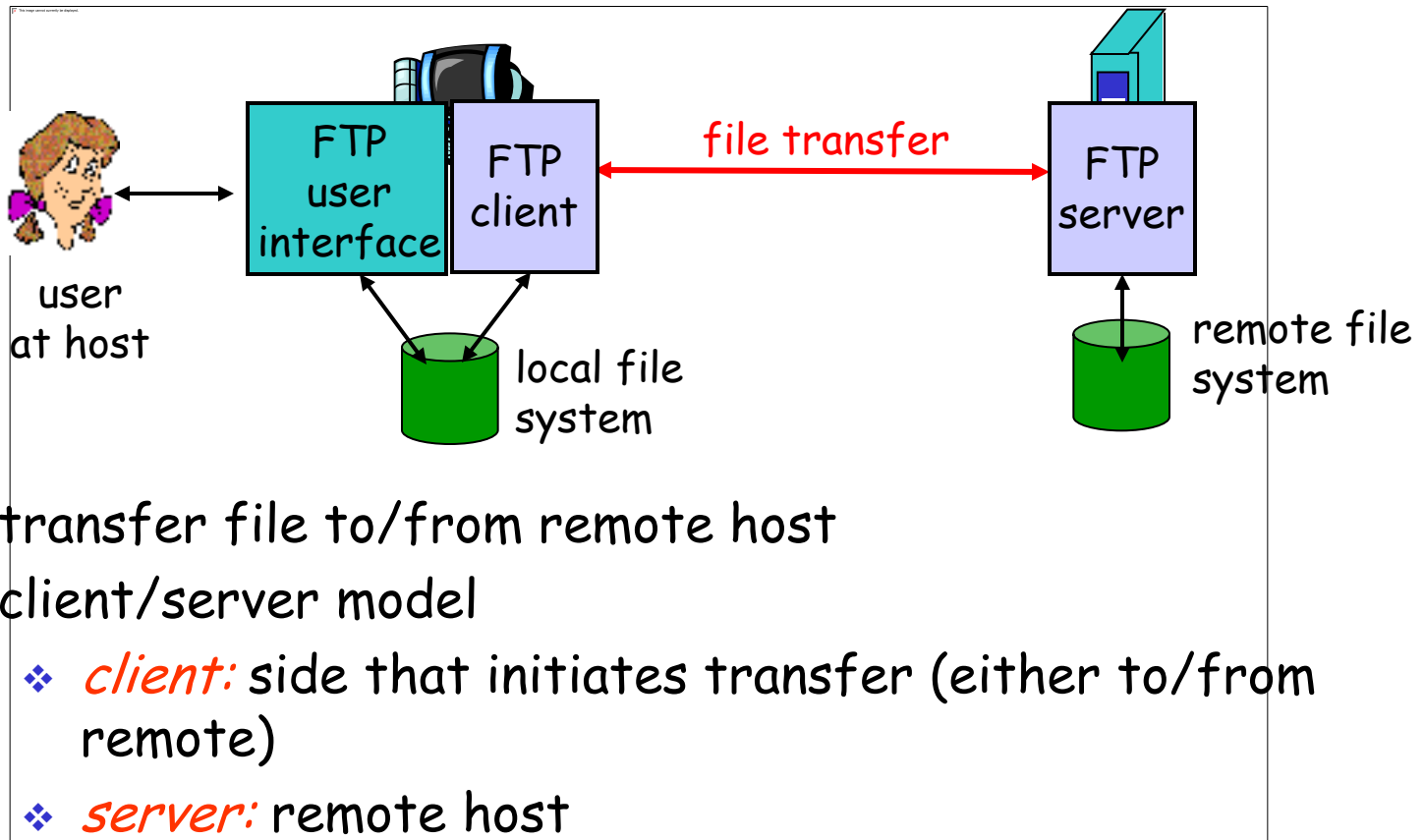


Chapter 2: Application layer

- ❑ 2.1 Principles of network applications
- ❑ 2.2 Web and HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 P2P applications
- ❑ 2.7 Socket programming with UDP
- ❑ 2.8 Socket programming with TCP

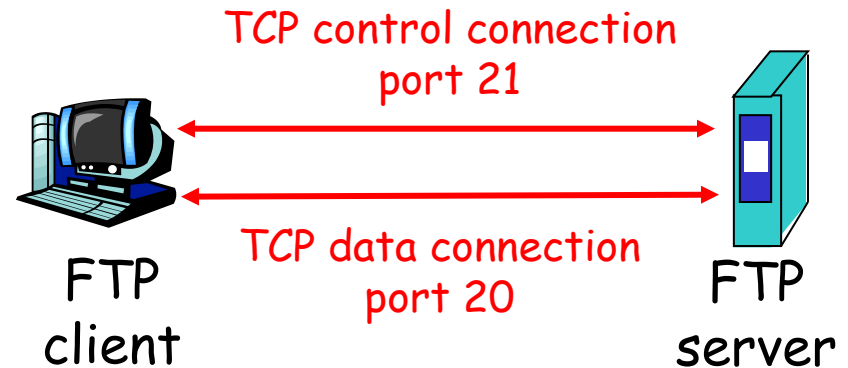
FTP: the file transfer protocol



- ❑ transfer file to/from 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 at port 21, TCP is transport protocol
- ❑ client authorized over control connection
- ❑ client browses remote directory by sending commands over control connection.
- ❑ when server receives file transfer command, server opens 2nd TCP connection (for file) to client
- ❑ after transferring one file, server closes data connection.



- ❑ server opens another TCP data connection to transfer another file.
- ❑ control connection: "out of band"
- ❑ FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

Sample commands:

- ❑ sent as ASCII text over control channel
- ❑ USER *username*
- ❑ PASS *password*
- ❑ LIST return list of file in current directory
- ❑ RETR *filename* retrieves (gets) file
- ❑ STOR *filename* stores (puts) file onto remote host

Sample return codes

- ❑ status code and phrase (as in HTTP)
- ❑ 331 Username OK, password required
- ❑ 125 data connection already open; transfer starting
- ❑ 425 Can't open data connection
- ❑ 452 Error writing file

Chapter 2: Application layer

- ❑ 2.1 Principles of network applications
- ❑ 2.2 Web and HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 P2P applications
- ❑ 2.7 Socket programming with UDP
- ❑ 2.8 Socket programming with TCP

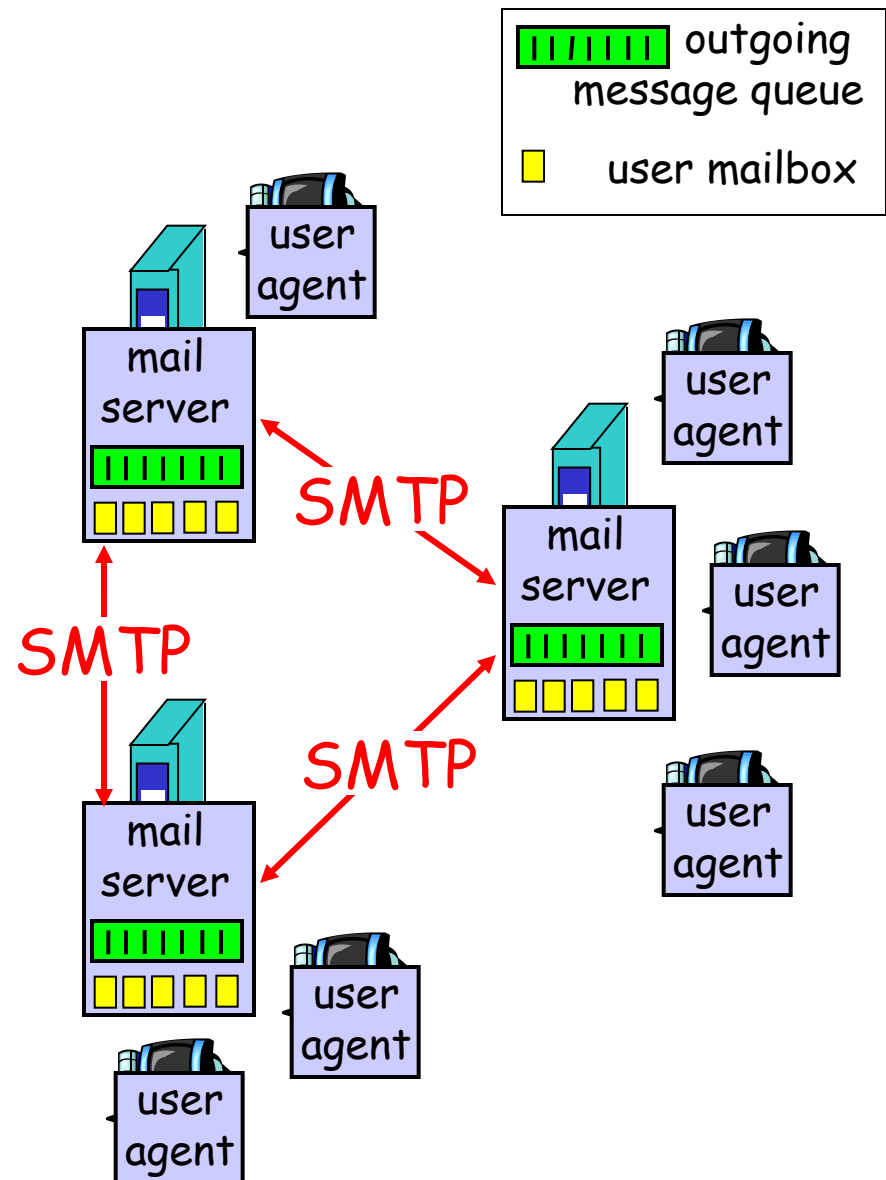
Electronic Mail

Three major components:

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

User Agent

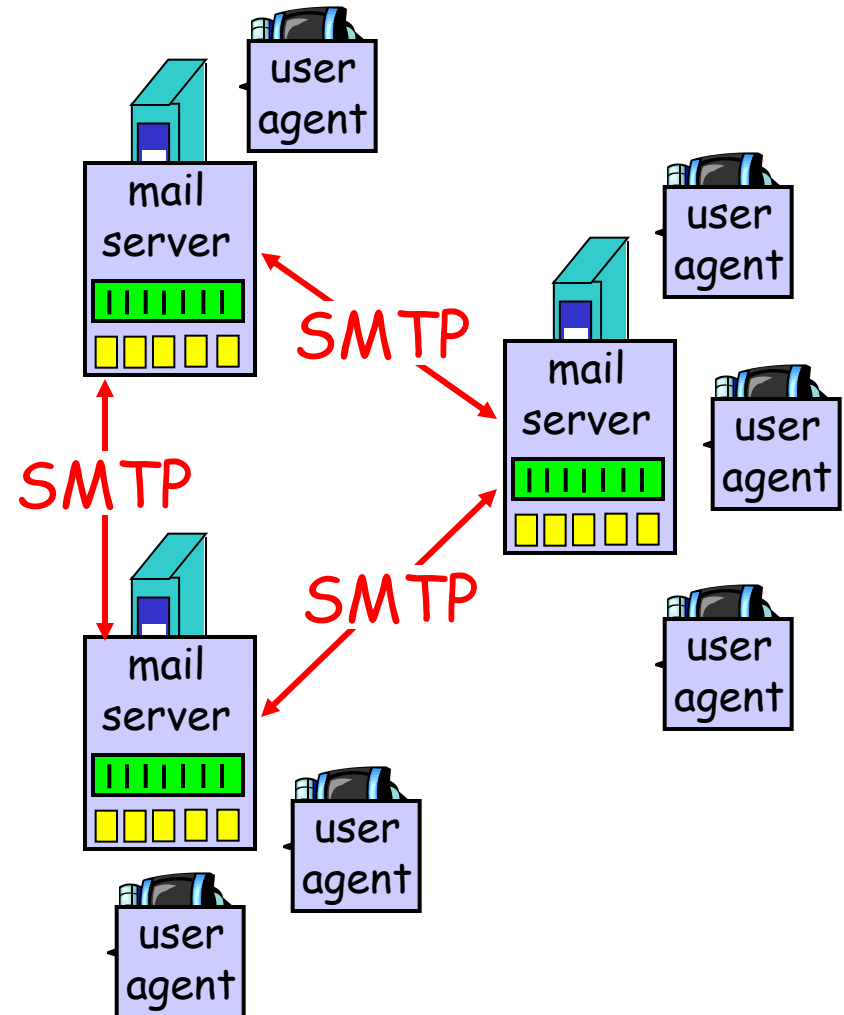
- ❑ a.k.a. "mail reader"
- ❑ composing, editing, reading mail messages
- ❑ e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- ❑ outgoing, incoming messages stored on server



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

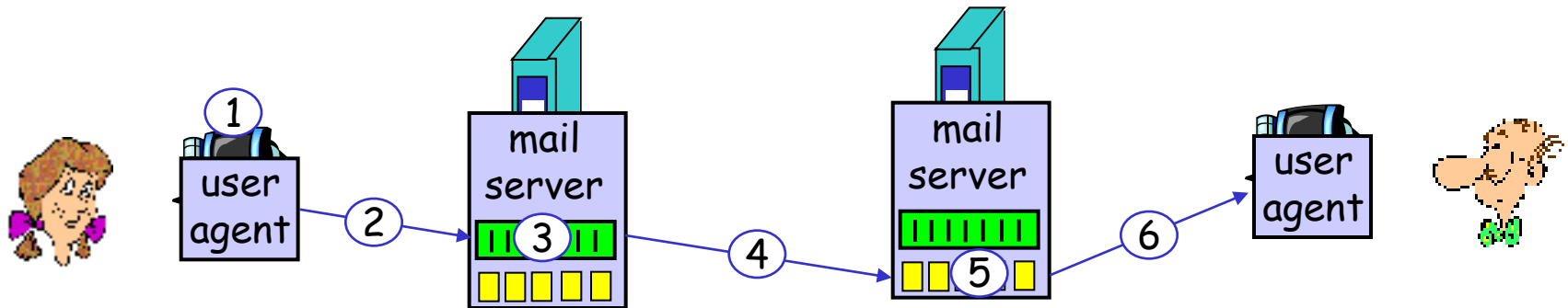


Electronic Mail: SMTP [RFC 2821]

- ❑ 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
 - ❖ **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 and "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
```

Try SMTP interaction for yourself:

- ❑ `telnet servername 25`
- ❑ see 220 reply from server
- ❑ enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

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:

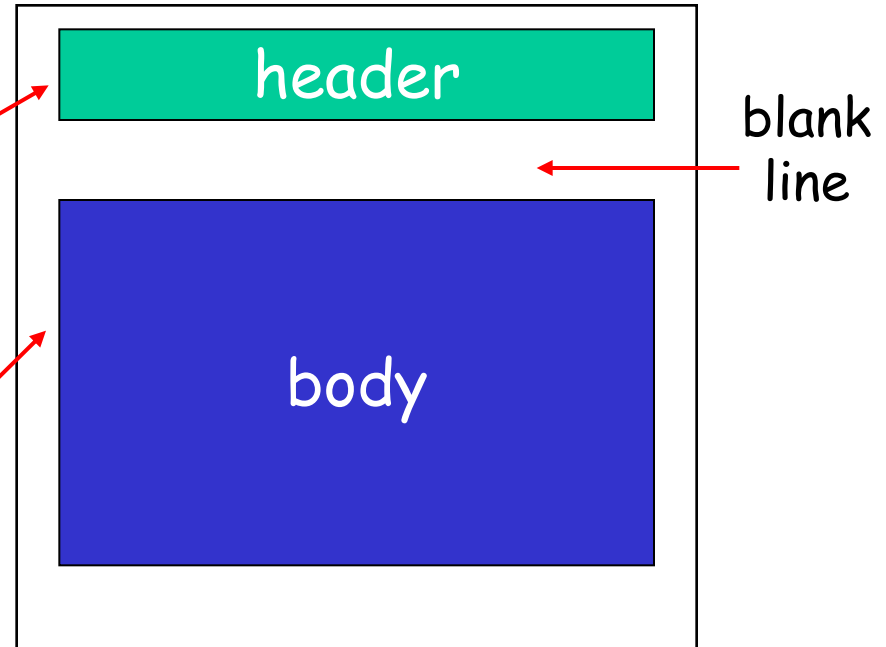
- ❑ HTTP: pull
- ❑ SMTP: push
- ❑ both have ASCII command/response interaction, status codes
- ❑ HTTP: each object encapsulated in its own response msg
- ❑ SMTP: multiple objects sent in multipart msg

Mail message format

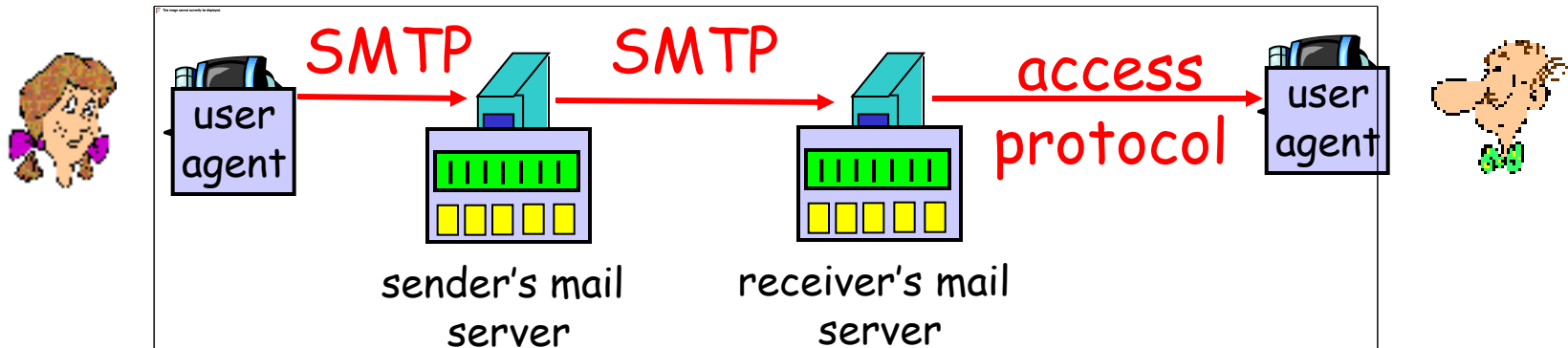
SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

- header lines, e.g.,
 - ❖ To:
 - ❖ From:
 - ❖ Subject:*different from SMTP 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 (agent <-->server) and download
 - ❖ IMAP: Internet Mail Access Protocol [RFC 1730]
 - more features (more complex)
 - 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:

- 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
```

POP3 (more) and IMAP

More about POP3

- ❑ Previous example uses “download and delete” mode.
- ❑ Bob cannot re-read e-mail if he changes client
- ❑ “Download-and-keep”: copies of messages on different clients
- ❑ POP3 is stateless across sessions

IMAP

- ❑ Keep all messages in one place: the server
- ❑ Allows user to organize messages in folders
- ❑ IMAP keeps user state across sessions:
 - ❖ names of folders and mappings between message IDs and folder name

Chapter 2: Application layer

- ❑ 2.1 Principles of network applications
- ❑ 2.2 Web and HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 P2P applications
- ❑ 2.7 Socket programming with UDP
- ❑ 2.8 Socket programming with TCP

DNS: Domain Name System

People: many identifiers:

- ❖ SSN, name, passport #

Internet hosts, routers:

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

Q: map between IP addresses and name ?

Domain Name System:

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

DNS

DNS services

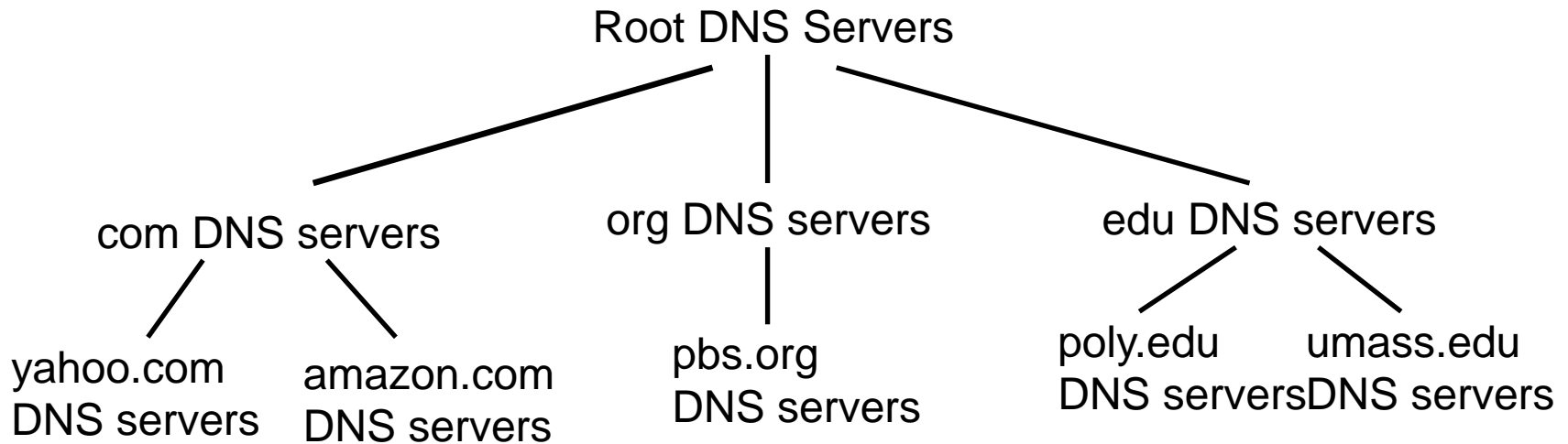
- ❑ hostname to IP address translation
- ❑ host aliasing
 - ❖ Canonical, alias names
- ❑ mail server aliasing
- ❑ load distribution
 - ❖ replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

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

doesn't *scale!*

Distributed, Hierarchical Database

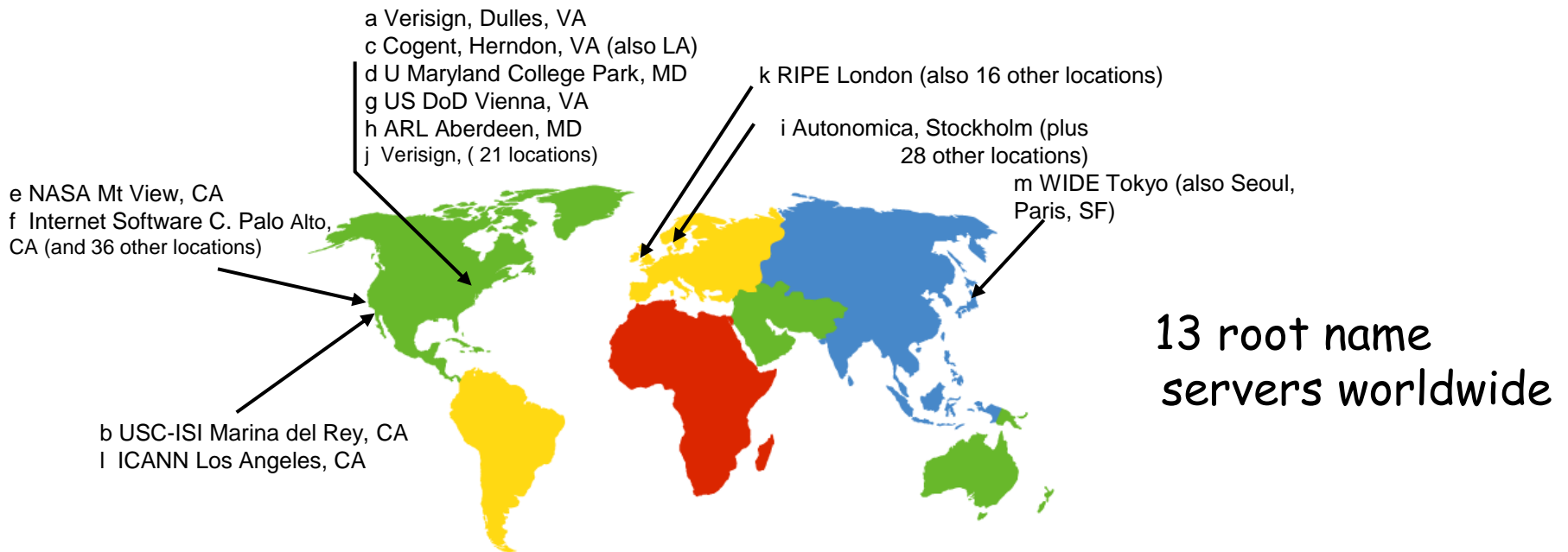


Client wants IP for www.amazon.com; 1st approx:

- ❑ client queries a 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

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

TLD and Authoritative Servers

- ❑ **Top-level domain (TLD) servers:**
 - ❖ responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
 - ❖ Network Solutions maintains servers for com TLD
 - ❖ 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

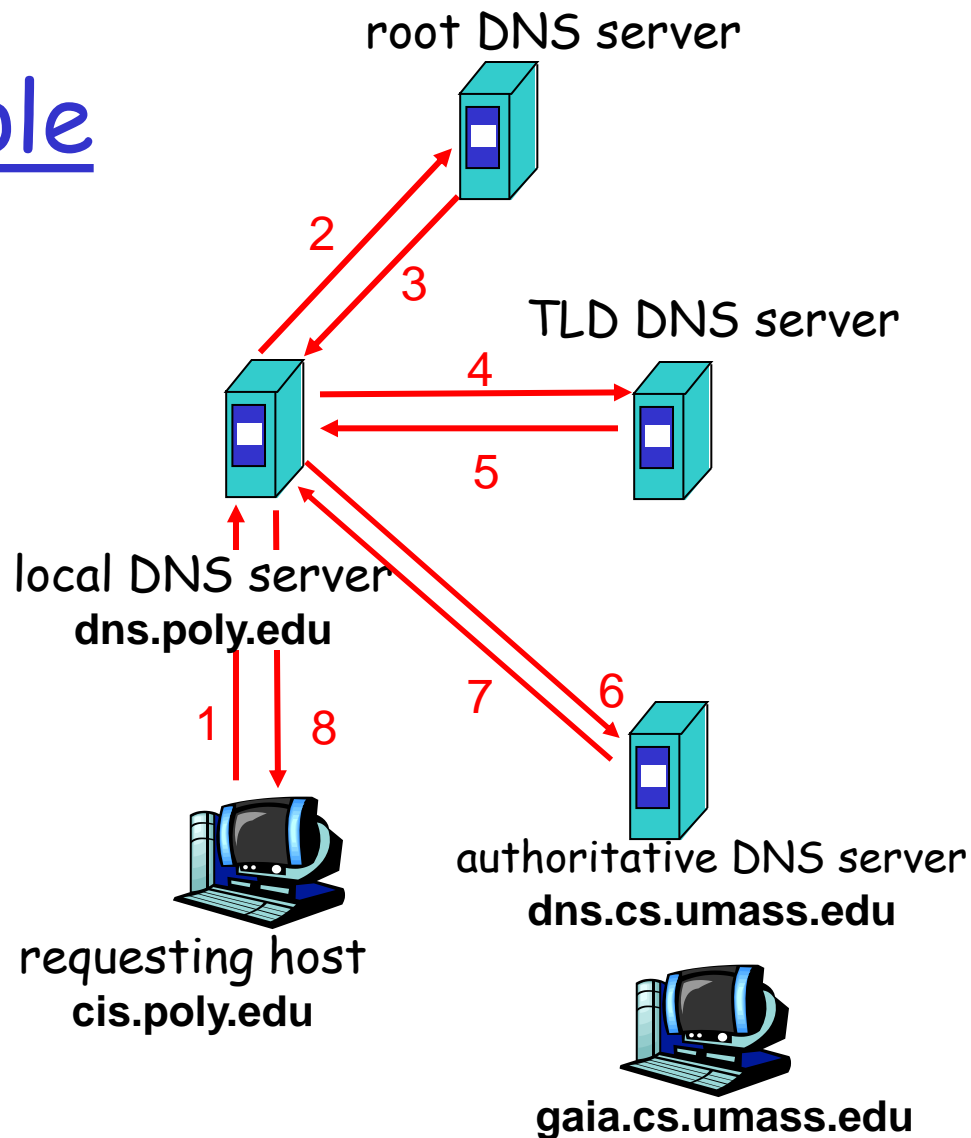
- ❑ does not strictly belong to hierarchy
- ❑ each ISP (residential ISP, company, university) has one.
 - ❖ also called "default name server"
- ❑ when host makes DNS query, query is sent to its local DNS server
 - ❖ acts as proxy, forwards query into hierarchy

DNS name resolution example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

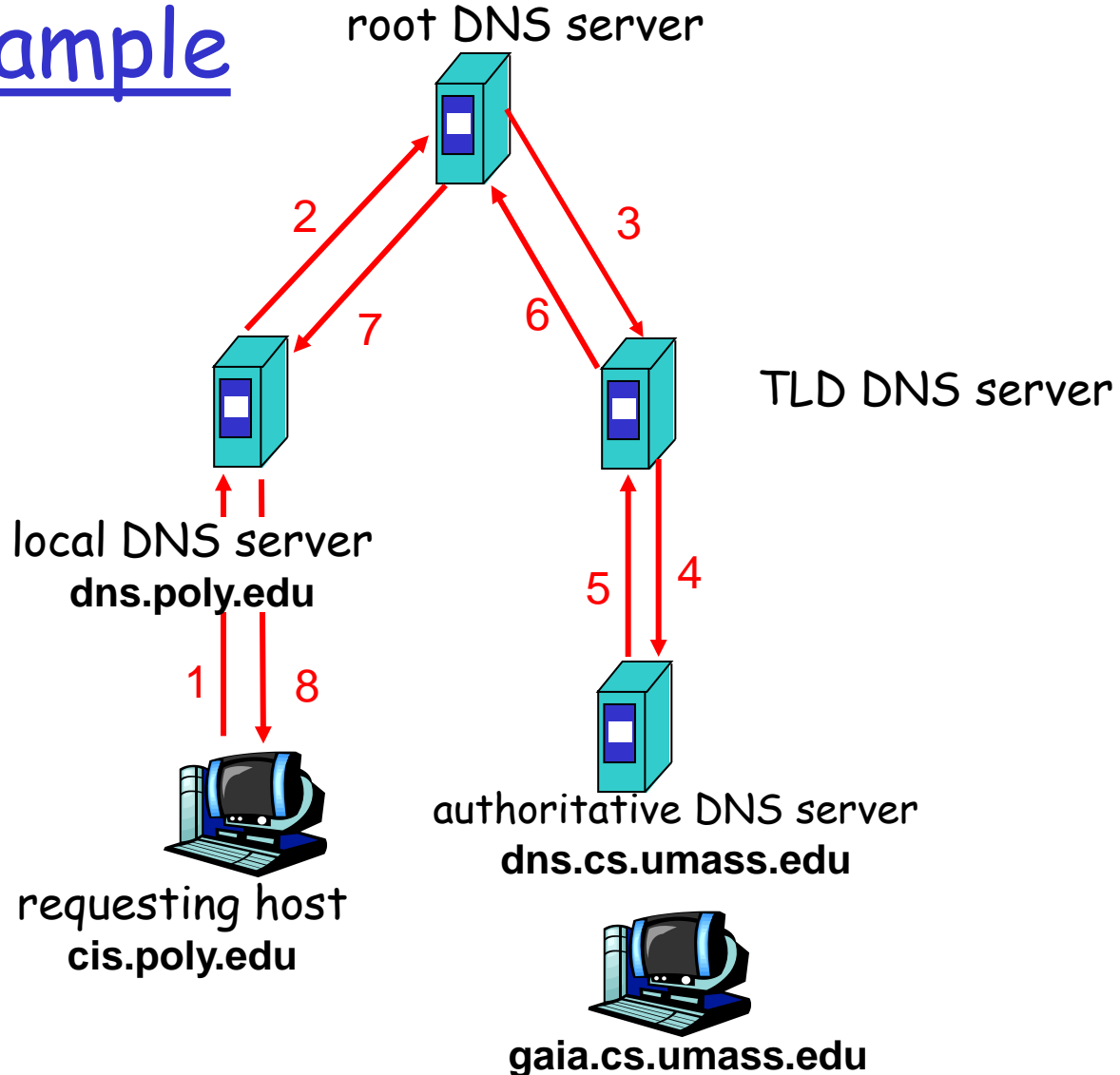
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- ❑ puts burden of name resolution on contacted name server
- ❑ heavy load?



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
 - ❖ cache entries timeout (disappear) after some time
 - ❖ TLD servers typically cached in local name servers
 - Thus root name servers not often visited
- update/notify mechanisms under design by IETF
 - ❖ RFC 2136
 - ❖ <http://www.ietf.org/html.charters/dnsind-charter.html>

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

DNS protocol : *query* and *reply* messages, both with same *message format*

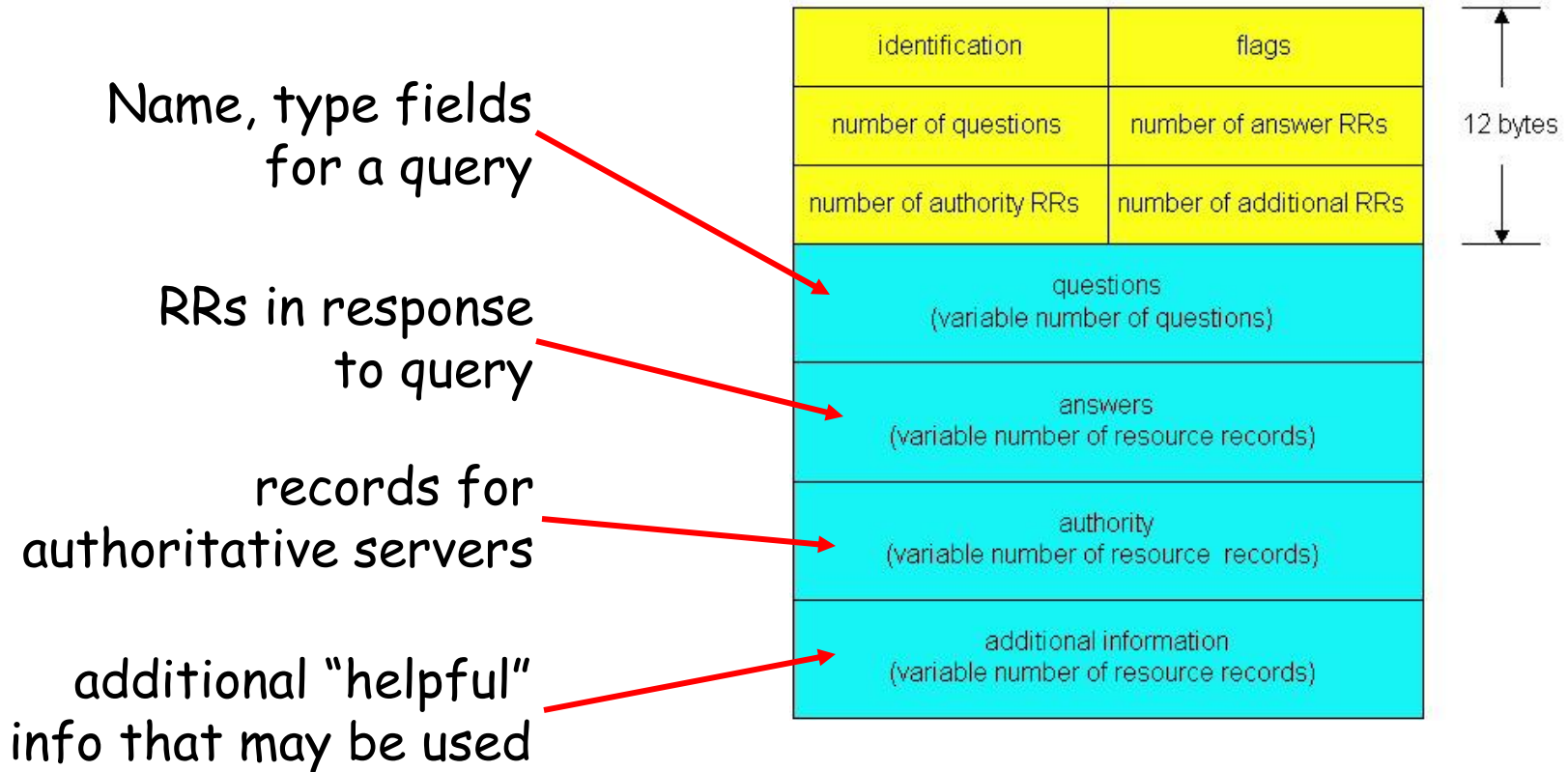
msg header

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

identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	



DNS protocol, messages



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.networkutopia.com`; Type MX record for `networkutopia.com`
- ❑ *How do people get IP address of your Web site?*