2. Introduction to Network Applications and the Web

- Application architectures
- Transport services
- Browser and web server interaction
  - Cookies
  - Web caches and conditional GET

From Jon Turner – based on material from Kurose and Ross

Goals

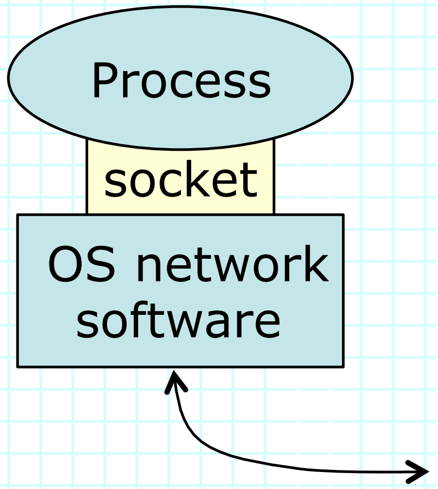
- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP (email)
  - DNS
- creating network applications
  - socket API

Application Architectures

- Applications implemented by software on end-systems
  - » no need to modify network components to enable new apps
- Client-server
  - » server expected to be “always-on”, has static IP address (usually)
  - » clients may come and go and change address frequently
  - » clients interact indirectly through servers (if at all)
- Peer-to-peer (P2P)
  - » all end systems play comparable roles
  - » may come and go at any time
- Hybrid of client-server and P2P
  - » client-server for setup/control, p2p for direct interaction
    - examples: Skype, instant messaging
  - » Commercial p2p for initial product distribution, fallback to server
    - example: Grooveshark

Communicating Processes

- A *process* is a program running on a host
  - » processes on same host communicate using OS-specific mechanisms
  - » processes on different hosts communicate by exchanging messages across a network
- Communicating pair of processes play distinct roles
  - » *server process* waits for communication requests
  - » *client process* initiates communication with waiting server
  - » a process may act both as a client and a server (for example, in p2p applications)
- Processes communicate thru *sockets*
  - » standard API to network software
  - » provides choice of transport service
  - » allows configuration of selected parameters



Addressing Processes

- An IP address identifies a *host*, but not a process
  - » since host may have many communicating processes running simultaneously, need some way to identify them
- Internet transport protocols use *port numbers* to designate specific programs within a host
  - » packets carry a *source port#* and a *destination port#*
  - » operating systems map port numbers to *sockets*
  - » so, packets sent through a socket are tagged with the source port# assigned to the socket
  - » packets received with a given destination port# are delivered to the socket to which that port# was assigned
- Some ports are reserved for specific applications
  - » for example: port 80 is used by web servers, port 25 for email
  - » allows remote client to easily connect to application

Review Questions

- What are port numbers used for?
  a) to identify specific interfaces of a router
  b) to identify a process running on a host
  c) to identify physical interfaces on a host
  d) to identify a socket belonging to an application program
- What popular application uses port number 80?
  a) email
  b) bit torrent
  c) web server
  d) Skype

Transport Services

- Key transport level services
  - » reliable data delivery
  - » guaranteed data rate
  - » bounded end-to-end delay
  - » secure communication
- Different applications have different needs
  - » highly variable delay may be disruptive in a phone conversation, but not be a problem for viewing web pages
  - » streaming HDTV requires a certain minimum data rate to ensure high quality video – and no use for much higher rates
  - » large file transfers should be “as fast as possible”, but can live with whatever is available – elastic

Internet Transport Services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

No widely deployed service offers:

  - » guaranteed data rate
  - » bounded end-to-end delay
  - » secure communication
- Secure communication is realized through application-level mechanisms

The Web and HTTP

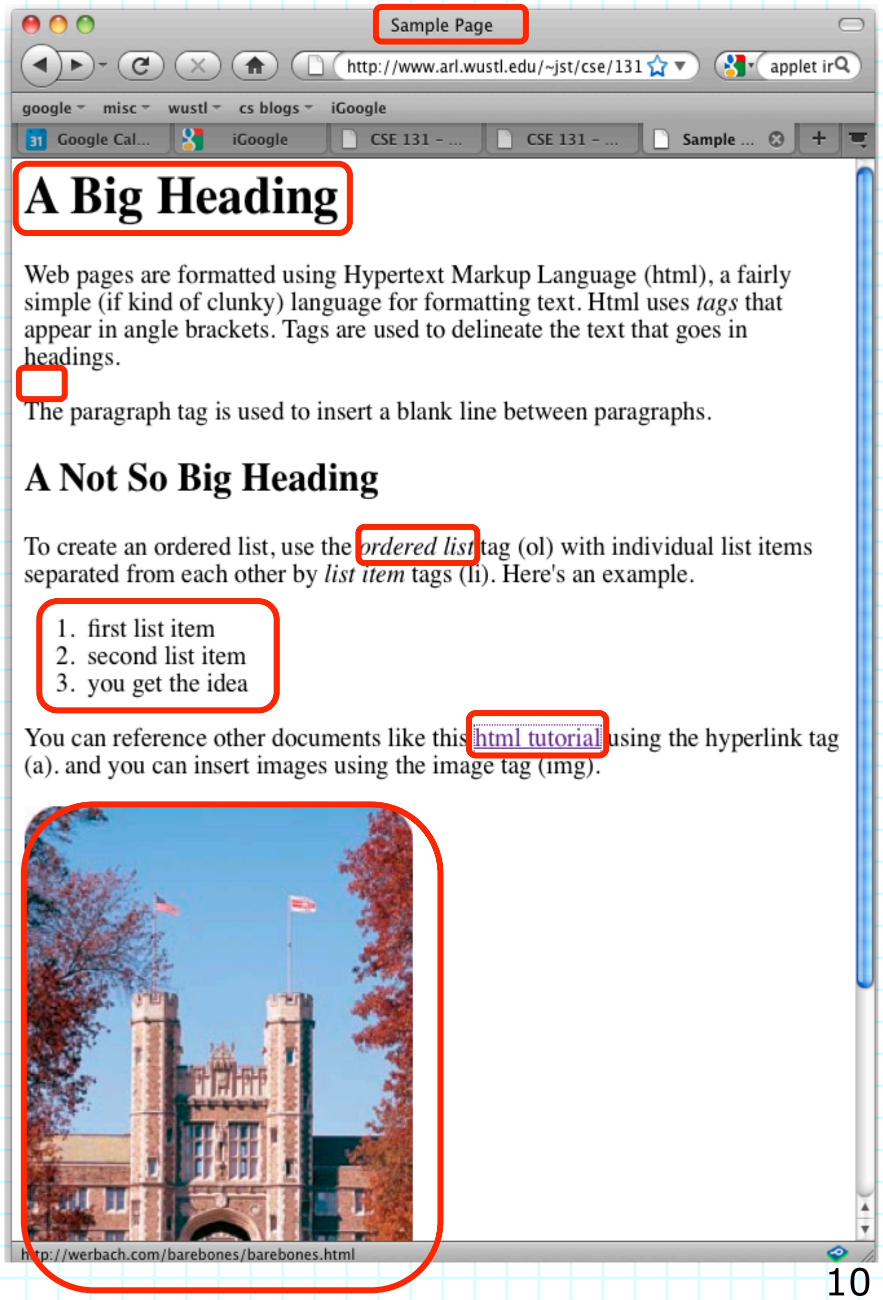
- HTTP is an *application layer* protocol used to transfer web pages between web servers and browsers
- Typical page includes some text along with references to other pages and *objects* (images, applets, audio files,...)
- Pages and objects are identified by URL (universal resource locator)
  - » $\underbrace{\text{www.wustl.edu/}}_{\text{identifies server (through its name)}}\underbrace{\text{\textasciitilde jst/cse/473/index.html}}_{\text{identifies file location}}$
- To display web page
  - » browser requests page from server
  - » parses page and finds URLs to referenced objects
  - » requests objects
  - » displays web page text and objects in browser window

Washington University in St.Louis

Engineering

Simple Web Page Example

```
<html>
<head> <title>Sample Page</title> </head>
<h1>A Big Heading</h1>
Web pages are formatted using Hypertext Markup
Language (html),... Html uses <i>tags</i> that
appear in angle brackets. ...
<p>
The paragraph tag is used to insert a blank
line between paragraphs.
<h2>A Not So Big Heading</h2>
To create an ordered list, use the <i>ordered
list</i> tag (ol) with individual list items
separated from each other by <i>list item</i>
tags (li). Here's an example.
<ol>
<li> first list item
<li> second list item ...
</ol> You can reference other documents like
this <a href="
http://werbach.com/barebones/
barebones.html">html tutorial</a> using the
hyperlink tag (a). and you can insert images
using the image tag (img).
<p> <img src="brookings.jpg”>
</body> </html>
```



A Big Heading

Web pages are formatted using Hypertext Markup Language (html), a fairly simple (if kind of clunky) language for formatting text. Html uses *tags* that appear in angle brackets. Tags are used to delineate the text that goes in headings.

The paragraph tag is used to insert a blank line between paragraphs.

A Not So Big Heading

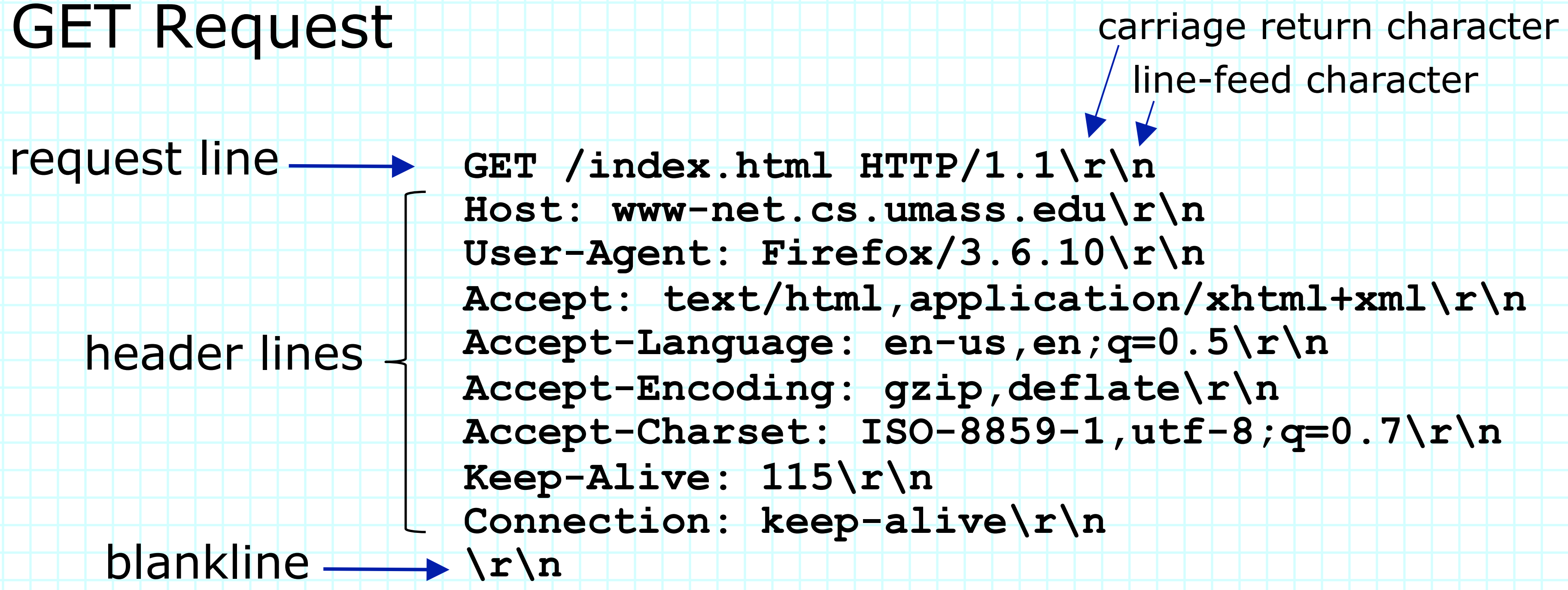
To create an ordered list, use the *ordered list* tag (ol) with individual list items separated from each other by *list item* tags (li). Here's an example.

1. first list item
2. second list item
3. you get the idea

You can reference other documents like this html tutorial using the hyperlink tag (a). and you can insert images using the image tag (img).

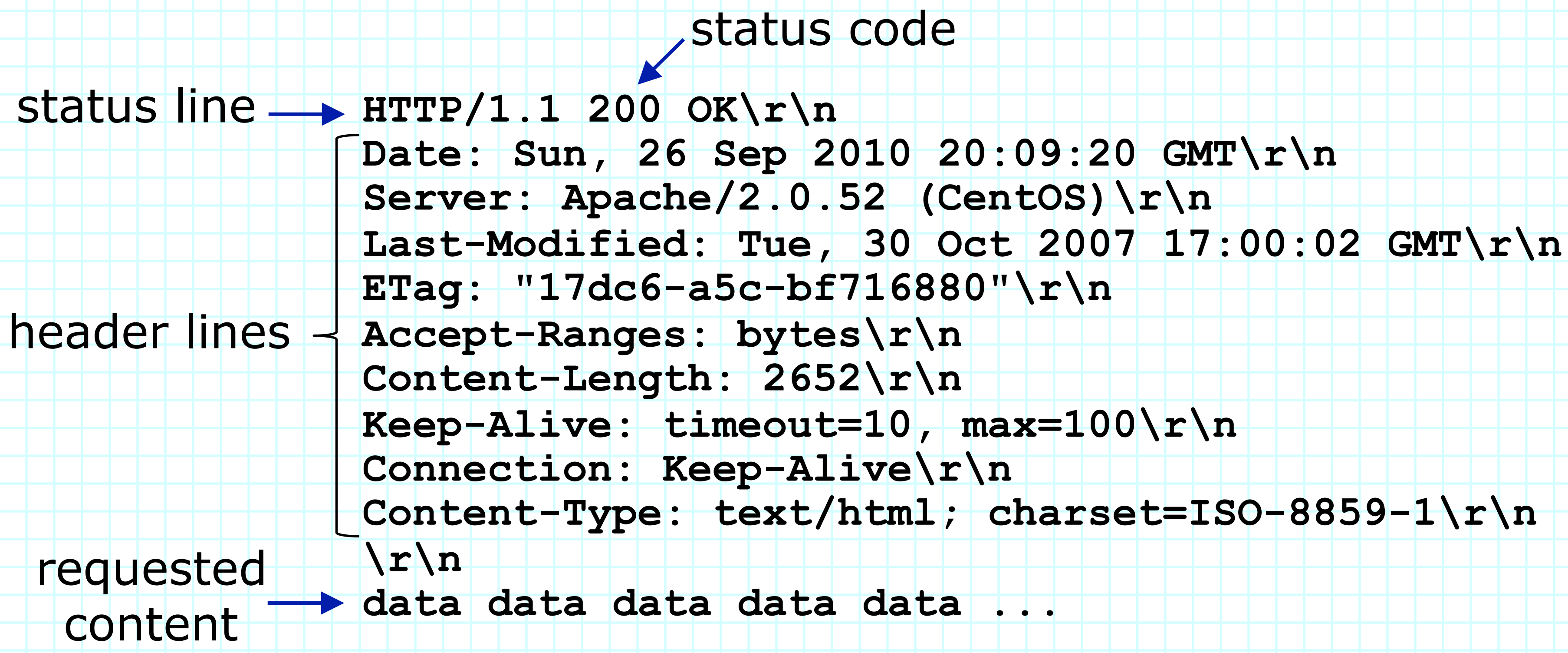
HTTP Details

- HTTP runs over TCP
  - » web browser initiates connection to server
  - » browser and server exchange formatted text messages
- Request/response protocol with *no server-side state*
  - » basic requests: GET, HEAD, POST, PUT, DELETE
- GET Request



```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP Response Message



```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

- Sample status codes
  - » 200 OK
  - » 301 Moved Permanently
  - » 400 Bad Request
  - » 404 Not Found
  - » 505 HTTP Version Not Supported

Exercises

You will need to refer to RFC 2616 to answer some of these questions

1. Explain the difference between HTML and HTTP
2. Why does HTTP use TCP rather than UDP?
3. What is the difference between a URI and a URL (and a URN)?
4. What is the maximum length of a URI in an HTTP message?
5. List five HTTP “methods”. What does the TRACE method do?
6. What does the date in an HTTP response signify?

Persistent vs. Non-Persistent HTTP

- Non-persistent (1.0)
  - » browser opens separate TCP connection for every GET request
  - » server closes connection after response is delivered
  - » simpler to implement but less-than-ideal performance
- Persistent (1.1)
  - » connection is kept open so long as requests continue
  - » browser may send GET requests for multiple objects, without waiting for responses to earlier requests
  - » allows all objects on a web page to be retrieved in one RTT
  - » signal by including "Connection: keep-alive" in GET header
- Performance comparison for page with *N* objects
  - » non-persistent: 2(*N*+1) RTTs plus transmission time
    - can improve using *N* parallel connections
  - » persistent: 3 RTTs plus transmission time

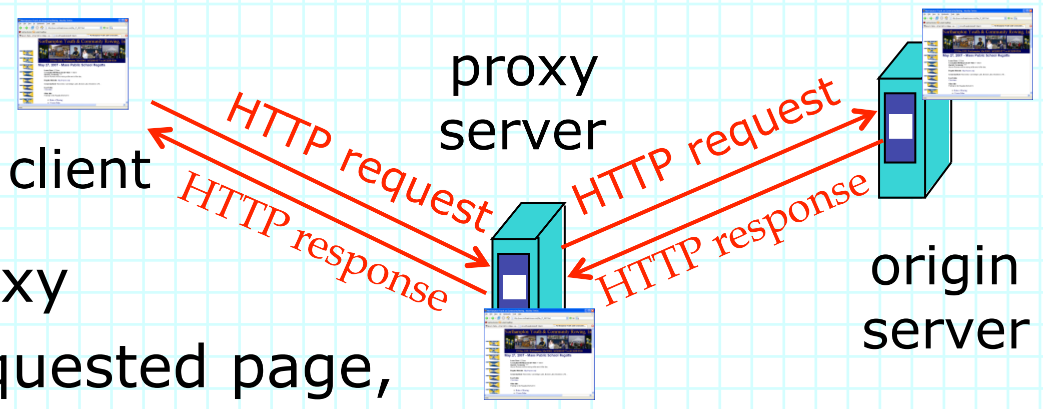
Web Applications and Cookies

- While HTTP remembers no per-client state, web apps often need to maintain state
  - » for example: to implement "shopping cart" in e-commerce apps
- A cookie is formatted text maintained by the browser at the request of a web site
  - » typically maintained in a file on client's computer
  - » may include information used by application to identify a user
    - such as key to a database entry with account data
- Cookie interaction
  - » server *sets* a cookie by including Set-Cookie header in response to a request
  - » when sending a request to a server for which it has a stored cookie, client includes Cookie: header and the associated text

Exercises

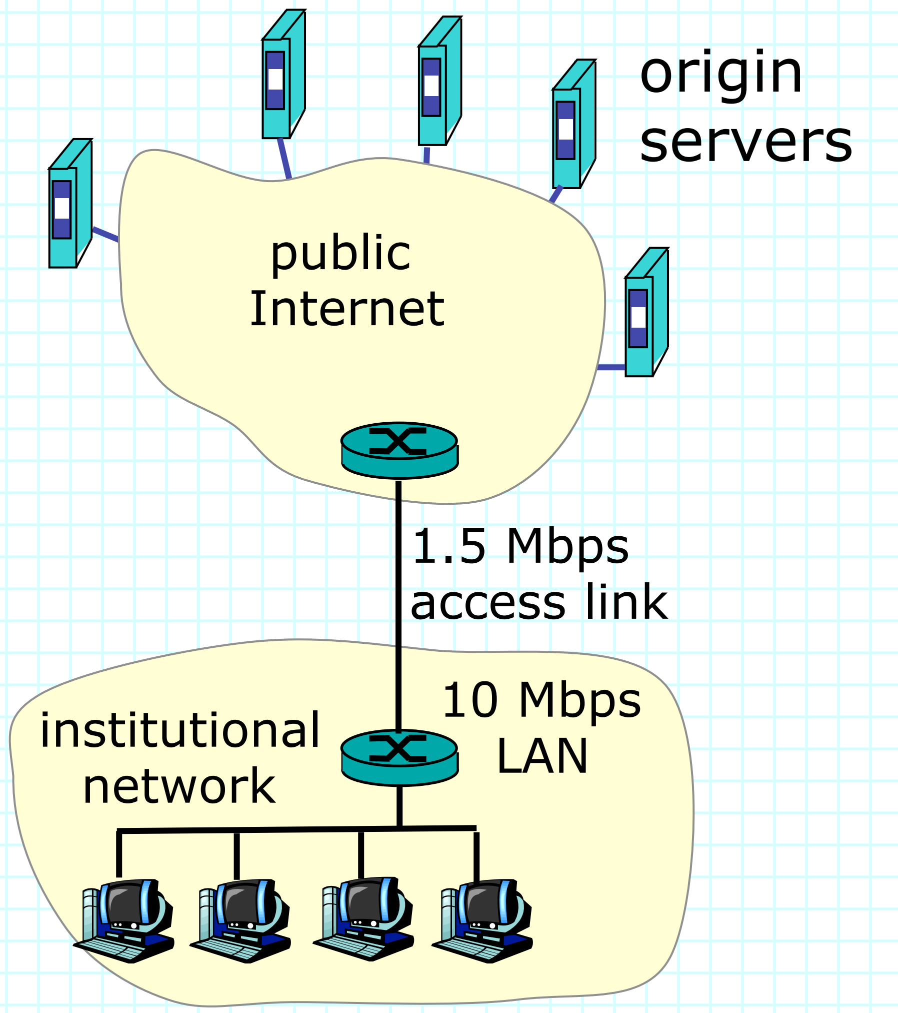
1. Consider a user with a 1 Mb/s DSL connection, requesting a web page with 10 KB of text and links to five images, each of which is 100 Kbytes long. Assume that the user is in LA and the web server is in NY, and the one way propagation delay is 25 ms.
   - » how long does it take to download the web page + images using non-persistent HTTP, with one connection open at a time?
   - » how long does it take if the browser opens multiple connections?
   - » how long does it take using persistent HTTP?
   - » how do these answers change if the images are 1 MB long?
2. We noted that web servers do not maintain state information about clients. Is this really true for servers that support persistent html? Since such servers maintain a connection with a client, they could reasonably maintain state for a client, while a connection remains active. Could you use this to get rid of cookies?

Web Caching with Proxy Servers



- Client may configure browser to use proxy
  - » all HTTP requests sent to proxy
  - » if proxy has *fresh* copy of requested page, it returns stored copy
  - » if not, it sends its own GET request to retrieve page from "origin server", sends page to client and stores a copy
- Two main benefits
  1. reduces web traffic over access link
  2. reduces response time and load on origin servers
- What about pages that may change?
  - » origin server can place limit on time object can be cached
  - » proxy can issue conditional GET – "If-modified-since:" header

Caching example



assumptions

- avg. http get request rate from users' browsers = 15/sec, average object size = 100,000 bits
  - » Aggregate rate ~1.5 Mbps
- Internet delay between ISP access router and any origin server ~ 200 msecs

consequences

- utilization on LAN ~ 15%
- utilization on access link ~ 100%
- total delay = Internet delay + access delay + LAN delay

  ~ 200 msecs + max access queuing delay + milliseconds

Caching example

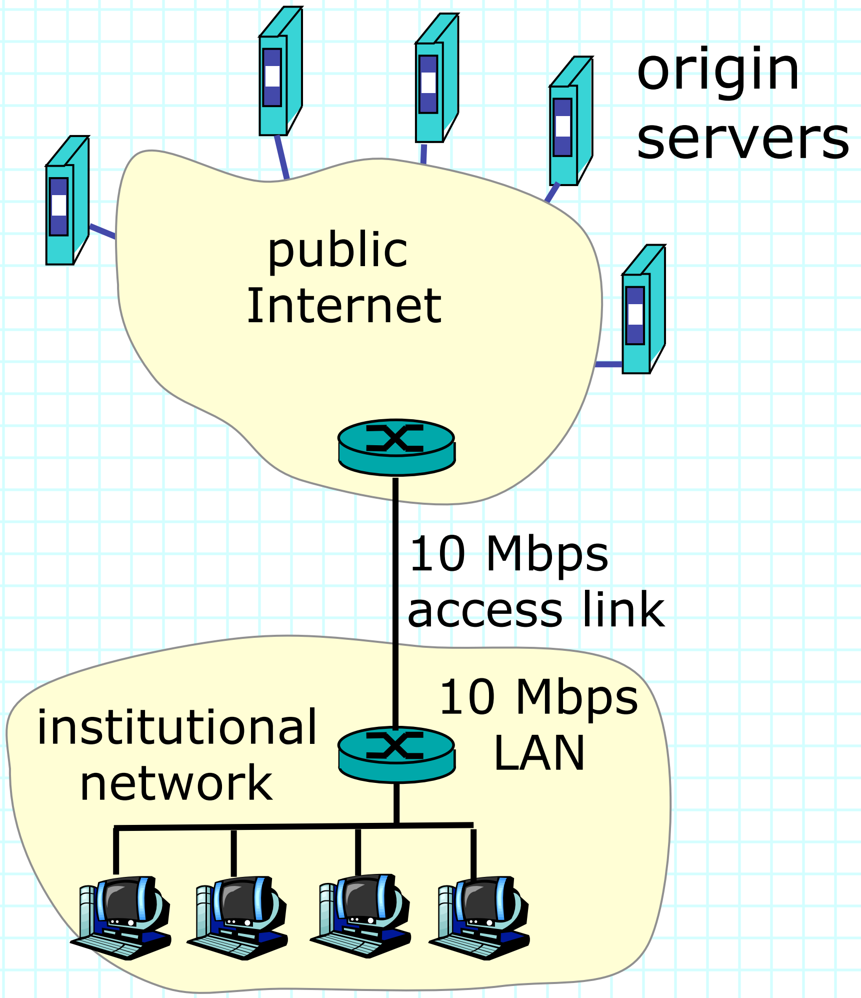
<u>possible solution</u>

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

<u>consequences</u>

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

  ~ 200 msecs + msecs + msecs
- often a costly upgrade



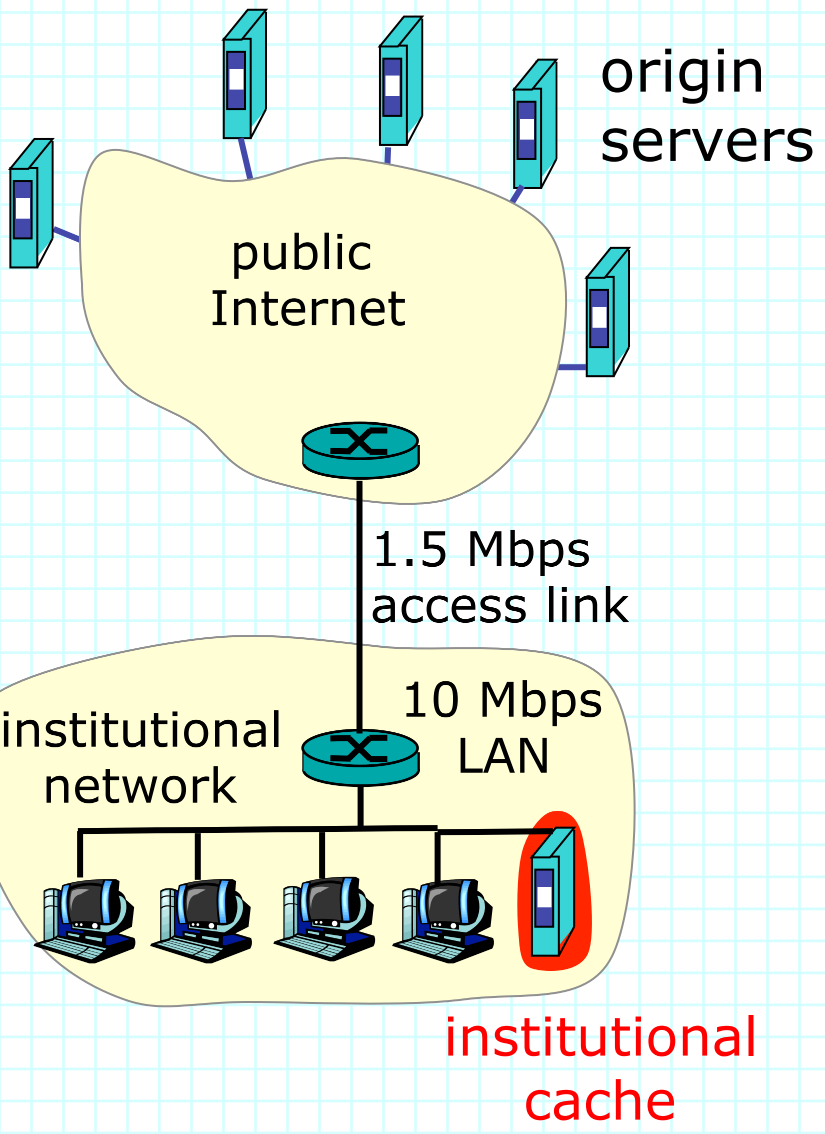
Caching example

<u>alternate solution</u>

- install proxy cache

<u>consequences</u>

- suppose hit rate is 0.4
  - » 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 = .6*(remote access delay)+.4*(cache delay) ~ .6*(210) msecs + msecs < 130 msecs



Exercises

1. Suppose Wash. U. has an institutional web cache used by all students and faculty and that during busy periods, the campus generates about 100 http get requests per second, with an average response size of 500 KB.
   - » What is the average resulting traffic on the university's internet connection if none of the requested web pages is in the cache?
   - » How does this change if the 95% of the requested pages can be retrieved from the cache?
2. Assume the world's web sites contain 100 TB of information, but that 5% of the stored web pages account for 90% of the requests.
   - » How much disk space does a web cache need to ensure that roughly no more than 10% of the requests it receives must be sent to the origin server?
   - » What fraction would approximately get sent to the origin server if the web cache could only store 500 GB? (Assume the top 5% pages are all equally popular)