DESIGN OF EFFICIENT MULTI CLIENT – SINGLE SERVER APPLICATION USING SOCKET PROGRAMMING

Mr. Praveen P. Korni, Student,
Dr. D.Y. Patil Institute of Management and Research
Pune, India,
kornipraveen78@gmail.com

Dr. Rupali Kalekar, Assistant Professor,
Zeal Institute of Business Administration Computer Application & Research,
Pune, India,
joshi.rups@gmail.com

Dr. Rajesh Kashyap, Professor,
Zeal Institute of Business Administration Computer Application & Research,
Pune, India,
rajesh.kashyap@zealeducation.com

ABSTRACT

In recent era web communication has become mandatory for all domains. When it comes to web communication, servers always have loads on it for communication. There are various server types which support this like single server communication, multi-client communication etc. There is a need to provide an effective way of communication between client and server. The present study will focus on the detailed overview of designing a client- server application that enables the operating of a single server from multiple clients using socket programming. Authors of this paper believe that the provided solution is effective as it reduces the time required in the communication. The authors of this paper will endow with the design of an application named "Remote Server Communication" using Java, TCP, UDP communication layers to demonstrate the communication in a distributed computing environment using socket programming concepts. We have tried to make utmost use of the latest technologies which will definitely enhance and support the topic. The current paper has delivered the insights on the concept of socket programming along with distributed computing architecture. The main objective behind this paper is to recognize the overview of designing a client-server application. The main principle of this research is to efficiently decrease the time consumption in making recursive calls to the server and making a concurrent server rather than an iterative server that can asynchronously manage multiple clients.

Keywords: Socket Programming, Threaded Server, Concurrent Server, Threading, Distributed Computing, Java.

Introduction

In Client server architecture the first one who is activated is the server. Server always looks for the call initiation from the client. Upon receiving the call from the client, the server reports the status of the clients where it mentioned the total number of servers connected to the respective networks. Multiple clients requesting servers at a single time may cause problems in the communication process. There will be communication delay between client and server. Further to add more liberty to the network use of connection oriented and connection less both approaches are considered.

A multi-client application can deliver an efficient solution by using socket programming. The basic approach behind this work is connecting multiple computers together to work efficiently and resolve the network related problems. We believe that implementation of such a concept will enhance scaling and solve complex problems associated with network communication. The main objective of the existing paper is to deliver insights on the overview of designing a client- server application. The main principle of this research is to efficiently decrease the time consumption in making recursive calls to the server and making a concurrent server rather than an iterative server that can asynchronously manage multiple clients.

A multi-client - single server is a distributed computing architecture in which there is a single server machine that handles multiple client machines. Multi server has the potential to deliver two or more services at a glance and this supports our idea to reduce time complexity using this network and offer the network a speedy communication. There are plenty of models available but for present study researchers have considered the use of socket programming. Using this architecture, multiple clients can connect and process data on the server machine without redundancy. In the native approach, if multiple clients had to connect to a server, they would do that using a LAN network cable or if they wanted to get some data, they would copy the data in a portable storage device e.g. an USB drive and would access data through the portable device.

This approach not only increases redundancy but also takes a lot of time. To cope with this, we can connect all the client machines through a wireless/ wired LAN network and let them access the server using a socket. The server machine can open the socket at their end and all the clients can connect to that socket.

Client server model in networking has connectivity between more than two nodes. The communication between networks is always initiated through the client by requesting the server with some commands. As per the request received from the client server gives response to the client. This elaborates the general way of communication between two nodes. But it has been observed that there is load on the server when multiple nodes want to initiate the communication to exchange the data at the same moment. The truth of the networking is the server never controls the clients; it just delivers the services to the client as per request. Hence, the server can use multiple techniques to handle the clients. The ways include 1.Iterative and 2.Concurrent. Our goal here is to make a server that is efficient, so we will discuss in the following paper that using which techniques, we can increase the efficiency of the server.

Review Of Literature

Kalita (2014) mentioned the importance of socket programming in client server applications. Xue & Zhu (2009) introduces socket based programming principles which are essential for our study. They have also briefed about connection oriented programming which can deliver sustainable communication throughout the network. Poorvika, Gowda, Poojary & Nayak (2023) elaborated on the fundamentals of network programming and socket-based network application along with its development methods. Kumar (2019) insights on principals, concepts, designing aspect of client-server web application which helps to understand about basic terminologies related to client server communication. Kadhim & Jasim (2014) suggested working on wireless connection to connect with client –server. Further they mentioned that such kinds of solutions will deliver plenty of services to the clients.

Maata, Cordova, Sudramurthy & Halibas (2017) has developed the OpTel Billing System (OBS) using Java NetBeans and TCP datagram for its communication in distributed computing. The communication approach elaborated here can be utilized for Interface design, socket programming. Singh, Sharma, and Kumar (2011) highlighted on File Server. File server is being implemented using Java Sockets, based on TCP protocol. The server was able to respond to file requests of different formats. A dedicated Server is meant for client authentication based on the IP address of the connecting client and maintains a list of files on it. BO & Karthik (2020) has created a server and a number of client's interfaces and clients communicate with the server using a socket module that is implemented using TCP socket. Puli. Srilakshmi, Sai , Thot & Tadala (2020) has developed an Multithreaded Server client Chatting application using a reliable and secure network programming based on Java socket programming using Transport Control Protocol (TCP).

Srivastava, Sinha & Gupta (2013) has investigated socket programming and remote method invocation RMI and discussed the advantage of Socket Programming over RMI. Karan, Krithika, Saurabh, Srishti & Manikandan (2018) has focused on secure communication between different groups of people on a network using TCP/IP and that uses RSA algorithm for encryption of messages. Shah, Server, & Tomer (2022) has developed an application that runs utilizing an attachment programming which is a blend of an IP address and a port number. Takawale & Jaiswal, (2018) has elaborated on transforming the data through connection oriented protocol and thus ensures a reliable transfer of data.

Objectives Of The Study

1. To recognize overview of designing a client- server application.
2. To make the server efficient.
3. To make a design for Remote Server Communication.

Research Methodology

This paper is based on secondary data. Authors have rigorously worked on the validity of the data.

Research Design

This research is based on exploratory research type.

Basic Concepts Of The Paper

The present study is based on the various socket programming concepts in distributed computing and fundamentals of networking, computing and java language.

A. Socket Programming:

Socket programming is a method in which we can connect two nodes in a network to communicate with each other. The server forms a socket on a particular port on the IP and acts as a listener while the client reaches out to the server using that port and connects to it.

B. Sockets:

A socket is an interface between the application and the system. Just like the electric sockets that we use at our home, sockets also work in similar fashion. It enables us to connect – from one machine to the other.

The application makes an attachment and the sort of attachment decides the style of correspondence i.e. connection oriented or connection less

a. Iterative Server:

A server that handles one client request at a time is called an Iterative Server. In an iterative server, when multiple clients are requesting data, the server is able to serve only a single request at a particular time. The server processes the request and sends the data then accepts the next request by another client, serves it and so on. A typical flow of an iterative server would be Accept connection => Get Request => Process => Serve => Close connection => Accept new request.

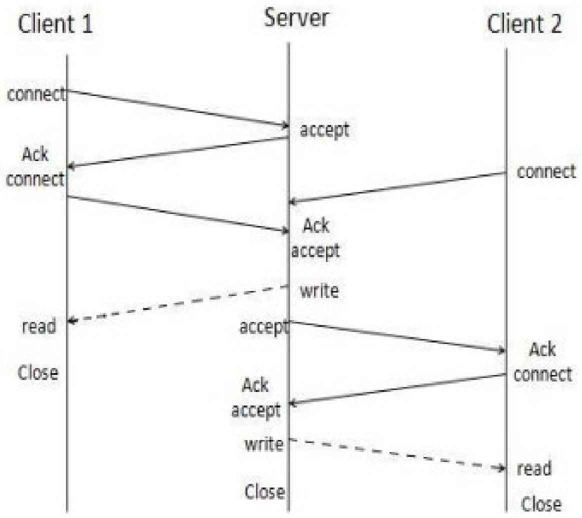

Fig. No. 1 Flow of an iterative server

In this type of server, the latency is very high as the server can serve only a single request at a time. So the next client needs to wait until the earlier request ends. Eventually, when there are many clients in the queue, the clients need to wait for a very long time.

b. Concurrent Server:

A Concurrent Server is a server that is capable of handling multiple client requests at a single time. As the name suggests, the server concurrently handles multiple clients, eliminating the waiting time which was a major drawback of the iterative server.

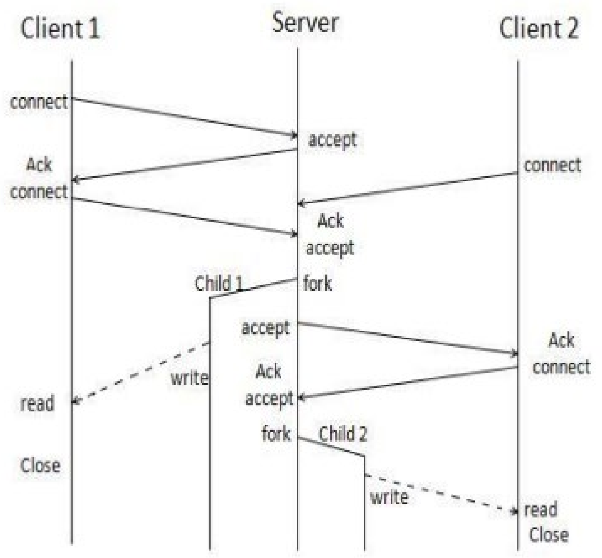

Fig. No. 2 Concurrent server

The server can obtain concurrency by two ways:
Forking: The server can fork a new process for each client that requests connection and serve all the requests from that process.

Threading: The server can create a new thread for each client that requests a connection and can serve all requests from that thread and end the thread when the client closes connection.

Communication
The communication between the client and the server may occur through the UDP or the TCP. The UDP is a connectionless protocol and is also light weight whereas the TCP is connection oriented as well as reliable.

a. Connection
For connection, we will create a socket on a port over the IP on the server side. The client can then connect to the server by using the IP and the port on which the socket was created.
Once the connection request is sent by the client, the server can then connect to the client by accepting the connection request.

b. Sending and Receiving Data
Post authentication and successful connection by the server, the client can send and receive data. The client can send data to the server over the socket using TCP which will then be interpreted by the server. After processing the request and subsequent data, the server sends the response which is read by the client.

The Efficient Design
Using the above mentioned concepts and the communication types and protocols, we have designed a server machine which is more efficient.

Programming Language
There are a variety of languages that can be used to program systems like C, Python, Java etc. but to program networks and for socket programming, Researchers of this paper have used Java Programming Language for the development.

The benefits of Java over other languages are:
1. Flexible and Powerful
2. Very Sufficient
3. Updated information can be used to send only between devices.
4. Low network traffic if efficient use.

Creating Socket:
Researchers of this paper have used the java socket () to create a new socket and open a port for the client to connect.

Handling Clients:
Once a server socket is created, we can now accept connection requests from the clients.

1. **Authenticating Client:**
   Once a client sends a connection request, the server would authenticate the client by a custom authentication algorithm. Once the client is authenticated, the server will start accepting data requests from the client.

2. **Handling Client Requests:**
   Post authentication, the server is now ready to handle client requests.
   As we saw in II C, II D we need to make the server a concurrent server to let it handle multiple clients.
   There is an option to either fork a process or initiate a thread for the clients. As a thread is much lightweight and faster than a process, we will initiate a thread for each of the client that is authenticated by the server.

3. **Closing the connection:**
   Once all the requests from the client are completed, the client may request to close the connection. On the close request, the server will close the connection for that client and will stop the thread on which the client was processing.

Conclusion

This client server model can be used wherever there is need for asynchronous data processing to and fro from the client to the server and back. This model is advantageous as it has less latency and is also secure. The server is able to handle multiple clients at once without interrupting the request processing of another client. This decreases the latency and enables faster connection rate.

This server is able to handle all types of requests, from system commands to handling file transfer processes. A use-case for this model would be supposed to be in a computer lab of a college and the professor has a document on the server machine. The professor needs the entire student to receive that document in the machine that they are using in the lab. This can be done by emailing all the students the copy of that document or by sharing it using a USB, both of which are tedious processes for sharing a file with a large group of people. By using this client-server model, all the computers in that network will be able to get the file without any hesitation. This server will be able to serve more than 200 clients at the same time. This is the greatest advantage of this model.

References

Bamane A, Bhoyar P, Dugar A & Antony L (2012):“Enhanced Chat Application” Global Journal of Computer Science and Technology Network, Web & Security”: Volume 12 Issue 11 Version 1.0 June 2012 Online ISSN: 0975-4172 & Print ISSN: 0975-4350.

Huang w (1996): “Automated computer of Chengdu Institute” Meteorology Journal, 1996, 11, 43.

BP, Gowda P , Poojary R & Nayak P (2023): “Review Paper on Socket Programming”: International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 3, Issue 2, January 2023.

Kadhim D & Jasim M (2014): “A New Approach of Designing a Client/Server Model with Smart Screen”: International Journal of Computer Applications (0975 – 8887) Volume 94 – No 5, May 2014

Karan , Krithika B, Saurabh M, Srishti B & Manikandan K (2018): “End to End Multi Client Encrypted Windows Chat System”, International Journal of Engineering Research & Technology (IJERT): ISSN: 2278-0181, IJERTV7IS030178 Vol. 7 Issue 03, March-2018

Kalita L (2014): “Socket Programming”: International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, 4802-4807

Kumar S (2019) : “A REVIEW ON CLIENT-SERVER BASED APPLICATIONS AND RESEARCH OPPORTUNITY” : International Journal of Recent Scientific Research Vol. 10, Issue, 07(H), pp. 33857-33862, July, 2019.

Kurose J & Keith (2014): “Computer Networking: A TopDown Approach featuring the Internet”.

Puli, Srilakshmi, Devada , VenkatSai , Thota G & Tadala T (2020): "BROADCASTING CHAT SERVER", International Journal of Research and Analytical Reviews (IJRAR), IJRAR March 2020, Volume 7, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138).

Ramya D & Rathinam. D (2018): "A REVIEW PAPER ON CRYPTOGRAPHY AND NETWORK SECURITY": International Journal of Pure and Applied Mathematics Volume 119 No. 17 2018, ISSN: 1279-1284.

Rolou M , Cordova R , Sudramurthy B & Halibas A (2017): "Design and Implementation of Client-Server Based Application using Socket Programming in a Distributed Computing Environment": 2017 IEEE International Conference on Computational Intelligence and Computing Research, 978-1-5090-6621-6/17/$31.00 ©2017 IEEE.

Shah M , Servar G & Tomer U (2022): "Realtime Chat Application using Client-Server Architecture", International Journal for Research in Applied Science & Engineering Technology: (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022.

Singh A, Sharma A & Kumar A (2011):" Multi-Client Multi-Instance Secured Distributed File Server", International Journal of Computer Applications (0975 – 8887) Volume 19– No.9, April 2011

Srivastava H, Sinha R & Gupta S (2013): "Implementation of Socket Programming and RMI Using Simulating Environment", International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May-2013 ISSN 2229-5518.

Takawale S, Rupesh & Jaiswal C (2018): "Multi-Client Server Communication Enhancement through Intranet ", International Journal for Research in Applied Science & Engineering Technology (IJRASET)", ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022.

Vikas BO & Karthik GR (2020): "Multi-User Chat Application using Client Server Architecture", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, © 2020 IJSRCSEIT | Volume 6 | Issue 3 | ISSN: 2456-3307.

Xue M & Zhu C (2009): "The Socket Programming and Software Design for Communication Based on Client/Server": Pacific-Asia Conference on Circuits, Communications and Systems, PACCS 2009