

# CS 330: Network Applications & Protocols

Application Layer: FTP, SMTP, DNS

---

Galin Zhelezov  
Department of Physical Sciences  
York College of Pennsylvania

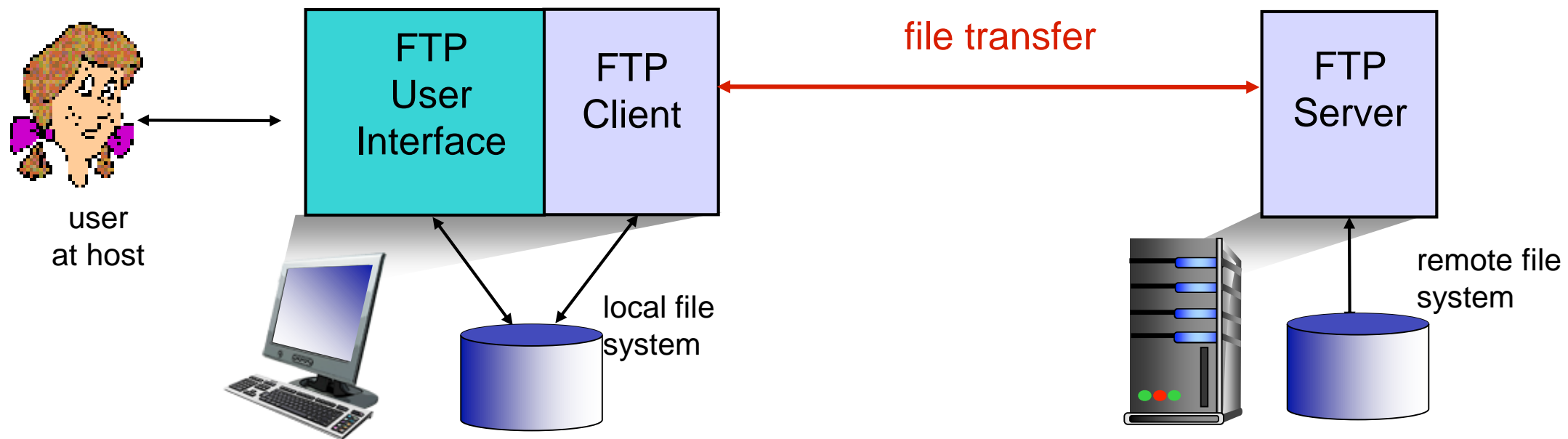


# Overview of Application Layer

---

- **Network Application Architectures**
- **HyperText Transfer Protocol (HTTP)**
- **File Transfer and Email protocols (FTP, SMTP)**
  - FTP
  - SMTP, POP3, IMAP
- **Domain Name System (DNS)**
- **Peer-to-Peer Applications (P2P)**

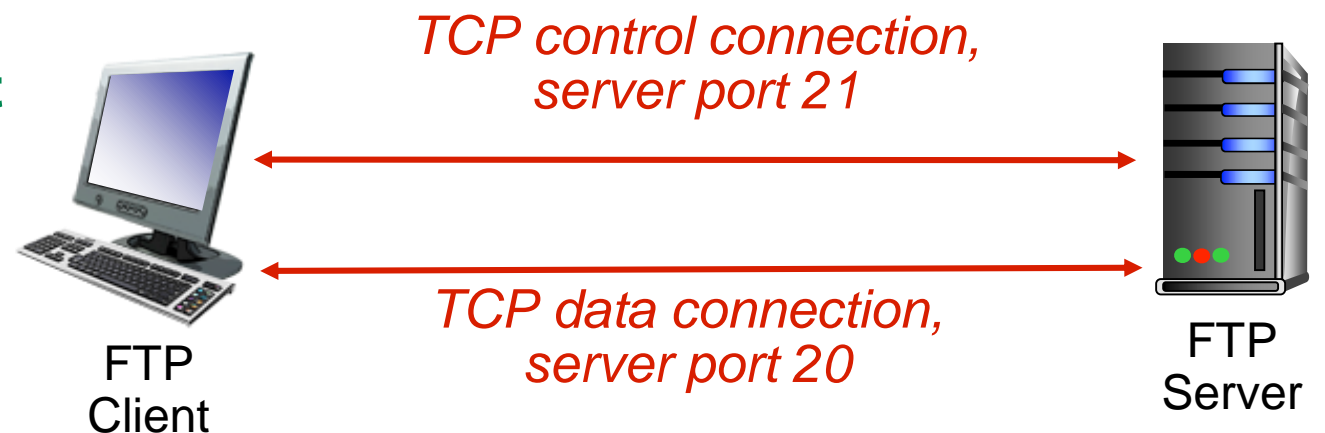
# FTP: File Transfer Protocol



- **Used to transfer files to/from a remote host**
- **Client/server model**
  - Client: side that initiates transfer (either to/from remote)
  - Server: remote host
- **ftp: RFC 959**
- **ftp server: port 21**

# FTP: Separate Control / Data Connections

- FTP client contacts FTP server on port 21, using TCP
- Uses **two parallel TCP connections**: Control and Data
- Client authorized over control connection
- Client browses remote directory, sends commands over control connection
- Control connection is persistent
- FTP **server maintains “state”**: (i.e. current directory, authentication information)



- When server receives file transfer command, server opens 2<sup>nd</sup> TCP data connection (for file) to client
- After transferring one file, server closes data connection
  - A separate TCP data connection is opened for each transferred file

# FTP Commands / Responses

---

- **Commands are sent as plain ASCII text over the control channel**

**USER** username : sends username to server

**PASS** password : sends password to server **in plain text!!**

**LIST** : returns a list of files in current directory

**RETR** filename : retrieves (gets) file

**STOR** filename : stores (puts) file onto remote host

- **FTP server responds with status codes on the control channel**

331 Username OK, password required

125 data connection already open; transfer starting

425 Can't open data connection

452 Error writing file

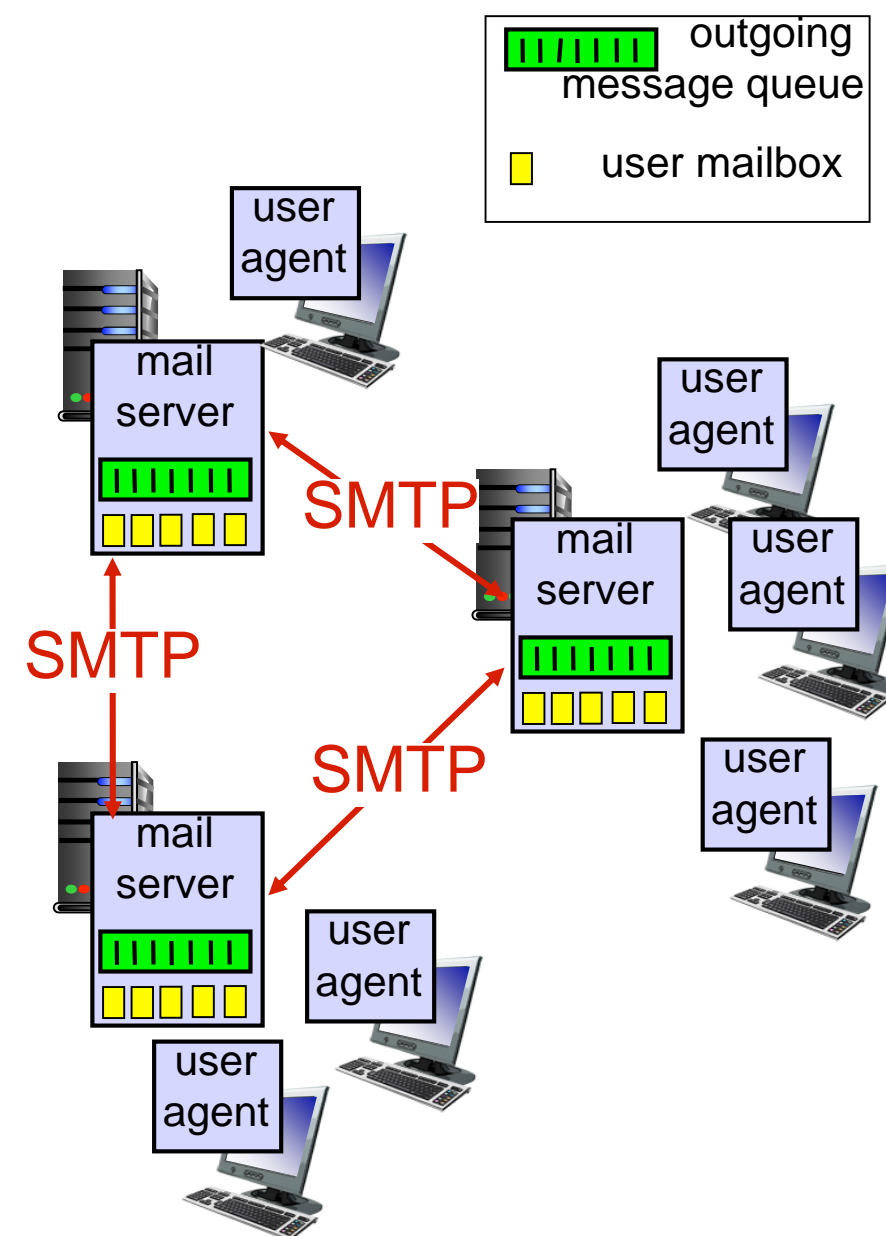
# Electronic Mail

- **Three major components:**

- User agents
- Mail servers
- Simple mail transfer protocol: SMTP

- **User Agent**

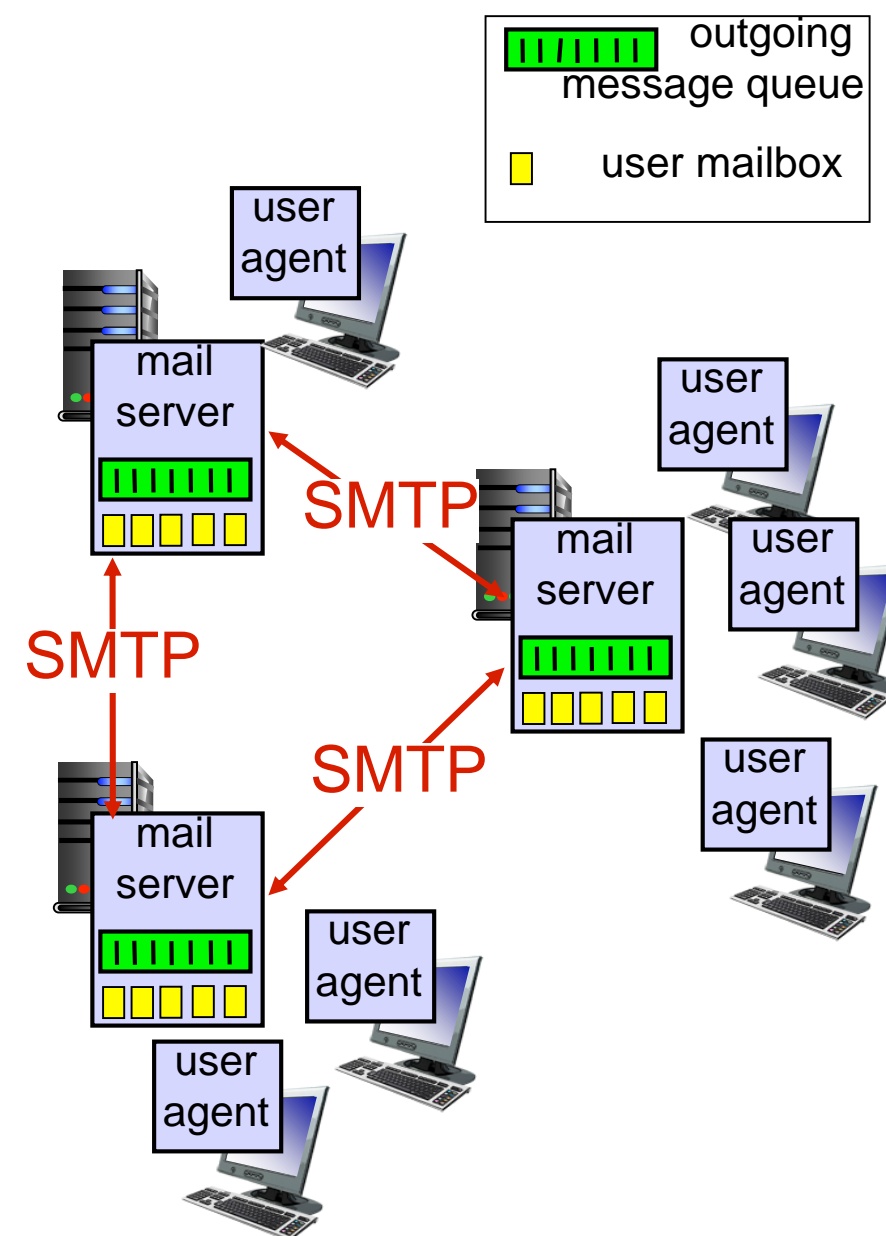
- Composing, editing, reading mail messages
- e.g. Outlook, Thunderbird, iPhone mail client
- Outgoing, incoming messages stored on server



# Electronic Mail: Mail Servers

- **Mail servers:**

- **Mailbox** contains incoming messages for user
- **Message queue** of outgoing (to be sent) mail messages
- Uses **SMTP protocol** between mail servers to send email messages
  - **Client:** sending mail server
  - **“Server”:** receiving mail server



# Electronic Mail: SMTP

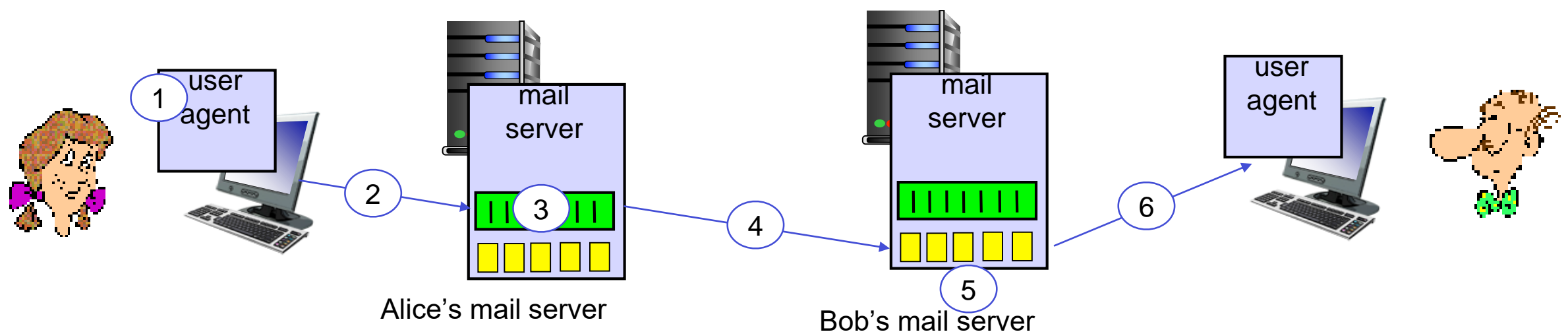
---

- **Uses TCP to reliably transfer email message from client to server on 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 are in ASCII text
  - Server responds with status code and phrase
- **Messages must be in 7-bit ASCII**
  - All binary objects (i.e. attachments **MUST** be converted to ASCII to send)



# Scenario: Alice Sends Message to Bob

- (1) Alice uses her mail client to compose message “to” bob@some school.edu
- (2) Alice’s mail client sends her message to her mail server; the message is placed in a 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 mail client to read message



# Sample SMTP Interaction

---

C: `telnet smtp.fakeplace.edu 25`

S: `220 fakeplace.edu`

C: `HELO ycp.edu`

S: `250 Hello ycp.edu, pleased to meet you`

C: `MAIL FROM: alice@ycp.edu`

S: `250 alice@ycp.edu... Sender ok`

C: `RCPT TO: bob@fakeplace.edu`

S: `250 bob@fakeplace.edu ... Recipient ok`

C: `DATA`

S: `354 Enter mail, end with "." on a line by itself`

C: `This is a test email.`

`More testing.`

`.`

S: `250 Message accepted for delivery`

C: `QUIT`

S: `221 fakeplace.edu closing connection`

# SMTP: Final Words

---

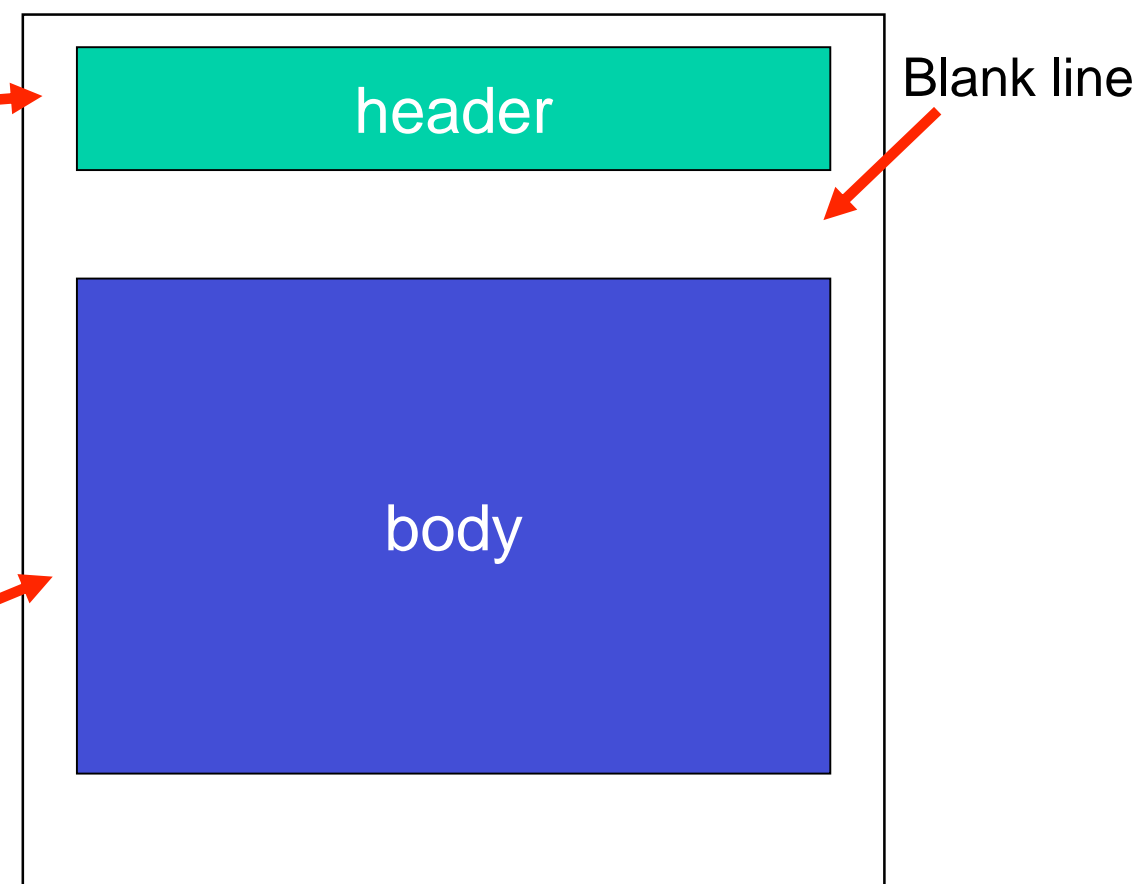
- SMTP uses **persistent connections**
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses **CRLF . CRLF** to determine end of message
- Comparison with HTTP:

HTTP	SMTP
Persistent/Non-persistent connections	Persistent connections
<b>Pulls</b> data from server	<b>Pushes</b> data to server
Accepts binary objects	Accepts only 7-bit ASCII
Each object in its own response msg	Multiple objects sent in multipart msg

# Mail Message Format

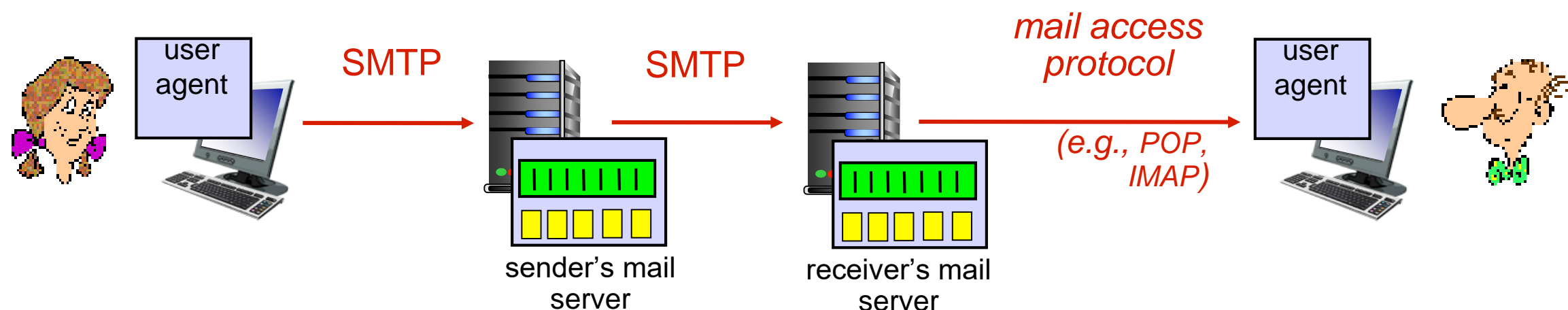
---

- **SMTP: protocol for exchanging email msgs**
- **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



# Mail Access Protocols

- **SMTP** - used for delivery/storage of message to receiver's mail server
- **Mail access protocols used to retrieve messages from mail 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**: gmail, Hotmail, Yahoo! Mail, etc.



# POP3 Protocol

- Authorization phase

- Client commands:

- `user` : declare username

- `pass` : password

- Server responses

- `+OK`

- `-ERR`

- Transaction phase

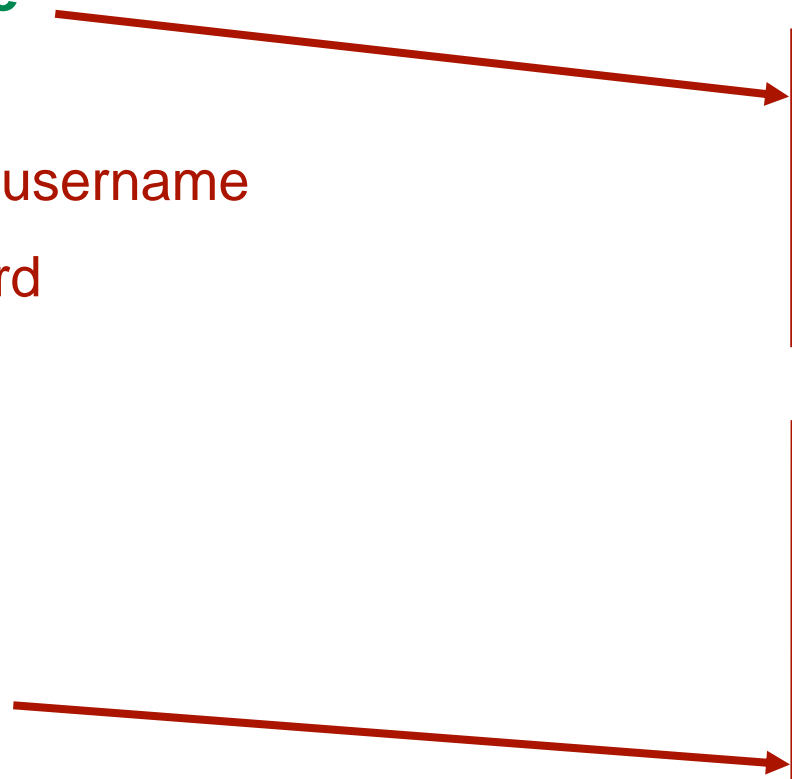
- Client commands:

- `list` : list message numbers


- `retr` : retrieve message by number

- `dele` : delete message

- `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
```

# IMAP

---

- **Internet Mail Access Protocol**
- **More sophisticated than POP3 (POP3 is stateless across sessions)**
- **Allows user to organize messages in folders**
- **Messages can be moved from one folder to another**
- **Users can get only headers or other components of the message**

# Overview of Application Layer

---

- **Network Application Architectures**
- **HyperText Transfer Protocol (HTTP)**
- **File Transfer and Email protocols (FTP, SMTP)**
- **Domain Name System (DNS)**
  - Function
  - Distributed Structure
  - DNS Caching
  - DNS Records
  - DNS Vulnerabilities
- **Peer-to-Peer Applications (P2P)**



# DNS: Domain Name System

---

- **DNS servers translate a host name to IP address**
  - e.g. `www.ycp.edu` → `192.245.87.37`
  - Would be painful to browse Internet and remember IP addresses
- **Hosts and name servers communicate to resolve names**
  - address → name translation
- **Distributed database of all hosts in the universe**
  - Avoids single point of failure
  - Distributes name resolution traffic
  - Geographically distributed
  - Easier to maintain
- **Often used by other application-layer protocols (e.g. SMTP, HTTP, FTP) to translate hostnames to IP addresses**

# Other Services Provided By DNS

---

- **Host aliasing**

- Provides canonical name when alias name is provided
- `www.gmail.com` → `googlemail.l.google.com`

- **Mail server aliasing**

- **Load distribution**

- Replicated web servers, many IP addresses correspond to one name

# DNS Example

---

```
> nslookup www.ycp.edu
Server: 192.168.1.1          <-- my name server
Address: 192.168.1.1#53
```

```
Non-authoritative answer:
www.ycp.edu canonical name = calypso.ycp.edu.
Name: calypso.ycp.edu
Address: 192.245.87.37
```

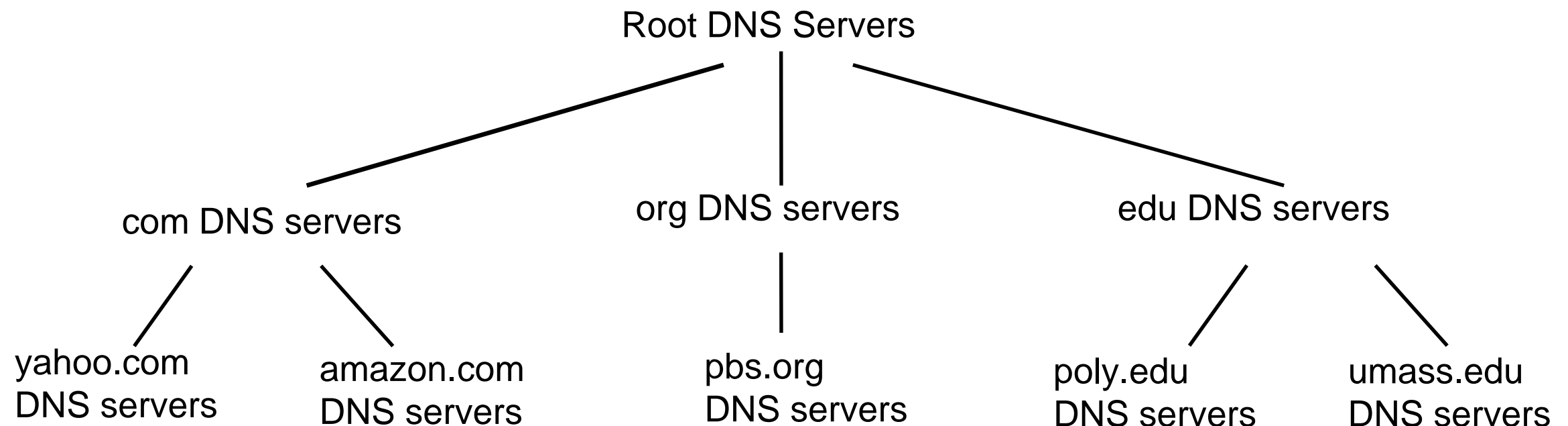
```
> nslookup www.google.com
Server: 192.168.1.1
Address: 192.168.1.1#53
```

```
Non-authoritative answer:
Name: www.google.com
Address: 173.194.73.106
Name: www.google.com
Address: 173.194.73.147
Name: www.google.com
Address: 173.194.73.103
Name: www.google.com
Address: 173.194.73.105
Name: www.google.com
Address: 173.194.73.99
Name: www.google.com
Address: 173.194.73.104
```

<-- Many servers

# DNS: A Distributed, Hierarchical Database

---

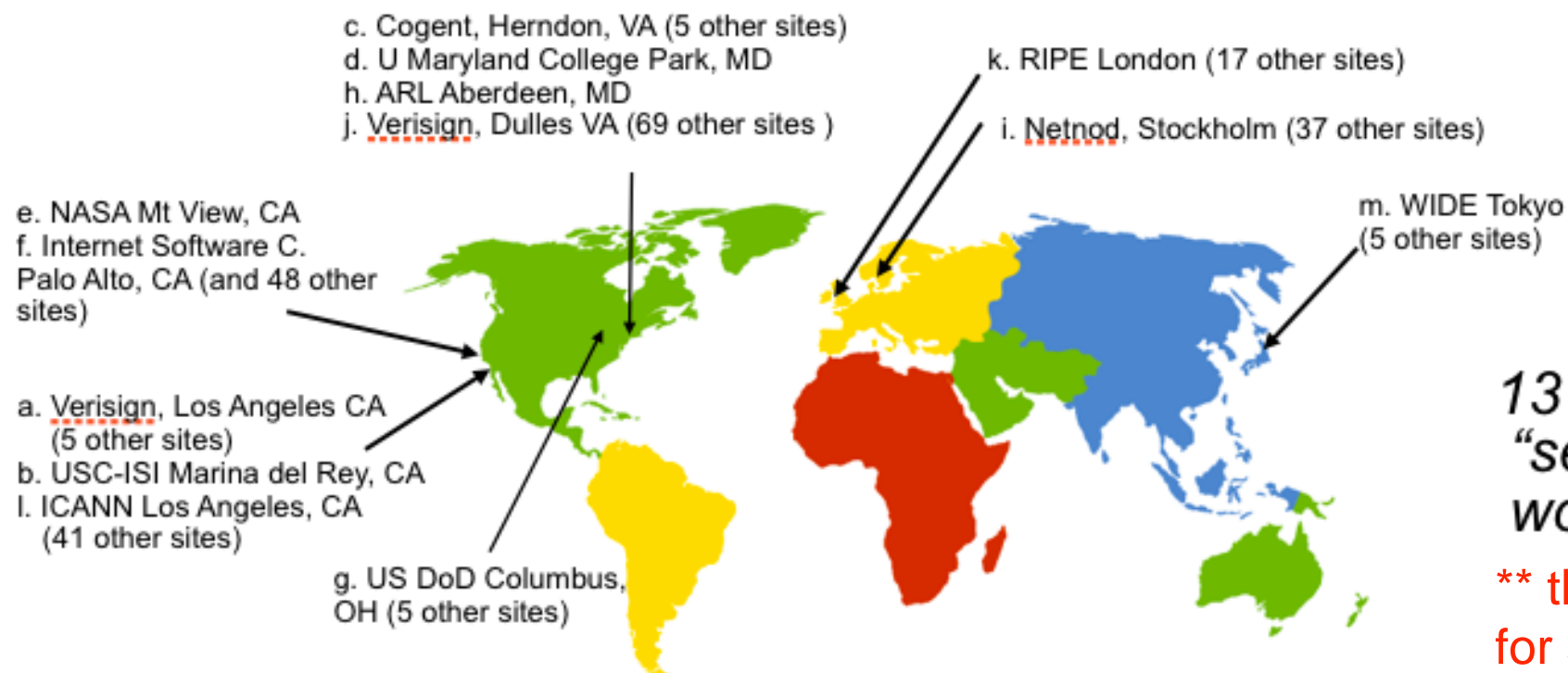


- **Client wants IP for [www.amazon.com](http://www.amazon.com)**

- Client queries root server to find com DNS server
- Client queries .com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for [www.amazon.com](http://www.amazon.com)

# Root DNS 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*

**\*\* though they are replicated  
for security and reliability**

# Top-Level Domain & Authoritative DNS 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)
- Verisign maintains servers for .com TLD (and many others)
- Educause maintains servers 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

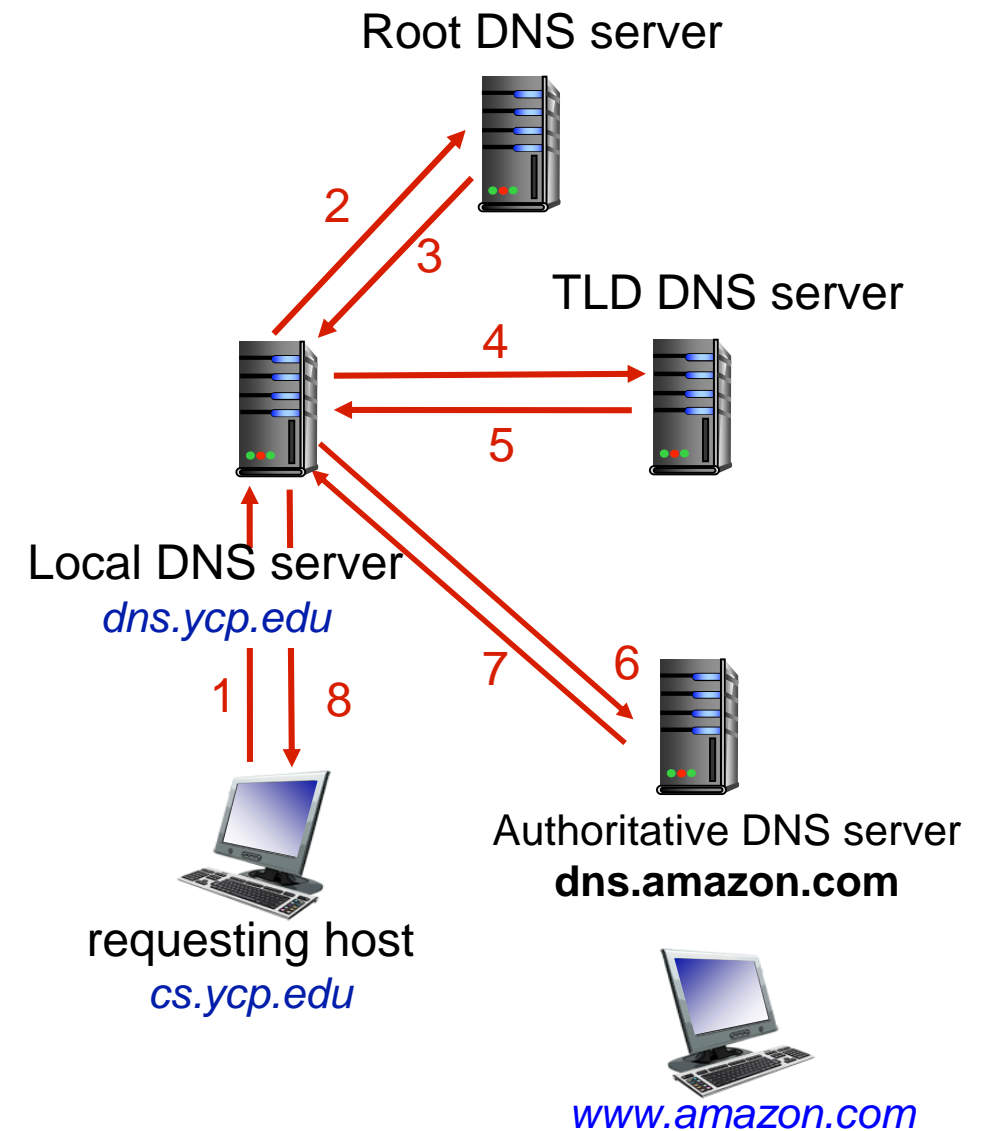
# Local DNS Name Server

---

- **Does not strictly belong to the DNS 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**
  - 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 **cs.ycp.edu** wants IP address for **www.amazon.com**
- **Iterated query:**
  - Contacted server replies with name/address of server to contact
  - “I don’t know this name, but ask this server”

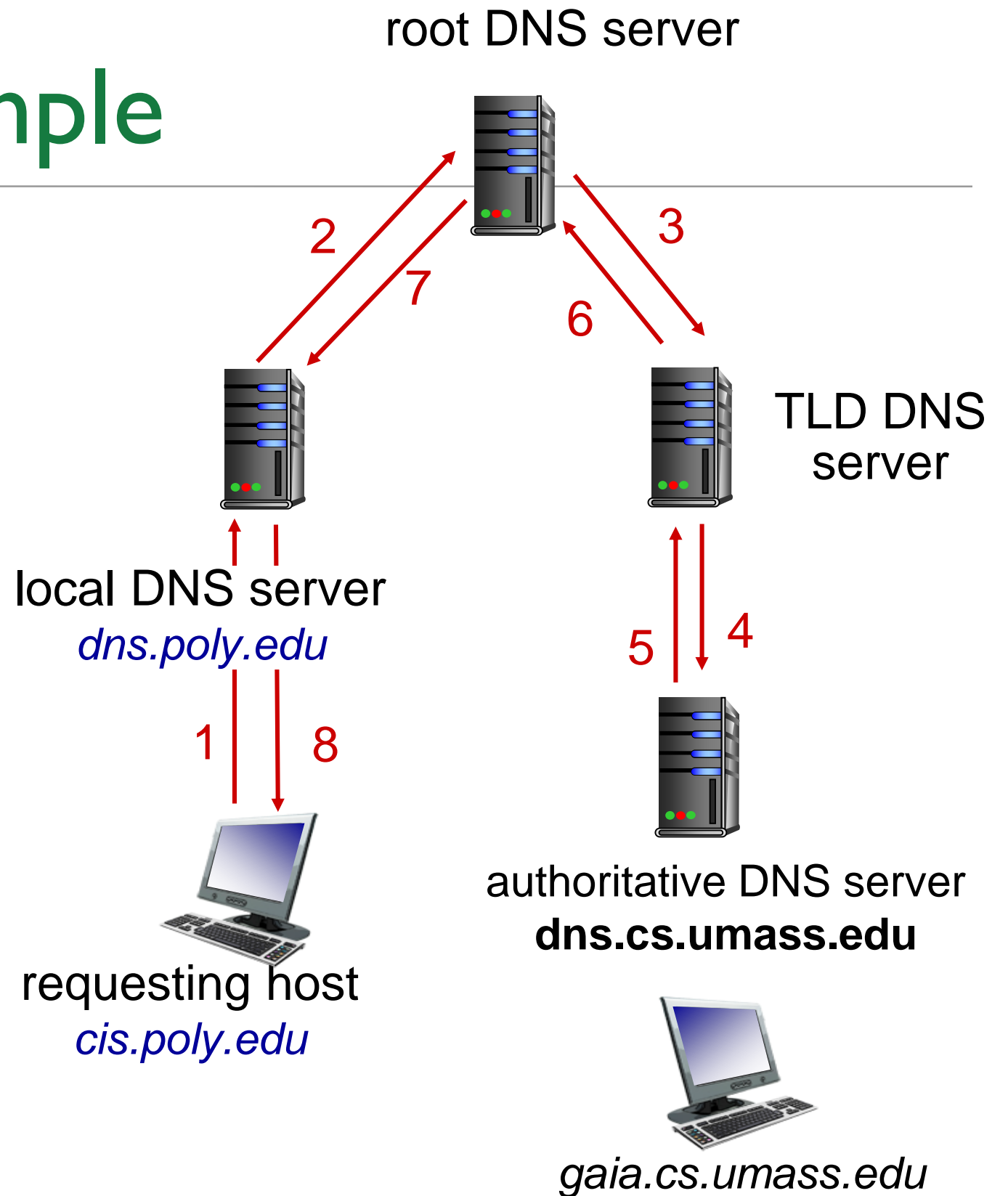




# DNS name resolution example

## *recursive query:*

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



# 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
- **Cached entries may be out-of-date (best effort name-to-address translation!)**
  - If a named host changes its IP address, may not be known Internet-wide until all TTLs expire

# DNS Records

---

- **Resource Record (RR) format stored by DNS servers**

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

- **Four different RR types**

## 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 mail server associated with `name`

# DNS Message Format

- **Query and Reply messages, both use same message format**
  - Message Header
    - **Identification:** 16 bit # for query, reply includes same #
    - **Flags:**
      - Query or reply
      - Recursion desired
      - Recursion available
      - Reply is authoritative
    - **Question section:** contains name and type fields for the query
    - **Answer section:** contains RRs in response to a query
    - **Authority section:** contains RR for authoritative servers

Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of questions)	
Answers (variable # of answers)	
Authority (variable # of RRs)	
Additional Info (variable # of RRs)	

# DNS Vulnerabilities

---

- **Distributed Denial of service attacks on name server**

- Bombard root servers with traffic
  - Not successful to date
  - Root servers are protected by traffic filters
  - Local DNS servers cache IP addresses of TLD servers, allowing root server bypass
- Bombard TLD servers
  - Potentially more dangerous

- **Redirect attacks**

- Man-in-middle
  - Intercept queries and return bogus replies
- DNS poisoning
  - Send bogus replies to DNS server which then caches that info