Networks and Distributed Systems

Lecture 21 – Applications (email and world wide web)

Problem

- Applications need their own protocols.
- These applications are
  - part network protocol
    - They exchange messages with their peers on other machines)
  - part traditional application program
    - They interact with the windowing system, the file system, and ultimately, the user).
- We will explores some of the most popular network applications available today.

Traditional Applications

- Two of the most popular
  - The World Wide Web and
  - Email.

- Broadly speaking, both of these applications use the request/reply paradigm
  - users send requests to servers, which then respond accordingly.

Traditional Applications

- It is important to distinguish between application *programs and application protocols.*
- For example, *the HyperText Transport Protocol* (HTTP) is an application protocol that is used to retrieve Web pages from remote servers.
- There can be many different application programs (Internet Explorer, Chrome, Firefox, and Safari) that provide users with a different look and feel
  - but all of them use the same HTTP protocol to communicate with Web servers over the Internet.

Traditional Applications

- Two very widely-used, standardized application protocols:
  - SMTP: Simple Mail Transfer Protocol is used to exchange electronic mail.
  - HTTP: HyperText Transport Protocol is used to communicate between Web browsers and Web servers.

Electronic Mail (SMTP, MIME, IMAP)

- Email is one of the oldest network applications

- It is important:
  - to distinguish the user interface (i.e., your mail reader) from the underlying message transfer protocols (such as SMTP or IMAP)

  - to distinguish between this transfer protocol and a companion protocol (RFC 822 and MIME) that defines the format of the messages being exchanged

Message Format

- RFC 822 defines messages to have two parts: a *header and a body. Both parts are* represented in ASCII text.
- Originally, the body was assumed to be simple text.
  - This is still the case, although RFC 822 has been augmented by MIME to allow the message body to carry all sorts of data.
- This data is still represented as ASCII text, but because it may be an encoded version of, say, a JPEG image, it's not necessarily readable by human users.
- The message header is a series of <CRLF>-terminated lines. (<CRLF> stands for carriage-return+ line-feed, which are a pair of ASCII control characters often used to indicate the end of a line of text.)

Message Format

- The header is separated from the message body by a blank line.
  - Each header line contains a type and value separated by a colon.
- Many of these header lines are familiar to users since they are asked to fill them out when they compose an email message.
- RFC 822 was extended in 1993 (and updated quite a few times since then) to allow email messages to carry many different types of data: audio, video, images, PDF documents, and so on.

Multipurpose Internet Mail Extensions

- MIME consists of three basic pieces.
  - A collection of header lines that augment the original set defined by RFC 822. They describe the data being carried in the body.
    - E.g. MIME-Version: (the version of MIME being used)
    - Content-Description: (a human-readable description of what's in the message, analogous to the Subject: line)
    - Content-Type: (the type of data contained in the message)
    - Content-Transfer- Encoding (how the data in the message body is encoded).
  - Definitions for a set of content types (and subtypes). For example, MIME defines two different still image types, denoted image/gif and image/jpeg, each with the obvious meaning.
  - A way to encode the various data types so they can be shipped in an ASCII email message.

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <Alice@cisco.com>
To: Bob@cs.Princeton.edu
Subject: promised material
Date: Mon, 07 Sep 1998 19:45:19 -0400

---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Bob,

Here's the jpeg image and draft report I promised.

--Alice

---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
... unreadable encoding of a jpeg figure

---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit
... readable encoding of a PostScript document
```

Message Transfer

- For many years, the majority of email was moved from host to host using only SMTP.
- While SMTP continues to play a central role, it is now just one email protocol of several
- IMAP and POP being two other important protocols for retrieving mail messages.
  - Internet Message Access Protocol (IMAP)
  - Post Office Protocol (POP)

Message Transfer

- To place SMTP in the right context, we need to identify the key players.

- First, users interact with a *mail reader when they compose, file, search, and read their email.*
  - *There* are countless mail readers available, just like there are many Web browsers to choose from.
  - In the early days, users logged into their mailbox machine*, and invoked a local application (mail reader)* that extracted messages from the file system.
  - Today, users remotely access their mailbox from their laptop or smartphone

Message Transfer

- Second, there is a *mail daemon (or process) running on each host that holds a mailbox.*

- This is called a Message Transfer Agent (MTA). It plays the role of a post office
  - Users give the daemon messages they want to send to other users
  - The daemon uses SMTP running over TCP to transmit the message to a daemon running on another machine
  - The daemon puts incoming messages into the user's mailbox (where that user's mail reader can later find it).

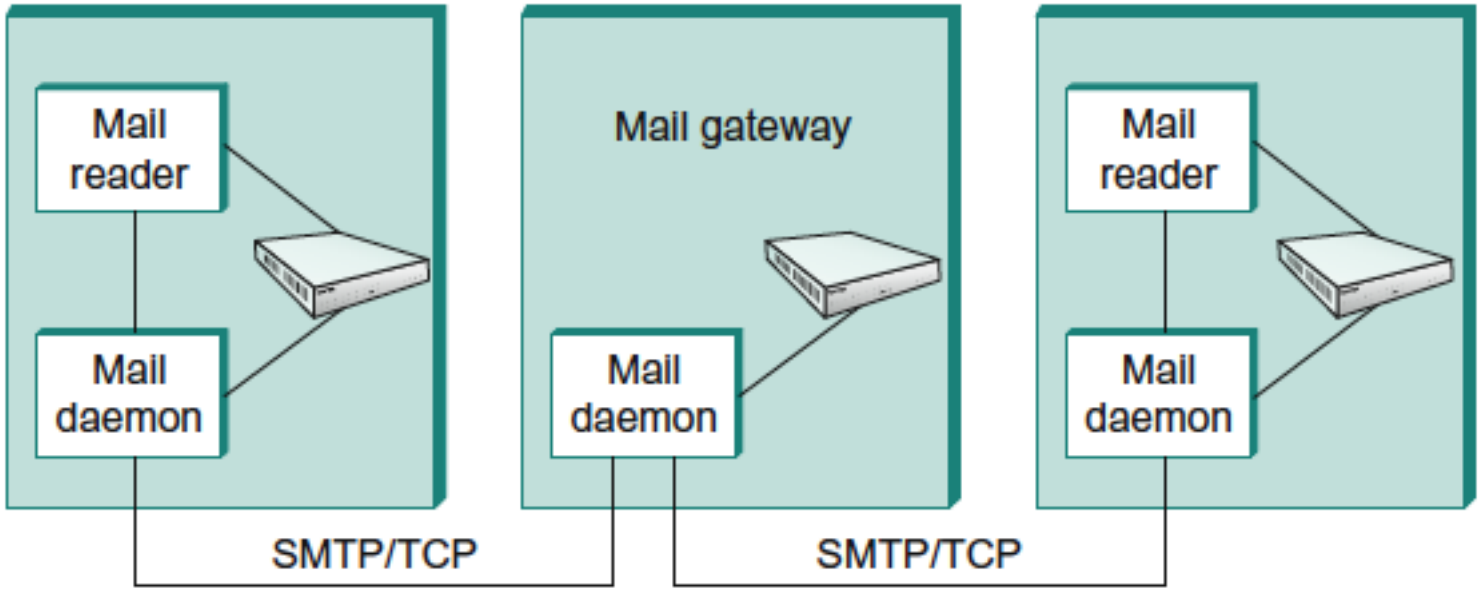
Message Transfer

- While it is certainly possible that the MTA on a sender's machine establishes an SMTP/TCP connection to the MTA on the recipient's mail server
  - In many cases the mail traverses one or more *mail gateways on its route from the sender's host to the* receiver's host.

- Like the end hosts, these gateways also run a message transfer agent process. They are called "gateways" since their job is to store and forward email messages, much like an "IP gateway"

- The only difference is that a mail gateway typically buffers messages on disk and is willing to try retransmitting them to the next machine for several days

Message Transfer



SMTP example

- The following is an exchange between sending host cs.princeton.edu and receiving host cisco.com.
- In this case, user Bob at Princeton is trying to send mail to users Alice and Tom at Cisco.
- The lines sent by cs.princeton.edu are shown in black and the lines sent by cisco.com are shown in teal.
- Extra blank lines have been added to make the dialog more readable.

SMTP example

```
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]

MAIL FROM:<Bob@cs.princeton.edu>
250 OK

RCPT TO:<Alice@cisco.com>
250 OK

RCPT TO:<Tom@cisco.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
...etc. etc. etc.

<CRLF>.<CRLF>
250 OK

QUIT
221 Closing connection
```

Mail Reader

- The final step is for the user to actually:
  - Retrieve his or her messages from the mailbox,
  - Read them, reply to them
  - Possibly save a copy for future reference.
- The user performs all these actions by interacting with a mail reader.
- As pointed out earlier, this reader was originally just a program running on the same machine as the user's mailbox,
- This was the common case in the pre-laptop era.

Mail Reader

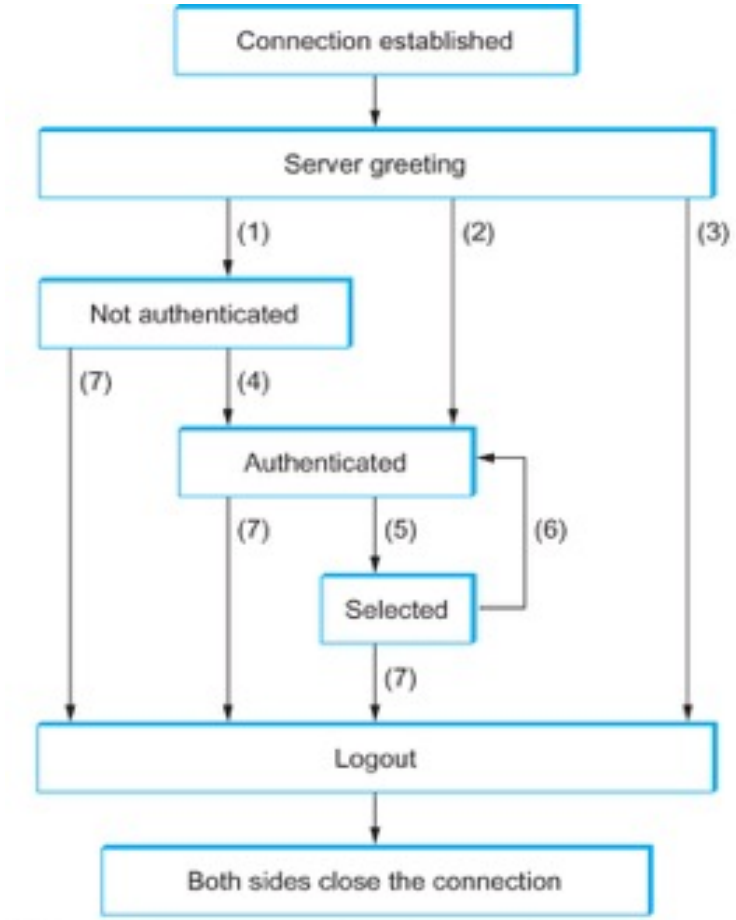
- Today, most often the user accesses his or her mailbox from a remote machine using yet another protocol
  - Such as the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP).

- It is beyond the scope to discuss the user interface aspects of the mail reader, but it is definitely within our scope to talk about the access protocol.

IMAP

- IMAP is similar to SMTP in many ways.
  - It is a client/server protocol running over TCP
  - The client (running on the user's desktop machine) issues commands in the form of <CRLF>-terminated ASCII text lines
  - The mail server (running on the machine that maintains the user's mailbox) responds in-kind.
- The exchange begins with the client authenticating him or herself, and identifying the mailbox he or she wants to access.

IMAP state transition diagram



(1) Connection without preauthentication (OK greeting)
(2) Preauthenticated connection (PREAUTH greeting)
(3) Rejected connection (BYE greeting)
(4) Successful LOGIN or AUTHENTICATE command
(5) Successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

World Wide Web

- The World Wide Web has been so successful and has made the Internet accessible to so many people that sometimes it seems to be synonymous with the Internet.
- In fact, the design of the system that became the Web started around 1989, long after the Internet had become a widely deployed system.
- The original goal of the Web was to find a way to organize and retrieve information
  - drawing on ideas about hypertext—interlinked documents—that had been around since at least the 1960s.

World Wide Web

- The core idea of hypertext is that one document can link to another document, and the protocol (HTTP) and document language (HTML) were designed to meet that goal.
- One helpful way to think of the Web is as a set of cooperating clients and servers, all of whom speak the same language: HTTP.
- Most people are exposed to the Web through a graphical client program, or Web browser, like Safari, Chrome, Firefox or Internet Explorer.

World Wide Web

- Clearly, if you want to organize information into a system of linked documents or objects, you need to be able to retrieve one document to get started.
- Hence, any Web browser has a function that allows the user to obtain an object by "opening a URL."
- URLs (Uniform Resource Locators) are so familiar to most of us by now
- They provide information that allows objects on the Web to be located, and they look like the following:
  - http://www.cs.princeton.edu/index.html

World Wide Web

- If you opened that particular URL:
  - Your Web browser would open a TCP connection to the Web server at a machine called www.cs.princeton.edu
  - And immediately retrieve and display the file called index.html.
- Most files on the Web contain images and text and many have other objects such as audio and video clips, pieces of code, etc.
- They also frequently include URLs that point to other files that may be located on other machines, which is the core of the “hypertext” part of HTTP and HTML.

World Wide Web

- When you ask your browser to view a page, your browser (the client) fetches the page from the server using HTTP running over TCP.

- Like SMTP, HTTP is a text oriented protocol.
- At its core, HTTP is a request/response protocol, where every message has the general form:

START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
MESSAGE_BODY <CRLF>

- The first line (START LINE) indicates whether this is a request message or a response message.

Request Messages

- The first line of an HTTP request message specifies three things:
  - the operation to be performed
  - the Web page the operation should be performed on
  - the version of HTTP being used.

- Although HTTP defines a wide assortment of possible request operations, the two most common operations are:
  - GET (fetch the specified Web page)
  - HEAD (fetch status information about the specified Web page).

Request Messages

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

HTTP request operations

Response Messages

- Like request messages, response messages begin with a single START LINE.

- In this case, the line specifies:
  - the version of HTTP being used
  - a three-digit code indicating whether or not the request was successful
  - a text string giving the reason for the response.

Response Messages

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

Five types of HTTP result codes

Example: www.nyuad.nyu.edu

HTTP request

```
GET / HTTP/1.1
Host: www.nyuad.nyu.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.86 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
```

HTTP response

```
HTTP/1.1 301 Moved Permanently
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 27 Apr 2016 04:26:46 GMT
Location: http://nyuad.nyu.edu/
Server: Apache/2.2.15 (Red Hat)
Content-Length: 229
Connection: keep-alive
```

TCP Connections

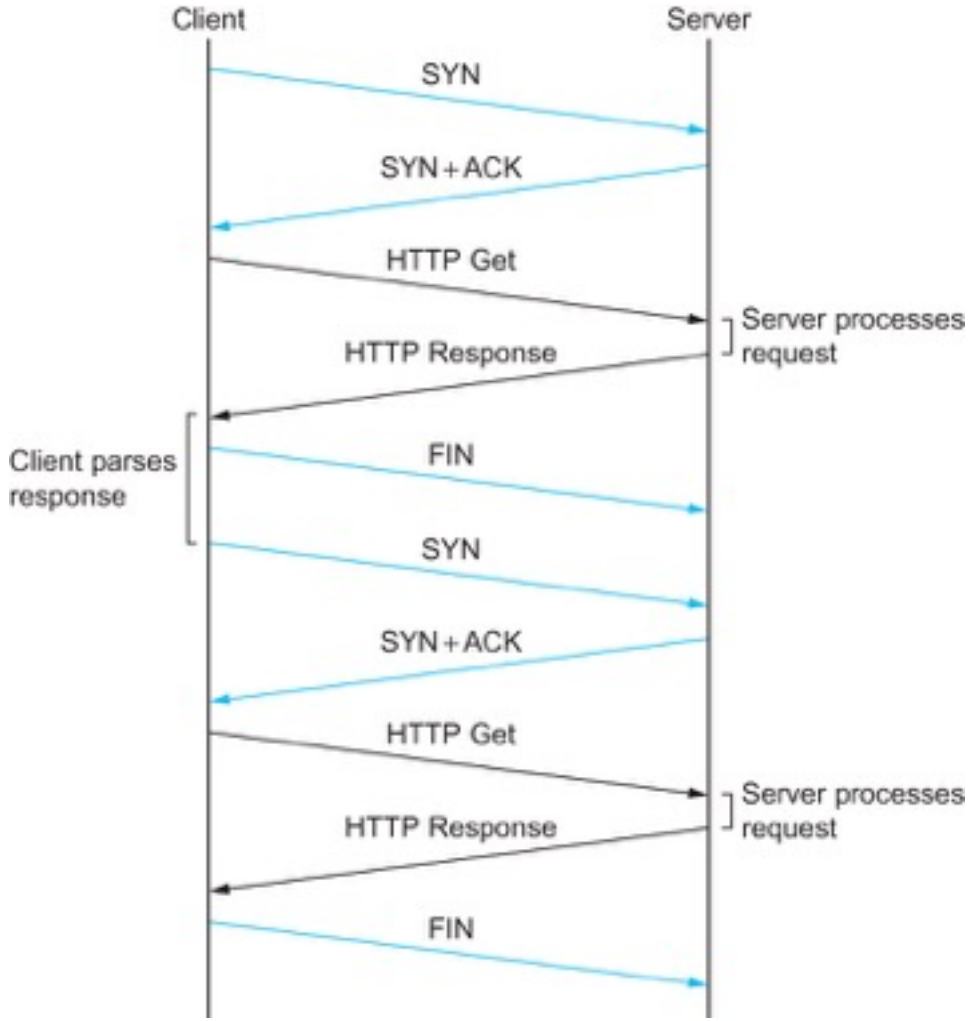
- The original version of HTTP (1.0) established a separate TCP connection for each data item retrieved from the server.
- It's not too hard to see how this was a very inefficient mechanism:
  - connection setup and teardown messages had to be exchanged between the client and server even if the client just wanted to verify that it had the most recent copy of a page.
- Thus, retrieving a page that included some text and a dozen icons or other small graphics would result in multiple separate TCP connections being established and closed.

TCP Connections

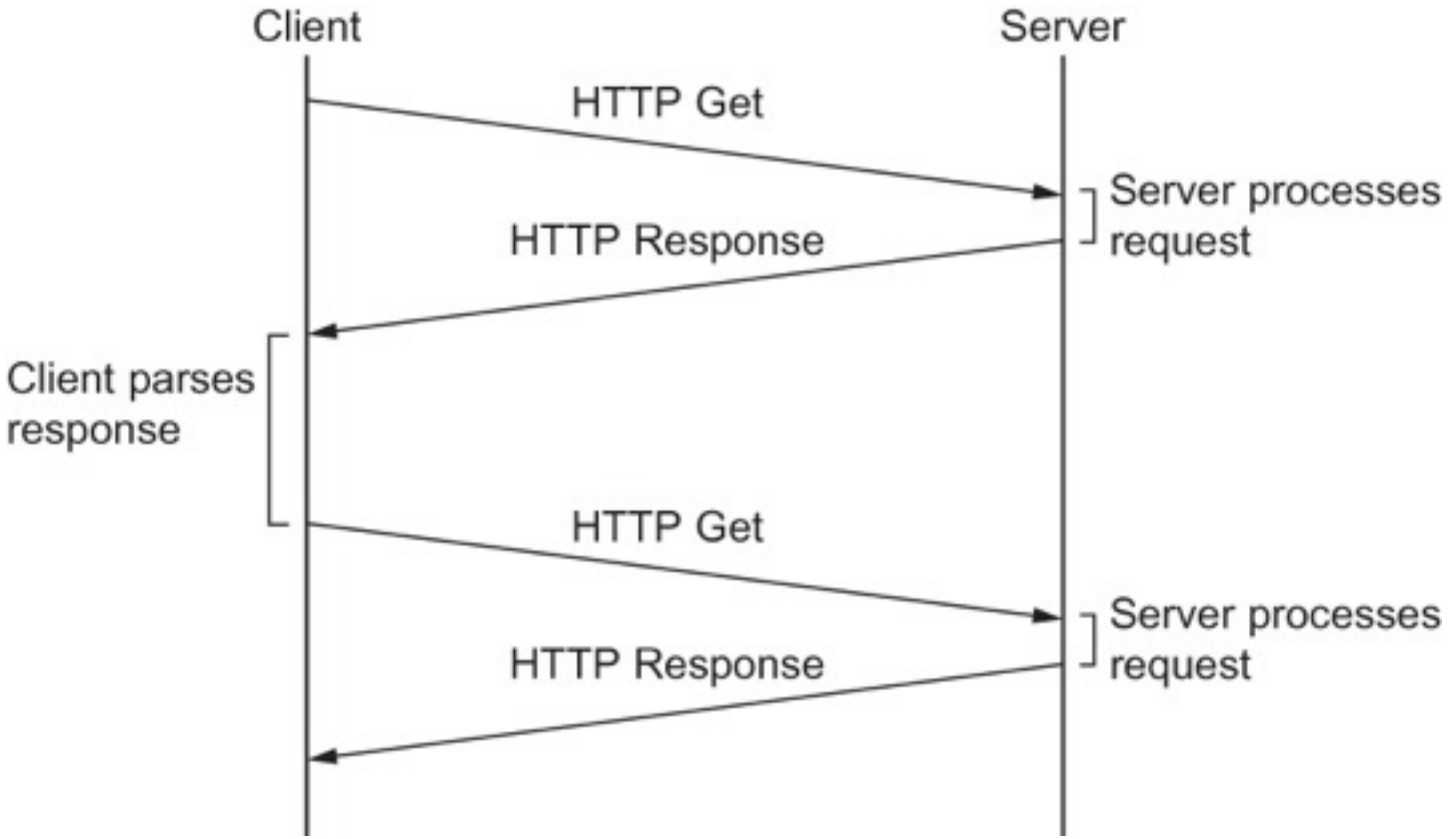
- To overcome this situation, HTTP version 1.1 introduced *persistent connections*
  - The client and server can exchange multiple request/response messages over the same TCP connection.

- Persistent connections have many advantages.
  - They eliminate the connection setup overhead
  - Reduce the load on the server,
  - Reduce the load on the network by the additional TCP packets
  - Reduce the delay perceived by the user.
  - TCP's congestion window mechanism is able to operate more efficiently. (not necessary to go through the slow start for each page)

HTTP 1.0 behavior



HTTP 1.1 (persistent connections)



Caching

- One of the most active areas of research (and entrepreneurship) in the Internet today is how to effectively cache Web pages.

- Caching has many benefits:
  - Client's perspective: a page can be retrieved from a nearby cache, thus can be displayed much more quickly than if it has to be fetched from across the world.

  - Server's perspective: having a cache intercept and satisfy a request reduces the load on the server.

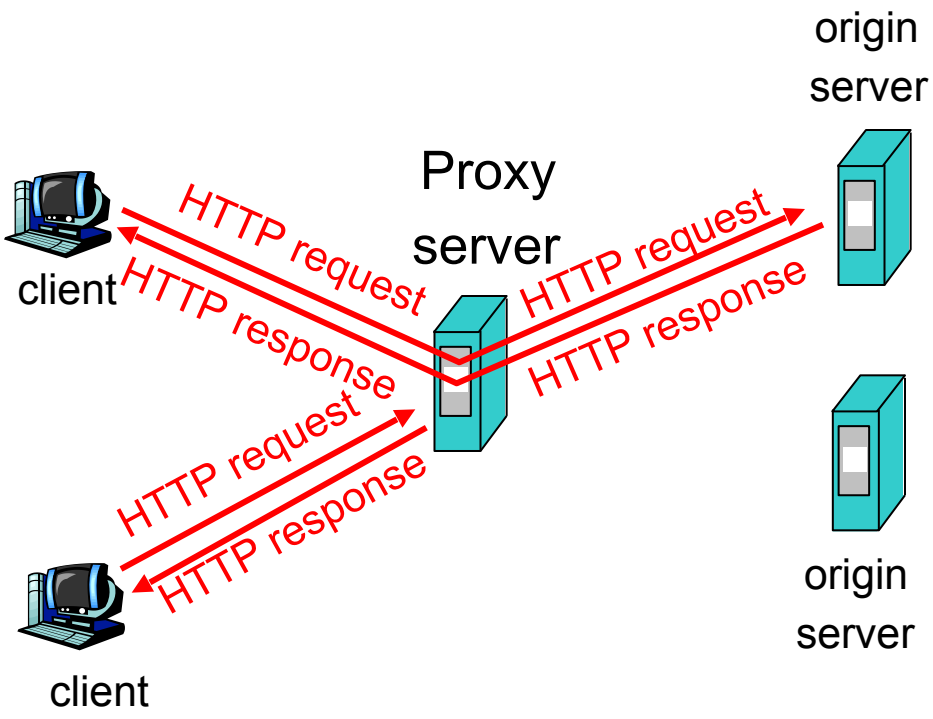
Caching

- Caching can be implemented in many different places.
  - A user's browser can cache recently accessed pages, and simply display the cached copy if the user visits the same page again.
  - A site can support a single site-wide cache. This allows users to take advantage of pages previously downloaded by other users.
  - Closer to the middle of the Internet, ISPs can cache pages.

- In the second case above, the users within the site most likely know what machine is caching pages on behalf of the site, and they configure their browsers to connect directly to the caching host. This node is sometimes called a *proxy*

Web Proxy Caches

- User configures 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



No Caching Example (1)

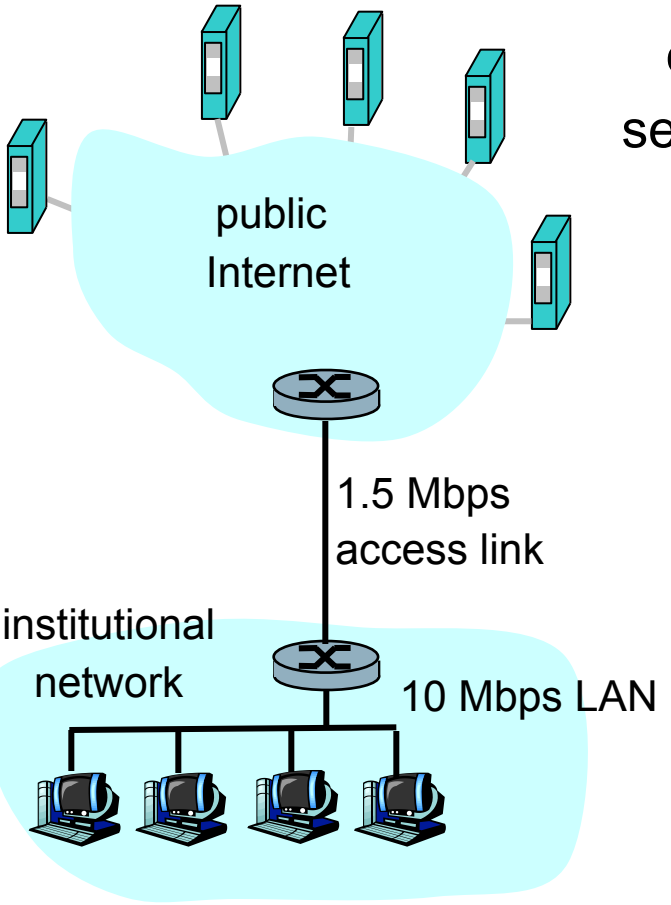
Assumptions

- Average object size = 100,000 bits
- Avg. request rate from institution's browser to origin servers = 16/sec
- Delay from institutional router to any origin server and back to router = 2 sec

Consequences

- Utilization on LAN = 16%
- Utilization on access link > 100%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + seconds + milliseconds



No Caching Example (2)

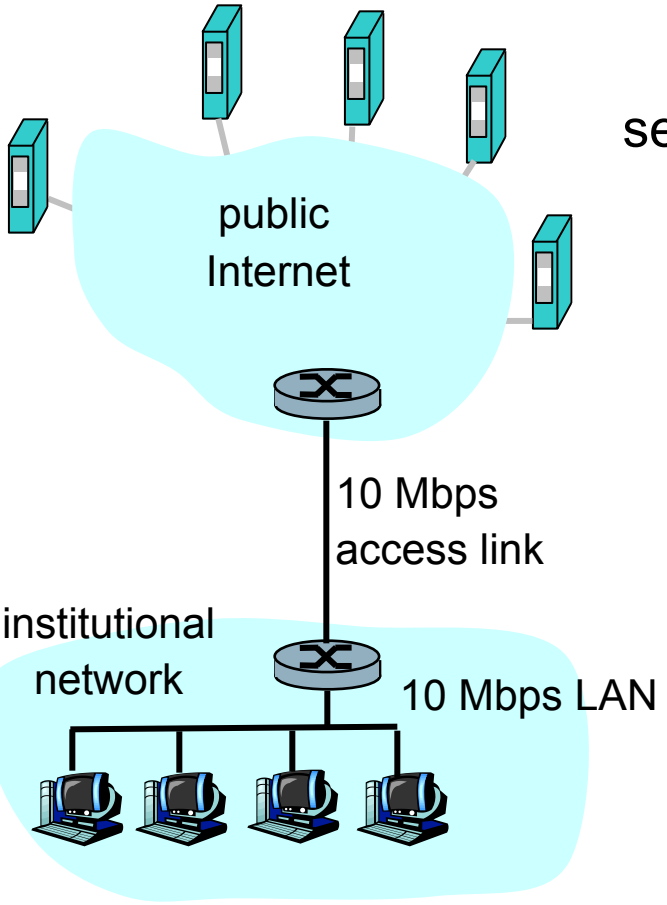
Possible solution

- Increase bandwidth of access link to, say, 10 Mbps
- Often a costly upgrade

Consequences

- Utilization on LAN = 16%
- Utilization on access link = 16%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs



W/Caching Example (3)

Install cache

- Suppose hit rate is 0.4

Consequence

- 40% requests will be satisfied almost immediately (say 10 msec)
- 60% requests satisfied by origin server
- Utilization of access link reduced to 60%, resulting in negligible delays
- Weighted average of delays

= 0.6*2 sec + 0.4*10msecs < 1.3 secs