[Unit 2: Issues of Web Technology]
Web Technology (CSC-353)

Jagdish Bhatta

Central Department of Computer Science & Information Technology

Tribhuvan University

Architectural Issues of Web Layer:

The **web layer** is also referred to as the UI layer. The web layer is primarily concerned with presenting the user interface and the behavior of the application (handling user interactions/events). While the web layer can also contain logic, core application logic is usually located in the services layer. T**he three Layers within the Web Layer are:**

- **HTML-The Content Layer:** The content layer is where you store all the content that your customers want to read or look at. This includes text and images as well as multimedia. It's also important to make sure that every aspect of your site is represented in the content layer. That way, your customers who have JavaScript turned off or can't view CSS will still have access to the entire site, if not all the functionality.

- **CSS - the Styles Layer:** Store all your styles for your Web site in an external style sheet. This defines the way the pages should look, and you can have separate style sheets for various media types. Store your CSS in an external style sheet so that you can get the benefits of the style layer across the site.

- **JavaScript - the Behavior Layer:** JavaScript is the most commonly used language for writing the behavior layer; ASP, CGI and PHP can also generate Web page behaviors. However, when most developers refer to the behavior layer, they mean that layer that is activated directly in the Web browser - so JavaScript is nearly always the language of choice. You use this layer to interact directly with the DOM or Document Object Model.

When you're creating a Web page, it is important to keep the layers separate. Using external style sheets is the best way to separate your content from your design. And the same is true for using external JavaScript files. Some of the benefits of separating the layers are:

- **Shared resources:** When you write an external CSS file or JavaScript file, you can use that file by any page on your Web site. There is no duplication of effort, and whenever the file changes, it changes for every page that uses it without you making more than one change.

- **Faster downloads:** Once the script or stylesheet has been downloaded by your customer the first time, it is cached. Then every other page that is downloaded loads more quickly in the browser window.

- **Multi-person teams:** If you have more than one person working on a Web site at once, you can divide up the workload without worrying about permissions or content management. You can also hire people who are style/design experts to work on the CSS while your scripters work on the JavaScript, and your writers work in the content files.

- **Accessibility:** External style sheets and script files are more accessible to more browsers, because they can be ignored more easily, and because they provide more options. For example, you can set up a style sheet that is displayed only for screen readers or a script library that's only used by people on cell phones.

- **Backwards compatibility:** When you have a site that is designed with the development layers, it will be more backwards compatible because browsers that can't use technology like CSS and JavaScript can still view the HTML.

HTTP (Hypertext Transfer Protocol):

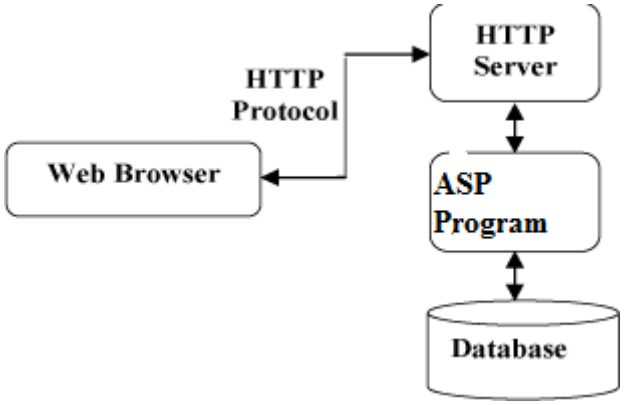
HTTP stands for **Hypertext Transfer Protocol**. It is a TCP/IP based communication protocol which is used to deliver virtually all files and other data, collectively called resources, on the World Wide Web. These resources could be HTML files, image files, query results, or anything else. A browser is works as an HTTP client because it sends requests to an HTTP server which is called Web server. The Web Server then sends responses back to the client. The standard and default port for HTTP servers to listen on is 80 but it can be changed to any other port like 8080 etc. There are three important things about HTTP of which you should be aware:

- **HTTP is connectionless:** After a request is made, the client disconnects from the server and waits for a response. The server must re-establish the connection after it processes the request.

- **HTTP is media independent:** Any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. How content is handled is determined by the MIME specification.

- **HTTP is stateless:** This is a direct result of HTTP's being connectionless. The server and client are aware of each other only during a request. Afterwards, each forgets the other. For this reason neither the client nor the browser can retain information between different requests across the web pages.

Following diagram shows where HTTP Protocol fits in communication;



Like most network protocols, HTTP uses the client-server model: An HTTP client opens a connection and sends a request message to an HTTP server; the server then returns a response message, usually containing the resource that was requested. After delivering the response, the server closes the connection. The format of the request and response messages is similar and will have following structure:

- An initial line CRLF
- Zero or more header lines CRLF
- A blank line i.e. a CRLF
- An optional message body like file, query data or query output.

CR and LF here mean ASCII values 13 and 10. The initial line is different for the request than for the response. A request line has three parts, separated by spaces: An HTTP Method Name, the local path of the requested resource, the version of HTTP being used. Example of initial line for Request Message is: "GET /path/to/file/index.html HTTP/1.0". The initial response line, called the status line, also has three parts separated by spaces: The version of HTTP being used, a response status code that gives the result of the request, an English reason phrase describing the status code. Example, HTTP/1.0 200 OK or "HTTP/1.0 404 Not Found"

Header lines provide information about the request or response, or about the object sent in the message body. The header lines are in the usual text header format, which is: one line per header, of the form "Header-Name: value", ending with CRLF. Example of Header Line is "User-agent: Mozilla/3.0Gold" or "Last-Modified: Fri, 31 Dec 1999 23:59:59 GMT".

An HTTP message may have a body of data sent after the header lines. In a response, this is where the requested resource is returned to the client (the most common use of the message body), or perhaps explanatory text if there's an error. In a request, this is where user-entered data or uploaded files are sent to the server.

HTTP: header fields

HTTP header fields are components of the message header of requests and responses in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

The header fields are transmitted after the request or response line, the first line of a message. Header fields are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence. The end of the header fields is indicated by an empty field, resulting in the transmission of two consecutive CR-LF pairs. Long lines can be folded into multiple lines; continuation lines are indicated by presence of space (SP) or horizontal tab (HT) as first character on next line. Few fields can also contain comments (i.e. in. User-Agent, Server, Via fields), which can be ignored by software.

There are no limits to size of each header field name or value, or number of headers in standard itself. However most servers, clients and proxy software, impose some limits for practical and security reasons. For example; Apache 2.3 server by default limits each header size to 8190 bytes, and there can be at most 100 headers in single request.

HTTP Session:

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80; see List of TCP and UDP port numbers). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested resource, an error message, or some other information

File Transfer Protocol:

File Transfer Protocol (FTP) lives up to its name and provides a method for transferring files over a network from one computer to another. More generally, it provides for some simple file management on the contents of a remote computer. It is an old protocol and is used less than it was before the World Wide Web came along. Today, its primary use is uploading files to a Web site. It can also be used for downloading from the Web but, more often than not, downloading is done via HTTP. Sites that have a lot of downloading (software sites, for example) will often have an FTP server to handle the traffic. If FTP is involved, the URL will have *ftp:* at the front.

The File Transfer Protocol is used to send files from one system to another under user commands. Both text and binary files are accommodated and the protocol provides features for controlling user access. When a user wishes to engage in File transfer, FTP sets up a TCP connection to the target system for the exchange of control messages. These allow used ID and password to be transmitted and allow the user to specify the file and file action desired. Once file transfer is approved, a second TCP connection is set up for data transfer. The file is transferred over the data connection, without the overhead of headers, or control information at the application level. When the transfer is complete, the control connection is used to signal the completion and to accept new file transfer commands.

FTP can be run in active or passive mode, which determines how the data connection is established. In active mode, the client sends the server the IP address and port number on which the client will listen, and the server initiates the TCP connection. at the condition when the client is behind a firewall and unable to accept incoming TCP connections, passive mode may be used. In this mode the client sends a PASV command to the server and receives an IP address and port number in return. The client uses these to open the data connection to the server. Data transfer can be done in any of three modes:

- Stream mode: Data is sent as a continuous stream, relieving FTP from doing any processing. Rather, all processing is left up to TCP. No End-of-file indicator is needed, unless the data is divided into records.

- Block mode: FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.

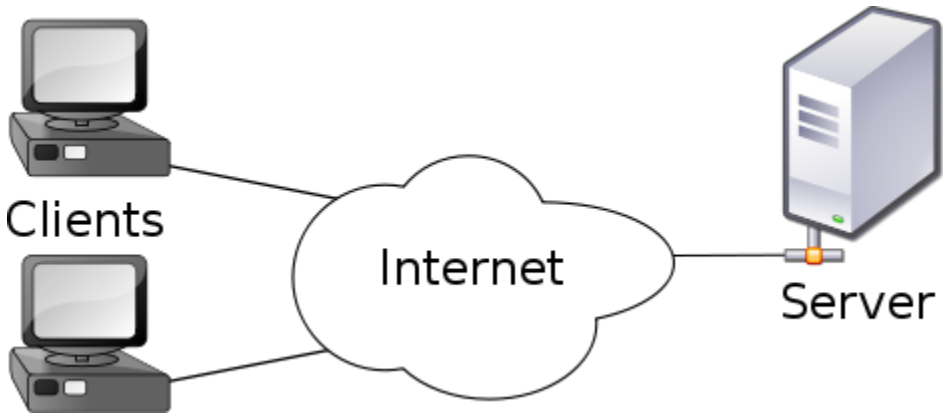
- Compressed mode: Data is compressed using a single algorithm (usually run-length encoding).

Client/Server Model:

The **client–server model** is a computing model that acts as distributed application which partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Client/Server Architecture:

Client server network architecture consists of two kinds of computers: clients and servers. Clients are the computers that that do not share any of its resources but requests data and other services from the server computers and server computers provide services to the client computers by responding to client computers requests. Normally servers are powerful computers and clients are less powerful personal computers. Web servers are included as part of a larger package of internet and intranet related programs for serving e-mail, downloading requests for FTP files and building and publishing web pages.



Advantages

- The client/ server architecture reduces network traffic by providing a query response to the user rather than transferring total files.
- The client/ server model improves multi-user updating through a graphical user interface (GUI) front end to the shared database.
- Easy to implement security policies, since the data are stored in central location
- Simplified network administration

Disadvantages

- Failure of the server causes whole network to be collapsed
- Expensive than P2P, Dedicated powerful servers are needed
- Extra effort are needed for administering and managing the server.

Client/Sever architecture can be of different model based on the number of layers it holds. Some of them are;

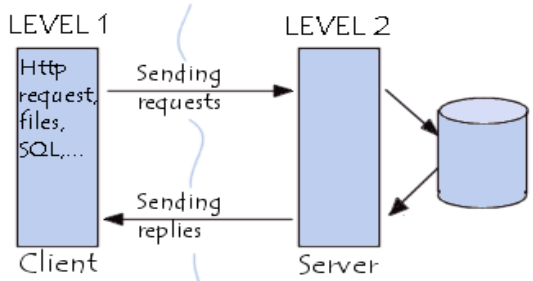
- **2-Tier Architecture**

  2-tier architecture is used to describe client/server systems where the client requests resources and the server responds directly to the request, using its own resources. This means that the server does not call on another application in order to provide part of the service. It runs the client processes separately from the server processes, usually on a different computer:

  - The client processes provide an interface for the customer, and gather and present data usually on the customer's computer. This part of the application is the presentation layer

  - The server processes provide an interface with the data store of the business. This part of the application is the data layer

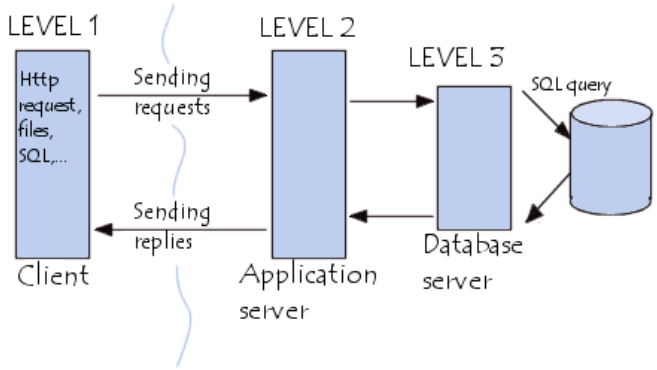
  - The business logic that validates data, monitors security and permissions, and performs other business rules can be housed on either the client or the server, or split between the two.
    - Fundamental units of work required to complete the business process
    - Business rules can be automated by an application program.



- **3-Tier Architecture**

  In 3-tier architecture, there is an intermediary level, meaning the architecture is generally split up between:

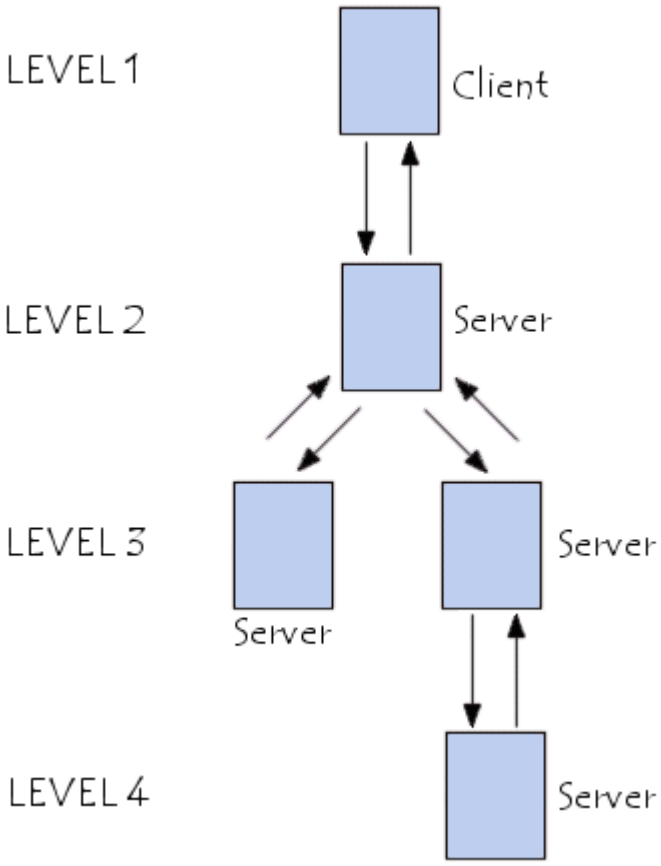
  - A client, i.e. the computer, which requests the resources, equipped with a user interface (usually a web browser) for presentation purposes
  - The application server (also called **middleware**), whose task it is to provide the requested resources, but by calling on another server
  - The data server, which provides the application server with the data it requires



- **N-Tier Architecture (multi-tier)**

  N-tier architecture (with N more than 3) is really 3 tier architectures in which the middle tier is split up into new tiers. The application tier is broken down into separate parts. What these parts are differs from system to system. The following picture shows it:

  The primary advantage of N-tier architectures is that they make load balancing possible. Since the application logic is distributed between several servers, processing can then be more evenly distributed among those servers. N-tiered architectures are also more easily scalable, since only servers experiencing high demand, such as the application server, need be upgraded. The primary disadvantage of N-tier architectures is that it is also more difficult to program and test an N-tier architecture due to its increased complexity.



Advantages of Multi-Tier Client/Server architectures include:

- Changes to the user interface or to the application logic are largely independent from one another, allowing the application to evolve easily to meet new requirements.
- Network bottlenecks are minimized because the application layer does not transmit extra data to the client, only what is needed to handle a task.
- The client is insulated from database and network operations. The client can access data easily and quickly without having to know where data is or how many servers are on the system.
- Database connections can be 'pooled' and thus shared by several users, which greatly reduces the cost associated with per-user licensing.

- The organization has database independence because the data layer is written using standard SQL which is platform independent. The enterprise is not tied to vendor-specific stored procedures.

- The application layer can be written in standard third or fourth generation languages, such as ASP, PHP with which the organization's in-house programmers are experienced.

What kind of systems can benefit?

Generally, any Client/Server system can be implemented in an 'N-Tier' architecture, where application logic is partitioned among various servers. This application partitioning creates an integrated information infrastructure which enables consistent, secure, and global access to critical data. A significant reduction in network traffic, which leads to faster network communications, greater reliability, and greater overall performance is also made possible in a 'N-Tier' Client/Server architecture.