

15-213

*“The course that gives CMU its
Zip!”*

Web Services Nov 26, 2002

Topics

- HTTP

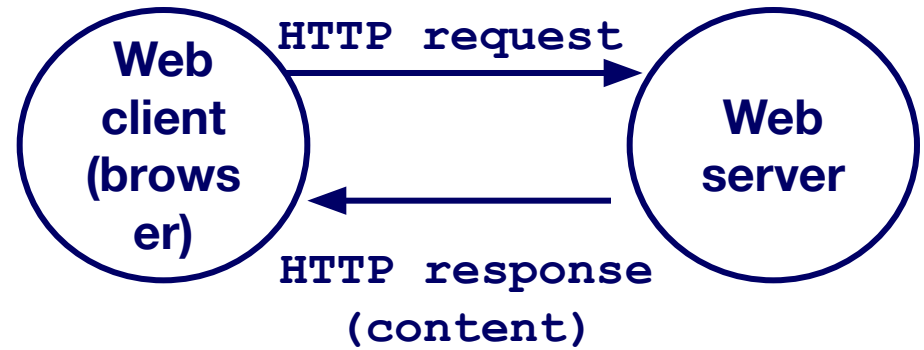
Web Servers

**Clients and servers
communicate using the
HyperText Transfer
Protocol (HTTP)**

- Client and server establish TCP connection
- Client requests content
- Server responds with requested content
- Client and server close connection (usually)

Current version is HTTP/1.1

- RFC 2616, June, 1999.



Web Content

Web servers return **content** to clients

- **content**: a sequence of bytes with an associated MIME (Multipurpose Internet Mail Extensions) type

Example MIME types

- `text/html` HTML document
- `text/plain` Unformatted text
- `application/postscript` Postscript document
- `image/gif` Binary image encoded in GIF format
- `image/jpeg` Binary image encoded in JPEG format

Static and Dynamic Content

The content returned in HTTP responses can be either *static* or *dynamic*.

- Static content: content stored in files and retrieved in response to an HTTP request
 - Examples: HTML files, images, audio clips.
- Dynamic content: content produced on-the-fly in response to an HTTP request
 - Example: content produced by a program executed by the server on behalf of the client.

Bottom line: *All Web content is associated with a file that is managed by the server.*

URLs

Each file managed by a server has a unique name called a **URL (Universal Resource Locator)**

URLs for static content:

- `http://www.cs.cmu.edu:80/index.html`
- `http://www.cs.cmu.edu/index.html`
- `http://www.cs.cmu.edu`
 - Identifies a file called `index.html`, managed by a Web server at `www.cs.cmu.edu` that is listening on port 80.

URLs for dynamic content:

- `http://www.cs.cmu.edu:8000/cgi-bin/adder?15000&213`
 - Identifies an executable file called `adder`, managed by a Web server at `www.cs.cmu.edu` that is listening on port 8000, that should be called with two argument strings: 15000 and 213.

How Clients and Servers Use URLs

Example URL: `http://www.aol.com:80/index.html`

Clients use *prefix* (`http://www.aol.com:80`) to infer:

- What kind of server to contact (Web server)
- Where the server is (`www.aol.com`)
- What port it is listening on (80)

Servers use *suffix* (`/index.html`) to:

- Determine if request is for static or dynamic content.
 - No hard and fast rules for this.
 - Convention: executables reside in `cgi-bin` directory
- Find file on file system.
 - Initial “/” in suffix denotes home directory for requested content.
 - Minimal suffix is “/”, which all servers expand to some default home page (e.g., `index.html`).

Anatomy of an HTTP Transaction

```
unix> telnet www.aol.com 80
Trying 205.188.146.23...
Connected to aol.com.
Escape character is '^]'.
GET / HTTP/1.1
host: www.aol.com
```

```
HTTP/1.0 200 OK
MIME-Version: 1.0
Date: Mon, 08 Jan 2001 04:59:42 GMT
Server: NaviServer/2.0 AOLserver/2.3.3
Content-Type: text/html
Content-Length: 42092
```

```
<html>
...
</html>
```

```
Connection closed by foreign host.
unix>
```

Client: open connection to server
Telnet prints 3 lines to the terminal

Client: request line

Client: required HTTP/1.1 HOST header

Client: empty line terminates headers .

Server: response line

Server: followed by five response headers

Server: expect HTML in the response body

Server: expect 42,092 bytes in the resp body

Server: empty line (“\r\n”) terminates hdrs

Server: first HTML line in response body

Server: 766 lines of HTML not shown.

Server: last HTML line in response body

Server: closes connection

Client: closes connection and terminates

HTTP Requests

HTTP request is a *request line*, followed by zero or more *request headers*

Request line: <method> <uri> <version>

- <version> is HTTP version of request (HTTP/1.0 or HTTP/1.1)
- <uri> is typically URL for proxies, URL suffix for servers.
- <method> is either GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE.

HTTP Requests (cont)

HTTP methods:

- GET: Retrieve static or dynamic content
 - Arguments for dynamic content are in URI
 - Workhorse method (99% of requests)
- POST: Retrieve dynamic content
 - Arguments for dynamic content are in the request body
- OPTIONS: Get server or file attributes
- HEAD: Like GET but no data in response body
- PUT: Write a file to the server!
- DELETE: Delete a file on the server!
- TRACE: Echo request in response body
 - Useful for debugging.

HTTP Requests (cont)

Request headers: <header name>: <header data>

- Provide additional information to the server.

Major differences between HTTP/1.1 and HTTP/1.0

- HTTP/1.0 uses a new connection for each transaction.
- HTTP/1.1 also supports *persistent connections*
 - multiple transactions over the same connection
 - Connection: Keep-Alive
- HTTP/1.1 requires HOST header
 - Host: kittyhawk.cmcl.cs.cmu.edu
- HTTP/1.1 adds additional support for caching

HTTP Responses

HTTP response is a **response line** followed by zero or more **response headers**.

Response line:

<version> <status code> <status msg>

- **<version>** is HTTP version of the response.
- **<status code>** is numeric status.
- **<status msg>** is corresponding English text.
 - **200 OK** Request was handled without error
 - **403 Forbidden** Server lacks permission to access file
 - **404 Not found** Server couldn't find the file.

Response headers: <header name>: <header data>

- **Provide additional information about response**
- **Content-Type:** **MIME** type of content in response body.
- **Content-Length:** Length of content in response body.

GET Request to Apache Server From IE Browser

```
GET /test.html HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
Host: euro.ecom.cmu.edu
Connection: Keep-Alive
CRLF (\r\n)
```

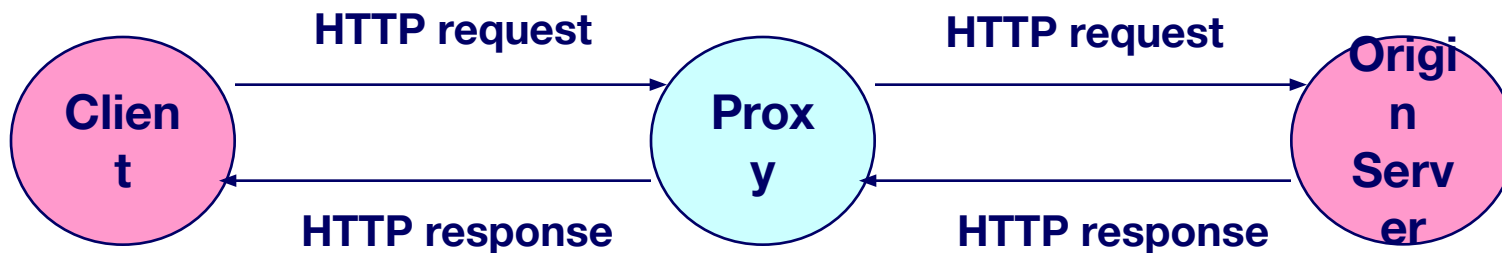
GET Response From Apache Server

```
HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 04:02:15 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body>
<h1>Test page</h1>
</html>
```

Proxies

A **proxy** is an intermediary between a client and an **origin server**.

- To the client, the proxy acts like a server.
- To the server, the proxy acts like a client.



Why Proxies?

Can perform useful functions as requests and responses pass by

- Examples: Caching, logging, anonymization

