Design and Architecture of Distributed Systems

[1]Nalin Chaudhary
[1]Research Scholar
[1]Bhagwant University, Ajmer, India

Abstract: ***Our application is based on handling resource discovery efficiently in a decentralised way in a distributed system. We have implemented a main information server which maintains information about all the servers in the network because in order to be the part of network all the servers have to get themselves registered with the information server. Each server keeps a record of all the resources present among its clients. Whenever a client issues a request for a resource it is firstly checked onto itself and if the requested resource is not available there then it makes a request to its immediate server. If it also fails at its immediate server then with the help of information server the request is broadcasted in the network. If this request reaches a server who possesses the resource then an acknowledgement will be passed to the server in the network which was responsible for generating the request. Our main aim of this paper is to have efficient resource discovery in a decentralised manner for the distributed architecture. A distributed system is a collection of autonomous computers with the aim of enhancing resource sharing. Resource management in a distributed system involves resource discovery and resource scheduling. Resource discovery can be implemented in a centralized or decentralized manner.***

Index Terms – Network, Centralized, Decentralized, Discovery, Resource, Distributed System, Sharing.

I. INTRODUCTION

Distributed system is a collection of loosely coupled processors interconnected by a communication network where each processor is addressed as a machine or a node or a host or a computer. The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. The basic architecture of a distributed system is shown as follow :-

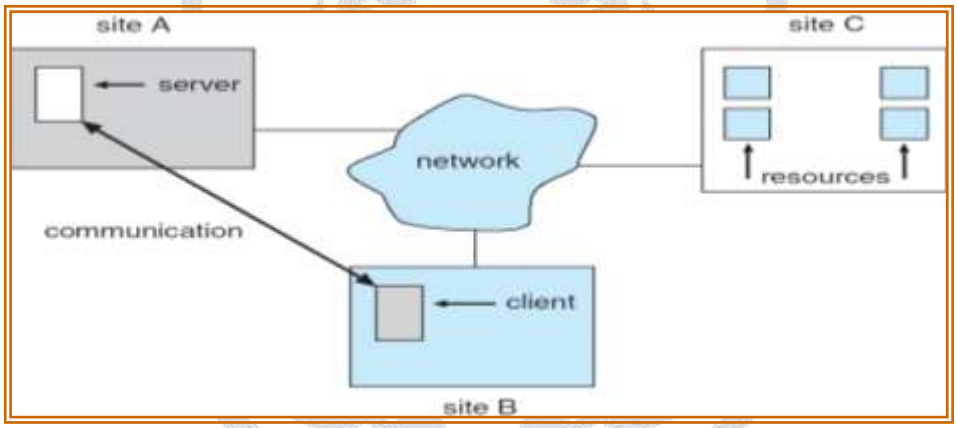

Figure 1.1 Distributed System Architecture

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[1].
- **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[1].
- **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[1].
- **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[1].
- **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[1].
- **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[1].
- **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[1].

II. REVIEW OF WORK

In words of Tanenbaum : "**A distributed system is a collection of independent computers that appear to the user of the system as a single computer**". It is indeed a bunch of "computers" connected by "wires". Nodes are (at least) semi-autonomous...but run software to coordinate and share resources. The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. In this focus is on systems to store/access/share data and operations on data. Data is moved around the network and is delivered to the right places at the right times, safely and securely. The prime focus is laid on Internet information services and their building blocks[6].

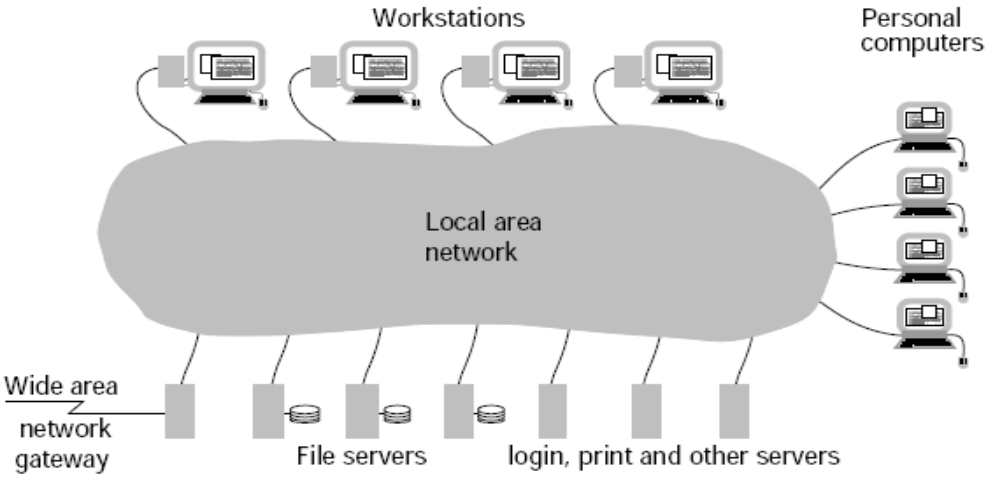

Fig. 1.2 A Simple Distributed System

i) 1945 – 1955 : – 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.
ii) 1955 - 1965 : – 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.
iii) 1965 - 1980 : – 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.
iv) 1980 - 1990 : – During this period the hardware used for designing of distributed systems was based on Large Scale Integrated Circuits which led to the development of personal computer. Further network technologies were improved. Network Operating Systems came into picture. It was this era only which marked the popular use of distributed systems.

System	*Organization*	*Network*	*Computers*	*Date*
Xerox DFS	Xerox PARC	Ethernet and Internet	Xerox Alto	1977
Cambridge DCS	Cambridge University	Cambridge Ring	LSI-4, M68000	1979
Locus	UCLA	Ethernet	VAX, IBM PC	1980
Apollo Domain	Apollo Computers	Token Ring	Apollo	1980
Newcastle Connection	Newcastle University	various	various	1980
Grapevine	Xerox PARC	Ethernet	Xerox Alto	1981
Cedar	Xerox PARC	Ethernet	Xerox Dorado and Dandelion	1982
V system	Stanford University	Ethernet	VAX, Sun	1982

Table 1 Milestones in Distributed Systems

III. 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].

IV. 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.

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.

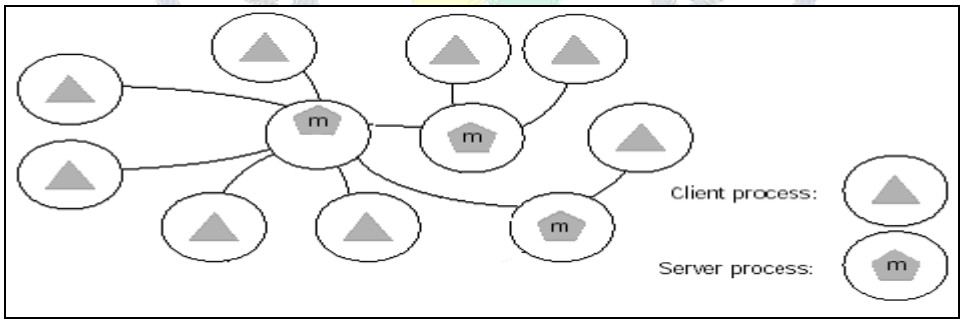

Fig. 1.3 Two Tier Client – Server Architecture

The interaction between client and server is often described using sequence diagrams. Sequence diagrams are standardized in the Unified Modeling Language.

V. DESIGN AND ARCHITECTURE

Each client has one solely dedicated server which bears the responsibility of forwarding the client's request from its internal networks to the global networks. Thus it becomes the job of the server to alert the main information server about the searching of a requested resource in the entire distributed network.

The project introduces the job of flooding which is required to look for the presence of any resource in the distributed architecture. Whenever any server receives a flooded request from the main information server it replies back the main information server with the acknowledgement that any of its clients possesses the requested resource in case its database contains a record for the resource being searched. In response to the received acknowledgement the main information server guides the requested server about the presence of the resource so that its respective client which made the request can utilize the resource. In case the requested client makes use of the resource then the main information server guides the client from which the resource was taken to set the availability status of the resource to false so that no other client in the distributed network can make use of the resource that has been given to a client. Thus all the servers in the distributed network are in contact with the main information server where each server has its respective clients.

The current work can be evolved to include a particular algorithm for routing the various servers in the network for a resource request in such a way that the routing approach has adaptability for network bandwidth available, network congestion, etc. Also trust and security could be added over this method that would imply authentication, integration and privacy[6].

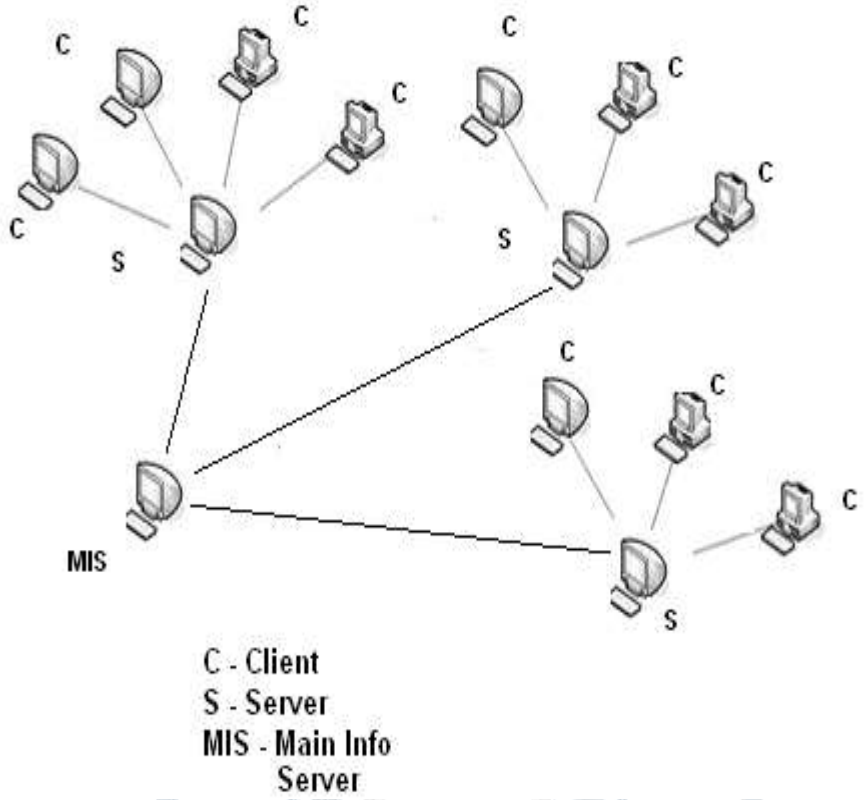

Fig. 1.3 Basic Architecture

Control Flow

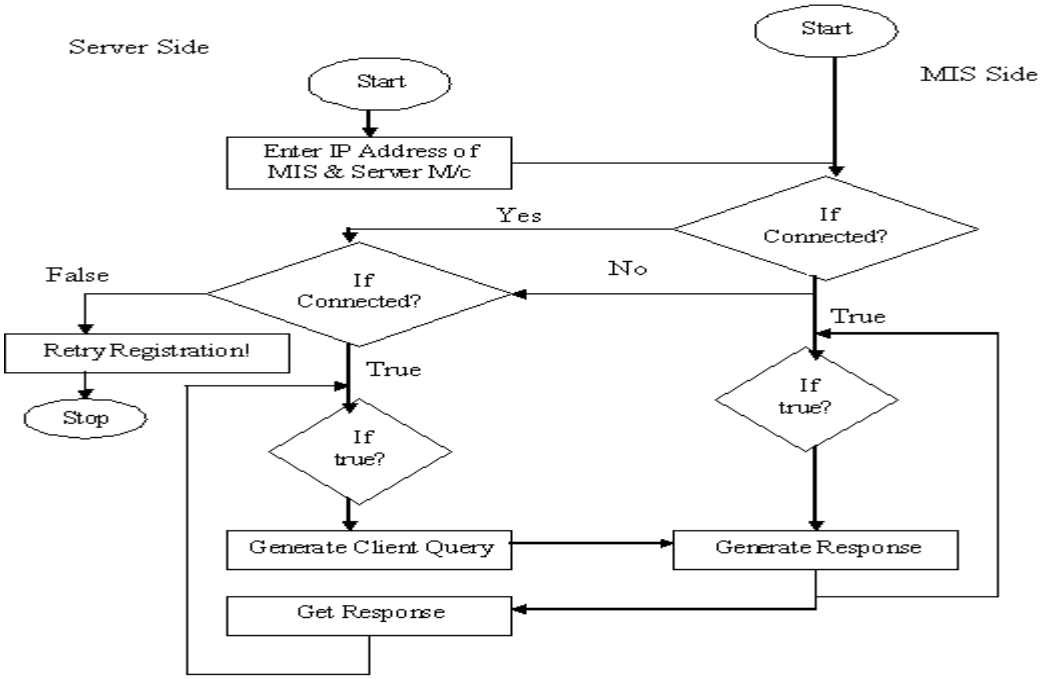

Fig. 1.4 Registration of Server with Main Information Server

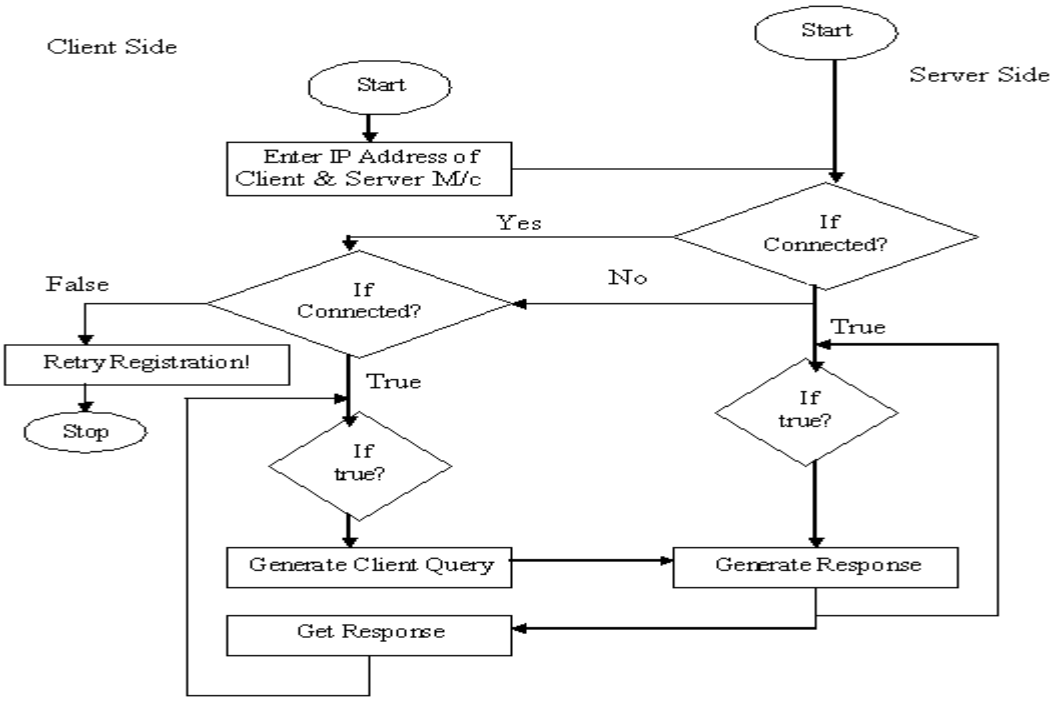

Fig. 1.5 Registration of Client with Server

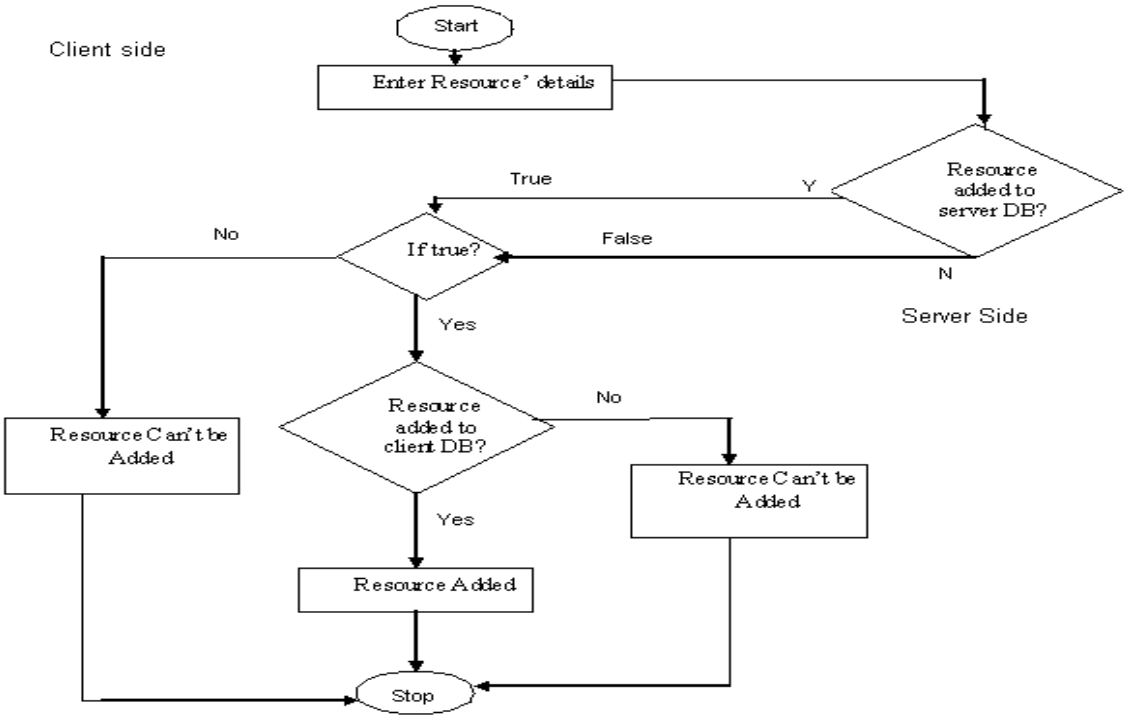

Fig. 1.6 Addition of a Resource at Client Site

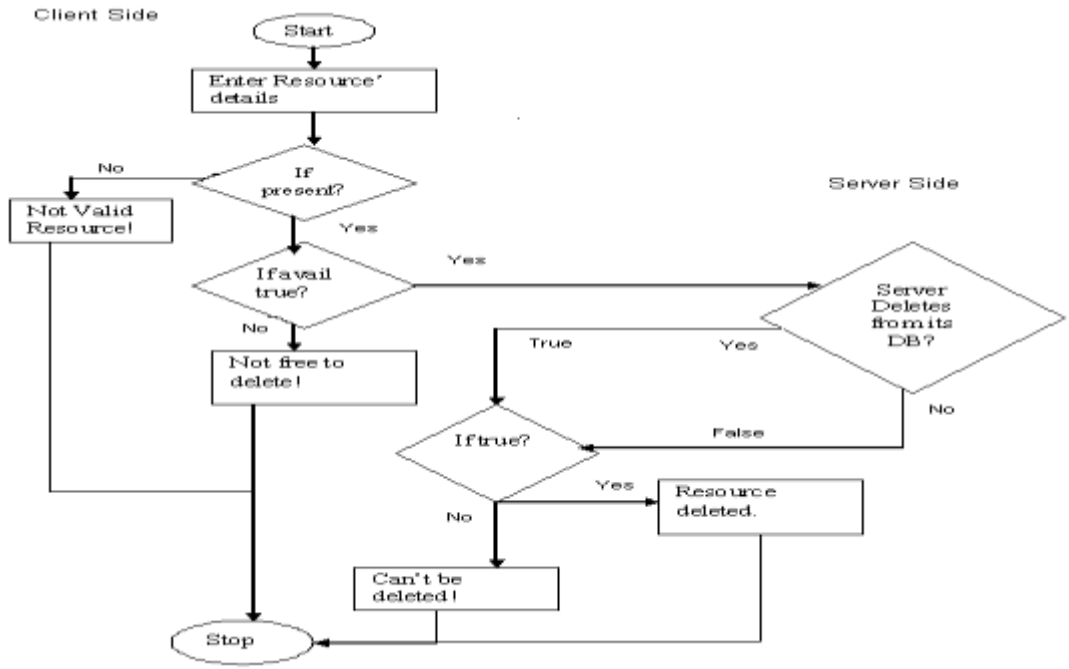

Fig. 1.7 Deletion of a Resource at Client Site

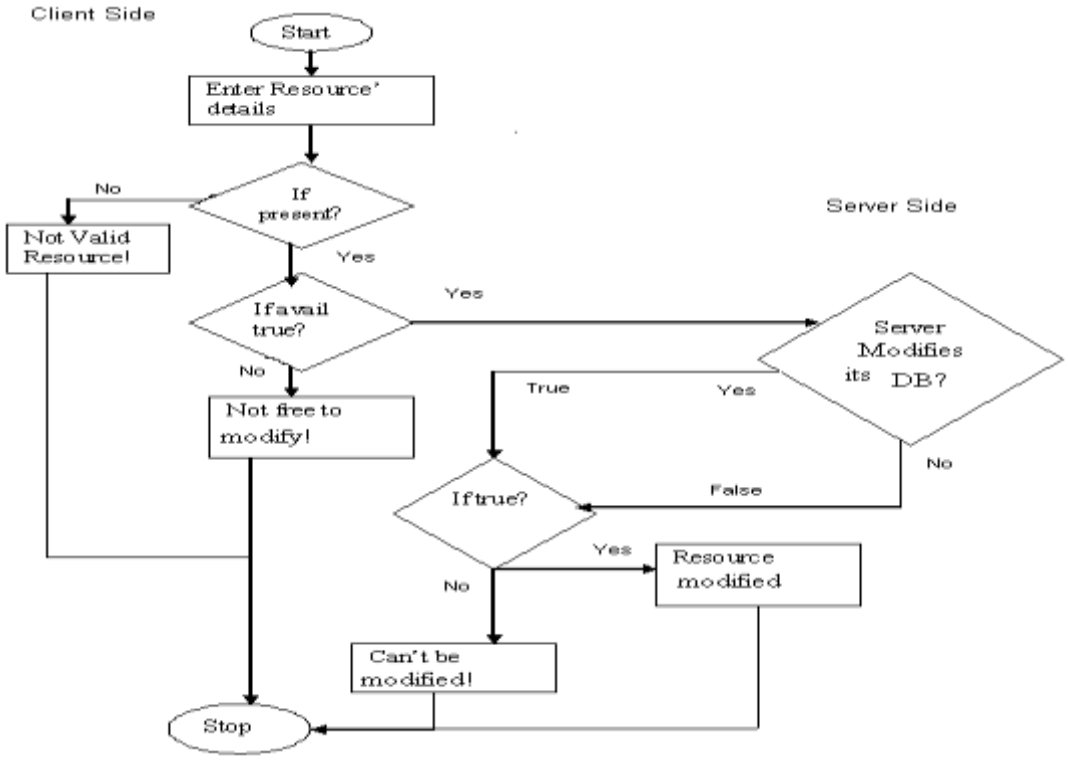

Fig. 1.8 Modification of a Resource at Client Site

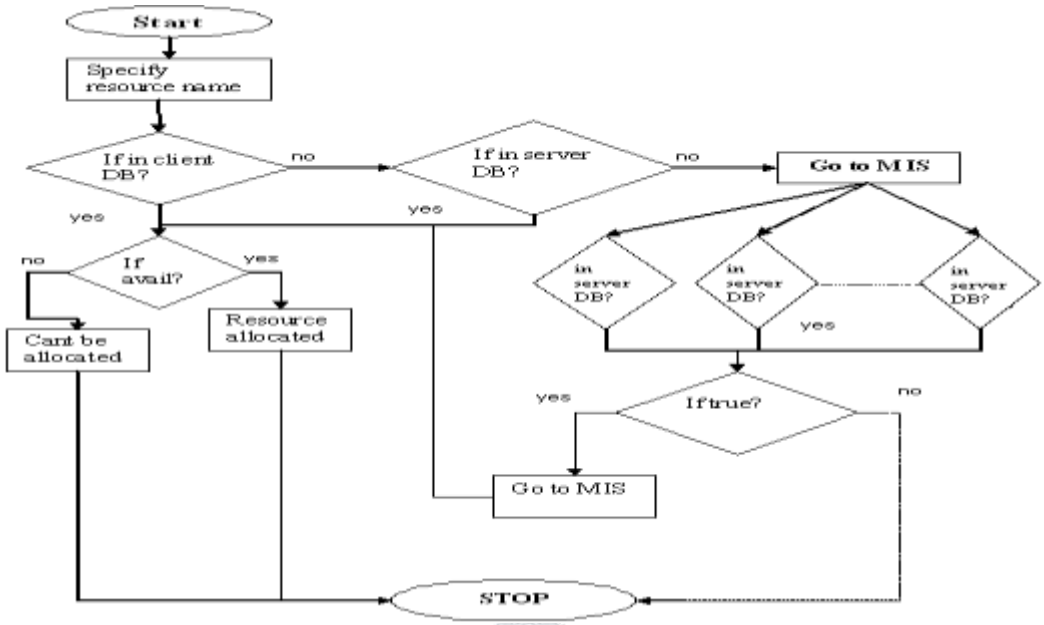

Fig. 1.9 Request of a Resource at Client Site

VI. CONCLUSION

In this paper we provides a complete overview of Distributed System, Client – Server Architecture with prime concentration on the Resource Discovery which forms the basis of the paper. It also gives basic idea about the frontend and backend platform of the paper. This paper clearly gives the outlook of the implemented distributed system and this design and architecture. It shows the Graphical User Interface (GUI) provided to the user and the various options the user can access in the project. This paper also acts as
the guide for the usage of this paper. The 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. Through this approach the main information server will flood all the servers of the network for the requested resource incase the requesting server fails to deal with the resource request in its own network and forwards its request to the main information server.

REFERENCES

[1] Akshaye Dhawan 2009. Distributed Algorithms for Maximizing the Lifetime of Wireles Sensor Networks. Doctor of Philosophy, Paper Under the direction of Sushil K. Prasad, December 2009,Georgia State University, Atlanta, Ga 30303. Available at:http://etd.gsu.edu/theses/available/etd.../Dhawan_Akshaye_200912_PhD

[2] Chinh Trung Vu, 2009. Distributed Energy-Efficient Solutions for Area Coverage Problems in Wireles Sensor Networks. Doctor of Philosophy, Paper Under the direction of Dr. Yingshu Li, August 2009, Georgia State University, Atlanta,Ga30303.

[3] Chee-Yee Chong and Srikanta P. Kumar 2003. Sensor Networks: Evolution, Opportunities and Challenges. Proceeding of the IEEE, vol. 91, no.8, Aug. 2003.

[4] R. Hahn and H. Reichl 1999. Batteries and power supplies for wearable and ubiquitous computing", in Proc. 3rd Intl. Symposium on Wearable computers.

[5] M. Cardei, J. Wu, N. Lu, M.O. Pervaiz 2005. Maximum Network Lifetime with Adjustable Range. IEEE Intl. Conf. on Wireles and Moble Computing, Networking and Communictions (WiMob'05), Aug.

[6] G. J. Pottie and W. J. Kaiser 2000. Wireles integrated network sensors. Communiction ACM, 43(5):51-58.

[7] P. Berman, G. Calinescu, C. Shah and A. Zelikovsky, "Power Efficient Monitoring Management in Sensor Networks," IEEE Wireles Communiction and Networking Conference (WCNC'04), pp. 2329-2334, Atlanta, March 2004.

[8] Brinza, D. and Zelikovsky, A 2006. DEEPS: Deterministic Energy-Efficient Protocol for Sensor networks", ACIS International Workshop on Self-Assembling Wireles Networks (SAWN'06), Proc. of SNPD, pp. 261-266, 2006.

[9] M. Cardei, J. Wu, "Energy-Efficient Coverage Problems in Wireles Ad-Hoc Sensor Networks", Computer Communictions Journal (Elsevier), Vol.29, No.4, pp. 413-420.

[10] Jim Kurose and Keith Ross 2004.Computer Networking: A Top Down Approach Featuring the Internet. 3rd edition. Addison-Wesley.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, 2002. A Survey on Sensor Networks. IEEE Communictions Magazine, pp 102-114.

[12] M. Cardei, M.T. Thai, Y. Li, and W. Wu, 2005. Energy-efficient target coverage in WSNs. In Proc. of IEEE Infocom.

[13] Computer Networks – Andrew S Tanenbaum

[14] Core Java 2 Volume 2-Advanced Features – By Cay S. Horstmann, Gary Cornell Sun Microsystems

[15] The Complete Reference Java 2 – By Herbert Schildt, Tata McGraw-Hill

[16] www.sun.java.com

[17] http://java.sun.com/docs/books/tutorial

[18] An introduction to Database Systems – By C.J. Date