CS348: Computer Networks

WWW, HTTP

Dr. Manas Khatua

Assistant Professor

Dept. of CSE, IIT Guwahati

E-mail: manaskhatua@iitg.ac.in

Standard Applications

- We would discuss few standard client-server applications
  - World Wide Web (WWW): Hyper Text Transfer Protocol (HTTP)
  - File Transfer Protocol (FTP)
  - Electronic Mail: SMTP, POP
  - TELNET
  - Secure Shell (ssh)
  - Domain Name System (DNS)

  - Dynamic Host Configuration Protocol (DHCP)
  - Simple Network Management Protocol (SNMP)

WWW

- Until the early 1990s, Internet was essentially unknown outside of the academic and research communities

- The idea of **Web** was first proposed by Tim Berners-Lee in 1989-90 at *CERN*
  - to allow the researchers at different locations to access each others' researches.

- The commercial Web started around 1994.

- The Web today is a *distributed repository of information*
  - the documents, called *web pages*, are distributed all over the world
  - related documents are linked together
  - the linking of web pages was achieved using *hypertext*
  - Today, the term *hypertext* has been changed to *hypermedia* (to indicate text, image, audio, video)

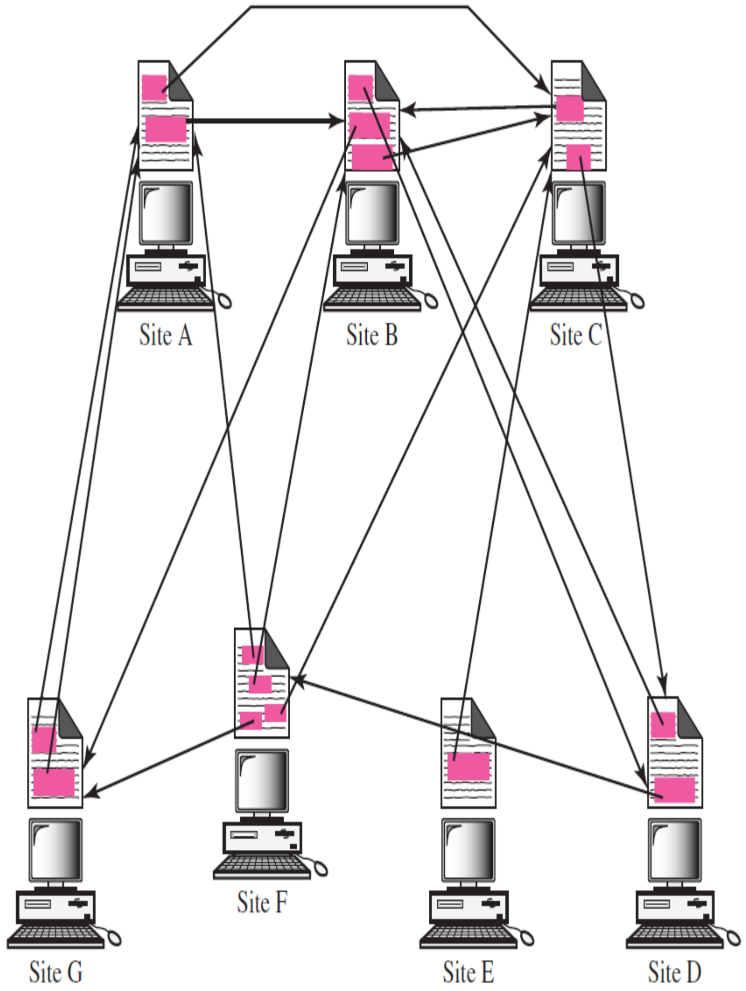
- The Web operates as *on-demand content provider* - users receive what they want, when they want it.

- Any individual user *can make information available* over the Web.

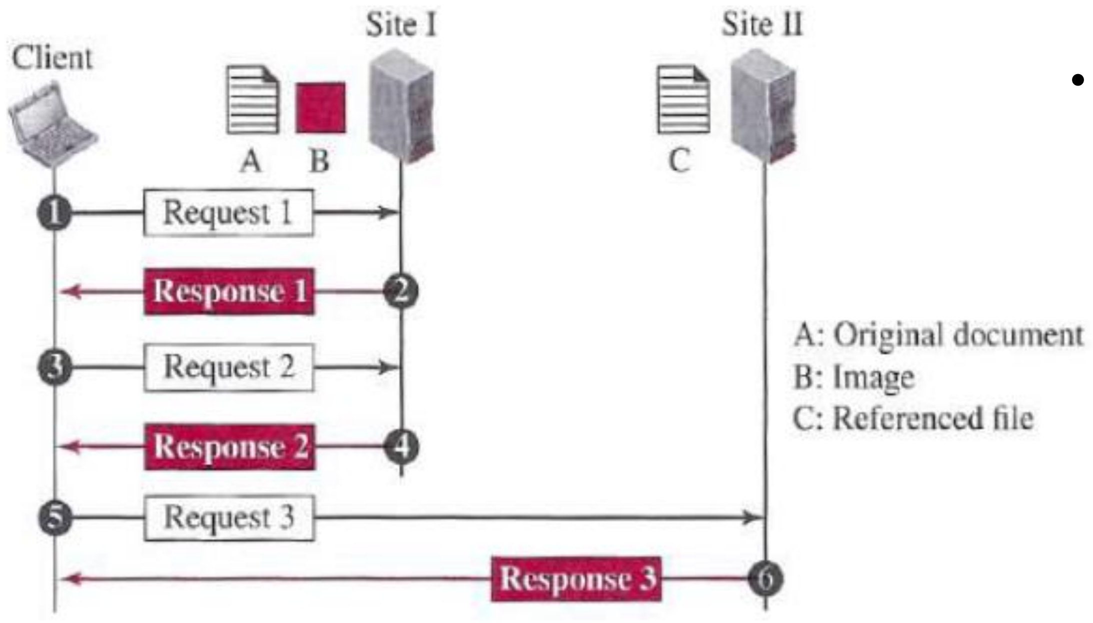
- Web *serves as a platform* for many emerging applications (YouTube, Gmail, Facebook)

Architecture of WWW

- The WWW today is a distributed client-server service
- A client using a browser can access the services running in server
- The provided service is distributed over many locations called *sites*
- Each site holds one or more web pages
- Each web page can contain some links to other web pages in the same or other sites
- Each web page is a file with a name and address.



Cont…

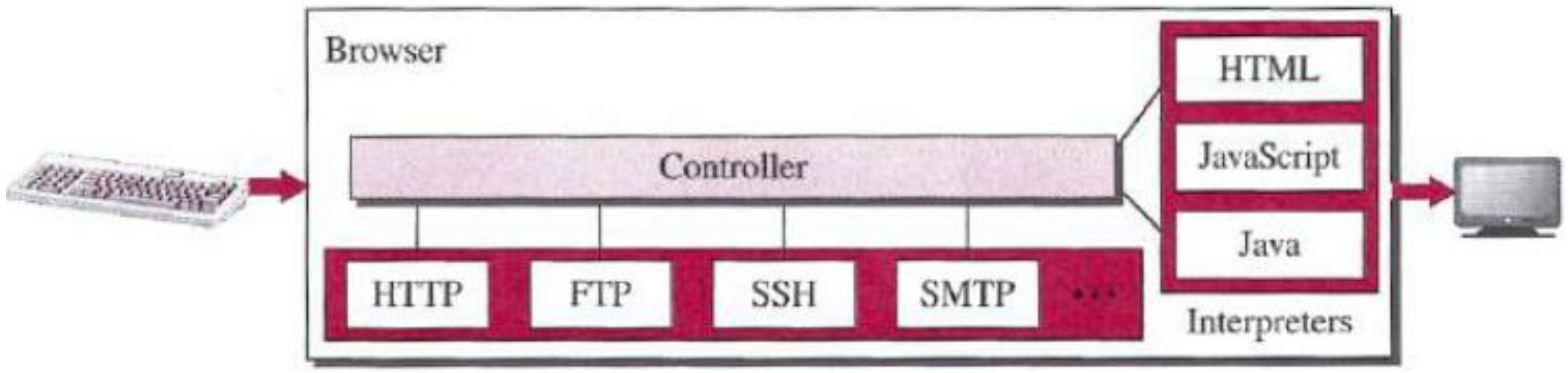


- An example:
  - we need three transactions to see the whole document
  - retrieves a copy of the main document (file A),
  - retrieve a copy of the image (file B)
  - retrieving a copy of file C.

- Web Application consists of
  - Web client
  - Web Server
  - Web document / page
  - URL
  - Application layer protocol

Web Client (Browser)

- Browser interprets and display a web page
- Each browser usually consists of three parts:
  - a controller
  - client protocols (used to access pages)
  - interpreters (used to display pages)
- Popular Web Clients:
  - Internet Explorer
  - Mozilla Firefox
  - Google Chrome



Web Server

- A web server is a computer system that processes requests via HTTP
- The web pages are stored at the server.
- Each time a request arrives, the corresponding document is sent to the client.
- A server can become more efficient using the concept of
  - cash memory, multithreading, multiprocessing.
- Popular Web Servers:
  - Apache
  - Microsoft IIS (Internet Information Server)
  - NGINX
  - Google Web Server (GWS)
  - Dell PowerEdge Web Server

Dell PowerEdge Web Server

Uniform Resource Locator (URL)

- A web page, as a file, needs to have a unique identifier to distinguish it from other web pages.

- To define a web page, we need three identifiers:

  - *host* (can be the IP-address/name of the server)
  - *port* (predefined for the client-server application, so optional)
  - *path* (the object's path name E.g., */top/next/last/myfile.txt*)

- Further, we need to tell the browser what client-server application protocol we want to use, which is called the *protocol.*

- To combine these pieces together, the URL has been designed.

URL:: protocol://host/path

Web Documents

- The documents or pages in WWW can be grouped into
  - Static
  - Dynamic
  - Active

- **Static** Documents:
  - fixed-content documents that are created and stored in a server.
  - i.e., contents of a file are determined when the file is created, not when it is used.
  - Static documents are prepared using :
    - HyperText Markup Language (HTML) - for creating web pages and web applications.
    - Extensible Markup Language (XML) - for encoding documents in a format that is both human-readable and machine-readable.
    - Extensible Style Language (XSL) - for expressing style sheets such as CSS
    - Extensible Hypertext Markup Language (XHTML) - for creating well-formed document which can be parsed by XML parser

Cont…

- **Dynamic** Documents:
  - A dynamic document is created by a web server whenever a browser requests the document.
  - E.g,: retrieval of the time & date from a server
  - to retrieve a dynamic document we use:
    - *Java Server Pages (JSP)*
    - *Active Server Pages (ASP)*
    - *ColdFusion*
- **Active** Documents:
  - When we need a program or a script to be run at the client site
  - E.g.: a program that interacts with the user.
  - One way to create active document is to use Java applets, a program written in Java on the server. It is compiled and ready to be run.
  - Another way is to use JavaScripts, but download and run the script at the client site.

Hypertext Transfer Protocol (HTTP)

- HTTP defines
  - how Web clients request Web pages from Web servers, and
  - how servers transfer Web pages to clients.
- HTTP uses the services of TCP.
  - TCP provides a reliable data transfer service to HTTP.
  - HTTP need not worry about lost data or the details of how TCP recovers from loss or reordering of data within the network.
- An HTTP client sends a request; an HTTP server returns a response.
- The HTTP client first initiates a TCP connection with the server.
- On the client side the socket interface is the door between the client process and the TCP connection; on the server side it is the door between the server process and the TCP connection.
- Client / Server sent(/receive) HTTP message into (/from) sockets interface.
- HTTP is said to be a **stateless protocol**, because an HTTP server maintains no information about the clients

Cont…

- In Internet application, the client and server communicate for an extended period of time – a series of requests may be made back-to-back, periodically at regular intervals, or intermittently.
  - Should each request/response pair be sent over a separate TCP connection or the same TCP connection?

- HTTP uses both the non-persistent and persistent connections.
- But, the default setup is persistent one.

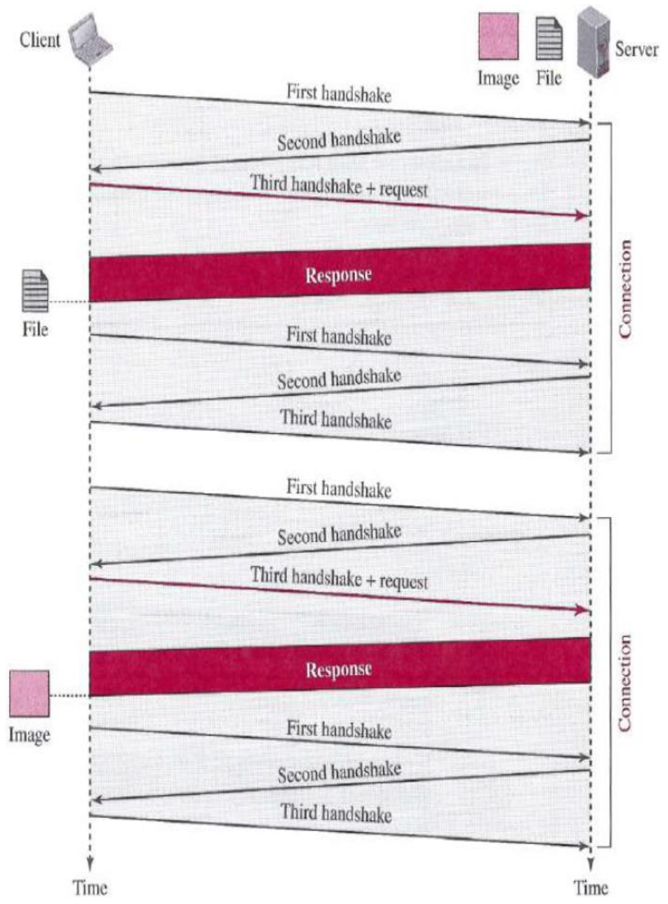
- If the web pages are located on different servers
  - *Non-persistent connection -* Create a new TCP connection for retrieving each object

- If some of the objects are located on the same server
  - *Non-persistent connection:* retrieve each object using a new TCP connection.
  - *Persistent connection:* make a TCP connection and retrieve them all

HTTP with Non-persistent Connection

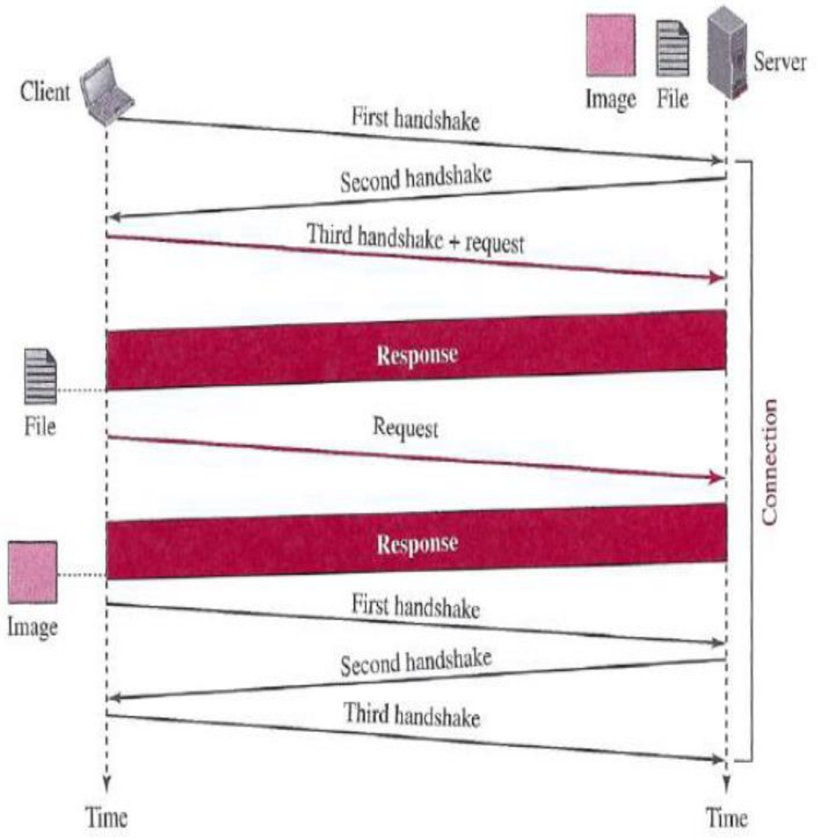
Let’s a page consists of a base HTML file and 10 JPEG images, and that all 11 of these objects reside on the same server. URL: http://www.iitg.ac.in/cse/home.index

1. HTTP client process initiates a TCP connection to the server www.iitg.ac.in on port number 80
2. The HTTP client sends an HTTP request message to the server via its socket
3. The HTTP server process receives the request message via its socket, retrieves the requested object from its storage, encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.
4. The HTTP server process tells TCP to close the TCP connection.
5. The HTTP client receives the response message. The TCP connection terminates.
6. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
7. The first four steps are then repeated for each of the referenced JPEG objects.



HTTP with Persistent Connection

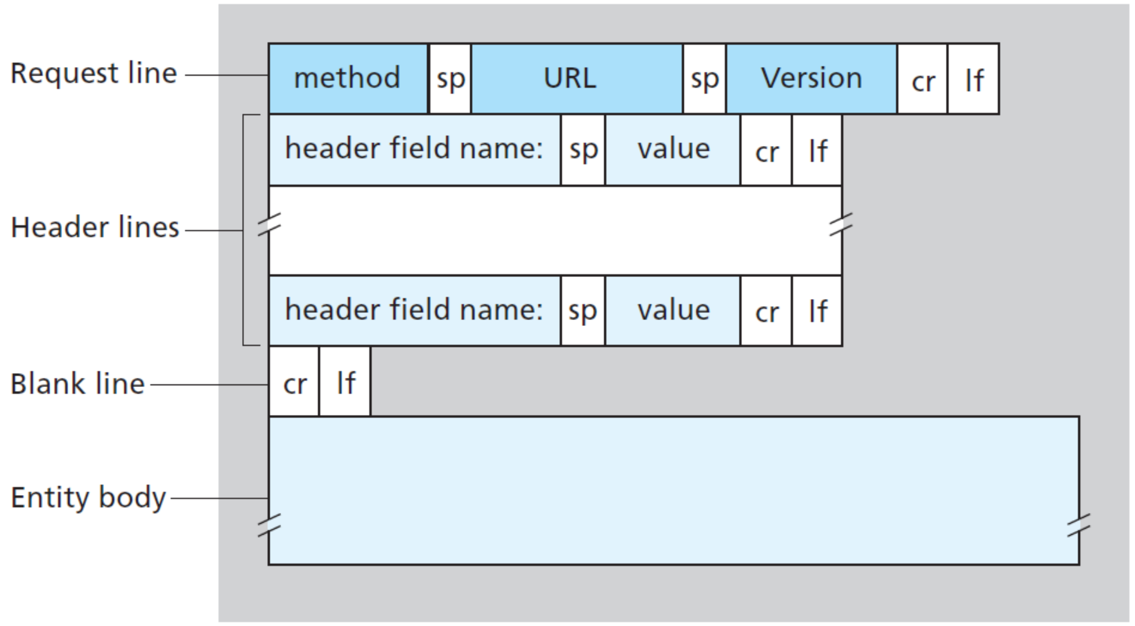
- An entire Web page (in the example above, the base HTML file and the 10 images) can be sent in a single persistent TCP connection.
- multiple Web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection also
- Typically, the HTTP server closes a connection when it isn't used for a certain time (a configurable timeout interval).



Persistent vs Non-persistent

- Disadvantages of Non-persistent connections:
  - a brand-new connection must be established and maintained for *each requested object*
  - each object suffers a delivery delay of two RTTs— one RTT to establish the TCP connection and one RTT to request and receive an object.
- Advantages of persistent connections :
  - Lower CPU and memory usage because there are less number of connections.
  - Allows HTTP pipelining of requests and responses.
  - Reduced network congestion (fewer TCP connections).
  - Reduced latency in subsequent requests (no handshaking).
  - Errors can be reported without the penalty of closing the TCP connection.
- Disadvantages of persistent connections :
  - Resources may be kept occupied even when not needed and may not be available to others.

HTTP Request Message Format



A request message consists of
(i) a request line,
(ii) zero or more header lines,
(iii) a blank line,
(iv) a body.

There are three fields (*method, URL, version*) in request line separated by one space and terminated by two characters \n, \r.

A line feed (\n) means moving one line forward.
A carriage return (\r) means moving the cursor to the beginning of the line.

Cont…

- Request Line:
- Method: defines the request types. E.g., GET, PUT, POST, HEAD, …
  - GET: allows the client to send a request
  - PUT: allows the client to post a new web page on the server
  - HEAD: is used when the client needs only some information about the web page from the server
  - POST: is used to send some information to the server to be added to the web page or to modify the web page
  - TRACE: is used for debugging
  - DELETE: allows the client to delete a web page on the server
  - OPTIONS: allows the client to ask about the properties of a web page
  - CONNECT: a reserve method; it may be used by proxy servers
- URL: defines the address and name of the corresponding web page
- Version: gives the version of the protocol

Cont…

- Header Line:
  - After the request line, we can have zero or more *header lines*.
  - Each header line sends additional information from the client to the server.
  - Each header line has a header name (e.g. request, host, date, etc.), a colon, a space, and a header value.
- Entity Body:
  - The body can be present in a request message.
  - Usually, it contains the comment to be sent or the file to be published on the website when the method is PUT or POST.

Cont…

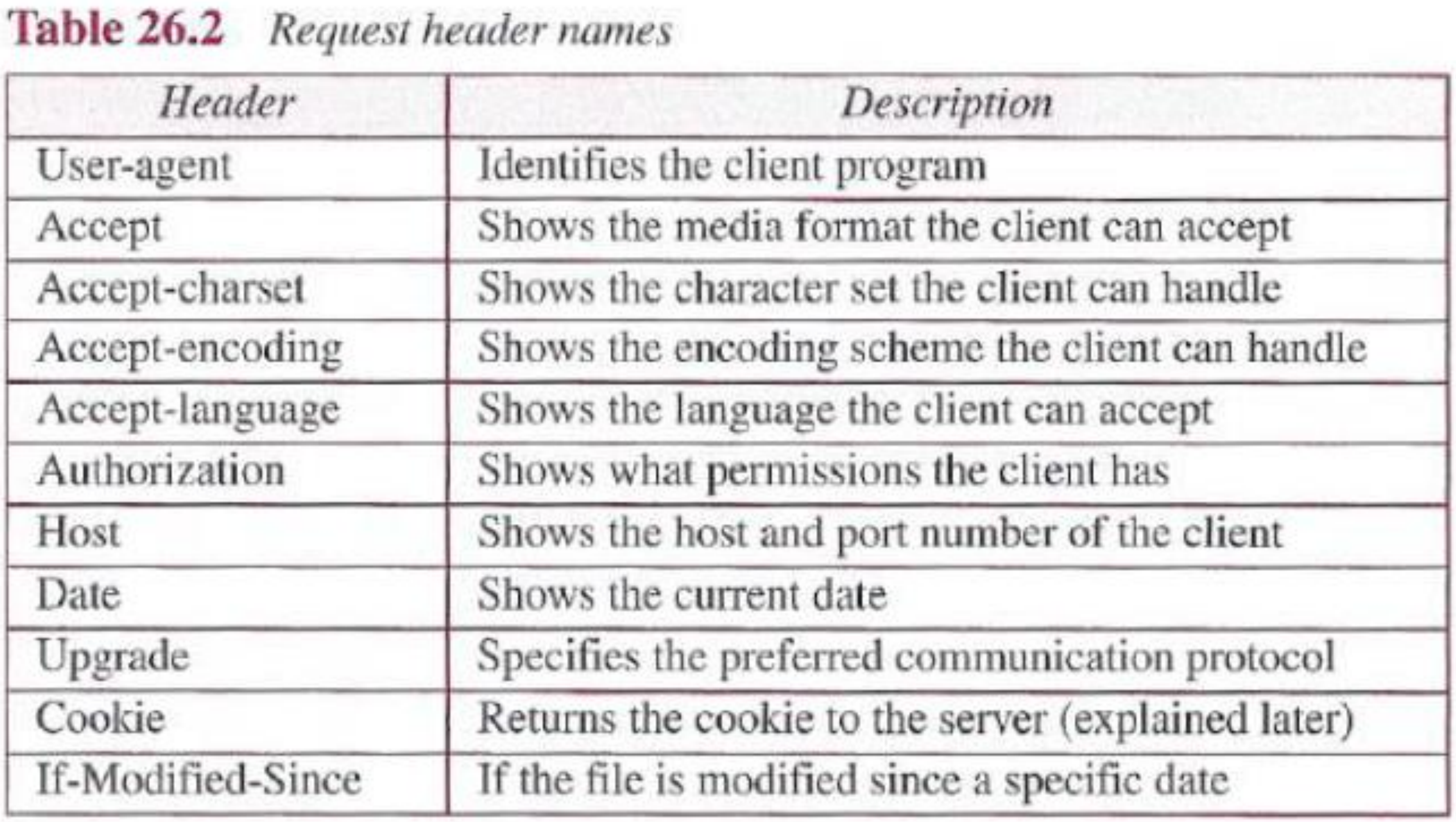
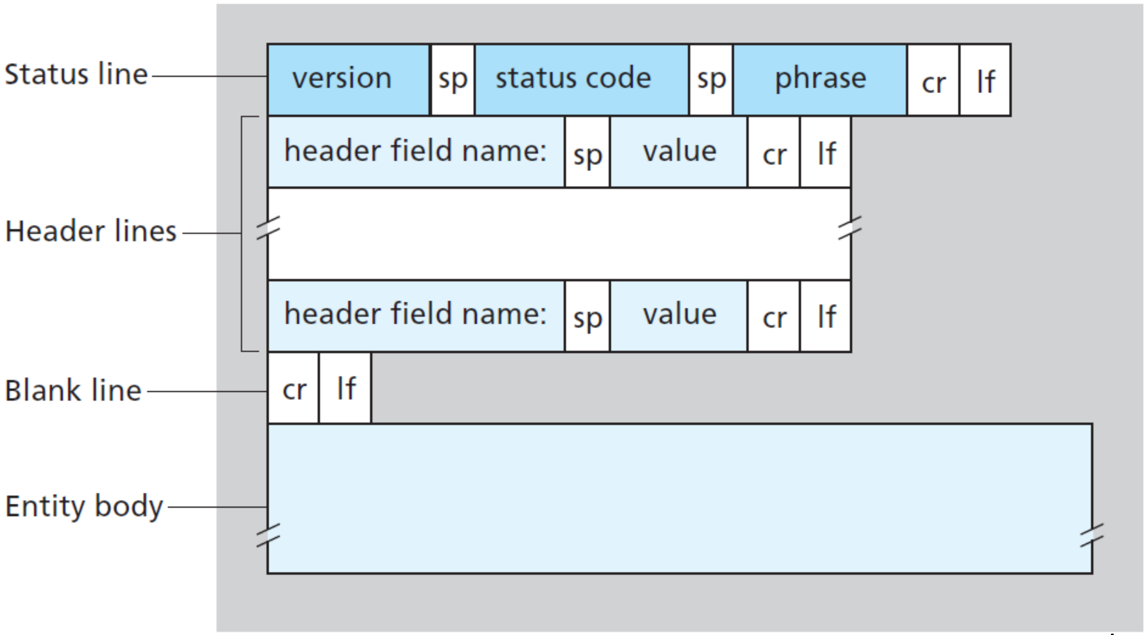


Table 26.2 *Request header names*

Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

HTTP Response Message Format



A response message consists of
- (i) a status line,
- (ii) zero or more header lines,
- (iii) a blank line,
- (iv) a body.

- Status Line:
- There are three fields (version, status code, status phrase) separated by spaces and terminated by \n,\r
  - Version: defines the version of HTTP protocol
  - status code: defines the status of the request
  - status phrase: explains the status code in text

- Header Line:
- We can have zero or more response header lines
- Each header line sends additional information from the server to the client
- Each header line has a header name (e.g. Location, Date, Set-Cookie, etc.) , a colon, a space, and a header value.
- Entity Body:
- The body contains the document to be sent from the server to the client.

Cont…

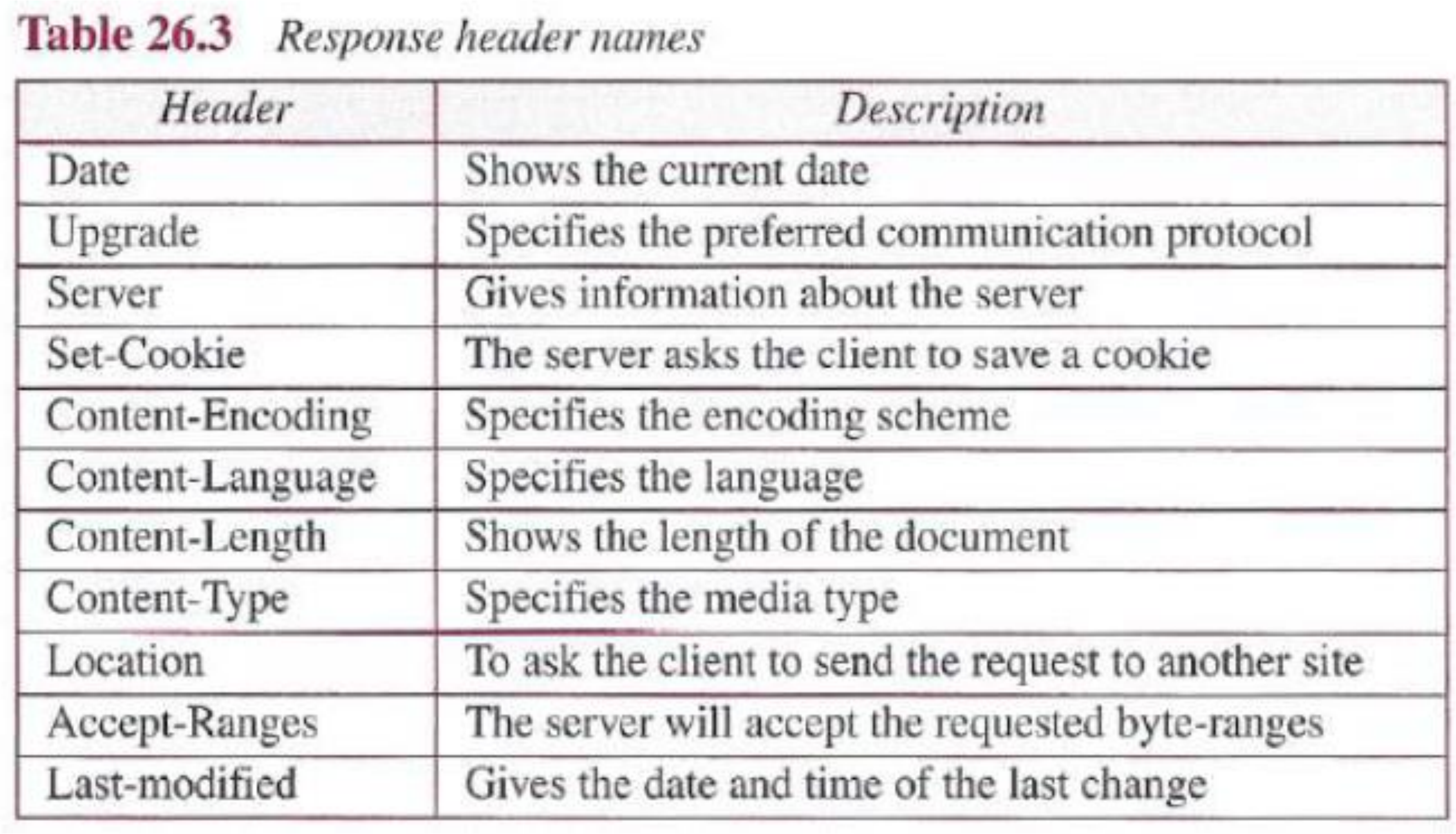
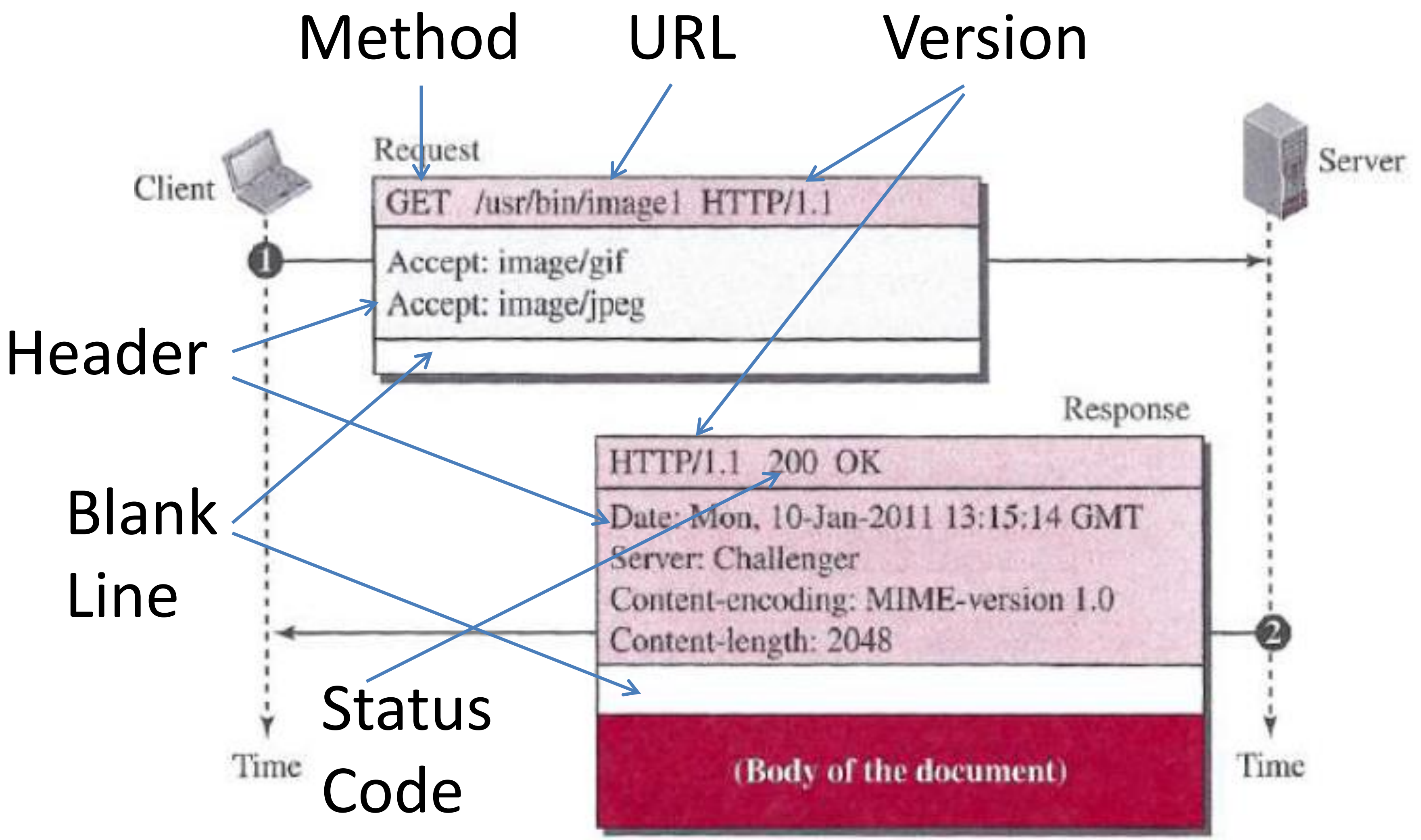


Table 26.3 *Response header names*

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

Example



```
GET  /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1  200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2048

(Body of the document)
```

User-Server Interaction: Cookies

- The WWW was originally designed as a stateless entity.
  - A client sends a request; a server responds. Their relationship is over.

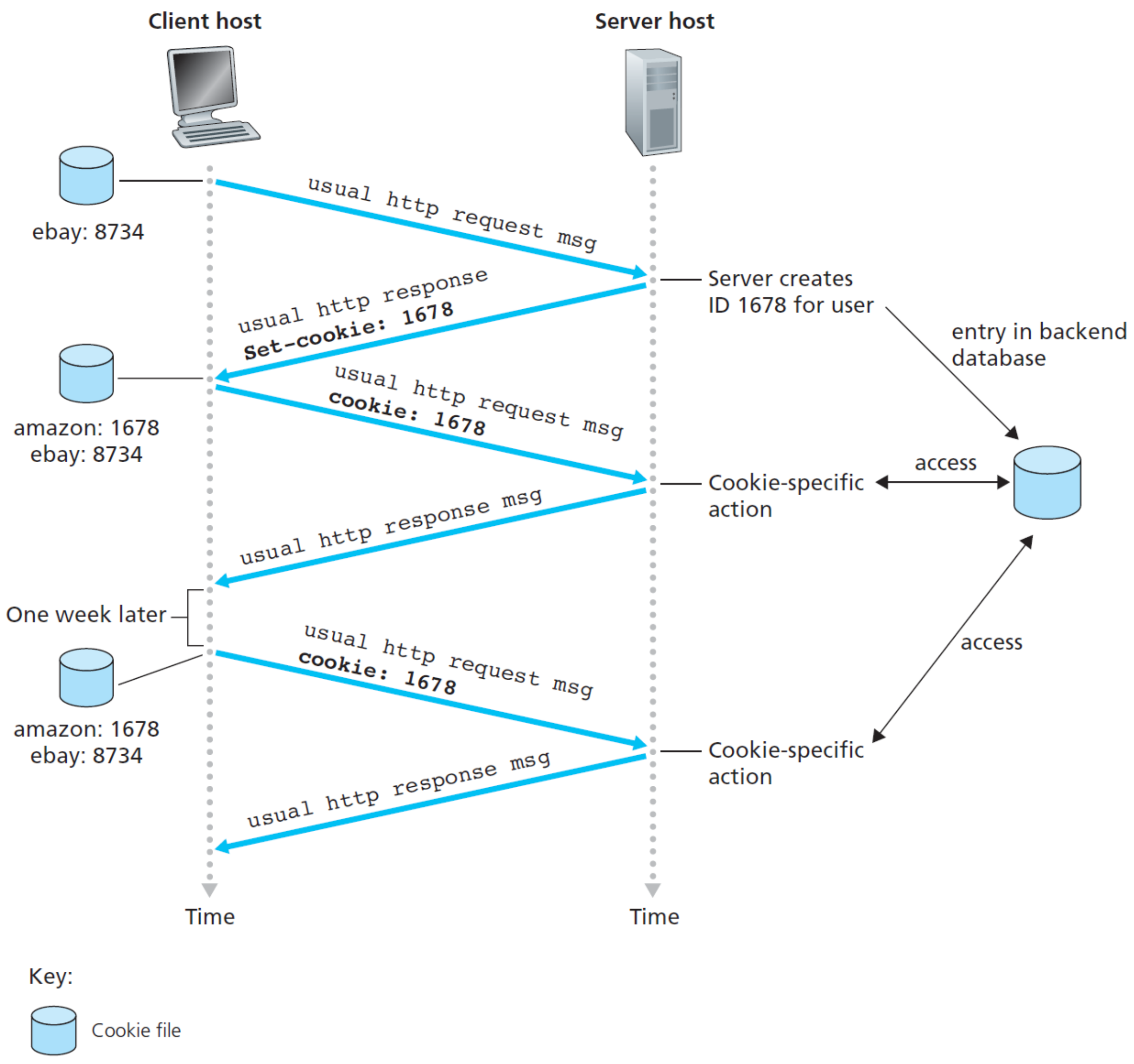
- Today the Web has other functions that need to remember some information about the clients.
  - Websites are being used as *electronic stores* that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
  - Some websites need to allow access to *registered clients* only.
  - Some websites are used as *portals:* user selects the web pages he wants to see.
  - Some websites are just *advertising* agencies.

- It is often desirable for a Web site to identify users,
  - either because the server wishes to restrict user access
  - or because it wants to serve content as a function of the user identity.

- For these purposes, the cookie mechanism was devised.

Cookies allow sites to keep track of users

Keeping user state with cookies



Creating & Storing Cookies

Three steps:

- When a server receives a request from a client, it stores information about the client in a file or a string.
  - The information may include
    - the domain name of the client,
    - the contents of the cookie (information about the client such as name, registration number, and so on),
    - a timestamp,
    - other information depending on the implementation.
- The server includes the cookie in the response that it sends to the client.
- When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the server domain name.

Using Cookies

- When a client sends a request to a server,
  - the browser looks in the cookie directory to see if it can find a cookie sent by that server.
  - If found, the cookie is included in the request.
- When the server receives the request with cookie,
  - it knows that this is an old client, not a new one.
- Note: the contents of the cookie are never read by the browser or disclosed to the user.

It is a cookie *made* by the server and *eaten* by the server !

Applications using Cookie

- *Electronic store* (e-commerce): When a client selects an item and inserts it in a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser. If the client selects a second item, the cookie is updated with the new selection information, and so on.

- *Registered clients:* sends a cookie to the client when the client registers for the first time. For any repeated access, only those clients that send the appropriate cookie are allowed.

- *Web portal:* When a user selects her favourite pages, a cookie is made and sent. If the site is accessed again, the cookie is sent to the server to show what the client is looking for.

- *Advertising agencies*: When a user visits the main website and clicks the icon of a corporation, a request is sent to the advertising agency. The advertising agency sends the requested banner, but it also includes a cookie with the ID of the user.

Web Caching: Proxy Servers

- HTTP supports Web cache i.e. proxy servers.

- A proxy server is a computer that keeps copies of responses to recent requests.
  - The HTTP client sends a request to the proxy server.
  - The proxy server checks its cache.
  - If the response is not stored in the cache, the proxy server sends the request to the corresponding server.
  - Incoming responses are sent to the proxy server and stored for future requests from other clients.

- Advantages:
  - reduces the load on the original server,
  - decreases network traffic,
  - improves latency

Cache Update

- To use the proxy server, the client must be configured to access the proxy instead of the target server.

- The proxy servers are normally located at the client site.
- Typically a Web cache is purchased and installed by an ISP.

How long a response should remain in the proxy server before being deleted and replaced?

- Many Solutions (depending on requirement):
  - store the list of sites whose information remains the same for a while. E.g., news agency
  - add some headers to show the last modification time of the information.

- Through the use of Content Distribution Networks (CDNs), Web caches are increasingly playing an important role in the Internet.

- A CDN company installs many geographically distributed caches throughout the Internet, thereby localizing much of the traffic.
  - shared CDNs (such as Akamai and Limelight)
  - dedicated CDNs (such as Google and Microsoft)

Conditional GET

- Although caching can reduce user-perceived response times, it introduces a new problem—
  - the copy of an object residing in the cache may be stale.

- Solution:
  - HTTP has a mechanism that allows a cache to verify that its objects are up to date.
  - It is conditional GET
  - a client can add a condition in its request
    - If-Modified-Since: header line.

- In this case, the server will send the requested web page if the condition is met or inform the client otherwise.

Cont…

- The following shows how a client imposes the modification date and time condition on a request.

GET http://www.commonServer.com/file1 HTTP/1.1	Request line
If-Modified-Since: Thu, Sept 04 00:00:00 GMT	Header line
	Blank line
(Empty Body)	Empty body

- The status line in the response shows the file was not modified after the defined point in time. The body of the response message is also empty.

HTTP/1.1 304 Not Modified	Status line
Date: Sat, Sept 06 08 16:22:46GMT	First header line
Server: commonServer.com	Second header line
	Blank line
(Empty Body)	Empty body

HTTP Security

- HTTP per se does not provide security.

- But, HTTP can be run over the Secure Socket Layer (SSL).
- In this case, HTTP is referred to as HTTPS.

- HTTPS provides
  - Confidentiality,
  - User Authentication (for client and server)
  - Data Integrity.

Thanks!