Client Server Application Programming

Week 1: Course overview(Introduction to client/server, Merit and demerits of client server, Types of servers etc.)

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com

jose@kumiuniversity.ac.ug

Agenda

1. Course Overview
2. Introduction to client/server
3. Merit and demerits of client server,
4. Types of servers,
5. Client server architecture, the fat servers and fat clients, ORB,
6. Remote procedure call (RPC) and Interposes communication)
7. A guide to good Client-Server Application Design

Course Overview

1. Course Description
2. Course Objectives
3. Course requirements
4. Course length
5. Text Books and References

Course Overview-Course Description

This course is designed to help Learners develop a basic understanding of how the Client/Server environment works. It creates an environment upon which Learners can develop a basic understanding of how to design a Client Server application.

Topics such as client/server, data warehouse, and data distribution that are reshaping the roles of both application development and information management are discussed.

In this course the Learner learns how to implement effective information management processes to manage data in client/server, distributed, and data warehouse environments as an enterprise resource.

Course Overview-Course Description+

Management, technical, and organizational issues required to implement information resource management (IRM) effectively in the distributed and decentralized computing environment are addressed.

Potential pitfalls facing IRM in the client/server environment are identified along with guidance on how to avoid them.

Course Overview-Course Objectives

The objectives of this course is to:-

- introduce learners to the strategic potential of distributed computing systems for business processes.
- provide an understanding of a framework for classifying distributed computing architectures and distributed applications.
- map out information systems architecture and assess the fit between existing and needed architectures.
- classify and evaluate the numerous flavors of middleware in order to make decisions about middleware acquisition.

Course Overview-Course Objectives+

The objectives of this course is to:-

- dissect the role of the transaction processing, object-oriented, and Internet-based technologies in distributed enterprise computing and make decisions about how and when to apply them.
- Walk learners through the factors that contributes to the performance of client/server systems and incorporates this understanding in the design of client/server systems.
- discuss the many issues including; tradeoffs, and decision points in developing, integration, and managing distributed applications.

Course Overview-Requirements

Creating a client-server application requires a strong understanding of networking concepts, programming languages, and communication protocols.

It can be a complex process, but the end result is a powerful application that can be used in a variety of contexts, from simple file sharing to complex business applications.

Course Overview - Course Length

As stipulated in the syllabus, this course is designed to run for 13 weeks. For details about the break down of the content, kindly view the syllabus.

Text Books

You can use the following books,

Java Network Programming (4th ed.). Sparks, R. (2015). O'Reilly Media, Inc.

Network Programming and Distributed Computing, David R, Michael R (2002), Publisher: Addison Wesley, ISBN: 0-201-71037-4

Java Network Programming 3rd, Beginning Android 4, O'Reilly, H. (2012) Grant Allen, Apress, 978-1-4302-3984

Object-Oriented Client/Server Internet Environments. Umar, A. Prentice Hall

ISBN: 0133755444

Introduction to Client-Server Application programming

Client-server application programming is the process of creating software applications that communicate over a network using the client-server model.

In this model, one program acts as the client and another program acts as the server, and they communicate with each other to fulfill a particular task

Introduction to Client-Server Application programming+

The word Client-server is made of two key words; Client and Server. Which means we are talking about two types of applications working together to achieve a common goal.

Client Application sends request to the server application which inturn serves it with the requested recourses.

While **Server Application** is an application that keeps on receiving requests from a client application and provide the requested recourse.

Client Application

The Client (typically)

- An application that runs on a personal computer.
- It has an extensive and appealing user interface, but it has no data that it can manipulate or present to the user.
- The client application 'requests' connects to the server application to obtain that data.
- Once the data has been obtained, it is presented to the user in a nice format.
- A client usually gets the data from a server at a time, and servers interacts with one or more users simultaneously.

Server Application

The Server (typically)

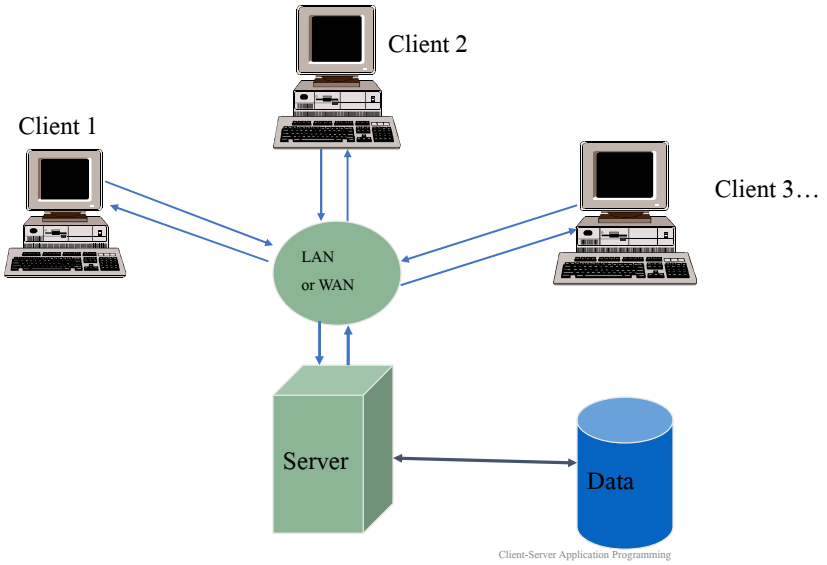
- An application that runs on a large computer (large meaning either fast or large storage space or both).
- Typically, a server application has control over large amounts of data, and can access that data fast and efficiently e.g. Bank databases
- It can also handle requests of many clients (more or less) simultaneously.

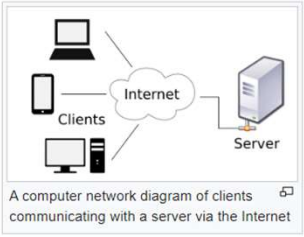
Client-Server System

One system made up of (the server) provides services for another (the client)

- The two systems usually reside on different computers, perhaps this could be with different offices, cities, continents and etc
- **Examples include;** mail , web, and database servers
- It is a model for developing network running applications for example in banks, security, academic etc places
- It is service-oriented, and employs a request-response protocol

Client-Server Environment/process





https://en.wikipedia.org/wiki/Client%E2%80%93server_model

Client-Server Process

What happens during Client-Server Application Servicing?

1. First, client machines initiates a request for a service(s)
2. In return a server, processes the requested services

Note.

Server or client may be running on different machines via the remote host machine (distributed systems)

Server goes to sleep and usually waits for requests from clients

A client opens the communication channel using IP address of the remote host and the port address of the specific server program running on the host. This is called Active open

Client-Server Process+

Request responses may be repeated several times and this process is referred to as finite

The client closes the communication channel with an Active close

A server program opens its door for incoming request from clients but never initiates a service unless explicitly requested. This is referred to as Passive open

A server program is infinite – runs unless a problem occur

Important Note

- Client and server programs need to be able to establish connections to each other over the network.
- Client and server applications connect through a "Transmission Control Protocol/Internet Protocol" -- or TCP/IP -- standards.
- The server is linked or bound to a specific IP address on the network that is reachable by the client.
- The server also has a TCP port number that identifies the type of service it provides.
- Like an electrical socket, the server and client programs first have to connect and data begins to flow between them.

What manages Client-Server Application process?

The Client–Server process are supported and managed by;

- Servers: Microsoft Windows Server distributions such as 2003, 2oo8, 2012 etc, Unix, Linux
- Clients: Microsoft Windows operating distributions such as 7,8,10,11, etc

Client Operating systems are further enhance by using some other client applications such as web browsers and other client-side applications.

Merits of Client-Server Applications

1. Simple, convenient and most widely adopted programming model for distributed systems
2. Distributed systems; clients and servers applications which usually run on different machines whereby one entity provides the service to another
3. Systems are programmed in a way that message acknowledgements are delivered.
4. The client sends a request in form of a message to the server and waits for reply
5. Processing and data are localized on the server – this reduces the network traffic, response time and bandwidth requirements

Demerits of client-server Applications

1. The server becomes a bottleneck
2. Messages from client may be delayed or lost
3. Duplicate requests, messages out of order
4. Distributed applications are much more complex than non-distributed ones i.e. in development, run time, maintenance, upgrades

Note:

The above problems should be considered while designing in order to avoid them from affecting the the system to be developed.

Examples of Client Applications

Server	Description	Client
SMTP	Small Mail Transport Protocol: handles sending and retrieving of electronic mail	built-in to many email programs
HTTP	Hyper Text Transfer Protocol: understands specific commands and serves web pages to the client	Internet explorer, Netscape, Mozilla etc
FTP	File Transfer Protocol: understands specific commands such as 'dir', 'get', etc, and acts according to them by serving files to the client	Database applications
POP	Post Office Protocol: understands specific commands and accesses electronic mail already received and stored on a machine	built-in to many email programs

Ordinary example of client application-ATM

Whenever you use an Automatic Teller Machine to check your balance or withdraw some money, you are actually engaging in a client-server application with a computer on your bank's network.

An ATM is actually a client programmed machine to connect you with your bank's servers.

Once a link is established with an ATM, programs on the bank's servers process and fulfill your request for cash or balance information.

Types of servers

Client and server applications are asymmetric; the process has to be controlled or managed at one end. The server does this.

In this case, we have two types of servers applications;

1. Iterative Server
2. Concurrent Server

Iterative Server Application

- One client must start, run and terminate before another client may start
- Often, client requests may have to wait if the server is busy
- It is easy to program
- Handles small, fixed size requests

E.g. a school database

Ordinary example of Iterative Server - Printer

Ordinary printer such as Hp LaserJet P102 is an example of iterative server since it prints out documents based on first come first serve.

It can never print request concurrently.

Concurrent Server Application

This kind of server is were two or more clients can run at the same time on the machine

Used when the amount of work required to handle is unknown; the server then starts another process to handle each request e.g. Mobile phone calls

- It is harder to program, given the larger or variable size requests it handles
- A program uses more system resources and this should be considered in its design

Key concepts to consider when programming client-server Applications

Here are some key concepts to keep in mind when programming client-server applications:

1. Communication Protocol: Client and server need to use a common communication protocol to exchange data. Popular communication protocols include HTTP, TCP/IP, UDP, and SMTP.
2. Socket Programming: To establish a connection between the client and server, the programmer must use socket programming. Socket programming provides an API for sending and receiving data between applications over a network.
3. Security: Client-server applications often require security measures to protect sensitive data. Encryption and authentication mechanisms can be used to secure the data transmission between the client and server.

Key concepts to consider when programming client-server Applications+

Here are some key concepts to keep in mind when programming client-server applications:

4. Application Layer: The client-server application must define a protocol for exchanging information at the application layer. This includes defining message formats and rules for the exchange of messages.
5. Scalability: Client-server applications must be designed to handle a large number of clients simultaneously. The server must be able to handle requests from multiple clients and respond quickly to each request

Client server architecture, the fat servers and fat clients

Client-server architecture refers to a system that hosts, delivers, and manages most of the resources and services that the client requests. In this model, all requests and services are delivered over a network, and it is also referred to as the networking computing model or client server network.

Client-server architecture is alternatively called a client-server model, is a network application that breaks down tasks and workloads between clients and servers that reside on the same system or are linked by a computer network

This model is also called a client-server network or a network computing model.

Summarily:

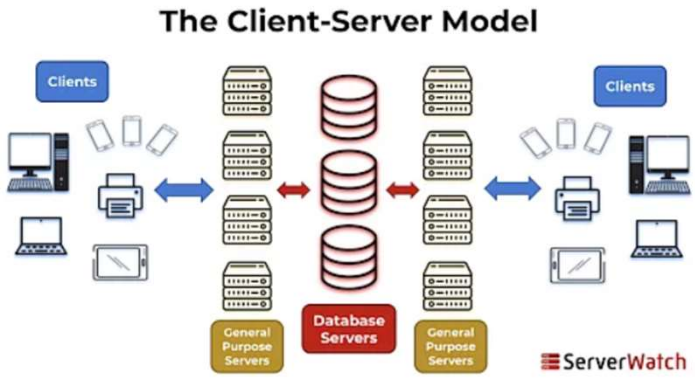
1. First, the client sends their request via a network-enabled device
2. Then, the network server accepts and processes the user request
3. Finally, the server delivers the reply to the client

The Characteristics of Client-Server Architecture

Client-server architecture typically features the following characteristics:

1. Client and server machines typically require different hardware and software resources and may come from different vendors.
2. The network has horizontal scalability, which increases the number of client machines and vertical scalability, an then moves the entire process to more powerful servers or a multi-server configuration.
3. One computer server can provide multiple services simultaneously, although each service requires a separate server program.
4. Both client and server applications interact directly with a transport layer protocol. This process establishes communication and enables the entities to send and receive information.
5. Both the client and server computers need a complete stack of protocols. The transport protocol employs lower-layer protocols to send and receive individual messages.

Visualizing Client-Server Architecture



Source: Serverwatch.

Examples of Client-Server Architecture

1. **Email servers:** E-mail servers, aided by various brands of dedicated software, send and receive e-mails between parties.
2. **File servers:** If you store files on cloud-based services such as Google Docs or Microsoft Office, you're using a file server. File servers are centralized locations for file storage and are accessed by many clients.
3. **Web servers**: These high-performance servers host many different websites, and clients access them through the Internet. Here's a step-by-step breakdown:
   i. The client/user uses their web browser to enter the URL they want
   ii. The browser asks the Domain Name System (DNS) for an IP address
   iii. The DNS server finds the desired server's IP address and sends it to the web browser
   iv. The browser creates either an HTTPS or HTTP request
   v. The server/producer sends the user the correct files
   vi. The client/user receives the files sent by the server, and the process is repeated as needed

Client-Server Architecture Assignment

1. Discuss the merits and demerits of Client-Server Architecture
2. Discuss Fat servers and fat clients
3. What's 3-Tier Client-Server Architecture?

ORB-Object Request Broker

The Object Request Broker (ORB) is middleware that uses the CORBA specification. The Object Request Broker or ORB takes care of all of the details involved in routing a request from client to object, and routing the response to its destination. The ORB is also the custodian of the Interface Repository (abbreviated variously IR or IFR), an OMG-standardized distributed database containing OMG IDL interface definitions.

On the client side, then, the ORB provides interface definitions from the IFR, and constructs invocations for use with the Dynamic Invocation Interface (DII). It also converts Object References between session and stringified format, and (for CORBA 2.4 and later ORBs) converts URL-format corbaloc and corbaname object references to session references.

On the server side, the ORB de-activates inactive objects, and re-activates them whenever a request comes in. CORBA supports a number of activation patterns, so that different object or component types can activate and de-activate in the way that uses resources best.

We will talk more o this concept later.

Remote procedure call (RPC) and Interposes communication)

A remote procedure call is an inter-process communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

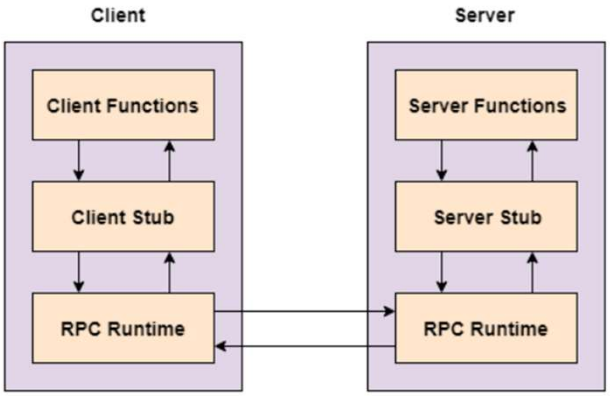
Remote procedure call (RPC) and Interposes communication)+

The sequence of events in a remote procedure call are given as follows –

1. The client stub is called by the client.
2. The client stub makes a system call to send the message to the server and puts the parameters in the message.
3. The message is sent from the client to the server by the client's operating system.
4. The message is passed to the server stub by the server operating system.
5. The parameters are removed from the message by the server stub.
6. Then, the server procedure is called by the server stub.

Remote procedure call (RPC) and Interposes communication)++

A diagram that demonstrates this is as follows –



https://www.tutorialspoint.com/remote-procedure-call-rpc

Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows —

1. Remote procedure calls support process oriented and thread oriented models.
2. The internal message passing mechanism of RPC is hidden from the user.
3. The effort to re-write and re-develop the code is minimum in remote procedure calls.
4. Remote procedure calls can be used in distributed environment as well as the local environment.
5. Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows –

1. The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
2. There is no flexibility in RPC for hardware architecture. It is only interaction based.
3. There is an increase in costs because of remote procedure call.

A guide to good Client-Server Application Design

While designing any client – server applications, its salient to have these considerations in mind;

1. Expected number of clients (simultaneous)
2. Transaction size (time to compute or lookup the answer) and their variability in terms of size
3. Available system resources (perhaps what external resources can be required in order to run the service)
4. You need to test a few alternatives of how the system will operate i.e. determine the best design

A guide to good Client-Server Application Design+

5. Graphical User Interface (GUI), it should front all ends especially from the client side
6. It should be secure! Identification required in form of passwords
7. should allow connection of other external devices such as bar code readers, finger print etc. It should hence begin processing the input and output information
8. should allow integration of both the computer information systems and non computer systems - e.g. finger print
9. Always be able to give feedback information e.g. in form of server available, unavailable, unknown etc

A guide to good Client-Server Application Design++

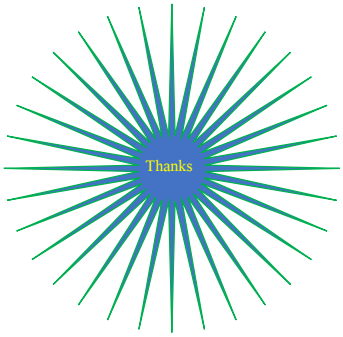
10. Timing
    - One must start before the other and wait
11. Privilege and Complexity
    i. Authentication verifying the identity of the client
    ii. Authorization determining if a given client is permitted to use the service
    iii. Data Security- data must not be revealed or compromised
    iv. Privacy protecting info about an individual

Summary

Summary

1. Course Overview
2. Introduction to client/server
3. Merit and demerits of client server,
4. Types of servers,
5. Client server architecture, the fat servers and fat clients, ORB,
6. Remote procedure call (RPC) and Interposes communication)
7. A guide to good Client-Server Application Design

Thank you for Listening



References

Java Network Programming and Distributed Computing, David R, Michael R (2002), Publisher: Addison Wesley, ISBN: 0-201-71037-4

Author Douglas K Barry Principal Barry & Associates, & Barry, D. K. (n.d.). *Object request broker (ORB)*. Service Architecture. Retrieved March 14, 2023, from https://www.service-architecture.com/articles/web-services/object-request-broker-orb.html

Bethea, A. (2017, November 21). *What is client server programming?* Small Business - Chron.com. Retrieved March 14, 2023, from https://smallbusiness.chron.com/client-server-programming-42314.html

Remote procedure call (RPC). Tutorials Point. (n.d.). Retrieved March 14, 2023, from https://www.tutorialspoint.com/remote-procedure-call-rpc

Terra, J. (2023, February 14). *What is client-server architecture? everything you should know: Simplilearn*. Simplilearn.com. Retrieved March 14, 2023, from https://www.simplilearn.com/what-is-client-server-architecture-article#:~:text=The%20client%2Dserver%20architecture%20refers,model%20or%20client%20server%20network.

Wikimedia Foundation. (2023, February 7). *Client–server model*. Wikipedia. Retrieved March 14, 2023, from https://en.wikipedia.org/wiki/Client%E2%80%93server_model