World Wide Web – Part I

Lecture 11: World wide web – Part I

On completion, the student will be able to:

- Explain the functions of the web clients (browsers) and the web servers.
- Explain the commands and responses of the hypertext transfer protocol (HTTP).
- State the mechanism to locate Internet resources using the uniform resource locator (URL).
- Demonstrate the way web servers can be accessed from a web client.

World Wide Web (WWW)

- Latest revolution in the internet scenario.
- Allows multimedia documents to be shared between machines.
  - Containing text, image, audio, video, animation.
- Basically a huge collection of inter-linked documents.
  - Billions of documents.
  - Inter-linked in any possible way.
  - Resembles a cob-web.

WWW (contd.)

- Where do the documents reside?
  - On web servers.
  - Also called Hyper Text Transfer Protocol (HTTP) servers.
- They are typically written in
  - Hyper Text Markup Language (HTML).
- Documents get formatted/displayed using
  - Web browsers
    - Internet Explorer
    - Netscape
    - Mosaic
    - Konquerer

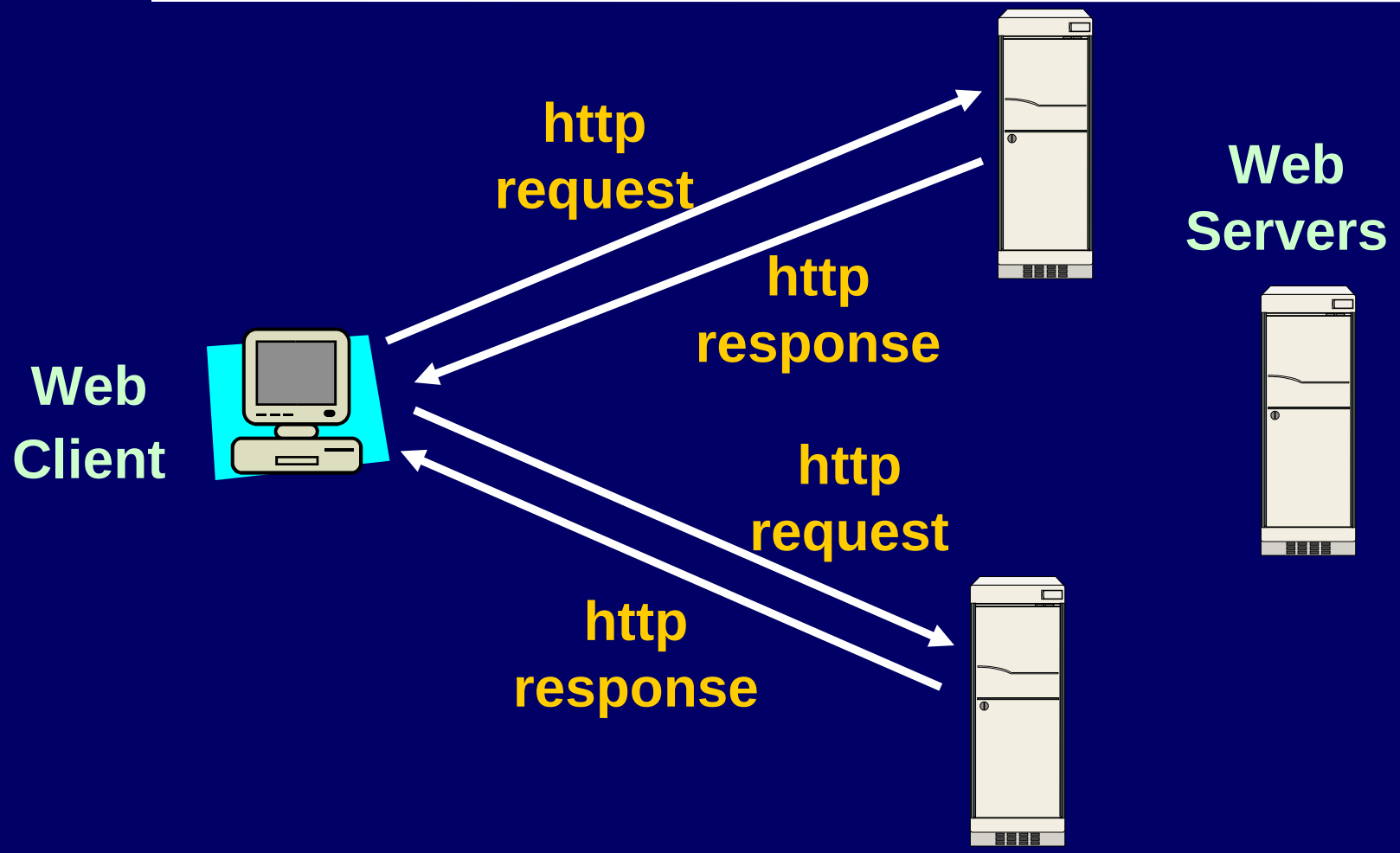
What is HTTP?

- Hyper Text Transfer Protocol
  - A protocol using which web clients (browsers) interact with web servers.
- It is a stateless protocol.
  - Fresh connection for every item to be downloaded.
- Transfers hypertext across the Internet.
  - A text with links to other text documents.
  - Resembles a cob-web, and hence the name World Wide Web (WWW).

HTTP Protocol

- Web clients (browsers) and web servers communicate via HTTP protocol.
- Basic steps:
  - Client opens socket connection to the HTTP server.
    - Typically over port 80.
  - Client sends HTTP requests to server.
  - Server sends back response.
  - Server closes connection.
    - HTTP is a stateless protocol.

Illustration



HTTP Request Format

- A client request to a server consists of:
  - Request method
  - Path portion of the HTTP URL
  - Version number of the HTTP protocol
  - Optional request header information
  - Blank line
  - POST or PUT data if present.

HTTP Request Methods

- GET
  - Most common HTTP method.
  - Returns the contents of the specified document.
  - Places any parameters in request header.
  - Can also be used to submit forms:
    - The form data is URL-encoded and appended to the GET command URL.

```
GET  /cgi-bin/myscript.cgi?Roll=1234&Sex=M  HTTP/1.0
```

Illustration of GET

- A very simple HTTP connection to a server.

  telnet www.facweb.iitkgp.ac.in http

- Client sends request for a file:

  GET /test.html HTTP/1.0

- The server sends back the response:

  HTTP/1.1 200 OK
  Date: Sun, 22 May 2005 09:51:42 GMT
  Server: Apache/1.3.33 (Win32)
  Last-Modified: Sun, 22 May 2005 09:51:10 GMT
  Accept-Ranges: bytes
  Content-Length: 119
  Connection: close

Illustration of GET (contd.)

Content-Type: text/html

<html> <head> <title> A test page </title> </head>
<body>
This is the body of the test page.
</body>
</html>

HTTP Request Methods (contd.)

- **HEAD**
  - Returns only the header information of the specified document.
  - Used by clients to determine the file size, modification date, server version, etc.

Illustration of HEAD

- Client sends

  HEAD /index.html HTTP/1.0

- Server responds back with:

  HTTP/1.1 200 OK

  Date: Sun, 22 May 2005 10:08:37 GMT

  Server: Apache/1.3.33 (Win32)

  Last-Modified: Thu, 03 May 2001 11:30:38 GMT

  Accept-Ranges: bytes

  Content-Length: 1494

  Connection: close

  Content-Type: text/html

HTTP Request Methods (contd.)

- **POST**
  - Used to send data to the server to be processed in some way, as in a CGI script.
  - Basic difference from GET:
    - A block of data is sent along with the request. Extra headers like `Content-Type` and `Content-Length` are used for this purpose.

- The requested object is not a resource to retrieve. Rather, it is a script that can handle the data being sent.
- The server response is not a static file; but is generated dynamically as the program output.

Illustration of POST

- A typical form submission, using POST is illustrated below:

```
POST  /cgi-bin/myscript.cgi  HTTP/1.0
From:  isg@hotmail.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

Roll=1234&Sex=M&Age=20
```

HTTP Request Methods (contd.)

- PUT
  - Replaces the contents of the specified document with data supplied along with the command.
  - Not used widely.
- DELETE:
  - Deletes the specified document from the server.
  - Not used widely.

HTTP Request Headers

- After a HTTP request line, a client can send any number of header fields.
  - Usually optional – used to convey some information.
  - Some commonly used fields:
    - Accept: MIME types client accepts, in order of preference.
    - Connection: connection options, close or Keep-Alive.

- **Content-Length: number of bytes of data to follow.**
- **Content-Type: MIME type and subtype of the data that follows.**
- **Pragma: "no-cache" option directs the server/proxy to return a fresh document even though a cached copy may exist.**

HTTP Request Data

- To be given if the request type is either PUT or POST.
  - Send the data immediately after the HTTP request header, and a blank line.

HTTP Response

- An initial response line.
  - Also called the status line.
  - Consists of three parts separated by spaces
    - The HTTP version
    - A 3-digit response status code
    - An English phrase describing the status code.

```
HTTP/1.0  200  OK
```

```
HTTP/1.0  404  Not Found
```

HTTP Response (contd.)

- Header information, followed by a blank line, and then the data.

```
HTTP/1.1 200 OK
Date: Sun, 22 May 2005 09:51:42 GMT
Server: Apache/1.3.33 (Win32)
Last-Modified: Sun, 22 May 2005 09:51:10 GMT
Content-Length: 119
Connection: close
Content-Type: text/html

<html>  <head> <title> A test page  </title> </head>
<body>
   This is the body of the test page.
</body> </html>
```

3-digit Status Code

- 1xx
  - Indicates informational messages only.
- 2xx
  - Indicates successful transaction.
- 3xx
  - Redirects the client to another URL.
- 4xx
  - Indicates client error, such as unauthorized request.
- 5xx
  - Indicates internal server error.

Common Status Codes

- 200 OK
- 301 Moved Permanently
- 302 Moved Temporarily
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

HTTP Response Headers

- Common response headers include:
  - Content-Length
    - Size of the data in bytes.
  - Content-Type
    - MIME type and subtype of data being sent.
  - Date
    - Current date.
  - Expires
    - Date at which document expires.
  - Last-Modified
  - Set-Cookie
    - Name/value pair to be stored as cookie.

HTTP Response Data

- A blank line follows the response header, and the data follows next.
  - No upper limit on data size.
- HTTP/1.0
  - Server typically closes connection after completing a transaction.
- HTTP/1.1
  - Server keeps the connection open by default, across transactions.

HTTP version 1.1

- Current standard and widely used.
  - Became IETF draft standard in 2001.
- Improvements over HTTP 1.0:
  - Requires host identification.

```
GET  /index.html  HTTP/1.1
Host: www.facweb.iitkgp.ac.in
<blank line>
```

  - Allows multi-homed servers.
  - More than one domain living on same server.

HTTP version 1.1 (contd.)

- Default support for persistent connections.
  - Multiple transactions over a single connection.
- Support for content negotiation.
  - Decides on the best among the available representations.
  - Server-driven or browser-driven.
- Browsers can request part of document.
  - Specify the bytes using **Range** header.
  - Browser can ask for more than one range.
  - Continue interrupted downloads.

```
Range: bytes=1200-3500
```

HTTP version 1.1 (contd.)

- Efficient caching support
  - A document caching model that allows both the server and the client to control the level of cachability and update conditions and requirements.
- HTTP 1.1 requires several extra things from both clients and servers.
  - Mandatory to know these if one is trying to write a HTTP client or server.

HTTP 1.1 Client Requirements

- The clients must do the following:
  - Include the `Host:` header with each request.
  - Either support persistent connections, or include the `Connection: close` header with each request.
  - Handle the `100 Continue` response.
  - Accept responses with *chunked data*.

HTTP 1.1 Server Requirements

- The servers must do the following:
  - Require the `Host:` header from HTTP 1.1 clients.
  - Accepts absolute URL's in a request.
  - Accept requests with chunked data.
  - Include the `Date:` header in each response.
  - Support at least the **GET** and **HEAD** methods.
  - Support HTTP 1.0 requests.
  - Either support persistent connections, or include the `Connection: close` header with each request.

HTTP Proxy servers

- What is a HTTP Proxy server?
  - A program that acts as an interface between a client and a server.
  - It receives requests from the clients, and forwards them to the server(s).
  - The responses are sent back in the same way.
  - A proxy thus acts both as a HTTP client and a server.

- **Request from a client to a proxy server differs from normal server requests in one way.**
  - **The complete URL of the resource being requested must be specified.**

```
GET  http://www.xyz.com/docs/abc.txt  HTTP/1.0
```

  - **Required by the proxy to know where to forward the request to.**

Uniform Resource Locators (URL)

What is a URL?

- They are the mechanism by which documents are addressed in the WWW.
- A URL contains the following information:
  - Name of the site containing the resource.
  - The type of service to be used to access the resource (ftp, http, etc.).
  - The port number of the service.
    - Default assumed, if omitted.
  - Location of the resource (path name) in the server.

- URLs specify Internet addresses.
- General format for URL:
  - scheme://address:port/path/filename
- Examples:

  http://www.rediff.com/news/ab1.html

  http://www.xyz.edu:2345/home/rose.jpg

  mailto://skdas@yahoo.co.in

  news:alt.rec.flowers

  ftp://kumar:km123@www.abc.com/docs/paper/x1.pdf

  ftp://www.ftpsite.com/docs/paper1.ps

Sending a Query String

- The mechanism can also be used to send a query string to a specified URL.
  - Used for CGI scripts.
  - Place a question mark at the end of the URL, followed by the query string.

```
http://www.xyz.com/cgi-bin/xyz.pl?Roll=1234&Sex=M
```

End of Lecture 11