ELEC / COMP 177 – Fall 2013

Computer Networking
➔ Email (SMTP, POP, IMAP)
Domain Name System (DNS)

Some slides from Kurose and Ross, *Computer Networking*, 5th Edition

Upcoming Schedule

- **Project 1 – Python HTTP Server**
  - Work day: Next Tuesday (Sept 24th)
  - **Due Thursday, September 26th by 11:55pm**
  - **Questions?**

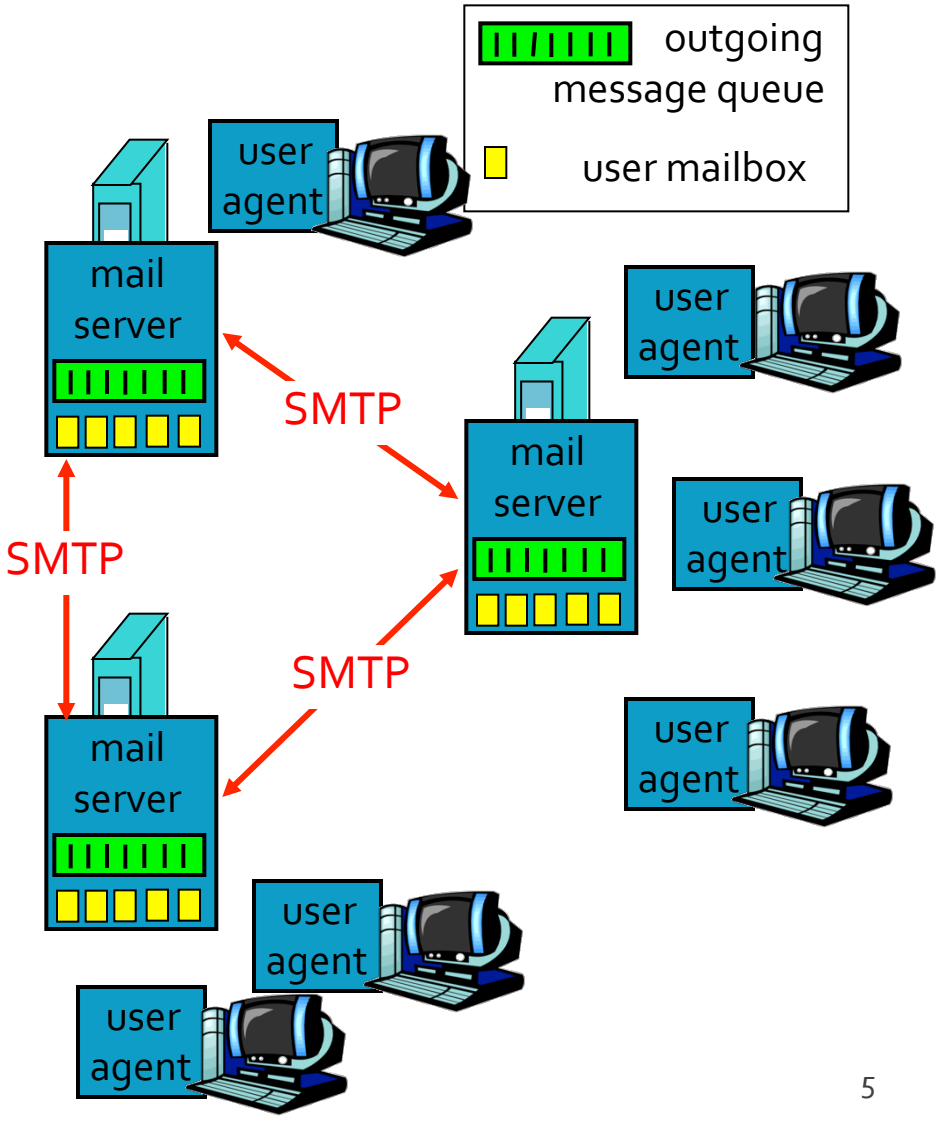
Upcoming Schedule

- **Presentation 1** – Application-Layer Protocol
  - Discuss requirements…
  - Topic Approval – **Due next Tuesday** (Sept 24th)
    - Email by start of class time
  - Presentations – **Oct 1st and Oct 3rd**
    - Upload slides to Sakai by midnight before (Sept 30th)

Email (SMTP, POP, IMAP)

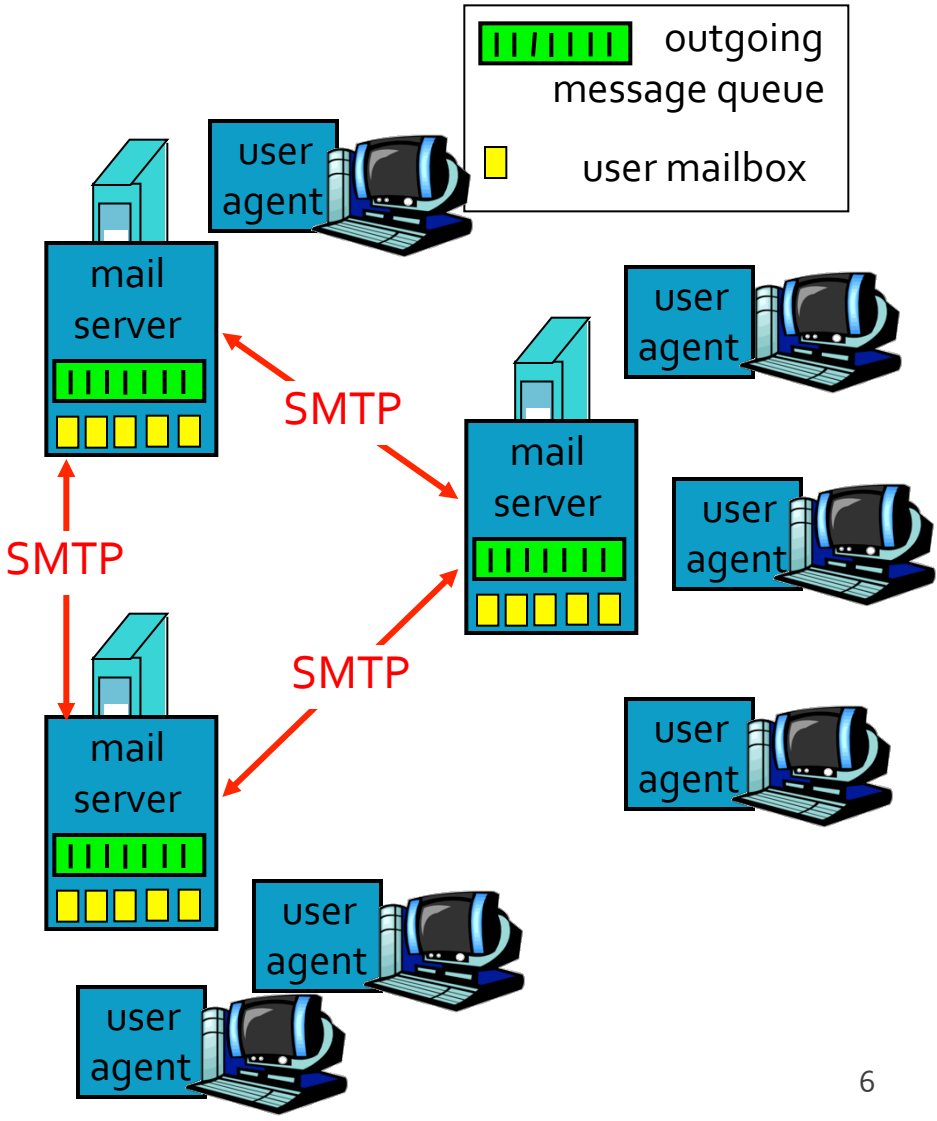
Electronic Mail

- **Three major components**
  - User agents
  - Mail servers
  - Protocol for message transfer (SMTP)
- **User Agent**
  - Your mail reader
    - Composing, editing, reading mail messages
    - e.g., Outlook, Thunderbird, Mail (Mac), …
  - Outgoing and incoming messages stored on server



Electronic Mail – Mail Servers

- **Mail Servers**
  - **Mailbox** contains incoming messages for user
  - **Message queue** of outgoing mail messages (to be sent)
- **SMTP protocol**
  - Used to move email messages between mail servers
  - Client: sending mail server
  - Server: receives messages

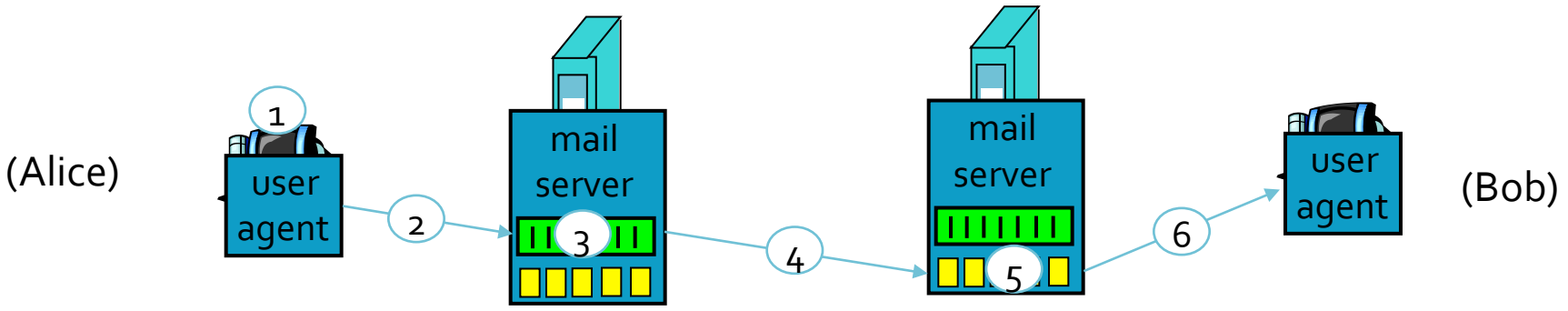


Simple Mail Transport Protocol (SMTP)

- 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
  - **Commands**: ASCII text
  - **Response**: status code and phrase
- Messages must be in 7-bit ASCII
  - Binary attachments are Base64 *encoded*

SMTP Scenario: Alice sends message to Bob

1) Alice uses UA to compose message to `bob@bigschool.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=Server, C=Client

```
S: 220 bigschool.edu
C: HELO smallschool.edu
S: 250  Hello smallschool.edu, pleased to meet you
C: MAIL FROM: <alice@smallschool.edu>
S: 250 alice@smallschool.edu... Sender ok
C: RCPT TO: <bob@bigschool.edu>
S: 250 bob@bigschool.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: This is a test message
C: This is still a test message
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 bigschool.edu closing connection
```

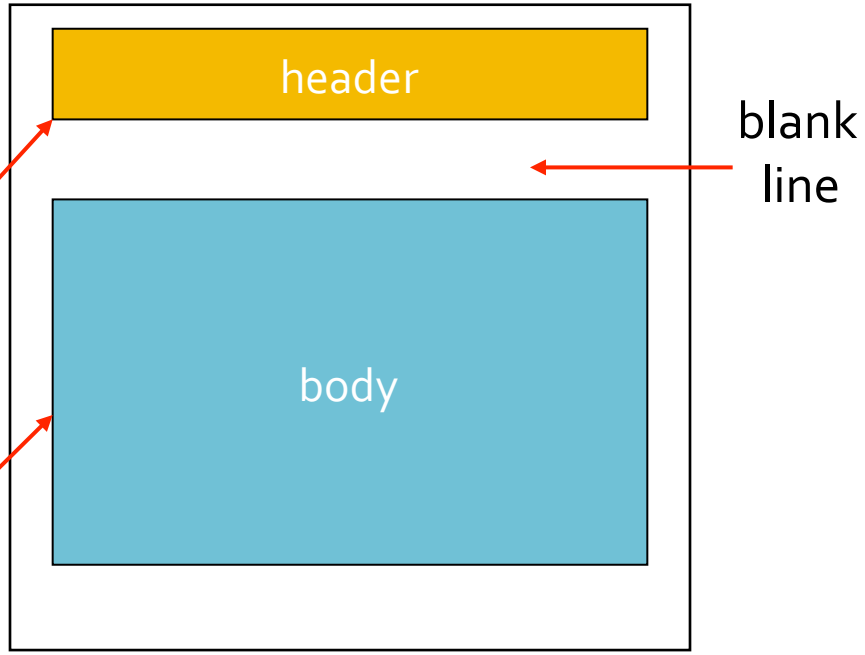
SMTP server uses CRLF.CRLF to determine end of message

SMTP versus HTTP

- "Direction" of transfer
  - HTTP: pull *from* server *(at least, HTTP GET)*
  - SMTP: push *to* server
- Protocol "style"
  - Both have ASCII command/response interaction and status codes
- Granularity
  - HTTP: each object encapsulated in its own response message *(version 1.0 only)*
  - SMTP: multiple objects sent in multipart message

Mail Message format

- SMTP defines exchanging messages between systems *(transport)*
  - It does **not** specify the format for data inside the message! *(content)*
- RFC 822 defines a standard for text message format
- Header lines
  - To / From / Subject / …
  - **Different from SMTP commands!**
- Body
  - The "message"

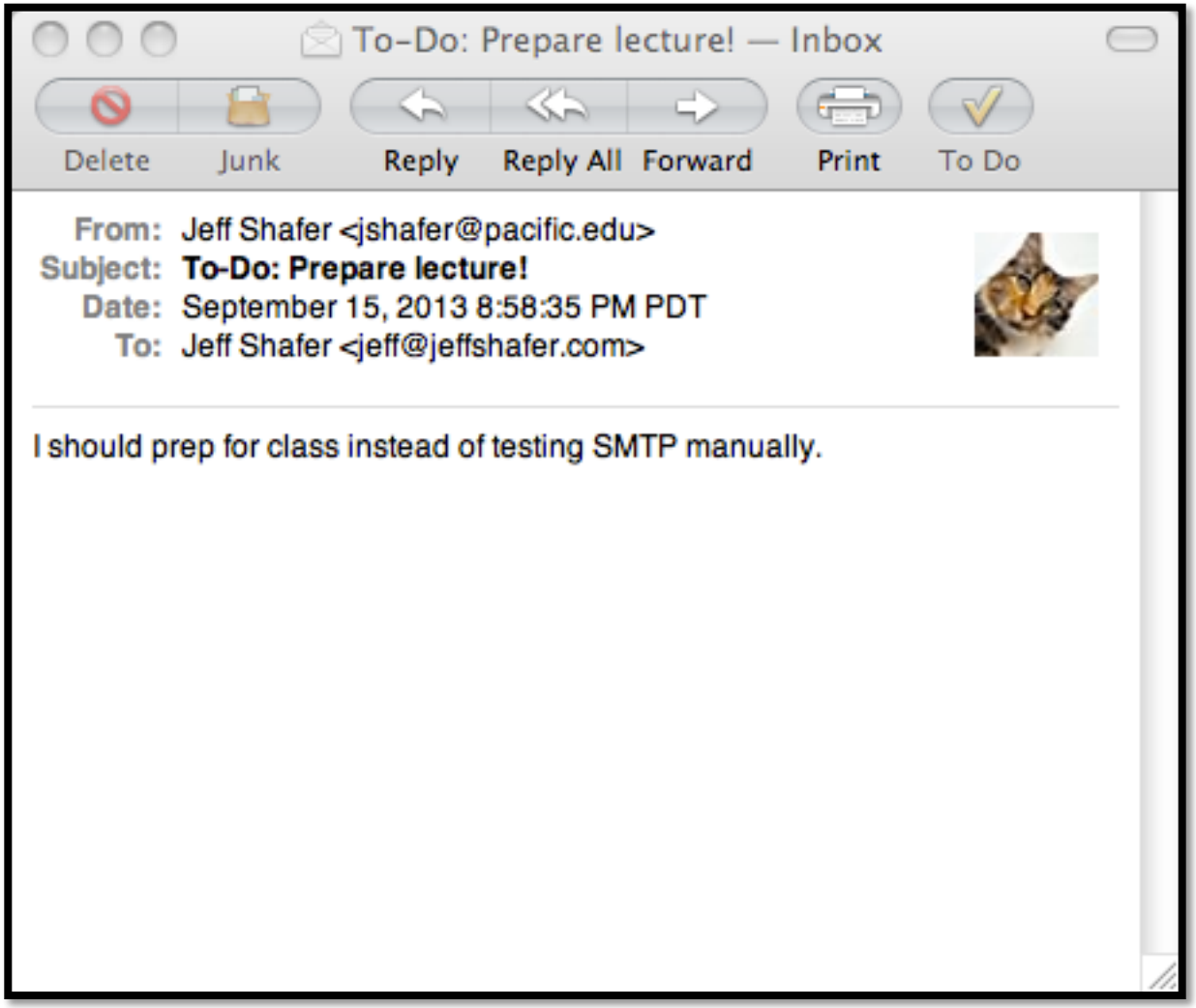


SMTP Manually

```
tiger [~] <!> telnet smtp.pacific.edu 25
Trying 192.168.100.100...
Connected to smtp.pacific.edu.
Escape character is '^]'.
220 mx20.pacific.edu ESMTP
HELO pacific.edu
250 mx20.pacific.edu
MAIL FROM: <jshafer@pacific.edu>
250 2.1.0 Ok
RCPT TO: <jeff@jeffshafer.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: "Jeff Shafer" <jeff@jeffshafer.com>
From: "Jeff Shafer" <jshafer@pacific.edu>
Subject: To-Do: Prepare lecture!

I should prep for class instead of testing SMTP manually.
.
250 2.0.0 Ok: queued as 9BD3478EC
QUIT
221 2.0.0 Bye
```

SMTP Manually – The result!



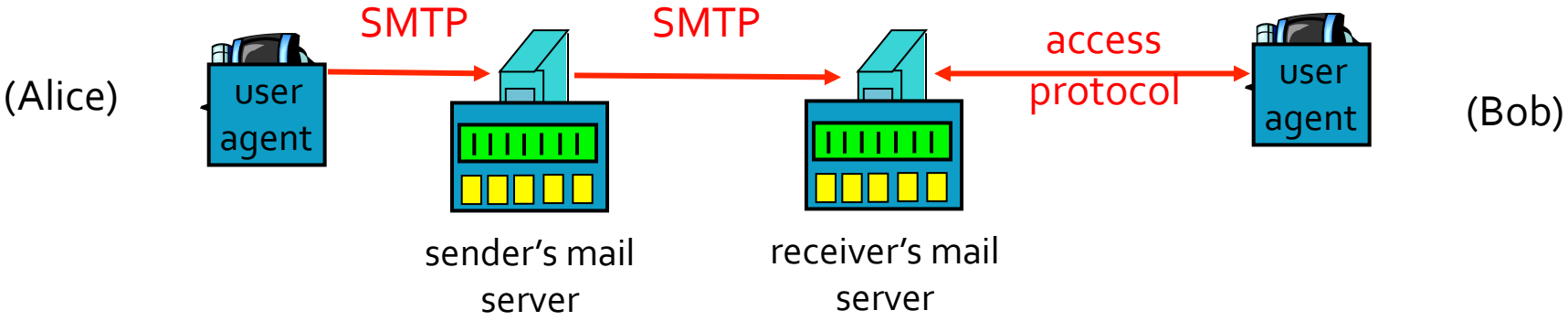
SMTP and SPAM

- Telnet example – did not have to log in!
  - Security an afterthought in original design
- Open relay
  - SMTP server that sends mail to *all* destinations for *all* clients
  - Typically blacklisted today in spam filters
- Optional security measures
  - Only accept clients inside your network?
    - `smtp.pacific.edu` will not respond on port 25 when I'm at home
  - Only accept destinations inside your network?
  - Require users to login? (ESMTP)

SMTP and SPAM

- You can lie to an SMTP server
  - Instead of claiming to be jshafer@pacific.edu, I could have said I was president@pacific.edu
- Countermeasures?
  - `smtp.pacific.edu` could prevent this by forcing me to log on
- What if I send mail via my own SMTP server?
  - Spam filter **challenge**
  - SPF – Sender Protection Framework
    - Puts notes into DNS specifying which IPs are allowed to send mail claiming to be from `pacific.edu`

Mail Access Protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - **POP**: Post Office Protocol
    - Authorization (agent <-->server) and download
  - **IMAP**: Internet Mail Access Protocol
    - More features (more complex)
    - Manipulation of stored messages on server
  - **HTTP**: Gmail, Hotmail, Yahoo! Mail, etc.

Post Office Protocol (POP3)

- Modes:
  - "download and delete from server" mode.
    - Only suitable for 1 email client
  - "Download and keep on server" mode
    - Allows copies of messages on different clients
- POP3 is stateless across sessions

Internet Message Access Protocol (IMAP)

- Keep all messages in one place: the server
  - Clients might have a temporary *cache* for offline access
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - Names of folders and mappings between message IDs and folder name
- Other features
  - Server-side searches (don't have to download mailbox!)
  - Multiple concurrent clients

Domain Name System (DNS)

Motivation

- IP addresses are hard to remember
  - `198.16.253.143`? Or was it `.146`?
- Human-friendly names are much better
  - `engineering.pacific.edu`
- How can we translate between the two?

Early Days (prior to 1983)

- Each computer on the ARPAnet (early Internet) had a single file
  - `hosts.txt` maps all known host names to IP address
- Master list maintained by SRI Network Information Center
  - Email them if your mapping changes
  - New list produced 1-2 times a week
  - All hosts download the new list
- **Problems with this approach?**

Domain Name System (DNS)

- **Distributed database** implemented in hierarchy of many **name servers**
- **Application-layer protocol**
  - Hosts, routers, and name servers communicate to resolve names (address/name translation)
  - Core Internet function, implemented as application-layer protocol
  - Complexity at network's "edge"

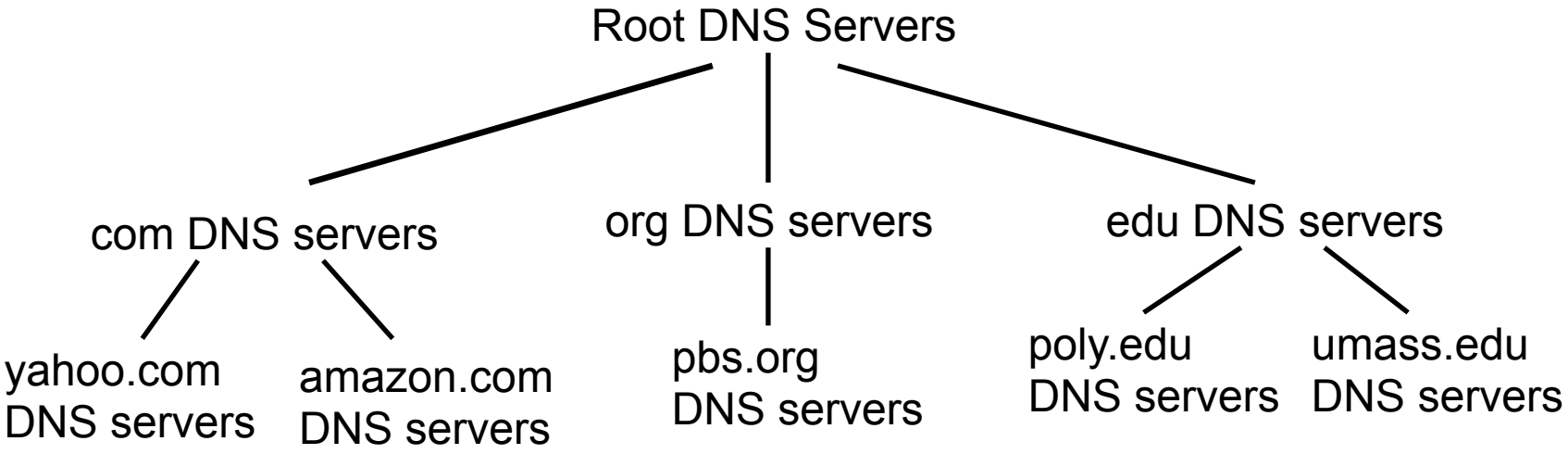
DNS

- **DNS services**
  - Hostname to IP address translation
  - Host aliasing
    - Canonical, alias names
  - Mail server aliasing
  - Load distribution
    - Replicated Web servers: set of IP addresses for one canonical name
- Why not centralize DNS?
  - Single point of failure
  - Traffic volume
  - Distant centralized database
  - Maintenance
  - **Doesn't scale!**

What's in a Name?

- `engineering.pacific.edu`
  - .edu is top-level domain
  - "pacific" belongs to .edu
  - "engineering" belongs to "pacific"
  - Hierarchical! Read from right to left
- Limits?
  - Up to 127 levels of hierarchy
  - Each label can have up to 63 characters
  - Full domain name cannot exceed 253 characters

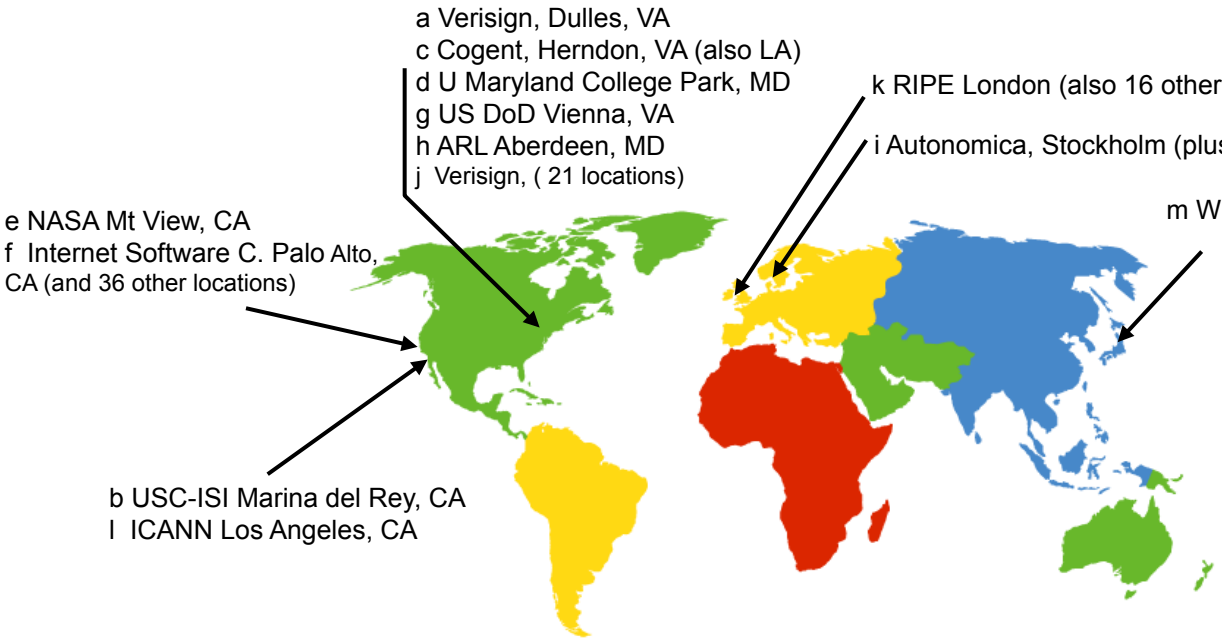
Distributed, Hierarchical Database



- Client wants IP for www.amazon.com
  1. Client queries a root server to find com DNS server
  2. Client queries com DNS server to get amazon.com DNS server
  3. Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: Root name servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server



13 root name "servers" worldwide!

(Each "server" is really a cluster)

(Some clusters are geographically distributed)

TLD and Authoritative Servers

- **Top-level domain** (TLD) servers
  - Responsible for com, org, net, edu,… and all top-level country domains (uk, fr, ca, jp, …)
  - Server maintainers
    - VeriSign for com, net, name TLDs
    - Educause for edu TLD
- **Authoritative** DNS servers:
  - Organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

Local Name Server

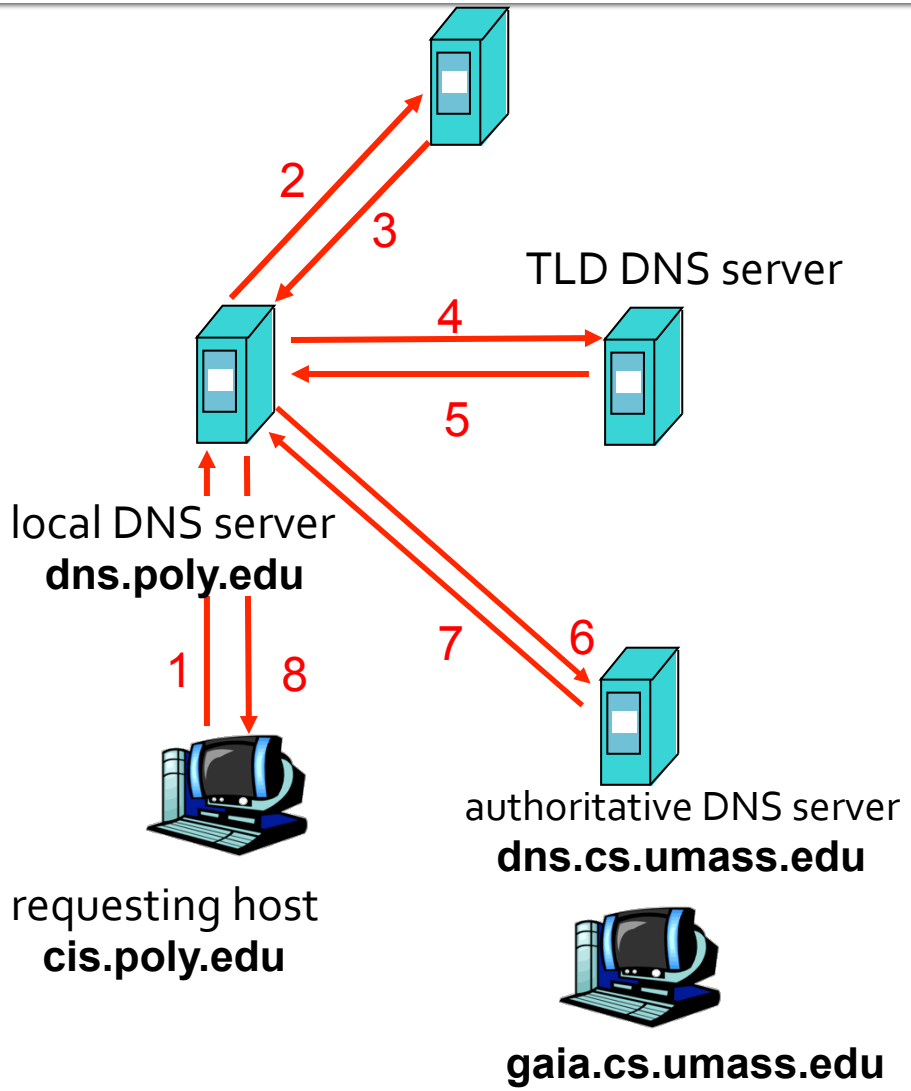
- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
- **You typically know this server's IP address from DHCP**

DNS Name Resolution

- Two types
- **Recursive**
  - The server you contact provides the final answer
  - *Behind the scenes, it may make several consecutive requests*
- **Iterative**
  - The server you contact directs you to a different server to get (closer to) the final answer

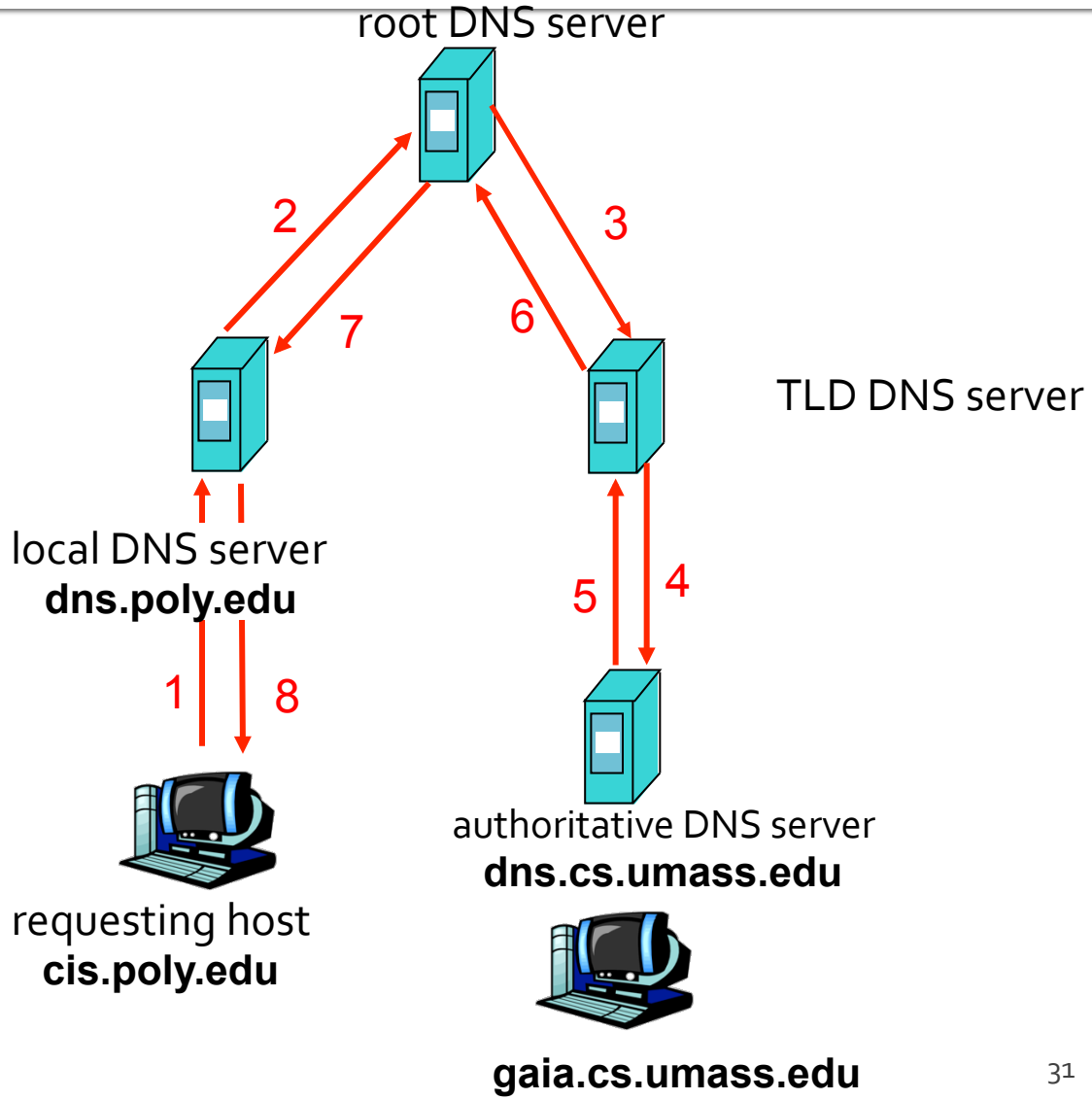
DNS Example

- Host at cis.poly.edu 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 Example

- Recursive query:
  - Puts burden of name resolution on contacted name server
  - Heavy load?



DNS: Caching and Updating records

- Once (any) name server learns mapping, it **caches** mapping
  - This includes your computer
  - This includes your ISP's name server
- Cache entries eventually timeout
  - Can be specified by the authoritative server, and/or overruled by the local server
- TLD (.com, .net, .org, etc…) servers are typically cached in local name servers
  - Reduces traffic on the root servers!

DNS: Distributed DB

Resource Record (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" (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

Multipurpose DNS

- What else do we stuff into DNS records?
  - SPF entries for email
    - Anti-spam
  - MX records for email
    - What are the **multiple** host names that receive mail for this domain?
    - 1st priority, then 2nd backup, then 3rd backup, etc…
    - Allows you to use 3rd party email services (e.g. Google Apps)

DNS and UDP

- DNS uses UDP by default
  - It *can* use TCP, but it's rare
  - **Isn't this unreliable?**
- Why use UDP
  - Faster (in three ways!)
    - No need to establish a connection (RTT/latency overhead)
    - Lower per-packet byte overhead in UDP header
    - Less packet processing by hosts
  - Reliability not needed
    - DNS will just re-request if no response received (2-5 seconds)