Network Applications – Part 2

Naming on the Internet • Email • Web • Caching

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 ?

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"
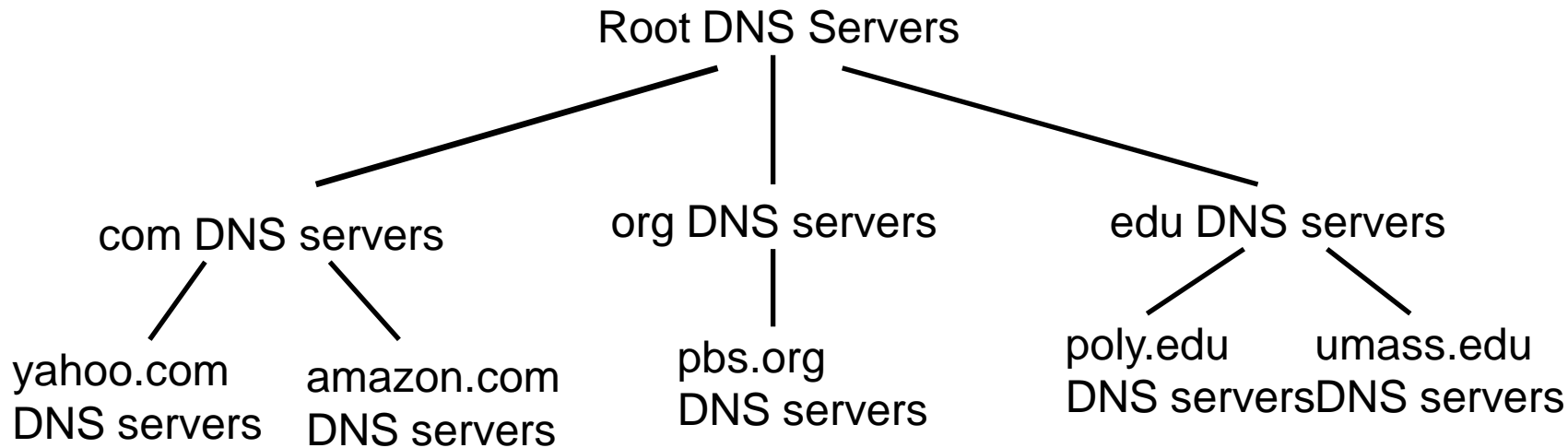
DNS…

DNS services used for

- Hostname to IP address translation
- Host aliasing
  - Canonical and 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

Root DNS Servers

com DNS servers | org DNS servers | edu DNS servers

yahoo.com DNS servers
amazon.com DNS servers
pbs.org DNS servers
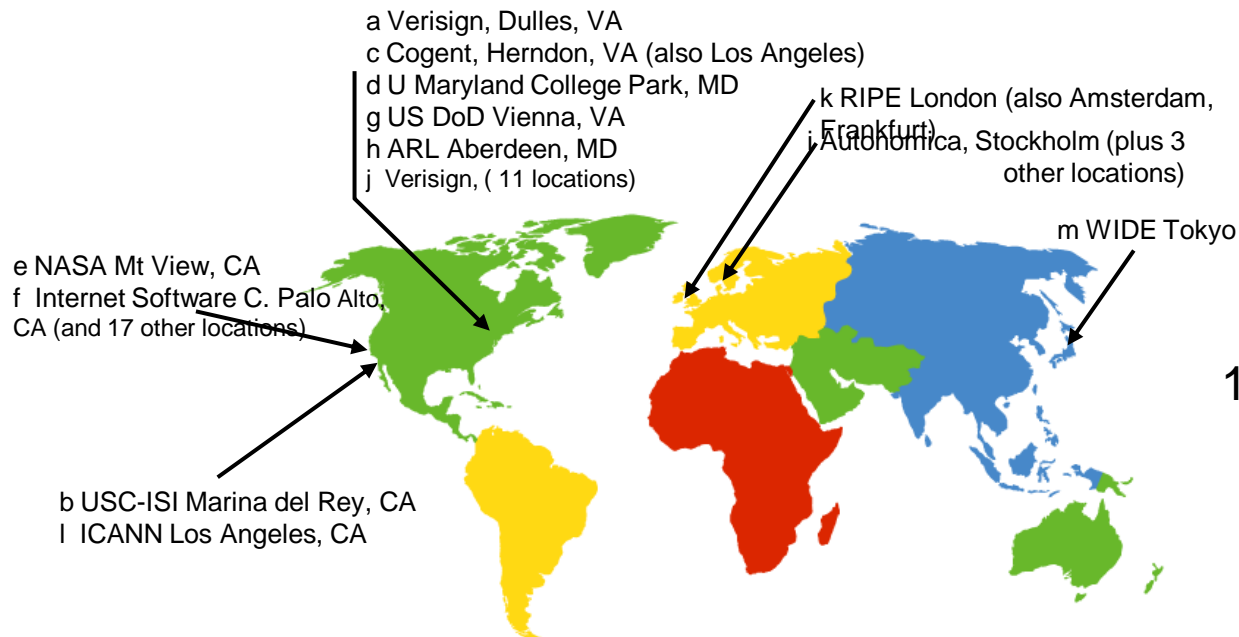poly.edu DNS servers
umass.edu DNS servers

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

a Verisign, Dulles, VA
c Cogent, Herndon, VA (also Los Angeles)
d U Maryland College Park, MD
g US DoD Vienna, VA
h ARL Aberdeen, MD
j Verisign, (11 locations)

k RIPE London (also Amsterdam, Frankfurt)
i Autonomica, Stockholm (plus 3 other locations)

m WIDE Tokyo

e NASA Mt View, CA
f Internet Software C. Palo Alto, CA (and 17 other locations)

b USC-ISI Marina del Rey, CA
l ICANN Los Angeles, CA

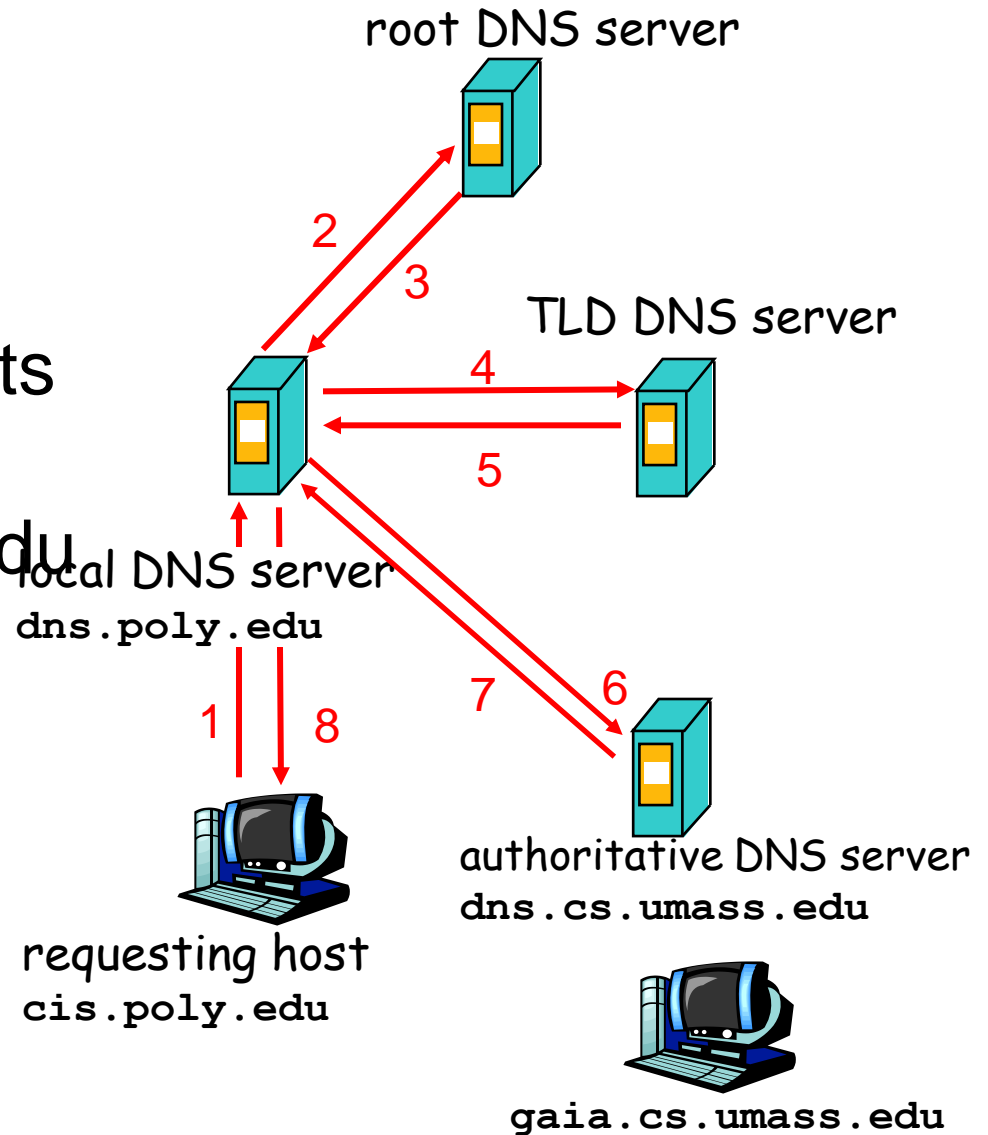13 root name servers worldwide

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 and 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 a host makes a DNS query, query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.

Example

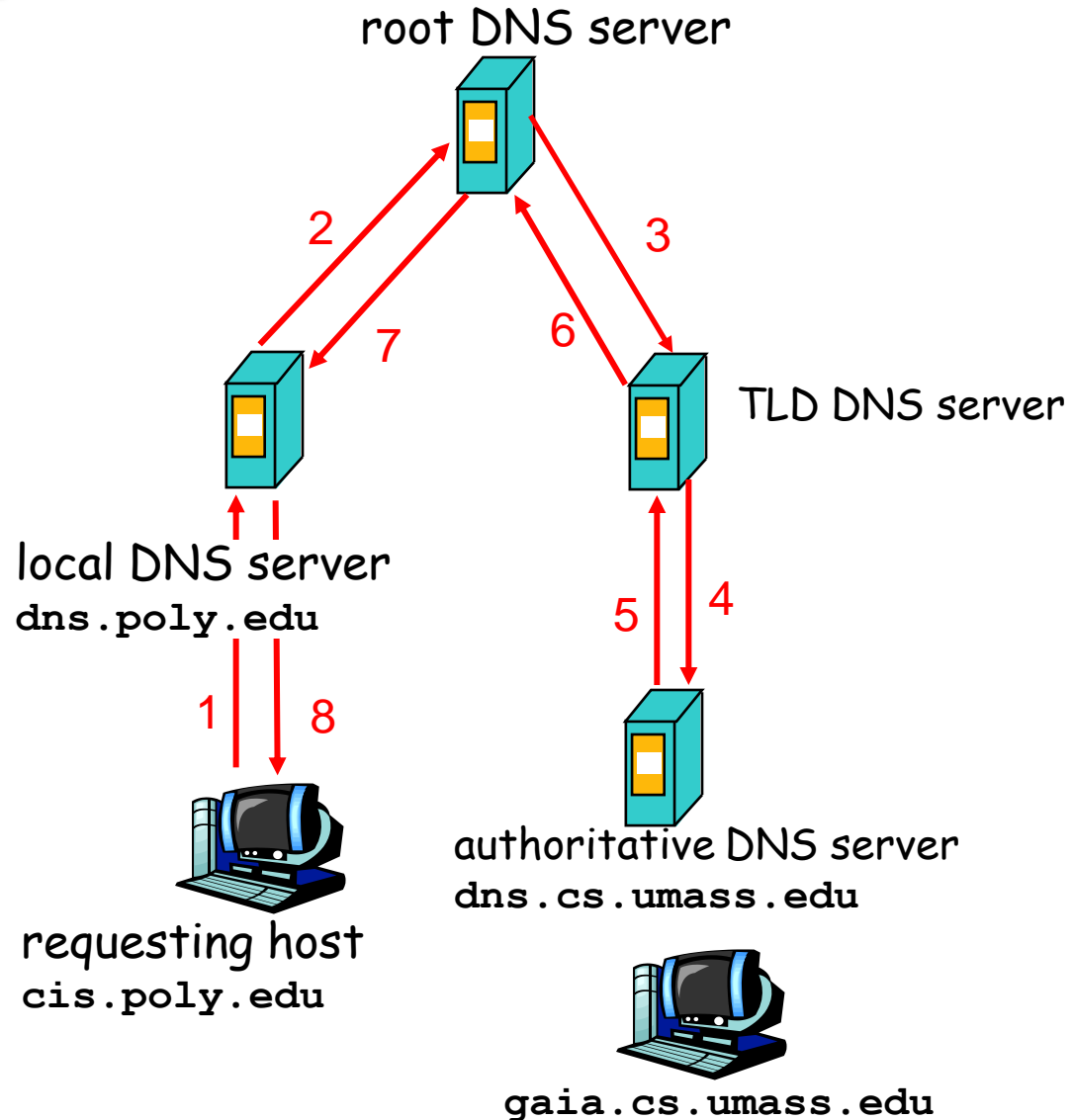- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

root DNS server

2

3

TLD DNS server

4

5

local DNS server
dns.poly.edu

1   8

7   6

requesting host
cis.poly.edu

authoritative DNS server
dns.cs.umass.edu

gaia.cs.umass.edu

Recursive queries

recursive query:

► puts burden of name resolution on contacted name server

► heavy load?

iterated query:

► contacted server replies with name of server to contact

► "I don't know this name, but ask this server"

root DNS server

2

3

7

6

TLD DNS server

local DNS server
`dns.poly.edu`

5

4

1

8

requesting host
`cis.poly.edu`

authoritative DNS server
`dns.cs.umass.edu`

`gaia.cs.umass.edu`

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 IP address of authoritative name server for this domain

► Type=CNAME
  ❖ **name** is alias name for some "cannonical" (the real) name
    `www.ibm.com` is really
    `servereast.backup2.ibm.com`
  ❖ **value** is cannonical name

► Type=MX
  ❖ **value** is name of mailserver associated with **name**

Resource Records

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

Resource Records Example

```
; Authoritative data for cs.vu.nl
cs.vu.nl.           86400   IN  SOA      star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.           86400   IN  TXT      "Divisie Wiskunde en Informatica."
cs.vu.nl.           86400   IN  TXT      "Vrije Universiteit Amsterdam."
cs.vu.nl.           86400   IN  MX       1 zephyr.cs.vu.nl.
cs.vu.nl.           86400   IN  MX       2 top.cs.vu.nl.

flits.cs.vu.nl.     86400   IN  HINFO    Sun Unix
flits.cs.vu.nl.     86400   IN  A        130.37.16.112
flits.cs.vu.nl.     86400   IN  A        192.31.231.165
flits.cs.vu.nl.     86400   IN  MX       1 flits.cs.vu.nl.
flits.cs.vu.nl.     86400   IN  MX       2 zephyr.cs.vu.nl.
flits.cs.vu.nl.     86400   IN  MX       3 top.cs.vu.nl.
www.cs.vu.nl.       86400   IN  CNAME    star.cs.vu.nl
ftp.cs.vu.nl.       86400   IN  CNAME    zephyr.cs.vu.nl

rowboat                     IN  A        130.37.56.201
                            IN  MX       1 rowboat
                            IN  MX       2 zephyr
                            IN  HINFO    Sun Unix

little-sister               IN  A        130.37.62.23
                            IN  HINFO    Mac MacOS

laserjet                    IN  A        192.31.231.216
                            IN  HINFO    "HP Laserjet IIISi" Proprietary
```

Inserting records into DNS

Potential Future Quiz Question:

- You just created a startup "Network Utopia" (domain: networkutopia.com). Give the update steps involved in setting up the DNS for authoritative name server and domain at the network registrar. Give the update steps you should do at the DNS server setup at networkutopia.com.
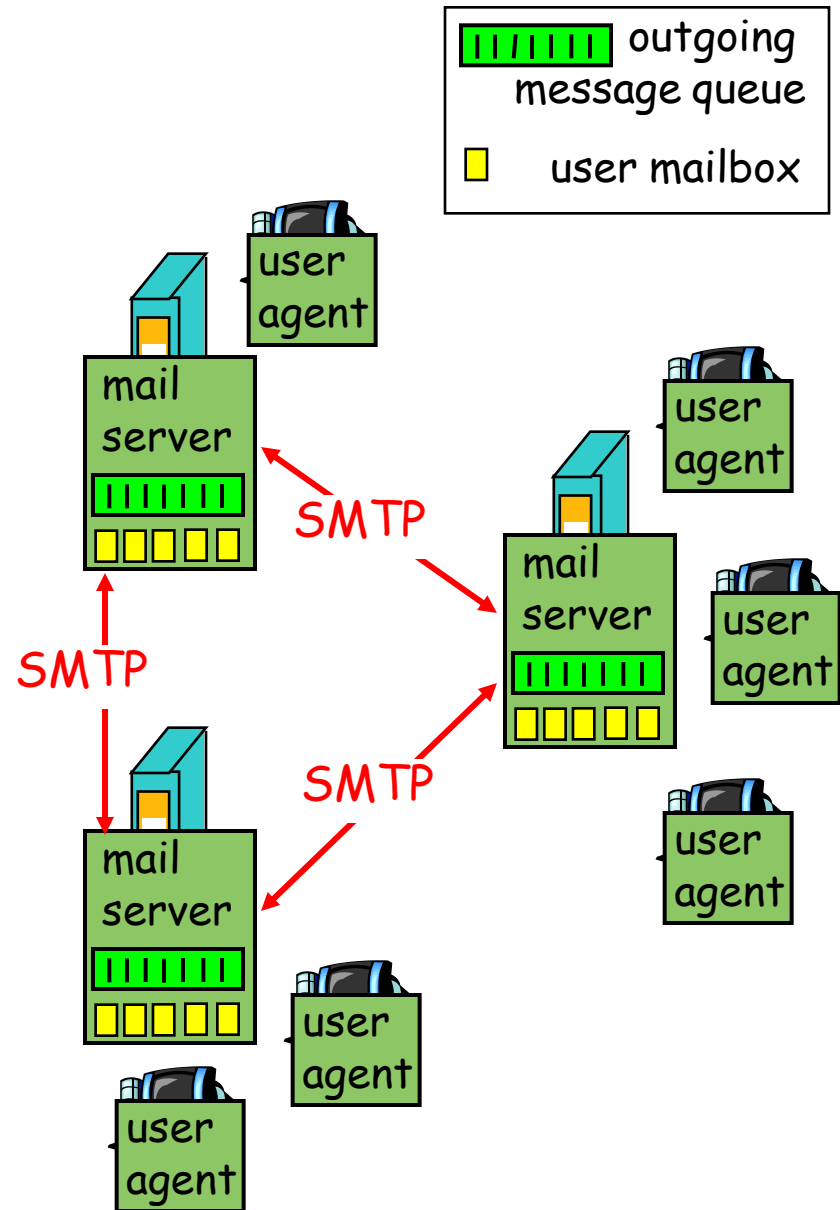
Electronic Mail

Three major components:

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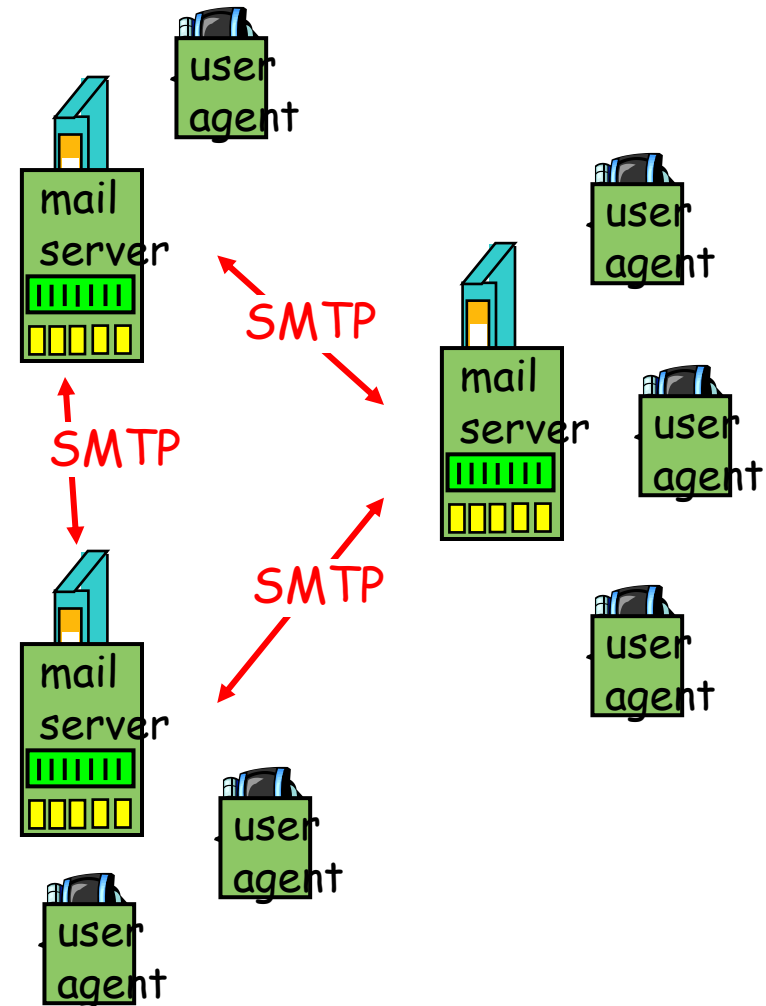- mail servers
- simple mail transfer protocol: SMTP

User Agent

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

outgoing message queue

user mailbox

SMTP

SMTP

SMTP

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
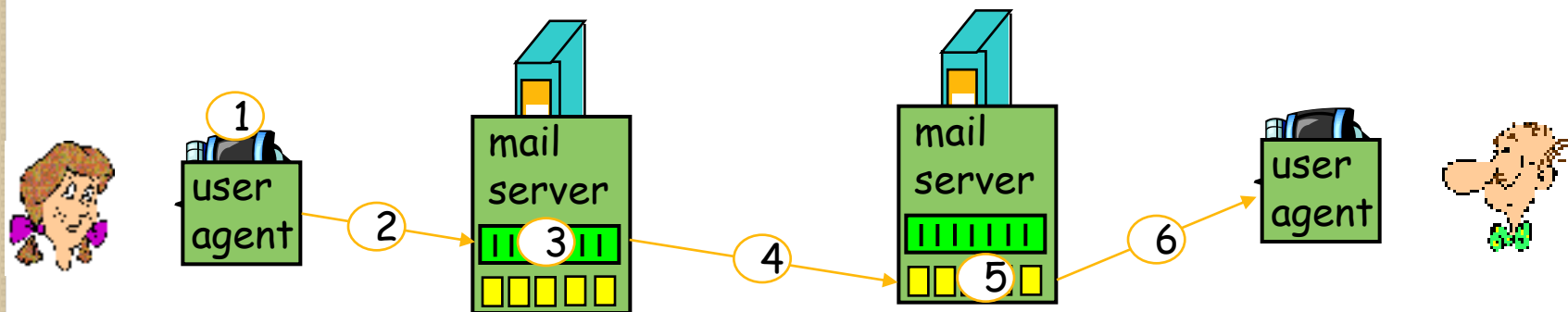
Web and HTTP

First some jargon

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
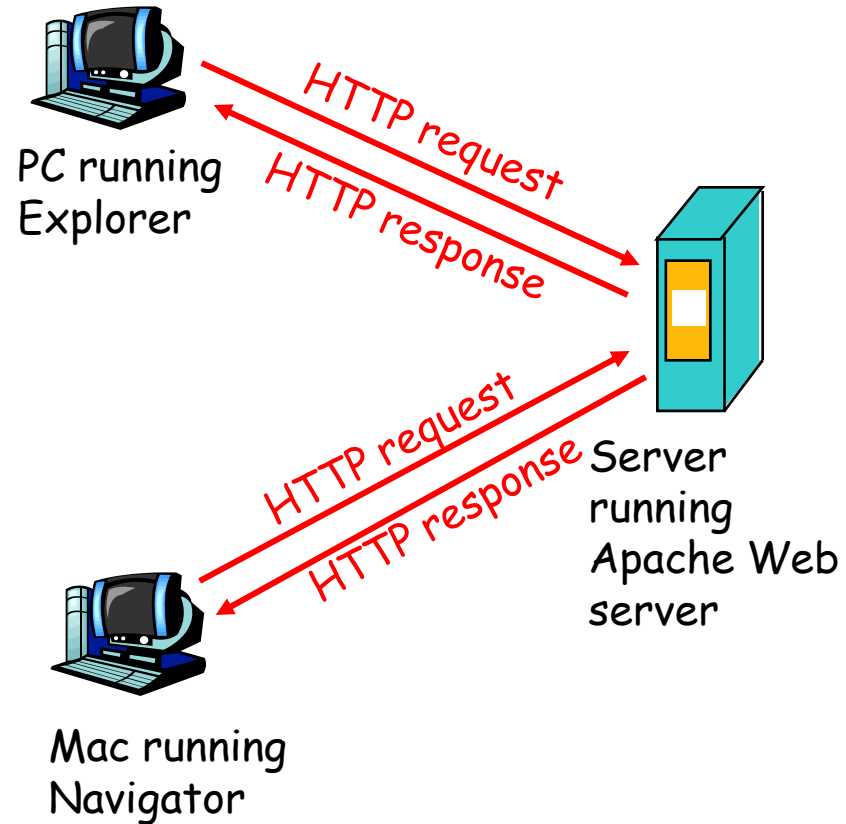- Example URL:

```
www.someschool.edu/someDept/pic.gif
```

host name · · · · · · · · · · · · · · · · path name

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

PC running Explorer

HTTP request

HTTP response

HTTP request

HTTP response

Server running Apache Web server

Mac running Navigator

HTTP overview (continued)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80

- server accepts TCP connection from client

- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)

- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

aside

Protocols that maintain "state" are complex!

► past history (state) must be maintained

► if server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Nonpersistent HTTP

Suppose user enters URL `www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client
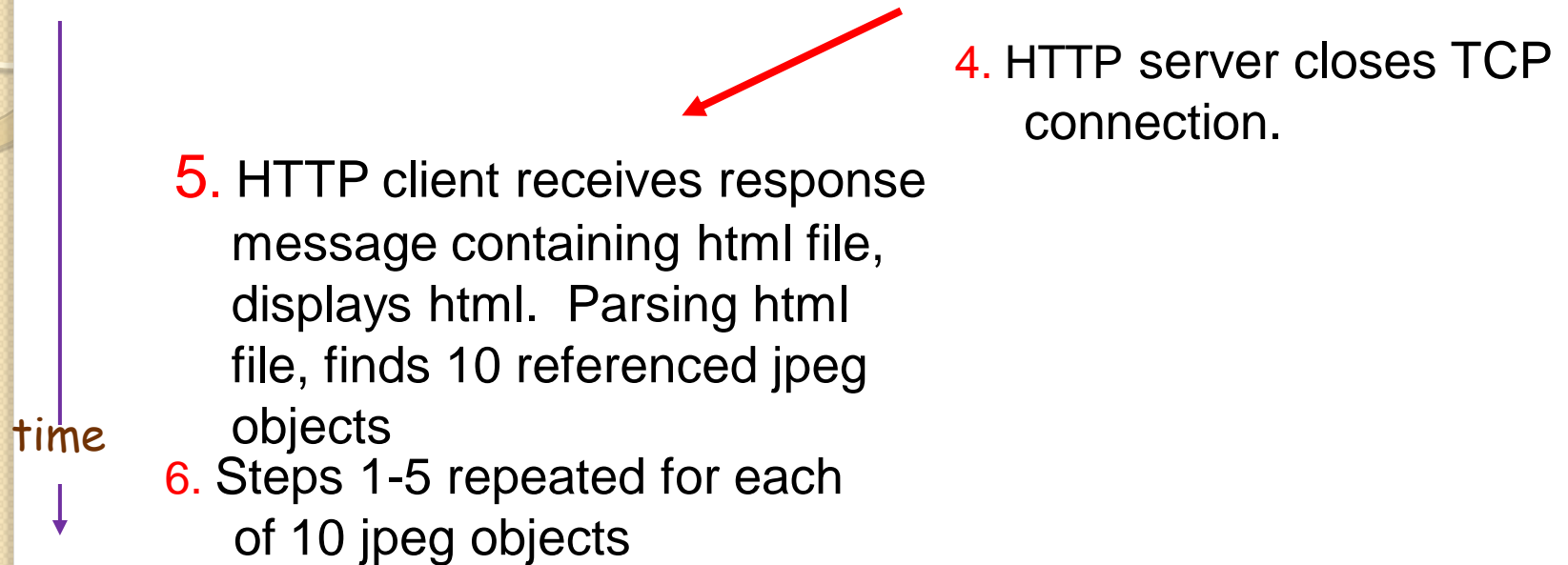
2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Nonpersistent HTTP (cont.)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

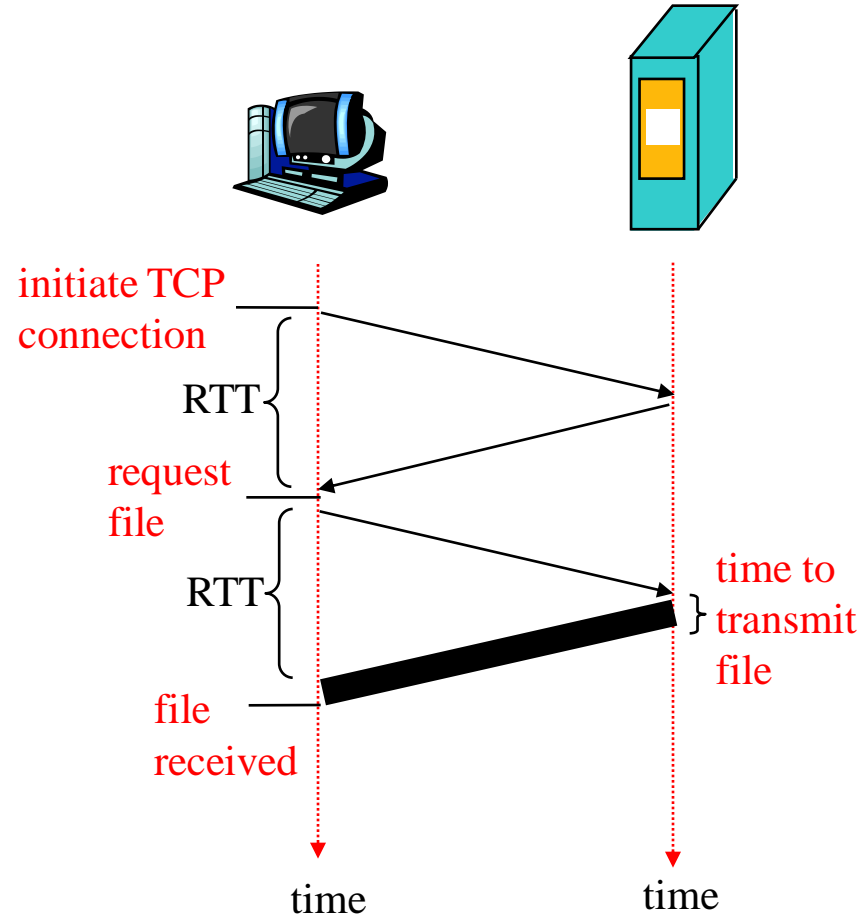**6.** Steps 1-5 repeated for each of 10 jpeg objects

time

Response time modeling

Definition of RTT: time to send a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time

Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- OS must allocate host resources for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server are sent over connection

Persistent without pipelining:

- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

- two types of HTTP messages: *request, response*

- HTTP request message:
  - ASCII (human-readable format)

request line
(GET, POST,
HEAD commands)

header
lines

Carriage return,
line feed
indicates end
of message

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

(extra carriage return, line feed)

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```
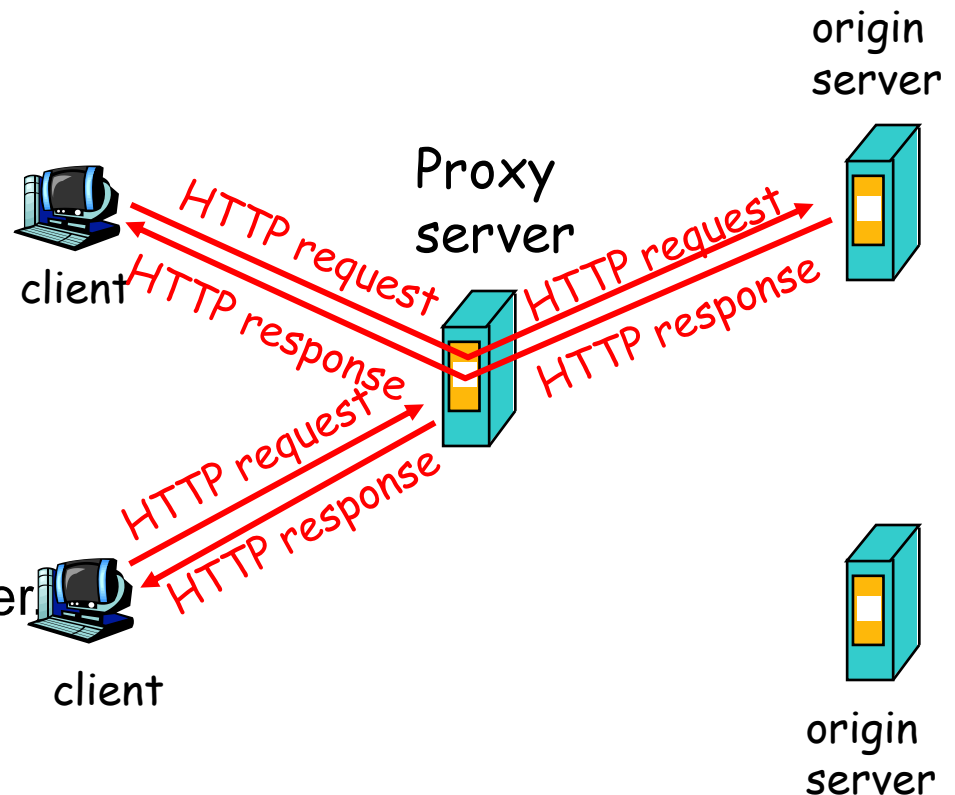
Web caches (proxy server)

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)

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

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
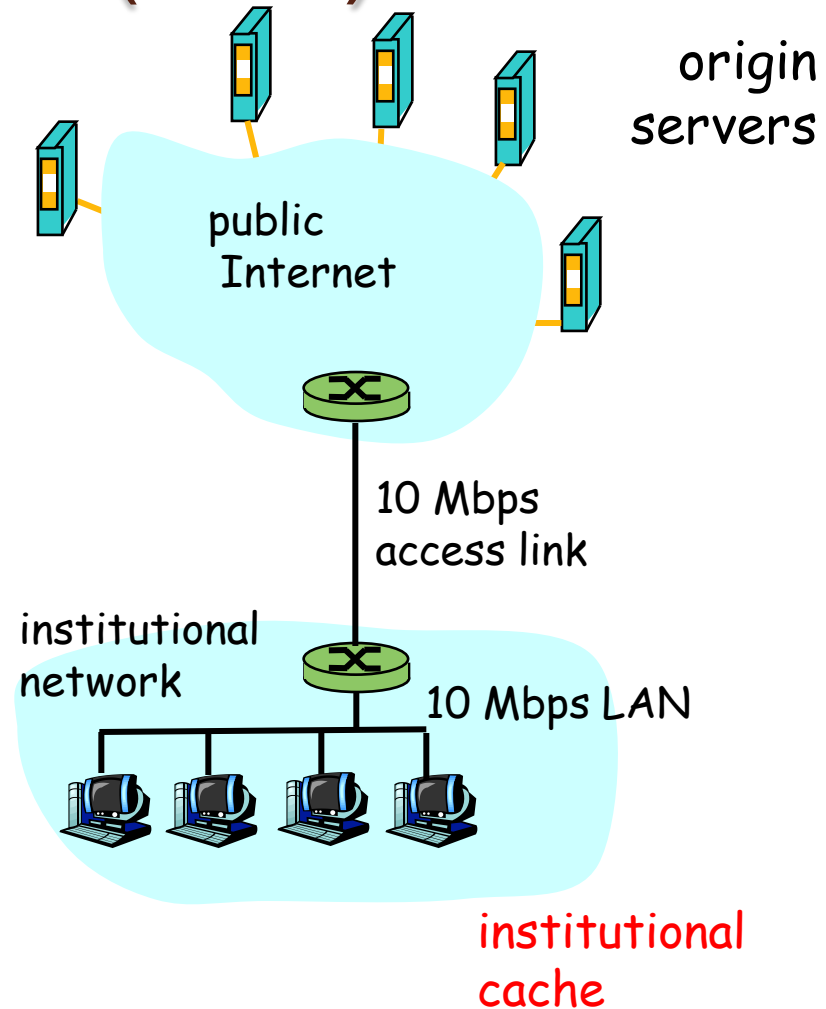
  = 2 sec + **minutes** + milliseconds

origin servers

public Internet

1.5 Mbps access link

institutional network

10 Mbps LAN

institutional cache
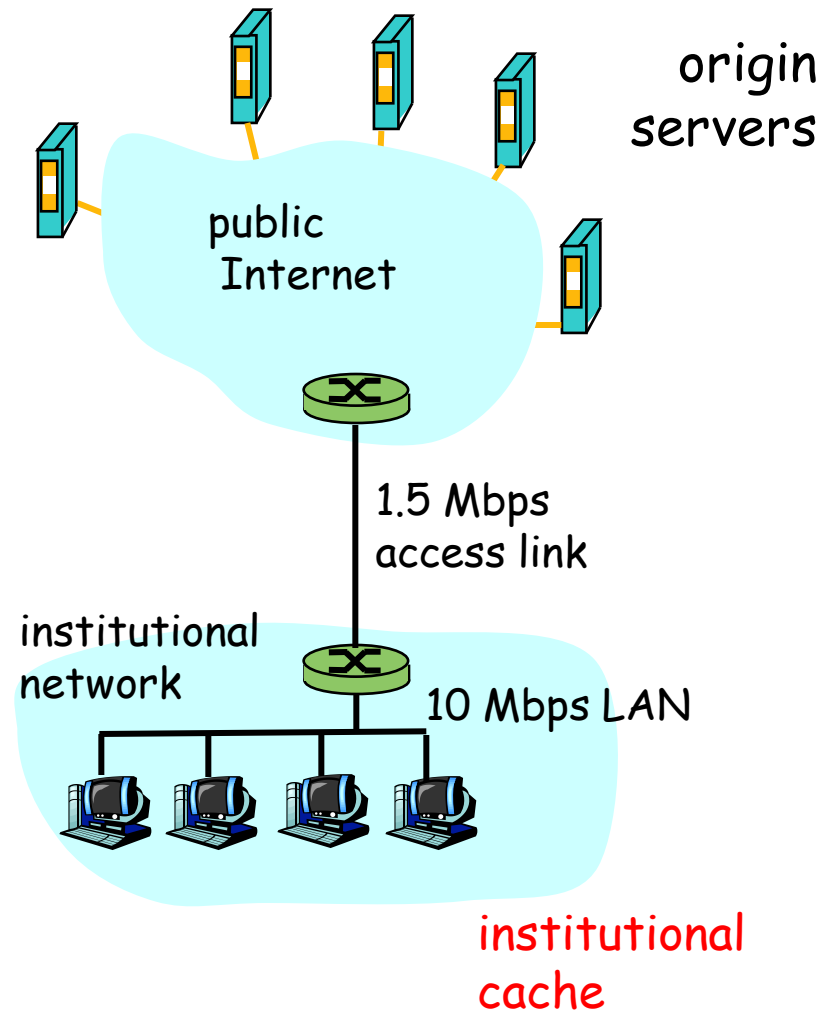
Caching example (cont)

Possible solution

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

Consequences

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs
- often a costly upgrade

origin servers

public Internet

10 Mbps access link

institutional network

10 Mbps LAN

institutional cache

Caching example (cont)

Install cache

- suppose hit rate is .4

Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by 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) secs + milliseconds < 1.4 secs