KTU NOTES

MODULE 5

Client/Server Technology and Web Services

Web Services History

- The World Wide Web was developed by Tim Berners-Lee for his employer CERN or the European Organization for Nuclear Research between 1989-1991.
- In 1990, he wrote the program for the World Wide Web. This program created the first web browser and HTML editor. It was the first program to use both FTP and HTTP.
- FTP (File Transfer Protocol) is used to transfer data over a network. HTTP is the protocol that supports hyper-text documents. Both of these protocols are necessary for communication over the Internet or World Wide Web.

- The source code for the World Wide Web was made public in 1993, making it available to everyone with a computer. The technology continued to develop and between 1991-1994, extended from communication only between scientific organizations, to universities and, finally, to industry.

- The first web server, also written by Berners-Lee, ran on NeXTSTEP, the operating system for NeXT computers.

- The other technology authored by Berners-Lee that is required for Web communication is URLs (Universal Resource Locators).

- Berners-Lee is also responsible for writing the initial specifications for HTML. The first web server was installed in the United States on December 12, 1991

- In 1994, Berners-Lee created the World Wide Web Consortium (WWC) to regulate and standardize the various technologies required for Web construction and communication.

- After 2000, the web exploded. Till date, there exist more than 110 million web sites on the World Wide Web.

Web Server Technology

At the most basic level, the process for web communication works as follows:

- a computer runs a web browser that allows it to request, communicate and display HTML documents (web pages).
- Web browsers are the software applications that allow users to access and view these web pages and they run on individual computers.
- The most popular web browsers are Internet Explorer, Mozilla, Firefox, and Safari (for Mac).
- After typing in the URL (or address) and pressing return, the request is sent to a server machine that runs the web server.
- The web server is the program that delivers the files that make up web pages.

  Every web site or computer that creates a web site requires a web server. The most popular web server program is Apache.

The server machine then returns the requested web page. See the Fig. 8.1(a) and 8.1(b), depicting the Web Technology yesterday and today.

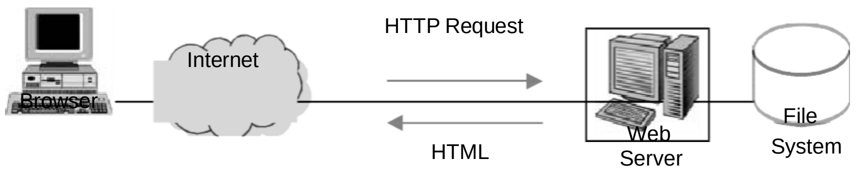


Fig. 8.1(a): Web Technology Yesterday

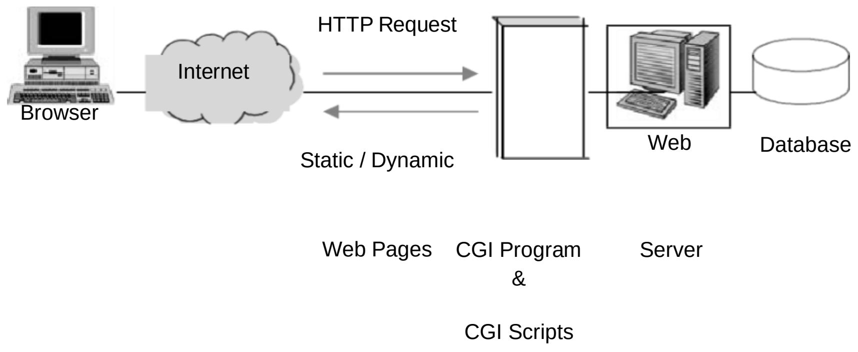


Fig. 8.1(b): Web Technology Today

Communication over the Internet can be broken down into two interested parties:

clients and servers.

The machines providing services are servers.

Clients are the machines used to connect to those services.

For example, the personal computer requesting web pages according to search parameters (defined by key words) does not provide any services to other computers. This is the client.

If the client requests a search from, for example, the search engine Yahoo!. Yahoo! is the server, providing the hardware machinery to service the request. As previously mentioned, each computer requesting information over the Internet requires a web server program like Apache to render the search result intelligible in HTML.

Web servers translate URL path components in local file systems. The URL path is dependent on the server's root directory. The root directory is the top directory of a file system that usually exists hierarchically as an inverted tree.

The typical client request reads, for example, "http://www.example.com/path/file.html". This client web browser translates this request through an HTTP request and by connecting to "www.example.com", in this case. The web server will then add the requested path to its root directory path. The result is located in the server's local file system or hierarchy of directories. The server reads the file and responds to the browser's request. The response contains the requested documents, in this case, web sites and the constituent pages.

Web Browser (Web Client)

A browser is a software (the most popular web browsers are Internet Explorer, Mozilla Firefox, Safari, Opera, and Netscape) that acts as an interface between the user and the inner workings of the internet, specifically the Word Wide Web .

Browsers are also referred to as web clients, or Universal Clients, because in the Client/Server model, the browser functions as the client program.

The browser acts on behalf of the user.

The browser:

- Contacts a web server and sends a request for information.
- Receives the information and then displays it on the user's computer.

A browser can be text-based or graphical and can make the internet easier to use.

The WWW incorporates hypertext, photographs, sound, video, etc. that can be fully experienced through a graphical browser.

Accessing Database on the Web Page

Generally, it has been observed that a remote user's web browser cannot get connected directly with database system.

But in most of the cases, the browsers are a program running on the web server that is an intermediary to the database.

This program can be a Common Gateway Interface (CGI) script, a Java servlet, or some code that lives inside an Active Server Page (ASP) or Java Server Page (JSP) document. The program retrieves the information from the page is an ordinary HTML document or the output of some script that Web-based database system.

Step1: The user types in a URL or fills out a form or submits a search on a Web page and clicks the Submit button.

Step 2: The browser sends the user's query from the browser to the Web server, which passes it on to a CGI script.

Step 3: The CGI script loads a library that lets it talk to an SQL database server, and it uses that library to send SQL commands to the database server.

Step 4: The database server executes the SQL commands and sends the request information to the CGI script.

Step 5: The CGI script generates an HTML document and writes the HTML document to the Web server.

Step 6: The Web server sends the HTML page back to the remote user.

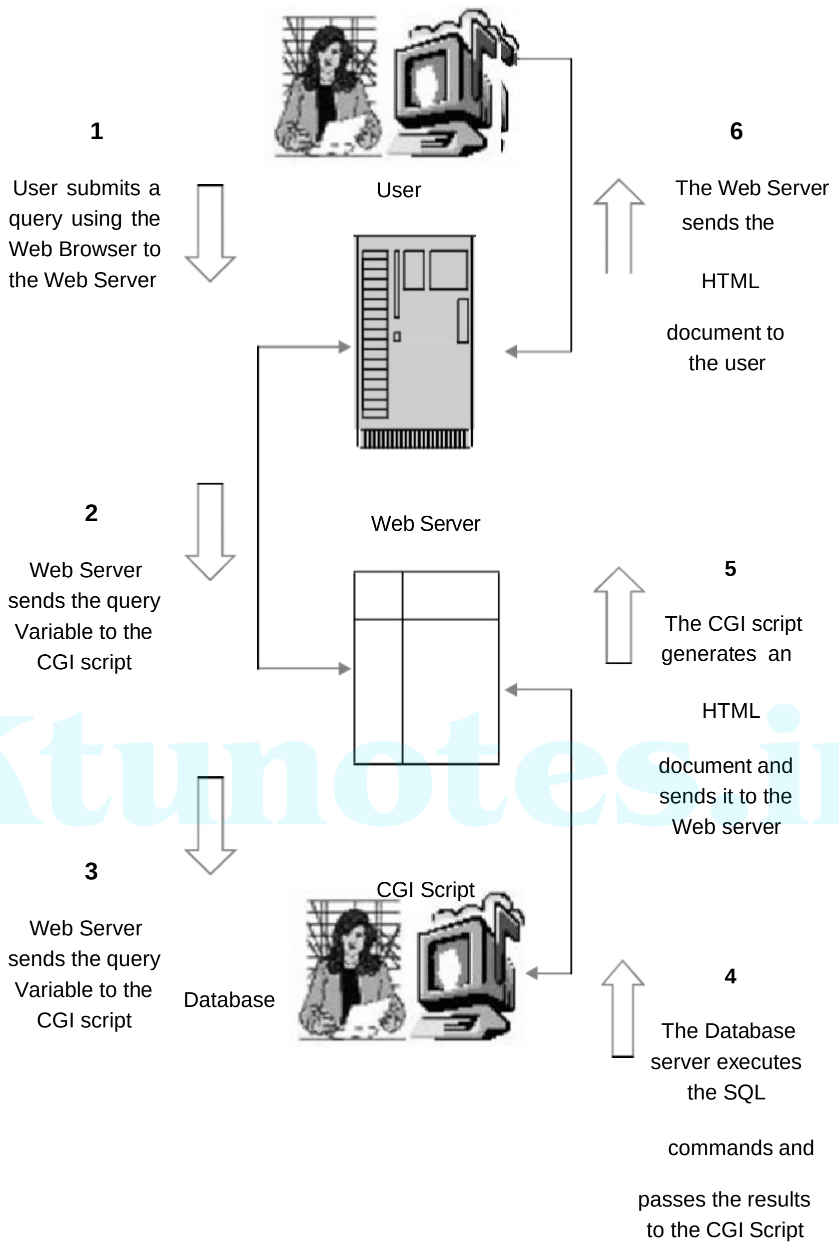

1
User submits a query using the Web Browser to the Web Server
User
6
The Web Server sends the HTML document to the user
2
Web Server sends the query Variable to the CGI script
Web Server
5
The CGI script generates an HTML document and sends it to the Web server
3
Web Server sends the query Variable to the CGI script
CGI Script
Database
4
The Database server executes the SQL commands and passes the results to the CGI Script

Fig. 8.2: Information Exchange between User and a Web-based Database

If the user has send some information to update a database. Then the CGI Script will generate the appropriate SQL commands and send it to the database server. The database server will execute the SQL commands and then inform the user about the result of execution.

Web Server

A computer that runs a software program that is responsible for accepting HTTP requests from clients, which are known as web browsers, and serving them HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects (images, etc.).

HTTP: Every web server program operates by accepting HTTP requests from the client, and providing an HTTP response to the client.

The HTTP response usually consists of an HTML document, but can also be a raw file, an image, or some other type of document (defined by MIME(Multipurpose Internet Mail extensions),a media type); if some error is found in client request or while trying to serve the request, a web server has to send an error response which may include some custom HTML or text messages to better explain the problem to end users.

Logging:web servers have also the capability of logging(the process of capturing,storing and displaying datasets for some activities)some detailed information, about client requests and server responses, to log files; this allows the webmaster to collect statistics by running log analyzers on log files.

In practice many web servers implement the following features also:

- Authentication, optional authorization request (request of user name and password) before allowing access to some or all kind of resources.
- Handling of not only static content (file content recorded in server's filesystem(s)) but of dynamic content too by supporting one or more related interfaces (SSI, CGI, SCGI, FastCGI, JSP, PHP, ASP, ASP.NET, Server API such as NSAPI, ISAPI, etc.).
- HTTPS support (by SSL or TLS) to allow secure (encrypted) connections to the

server on the standard port 443 instead of usual port 80.

- Content compression (i.e., by gzip encoding) to reduce the size of the responses (to lower bandwidth usage, etc.).
- Virtual hosting to serve many web sites using one IP address.
- Large file support to be able to serve files whose size is greater than 2 GB on 32 bit OS.
- Bandwidth throttling to limit the speed of responses in order to not saturate the network and to be able to serve more clients.

Web servers must also be able to manage static and dynamic content. Static content exists as a file in a file system. Dynamic content is content (text, images, form fields) on a web page that changes according to specific contexts or conditions.

Dynamic content is produced by some other program or script (a user-friendly programming language that connects existing components to execute a specific task) or API (Application Programming Interface the web server calls upon).

It is much slower to load than static content since it often has to be pulled from a remote database. It provides a greater degree of user interactivity and tailor responses to user requests.

To handle dynamic content, web servers must support at least any one of interface like JSP (Java Server Pages), PHP (a programming language that creates dynamic web pages), ASP (Active Server Pages) or ASP.NET (it is the successor of ASP).

Web Server Communication

Web servers are one of the end points in communication through the World Wide Web. The World Wide Web is the global structure of electronically connected information. It refers to the global connections between computers that allow users to search for documents or web pages by requesting results from a web server.

These documents are hyper-text based (written in HTML- Hypertext Markup Language), allowing users to travel to other pages and extend theirresearch through links.

They are delivered in a standardized protocol, HTTP (Hypertext Transfer Protocol, usually written in lower case letters), making HTML documents intelligible across hardware and software variations.

This information travels through web servers and web browsers. The communication initiates from a user request through a web browser.

The request is delivered to a web server in 'HTTP' format. The server then processes the request, which can be anything from a general search to a specific task, and returns the results in the same format.

The results are written in HTML, which is the language web pages are written in that supports high-speed travel between web pages.

HTML is also essential for displaying many of the interactive features on web pages, such as linking web pages to other objects, like images.

An important distinction when defining web servers is between hardware and software.

A web server is also a computer program (software) that performs the functions outlined above.

ROLE OF JAVA FOR CLIENT/SERVER ON WEB

- Client server models provide the essential mechanisms for working with the Internet.

- In client server models the web browsers run by millions of users are the clients. On the other side of the equation, is the web hosting systems that run at host sites and provide access to processes and data requested by the client.These hosting systems are the server.
- This definition is based on software programs, where the client is a program running on a remote machine that communicates with a server, a program running at a single site and providing responses to client requests, such as web pages or data.
- Java is a programming language that has been developed specifically for the distributed environment of the Internet.
- Java, like C++, is a language that is multi-paradigm and supports object-oriented programming (OOP), procedural programming, and data abstraction.

  - Object-oriented programming is increasingly being used in client server technology.
  - It refers to a programming language model that is organized by objects rather than

actions, data rather than logic. OOP identifies objects (sets of data) and defines their relationship to each other. These objects are then generalized into a class.

- ➢ Methods or sequences of logic are applied to classes of objects. Methods provide computational instructions and class object features provide the data that is acted upon.

- ➢ Users communicate with objects and objects with each other through specifically defined interfaces called messages.

- ➢ Java can create applications that are distributed between clients and servers in a network. Java source code (signaled by .java extension) is compiled (source code transformed into object code) into byte code (signaled by .class extension). A Java interpreter then executes this byte code format. Java interpreters and runtime environment run on Java Virtual Machines (JVMs).

- ➢ The portability and usability that characterizes Java stems from the fact that JVMs exist for most operating systems, from UNIX, to Windows, to Mac OS.

Java is one of the most well-suited languages for working on the World Wide Web and the client server .

To understand how Java is used in client server systems it becomes essential to understand the major characteristics of Java.

Syntactic of Java are similarity to C and C++ languages,

Java is also a robust programming language, which means it creates software that will identify and correct errors and handle abnormal conditions intelligently.

Another major characteristic of Java is that it is object oriented programming,

OOP is characterized by three properties also present in Java programming: inheritance, encapsulation, and polymorphism.

Multi threading is also an important characteristic of Java that increases interactive responsiveness and real time performance. Threading is the way a program splits or forks itself into two or more tasks running simultaneously.

This allows for thread based multi tasking. Multi threading creates the effect of multiple threads running in parallel on different machines simultaneously.

Socket-based Client Server Systems in Java

Java builds client server systems with sockets. Sockets are the endpoints of two-way communication between programs running in a network. They are software objects that connect applications to network protocols, so they become intelligible.

For example, in UNIX a program opens a socket that enables it to send and receive messages from the socket. This simplifies software development because programmers only have to change or specify the socket, while the operating system is left intact.

In client/server models, the server contains sockets bound to specific port numbers. Port numbers identify specific processes that are to be forwarded over a network, like the Internet, in the form of messages to the server. The server only has to monitor the socket and respond when a client requests a connection.

Once connections have been made and bound to port numbers, the two endpoints, client and server, can communicate through the socket.

Java sockets are client side sockets, known simply as sockets and server side sockets known as server sockets. Each belong to their own class within a Java package. The client initiates the socket, which contains a host name and port number, by requesting connections.

The server socket class in Java allows for servers to monitor requests for connections. As the socket communicates over the network, Java's server socket class assigns a one port number to each client in the server and creates a socket object or instance.

This server socket instance creates a connection and produces a thread or string of processes through which the server can communicate with the client through the socket.

Java web servers are built according to this model. TCP (Transmission Control Protocol) works in conjunction with IP (Internet Protocol) to send messages between computers over the Internet.

When communicating over the Internet, the client program and the server program each attach a socket to their end of the connection. Then the client and server can read from and write to the socket. Java provides operating system sockets that allow two or more processes to send and receive data, regardless of computer location.

Java's RMI System

The other method for using Java to build client server systems is RMI. RMI stands for Remote Method Invocation. By using Java language and functions, programmers write object oriented programming, so that objects that are distributed over a network can interact with each other.

RMI allows for client objects in JVMs to request services through a network from another computer (host or server). Client objects included with the request may call upon methods in the remote computer that change the results.

Methods are programmed procedures attributed to a class that are contained in each of its objects (or instances). It is a characteristic of object-oriented programming.

Classes and, therefore, objects can have more than one method and methods can be used for more than one object. Responses then run as if they were local (on the same computer.) This passing back and forth of objects and methods attached to objects is called 'object serializations'.

Simply put, RMI requests call upon the method of a remote object. As previously stated, it uses the same syntax it would locally. To make this intelligible to the servers or sites being called upon requires three layers: a client side stub program, a remote reference layer, and a transport connection layer. Each request travels down the layers of the client computer and up the layers of the server.

The client side stub program initiates the request. Stub programs are small sections of programs containing routines from larger programs. It is a substitute for programs that may take too long to load or are located remotely on a network. They are provided by the server at the client's request.

Stubs accept client requests and communicate requested procedures (through another program) to remote applications. They also return the results to the client or requesting program. These stubs mimic the program being called for service.

In Java, stub programs are also referred to as 'proxy'. Remote reference layers manage reference variables for remote objects, using the transport layer's TCP (Transmission Control Protocol) connection to the server. Reference variables contain class data and therefore include methods.

The transport layer protects and maintains end to end communication through a network. The most used transport layer is TCP. After the client requests pass through the transport layer, they pass through another remote reference layer before requesting implementation of the request by the server from a skeleton. These skeletons are written in high level IDL (Interface Definition Language).

The server receives the request and sends the response along the same channels, but in the other direction.

- A web service is an internet-based interface that makes the "services" of an application available to other applications running on different platforms.

- A microservice is a small, independent application that performs a highly focused service within a larger application.

- Each microservice is a modular, unique process that can be developed and deployed independently.

The main difference between the two is that a web service requires a network to function. As the name implies, web services operate on the World Wide Web, providing a service to other applications running on the internet.

On the other hand, microservices operate within a single application, providing services within that app's framework.

Some basic definitions of microservices and web services are:

- Microservice: A small, autonomous application that performs a specific service for a larger application architecture.
- Web service: A strategy to make the services of one application available to other applications via a web interface.

- Microservices application architecture: A modular, services-oriented application architecture comprised of loosely connected, independently running microservices. These microservices usually offer APIs (Application Programming Interfaces), so other microservices and apps can integrate with them.
- Web services application architecture: A modular, services-oriented application architecture where the applications that comprise the architecture connect via web services.

- Developers can use web services to connect microservices, monolithic applications, and more to form a larger application.

What is an API?

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications.

This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

How do APIs work?

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau's weather database is the server, and the mobile app is the client.

There are four different ways that APIs can work depending on when and why they were created.

SOAP APIs

These APIs use Simple Object Access Protocol. Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.

RPC APIs

These APIs are called Remote Procedure Calls. The client completes a function (or procedure) on the server, and the server sends the output back to the client.

Websocket APIs

Websocket API is another modern web API development that uses JSON objects(JavaScript **Object** Notation (**JSON**) is a standard text-based format for representing structured data based on JavaScript **object** syntax.)to pass data.

A WebSocket API supports two-way communication between client apps and the server.

The server can send callback messages to connected clients, making it more efficient than REST API.

REST APIs

These are the most popular and flexible APIs found on the web today. The client sends requests to the server as data. The server uses this client input to start internal functions and returns output data back to the client.

API GATEWAYS

Application programming interfaces (APIs) are the most common way to connect users, applications, and services to each other in a modern IT environment. An *API gateway* is a component of the app-delivery infrastructure

For microservices-based applications, an API gateway acts as a single point of entry into the system. It sits in front of the microservices and simplifies both the client implementations and the microservices app by decoupling the complexity of an app from its clients.

In a microservices architecture, the API gateway is responsible for request routing, composition, and policy enforcement. It handles some requests by simply routing them to the appropriate backend service, and handles others by invoking multiple backend services and aggregating the results.

An API gateway might provide other capabilities for microservices such as authentication, authorization, monitoring, load balancing, and response handling, offloading implementation of non-functional requirements to the infrastructure layer and helping developers to focus on core business logic, speeding up app releases.

Web Services Authentication

When users send a request for a web service, they're authenticated according to the credential type that is configured for Business Central Server. To access a web service, users must provide valid credentials for the credential type being used.

Common API authentication methods

While there are dozens of open and proprietary API authentication methods available, there are four common methods most administrators will see and interact with over the course of their professional career.

1. HTTP basic authentication

If a simple form of HTTP authentication is all an app or service requires, HTTP basic authentication might be a good fit. It uses a locally derived username and password and relies on Base64 encoding. Authentication uses the HTTP header, making it easy to integrate. Because this method uses shared credentials, however, it's important to rotate passwords on a regular basis.

2. API access tokens

API keys have unique identifiers for each user and for every time they attempt to authenticate. Access tokens are suitable for applications where many users require access. Access tokens are secure and easy to work with from an end-user perspective.

3. OAuth with OpenID

Despite having *auth* in the name, OAuth is not an authentication mechanism. Instead, it provides authorization services to determine which users have access to various corporate resources.

OAuth is used alongside OpenID, an authentication mechanism. Using OAuth and OpenID together provides authentication and authorization. With OAuth 2.0, OpenID can authenticate users and devices using a third-party authentication system. This combination is considered one of the more secure authentication/authorization options available today.

4. SAML federated identity

Security Assertion Markup Language (SAML) is another tokenlike authentication method often used in environments that have federated single sign-on (SSO) implemented. This XML-derived open standard framework helps seamlessly authenticate users through the organization's respective identity provider. For larger organizations working to consolidate the number of authentication mechanisms within the company, the use of SAML and federated SSO is a great fit.

Choosing the right API authentication mechanism

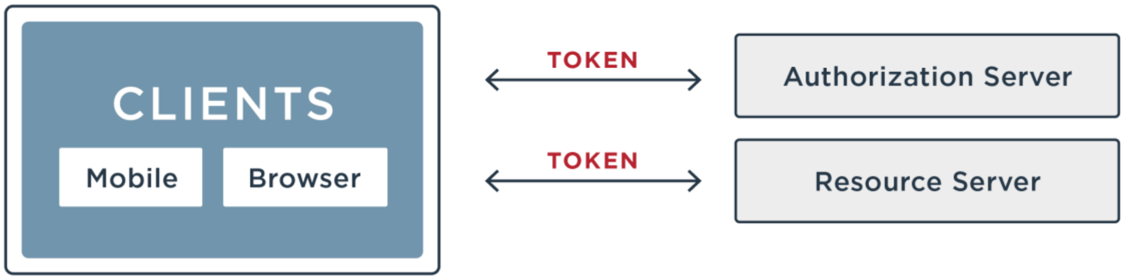
When choosing the type of API authentication mechanism to implement, there are three factors to consider:

1. Understand what API authentication methods are available in your given API framework.
2. Choose the API authentication that provides the proper level of security without being overly complex.
3. From an ongoing administration and management perspective, choose the API authentication method that fits into your existing, corporatewide authentication infrastructure.

What is Token-based Authentication for Web APIs?

Token-based authentication for web APIs is the process of authenticating users or processes for applications in the cloud.

The user's application sends a request to the authentication service, which confirms the user's identity and issues a token.
The user is then able to access the application.



- **What is a Token-based Authentication?**
- Token-based authentication is a two-step authentication strategy to enhance the security mechanism for users to access a network. The users once register their credentials, receive a unique encrypted token that is valid for a specified session time. During this session, users can directly access the website or application without login requirements. It enhances the user experience by saving time and security by adding a layer to the password system.
- A token is stateless as it does not save information about the user in the database. This system is based on cryptography where once the session is complete the token gets destroyed. So, it gets the advantage against hackers to access resources using passwords.
- The most friendly example of the token is OTP (One Time password) which is used to verify the identity of the right user to get network entry and is valid for 30-60 seconds. During the session time, the token gets stored in the organization's database and vanishes when the session expired.

How does Token-based Authentication work?

4 steps

1. Request: The user intends to enter the service with login credentials on the application or the website interface. The credentials involve a username, password, smartcard, or biometrics

2. Verification: The login information from the client-server is sent to the authentication server for verification of valid users trying to enter the restricted resource. If the credentials pass the verification the server generates a secret digital key to the user via HTTP in the form of a code. The token is sent in a JWT open standard format which includes-

- Header: It specifies the type of token and the signing algorithm.
- Payload: It contains information about the user and other data

- Signature: It verifies the authenticity of the user and the messages transmitted.

2. Token validation: The user receives the token code and enters it into the resource server to grant access to the network. The access token has a validity of 30-60 seconds and if the user fails to apply it can request the Refresh token from the authentication server. There's a limit on the number of attempts a user can make to get access. This prevents brute force attacks that are based on trial and error methods.

3. Storage: Once the resource server validated the token and grants access to the user, it stores the token in a database for the session time you define. The session time is different for every website or app. For example, Bank applications have the shortest session time of about a few minutes only.

What is an API key?". Application programming interfaces (APIs) allow software programs to interact, share data, and integrate their functionalities. API keys are one such security measure — they act like an ID card for the client making an API request, helping APIs assign the proper access permissions and track how their data are being used.

An API key is an identifier assigned to an API client, used to authenticate an application calling the API. It is typically a unique alphanumeric string included in the API call, which the API receives and validates. Many APIs use keys to keep track of usage and identify invalid or malicious requests.

API Keys: API Keys came into picture due to slow speed and highly vulnerable nature of HTTP Basic Authentication. API Key is the code that is assigned to the user upon API Registration or Account Creation. API Keys are generated using the specific set of rules laid down by the authorities involved in API Development. This piece of code is required to pass whenever the entity (Developer, user or a specific program) makes a call to the API. Despite easy usage and fast speed, they are highly insecure.

Service Mesh

is a mechanism for managing communications between the various individual services that make up modern applications in a microservice-based system.

When a service mesh is applied, all inter-service communication is routed through proxies, which can be used to implement networking features such as encryption and load balancing. The service mesh decouples the network logic from the application or business logic of each microservice so that it can be implemented and managed consistently across the whole system.

The dedicated infrastructure layer of the service mesh is something of a complimentary piece of technology to an API gateway. But a service mesh only handles communication between services that make up a system, while an API gateway decouples the underlying system from the API that is exposed to clients (which can be other systems within the organization or external clients).

The difference between API gateway vs service mesh is sometimes characterized as north-south (API gateway) versus east-west (service mesh) communication, but that's not strictly accurate.

While the service mesh pattern was designed to handle network connectivity between microservices, it can also be applied to other architectures (monolithic, mini-services, serverless) wherever there are multiple services communicating across a network.

service mesh is a dedicated infrastructure layer built into an application that controls service-to-service communication in a microservices architecture. It controls the delivery of service requests to other services, performs load balancing, encrypts data, and discovers other services.

Although you can code the logic that governs communication directly into the microservices, a service mesh abstracts that logic into a parallel layer of infrastructure using a proxy called a sidecar, which runs alongside each service.

Sidecar proxies make up a service mesh's data plane, which manages the exchange of data between services. Management processes make up the control plane, which coordinates the proxies' behavior. The control plane also provides an API so operators can easily manage traffic control, network resiliency, security and authentication, and custom telemetry data for each service.

From a secure communications standpoint, service meshes are useful for managing the secure TLS (mTLS) connections between services.

Because service meshes manage the communication layer, they allow developers to focus on adding business value with each service they build, rather than worrying about how each service communicates with all other services.

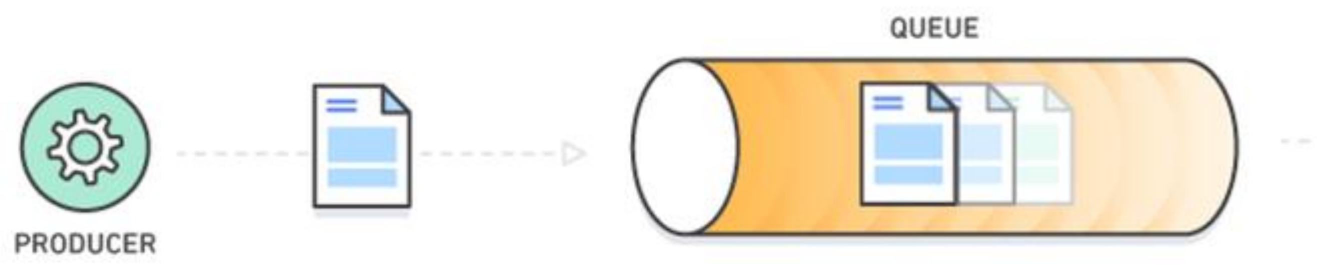
What is a Message Queue?

A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer. Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

Message queues provide communication and coordination for these distributed applications. Message queues can significantly simplify coding of decoupled applications, while improving performance, reliability and scalability.

Message queues allow different parts of a system to communicate and process operations asynchronously. A message queue provides a lightweight buffer which temporarily stores messages, and endpoints that allow software compone

The messages are usually small, and can be things like requests, replies, error messages, or just plain information. To send a message, a component called a producer adds a message to the queue. The message is stored on the queue until another component called a consumer retrieves the message and does something with it.



Many producers and consumers can use the queue, but each message is processed only once, by a single consumer. For this reason, this messaging pattern is often called one-to-one, or point-to-point, communications.

Software as a service(SaaS)

It is the idea that clients access and use software using a thin client or a web browser. The software itself is running on computers accessible through the Internet ("in the cloud"). Clients usually pay a fee to get access to the software.

The benefit of this approach is that clients do not need to manage the software installation, and that they also do not need expensive hardware.

Software providers do not need to use traditional methods of supply, such as distributors, to sell software to clients. One of the problems is that the data is also on the site of the company running the software - that is "in the cloud". Getting the data may be difficult, if the software vendor goes bankrupt.

Richard Stallman of the Free Software Foundation considers SaaS as a violation of the principles of free software, because users have no access to the program that runs the software, and hence do not know what it does, and cannot change it.

SaaS is a licensing model in which access to software is provided on a subscription basis, where the software is located on external servers rather than on servers located in-house.

Software as a Service is commonly accessed through a web browser, with users logging into the system using a username and password.

Instead of each user having to install the software on their computer, the user can access the program via the internet.

- Software as a Service (SaaS) is a software licensing model, which allows access to software on a subscription basis using external servers.
- SaaS allows each user to access programs via the Internet, instead of having to install the software on the user's computer.
- SaaS has many business applications, including file sharing, email, calendars, customer retention management, and human resources.
- SaaS is easy to implement, easy to update and debug, and can be less expensive than purchasing multiple software licenses for multiple computers.
- Drawbacks to the adoption of SaaS include data security, speed of delivery, and lack of control.

SaaS Advantages

- Accessible from anywhere
- Cost effective
- Easy to implement, update, and debug
- Easy to scale

SaaS Disadvantages

- Increased security risks
- Slower speed
- Loss of control
- Lack of customization

Examples of SaaS

- Google Docs
- Dropbox

WebSocket

It is a persistent bi-directional communication channel between a client (e.g. a browser) and a backend service. WebSocket: WebSocket is bidirectional, a full-duplex protocol that is used in the same scenario of client-server communication, unlike HTTP it starts from ws:// or wss://.

It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by either party (client or server). After closing the connection by either of the client and server, the connection is terminated from both ends.

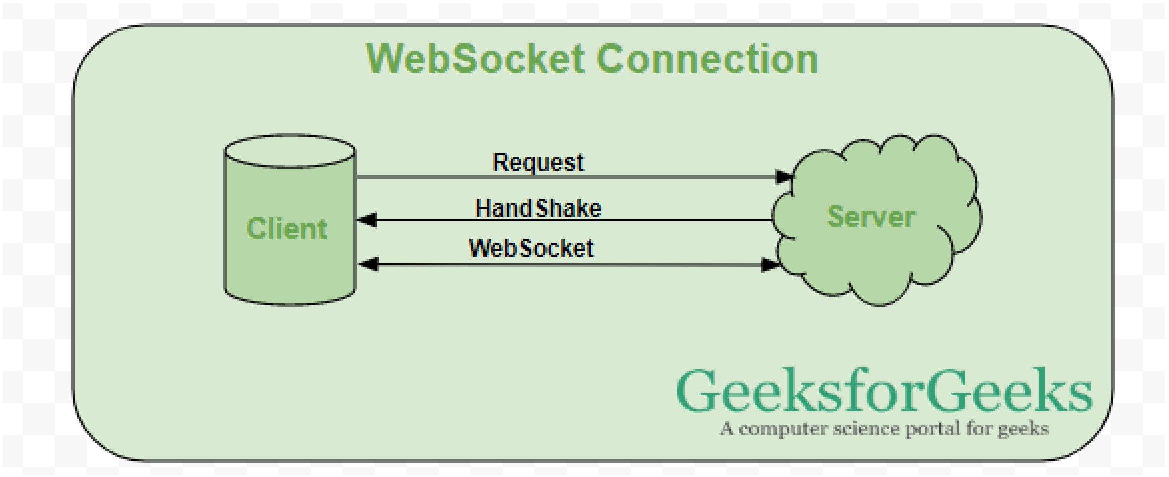
In contrast with HTTP request/response connections, websockets can transport any number of protocols and provide server-to-client message delivery without polling.

WebSockets establish TCP-style connections in a browser-compatible fashion using HTTP during initial setup.

Messages over websockets can be provided in any protocol, freeing the application from the sometimes unnecessary overhead of HTTP requests and responses (including headers, cookies, and other artifacts).

WebSockets are available on many platforms, including the most common browsers and mobile devices. They're often applied to solve problems of millisecond-accurate state synchronization and publish-subscribe messaging, both of which leverage Websockets' provision for downstream pushes.

A challenge of operating a WebSocket-based system is the maintenance of a stateful gateway on the backend. A WebSocket is erected by making a common HTTP request to that server with an Upgrade header, which the server (after authentating and authorizing the client) should confirm in its response. After this, the connection remains established between that physical client-server pair; if at some point the service needs to be redeployed or the load redistributed, its WebSocket connections needs to be re-established.



When can a web socket be used:

- Real-time web application: Real-time web application uses a web socket to show the data at the client end, which is continuously being sent by the backend server. In WebSocket, data is continuously pushed/transmitted into the same connection which is already open, that is why WebSocket is faster and improves the application performance.
- Gaming application: In a Gaming application, you might focus on that, data is continuously received by the server, and without refreshing the UI, it will take effect on the screen, UI gets automatically refreshed without even establishing the new connection, so it is very helpful in a Gaming application.

- Chat application: Chat applications use WebSockets to establish the connection only once for exchange, publishing, and broadcasting the message among the subscribers. It reuses the same WebSocket connection, for sending and receiving the message and for one-to-one message transfer.

TRANSFORMATIONAL SYSTEM

The working environment of many of the organizations has been greatly affected by applications of Client/Server technologies. Following are the examples of technologies that have changed the trade processes.

(i) Electronic mail.
(ii) Client/server and user security.
(iii) Object oriented technology: CORBA.
(iv) Electronic data interchange.

Electronic Mail

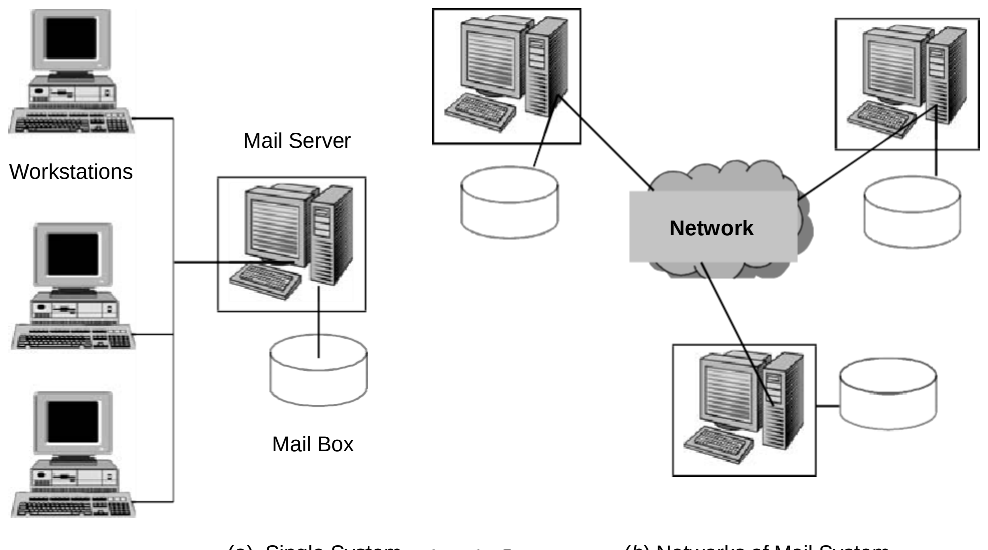
Electronic mail is already the most heavily used network application in the corporate world. It is a facility that allows users at workstations and terminals to compose and exchange messages.

However, the traditional e-mail is generally limited and inflexible. Intranet mail products provide standards, simple methods for attaching documents, sound, images, and other multimedia to mail messages.

The simplest form of electronic mail is the single system facility allowing the users of a shared computer system to exchange messages (see the Fig. 9.4 (a) given). The electronic mail facility is an application program available to any user logged onto the system. A user may invoke the electronic mail facility, prepare a message, and "send" it to any other user on the system.

The act of sending simply involves putting the message in the recipient's mailbox. Mailbox is actually an entity maintained by the file management system and is in nature of a file directory. Any incoming mail is simply stored as a file under that user's mailbox directory. One mailbox is associated with each user.

With a single system electronic mail facility, messages can only be exchanged among users of that particular system. For a distributed mail system, a number of mail handlers (mail servers) connect over a network facility (e.g., WAN or internet) and exchange mail. That is illustrated in Fig. 9.4(b) given below:



(a) Single System (b) Networks of Mail System

Fig. 9.4: Electronic Mail Configurations

Intranet mail system creates and manages an electronic mailing list that is an alias to multiple destinations.

Client/Server and User Security

However, the very characteristic that make Client/Server popular are also what make it the most vulnerable to breaches in security. It is precisely the distribution of services between client and server that open them up to damage, fraud, and misuse.

Security consideration must include the host systems, personal computers (PCs), Local Area Networks (LANs), Global Wide Area Networks (WANs), and users.

Because security investments don't produce immediately visible returns and Client/Server buyers sometimes don't educate themselves about security, this area of development is often overlooked until a problem occurs.

Desktops are the front-end system devices, the ones that deal most directly with user input.

They are also the least secure environments in Client/Server models.

Clients connect to servers and these connections, if left open or not secured, provide entry points for hackers and other intruders that may use data for nefarious purposes.

Aside from physical client security in the form of disk drive locks or diskless workstations that prohibit the loading of unauthorized software or viruses, accessibility to all files stored on a workstation operating system is the other gaping security hole in clients.

For example, the machine assumes that whoever turns on the computer is the owner of all the files stored on it. They even have access to configuration files. This could result in sabotage or the leaking of sensitive data.

The transmission of corrupted data may also occur on the level of the operating system, outside the realm of Client/Server application security, as data is transferred to different tiers of the architecture.

However, the primary culprits of breaching client security are not hackers or viruses, but the users themselves. The front-line of defense in client security is user identification and authentication.

The easiest way to gain illegal access to computers is to get users' login ID and passwords. Sometimes users pick short or easily guessed passwords or share their passwords with others.

Password management provides a security measurement for thisby requiring a minimum amount of characters to be used in passwords checking passwords for guess ability, and regularly asking users to change their passwords.

For example, more organizations are adopting policies of 'pass phrases' rather than passwords that are more complicated and harder to identify or guess. The system contains a scheme (minimalist, multi-paradigm programming language) that proactively detects and blocks spyware. Italso updates daily.

Gateways are nodes on a network that create entrances to other networks. It routes traffic from workstations to broader networks. Therefore, securing the gateways will prevent malware from ever reaching the client.

Using Client/Server computing some of the secure systems can also be implemented, having a goal to provide secure services to clients with maximum possible performance.

Emergency response vehicle can be quickly dispatched by dispatch operator to an incident and at the same time dealing can also be reported over the phone. This functionality can be provided 24 hours. It is now possible to duplicate all of the functionality of such an existing traditional design with the additional advantages of better performance, a Graphical User Interface (GUI), a single point of contact, higher reliability, and lower costs. With the help of a Client/Server-based system, the dispatch operator is empowered to oversee how staff and equipment are allocated to each incident.

The operator uses a GUI to dynamically alter vehicle selection and routing. Maps may be displayed that show the location of all incidents, emergency response centers, and vehicles.

Vehicles are tracked using Automatic Vehicle Locator (AVL) technology. Obstacles, such as traffic congestion, construction, and environmental damage (such as earthquakes) are shown on the map so the dispatcher can see potential problems at a glance. Workstation technology provides the dispatcher with a less stressful and more functional user interface.

The dispatcher can respond quickly to changes in the environment and communicate this information immediately to the vehicle operator. The system is remarkably fault-tolerant. If a single workstation is operating, the dispatcher can continue to send emergency vehicles to the incident.

Object-oriented Technology: CORBA

As object oriented technology becomes more prevalent in operating system design; Client/ Server designers have begun to embrace this approach. Here client and servers ship messages back and forth between objects. Object communication may rely on an underlying message or Remote Procedure Call (RPC) structure or be developed directly on top of object oriented capabilities in the operating system. Clients that need a service send a request to an object request broker, which acts as a directory of all the remote services available on the network. The broker calls the appropriate object and passes along any relevant data. Then the remote object services the request and replies to the broker, which returns the response to the client. There are several object oriented approach for standardization of these object mechanism are COM (Common Object Model), OLE (Object Linking and Embedding), Common Object Request Broker Architecture (CORBA, see the Fig. 9.5).

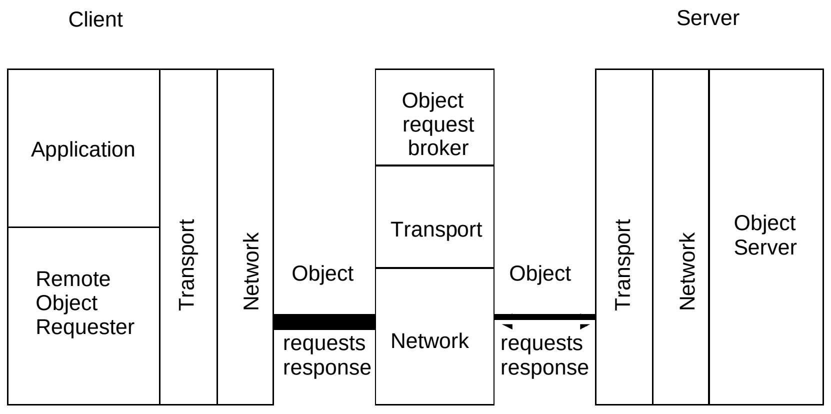


Fig. 9.5: Object Request Broker

An Overview of CORBA

The Object Management Group (OMG) was created in 1989. The OMG solicited input from all segments of the industry and eventually defined the CORBA standards.

CORBA specification has been implemented by numerous hardware and system software manufacturers, provides a rich and robust framework that operates across the heterogeneous computing platform. CORBA is a specification for an emerging technology known as distributed object management (DOM). DOM technology provides a higher level, object oriented interface on top of the basic distributed computing services.

At its most basic level, CORBA defines a standard framework from which an information system implementer or software developer can easily and quickly integrate network resident software modules and applications to create new, more powerful applications. It combines object technology with a Client/Server model to provide a uniform view of an enterprise computing system-everything on the network is an object. The highest level specification is referred to as the object management architecture (OMA), which addresses four architectural elements (ORB, CORBA services, CORBA facilities and CORBA domains are also defined as a part of specifications. The term CORBA is often used to refer to the object request broker itself, as well as to the entire OMG architecture.

The role of ORB is to route request among the other architectural components. CORBA services, CORBA facilities and CORBA domains are also defined as a part of specifications. The key to integrating application object is the specification of standard interfaces using the interface definition language (IDL). Once all applications and data have an IDL-compliant interface, communication is independent of physical location, platform type, networking protocol, and programming language. An information system is created by using CORBA to mediate the flow of control and information among these software objects.

CORBA an Introduction

Mechanism that allows various clients to share/call the object (applications) over a mixed network, more specifically CORBA is a process of moving objects over network providing cross platform for data transfer.

CORBA compliance provides a high degree of portability. Within CORBA, objects are an identifiable entity which provides one or more services to clients. CORBA manages the identity and location of the objects, transport the request to the target object, and confirms that the request is carried out. A client that needs a service sends a request to an object request broker (which acts as a directory) of all the remote services available on the network, illustrated in Fig. 9.6.

The broker calls the appropriate object and passes along any relevant data, and then the remote object services the request and replies to the broker, which returns to the client. The object communication may rely on an underlying message or RPC structure or be developed directly on top of object-oriented capability in the operating system.

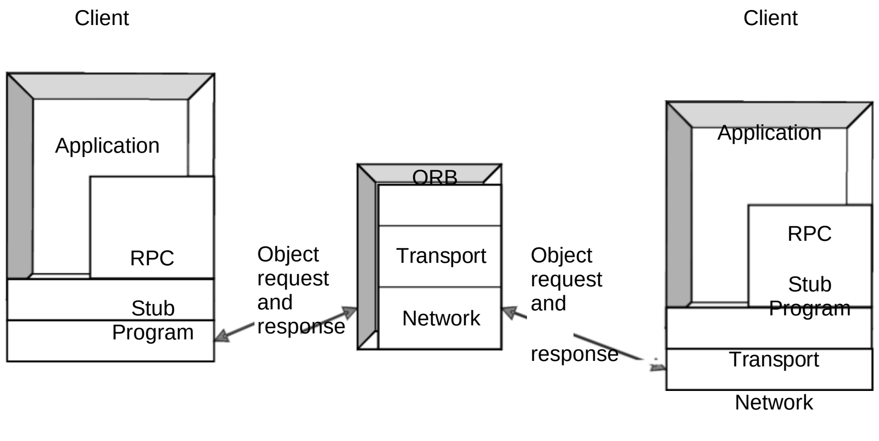


Fig. 9.6: *CORBA Remote Services*

CORBA Client and Servers

Like the Client/Server architecture, CORBA maintains the notions of Client and Server. In CORBA, a component can act as a client and as a server. Morever a component is considered a server if it contains CORBA objects whose services are accessible to other objects. Similarly, a component is considered a client if it accesses services from some other CORBA object.

Thus, a component can simultaneously provide and use various services, and so a component can be considered as a client or a server depending on the way.

More specifically, in CORBA application, any component that provides an implementation for an object is considered as a server, at least where that objects are concerned. If a component creates an object and provides others components with visibility to that object (or in other words, allows other components to obtain references to that object), that component acts as a server for that object; any requests made on that object by other components will be processed by the component that created the object. Being a CORBA server means that the component (the server) executes methods for a particular object on behalf of other components (the clients).

An application component can provide services to other application components while accessing services from other components. Here, the component is acting as a client of one component and as a server to the other components; see the Fig. 9.7 given below, illustrating those two components can simultaneously act as clients and servers to each other. In a CORBA application, the term client and server might depends on the context of the method being called and in which component that method's object resides. Although an application component can function as both a client and a server.

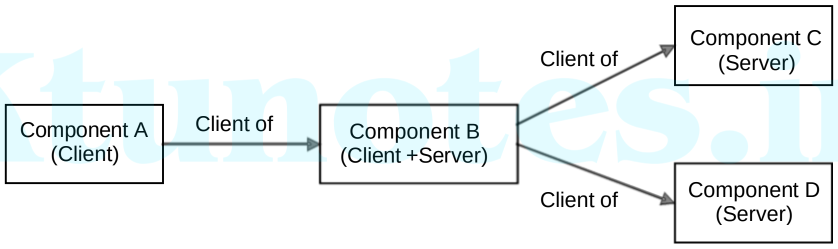


Fig. 9.7: *Acting as a Client and a Server*

CORBA Concepts

The basic idea is distributed computing, nowadays, most of the application are across the open environment, based on the connection of heterogeneous platforms. All modern business systems employ a network to connect a variety of computers, facilitating among applications. In the future, there will be continued evolution toward applications that exist as components across a network, which can be rapidly migrated and combined without significant effort. This is where CORBA shines, by providing unified access to applications, independent of the location of each application on network, also it provides:-

- Uniform access to services.
- Uniform discovery of resource/object.
- Uniform error handling methods.
- Uniform security policies.

These capabilities facilitate the integration and reuse of systems and system components, independent of network location and the details of underlying implementation technology. CORBA can be theoretically described based on the following three important concepts:

(i) Object-oriented model.

(ii) Open distributed computing environment.

(iii) Component integration and reuse.

(i) Object-oriented model: CORBA's object model is based on complete object approach in which a client sends a message to an object. The message identifiesan object, and one or more parameters are included. The first parameter defines the operation to be performed, although the specific method used is determined by the receiving object. The CORBA object model comprises of:

Objects: An encapsulated entity that provides services to a client.

Request: An action created by a client directed to a target object that includes information on the operation to be performed and zero or more actual parameters.

Object creation and destruction: Based on the state of request, objects are created or deleted.

Types: An identifiable entity defined over values.

Interfaces: The specification of operations that a client can request from an object.

Operations: An identifiable entity that defines what a client can request from an object.

Object implementation in CORBA can be constituted in two parts, illustrated in the Fig. 9.8, first one is construction part and second one is execution part.

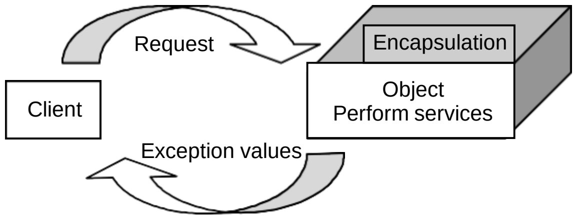


Fig. 9.8: Object Implementation

(ii) Open distributed computing environment: As we have earlier discussed that CORBA is based on a client server model of distributed computing. Within the Client/Server model, requests for services are made from one software component to another on a network. CORBA adds an additional dimension to this model by inserting a broker between the client and server components. The main objective of the broker is to reduce the complexity of implementing the interaction between the client and server. The broker plays two major roles. Primarily, it provides common services, including basic messaging and communication between client and server, directory services, meta-data description, security services, and location transparency. Secondly, it insulates the application from the specifics of the system configuration, such as hardware platform and operating system, network protocol, and implementation languages.

(iii) **Component integration and reuse:** The integration is the combination of two or more existing components. With a good integration techniques and tools, reuse can be achieved up to significant degree. Broker defines custom interfaces for each interaction between components. Each interface is defined just once and subsequent interactions are handled by the broker. With CORBA IDL, these interfaces can be defined in a standardized, platform independent fashion.

Electronic Data Interchange

Electronic data Interchange uses direct link between computers, even computers on different sites, to transmit data to eliminate data sent in printed form.

It is a controlled transfer of data between business and organizations via established security standards.One of the examples of EDI is shown in Fig. 9.9.

EDI is generally thought of as replacing standardized documents such as order forms, purchase orders, delivery notes, receiving advices and invoices in a standardized electronic message format.

EDI documents are electronically exchanged over communication networks which connect trading partners to one another. These documents are stored in user mailboxes on the networks' EDI server from where they can be downloaded/uploaded at the user is convenience from any one of the workstations. But it differs from electronic mail in that it transmits an actual structured transaction (with field such as the transition date, transaction amount, senders name and recipient name) as opposed to unstructured text message such as a letter.

The main purpose of EDI is cost reduction by eliminating paper document handling. Faster electronic document transmission further saves time and man power by avoidingthe need to re-key data. And the data arrival rate is much faster that mail.

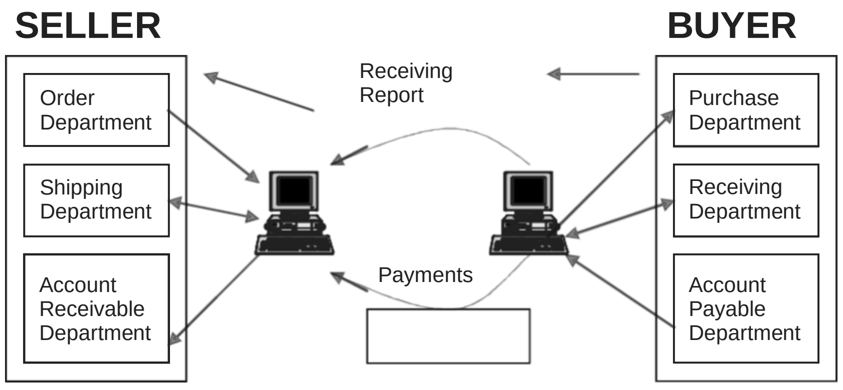


Fig. 9.9: EDI as an Example

Geographic Information Systems

Geographic information systems (GISs) provide the capability to view the topology of a landscape, including features such as roads, sewers, electrical cables, and mineral and soil content. GIS is a technology that has promised much and finally is beginning to deliver. As with the expert systems technology, GISs are truly useful when they integrate with the business process. From a technological perspective, GISs must operate on standard technologies, integrate with the organization SDE, and directly access the organizational databases.

Conceptually, GISs enable users to store virtually unlimited geographic information as a series of layers. Some layers, such as street layouts, compose the *base map*. Other layers, such as wetlands and subterranean water sources, are *thematic* layers that serve a specific, sometimes narrow purpose. A GIS user can custom design a printed map to fill a particular need by simply selecting the relevant layers. Selecting the street layer and the wetlands layer would produce a map of wetlands and their relationship to the streets. Selecting the subterranean water sources layer and the wetlands layer would show the wetlands superimposed on the features of the underlying aquifer.

Each line, curve, and symbol in a map is fixed in space by a series of numbers, called the *spatial data*. Spatial data describes the precise positioning of map objects in three-dimensional space.

Besides storing map objects such as street segments and wetland boundaries, GISs enable designers to specify attributes the users want to associate with any map object. Such attributes may be descriptive data, detailed measurements of any kind, dates, legal verbiage, or other comments. When viewing a map on-screen, the user can click any map object, and a data-entry window will open to display the attributes associated with that object. Attribute information is usually stored in RDBMS tables, and each map layer can draw attributes from multiple tables.

GIS applications are naturals for client/server technology. Powerful workstations manage the mapping. Connectivity enables shared access to the layers maintained by various departments. The GIS database is related to attributes stored in other databases that provide considerably more value in combination. For example, combining the voters list with the street maps allows polling booths to be located with easy access for all and ensures that no natural or artificial barriers are blocking the way.

Electronic Document Management—Multimedia

The concepts of electronic image management relate to the manipulation of information contained in forms, blueprints, x-rays, microfilm, fingerprints, photographs, and typewritten or handwritten notes. Electronic document management adds the capability to manipulate information from other media, such as audio and video. In addition, the "folder" device gives the user an intuitive interface to information equivalent to, but more flexible than, a filing cabinet. The information is always available on the desktop. Several users can use the folder simultaneously. Folders are always refiled in the correct location. Billions of these documents exist and are used daily in the process of providing government services. Consider that the Los Angeles County municipal hospitals alone have 5 billion pieces of paper, x-rays, scans, photographs, audio reports, and videos (and so on) filed to maintain patient records. Currently, the cost of this technology is prohibitively high for most organizations, but these systems will come down in price as all computer components do.

Figure 10.5 illustrates the range of information sources that can be manipulated digitally. To make efficient and effective use of this information, the means must exist for rapid filing, retrieval, and sharing of this information among all persons. This is the principle of making information available only to those with a "need and a right to know."

Electronic mail can be delivered routinely in seconds anywhere in the United States. Consumers can have direct access to suppliers. Goods can be ordered and paid for electronically. A retired engineer in Northern California can teach algebra to disadvantaged children in Compton, located in the southern part of the state. A parent can deliver office work to an employer in downtown Los Angeles while he cares for children at home. Library and museum materials can be explored at the users' own pace, with their personal interests in mind, to tap into a rich assortment of interactive, graphical how-to lessons. The community library can provide the conduit to government services: taking drivers' photographs and producing drivers' licenses on-site, producing birth certificates, or searching the titles on properties. Lawyers can file case data, review their calendars, or locate their clients' criminal records all from the law office and under the safeguards provided by electronic passwords, user auditing, and caller ID validation.

Each of these functions can be conveniently provided to citizens and consumers without them having to travel to an office location. The cost savings and environmental impact of this convenience are important considerations in today's society. Businesses no longer need to rent or buy expensive office space close to clients and suppliers. Individuals can live where they want and commute electronically to work. It is easy to imagine how the provision of these services in such a convenient manner can generate significant revenues that more than offset the cost of providing the service.

High-speed communications networks can provide the capability to distribute information other than voice conversations throughout a county, state, or country. With the advent of fiber-optic cabling, the capacity for information distribution to a location, office, library, or home is essentially infinite. As this technology become readily available, we will be able to consider where best to store and use information without concern for transmission time or quality. This is particularly true within a small geographical area, such as a county where the "right of way" is owned and private fiber-optic networks can be installed. High-speed networks in conjunction with new standards for data integrity ensure that information can be stored throughout the network and properly accessed from any point in the network.

Electronic documents can be transmitted and received just like any other digital information. The same networks and personal computers can send and receive. The major stumbling blocks to widespread sharing of electronic documents have been the incompatible formats in which various vendors store and distribute the digital image and the lack of a central repository of indexes to the documents. These indexes should describe the document content to enable users to select the correct folder and document.

Most information used by business and government today is contained in formats that are not manipulatable through traditional data-processing techniques. This is consistent with the "need and a right to know," mentioned earlier. Los Angeles County, for example, decided to overcome these problems through the definition of standards that must be adhered to by all products acquired for county projects.

Full-Text Retrieval

An area of explosive growth, coincident with the availability of high-powered workstations and RISC servers, is *full-text retrieval*. Originally a technology used by the military to scan covert transmissions and media communications, full-text retrieval is now a mainstream technology. Vendors such as Fulcrum and PLS have packaged their technology to be used in more traditional business applications. Northern Telecom bundles a product, Helmsman, with all its switch documentation to facilitate document access. All major news services provide an electronic feed of their information. This information is continuously scanned by reporters, businesses, and government offices to identify significant events or locate trends. Dow Jones provides their news retrieval system with access to 10**12bytes (that's three billion pages) of textual data. Many criteria searches can be run against all this text in a few seconds. Figure 10.6 illustrates the flow of information in a full-text retrieval application.

The major hardware and software technologies that have made this technology production viable are Optical Character Recognition (OCR), ICR, optical storage, powerful workstations, large D-RAM, software algorithms, and high-resolution monitors. OCR and ICR technologies convert the paper documents to text files. Companies such as Colera provide software to convert typewritten documents directly into WordPerfect format. Recent improvements in these algorithms provide support for most major fonts. Improvements in handwriting recognition promise to enable users to enter data from handwritten documents as well. Colera provides a fax link that enables documents to be entered by way of OCR as they are received from a fax. Mitek provides high-speed ICR engines to be used with document workflow applications. Embedded diagrams are maintained in image format.

Full-text indexing of documents is a CPU-intensive function, and the availability of low-cost, high-powered workstations has made the technology viable. (See Figure 10.7.) PC products such as Lotus Magellan enable the casual user to create full-text indexes of all their files. Viewers and launchers within the products enable users to access these files in their native format and manipulate them using a data editor of choice. With the advent of Object Linking and Embedding (OLE 2.x) and CORBA-based object solutions such as DOE, full-text access will become much more common to support capture and inclusion of source material. For high-performance retrievals, the indexes must support boolean search requests. The availability of large and low-cost D-RAM provides the necessary environment. High-resolution monitors are necessary as we move to a multiwindowed environment using facilities such as OLE and DOE. Extensive use of these facilities will not be viable without the appropriate resolution, because eyestrain will discourage use. We recommend Super VGA, a resolution of 1024 by 768, as a minimum for this type of multiwindowed work.

What are the Client and Servers in the Computer Network?

Clients are computer hardware or server software that makes requests for resources and services that a server makes available. Clients are often referred to as "service requesters". Thick, Thin, or Hybrid client computing are the three categories.

- **Thick Client:** A client that offers extensive functionality, does the majority of data processing on its own, and depends on the server only a little.
- **Thin Client:** An application server handles the majority of the necessary data processing for a thin-client server, which is a lightweight computer that heavily relies on the resources of the host computer.
- **Hybrid Client:** A hybrid client combines the elements of a thin client and a thick client. It may do local processing but must rely on the server to keep persistent data.

A device or computer program that serves as a hub for other components or programs is known as a server. A server is any computerized system that a client may access or utilize to share resources and distribute tasks. Typical servers include the following:

- Application Server
- Computing Server
- Database Server
- Web server

Client-Server Model

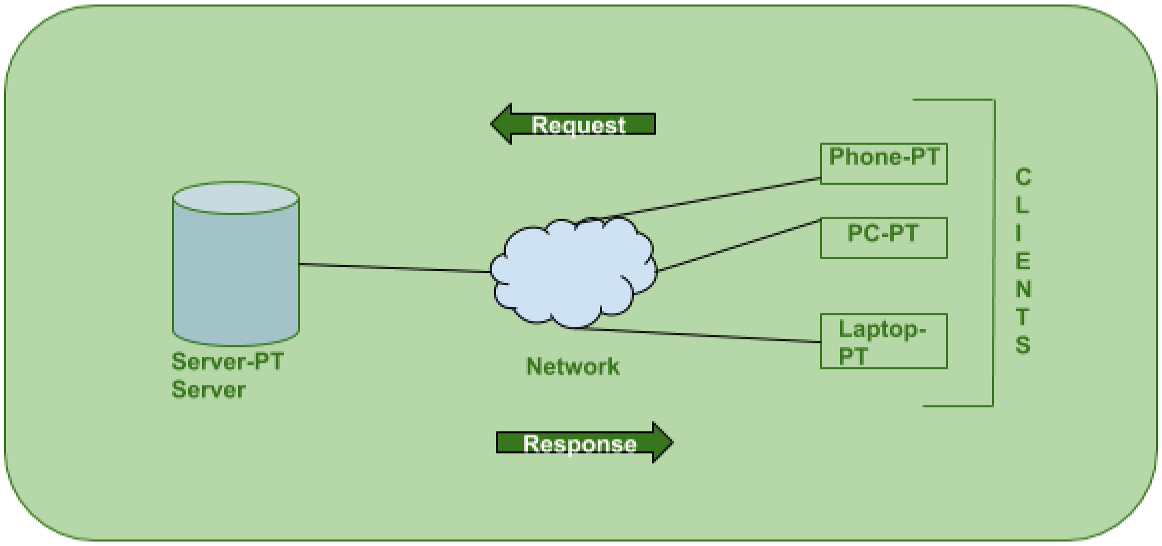
The Client-server model is a distributed application structure that partitions task or workload between the providers of a resource or service, called servers, and service requesters called clients. In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and deliver the data packets requested back to the client. Clients do not share any of their resources. Examples of Client-Server Model are Email, World Wide Web, etc.

How the Client-Server Model works ?

In this article we are going to take a dive into the **Client-Server** model and have a look at how the **Internet** works via, web browsers. This article will help us in having a solid foundation of the WEB and help in working with WEB technologies with ease.

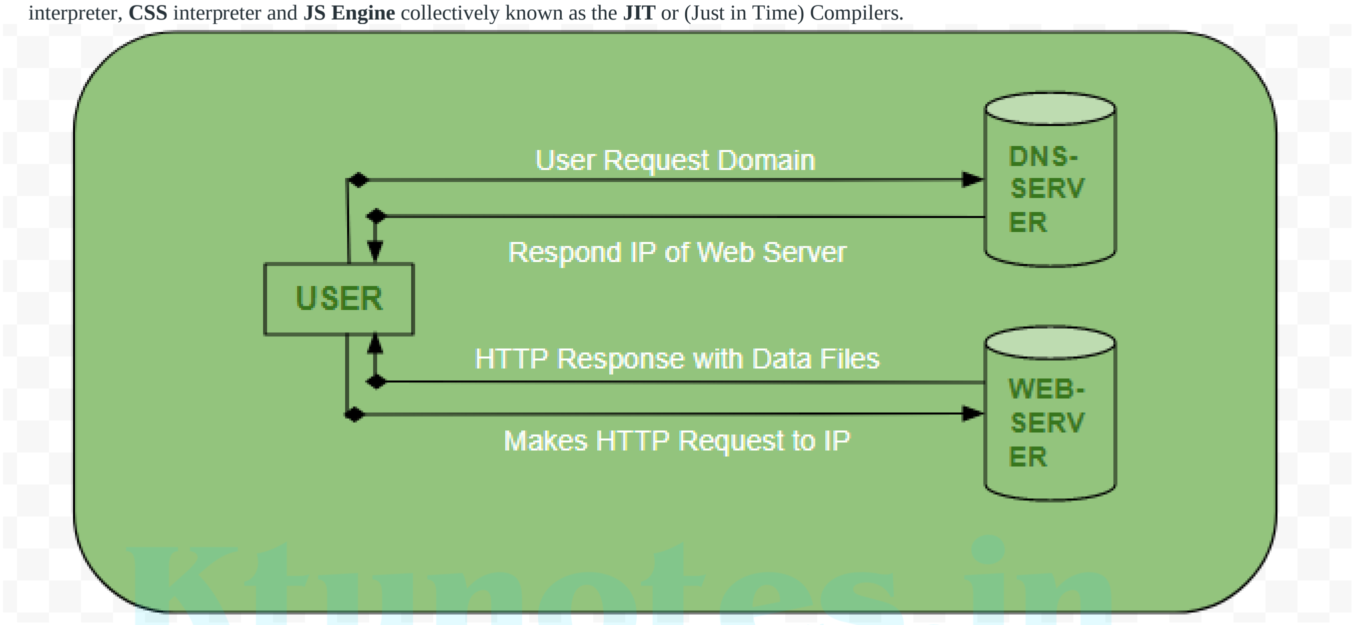
- **Client:** When we talk the word **Client**, it mean to talk of a person or an organization using a particular service. Similarly in the digital world a **Client** is a computer (**Host**) i.e. capable of receiving information or using a particular service from the service providers (**Servers**).
- **Servers:** Similarly, when we talk the word **Servers**, It mean a person or medium that serves something. Similarly in this digital world a **Server** is a remote computer which provides information (data) or access to particular services.

So, its basically the **Client** requesting something and the **Server** serving it as long as its present in the database.



How the browser interacts with the servers ?
There are few steps to follow to interacts with the servers a client.

- User enters the **URL**(Uniform Resource Locator) of the website or file. The Browser then requests the **DNS**(DOMAIN NAME SYSTEM) Server.
- **DNS Server** lookup for the address of the **WEB Server**.
- **DNS Server** responds with the **IP address** of the **WEB Server**.
- Browser sends over an **HTTP/HTTPS** request to **WEB Server's IP** (provided by **DNS server**).
- Server sends over the necessary files of the website.
- Browser then renders the files and the website is displayed. This rendering is done with the help of **DOM** (Document Object Model) interpreter, **CSS** interpreter and **JS Engine** collectively known as the **JIT** or (Just in Time) Compilers.

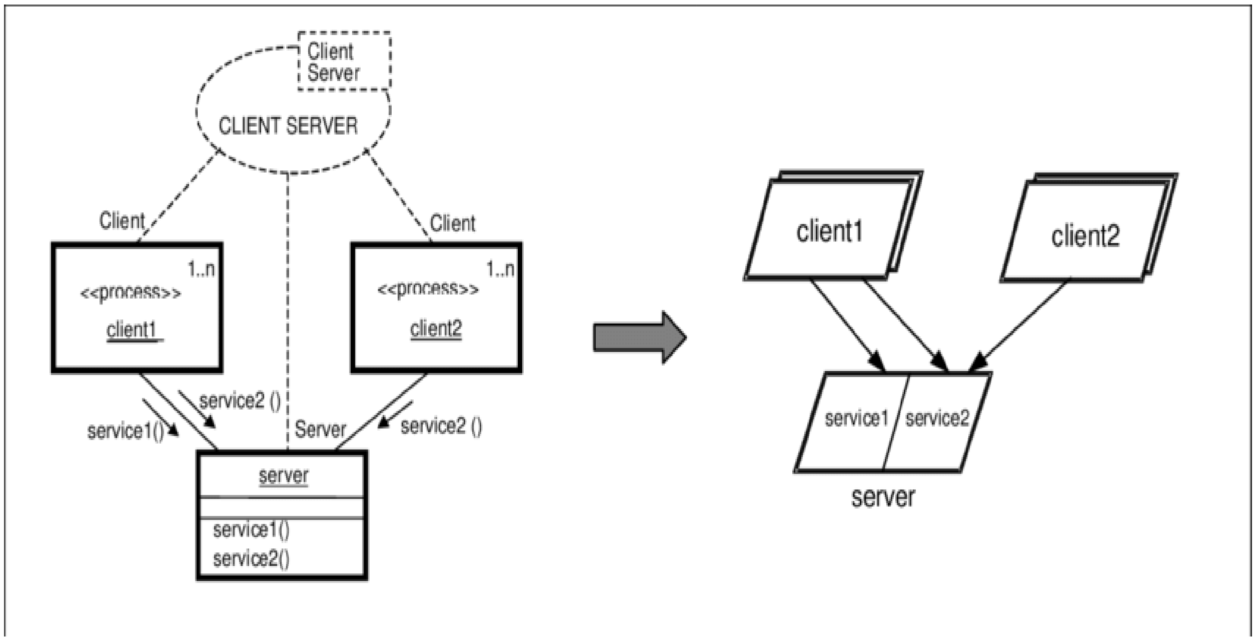


Advantages of Client-Server model:

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model:

- Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.
- Server are prone to Denial of Service (DOS) attacks.
- Data packets may be spoofed or modified during transmission.
- Phishing or capturing login credentials or other useful information of the user are common and MITM(Man in the Middle) attacks are
- common
-
-
-
- Architecture of client server transformation system
- .


Client
Server
CLIENT SERVER
Client
Client
1..n
<<process>>
client1
1..n
<<process>>
client2
service2 ()
service1()
Server
service2 ()
server
service1()
service2()
client1
client2
service1
service2
server