Thought questions

1. What was the first web page you ever visited?

2. What was the most recent page you visited?

3. What page have you accessed the most often?

4. What page are you the most proud of visiting?
Announcements

• Project 1 is out, due on the 26th
• Textbook is in the bookstore

• Today’s plan:
  – HTTP
Layering

- Ethernet: (unreliable) packet delivery on a link
- IP: (unreliable) packet delivery across the Internet
- TCP: reliable delivery of a stream of bytes across the Internet
- HTTP: delivery of text, images, video, ...
  - Runs on top of TCP
Traditional Applications

• 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.
Traditional Applications

• 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.
  – 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>
    where <CRLF>stands for carriage-return-line-feed.
  – The first line (START LINE) indicates whether this is a request message or a response message.
HTTP 1.0 Example

Client

HTTP request

Server

Server generates response

HTTP response

Client

Server
HTTP requests

– Request Messages
  • The first line of an HTTP request message specifies three things: the operation (called *method*) to be performed, the Web page the operation should be performed on, and the version of HTTP being used.
  • We only focus on the ‘GET’ method in CSE 124

– Examples:
  • GET /index.html HTTP/1.0
  • GET /images/catimg23.jpg HTTP/1.1
  • GET /contracts/contract3.txt HTTP/1.1
Request headers

• After the start line are zero or more request headers:
  – Text-based, key and value separated by a colon

• Example 1:
  GET /index.html HTTP/1.0
  User-Agent: Firefox 23.3.1

• Example 2:
  GET /images/cat2.jpg HTTP/1.1
  Host: www.cs.ucsd.edu
  User-Agent: Chrome 12.1
HTTP Responses

- 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, and a text string giving the reason for the response.

- Example:
  HTTP/1.1 200 OK
  Content-Type: text/html
  Content-Length: 291
borabora:~ gmporter$ telnet oec-vmweb09.ucsd.edu 80
Trying 132.239.8.67...
Connected to oec-vmweb09.ucsd.edu.
Escape character is '^['].
GET /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 12 Jan 2015 19:36:37 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Thu, 28 Feb 2013 17:35:36 GMT
ETag: "fc7b21-a-4d6cc51858aec"
Accept-Ranges: bytes
Content-Length: 10
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

It works!

Connection closed by foreign host.
borabora:~ gmporter$
**HTTP Response Codes**

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

Five types of HTTP result codes

200: OK  
400: Client Error  
   403: Not allowed  
   404: Not found
Traditional Applications

• World Wide Web
  – 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 all the client wanted to do was 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 13 separate TCP connections being established and closed.
HTTP 1.1

– 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.
  - First, they obviously eliminate the connection setup overhead, thereby reducing the load on the server, the load on the network caused by the additional TCP packets, and the delay perceived by the user.
  - Second, because a client can send multiple request messages down a single TCP connection, TCP’s congestion window mechanism is able to operate more efficiently.
    » This is because it’s not necessary to go through the slow start phase for each page.
HTTP 1.1 “Persistent Connections”
Request headers

• Host:
  – Indicates the name of the server you are accessing
  – Used to implement virtual hosts

• User-Agent:
  – Identifies what software is issuing the request
  – E.g.:
    • Opera/9.25 (Windows NT 6.0; U; en)
    • Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en)
      AppleWebKit/125.2 (KHTML, like Gecko) Safari/125.8
Response headers

• Server:
  – Identifies the server
    • Server: Apache/2

• Content-Length:
  – How many octets in the response

• Content-Type:
  – text/html
  – image/jpg
  – image/png
Demos

• Follow along at
  – http://cseweb.ucsd.edu/~gmimporter/classes/wi16/cse124/projects/proj1/doc_root/

• Usage:  
curl –v http://...