Computer Networks and Applications

COMP 3331/COMP 9331

Week 2

Application Layer (Email, DNS, P2P)

Reading Guide: Chapter 2, Sections 2.3, 2.4, 2.5

Application Layer: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 electronic mail
- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Self study

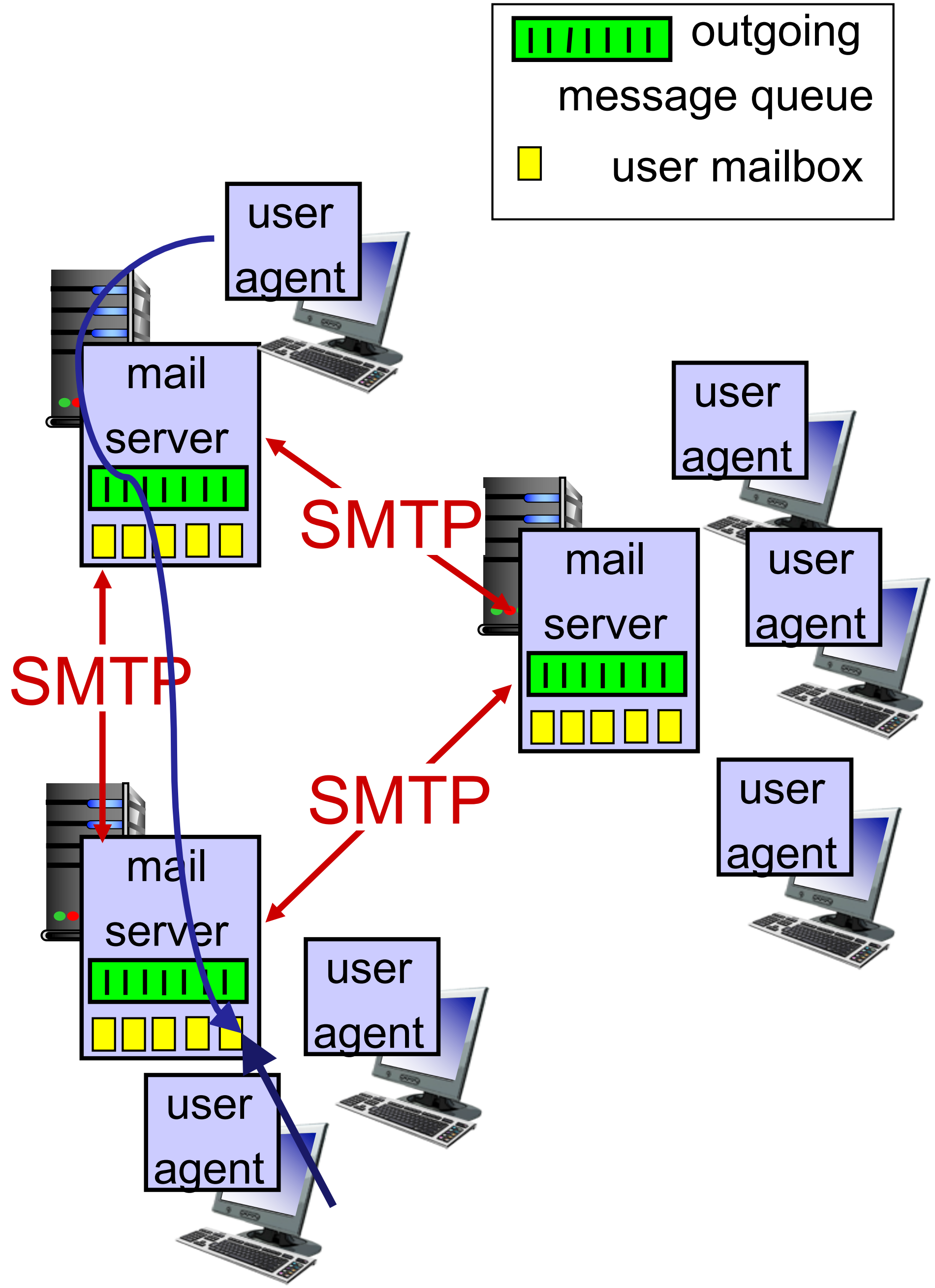
Electronic mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

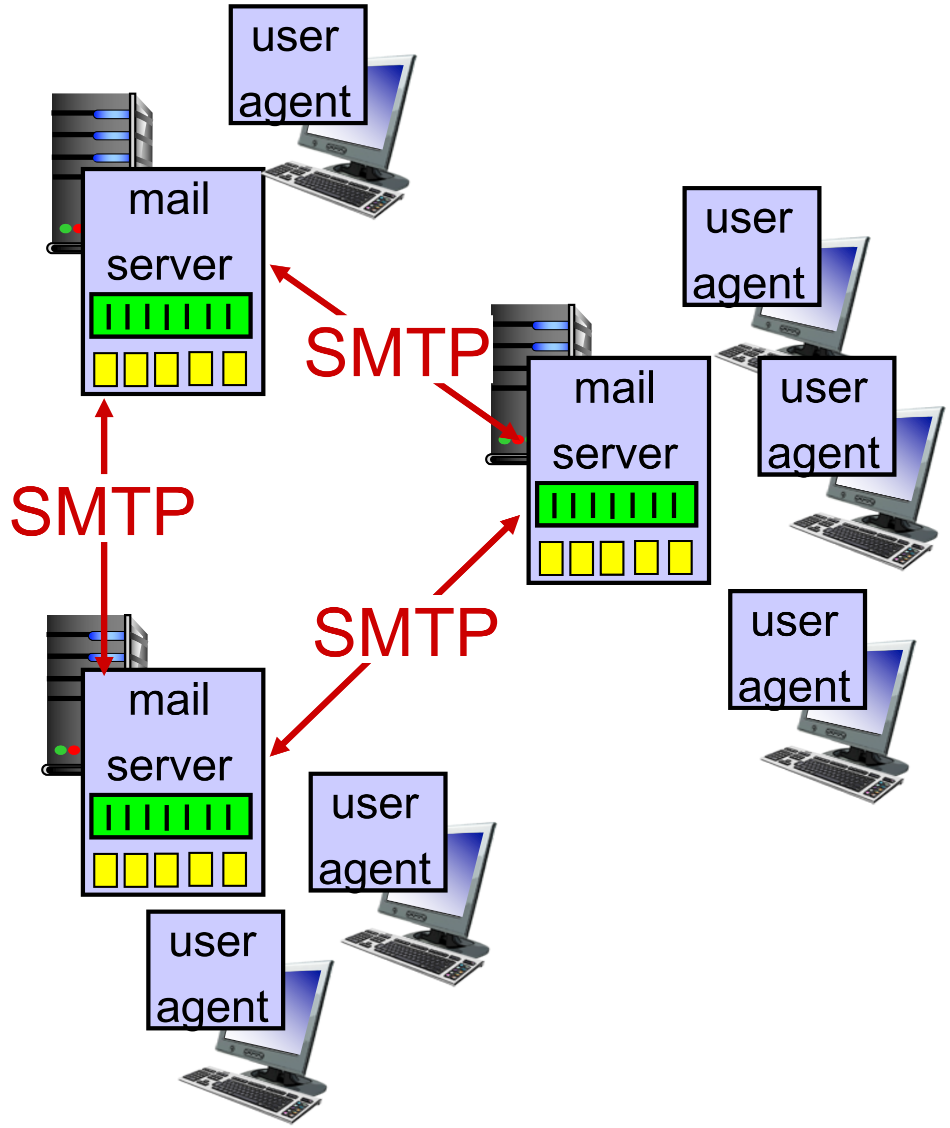
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



Electronic mail: mail servers

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
  - client: sending mail server
  - "server": receiving mail server

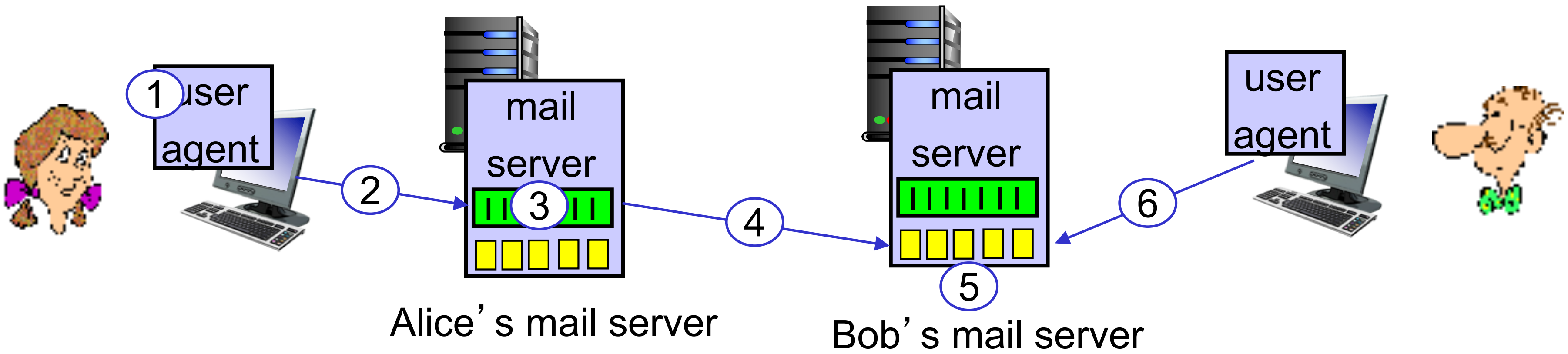


Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction (like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "to" `bob@someschool.edu`
2) Alice's UA 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 user agent to read message



Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

How to tell a fake email?

sheldon.cooper@bigbang.com 11 March 2013 11:44 AM
To: salilk@cse.unsw.edu.au
(No Subject)

Helo Salil

BAZINGA !!

best
Dr. Sheldon Cooper

Examine Long Headers or Raw Source

sheldon.cooper@bigbang.com 11 March 2013 11:44 AM
To: salilk@cse.unsw.edu.au Hide Details
Return-Path: <sheldon.cooper@bigbang.com>
Received: From bigbang.com ([129.94.242.19] == wagner.orchestra.cse.unsw.EDU.AU) (ident-user cs3331) (cse-authentic-sender cs3331) (for <salilk@cse.unsw.edu.au>) By note With Smtp ; Mon, 11 Mar 2013 11:44:05 +1100
Message-Id: <1130311004405.4478@cse.unsw.edu.au>
(No Subject)

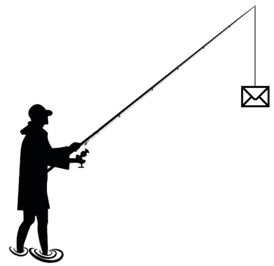
Helo Salil

BAZINGA !!

best
Dr. Sheldon Cooper

Further reading: http://www.millersmiles.co.uk/identitytheft/spoofemail-060603.htm

Phishing



- Spear phishing
  - Phishing attempts directed at specific individuals or companies
  - Attackers may gather personal information (social engineering) about their targets to increase their probability of success
  - Most popular and accounts for over 90% of attacks

- Clone phishing
  - A type of phishing attack whereby a legitimate, and previously delivered email containing an attachment or link has had its content and recipient address(es) taken and used to create an almost identical or cloned email.
  - The attachment or link within the email is replaced with a malicious version and then sent from an email address spoofed to appear to come from the original sender.

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

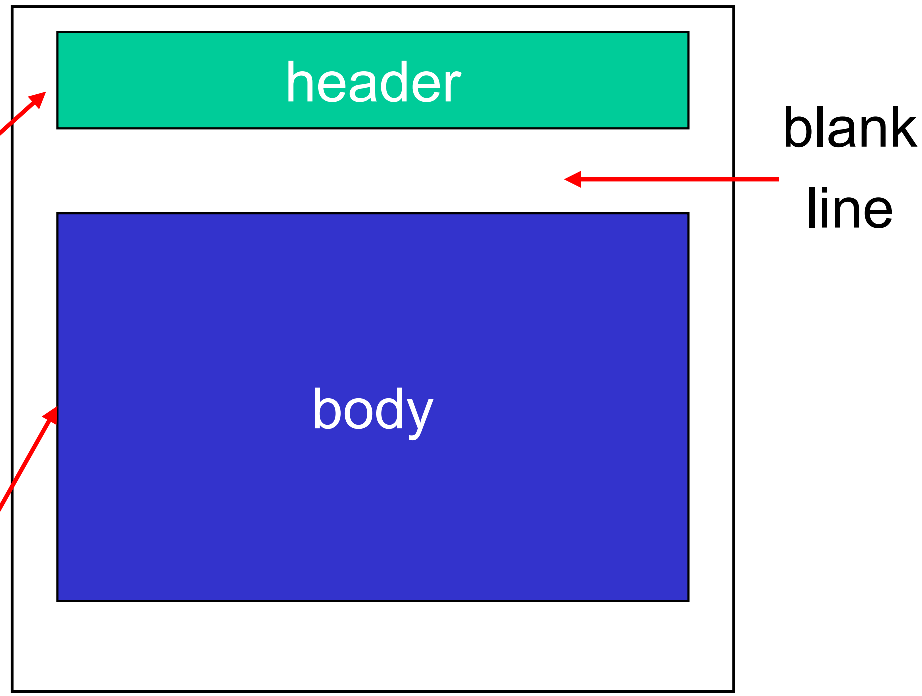
Mail message format

SMTP: protocol for exchanging email msgs

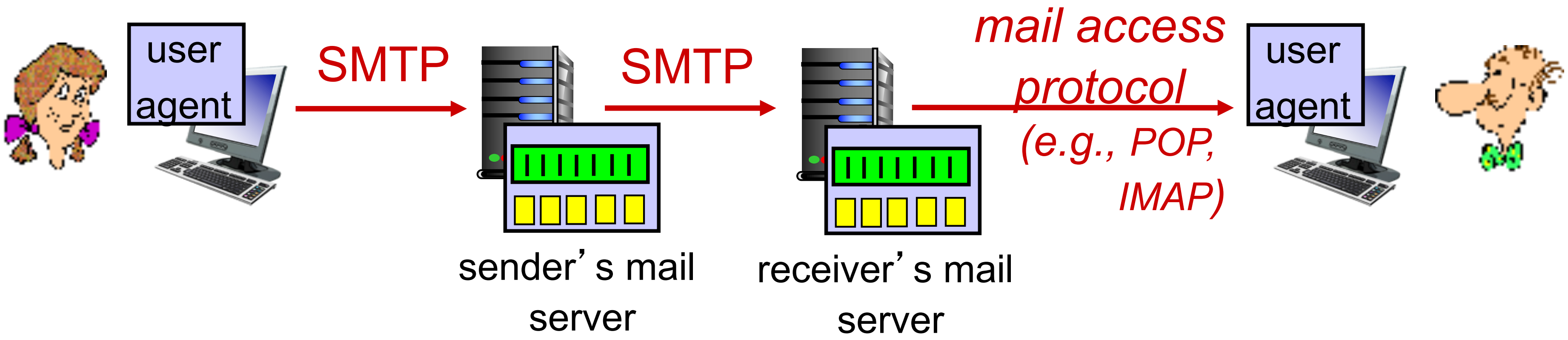
RFC 5322 (822,2822): standard for text message format (Internet Message Format, IMF):

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

  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only



Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - HTTP(S): Gmail, Yahoo! Mail, etc.

Quiz: SMTP

Why do we have Sender's mail server?

- User agent can directly connect with recipient mail server without the need of sender's mail server? What's the catch?

Why do we have a separate Receiver's mail server?

- Can't the recipient run the mail server on own end system?

Quiz: E-mail attachments?

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

A: We use a different protocol instead of SMTP

B: We encode these objects as 7-bit ASCII

C: We're really sending links to the objects, rather than the objects themselves

D: Like HTTP, we can send these in binary

Quiz: HTTP vs SMTP

❖ Which of the following is not true?

A. HTTP is pull-based, SMTP is push-based

B. HTTP uses a separate header for each object, SMTP uses a multipart message format

C. SMTP uses persistent connections

D. HTTP uses client-server communication but SMTP does not

2. Application Layer: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 electronic mail
- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

A nice overview: https://webhostinggeeks.com/guides/dns/

DNS: domain name system

people: many identifiers:

- TFN, name, passport #

Internet hosts, routers:

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

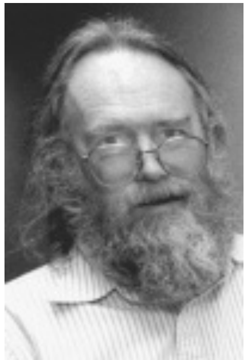
Q: how to map between IP address and name, and vice versa ?

Domain Name System:

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

DNS: History

- Initially all host-address mappings were in a hosts.txt file (in /etc/hosts):
  - Maintained by the Stanford Research Institute (SRI)
  - Changes were submitted to SRI by email
  - New versions of hosts.txt periodically FTP'd from SRI
  - An administrator could pick names at their discretion


Jon Postel

- As the Internet grew this system broke down:
  - SRI couldn't handle the load; names were not unique; hosts had inaccurate copies of hosts.txt

- The Domain Name System (DNS) was invented to fix this

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

DNS: services, structure

DNS services

- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name
  - Content Distribution Networks: use IP address of requesting host to find best suitable server
    - Example: closest, least-loaded, etc

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: *doesn't scale!*

Goals

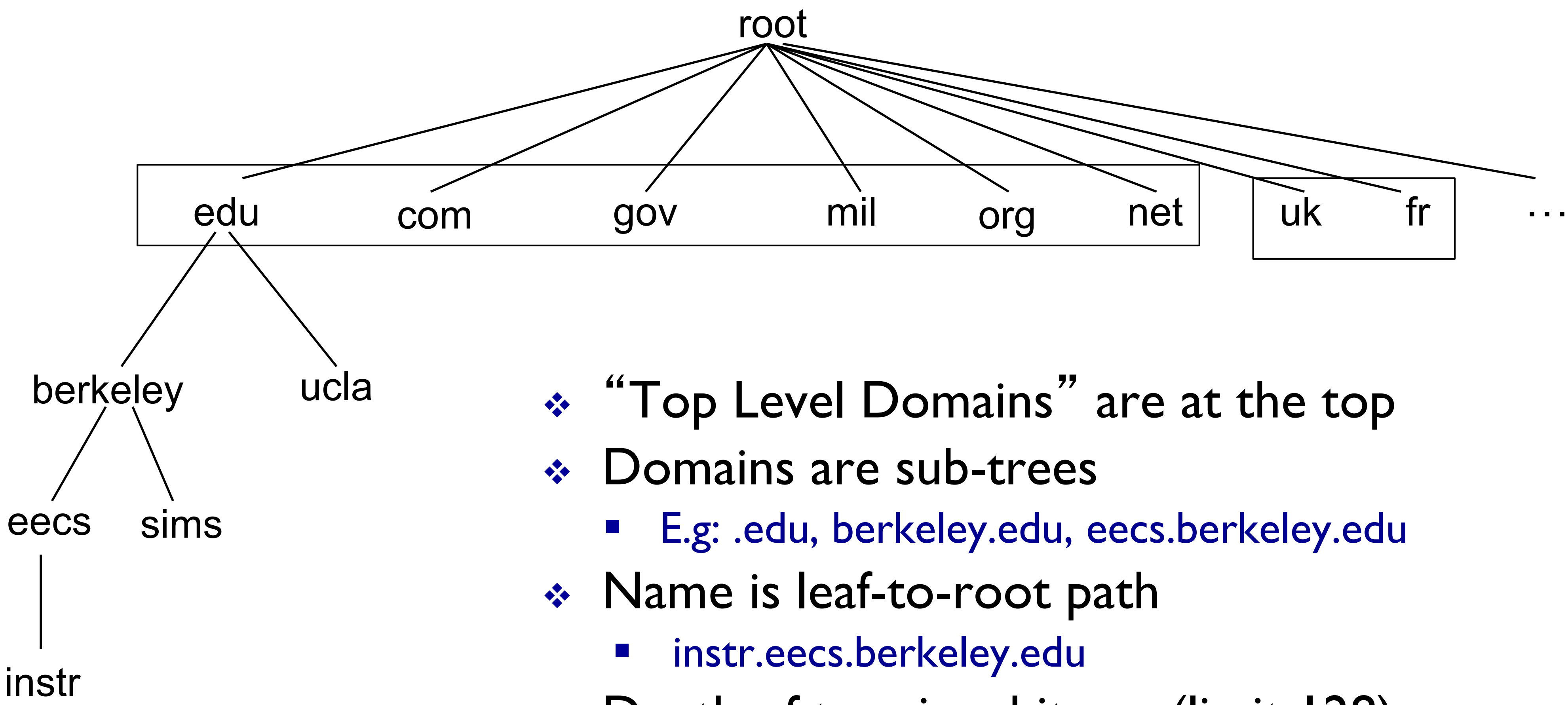
- No naming conflicts (uniqueness)
- Scalable
  - many names
  - (secondary) frequent updates
- Distributed, autonomous administration
  - Ability to update my own (machines') names
  - Don't have to track everybody's updates
- Highly available
- Lookups should be fast

Key idea: Hierarchy

Three intertwined hierarchies

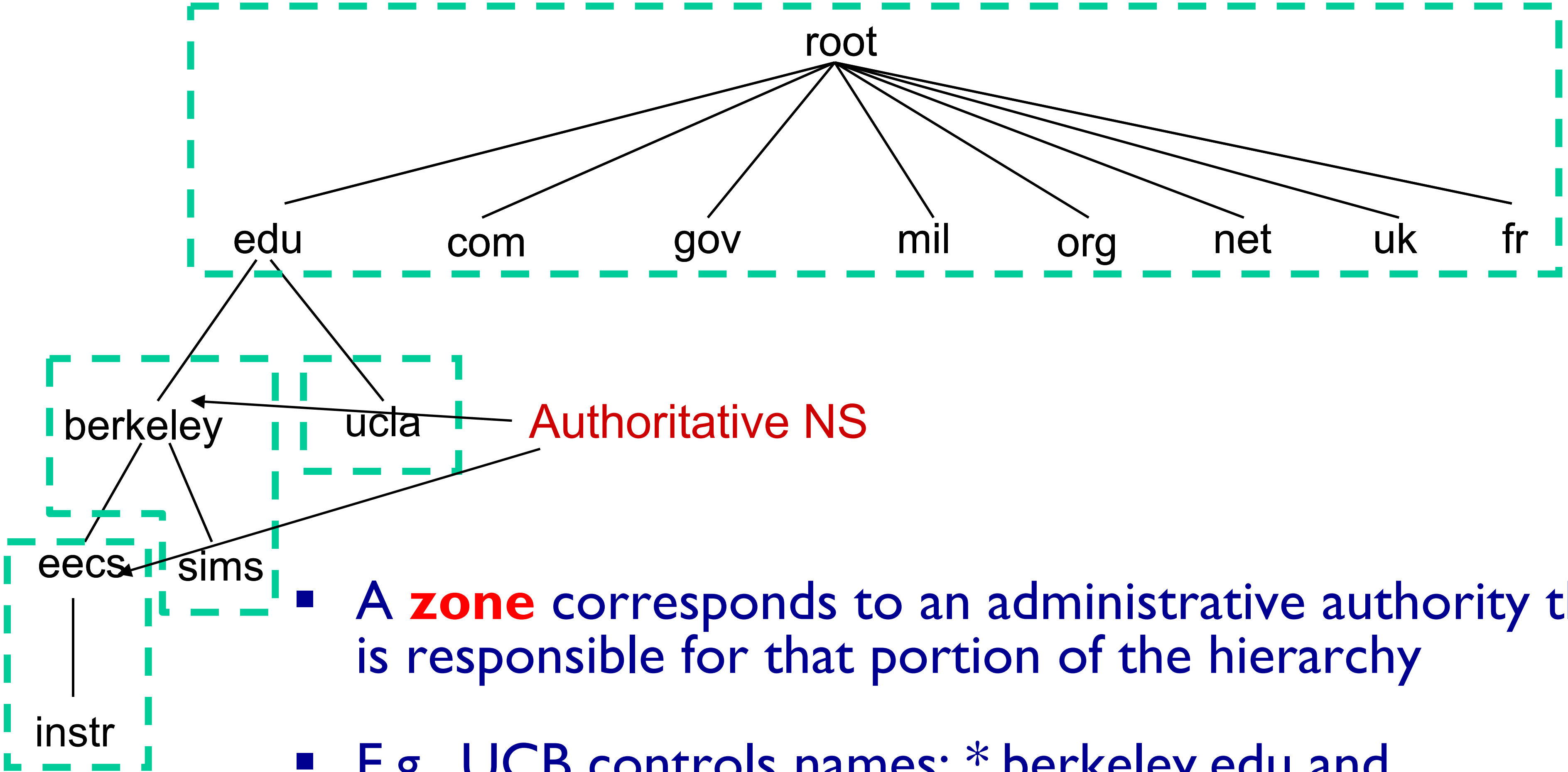
- Hierarchical namespace
  - As opposed to original flat namespace
- Hierarchically administered
  - As opposed to centralised
- (Distributed) hierarchy of servers
  - As opposed to centralised storage

Hierarchical Namespace



- "Top Level Domains" are at the top
- Domains are sub-trees
  - E.g: .edu, berkeley.edu, eecs.berkeley.edu
- Name is leaf-to-root path
  - instr.eecs.berkeley.edu
- Depth of tree is arbitrary (limit 128)
- Name collisions trivially avoided
  - each domain is responsible

Hierarchical Administration



- A **zone** corresponds to an administrative authority that is responsible for that portion of the hierarchy
- E.g., UCB controls names: *.berkeley.edu and *.sims.berkeley.edu
- E.g., EECS controls names: *.eecs.berkeley.edu

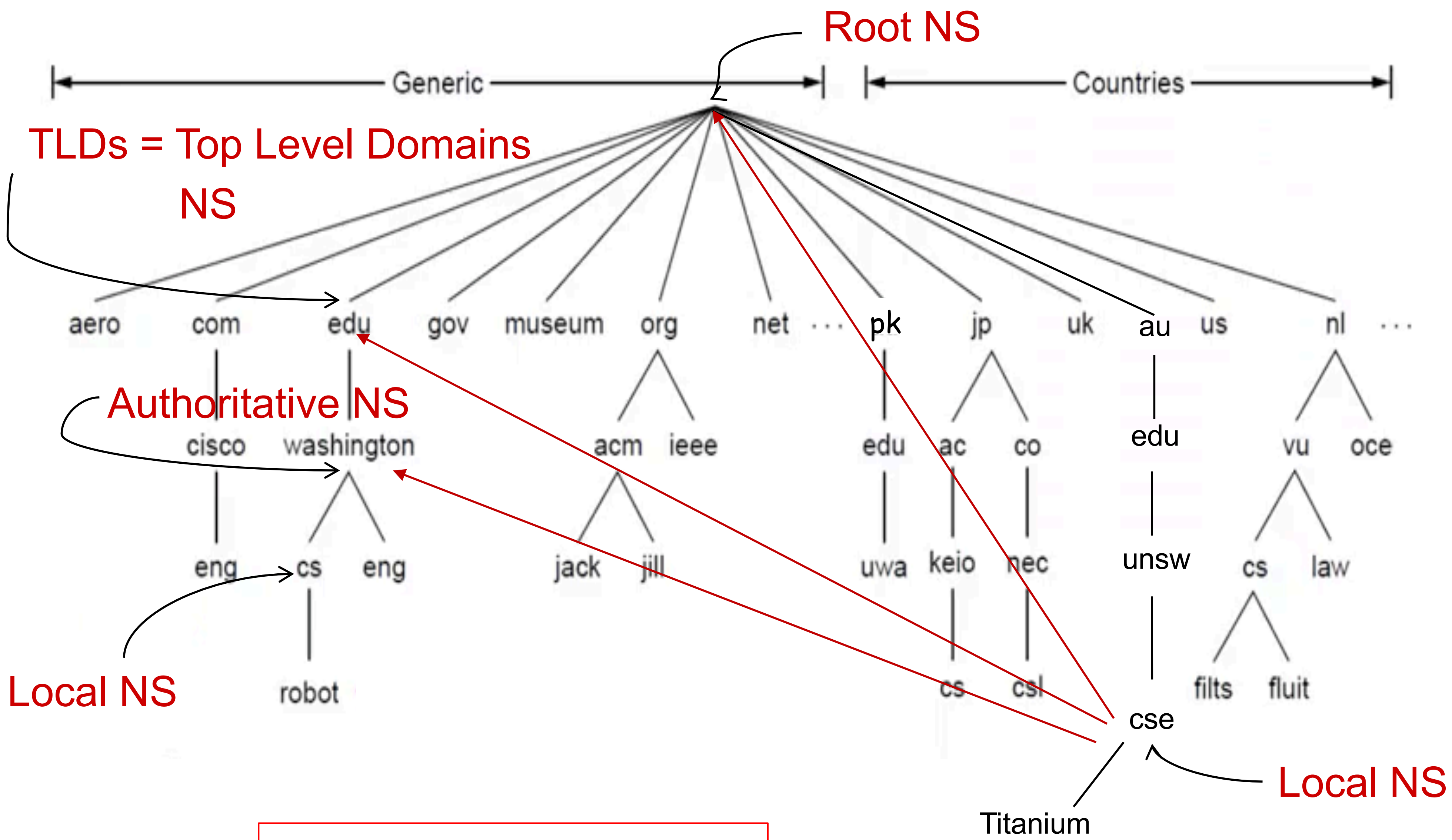
Server Hierarchy

- Top of hierarchy: Root servers
  - Location hardwired into other servers
- Next Level: Top-level domain (TLD) servers
  - .com, .edu, etc.
  - Managed professionally
- Bottom Level: Authoritative DNS servers
  - Actually store the name-to-address mapping
  - Maintained by the corresponding administrative authority

Server Hierarchy

- Each server stores a (small!) subset of the total DNS database
- An authoritative DNS server stores "resource records" for all DNS names in the domain that it has authority for
- Each server needs to know other servers that are responsible for the other portions of the hierarchy
  - Every server knows the root
  - Root server knows about all top-level domains

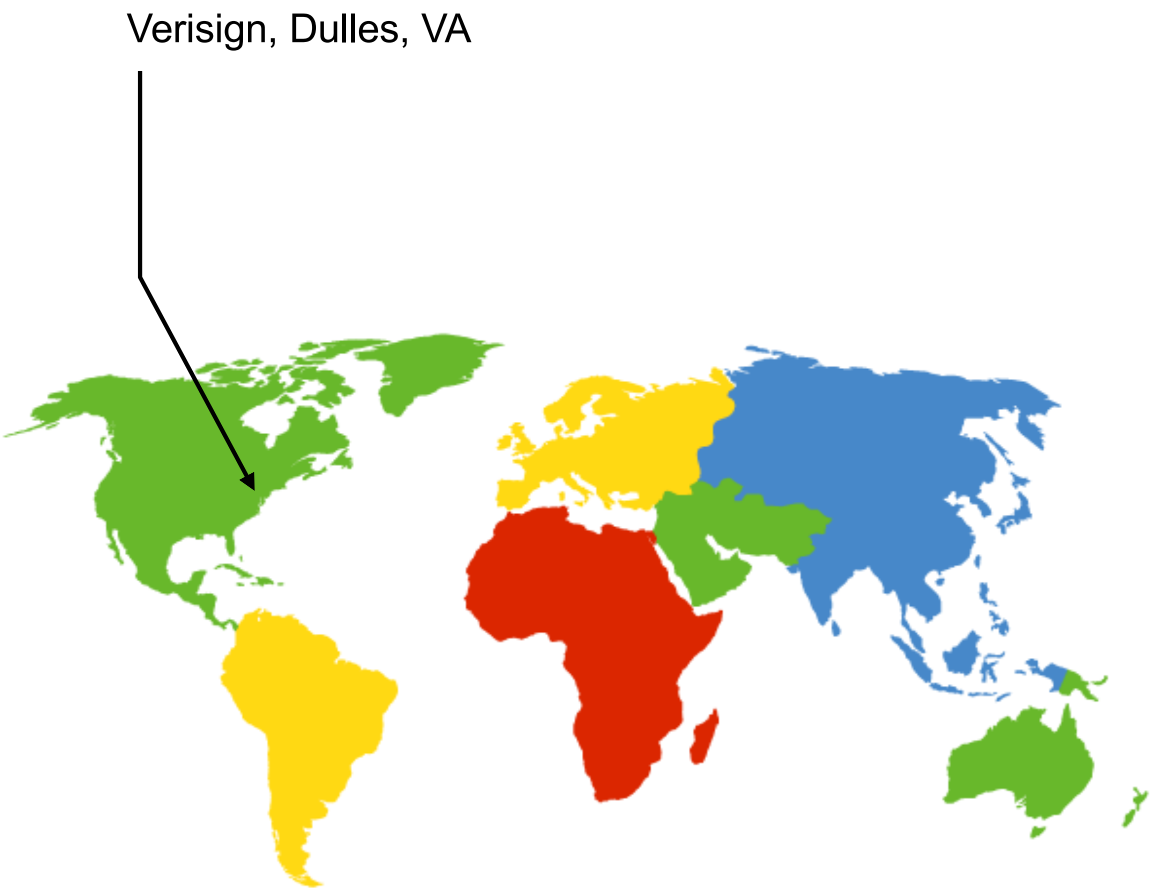
DNS: a distributed, hierarchical database



robot.cs.washington.edu

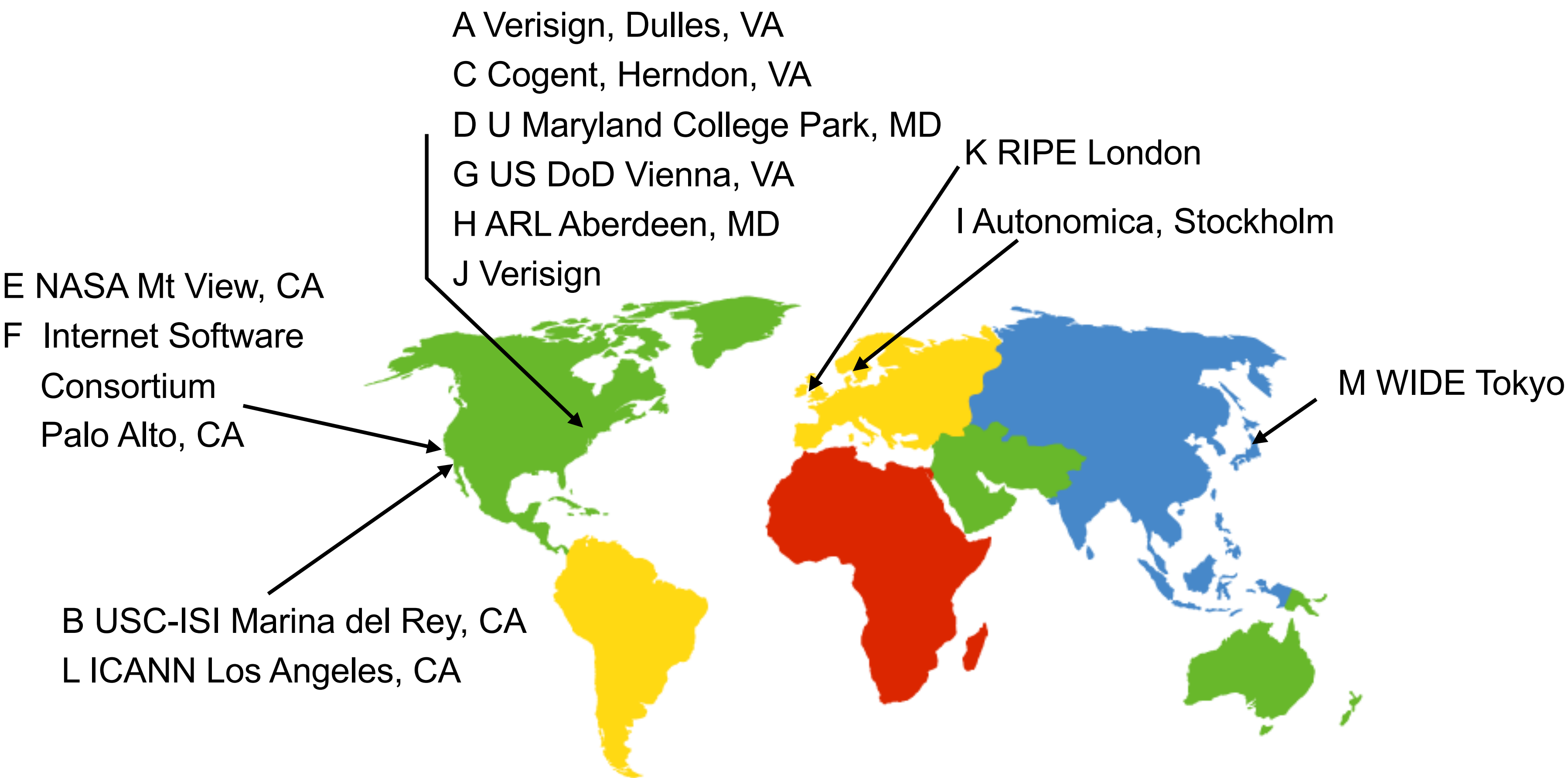
DNS Root

- Located in Virginia, USA
- How do we make the root scale?



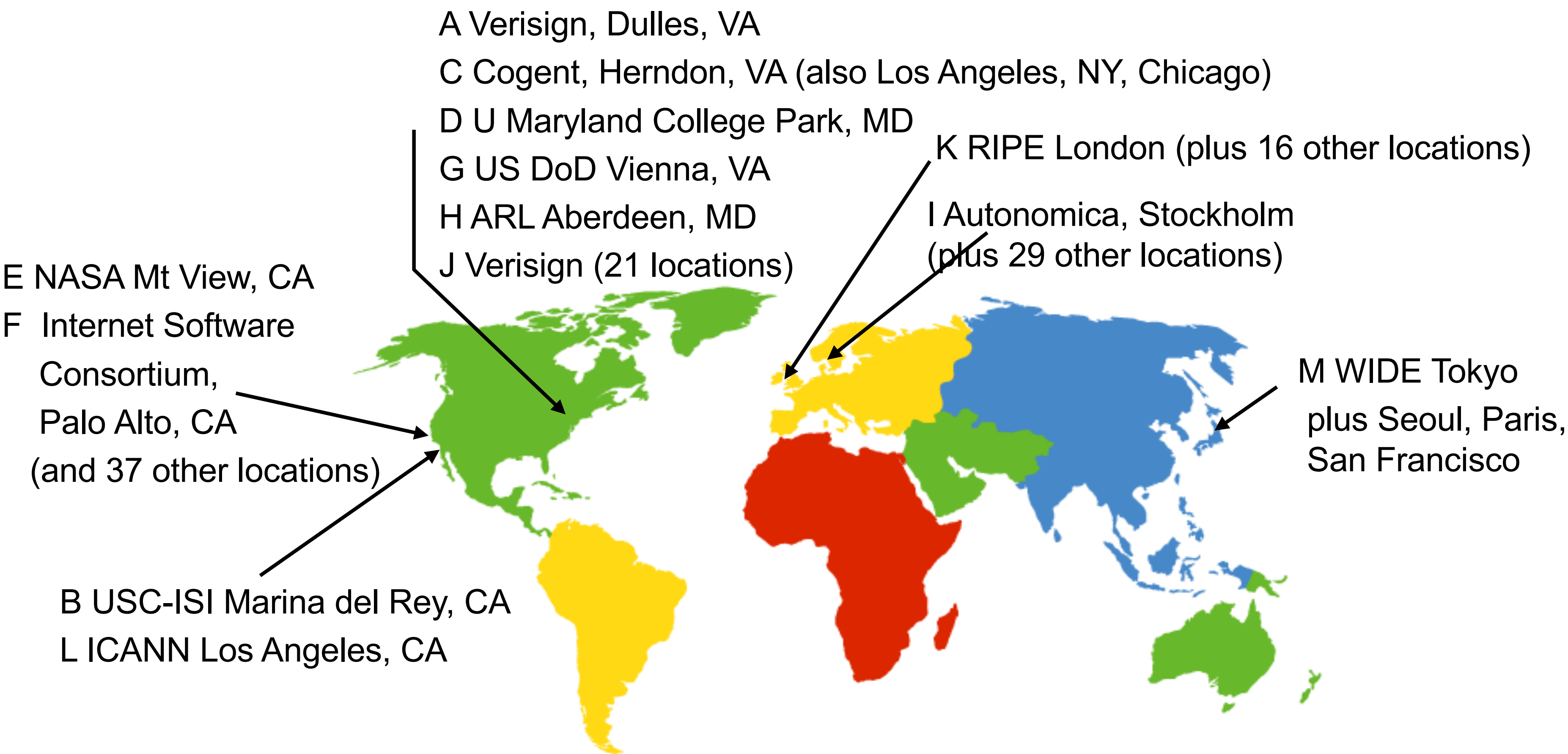
DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers.org/)



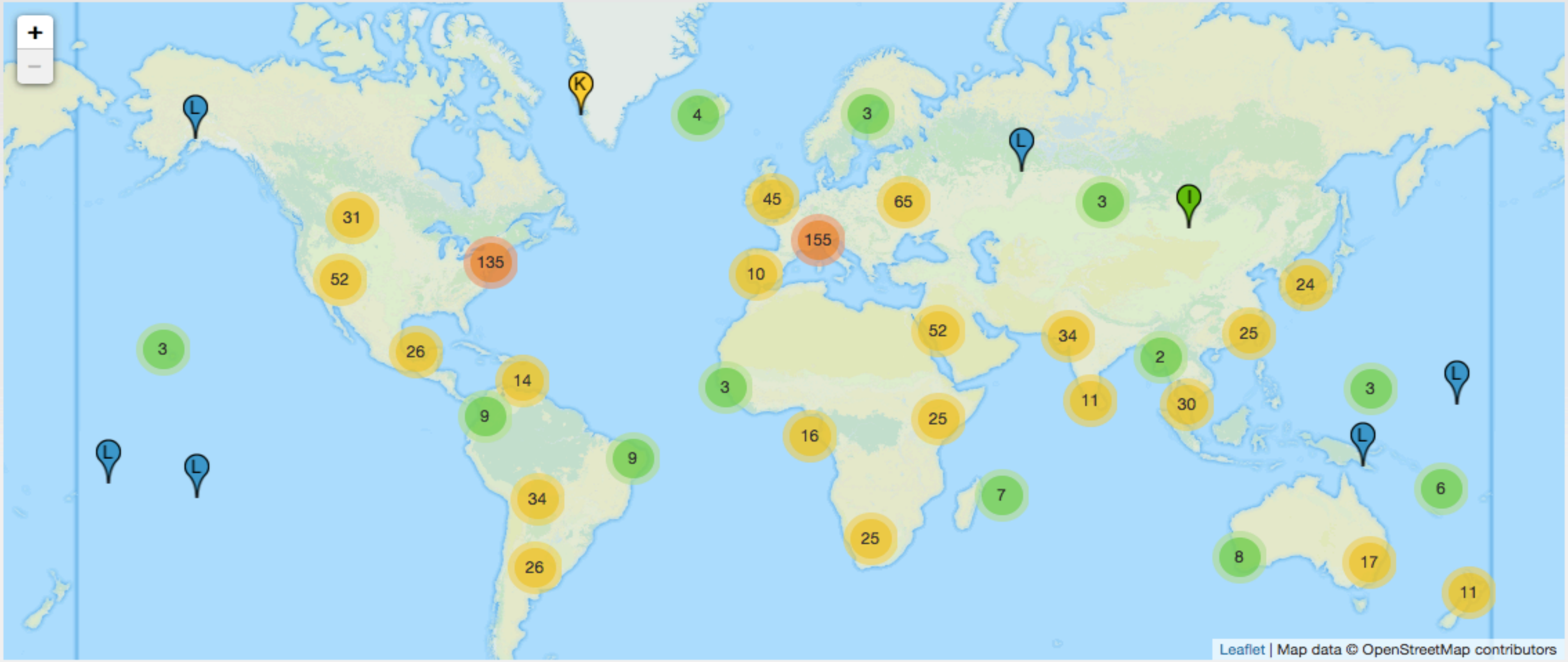
DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers.org/)
- Replicated via any-casting



Root Server health: https://www.ultratools.com/tools/dnsRootServerSpeed

DNS: root name servers



As of 2018-08-01, the root server system consists of 931 instances operated by the 12 independent root server operators.

www.root-servers.org

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

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

Local DNS name server

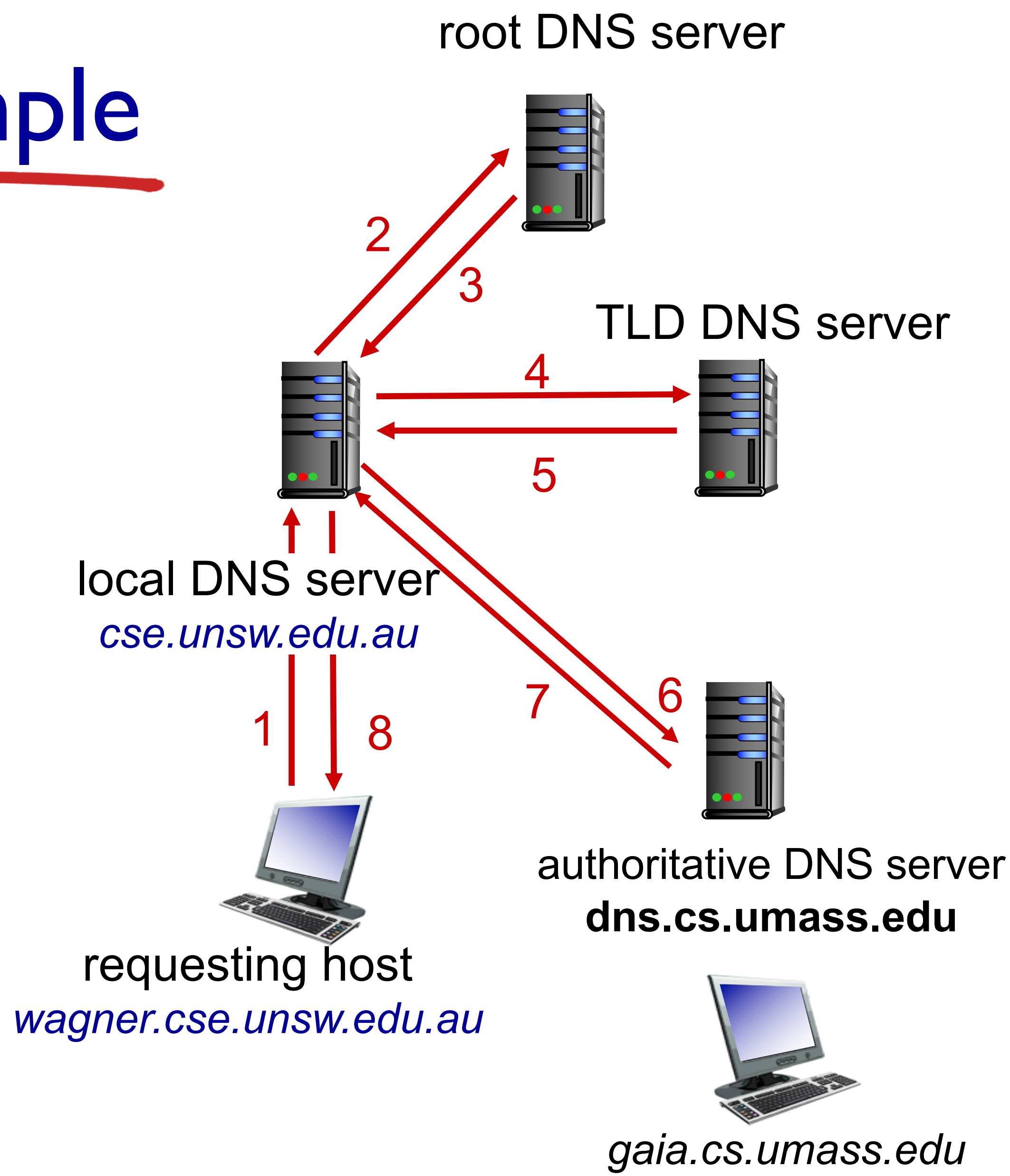
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- Hosts configured with local DNS server address (e.g., /etc/resolv.conf) or learn server via a host configuration protocol (e.g., DHCP)
- Client application
  - Obtain DNS name (e.g., from URL)
  - Do **gethostbyname()** to trigger DNS request to its local DNS 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

- host at wagner.cse.unsw.edu.au wants IP address for gaia.cs.umass.edu

iterated query:

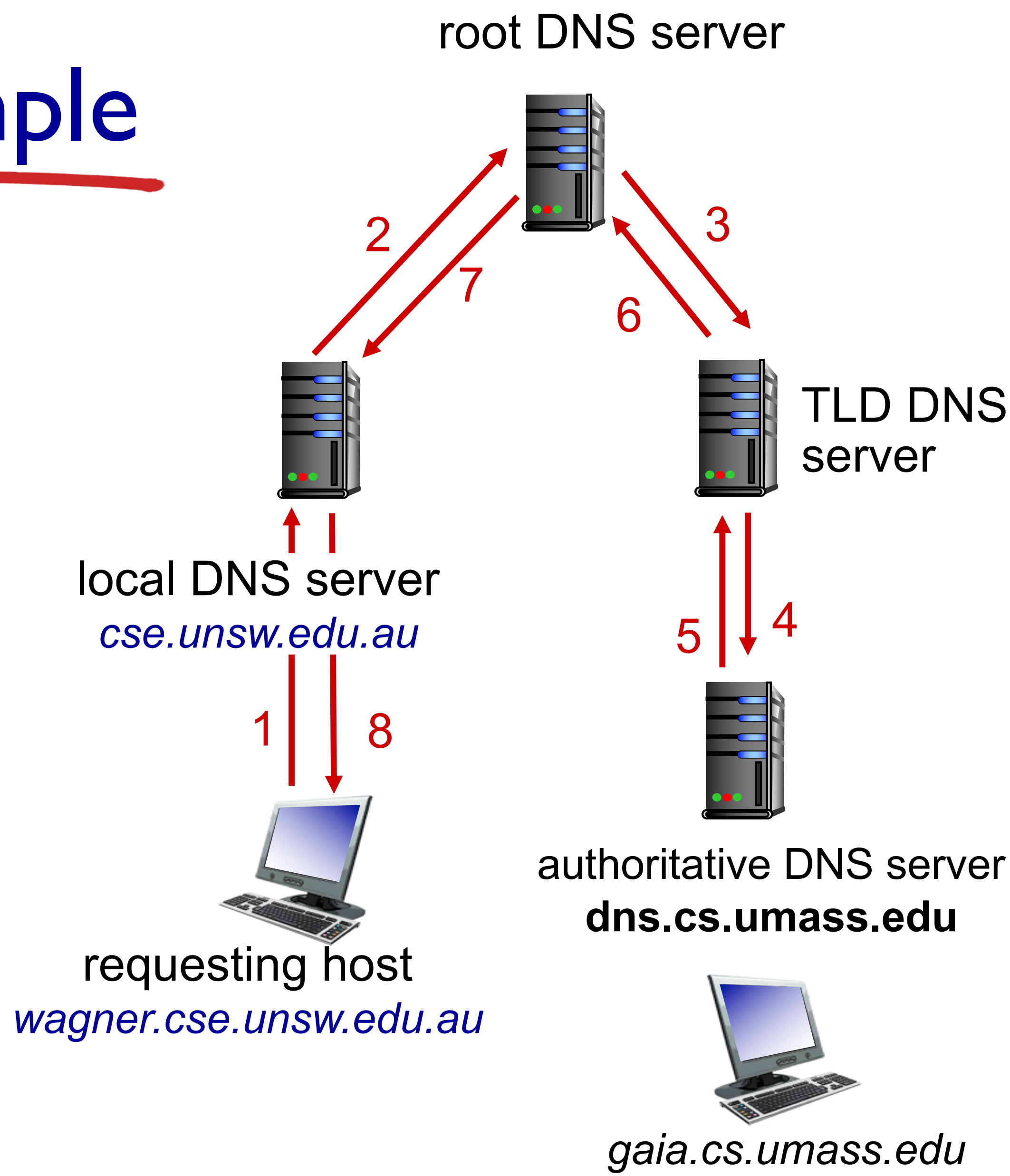
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- Subsequent requests need not burden DNS
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire

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 protocol, messages

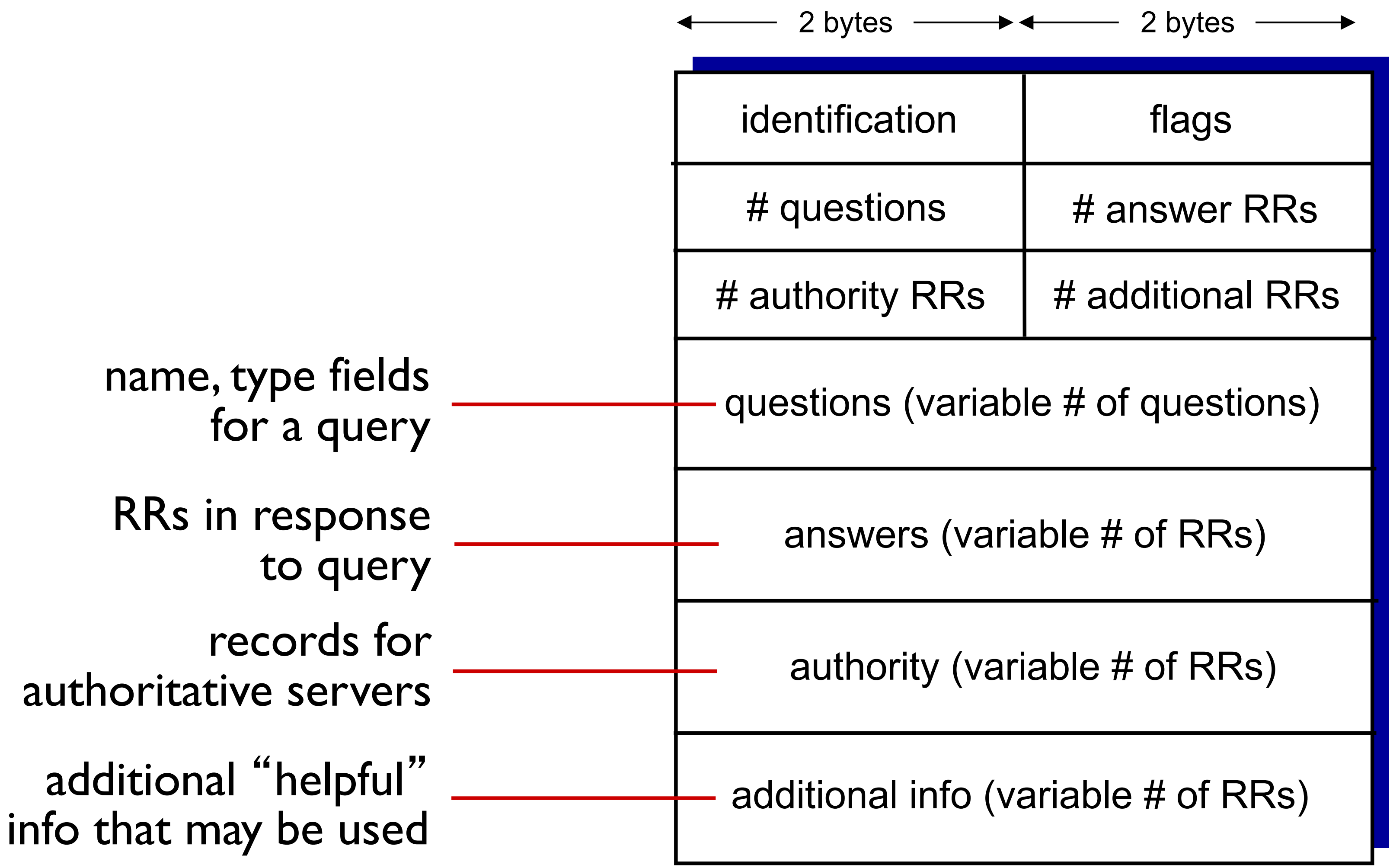
- *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

DNS protocol, messages



An Example

Try this out yourself. Part of one of the lab

```
bash-3.2$ dig www.oxford.ac.uk

; <<>> DiG 9.8.3-P1 <<>> www.oxford.ac.uk
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35102
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 5

;; QUESTION SECTION:
;www.oxford.ac.uk.              IN      A

;; ANSWER SECTION:
www.oxford.ac.uk.       300     IN      A       129.67.242.154
www.oxford.ac.uk.       300     IN      A       129.67.242.155

;; AUTHORITY SECTION:
oxford.ac.uk.           86399   IN      NS      dns2.ox.ac.uk.
oxford.ac.uk.           86399   IN      NS      dns1.ox.ac.uk.
oxford.ac.uk.           86399   IN      NS      ns2.ja.net.
oxford.ac.uk.           86399   IN      NS      dns0.ox.ac.uk.

;; ADDITIONAL SECTION:
ns2.ja.net.             33560   IN      A       193.63.105.17
ns2.ja.net.             33560   IN      AAAA    2001:630:0:45::11
dns0.ox.ac.uk.          48090   IN      A       129.67.1.190
dns1.ox.ac.uk.          86399   IN      A       129.67.1.191
dns2.ox.ac.uk.          54339   IN      A       163.1.2.190

;; Query time: 589 msec
;; SERVER: 129.94.172.11#53(129.94.172.11)
;; WHEN: Thu Mar  9 17:53:52 2017
;; MSG SIZE  rcvd: 242
```

Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkutopia.com at *DNS registrar* (e.g., Network Solutions)
  - 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)`
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com
- Q: Where do you insert these type A and type MX records?

  A: ??

Reliability

- DNS servers are replicated (primary/secondary)
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas
- Usually, UDP used for queries
  - Need reliability: must implement this on top of UDP
  - Spec supports TCP too, but not always implemented
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

DNS provides Indirection

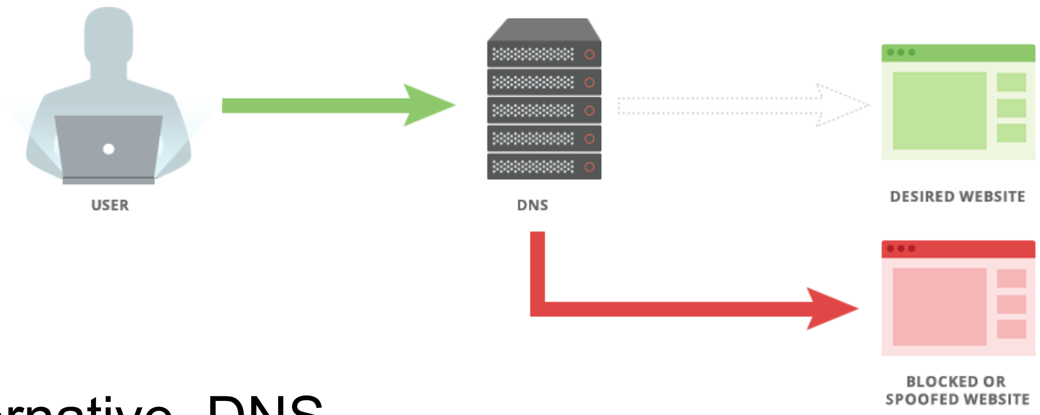
- Addresses can change underneath
  - Move www.cnn.com to 4.125.91.21
  - Humans/Apps should be unaffected
- Name could map to multiple IP addresses
  - Enables
    - Load-balancing
    - Reducing latency by picking nearby servers
- Multiple names for the same address
  - E.g., many services (mail, www, ftp) on same machine
  - E.g., aliases like www.cnn.com and cnn.com
- But, this flexibility applies only within domain!

Reverse DNS

- IP address -> domain name
- Special PTR record type to store reverse DNS entries
- Where is reverse DNS used?
  - Troubleshooting tools such as traceroute and ping
  - “Received” trace header field in SMTP e-mail
  - SMTP servers for validating IP addresses of originating servers
  - Internet forums tracking users
  - System logging or monitoring tools
  - Used in load balancing servers/content distribution to determine location of requester

Do you trust your DNS server?

- Censorship



https://wikileaks.org/wiki/Alternative_DNS

- Logging
  - IP address, websites visited, geolocation data and more
  - E.g., Google DNS:

    https://developers.google.com/speed/public-dns/privacy

Attacking DNS

DDoS attacks

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

Redirect attacks

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

Exploit DNS for DDoS

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

Want to dig deeper?

http://www.networkworld.com/article/2886283/security0/top-10-dns-attacks-likely-to-infiltrate-your-network.html

Schneier on Security

Blog | Newsletter | Books | Essays | News | Talks | Academic | About Me

Blog >

IoT Attack Against a University Network

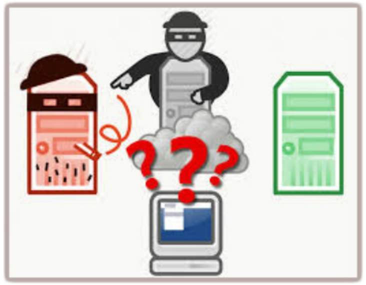
Verizon's *Data Brief Digest 2017* describes an attack against an unnamed university by attackers who hacked a variety of IoT devices and had them spam network targets and slow them down:

> Analysis of the university firewall identified over 5,000 devices making hundreds of Domain Name Service (DNS) look-ups every 15 minutes, slowing the institution's entire network and restricting access to the majority of internet services.
>
> In this instance, all of the DNS requests were attempting to look up seafood restaurants -- and it wasn't because thousands of students all had an overwhelming urge to eat fish -- but because devices on the network had been instructed to repeatedly carry out this request.
>
> "We identified that this was coming from their IoT network, their vending machines and their light sensors were actually looking for seafood domains; 5,000 discreet systems and they were nearly all in the IoT infrastructure," says Laurance Dine, managing principal of investigative response at Verizon.

The actual Verizon document doesn't appear to be available online yet, but there is an advance version that only discusses the incident above, available here.

Detailed Report at - http://www.verizonenterprise.com/resources/reports/rp_data-breach-digest-2017-sneak-peek_xg_en.pdf

DNS Cache Poisoning



- Suppose you are a bad guy  and you control the name server for drevil.com. Your name server receives a request to resolve www.drevil.com. and you respond as follows:

```
;; QUESTION SECTION:
;www.drevil.com.          IN     A

;; ANSWER SECTION:
www.drevil.com      300    IN     A   129.45.212.42

;; AUTHORITY SECTION:
drevil.com     86400 IN    NS     dns1.drevil.com.
drevil.com     86400 IN    NS     google.com

;; ADDITIONAL SECTION:
google.com  600 IN A  129.45.212.222
```

A drevil.com machine, **not** google.com

- Solution: Do not allow DNS servers to cache IP address mappings unless they are from authoritative name servers

Dig deeper?

DNS Cache Poisoning Test
https://www.grc.com/dns/dns.htm

DNSSEC: DNS Security Extensions,
http://www.dnssec.net

Quiz: DNS

- If a name server has no clue about where to find the address for a hostname then

A. Server asks the authoritative name server

B. Server asks its root name server

C. Request is not processed

D. Server asks another name server in its domain

Quiz: DNS

- Which of the following is an example of a Top Level Domain?

  A. yoda.jedi.starwars.com

  B. jedi.starwars.com

  C. starwars.com

  D. .com

Quiz: DNS

- A web browser needs to contact www.cse.unsw.edu.au. The minimum number of DNS requests sent is:

A. 0

B. 1

C. 2

D. 3

Application Layer: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

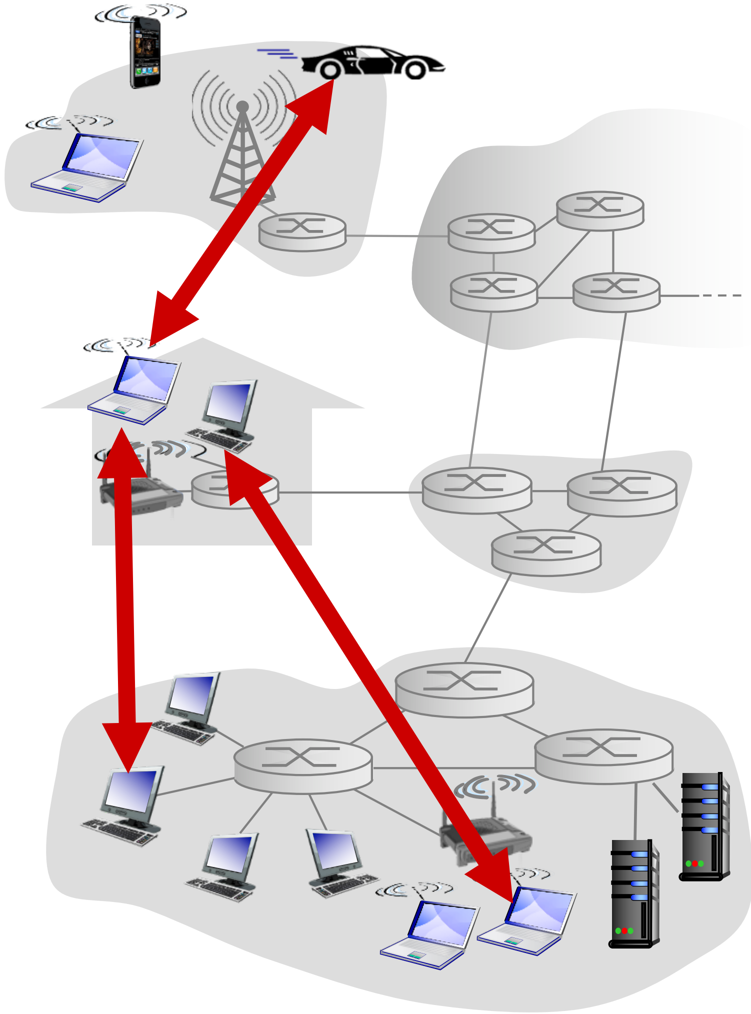
Self study

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

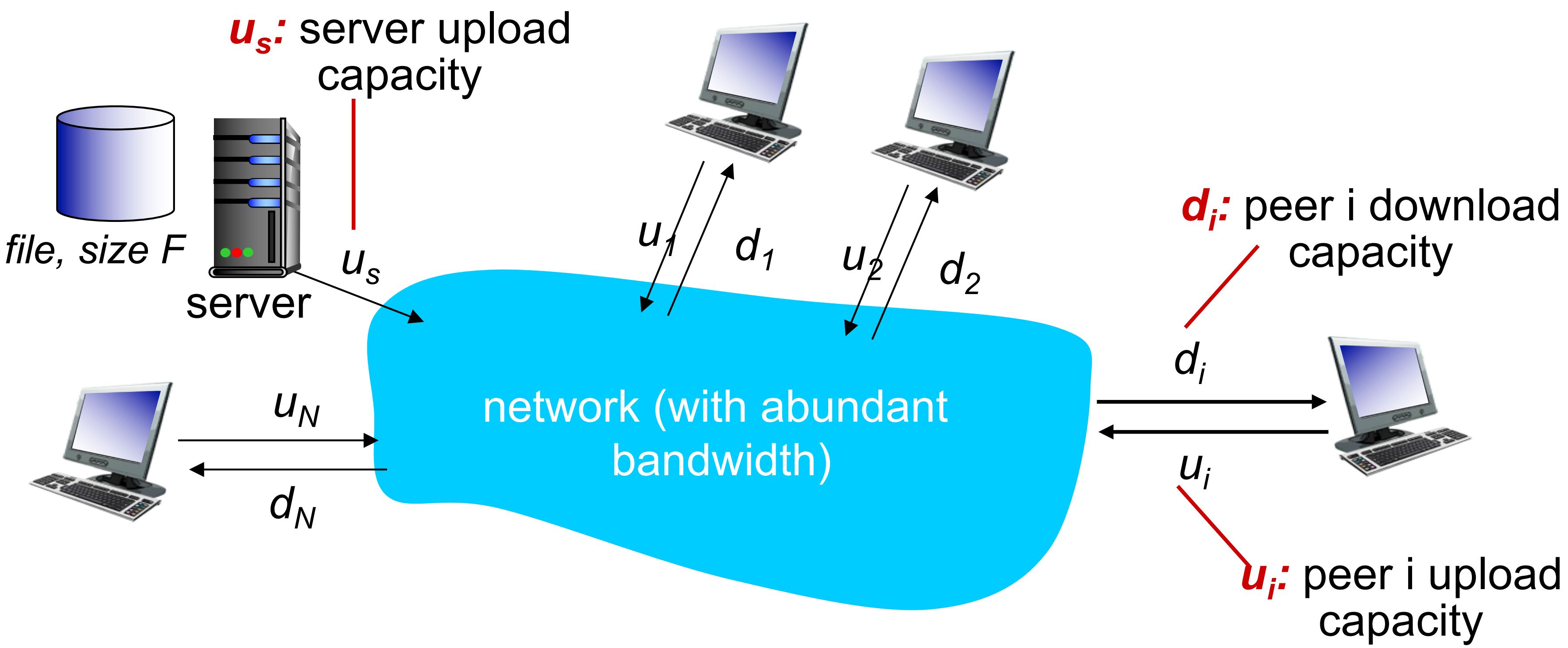
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



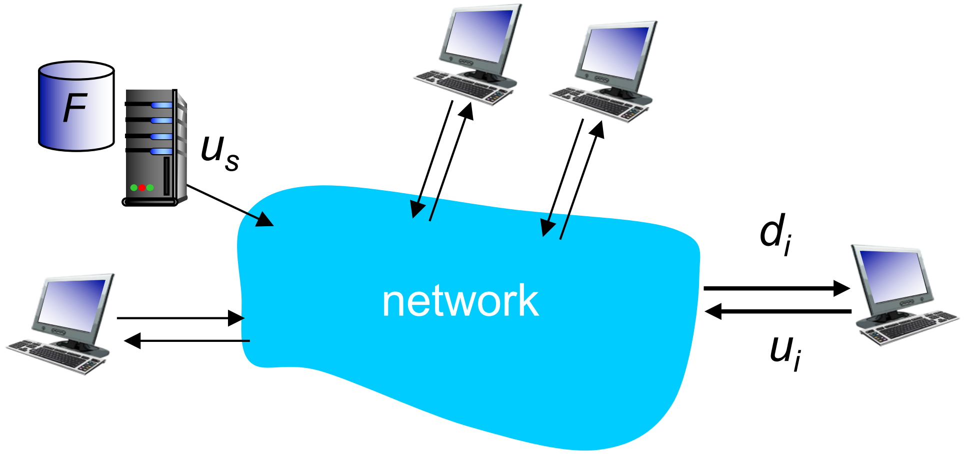
File distribution: client-server vs P2P

Question: how much time to distribute file (size F) from one server to N peers?

- peer upload/download capacity is limited resource



File distribution time: client-server



- *server transmission:* must send (upload) N file copies:
  - time to send one copy: F/u_s
  - time to send N copies: NF/u_s

- *client:* each client must download file copy
  - d_{min} = min client download rate
  - client download time: F/d_{min}

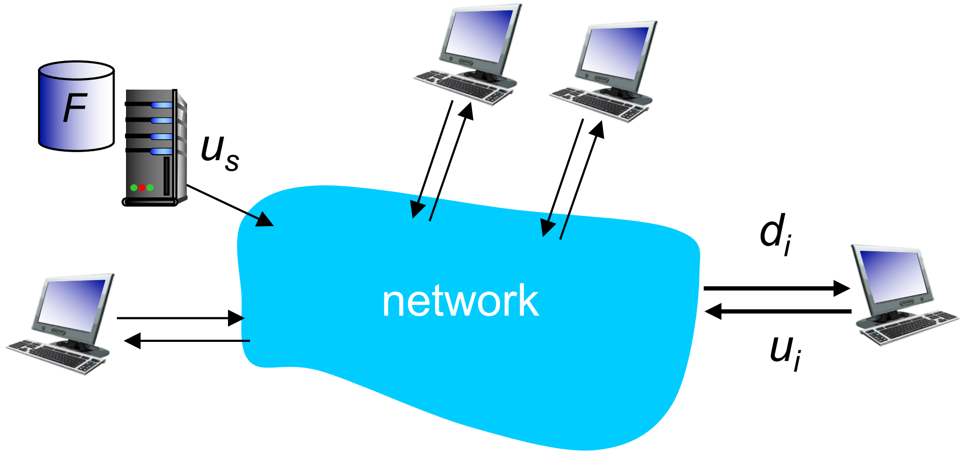
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_{s},F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - client download time: F/d_{min}



- *clients:* as aggregate must download *NF* bits
  - max upload rate (limiting max download rate) is $u_s + \sum u_i$

time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \sum_{i=1}^{N} u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$