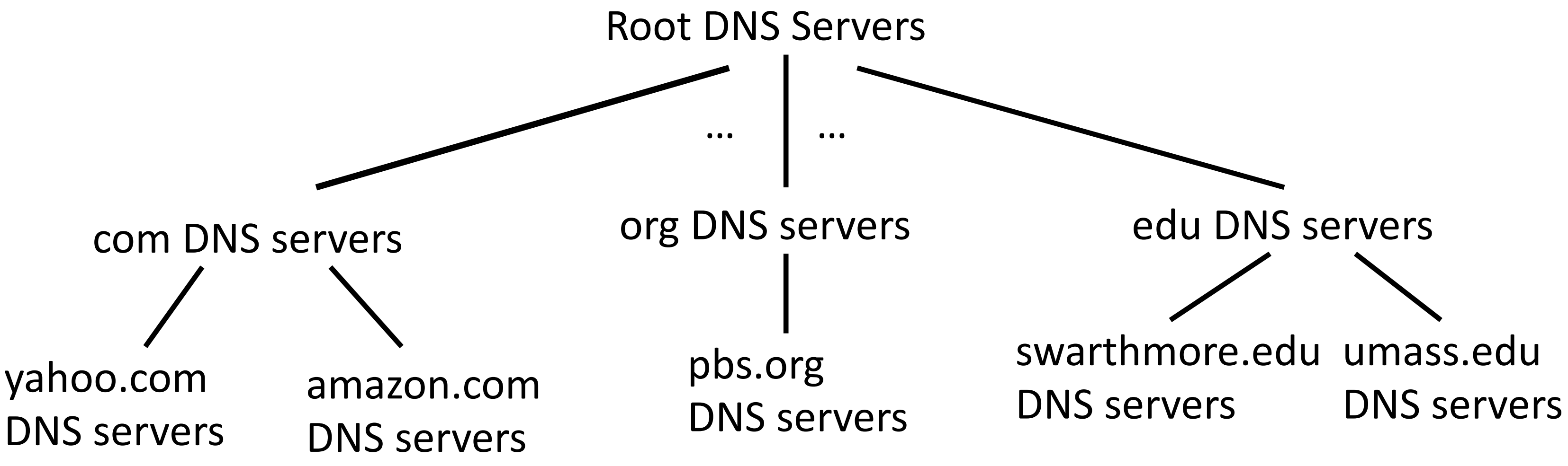
Domain Name System (DNS)

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

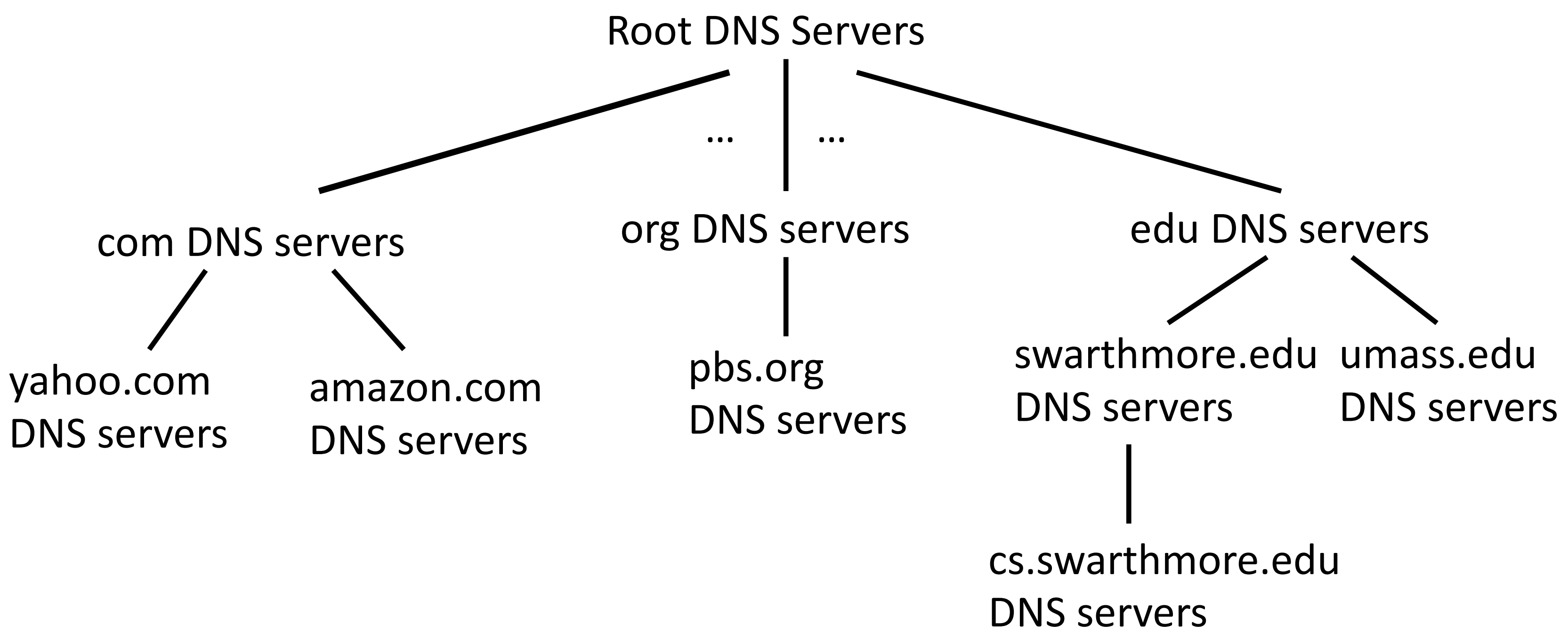
Domain Name System (DNS)

- Performing the translations
  - Local DNS servers
  - Resolver software

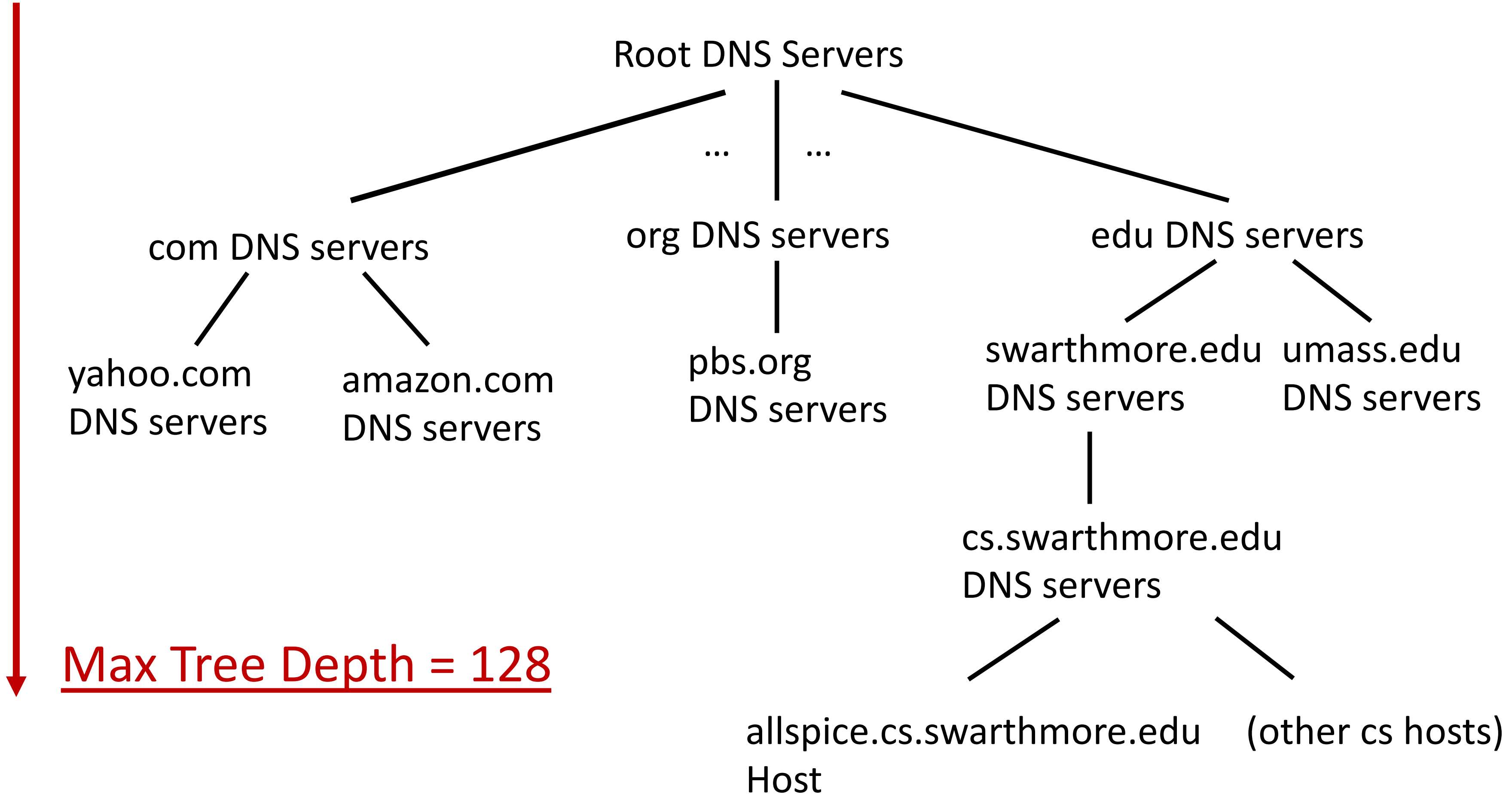
DNS: a distributed, hierarchical database



DNS: a distributed, hierarchical database



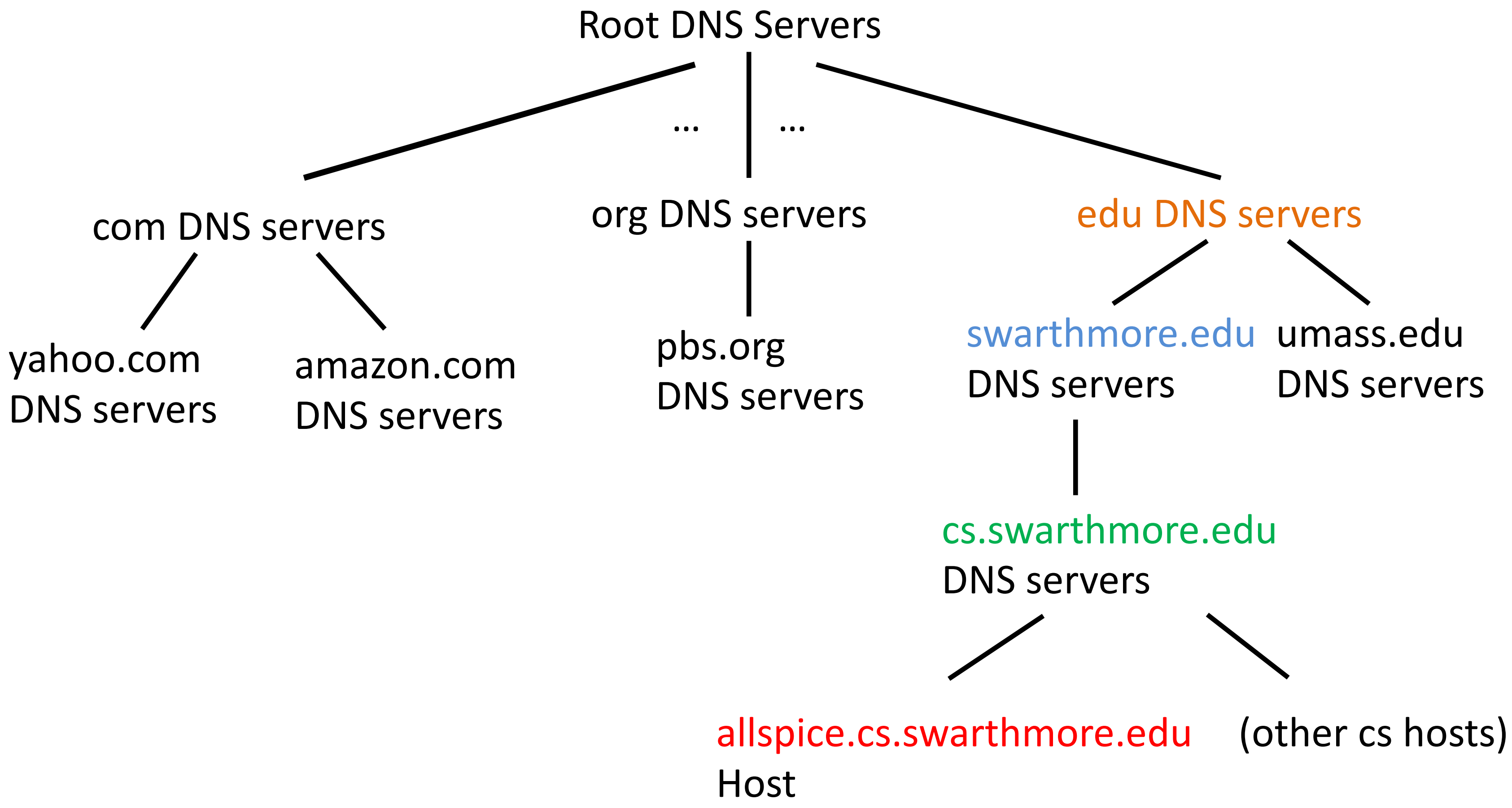
DNS: a distributed, hierarchical database



allspice.cs.swarthmore.edu.

Nameless root,
Usually implied.

DNS: a distributed, hierarchical database



- allspice.cs.swarthmore.edu.

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

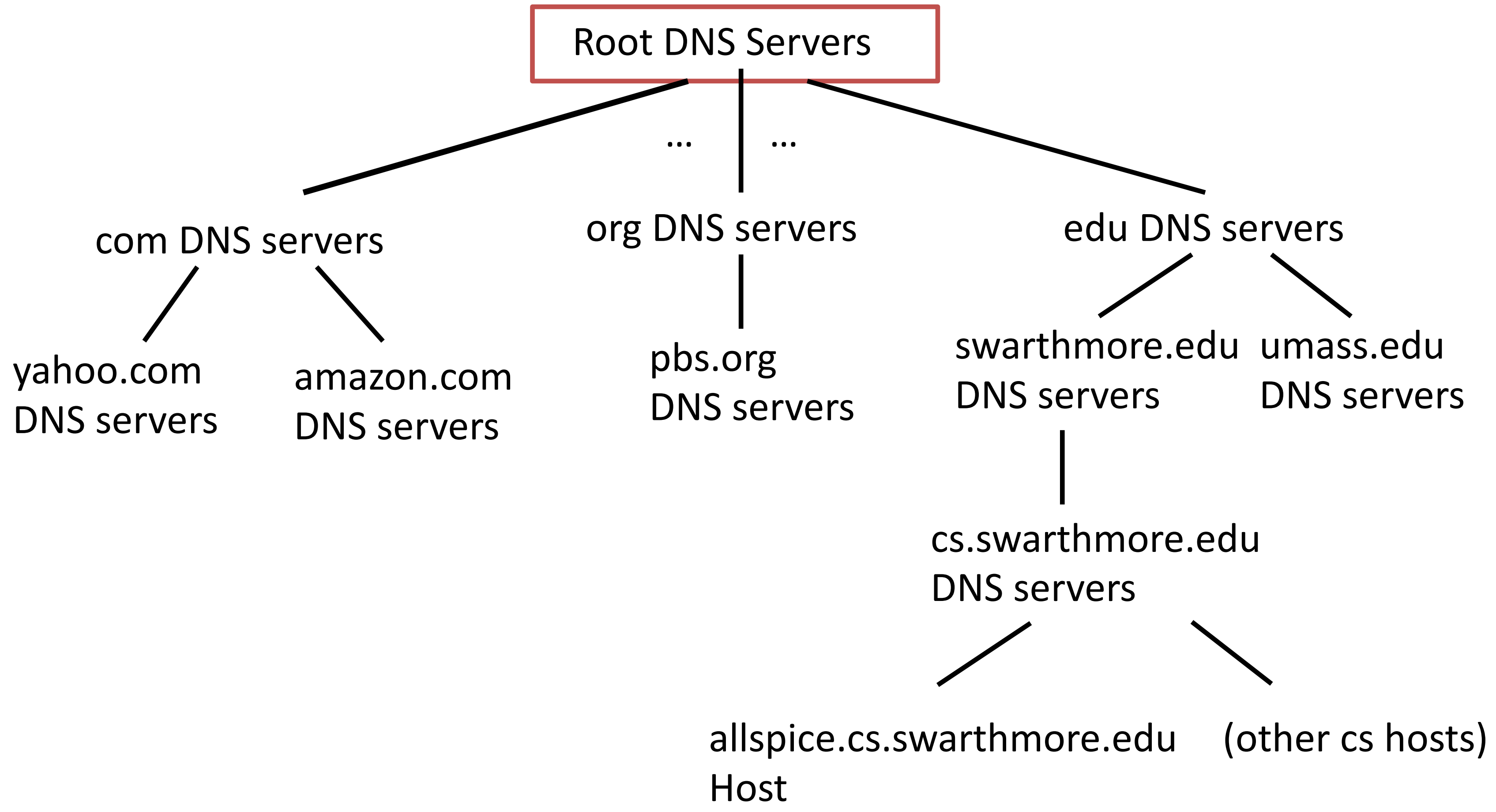
A. It divides up responsibility among parties.

B. It improves performance of the system.

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

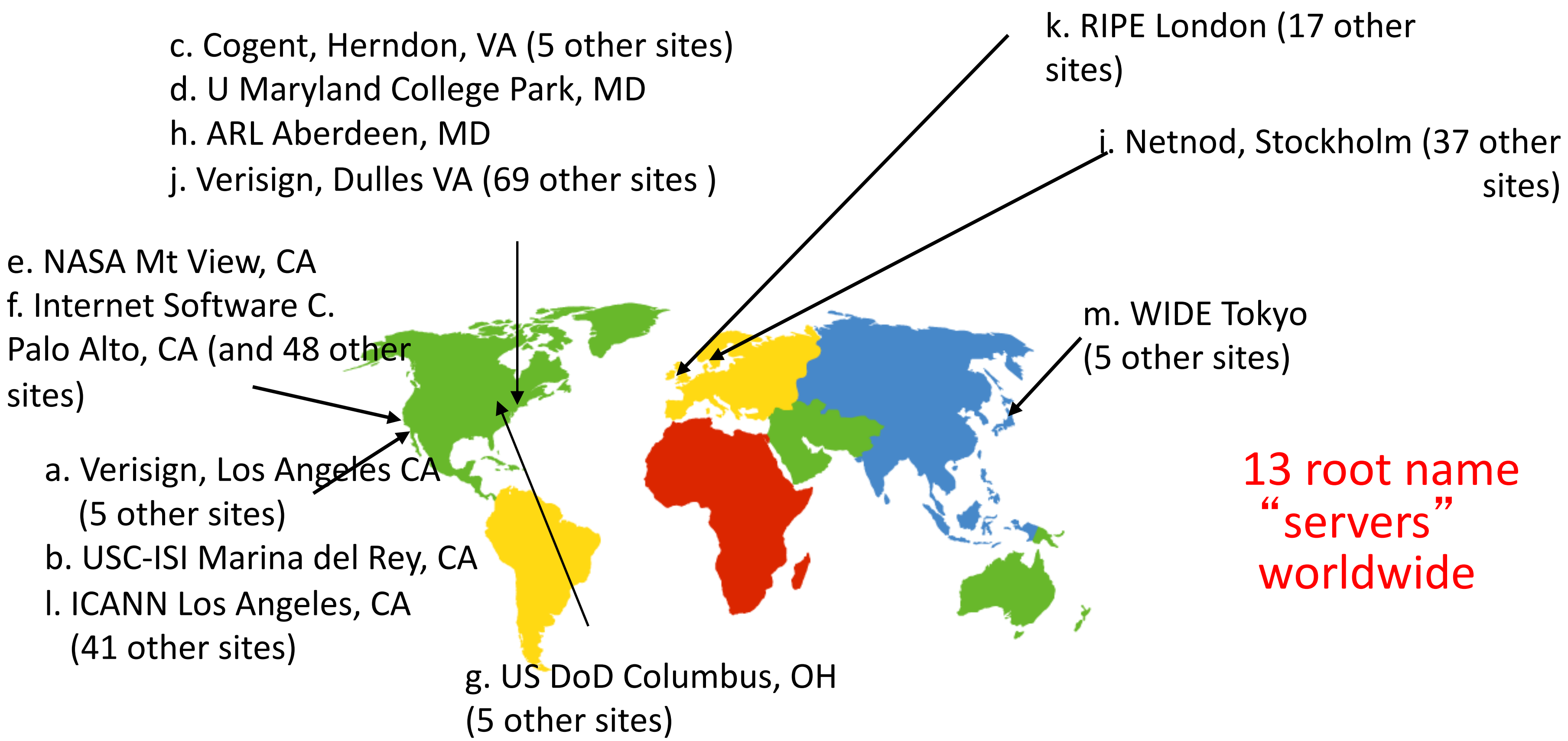
D. Some other reason.

DNS: a distributed, hierarchical database



DNS: Root Name Servers

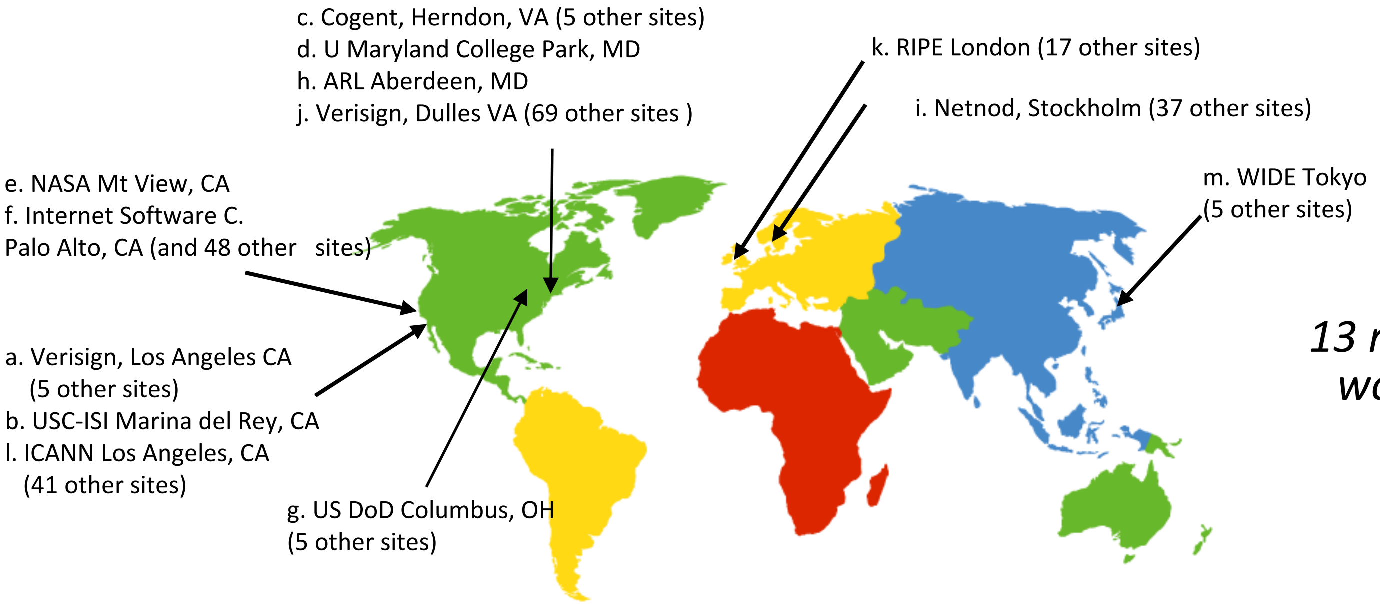
- Root name server:
  - Knows how to find top-level domains (.com, .edu, .gov, etc.)
  - How often does the location of a TLD change?



13 root name "servers" worldwide

DNS: Root Name Servers

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



13 root name "servers" worldwide

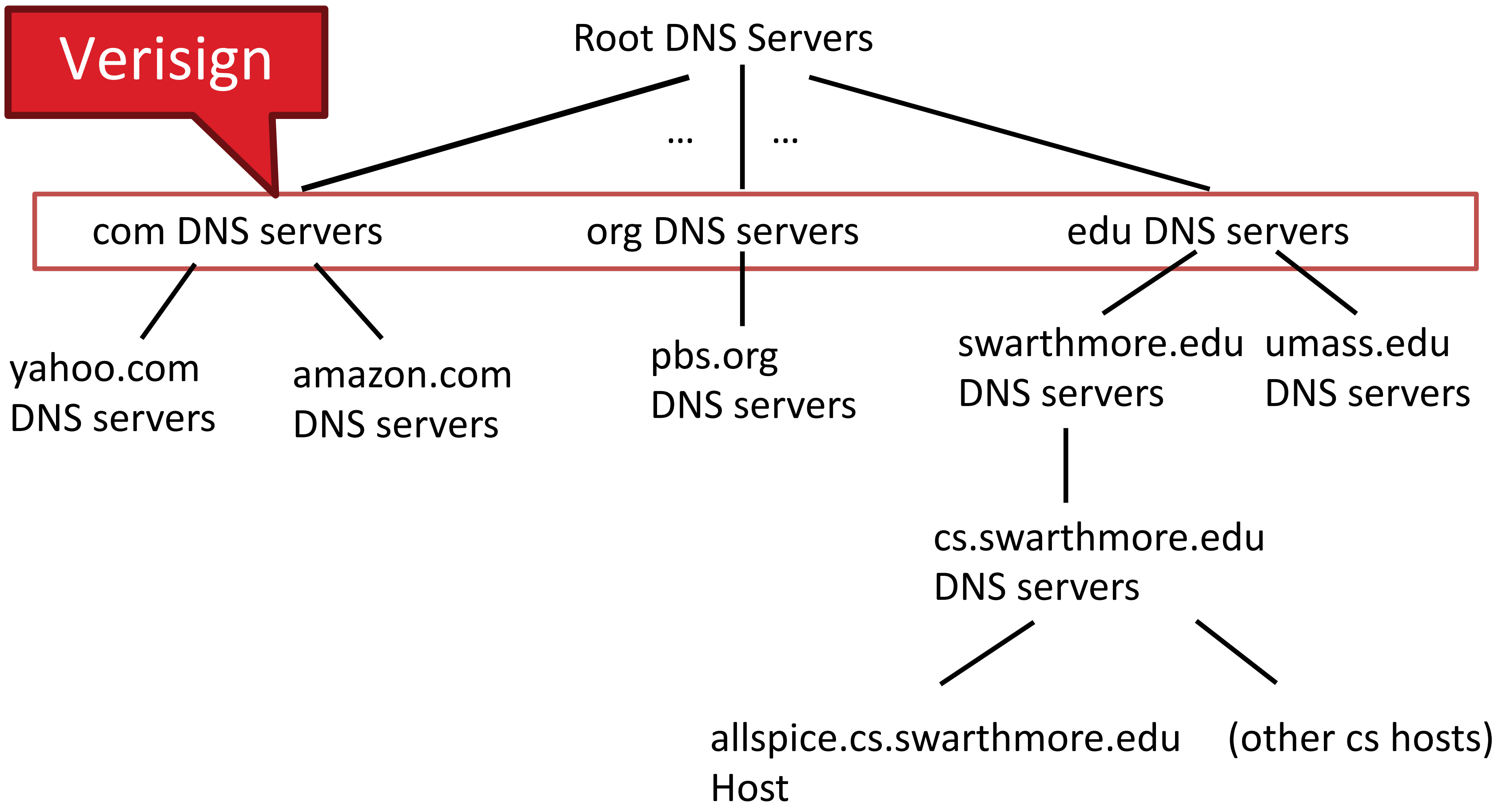
Root Name Servers

- Responsible for the Root Zone File

com.	172800 IN	NS	a.gtld-servers.net.
com.	172800 IN	NS	b.gtld-servers.net.
com.	172800 IN	NS	c.gtld-servers.net.

  - In practice, most systems cache this information
  - Lists the TLDs and who controls them
  - ~272KB in size

Top Level Domain (TLD) servers

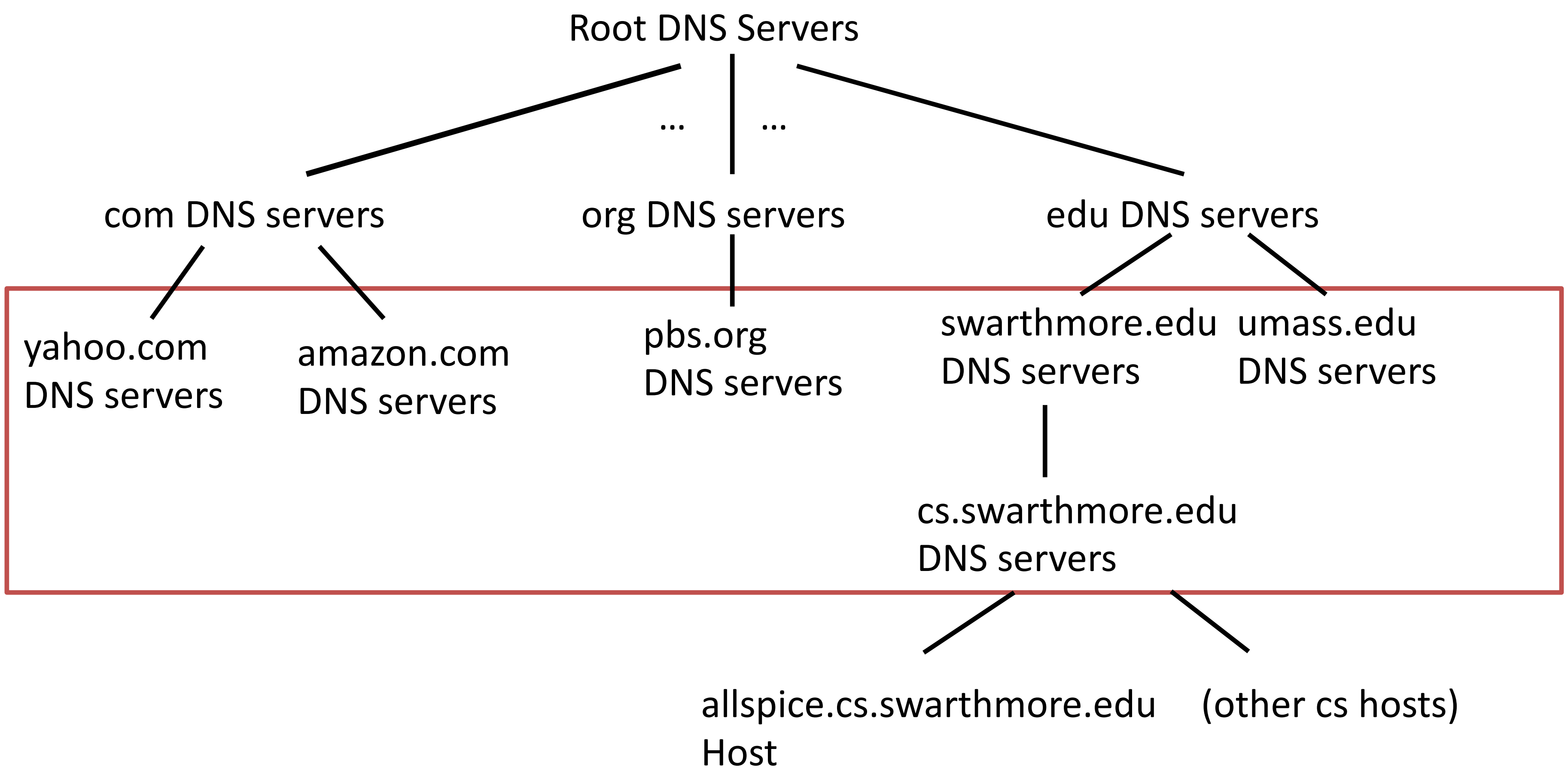


Top Level Domain (TLD) servers

- who maintains the servers?:
  - Verisign: .com, .net
  - Educause: .edu (Verisign backend)
  - local governments or companies

- Responsible for:
  - com, org, net, edu, gov, aero, jobs, museums,
  - all top-level country domains, e.g.: uk, fr, de, ca, jp, etc

Authoritative Servers



Authoritative Servers

Authoritative DNS servers:

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

Resolution Process

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

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

Local DNS Name Server

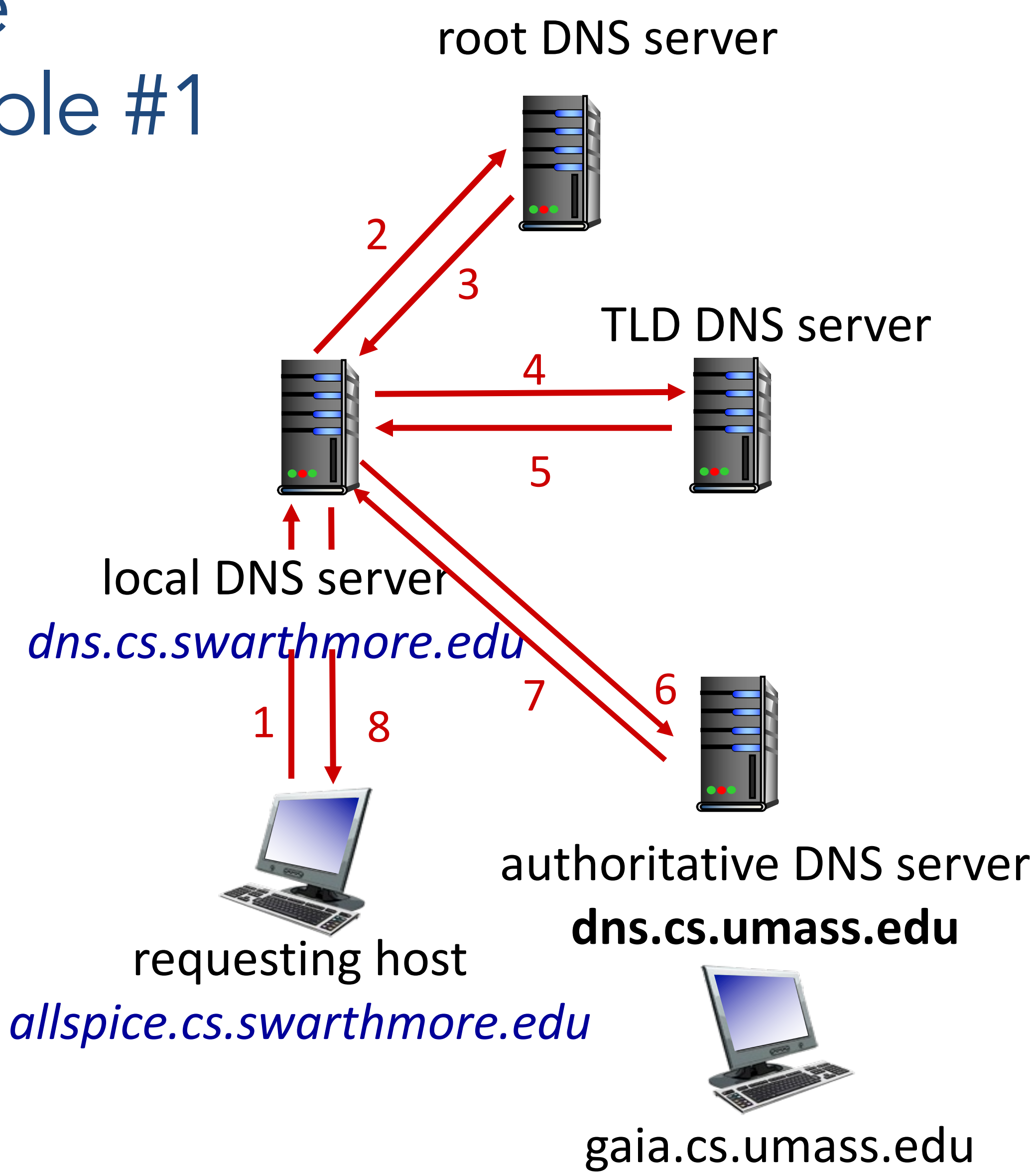
- Each ISP (residential ISP, company, university) has (at least) one
  - also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into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

DNS name resolution example #1

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

iterative query:

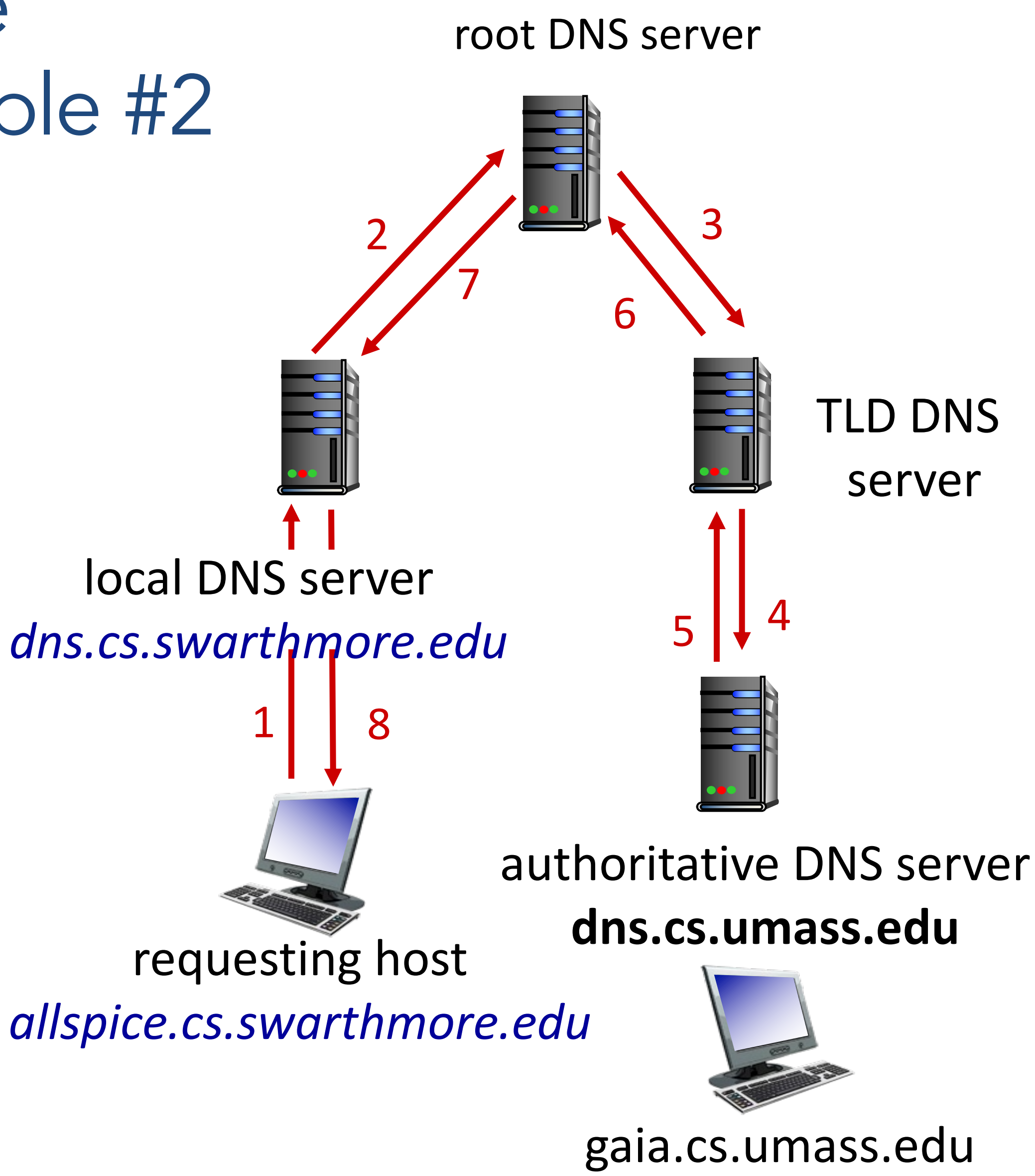
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example #2

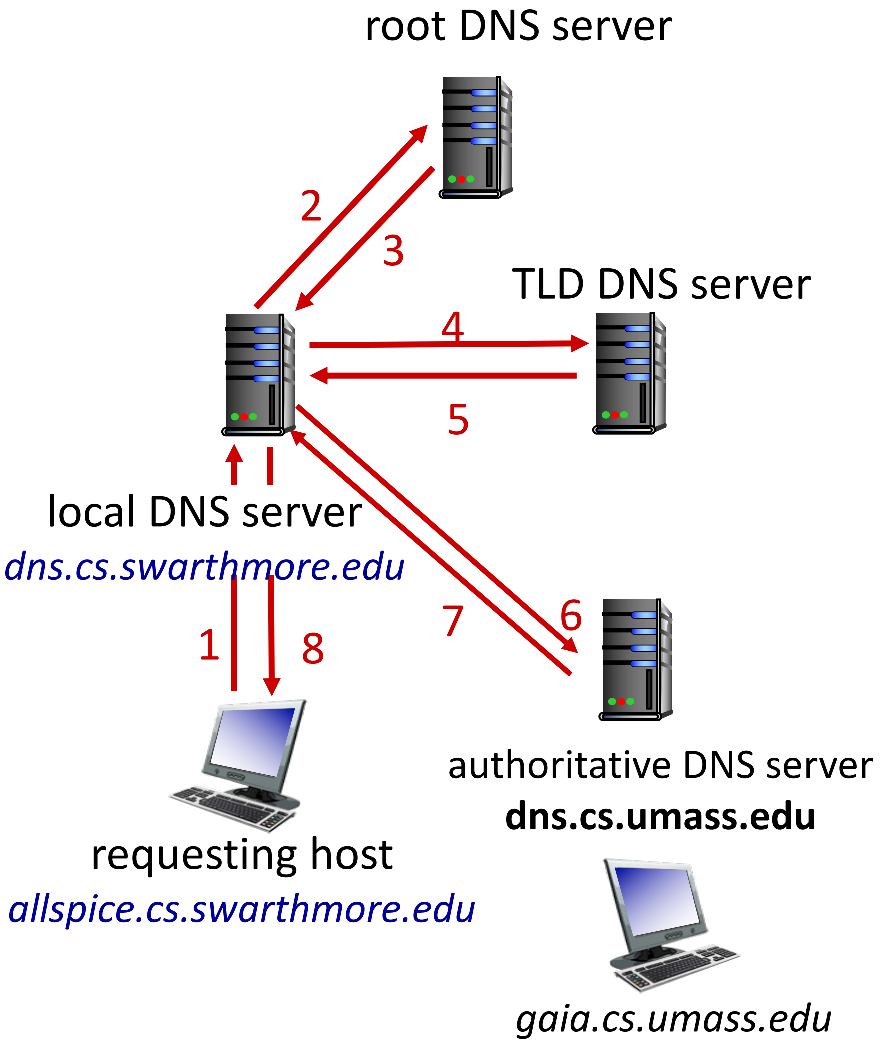
recursive query:

- each server asks the next one, in a chain

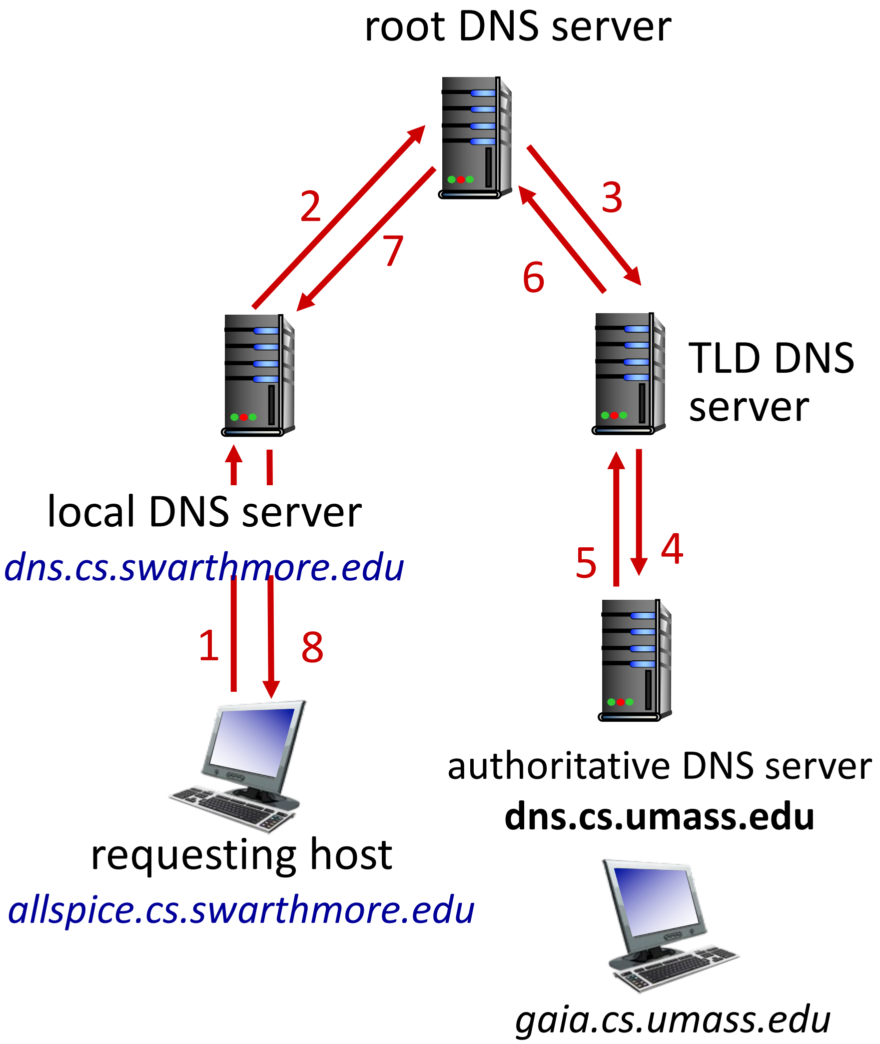


Which would you use? Why?

A. Iterative



B. Recursive



Example: iterative query using dig()

```
dig . ns

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

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

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

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

How many answers
Time to live in seconds
How many additional records?

```
$ dig @a.root-servers.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.freebsd.org.          IN     A

;; AUTHORITY SECTION:
org.                  172800 IN     NS     b0.org.afilias-nst.org.
org.                  172800 IN     NS     d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
b0.org.afilias-nst.org.   172800 IN     A      199.19.54.1
d0.org.afilias-nst.org.   172800 IN     A      199.19.57.1
```

How many answers
Time to live in seconds
How many additional records?

```
$ dig @a.root-servers.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.freebsd.org.          IN     A

;; AUTHORITY SECTION:
org.                172800 IN     NS     b0.org.afilias-nst.org.
org.                172800 IN     NS     d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
b0.org.afilias-nst.org.    172800 IN     A      199.19.54.1
d0.org.afilias-nst.org.    172800 IN     A      199.19.57.1
```

Glue records

How many answers?
How many additional records?

(authoritative for org.)

```
$ dig @199.19.54.1 www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39912
;; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;www.freebsd.org.         IN    A

;; AUTHORITY SECTION:
freebsd.org.        86400  IN    NS    ns1.isc-sns.net.
freebsd.org.        86400  IN    NS    ns2.isc-sns.com.
freebsd.org.        86400  IN    NS    ns3.isc-sns.info.
```

How many answers?
How many additional records?

(authoritative for org.)

```
$ dig @199.19.54.1 www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39912
;; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;www.freebsd.org.          IN     A

;; AUTHORITY SECTION:
freebsd.org.        86400  IN     NS     ns1.isc-sns.net.
freebsd.org.        86400  IN     NS     ns2.isc-sns.com.
freebsd.org.        86400  IN     NS     ns3.isc-sns.info.
```

(authoritative for freebsd.org.)

```
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.freebsd.org.            IN      A

;; ANSWER SECTION:
www.freebsd.org.    3600    IN      A       69.147.83.33

;; AUTHORITY SECTION:
freebsd.org.        3600    IN      NS      ns2.isc-sns.com.
freebsd.org.        3600    IN      NS      ns1.isc-sns.net.
freebsd.org.        3600    IN      NS      ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net.    3600    IN      A       72.52.71.1
ns2.isc-sns.com.    3600    IN      A       38.103.2.1
ns3.isc-sns.info.   3600    IN      A       63.243.194.1
```

How many answers?
How many authoritative records?
How many additional records?

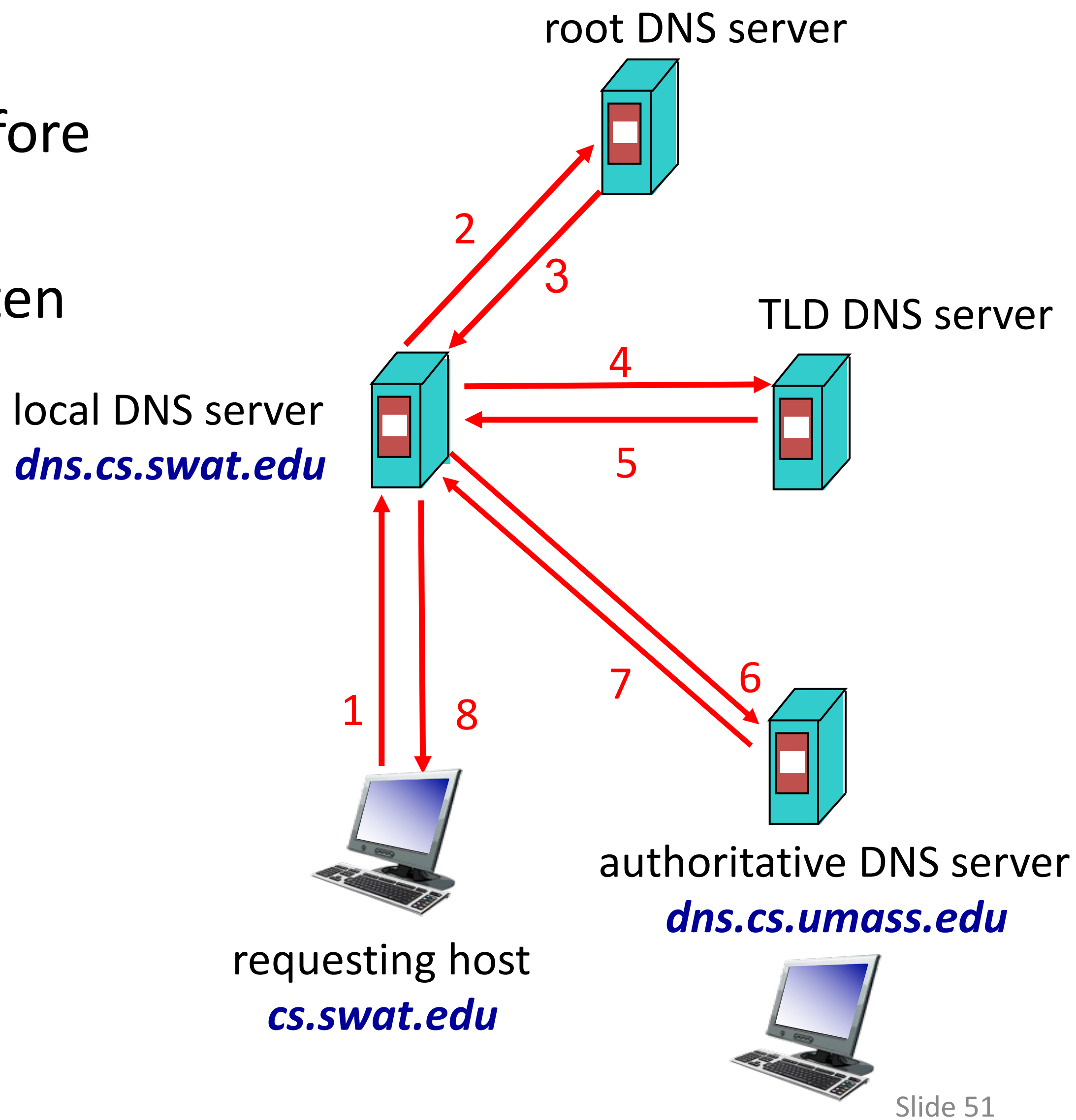
(authoritative for freebsd.org.)

```
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.freebsd.org.          IN     A

;; ANSWER SECTION:
www.freebsd.org.    3600   IN     A      69.147.83.33

;; AUTHORITY SECTION:
freebsd.org.        3600   IN     NS     ns2.isc-sns.com.
freebsd.org.        3600   IN     NS     ns1.isc-sns.net.
freebsd.org.        3600   IN     NS     ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net.    3600   IN     A      72.52.71.1
ns2.isc-sns.com.    3600   IN     A      38.103.2.1
ns3.isc-sns.info.   3600   IN     A      63.243.194.1
```

How many answers?
How many authoritative records?
How many additional records?

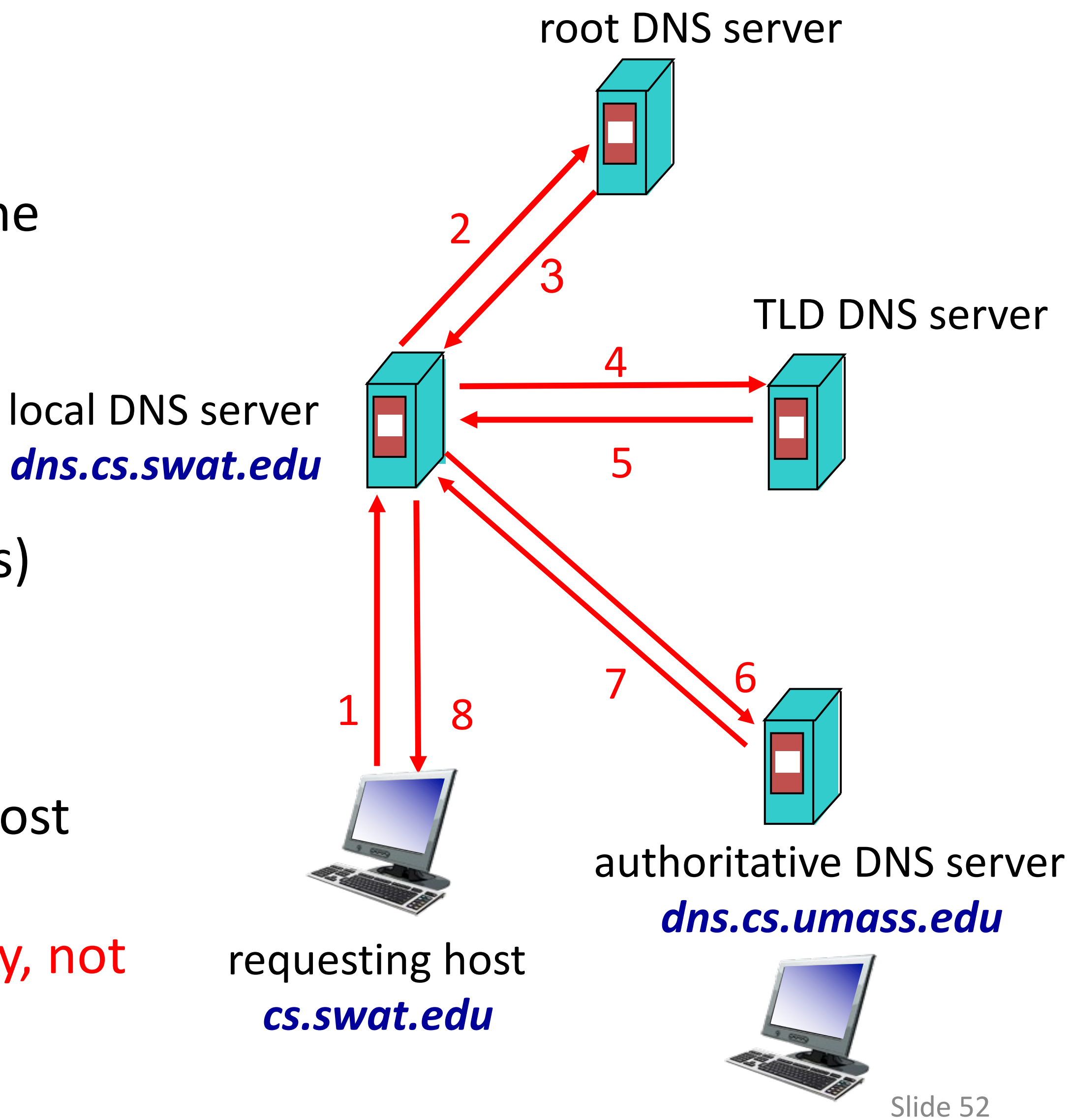
DNS Caching

- Why cache?
  - apprx. 1 sec latency before starting a download
  - Popular sites visited often
- Where to cache?
  - Local DNS server
  - Browser



DNS Caching

- When to cache?
  - learn a mapping? cache!
  - any name server can cache
- For how long?
  - until Time To Live (expires)
- What to cache?
  - TLD servers cached – almost never change
  - Root name servers usually, not visited legitimately



The TTL value should be…

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

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

C. Something else

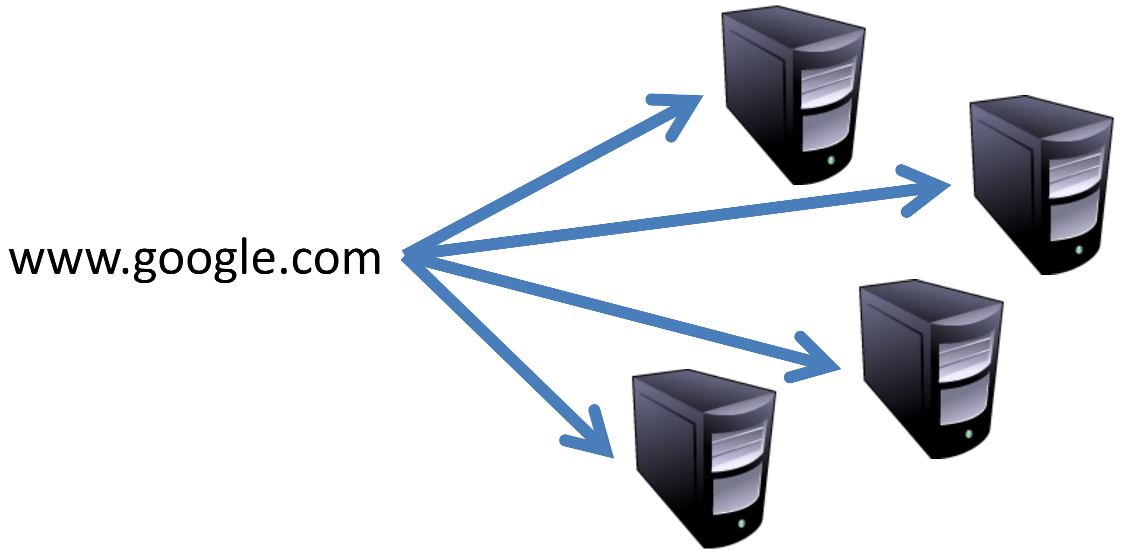
DNS as Indirection Service

- DNS gives us very powerful capabilities
  - Not only easier for humans to reference machines!
- Changing the IPs of machines becomes trivial
  - e.g. you want to move your web server to a new host
  - Just change the DNS record!

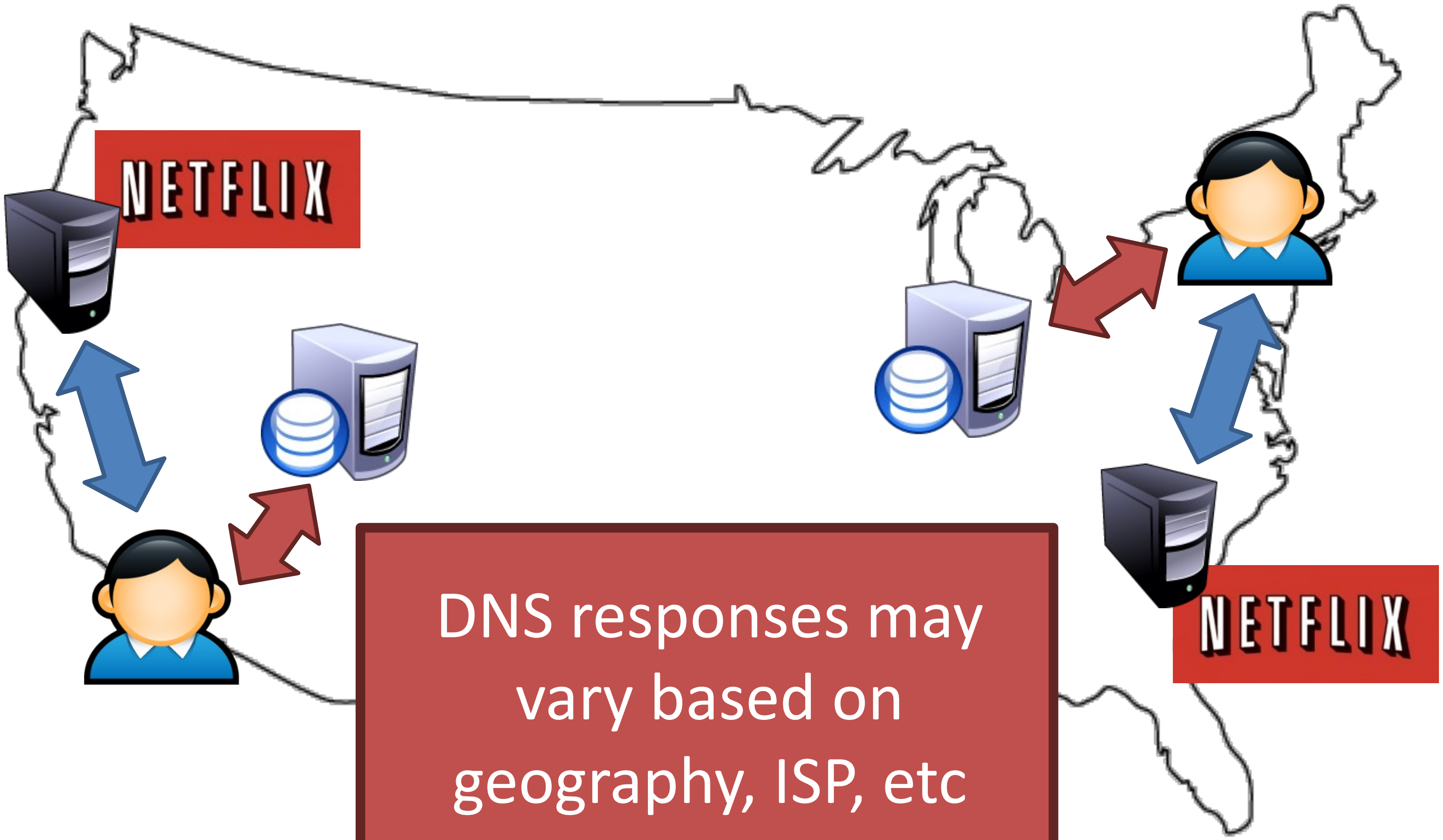
Aliasing and Load Balancing

- One machine can have many aliases

- One domain can map to multiple machines

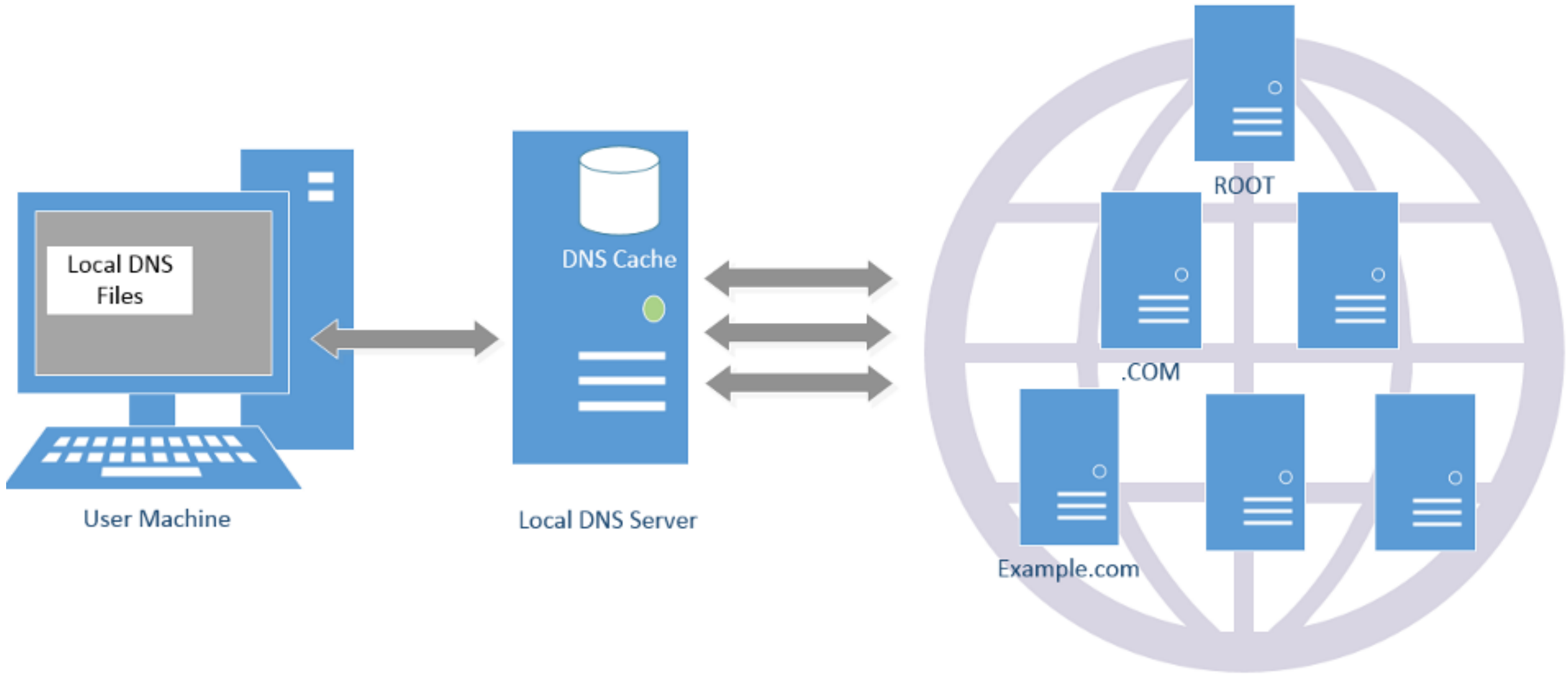


Content Delivery Networks



DNS responses may vary based on geography, ISP, etc

DNS Query Process and Cache



Caching

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

Caching

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

The TTL value should be…

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

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

C. Something else

Inserting (or changing) records

Example: new startup “Network Utopia”

- Step 1: Register networkuptopia.com at DNS registrar
  - provide names, IP addresses of authoritative name server (primary and secondary)

Inserting (or changing) records

Example: new startup "Network Utopia"

- Step 2: Registrar inserts two RRs into .com TLD server
  - (networkutopia.com, dns1.networkutopia.com, NS)
  - (dns1.networkutopia.com, 212.212.212.1, A)

Inserting (or changing) records

Example: new startup "Network Utopia"

- Step 3: Set up authoritative server at that name/address
  - Create records for the services:

Inserting (or changing) records

Example: new startup "Network Utopia"

- Step 3: Set up authoritative server at that name/address
  - Create records for the services:
    - type A record for www.networkuptopia.com
    - type MX record for @networkutopia.com email

Inserting (or changing) records

- Example: new startup "Network Utopia"
- Register networkuptopia.com at DNS registrar
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server
    - (networkutopia.com, dns1.networkutopia.com, NS)
    - (dns1.networkutopia.com, 212.212.212.1, A)
- Set up authoritative server at that name/address
  - Create records for the services:
    - type A record for www.networkuptopia.com
    - type MX record for @networkutopia.com email

Attacking DNS

DDoS attacks

- Bombard root servers with traffic
  - Not successful to date
  - Traffic Filtering
  - Local DNS servers cache IPs of TLD servers, bypassing root
- Bombard TLD servers
  - Potentially more dangerous

Redirect attacks

- Man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus replies to DNS server that caches

Exploit DNS for DDoS

- Send queries with spoofed source address: target IP
- Requires amplification

Tools

- dig
  - $ dig cs.swarthmore.edu
  - $ dig cs.swarthmore.edu ns
  - $ dig @dns.cs.swarthmore.edu cs.swarthmore.edu mx
  - $ man dig
- host
  - $ host cs.swarthmore.edu
  - $ host -t ns cs.swarthmore.edu
  - $ host -t mx cs.swarthmore.edu dns.cs.swarthmore.edu
  - $ man host

Tools (cont)

- nslookup
  - $ nslookup cs.swarthmore.edu
  - $ nslookup cs.swarthmore.edu dns.cs.swarthmore.edu
- whois
  - $ whois google.com
  - $ whois swarthmore.edu

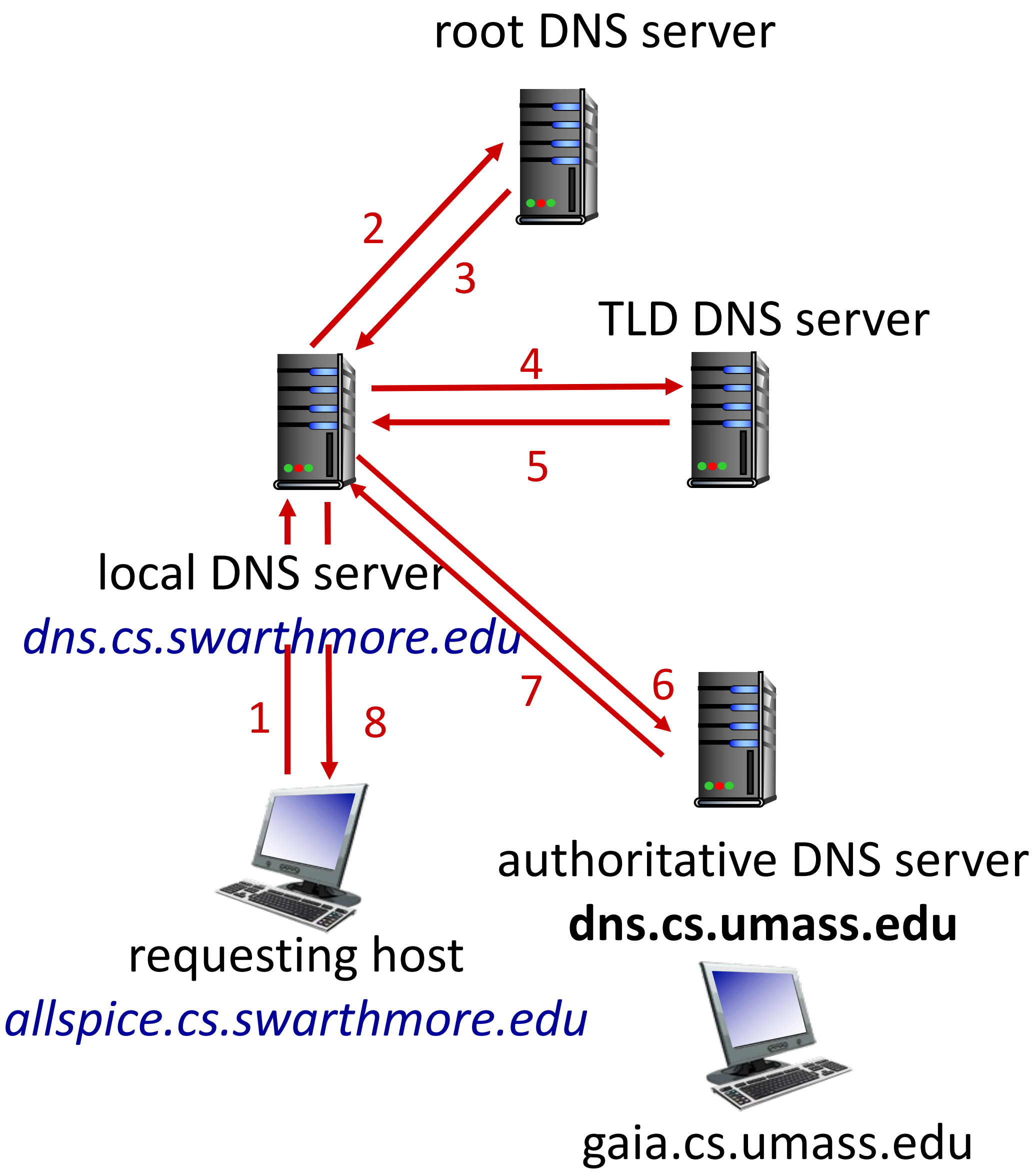
DNS security

DNS Vulnerabilities:

- No authentication
- Connectionless transport layer protocol (UDP)

DNS Attacks:

- Amplification Attack
- Cache Poisoning
- Man-in-the-middle
- DNS Redirection
- DDoS
- DNS Injection



Attacking DNS

DDoS attacks

- Bombard root servers with traffic
  - Not successful to date
  - Traffic Filtering
  - Local DNS servers cache IPs of TLD servers, bypassing root
- Bombard TLD servers
  - Potentially more dangerous

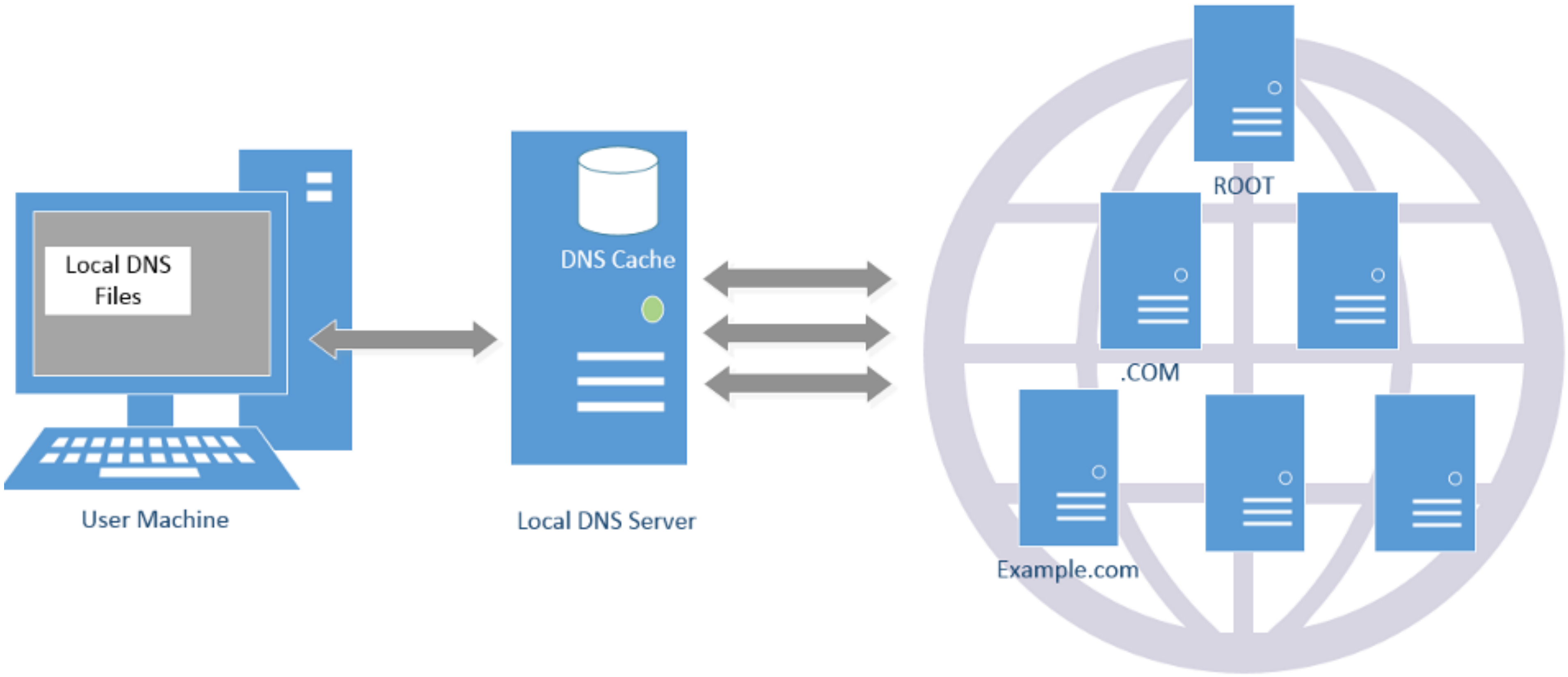
Redirect attacks

- Man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus replies to DNS server that caches

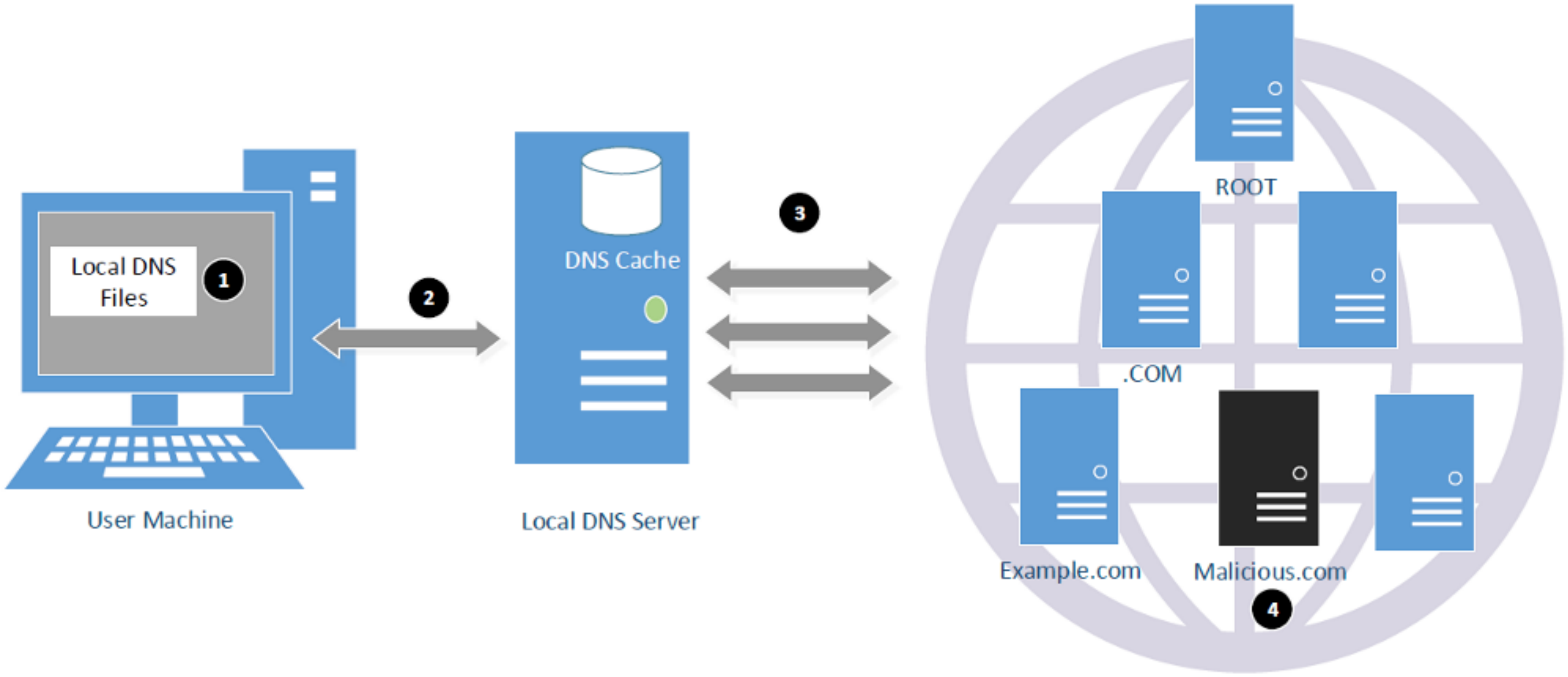
Exploit DNS for DDoS

- Send queries with spoofed source address: target IP
- Requires amplification

DNS Query Process and Cache



Attack Surface Overview



Denial Of Service

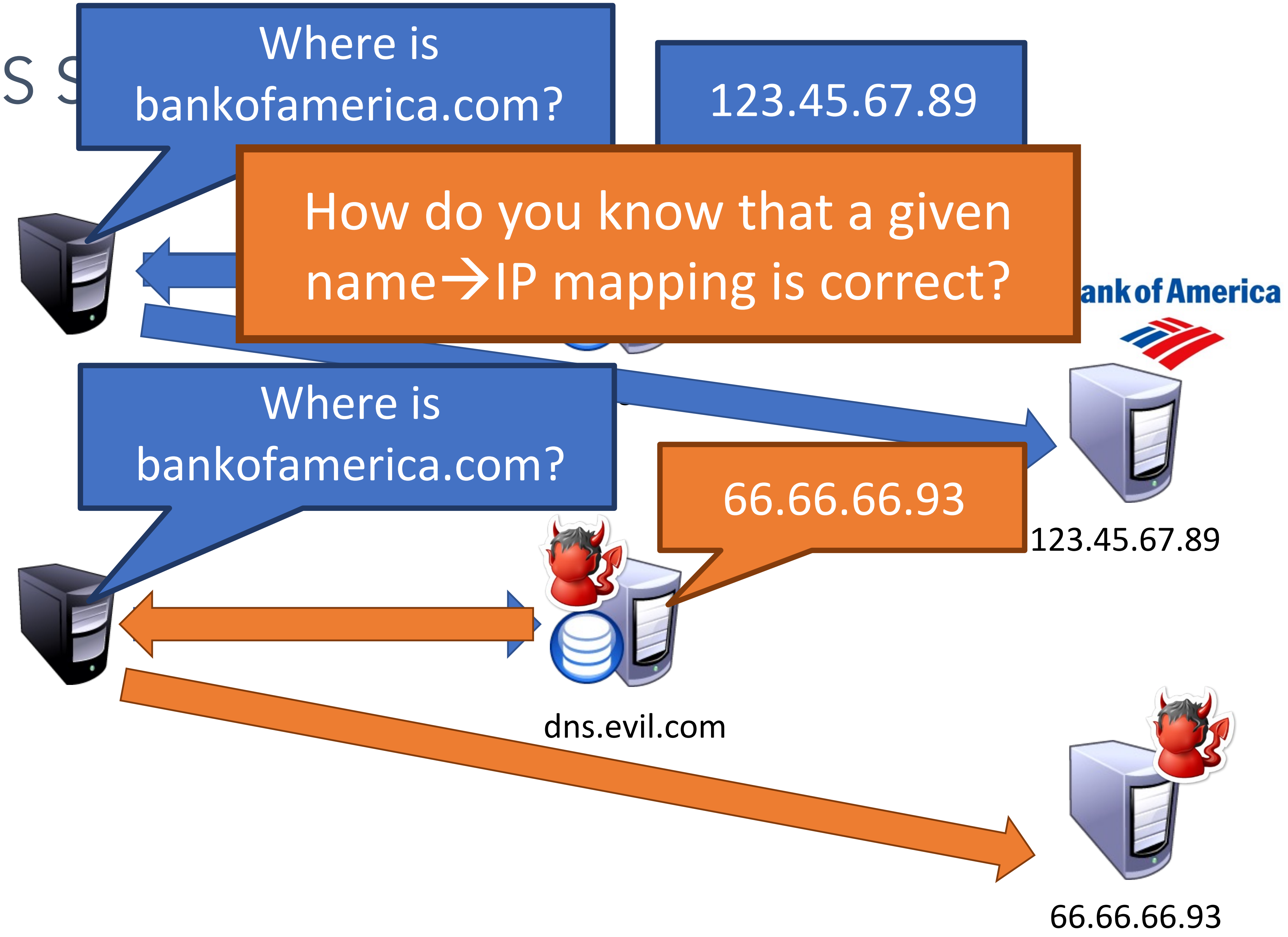
- Flood DNS servers with requests until they fail
- October 2002: massive DDoS against the root name servers
  - What was the effect?
  - … users didn't even notice
  - Root zone file is cached almost everywhere
- More targeted attacks can be effective
  - Local DNS server → cannot access DNS
  - Authoritative server → cannot access domain

DNS Hijacking

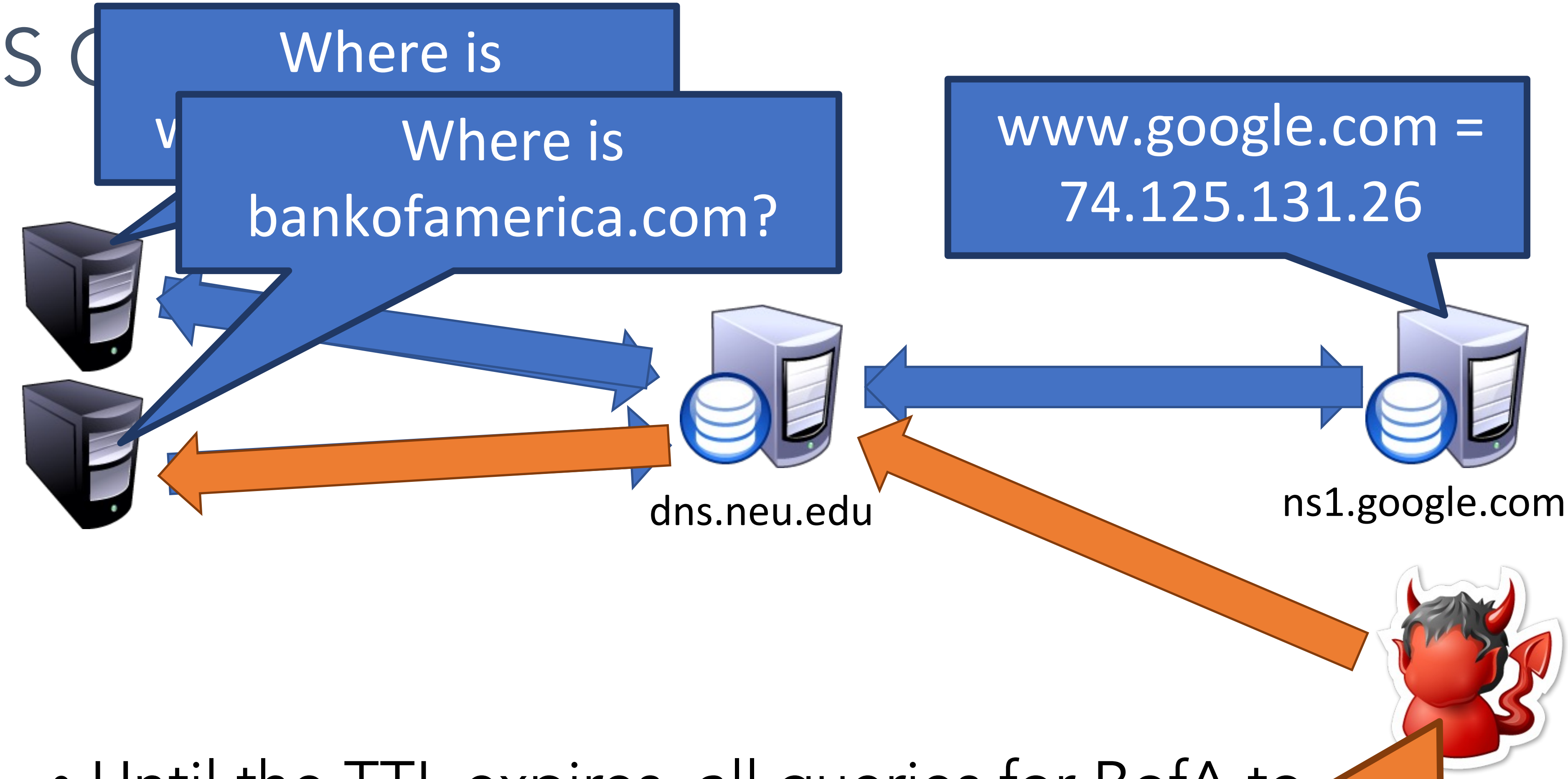
- Infect their OS or browser with a virus/trojan
  - e.g. Many trojans change entries in /etc/hosts
  - *.bankofamerica.com → evilbank.com
- Man-in-the-middle

- Response Spoofing
  - Eavesdrop on requests
  - Outrace the servers response

DNS S

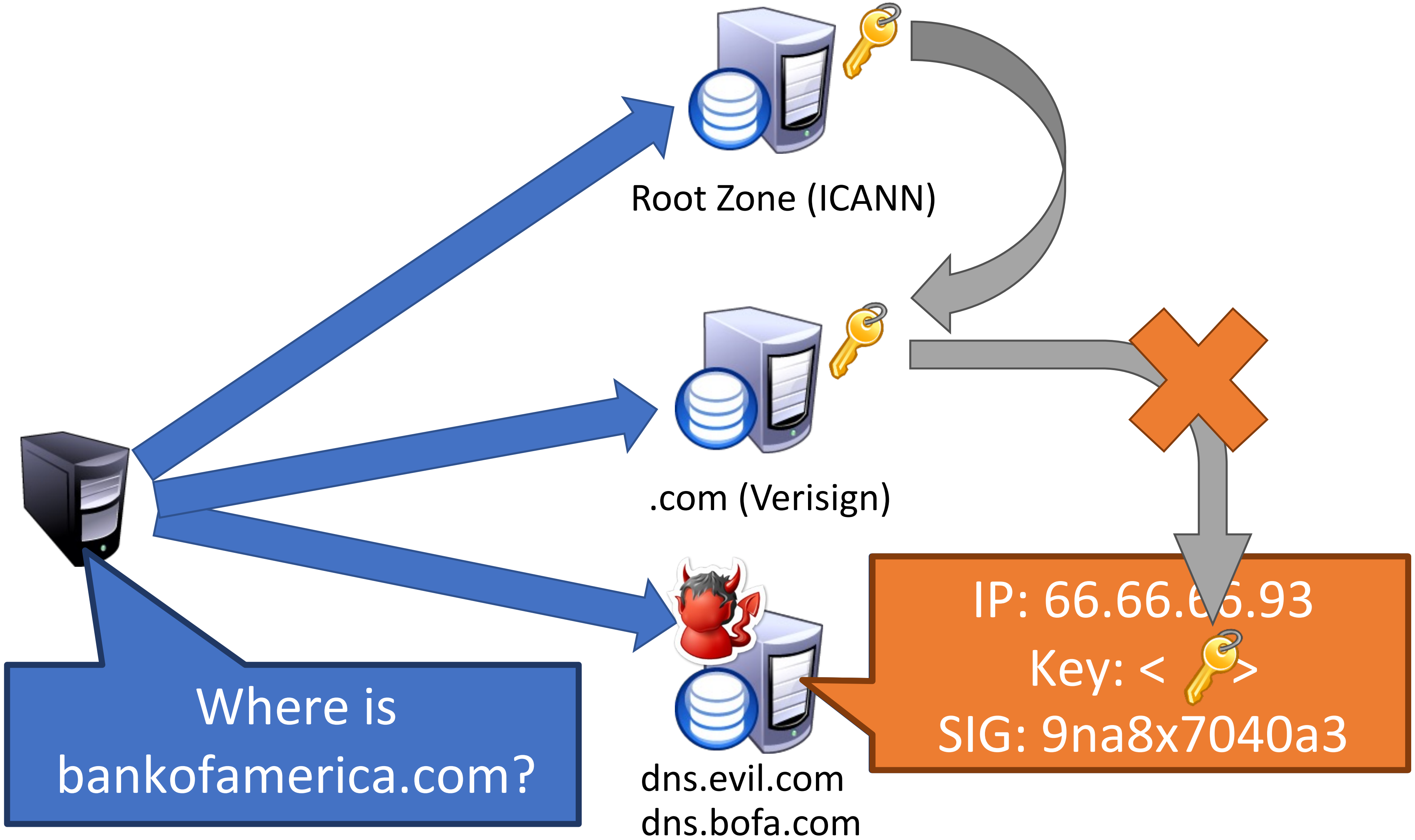

Where is
bankofamerica.com?
123.45.67.89
How do you know that a given
name→IP mapping is correct?
ank of America
123.45.67.89
Where is
bankofamerica.com?
66.66.66.93
dns.evil.com
66.66.66.93

DNS C



- Until the TTL expires, all queries for BofA to dns.neu.edu will return poisone
- Much worse than spoofing/mar
  - Whole ISPs can be impacted!

DNSSEC Hierarchy of Trust



Solution: DNSSEC

- Cryptographically sign critical resource records
  - Resolver can verify the cryptographic signature
- Two new resource types
  - Type = DNSKEY
    - Name = Zone domain name
    - Value = Public key for the zone
  - Type = RRSIG
    - Name = (type, name) tuple, i.e. the query itself
    - Value = Cryptographic signature of the query results

Creates a hierarchy of trust within each zone

Prevents hijacking and spoofing

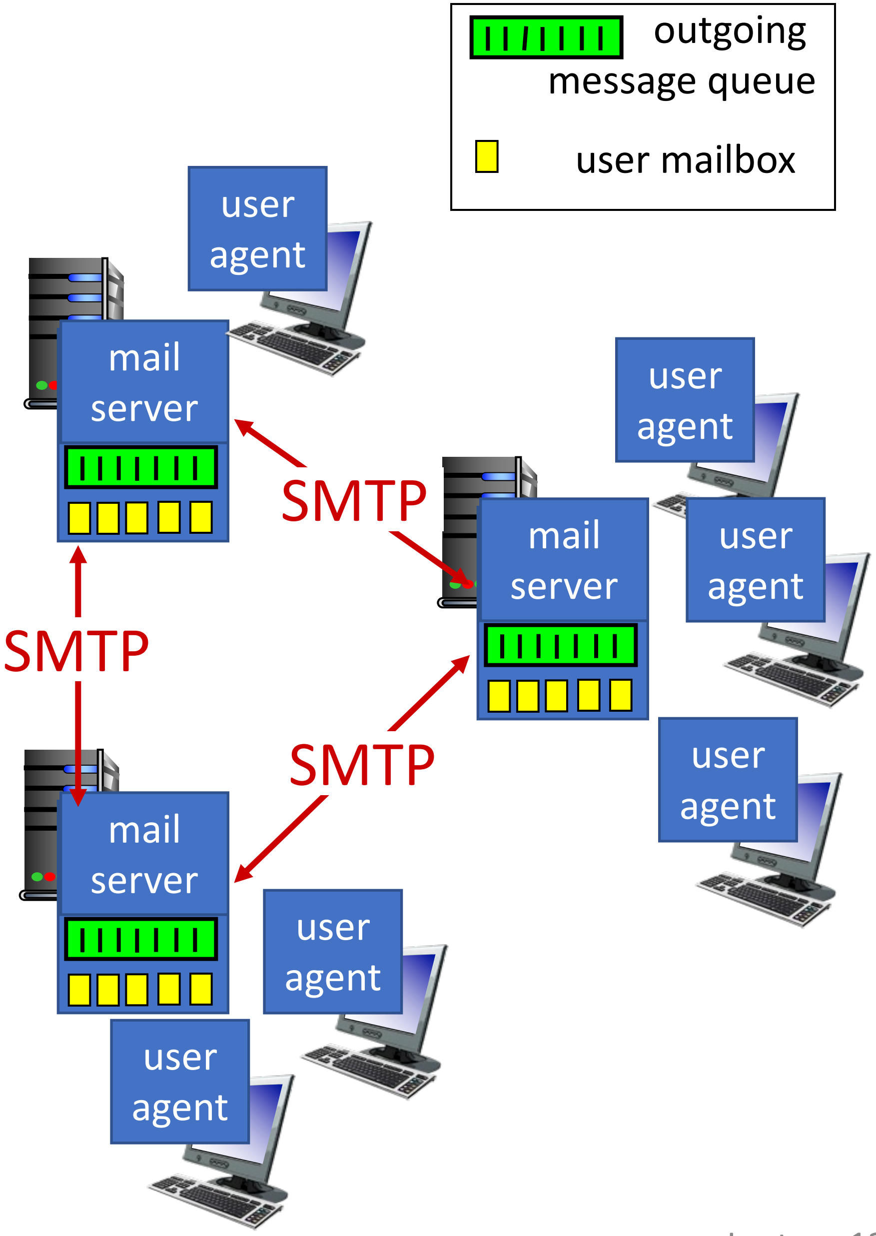
Today

- Three main parts to email:
  - Mail User Agent
  - Mail Transfer Agent
  - SMTP protocol used to negotiate transfers
- SMTP Protocol
- Mail Access Protocols
  - POP3
  - IMAP
  - Webmail

Electronic mail

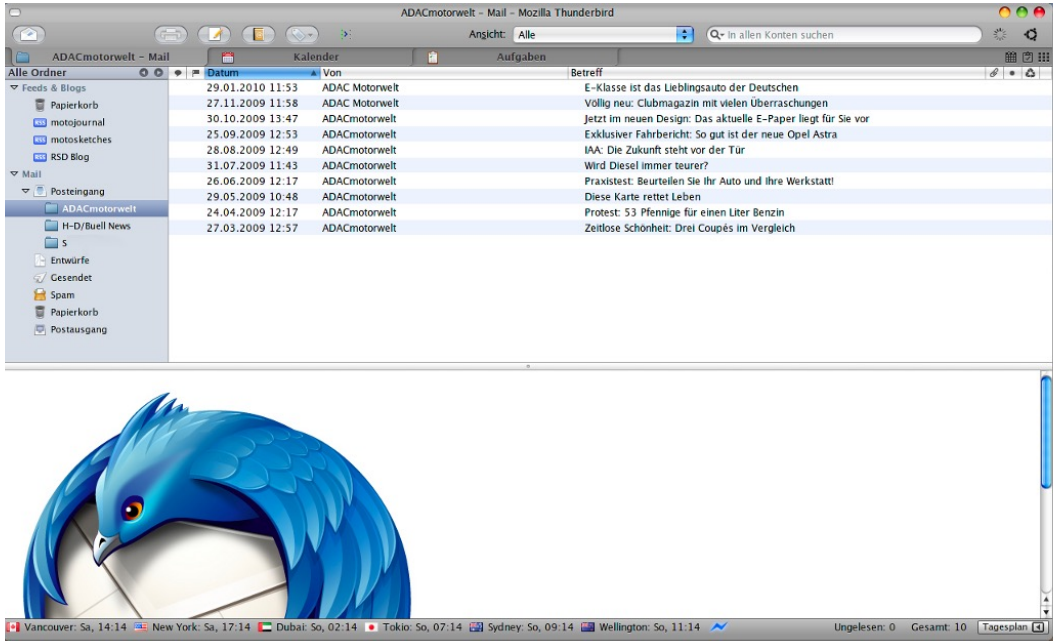
Three major components:

- mail user agent (MUA)
- mail transfer agent (MTA)
- simple mail transfer protocol



Mail User Agent a.k.a "mail reader"

- composing, editing, reading mail messages
- Outlook, Thunderbird, iPhone mail client



Mail Transfer Agent. a.k.a mail servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages (one-way)
  - client: sending mail server
  - “server”: receiving mail server

If you were designing email, what would happen when Alice sends an email to Bob?

A. Alice mail **client** -> Bob's mail **server**

B. Alice mail **server** -> Bob's mail **server**

C. Alice mail **client** -> Bob's mail **client**

D. Alice mail **server** -> Bob's mail **client**

Scenario: Alice sends message to Bob

1) Alice uses a MUA to compose message "to" bob@swarthmore.edu

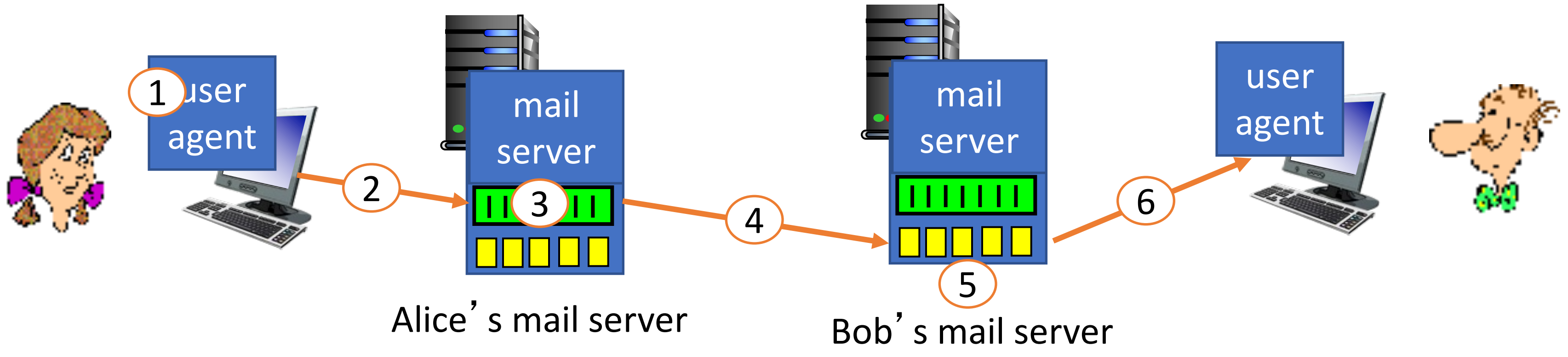
2) Alice's MUA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his MUA to read message



Mail Servers: Ever Vigilant

- Always on, because they always need to be ready to accept mail.

- Usually owned by ISP
  - You use the email server for either Swarthmore College, or the CS department.

Simple Mail Transfer: SMTP [RFC 2821]

- TCP: reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server
- Messages must be in 7-bit ASCII
- Command/response interaction (like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase

Simple Mail Transfer: SMTP [RFC 2821]

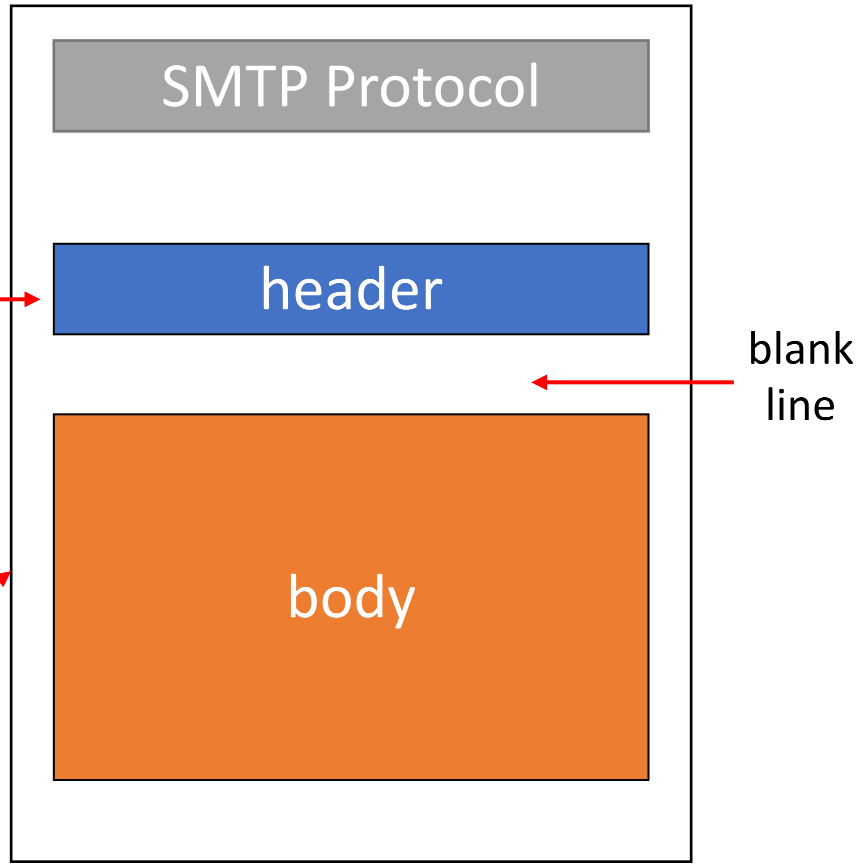
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - handshaking (greeting), MAIL FROM:, RCPT TO:
  - transfer of messages
  - closure

SMTP Message Format

RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only
  - Signal EOM with "\r\n.\r\n"



Try SMTP interaction for yourself:

- **`telnet allspice.cs.swarthmore.edu 25`**
- You should see a 220 reply from the server.
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

(lets you send email without using email client (MUA))

Demo

Sample SMTP interaction

```
$ telnet allspice.cs.swarthmore.edu 25
Trying 130.58.68.9…
Connected to allspice.cs.swarthmore.edu
220 allspice.cs.swarthmore.edu ESMTP Postfix
HELO cs.swarthmore.edu
250 allspice.cs.swarthmore.edu
MAIL FROM:<chaganti@cs.swarthmore.edu>
250 2.1.0 OK
RCPT TO:<chaganti@cs.swarthmore.edu>
250 2.1.5 OK
DATA
354 End data with <CR><LF>.<CR><LF>
To: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
From: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
Subject: Telnet test message

This is a test message, via telnet, to myself.
.
```

Sample SMTP interaction

$ telnet allspice.cs.swarthmore.edu 25

Trying 130.58.68.9…

Connected to allspice.cs.swarthmore.edu

220 allspice.cs.swarthmore.edu ESMTP Postfix

HELO cs.swarthmore.edu

250 allspice.cs.swarthmore.edu

MAIL FROM:<chaganti@cs.swarthmore.edu>

250 2.1.0 OK

RCPT TO:<chaganti@cs.swarthmore.edu>

250 2.1.5 OK

DATA

354 End data with <CR><LF>.<CR><LF>

To: Vasanta Chaganti <chaganti@cs.swarthmore.edu>

From: Vasanta Chaganti <chaganti@cs.swarthmore.edu>

Subject: Telnet test message

This is a test message, via telnet, to myself.

End of message:
CRLF (Dot) CRLF

What keeps us from entering a fake information (e.g., FROM address)?

A. Nothing.

B. The MTA checks that the FROM is valid.

C. We enter a name/password logging into the MTA.

Fun Demo

Wait, this seems too horrible to be true. Surely we can prevent header forging?

(How or why not?)

A. Yes

B. No

Message Signing

1. Sender creates cryptographic public/private key pair, publishes public key to the world.

2. Sender uses private key to sign messages.

3. Receiver can verify*, using published public key, that only the holder of the corresponding private key could have sent the message.

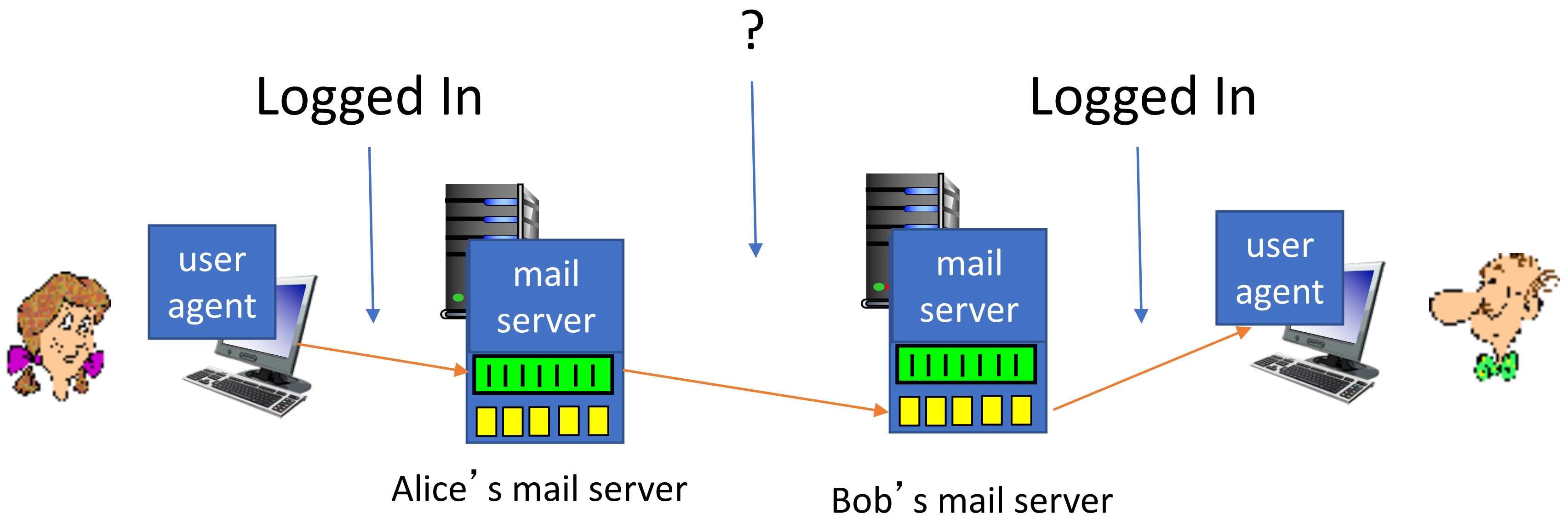
* With very high probability.

Message Signing: Challenges

- Disseminating public keys
  - How do you trust that the published public key isn’t also a lie?

- It’s more work, can’t be bothered…
  - Adoption is very low

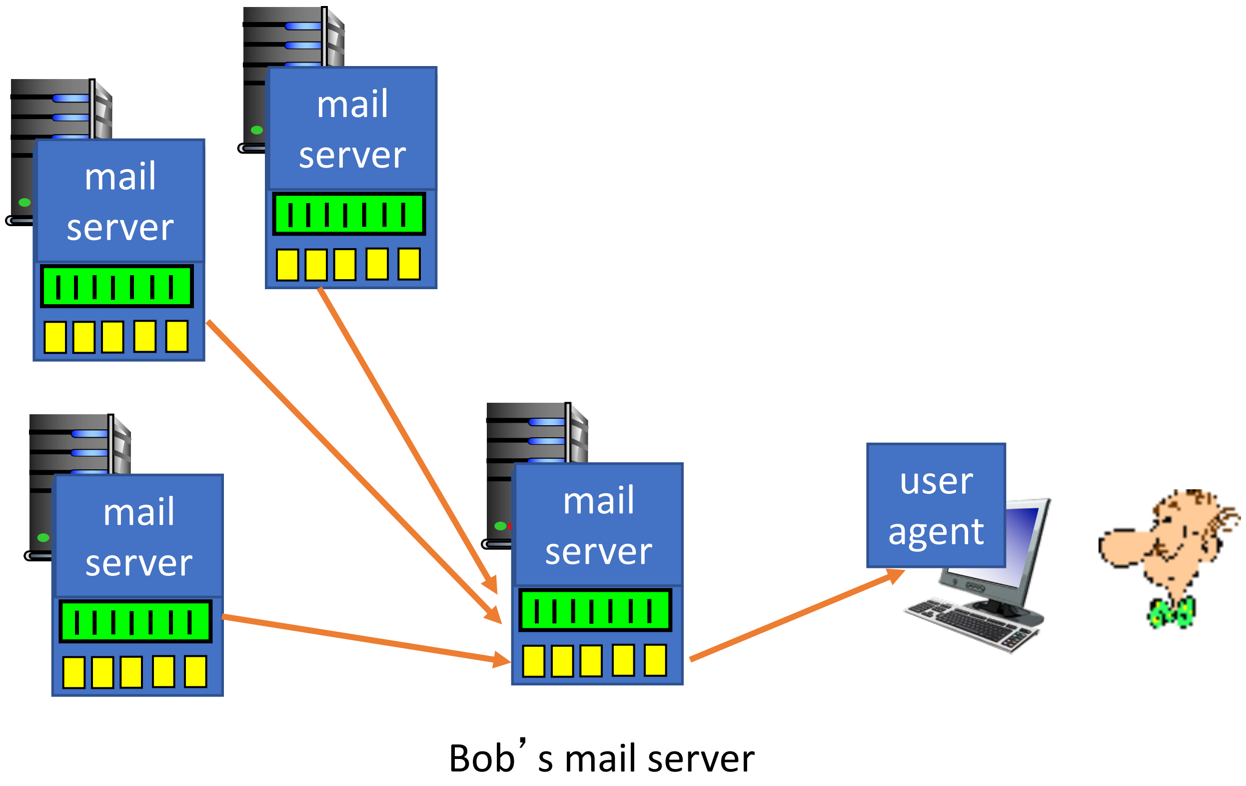
Logging In / Passwords



Logging In / Passwords

Any mail server may need to send a message to Bob's.

Tough for them all to share credentials…



SMTP versus HTTP

- HTTP: pull
- SMTP: push

- Both have ASCII command/response interaction, status codes

- HTTP: each object encapsulated in its own response message

- SMTP: multiple objects sent in multipart message

SMTP: final words

- SMTP uses persistent connections
  - Can send multiple emails in one session
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

If SMTP only allows 7-bit ASCII, how do we send pictures/videos/files via email?

A. We encode these objects as 7-bit ASCII

B. We use a different protocol instead of SMTP

C. We’re really sending links to the objects, rather than the objects themselves

Base 64

- Designed to be an efficient way to send binary data as a string
- Uses A-Z, a-z,0-9, “+” and “/” as digits
- A number with digits $d_n d_{n-1} \ldots d_1 d_0 =$
  $64^n * d_n + 64^{n-1} * d_{n-1} + \ldots + 64^* d_1 + d_0$
- Recall from CS 31: Other non-base-10 number systems (binary, octal, hex).

Multipurpose Internet Mail Extensions (MIME)

- Special formatting instructions

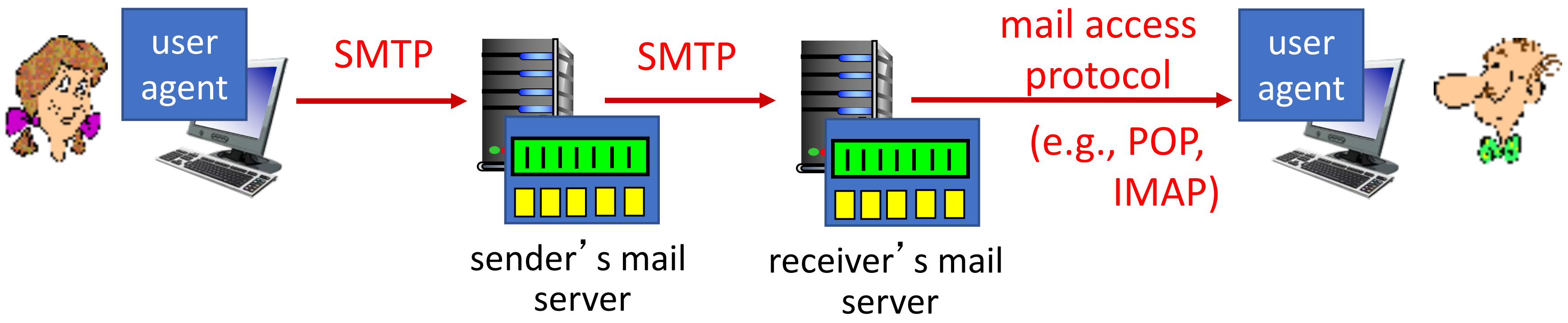
- Indicated in the header portion of message (not SMTP)
  - SMTP does *not* care, just looks like message data

- Supports
  - Text in character sets other than ASCII
  - Non-text attachments
  - Message bodies with multiple parts
  - Header information in non-ASCII character sets

MIME

- Adds optional headers
  - Designed to be compatible with non-MIME email clients
  - Both clients must understand it to make sense of it
- Specifies content type, other necessary information
- Designates a boundary between email text and attachments

Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol: authorization, download
  - IMAP: Internet Mail Access Protocol: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol

authorization phase

- client commands:
  - **user:** declare username
  - **pass:** password
- server responses
  - **+OK**
  - **-ERR**

transaction phase, client:

- **list:** list message numbers
- **retr:** retrieve message by number
- **dele:** delete
- **quit**

```
S:  +OK POP3 server ready
C:  user bob
S:  +OK
C:  pass hungry
S:  +OK user successfully logged on
```

```
C:  list
S:  1 498
S:  2 912
S:  .
C:  retr 1
S:  <message 1 contents>
S:  .
C:  dele 1
C:  retr 2
S:  <message 1 contents>
S:  .
C:  dele 2
C:  quit
S:  +OK POP3 server signing off
```

More about POP3

- Previous example uses "download and delete" mode
  - Bob cannot re-read e-mail if he changes client

- POP3 "download-and-keep": copies of messages on different clients

- POP3 is stateless across sessions

- Limitations:
  - Can't retrieve just the headers
  - Can't impose structure on messages

IMAP

- Keeps all messages in one place: at server
- Allows user to organize messages in folders
- Keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name
- Can request pieces of a message (e.g., text parts without large attachments)

Webmail

- Uses a web browser
- Sends emails using HTTP rather than POP3 or IMAP
- Mail is stored on the 3rd party webmail company's servers

Summary

- Three main parts to email:
  - Mail User Agent (mail client): read / write for humans
  - Mail Transfer Agent: server that accepts / sends messages
  - SMTP protocol used to negotiate transfers
- No SMTP support for fraud detection
- Extensions (MIME) and encodings (Base64) for sending non-text data