Lecture 4
Web browsers, servers and HTTP

Boriana Koleva
Room: C54
Email: bnk@cs.nott.ac.uk

Overview

- Client-server paradigm
- Web browsers
- Web servers
- URLs
- MIME
- HTTP
- 'Warriors of the net' video

The client server paradigm

- A widely used form of communication
- Server application waits passively for contact from clients
- A server provides a specific service
- Client application actively initiates contact with the server
- Information can flow in both directions
- Typical situation is many clients interacting with each server

Web Browsers

- Browsers are clients
  - always initiate, servers react
- Allow user to browse resources available on server
  - either existing or dynamically built documents
- Mosaic - NCSA (Univ. of Illinois), in early 1993
  - First to use a GUI, led to explosion of Web use
  - Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
- Current common browsers
  - Firefox, Internet Explorer, Google Chrome, Safari

Web Servers

- Provide responses to browser requests
- All communications between browsers and servers use Hypertext Transfer Protocol (HTTP)
- Web servers run as background processes in the operating system
  - Monitor a communications port on the host, accepting HTTP messages when they appear
- Common servers
  - Apache, Internet Information Server (IIS), Google Web Server

Uniform Resource Locators (URLs)

- Standard way of specifying entities on networks
- First part - protocol
  - terminated by colon (:)
  - common protocols are http, ftp, mailto, telnet,
  - i.e.: http: ftp: mailto: telnet:
- Second part - varies according to protocol
  - mailto - e-mail address e.g.:
    - mailto: David.Brailsford@nottingham.ac.uk
  - resource-oriented protocols (http, ftp etc)
    - Host name + domain names (preceded by //)
      - may optionally include username, password and port
    - Pathname (usually related to the path of a file on the server)
    - i.e. //fully-qualified-domain-name/path-to-document
- Optional third parts
  - Query string (preceded by ?)
  - Fragment identifier (preceded by #)

Example URLs

- mailto:steve.benford@nottingham.ac.uk
- http://www.crg.cs.nott.ac.uk/~bnk/index.html
- http://www.nottingham.ac.uk:80/
- http://acomputer.cs.nott.ac.uk:8799/
- http://uname:pword@acomputer.cs.nott.ac.uk/private/secret.html
- http://acomputer.cs.nott.ac.uk/dbase?stuff
- http://acomputer.cs.nott.ac.uk/myfile.html#third
- ftp://uname:pword@acomputer.cs.nott.ac.uk/
- ftp://acomputer.cs.nott.ac.uk/

General Server Characteristics

- Web servers have two main directories:
  - 1. Server root (server system software)
  - 2. Document root (servable documents)
    - This will map to the URL of the full domain name, e.g.: http://www.cs.nott.ac.uk/
  - User document root directory
    - Directories of a standard name in the users home directory
    - Usually this is called public_html
    - The URL is then mapped as ~*username* e.g.: http://www.cs.nott.ac.uk/~bnk/

General Server Characteristics

- Document root is accessed indirectly by clients
  - Its actual location is set by the server configuration file
  - Requests are mapped to the actual location
    - E.g. doc root is `topdocs` and stored in `/admin/web`
    - Site is `http://www.flowers.com`
    - When there is a request for `http://www.flowers.com/bulbs/tulips.html`
    - Server searches for file with address `/admin/web/topdocs/bulbs/tulips.html`

Additional Server Features

- Virtual document trees
  - Part of servable document collection stored outside the document root
- Virtual hosting
  - Support for more than one site on a computer
- Proxy servers
  - Serve documents that are in the document root of other machines

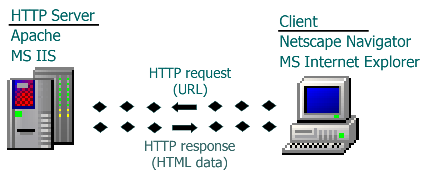
Multipurpose Internet Mail Extensions (MIME)

- Originally developed for email
- Used to specify document types transmitted over the Web
  - MIME type attached by the server to the beginning of the document
- Type specifications
  - Form: type/subtype
  - Examples: text/plain, text/html, image/gif, image/jpeg

MIME

- Server gets type from the requested file name's suffix (.html implies text/html)
- Browser gets the type explicitly from the server
- Experimental types
  - Subtype begins with x-
  - e.g. video/x-msvideo
  - Experimental types require the server to send a helper application or plug-in so the browser can deal with the file

World Wide Web Overview



Design Paradigm of the WWW

- WWW is a global hypertext system
- The page is the basic unit of the WWW
- Each page has a unique identifier – the URL
- Pages may contain links to data of any type
- Some data (e.g. images) may be interpreted by the browser and displayed "inline"
- Pages may contain links to other URLs

The HTTP Protocol

- Invented by Tim Berners-Lee in 1990
- RFC 1945 (1996) - HTTP/1.0
- RFC 2068 (1997) - HTTP/1.1
- RFC 2616 (1999) - HTTP/1.1
  - (update to 2068)

Features of HTTP

- Application level, client-server protocol
  - Primarily for distributed hypermedia systems
  - Flexible - thus has many other uses - e.g.:
    - Nameservers
    - Distributed & collaborative document management systems
- HTTP is small and fast
  - Minimal performance overhead
  - Easy to implement
- HTTP is a stateless protocol
  - Each request is an independent transaction - unrelated to any previous requests (unlike session-based protocols, e.g. FTP)
  - Advantage
    - Simplifies server design - information about previous transactions does not need to be stored
  - Disadvantage
    - More information must be included in each request

HTTP Operation

- On the Internet HTTP usually uses TCP/IP connections
- TCP Port 80 is the default (though others can be specified)
- HTTP uses a Request/Response paradigm
  - Client establishes a connection to the server, and sends it a request
  - Server responds to the request by generating a response (which may or may not contain content)

HTTP Request

- Delivered from a client to a server containing instructions for the server
- Contains
  - the method to be applied to the data resource
  - the identifier of the resource
  - the protocol version in use
- Most commonly used methods:
  - GET - Fetch a document
  - HEAD - Fetch just the header of the document
  - POST - Execute the document, using the data in body
  - PUT - Store a new document on the server
  - DELETE - Remove a document from the server

Request message

General request message structure

```
METHOD  /path-to-resource  HTTP/version-number
Header-Name-1: value
Header-Name-2: value

[optional request body]
```

Example

```
GET /index.html  HTTP/1.1
Host: www.cs.nott.ac.uk
Accept: text/*
User-Agent: Mozilla/2.02Gold (WinNT; I)
```

telnet HTTP request

- A browsers is not necessary to communicate with a web server

```
> telnet blanca.uccs.edu http

GET /respond.html HTTP/1.1
Host: blanca.uccs.edu
```

HTTP Response

- Message generated by a server after receiving and interpreting a request
- Responses contain:
  - Status line with the protocol version, a status code, and a "reason phrase"
  - Response-Header (containing information about the server)
  - Entity Header (meta-information)
  - Entity Body (data)

Response message

General response message structure

```
HTTP/version-number  status-code  message
Response-Header-Name-1: value
Response-Header-Name-2: value
Entity-Header-Name-1: value
Entity-Header-Name-2: value

[optional entity body]
```

Example

```
HTTP/1.1  200  OK
Server: Apache (Red-Hat/Linux)
Content-Type: text/html
Content-Length: 9934

<HTML>
<HEAD>
<TITLE>School of Computer Science</TITLE>
…
```

Some HTTP Status Codes

- 200 : OK
- 201 : Created
- 202 : Accepted
- 204 : No Content
- 301 : Moved Permanently
- 302 : Moved Temporarily
- 400 : Bad Request
- 401 : Unauthorized
- 403 : Forbidden
- 404 : Not Found
- 500 : Internal Server Error
- 503 : Service Unavailable

Summary

- Client-server paradigm
- Web browsers
- Web servers
- URLs
- MIME types
- HTTP protocol
  - Requests and responses