We start our study of common distributed systems with the web and HTTP. Then we look at FTP and consider how HTTP differs from a file transfer protocol. Next we look at a push protocol — SMTP. Lastly, we look at the protocol and distributed system that underlies all we do on the Internet — the DNS.

The DNS is interesting from both the point of view of the design of a distributed system (scalability, transparency, etc.) as well as having a novel service architecture.
What is the difference between the web and HTTP?

The web has a client-server architecture.

- Clients are the browsers
- Servers are also publicly available (e.g., Apache)

Applications and Application-Layer Protocols

Overview

- Applications:
  - Communicating, distributed processes
  - Running in network hosts in "user space"
  - Exchange messages

- Application-layer protocols
  - One "piece" of an application
  - Defines messages exchanged and actions taken
  - Uses services provided by lower layer protocols

Application-Layer Protocols

Web Terminology

- User agent (client) for the Web is called a *browser*:
  - MS Internet Explorer
  - Google Chrome
  - Apple Safari

- Server for the Web is called a *Web server*:
  - Apache (public domain)
  - MS Internet Information Server (IIS)
Application-Layer Protocols

Web Terminology

- **Web page:**
  - Addressed by a URL
  - Consists of "objects"

- **Most Web pages consist of:**
  - Base HTML page
  - Embedded objects

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8">
    <title>CNN.com</title>
  </head>
  <body>
    
    </body>
</html>
```

Web Terminology

**URLs (Universal Resource Locators)**

- **URL components**
  - Server address
  - (Optional port number)
  - Path name

```text
www.someSchool.edu:8080/someDept/pic.gif
```

Optional server port (Default = port 80)
HTTP stands for Hypertext Transfer Protocol — used by both clients (to send request) and servers (to send response).

What's the difference between version 1.0 and 1.1?

HTTP 1.1 has two features to increase efficiency:
- **Persistence**: since a typical web page has several embedded objects (images)
- **Pipelining**: can the embedded images be obtained in a parallel manner?

If an HTTP server did maintain state about a client — it could, for example, remember which objects it has previously sent to a client and only send objects not previously sent.

HTTP is "stateless" — server maintains no information about past browser requests.

Protocols that maintain "state" are complex!:
- Past history (state) must be maintained
- If server or client crashes, their views of "state" may be inconsistent and must be reconciled.

**Reasons for port 80:**
- Why not port 123?

**Aside:**
- HTTP/1.0: RFC 1945
  - One request/response interaction per connection
- HTTP/1.1: RFC 2616
  - Persistent connections
  - Pipelined connections
The Hypertext Transfer Protocol

HTTP example

- User enters URL `www.someschool.edu/someDept/home.index`
  - Referenced object contains HTML text and references 10 JPEG images
- Browser sends an HTTP “GET” request to the server `www.someschool.edu`
- Server will retrieve and send the HTML file
- Browser will read the file and sequentially make 10 separate requests for the embedded JPEG images

HTTP 1.0 Example

URL `www.someschool.edu/someDept/home.index`

1) Browser initiates TCP connection to server at `www.someschool.edu`
   Port 80 is “well known” for server
0) Server process at host `www.someschool.edu` waiting for TCP connections on port 80

3) Client writes an HTTP GET request message (containing path) to TCP connection socket
2) Server “accepts” connection

4) Server reads request message, forms response message containing requested object, writes message to socket

5) Server closes TCP connection
Processing of the embedded objects is sequential — It’s possible that a browser may make parallel requests and we’ll consider that possibility in the next lecture.

By far the most common method is the GET method.

The Hypertext Transfer Protocol
HTTP message format

- Two types of HTTP message formats: request and response messages
  - ASCII (human-readable format)

- HTTP request message:
  - Request line
  - Optional header lines
  - Present only for some methods (e.g., POST)

HTTP 1.0 Example
URL www.someschool.edu/someDept/home.index

6) Browser reads response message containing the HTML file. Ten references to JPEG objects are found during the HTML parse

7) Browser initiates TCP connection to server at www.someschool.edu

8) Server “accepts” connection

TCP 3-way handshake

The above steps are repeated for each of the 10 JPEG objects
The structure of these request messages can be described by a BNF just as was done for the SMTP/FTP messages.

—

Note the extra CRLF at the end of the headers to terminate the request.

The Accept headers are called content negotiation headers.

—

The UTF-8 encoding allows Unicode to be used in a convenient and backwards compatible way in environments that, like Unix, were designed entirely around ASCII. UTF-8 is the way in which Unicode is used under Unix, Linux, and similar systems.

The Connection header deals with persistent connections.

—

More in the next lecture.

Note that strictly speaking, HTTP does not specify the interpretation of the headers, only the format structure of the header block.

—

You can put in a header field of Dog if you like.

HW: Write a Java program to capture HTTP requests from a browser.

♦♦

How does Mozilla process: http://dove.cs.unc.edu:80/~jasleen ??
The response status is a common messaging paradigm in client server systems.

We'll see similar chatty server response messages in SMTP/FTP.

These messages are in the first server-to-client response message.

**HTTP Message Format**

**General response message format**

- Response messages
  - ASCII (human-readable format)

- Message structure:
  - Response line
    - version <SP> code <SP> phrase <CR><LF>
  - Optional header lines
  - Requested object, error message, etc.

- header field name “:” value <CR><LF>
- entity body

**HTTP Message Format**

**HTTP response status codes**

- Sample response codes:
  - 200 OK
    - Request succeeded, requested object later in this message
  - 301 Moved Permanently
    - Requested object moved, new location specified later in this message (Location:)
  - 400 Bad Request
    - Request message not understood by server
  - 404 Not Found
    - Requested document not found on this server
  - 505 HTTP Version Not Supported
HTTP Message Format

Typical Request and Response Headers

Request headers
- Connection: Keep-Alive
- User-Agent: Mozilla/4.74 [en] (WinNT; U)
- Host: dove.cs.unc.edu:80
- Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
- Accept-Encoding: gzip
- Accept-Language: en
- Accept-Charset: iso-8859-1,*,utf-8
- Cookie: SITE_SERVER_ID=8a064b785a043146e4599174a3d970

Response headers
- Date: Fri, 02 Feb 2001 19:10:11 GMT
- Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
- ETag: "1807135e-67-3a77369e"
- Accept-Ranges: bytes
- Content-Length: 103
- Connection: close
- Content-Type: text/plain

HTTP Message Format

Telnet example

Connect to HTTP server port
- > telnet www.cs.unc.edu 80
- Trying 152.2.131.240...

Telnet output
- Connected to rock.cs.unc.edu.
- Escape character is ^]'.

Type GET command
- GET /jasleen/foo.txt HTTP/1.0
- HTTP response status line
- HTTP/1.1 200 OK
- Date: Tue, 12 Feb 2013 01:43:25 GMT
- Server: Apache/2.2.15 (Red Hat)
- Last-Modified: Tue, 12 Feb 2013 01:43:02 GMT
- ETag: "4d2e7a0-6e-4d57d25675580"

HTTP response headers plus blank line
- Accept-Ranges: bytes
- Content-Length: 110
- Connection: close
- Content-Type: text/plain; charset=UTF-8
- Object content
- ** This test file is stored in the UNIX
- ** file system at
- ** /afs/cs.unc.edu/home/jasleen/public_html/foo.txt
- Telnet output
- Connection closed by foreign host.
In this example, we request an object that doesn't exist on the server. The object returned is an HTML page that describes the error that occurred.

Today, when you do this, you get an Object Moved Permanently message (moved to www.bing.com).

### HTTP Message Format

<table>
<thead>
<tr>
<th>Telnet example (2)</th>
<th>Telnet output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to HTTP server port</td>
<td>&gt; telnet <a href="http://www.msn.com">www.msn.com</a> 80</td>
</tr>
<tr>
<td>Connected to <a href="http://www.msn.com">www.msn.com</a>.</td>
<td>Trying 207.46.179.134...</td>
</tr>
<tr>
<td>Escape character is '{'.</td>
<td>GET /~index.html HTTP/1.0</td>
</tr>
<tr>
<td>HTTP response status line</td>
<td>HTTP/1.1 404 Object Not Found</td>
</tr>
<tr>
<td>Server: Microsoft-IIS/5.0</td>
<td>Date: Mon, 11 Feb 2002 18:33:15 GMT</td>
</tr>
<tr>
<td>Content-Length: 1638</td>
<td>Content-Type: text/html</td>
</tr>
<tr>
<td>&lt;HTML&gt; &lt;HEAD&gt; . . .</td>
<td></td>
</tr>
<tr>
<td>. . . .</td>
<td>Error type 404 - Object Not Found</td>
</tr>
<tr>
<td>. . .</td>
<td>&lt;/body&gt; &lt;/html&gt;</td>
</tr>
<tr>
<td>Object content</td>
<td>Connection closed by foreign host.</td>
</tr>
<tr>
<td>Telnet output</td>
<td></td>
</tr>
</tbody>
</table>

COMP 431 Lecture 2b, The Web & HTTP Page 19