

User-server state: cookies

Many major Web sites use cookies

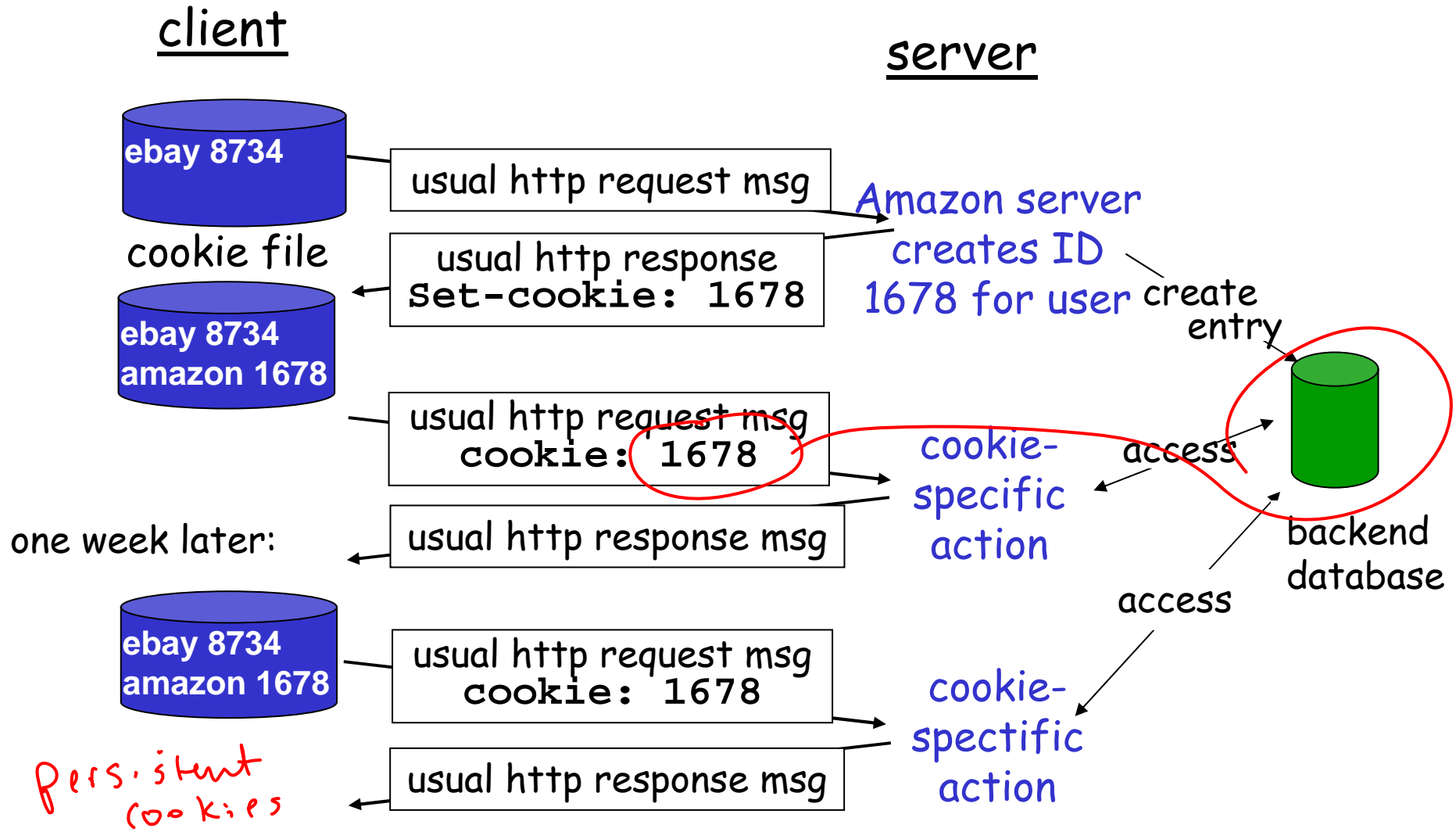
Four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

- Susan always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - ❖ unique ID
 - ❖ entry in backend database for ID

Cookies: keeping "state" (cont.)



Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state
(Web e-mail)

How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

aside

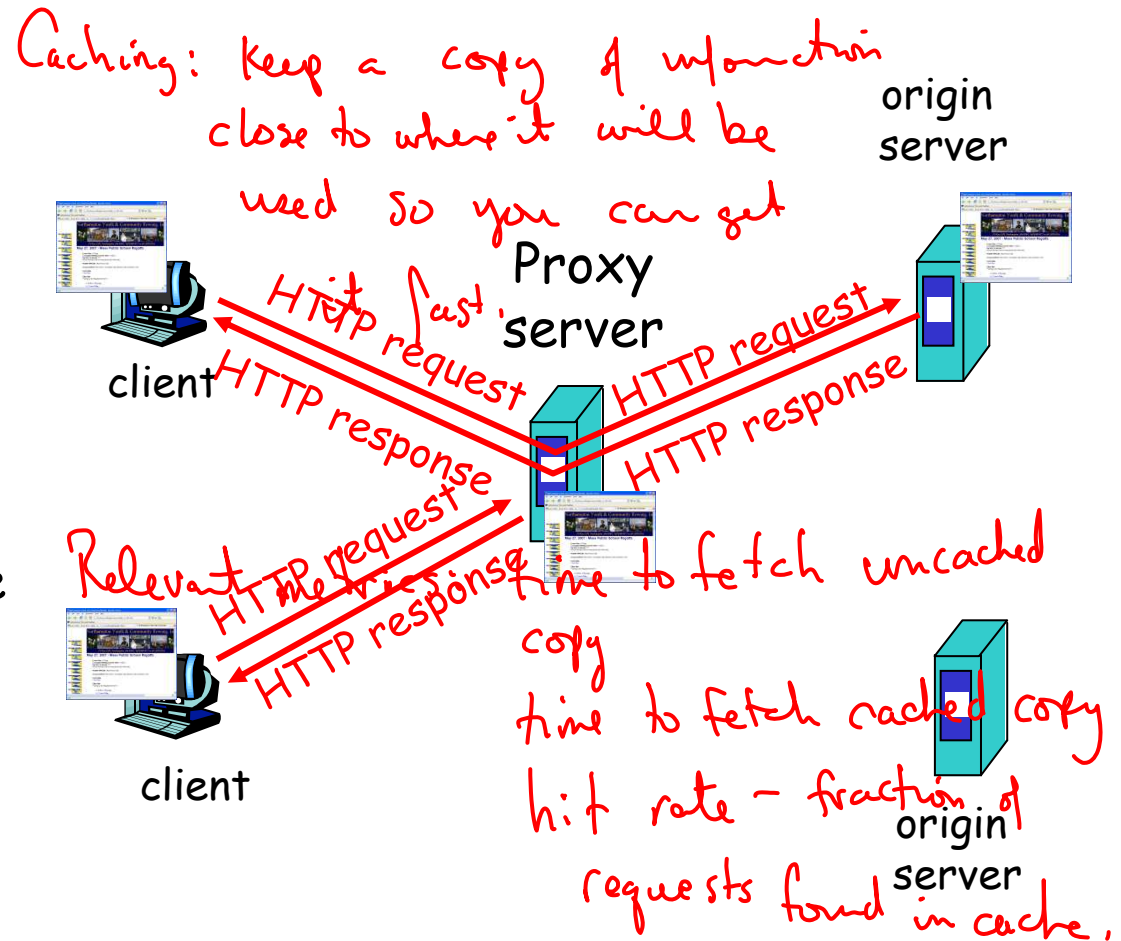
Cookies and privacy:

- cookies permit sites to learn a lot about you

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
 - ❖ object in cache: cache returns object
 - ❖ else cache requests object from origin server, then returns object to client



More about Web caching

- ❑ cache acts as both client and server
 - ❑ typically cache is installed by ISP (university, company, residential ISP)
- General for caches*

Avg access time =

*hit rate * time to fetch from cache*

*+ (1 - hit rate) * time to fetch from home location*

Why Web caching?

- ❑ reduce response time for client request
- ❑ reduce traffic on an institution's access link.
- ❑ Internet dense with caches: enables "poor" content providers to effectively deliver content (but so does P2P file sharing)

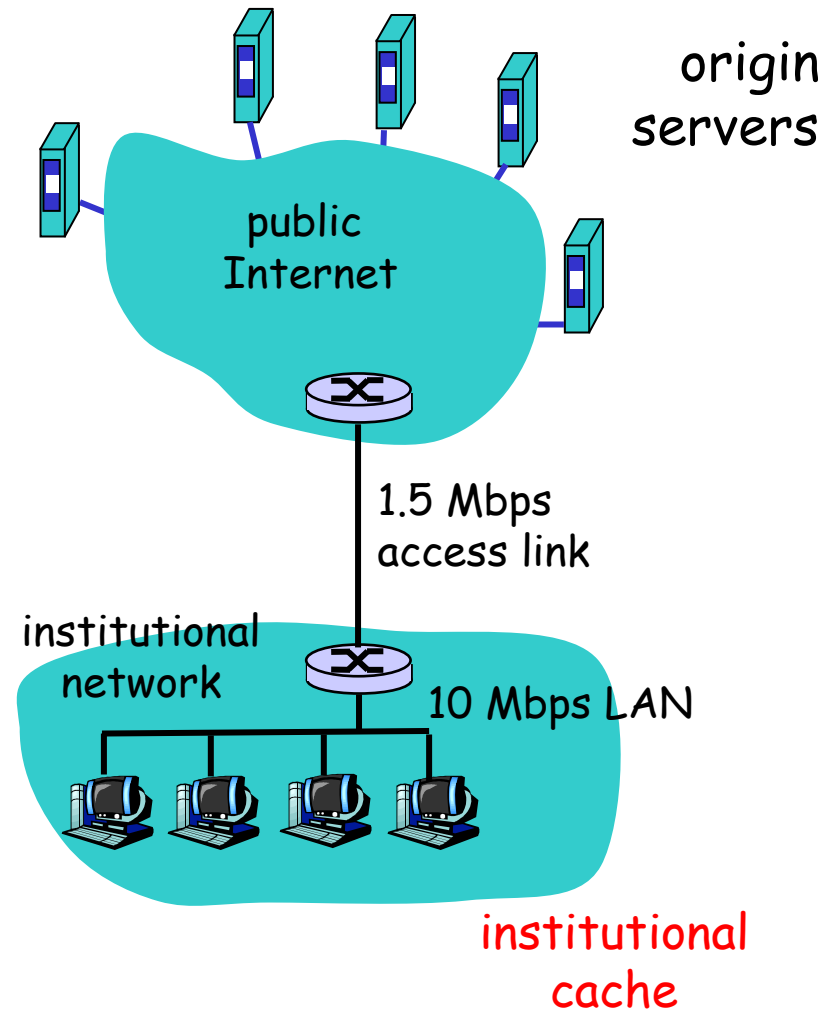
Caching example

Assumptions

- ❑ average object size = 100,000 bits
- ❑ avg. request rate from institution's browsers to origin servers = 15/sec
- ❑ delay from institutional router to any origin server and back to router = 2 sec

Consequences

- ❑ utilization on LAN = 15%
- ❑ utilization on access link = 100%
- ❑ total delay = Internet delay + access delay + LAN delay
= 2 sec + minutes + milliseconds



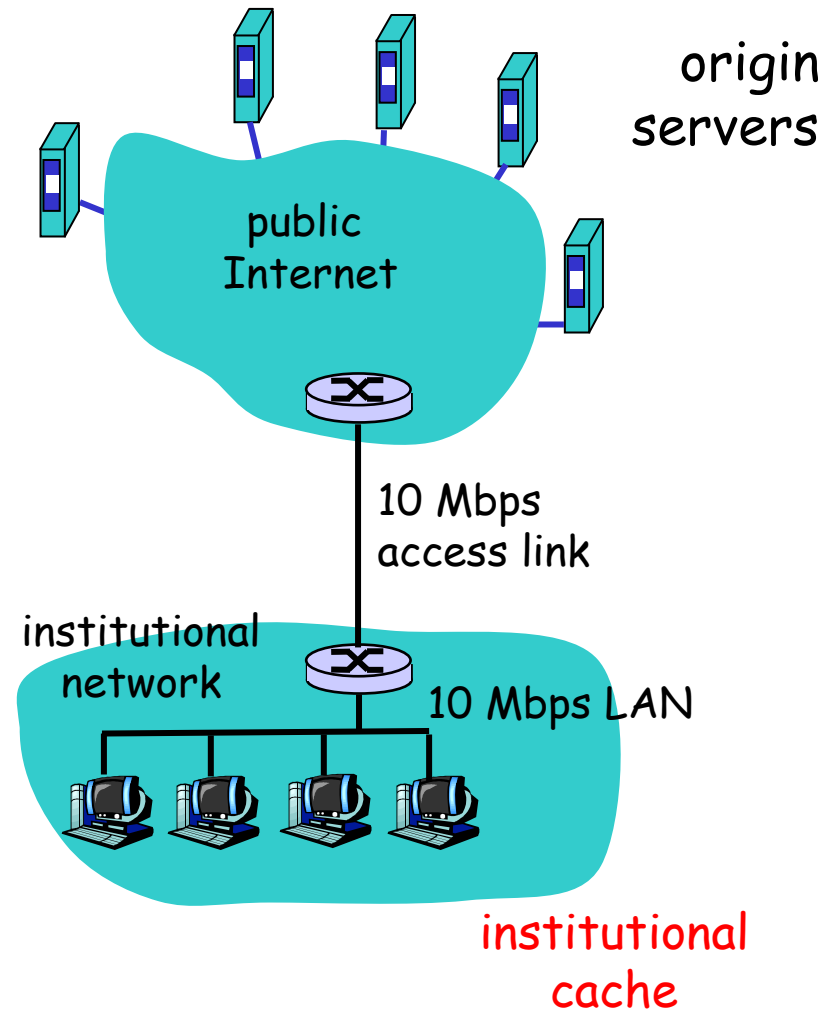
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
= 2 sec + msec + msec
- often a costly upgrade



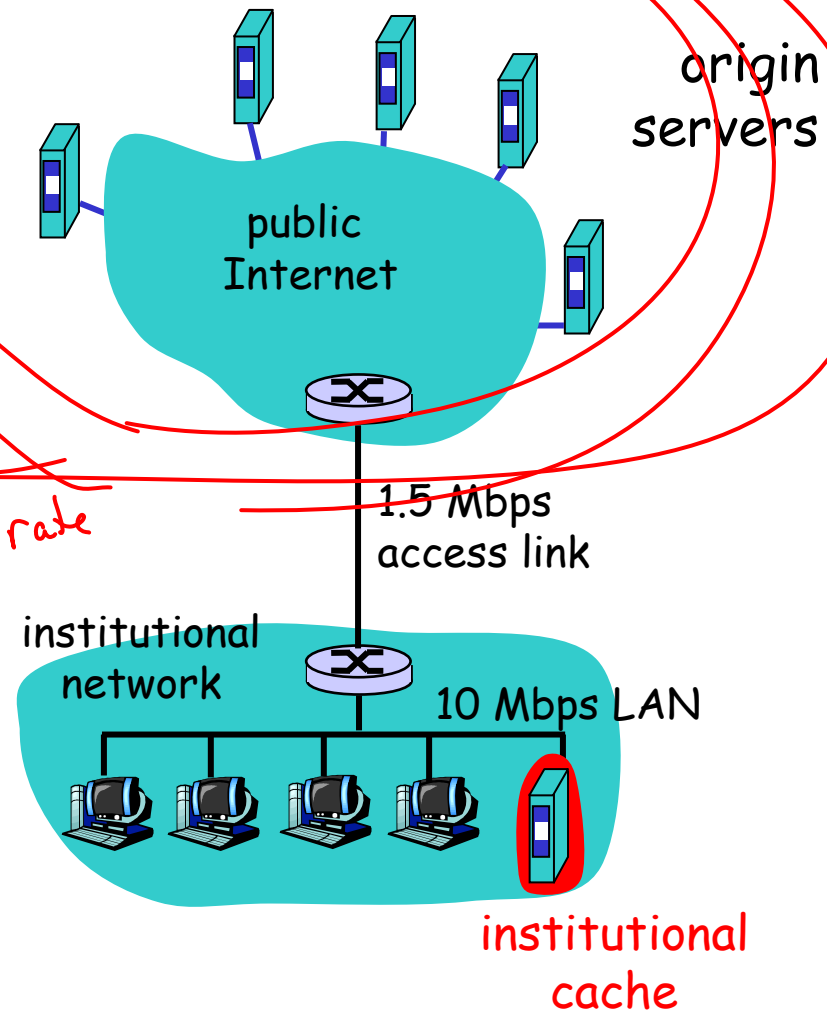
Caching example (cont)

possible solution: install cache

- suppose hit rate is 0.4

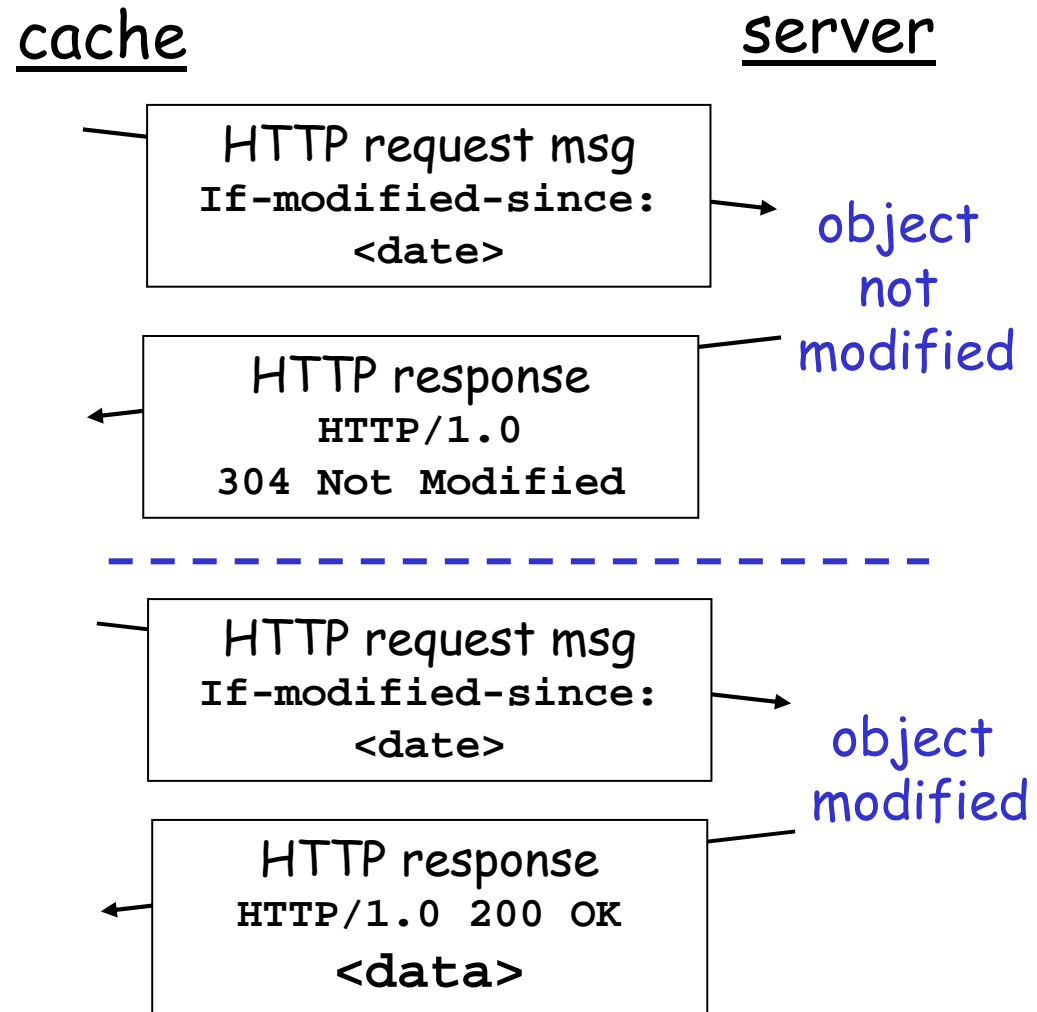
consequence

- 40% requests will be *hit rate* satisfied almost immediately
- 60% requests satisfied by *miss rate* origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = $.6 * (2.01) \text{ secs} + .4 * \text{milliseconds} < 1.4 \text{ secs}$



Conditional GET

- **Goal:** don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
If-modified-since:
<date>
- server: response contains no object if cached copy is up-to-date:
HTTP/1.0 304 Not Modified



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 TCP
- 2.8 Socket programming with UDP

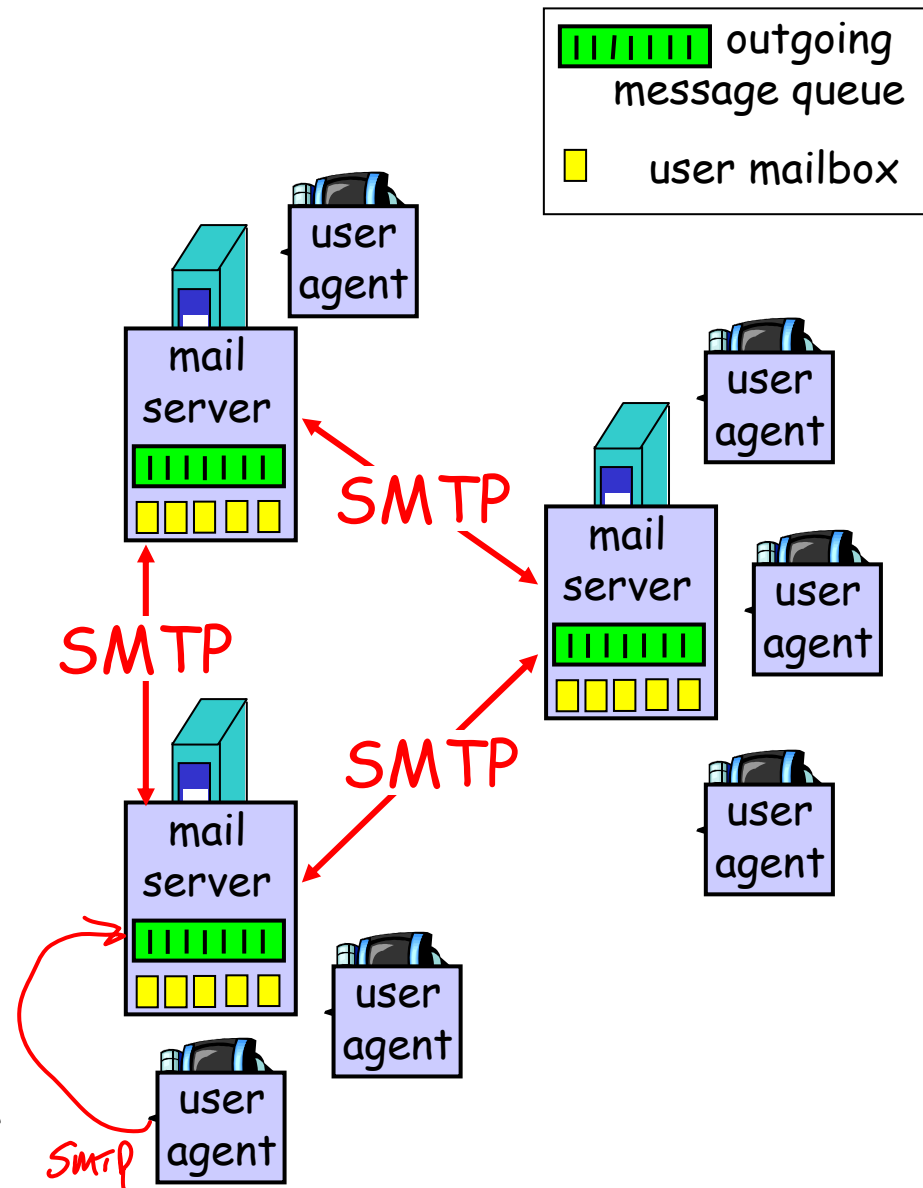
Electronic Mail

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

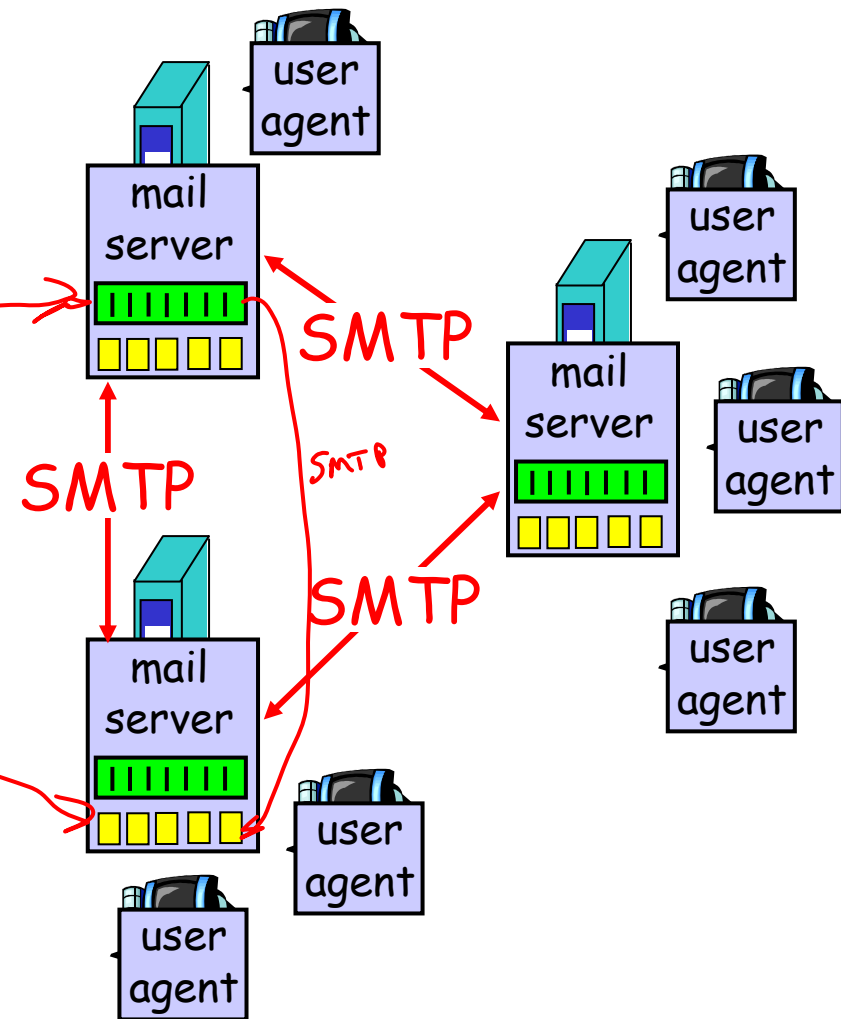


View Source for mail message.

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

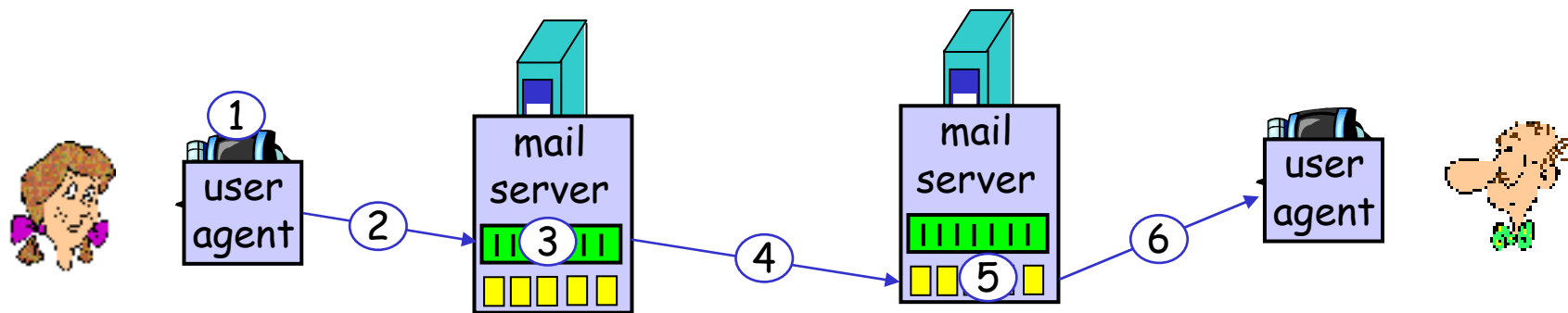


Electronic Mail: SMTP [RFC 2821]

- 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 characters.
- ❑ MIME: Multi-purpose Internet Mail Extensions - encode binary data in 7-bit ASCII.

Comparison with HTTP:

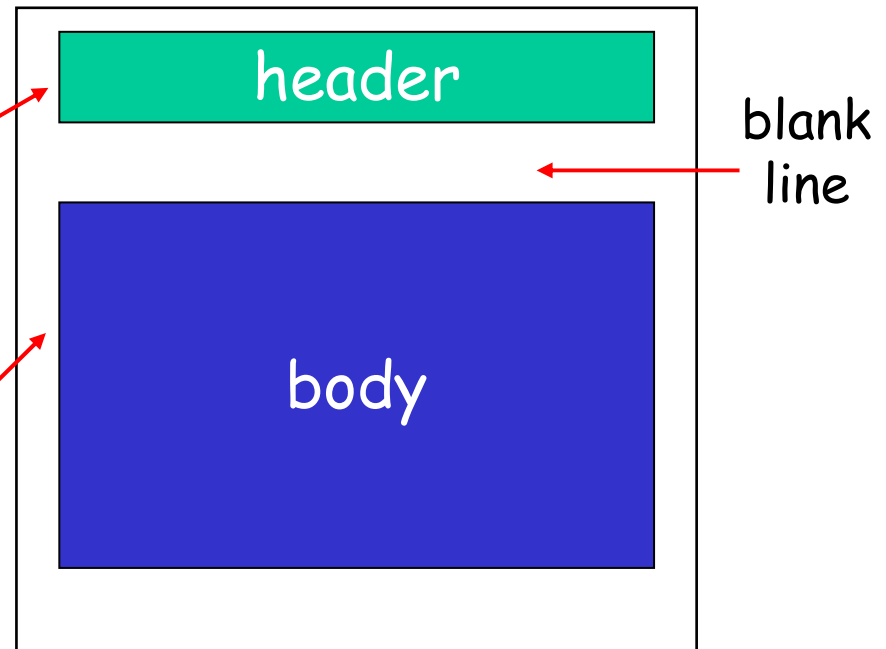
- ❑ HTTP: pull
- ❑ SMTP: push
- ❑ both have ASCII command/response interaction, status codes
- ❑ HTTP: each object encapsulated in its own response msg
- ❑ SMTP: multiple objects sent in multipart msg

Mail message format

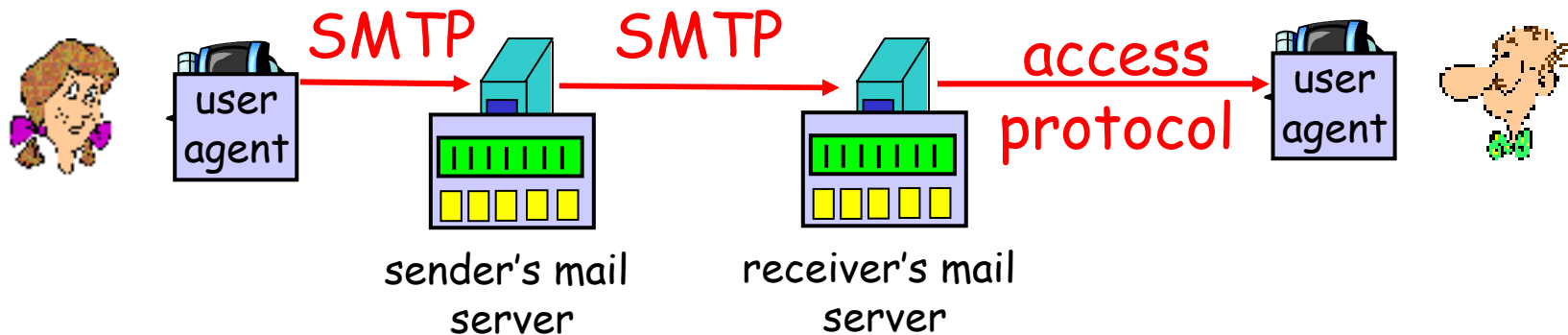
SMTP: protocol for exchanging email msgs

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.

Text-based

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 TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server

DNS: Domain Name System

People: many identifiers:

- ❖ SSN, name, passport #

Internet hosts, routers:

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

Q: map between IP addresses and name ?

Name service maps names to values.

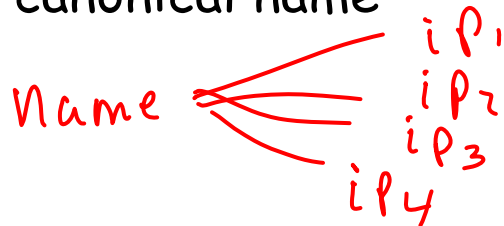
Domain Name System:

- ❑ *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

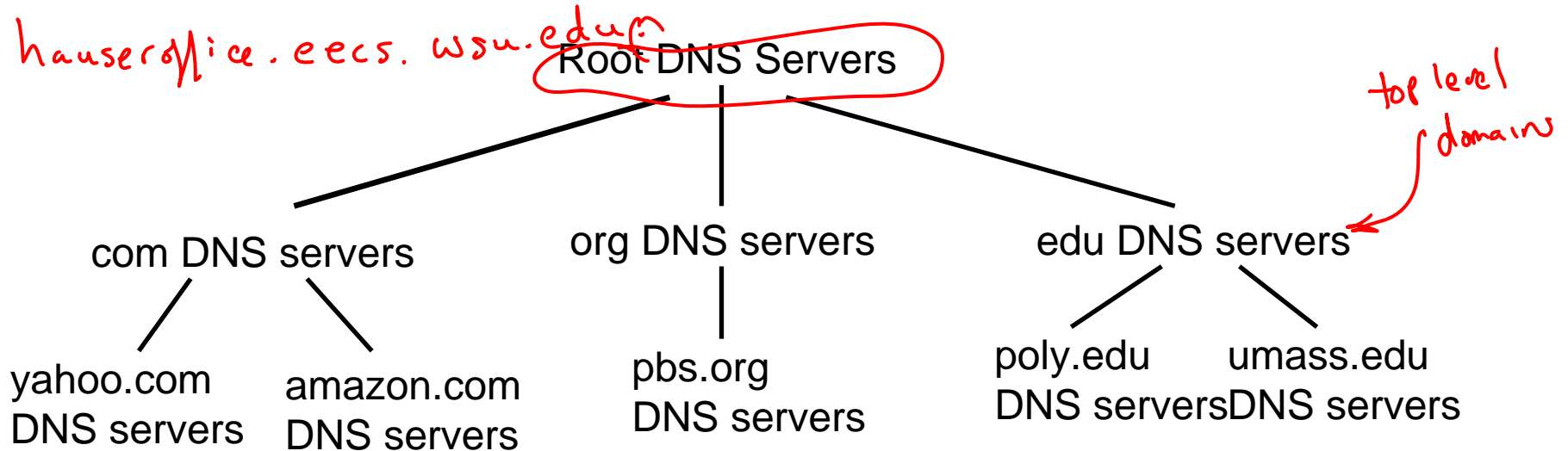


Why not centralize DNS?

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

doesn't *scale!*

Distributed, Hierarchical Database

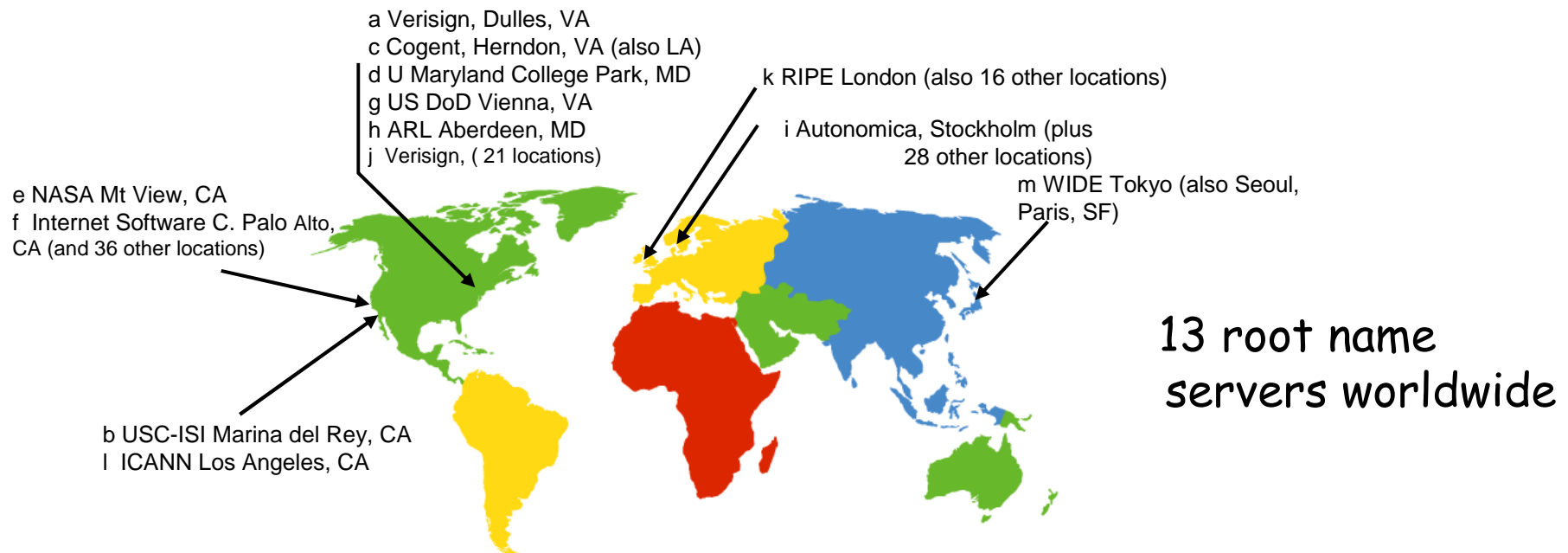


Client wants IP for www.amazon.com; 1st approx:

- ❑ 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



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:

occur at every level

- ❖ 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

vs cached.

Local Name Server

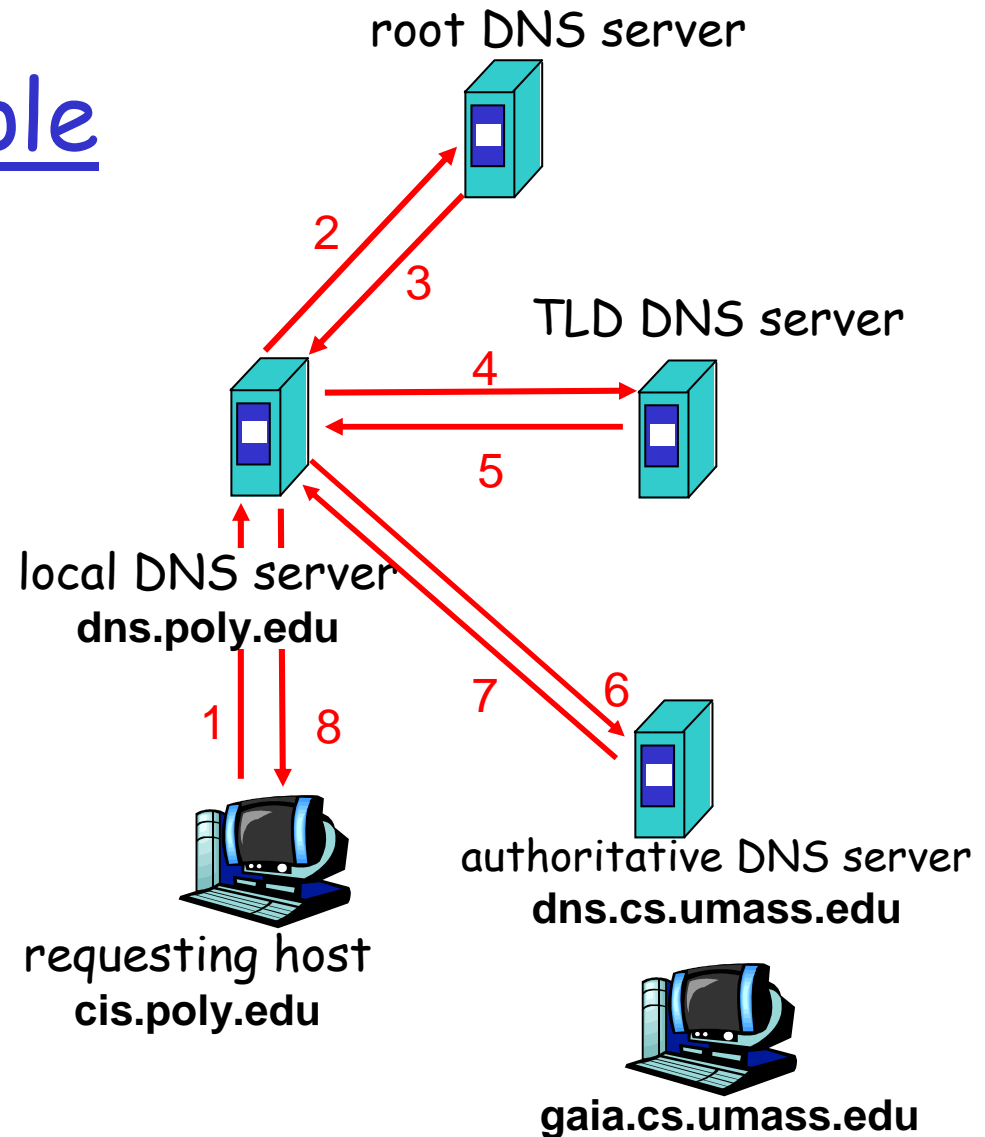
- ❑ does not strictly belong to hierarchy
- ❑ each ISP (residential ISP, company, university) has one.
 - ❖ Typically configured in hosts as "default name server"
- ❑ when host makes DNS query, query is sent to its local DNS server
 - ❖ acts as proxy, forwards query into hierarchy

DNS name resolution example

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

iterated query:

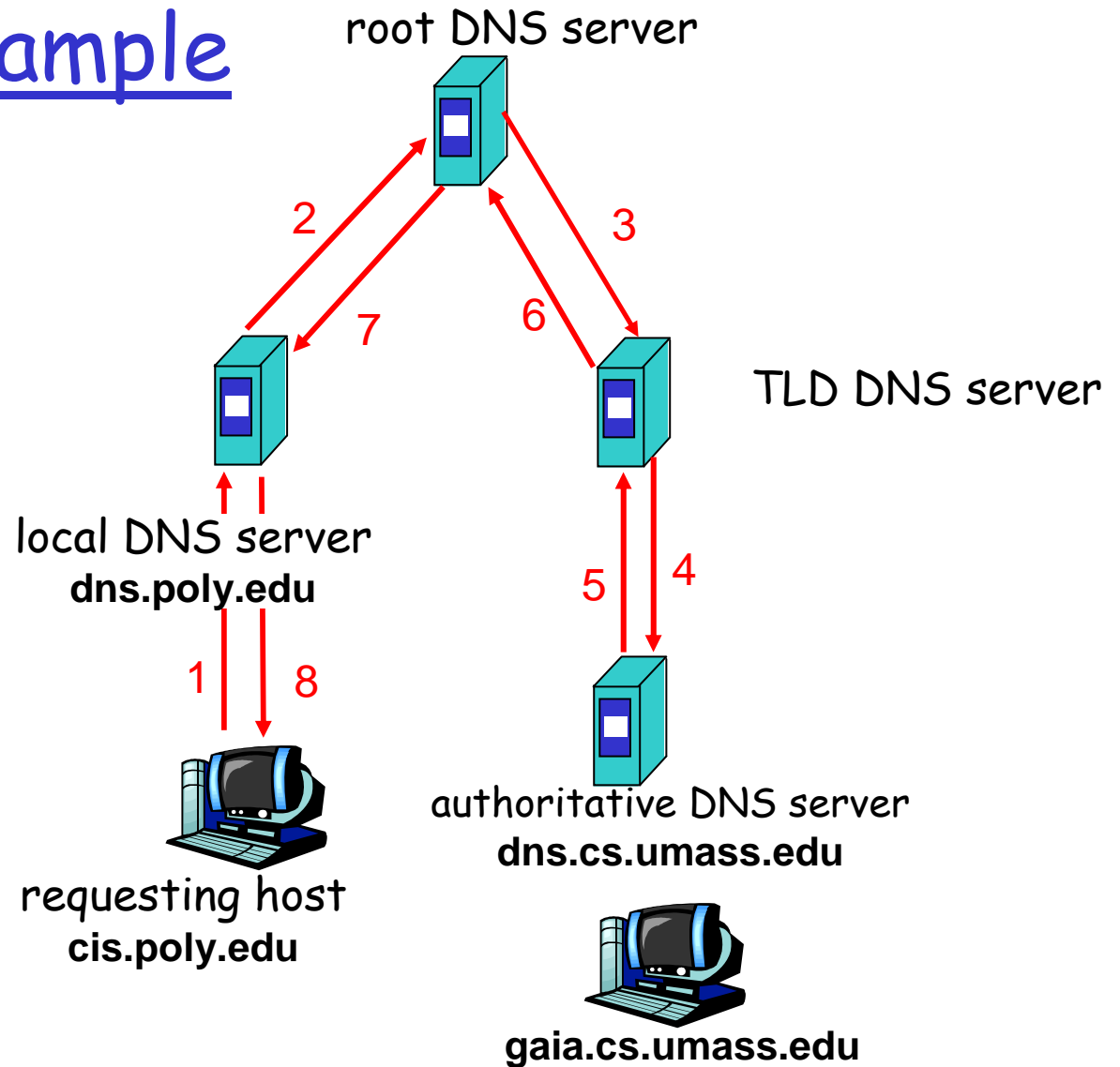
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



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

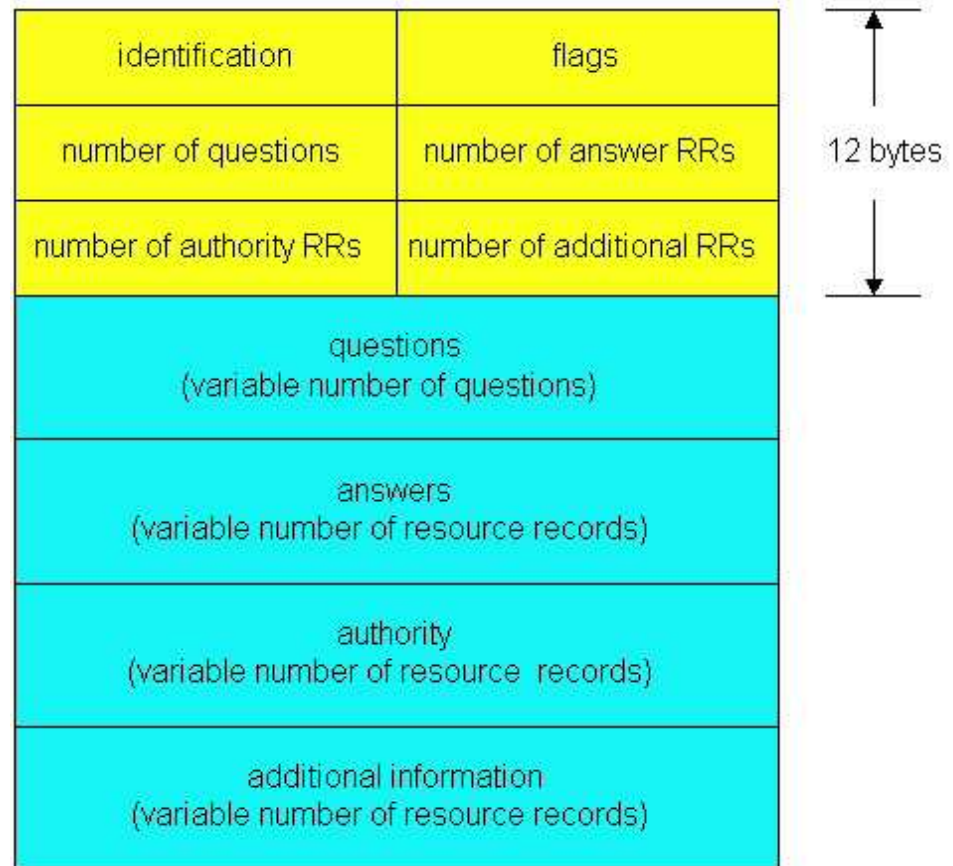
- ❖ value is name of mailserver associated with name

DNS protocol, messages

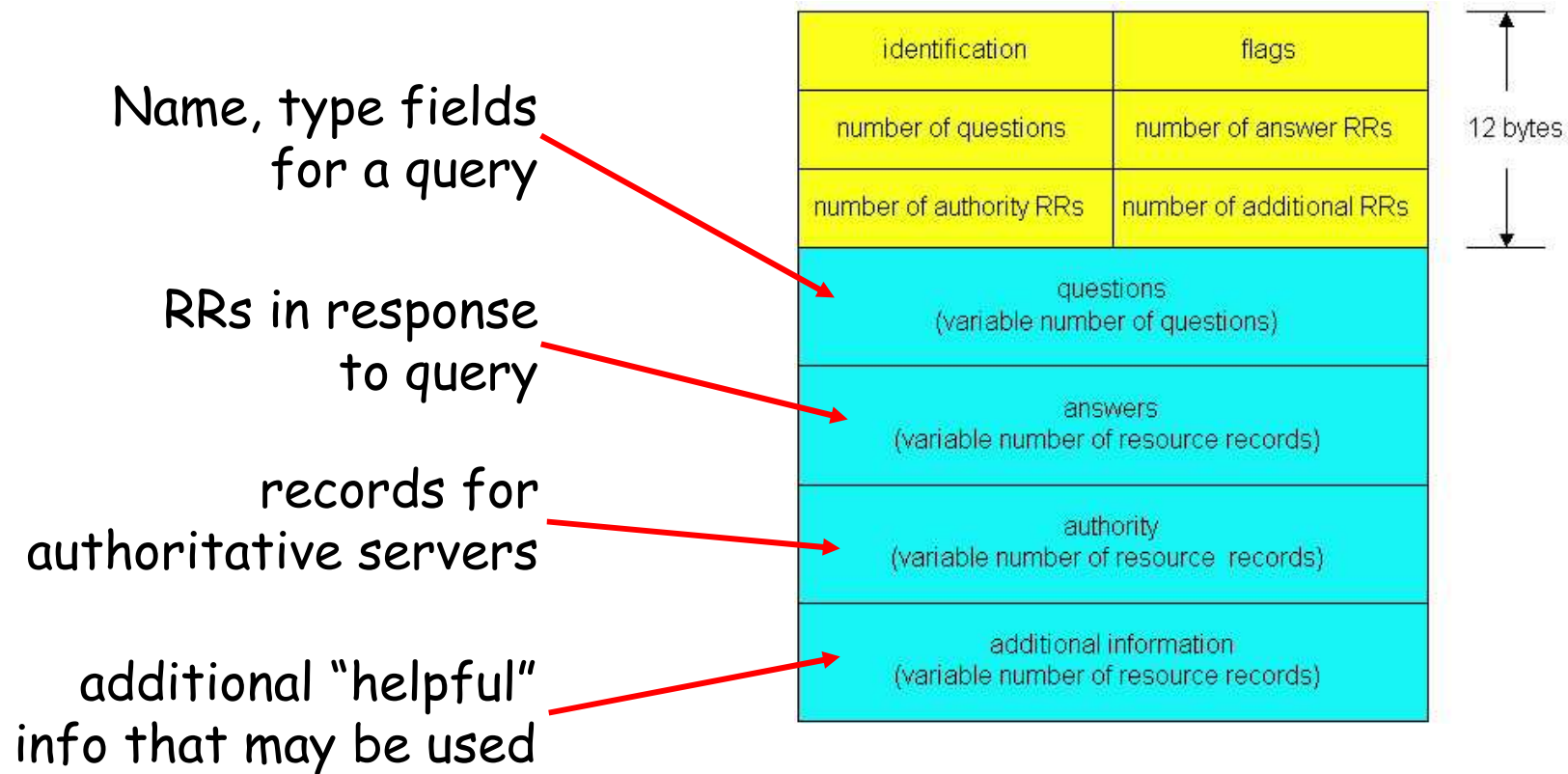
DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

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



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)

- In authoritative server, create Type A record for `www.networkutopia.com`; Type MX record for `networkutopia.com`
- *How do people get IP address of your Web site?*

Chapter 2: Summary

our study of network apps now complete!

- application architectures
 - ❖ client-server
 - ❖ P2P
 - ❖ hybrid
- application service requirements:
 - ❖ reliability, bandwidth, delay
- Internet transport service model
 - ❖ connection-oriented, reliable: TCP
 - ❖ unreliable, datagrams: UDP
- specific protocols:
 - ❖ HTTP
 - ❖ FTP
 - ❖ SMTP, POP, IMAP
 - ❖ DNS
 - ❖ P2P: BitTorrent, Skype
- socket programming

Chapter 2: Summary

Most importantly: learned about *protocols*

- typical request/reply message exchange:
 - ❖ client requests info or service
 - ❖ server responds with data, status code
- message formats:
 - ❖ headers: fields giving info about data
 - ❖ data: info being communicated

Important themes:

- control vs. data msgs
 - ❖ in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"