The client/server model has become one of the central ideas of network computing

[1]Nalin Chaudhary, [2]Anurag Goatam
[1]Assistant Professor, 2Computer Application Student
[1]Department of Computer Science & Engineering, [2]Department of Computer Application
[1]Bhagwant University, Ajmer, Rajasthan, India

Abstract: In this paper we are studying the client/server model has become one of the central ideas of network computing. A Main Information Server is a node that is used to keep track of all the servers within the distributed architecture. Thus all the nodes which want to act as a server within the distributed architecture must be registered with this main information server. The main information server is used to route a resource request to all the servers in the distributed architecture if a particular server fails to handle the resource request of its client. In this focus is on systems to store/access/share data and operations on data. Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server. Standard networked functions such as email exchange, web access and database access, are based on the client/server model.

***IndexTerms* – Client, Server, Network, Route, Resource, Request.**

I. Introduction

Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems. Web services protocols are standards which enable distributed systems to be extended and scaled. In general, an open system that scales has an advantage over a perfectly closed and self-contained system. Openness cannot be achieved unless the specification and documentation of the key software interface of the component of a system are made available to the software developer. During this period the hardware used for designing of distributed systems was based on based on Integrated Circuits. Backward compatible computers were used , first real operating systems. Proliferation of computers in large business'. Networked computers were designed which marked towards the beginning of the internet. In this period there was use of single users systems and all custom built to different specifications. Thus each system was designed according to the requirements of its users. During this period the hardware used for designing of distributed systems was based on transistors. The distributed systems made use of batch systems. Languages were first introduced in order to implement distributed systems. During this period the hardware used for designing of distributed systems was based on based on Integrated Circuits. Backward compatible computers were used , first real operating systems. Proliferation of computers in large business'. Networked computers were designed which marked towards the beginning of the internet.

A **Client** is a node or a simple part of the local network that can be part of the distributed system by connecting it to a server. This node is equipped with the privileges of adding, deleting, modifying a resource present with it. In order to access the resources of the network connected using distributed architecture the client makes request to its server which provides it with requested resource [6].

A **Server** is a node or a simple part of the local network that can be part of the distributed system by connecting it to a main information server. Thus it is a node that is particularly used to connect several clients (or nodes) together and is particularly use to deal with resource requests of its clients [6].

A **Main Information Server** is a node that is used to keep track of all the servers within the distributed architecture. Thus all the nodes which want to act as a server within the distributed architecture must be registered with this main information server. The main information server is used to route a resource request to all the servers in the distributed architecture if a particular server fails to handle the resource request of its client [6].

II. ARCHITECTURE

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely-coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.

Distributed programming typically falls into one of several basic architectures or categories: Client-server, 3-tier architecture, N-tier architecture, Distributed objects, loose coupling, or tight coupling.

- **Client-server** — Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.
- **3-tier architecture** — Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.
- **N-tier architecture** — N-Tier refers typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
- **Tightly coupled** (clustered) — refers typically to a cluster of machines that closely work together, running a shared process in parallel. The task is subdivided in parts that are made individually by each one and then put back together to make the final result.
- **Peer-to-peer** — an architecture where there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and servers.
- **Space based** — refers to an infrastructure that creates the illusion (virtualization) of one single address-space. Data are transparently replicated according to application needs. Decoupling in time, space and reference is achieved.
- **Grid approach** — A grid uses the resources of many separate computers, loosely connected by a network (usually the Internet), to solve large-scale computation problems. Public grids may use idle time on many thousands of computers throughout the world. Such arrangements permit handling of data that would otherwise require the power of expensive supercomputers or would have been impossible to analyze[6].

III. CLIENT - SERVER ARCHITECTURE

The **client-server** software architecture model distinguishes client systems from server systems, which communicate over a computer network. A client-server application is a distributed system comprising both client and server software. A client software process may initiate a communication session, while the server waits for requests from any client. Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server. Standard networked functions such as email exchange, web access and database access, are based on the client/server model. For example, a web browser is a client program at the user computer that may access information at any web server in the world. To check your bank account from your computer, a web browser client program in your computer forwards your request to a web server program at the bank. That program may in turn forward the request to its own database client program that sends a request to a database server at another bank computer to retrieve your account balance. The balance is returned to the bank database client, which in turn serves it back to the web browser client in your personal computer, which displays the information for you.

IV. THE CLIENT/SERVER MODEL HAS BECOME ONE OF THE CENTRAL IDEAS OF NETWORK COMPUTING

Most business applications being written today use the client/server model. So do the Internet's main application protocols, such as HTTP, SMTP, Telnet, DNS, etc. In marketing, the term has been used to distinguish distributed computing by smaller dispersed computers from the "monolithic" centralized computing of mainframe computers. But this distinction has largely disappeared as mainframes and their applications have also turned to the client/server model and become part of network computing. Each instance of the client software can send data requests to one or more connected *server*s. In turn, the servers can accept these requests, process them, and return the requested information to the client. Although this concept can be applied for a variety of reasons to many different kinds of applications, the architecture remains fundamentally the same. The most basic type of client-server architecture employs only two types of hosts: clients and servers. This type of architecture is sometimes referred to as *two-tier*. It allows devices to share files and resources. The two tier architecture means that the client acts as one tier and application in combination with server acts as another tier[6]. These days, clients are most often web browsers, although that has not always been the case. Servers typically include web servers, database servers and mail servers. Online gaming is usually client-server too.

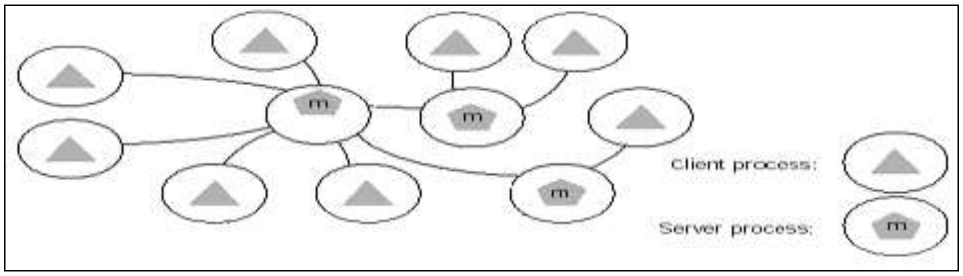


Fig. 1: Two Tier Client – Server Architecture

4.1 Characteristics of a Client

- Initiates requests to server.
- Waits for replies from server.
- Receives replies from server.
- Usually connects to a single server at one time.
- Typically interacts directly with end-users using a graphical user interface.

4.2 Characteristics of a Server

- Never initiates requests or activities to its clients.
- Waits for requests from clients and replies to requests from the respective connected clients.

A server can remotely install/uninstall applications and transfer data to the intended clients.

4.3 Advantages of Client – Server Approach

- In most cases, a client-server architecture enables the **roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network**. This creates an additional advantage to this architecture: greater **ease of maintenance**. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change. This independence from change is also referred to as *encapsulation*.
- All the data is stored on the servers, which generally have far **greater security** controls than most clients. **Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data.**
- Since **data storage is centralized**, **updates to that data are far easier to administer than what would be possible under a P2P paradigm**. Under a P2P architecture, data updates may need to be distributed and applied to each "peer" in the network, which is both time-consuming and error-prone, as there can be thousands or even millions of peers.
- Many mature client-server technologies are already available which were designed to ensure security, '**friendliness**' of the user interface, and ease of use.
- It functions with multiple **different clients of different capabilities[6]**.

V. The Flooding Algorithm And Flowchart

A flooding algorithm is an algorithm for distributing material to every part of a connected network. The name derives from the concept of inundation by a flood.

Flooding algorithms are used in systems such as Usenet and peer-to-peer file sharing systems and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks.

There are several variants of flooding algorithm : most work roughly as follows:-

- Each node acts as both a transmitter and a receiver.
- Each node tries to forward every message to every one of its neighbors except the source node.

This results in every message eventually being delivered to all reachable parts of the network.

Real-world flooding algorithms have to be more complex than this, since precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system[1].

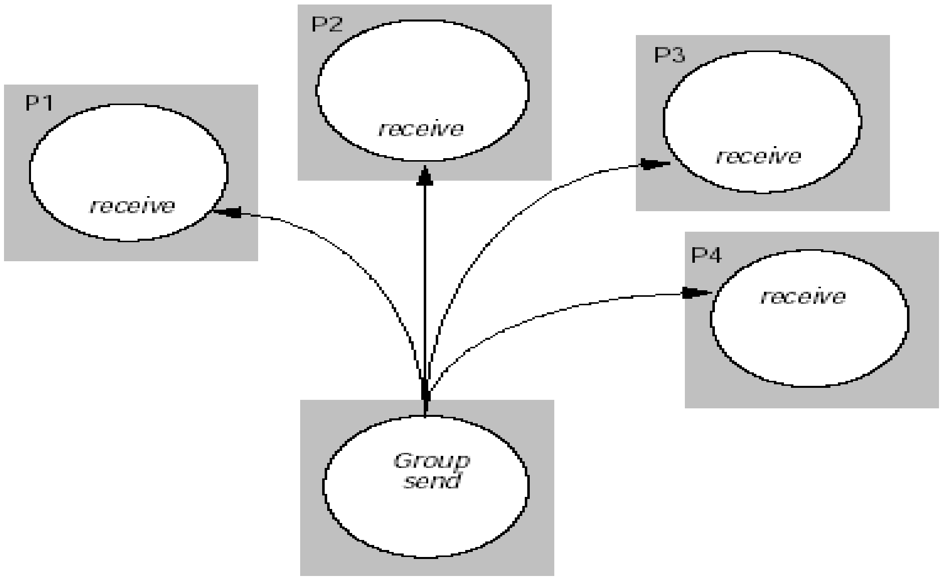
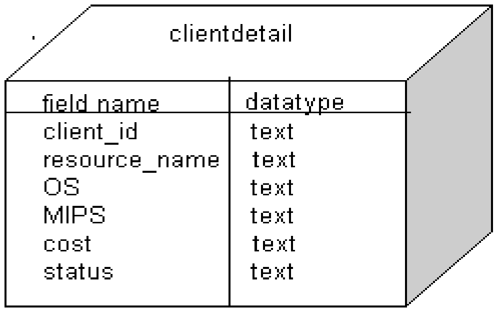


Fig. 2 : Flooding Approach

VI. CLIENT - SERVER ARCHITECTURE DATABASE MODEL

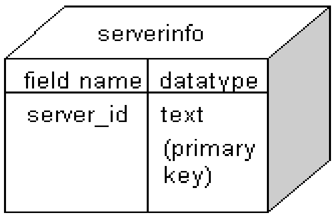
Data Base models are used to pictorially represent the database, which tells the user that what all tables are being used, what all fields are there in the table and also the type (data type of the language used to implement the table) of the field.
The following figure give what exactly the database (with its different tables) would like for a server node.



clientdetail

field name	datatype
client_id	text
resource_name	text
OS	text
MIPS	text
cost	text
status	text

Fig. 3 : DATABASE MODEL – Server
The following figures give what exactly the database (with its different tables) would like for a main information server node.



serverinfo

field name	datatype
server_id	text (primary key)

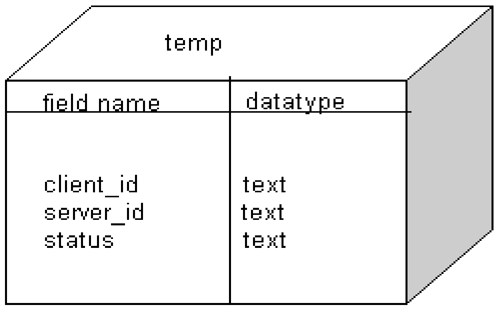


Fig. 4: DATABASE MODEL – Main Information Server

The following figure give what exactly the database (with its different tables) would like for a client node.

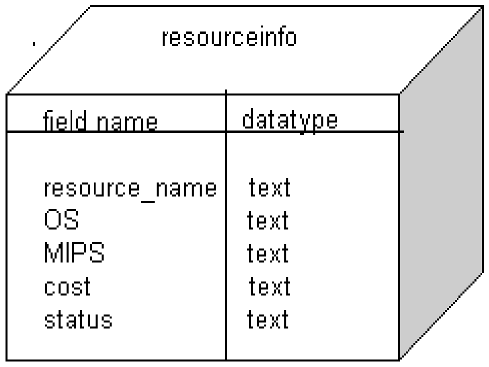


Fig. 5: DATABASE MODEL – Client

VII. CONCLUSION

This paper has a wide scope of enhancing its power of searching the servers for the presence of a resource. Thus instead of going for flooding via the main information server for a query of a Resource's presence any particular Routing Protocol can be implemented based on adaptability for network bandwidth available, network congestion, etc and this routing protocol approach can be compared with the flooding approach. Also trust and security could be added over this method that would imply authentication, integration and privacy. In this paper we are studying the client/server model has become one of the central ideas of network computing. A Main Information Server is a node that is used to keep track of all the servers within the distributed architecture. The servers can accept these requests, process them, and return the requested information to the client. Although this concept can be applied for a variety of reasons to many different kinds of applications, the architecture remains fundamentally the same. The most basic type of client-server architecture employs only two types of hosts: clients and servers.

REFERENCES

[1] Computer Networks – Andrew S Tanenbaum
[2] Core Java 2 Volume 2-Advanced Features – By Cay S. Horstmann, Gary Cornell Sun Microsystems
[3] The Complete Reference Java 2 – By Herbert Schildt, Tata McGraw-Hill
[4] www.sun.java.com
[5] http://java.sun.com/docs/books/tutorial
[6] An introduction to Database Systems – By C.J. Date
[7] Forman G.H., Zahorjan J., The Challenges of Mobile Computing", University of Washington, Tech Report #93-11- 03, ftp.cs.washington.edu, March 1994.
[8] Kajan E., Open Systems: Concepts, Components and future applications, Prosveta, Nis, 1994.
[9] Mitrovi} A., et all., A Scalable Object-Oriented GIS Framework, Workshop on New Developments in Geographic Information Systems, Milan, Italy, 6-8 March 1996, pp. 20-21.
[10] Kajan E., et all., Domestic GIS Software Development: A Historical Point of View, YUGIS '96, Belgrade, Yugoslavia, 14-15 March 1996, pp. 223-227.

[11] Djordjevi}-Kajan S., Milovanovic B., et all., Hail Suppression Information System of a Radar Center, GIS/LIS '95/'96, Budapest, Hungary, June 1—14, 1996, pp. 102-111.

[12] Mitrovi} A., et all., Object-Oriented Paradigm Meets GIS, a New Era in Spatial Data Management, YUGIS '96, Belgrade, Yugoslavia, 14-15 March 1996, pp. 141-148.

[13] Notkin D., et all, Interconnecting Heterogeneous Computer Systems, Communications of the ACM, Vol. 31, No 3, March 1988, pp. 258-273.

[14] Harrison C. G., et all., Mobile Agents: Are they a good idea?, IBM Research Report RC 19887, 1994.

[15] Sterbenz J. P. G., Protocols for High Speed Networks: Life After ATM, The 4th IFIP Workshop on protocols for high speed networks, Vancouver, Canada, 1995, pp. 3-18.

IJCRT