iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Network programming in Java using Socket

Sweety Sen[1], Sonali Samanta[1]
B.Tech in Information Technology, Dronacharya College of Engineering, Gurgaon, India

Abstract: ***This paper describes about Network programming using java. The Network programming is similar to socket programming or Client-Server programming. Where Socket programming is important to understand how internet based interprocess communication work. In this we describe about different types of socket used in interprocess communicate on. Network programming basically uses the Client Server model. In Client-Server programming there are two different programs or process, one which initiates communication called Client process and other who is waiting for communication to start called Server process. In this paper we also focus on the Secure Socket Layer needed for security purpose.***
Keywords— Network programming, java, socket, Client-Server, Secure socket layer

I. INTRODUCTION

In today's world Internet is used by each and everybody. The internet is all about connecting machine together and communication. This is where Network Programming comes. Network programming allows Interprocess Communication. It means it involves writing computer programs that communicate with other program across a computer network.
Network programming uses Client-Server model,so network programming is also Client-Server Programming. Where one program start the communication called client process or program and other who is waiting for communication to start called the server process or program. In the simplest case, Client program sends the request to the server. Server sends the response. Client then obtains the data from the server and displays it. Complex clients filter and reorganize data, repeatedly obtain changed data, send data to other people , and allows real time chatting, multiplayer gaming. Complex servers often do a lot of processing on the data before answering the question. Network programming makes use of socket for interprocess communication. Where socket act as the endpoint of the interprocess communication. Here sockets can also be termed as network socket or Internet socket since communication between computers is based on Internet protocol. So Network programming is also Socket Programming.

A. Client Server Model

Network programming uses Client-Server Model. Client program or process initiates the communication and servers are program or process who waits for communication to start. Some time program may be both a client and server. Here Client sends the request to which the server sends the response. Typically we have single server multiple client's model. The server does not need to know anything about client even that it exists or not. The client should always know something about the server atleast where it is located. The IP and Port number of the server is generally well known. So the client knows where to send the request. In Contrast the port number on client side is generally allocated automatically by the Kernel.

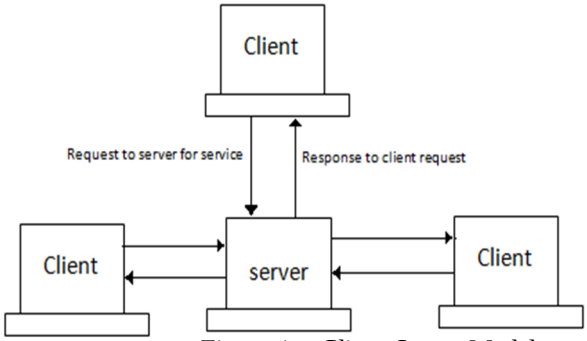


Figure 1: - Client-Server Model

In Client Server Program there are two principal programming models. They are Iterative Server model and Concurrent Server Model.

1) *Iterative Server Model:* In Iterative Server Model Only one request is processed at a time. In this listener and server portions of the application coexist and run as part of the same task. The server application holds the socket until all application processing has completed. In this client has to wait until server does not respond. It is Easy to build and it is mostly used when request can be completed in a small time. The problem with this client server model it causes unnecessary delay.

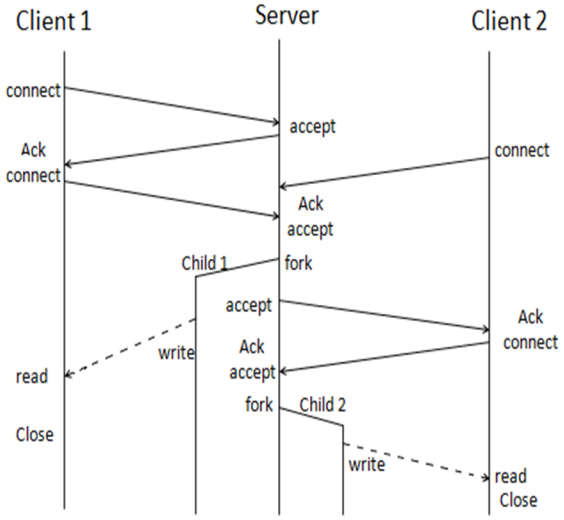


Figure 2:-Iterative server Model working

Algorithm Iterative Server

```
begin
Create Socket
bind to well Know port
while(1)
{
  read request from client
 send reply to client
 exit
}
end
```

Figure 3:-Iterative server using Socket

A typical Iterative server using socket is shown in Figure.3.In this first socket is created. Then bind the socket to well-known port. When the request from the client has been received the server sends the reply to the client. No new request will be processed until the previous request is processed. After request is processed exit and server accept new request and follows the same procedure.

2) *Concurrent Server Model:* In Concurrent Server Model many request are processed at same time. In this the server role is to listen for service request from different host or clients. Once the request has been received the servers fork the child process to handle it and server goes back to listening to other request. In this new request can be processed while other request still being served. It gives better performance than Iterative server model. The disadvantage of this model is it is difficult to design and build.

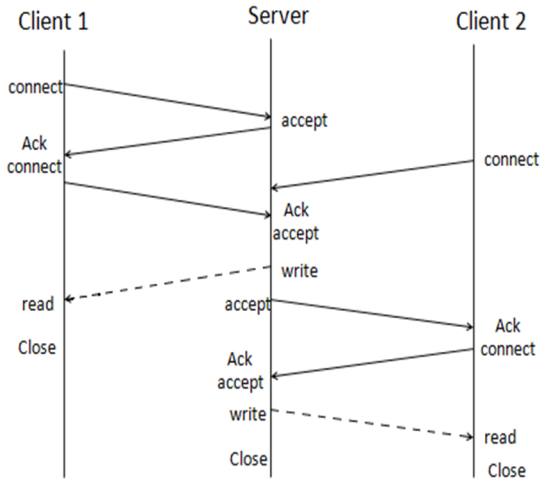


Figure 4:-Concurrent server model working

Algorithm Concurrent Server

```
begin
        create socket
        bind to well know port while (1)
        {
            read request from client
            if(fork()==0)
            {//child Serves request from client
          exit }
          else{ //parent

        }   }
end
```

Figure 5:-Concurrent Server Using Socket

A typical Concurrent server using socket is shown in Figure.5.In which first socket is created. Then bind the socket to well-known port. When the request from the client has been received the network connection is setup. The server then forks a child process to serve the request from client and it goes back to listen to other client request. Here new request is accepted even if previous one is still remaining to be served.

B. Socket Programming

Network programming makes use of socket for Interprocess Communication. Due to which Network programming is also termed as socket programming. In Socket programming using Java, BSD style Socket to Interface with TCP/IP services is used. BSD Socket Interface provides facilities for Interprocess Communication. BSD Socket Interface Supports different domain, the UNIX Domain, the Internet Domain and the NS Domain. Java Basically supports the Internet Domain to maintain cross platform. In Internet Domain, the BSD Socket Interface is built on the top of either TCP/IP or UDP/IP or the raw Socket. Socket Programming is important to understand how internet based interprocess communication work but not at the level program developed but at a higher level that is compiled to set of Socket Programs. Before going in detail about Socket let's focus on the Communication API.

1) Communication API: Communication APIs enable access to services for local and remote communication. They are

present in computing system at different level. There are basically three levels High level Communication APIs, Low Level Communication APIs and Basic Level Communication APIs.

High Level Communication APIs are typically present in the User Space which provides programming language approach to middleware communication service.

Low level APIs are present in kernel space.

The Basic level Communication APIs are usually located between the user space and Kernel space. It usually allows application programs to gain services for Interprocess communication, network communication and device - communication. At this Basic Level Communication Sockets are present which is used for Interprocess Communication i.e for Network Programming.

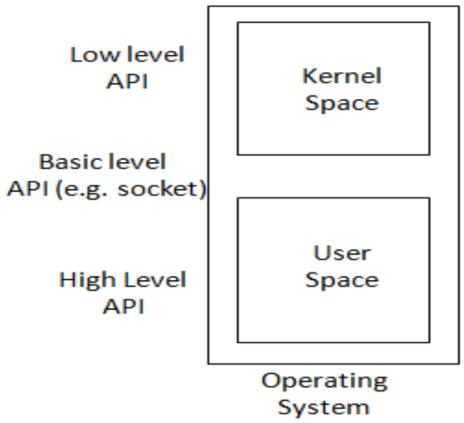


Figure 6:-levels of Communication APIs

2) *Sockets:* Socket is an Interface between the application and the network. Where Application create socket and socket type dictates style of communication i.e. connectionless or connection oriented. Socket basically acts as endpoint of Interprocess communication. Sockets can also be termed as network socket or Internet socket since communication between computers is based on Internet protocol. Sockets are a means of using IP to communicate between machines. Socket is one major feature that allows Java to interoperate with legacy system by just talking to existing server using predefined protocol. Socket also has some transparency that allows kernel to redirect output of one process to the input of another machine. Function Call in Socket API are divided into three categories.

a) *Managing Connection:* In this we have socket(), bind(), listen(), connect() and accept(). Where socket() call is used to create a Socket. An application opens a socket by using the socket() call. bind() call is used to bind a socket to local IP address and port. listen() call is used in passive open .it is used by server to announce its objective of receiving incoming connection request. connect() call is used to start connection to another socket. accept() call is used to accept a new connection.

b) *Sending and Receiving Data:* In this we have send(), sendto(), recv(), recvfrom() and etc. Where send() call sends data on a connected socket. sendto() call is used to send the datagram to another UDP socket. recv() call is used to receive message from socket. recvfrom() is used to read a datagram from a UDP socket.

c) *Managing Endpoint Characteristics:-* In this we have Close() function which is basically used to close the socket i.e. shutdown the connection. Sockets are divided into different types. They are distinguished between TCP Client Socket, TCP Server Socket and UDP Socket.

d) *TCP Socket:* TCP Socket is that socket which is used for more reliable connection. They perform more reliable and in-order delivery of the packets. They are used when there is Connection oriented Communication.TCP socket is also called Stream Socket. TCP Socket supports out of bound data transmission. In TCP Socket there is TCP Client Socket and TCP Server Socket.

i. *Client Socket:* The Client Program or process basically uses the TCP Client Socket. Client Socket are also just called Sockets. To implement the Client Socket the Client program makes use of java.net.Socket class present in java.net package. The java.net.Socket class is used for doing client side TCP operations. Client Socket acts as an endpoint for communication between

two machines. The Socket class contains constructor that is used to create stream socket that connects to specific port number associated with exactly one host.TCP Client Socket encapsulates a java.io.InputStream and java.io.OutputStream.

Client program written using Client Socket will have following life cycle:-

1) Creates a stream socket using constructor Socket(String host, int port).
2) Sockets try to connect to remote host.
3) Then, Input stream and output stream are used by client and server to send data to each other. Both Client and server agree upon handshaking before sending data.
4) When communication is over or data transfer is complete one or both side close the connection.

ii. *Server Socket:* The Server Program or process basically uses the TCP Server Socket. To implement server sockets java.net.ServerSocket class is used which is present in the java.net package. The java.net.ServerSocket class is used for doing server side TCP operations. It waits for request to come over the network and then perform operation based on request. Once ServerSocket has setup connection the server uses Socket object to send data to client. The ServerSocket Class contains Constructor that is used to create new ServerSocket Object on a particular local port and also contains method like accept() to listen for connection on a specified port.

Server program written using Server Socket will have following life cycle:-

1) Creates a ServerSocket on a particular port using constructor ServerSocket().
2) Listen for connection to be made to this ServerSocket by using accept() method. This method waits till client connects to server and then returns the Socket object.
3) Then, Input stream and output stream are used by client and server to send data to each other. Sever hear the client using input stream. Server talk to client using output stream.
4) Both Client and server agree upon handshaking before sending data.
5) When communication is over or data transfer is complete one or both side close the connection.
6) The Server then returns back to second step and waits for next connection.

e) UDP Socket: UDP socket uses User Datagram Protocol (UDP) alternative protocol for sending data. UDP provides connectionless communication. The connection here would be unreliable. It does not have starting handshaking phase and here data received would not be in order or lost. Doing socket programming using UDP no streams are attached to socket.UDP socket is also called as DatagramSocket. Datagram Socket is opened to send and receive DatagramPacket. To implement the Datagram Socket the program make use of java.net.DatagramSocket Class which is present in java.netpackage.DatagramSocket class contains Constructor that is used to construct a datagram socket and bind it to the specified port on the local host machine It also contains method like receive() and send() to receive and send the datagram packets. In this Socket used by client and the server are almost same only thing they differ in whether they use anonymous or well-known port. So there are no special socket class for server.

Client program written using Datagram Socket will have following life cycle:-

1) Creates a Datagram socket using constructor DatagramSocket().
2) Translate hostname to IP address.
3) Create the Datagram Packet using DatagramPacket (Data,Data.length,IP,Port)
4) Send datagram Packet to server using send().
5) Read datagram Packet from Server using receive().
6) When communication is over or data transfer is complete close the connection.

Server program written using Datagram Socket will have following life cycle:-

1) Creates a DatagramSocket on a particular port using DatagramSocket(int port).
2) Read Datagram Packet from client using receive().
3) Get IP address and port no. of client.
4) Create the Datagram Packet using DatagramPacket (Data,Data.length,IP,Port)
5) Send datagram Packet to client using send().
6) The Server then returns back to second step and waits for next connection.

C. Secure Sockets

Security is important on Internet and people want authentication, integration and privacy. Secure Socket layer(SSL) protocol basically helps in achieving this security goals. SSL was developed by Netscape as security measure for web server and web browser. At present SSL is largely used in intranets and also in many public internets. SSL has become de facto standard for providing secure e-commerce transaction over the web.

SSL allows web browser to authenticate the web server. SSL requires web server to have digital certificate on it for SSL connection to be made.SSL is mostly used as Secure transport below HTTP. Secure hypertext transfer protocol i.e. HTTPs is a http who is using the SSL.SSL basically sit on top of the TCP i.e. it lays between TCP and other upper layer can be seen in figure. Developer takes advantage of this by replacing all TCP socket call with the new SSL calls.

SSL consist of two phases: Handshake and data transfer. During handshake phase Client and Server use a public key encryption Algorithm to determine secret key parameters. In this phase the communicating parties optionally authenticate each other and then exchange the session key. During Data transfer phase both sides use the Secret key to encrypt the data transmission.

SSL is an Asymmetric protocol. It differentiates between a client and server. The SSL handshake sequence may vary depending on whether the RSA or Diffie Hellman key exchange is used. Mostly SSL session make use of RSA key exchange algorithm with only server authenticated. Here client authentication is optional and is omitted in most cases.

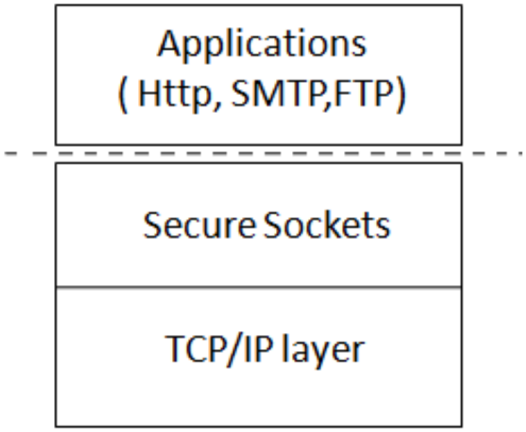


Figure 7:- SSL above Transport Layer

1) *Creating Secure Client Socket:* For creating Secure Client Socket instead of creating or constructing java.net.Socket object with a constructor, use javax.net.ssl.SSLSocketFactory to create Client socket using its createSocket() method. SSLSocketFactory is an abstract class that follows the abstract factory design pattern: public abstract class SSLSocketFactory extends SocketFactory

Since SSLSocketFactory is itself abstract you get an instance of it by invoking the static SSLSocketFactory.getDefault() method. Once reference to the factory is done, then createSocket() methods to build an SSLSocket. The Socket that createSocket() method return will be javax.net.ssl.SSLSocket a subclass of java.net.socket. Once secure socket is created use it like any other socket through its getInputStream(), getOutputStream() and other methods.

2) *Creating Secure Server Socket:* For creating Secure Server Socket instead of creating or constructing java.net.ServerSocket object with a constructor, use javax.net.SSLServerSocket to create Server socket. Like SSL Socket all the constructors in this class are protected.Like SSLSocket, instance of SSLServerSocket are created by an abstract factory class javax.net.SSLServerSocketFactory. Also like SSLSocketFactory an instance of SSLServerSocketFactory is returned by a static SSLServerSocketFactory.getDefault() method. Like SSLSocketFactory, SSLServerSocketFactory has CreateServerSocket() method that returns instance of SSLServerSocket. The factory that SSLServerSocket Factory.getDefault() returns generally only support server authentication. It does not support encryption. To get encryption server side secure socket requires more initialization and setup.

3) **Secure Socket Benefit:** Secure Socket Layer Provides the following Benefits:-
   a) *Authentication:* - It ensures that the end systems are the same systems that they say they are.

b) *Message privacy:* - It ensures that the data exchanged is not viewed by other person other then receiver even if it is intercepted by unauthorized party.

c) *Integrity:* - It ensures that the data is not modified or tampered over the Internet.

IV. CONCLUSIONS

This paper on Network programming in java describe in detail about concepts used in network programming in java. It describes about Java network Programming and its application. Network programming is Client server programming, so it describe about different client server model. It also explains about Socket programming, different types of sockets used like TCP or UDP. From security perspective we have SSL in network programming using java. This paper gives details about SSL, Creating secure Socket and its benefit. Finally there is comparison of Network programming using Java and Network programming using C, provided the advantages and disadvantages of both. Network programming using Java have lots of advantage due to which it is preferred.

REFERENCES

[1] Mengjou Lin, Jenwei Hsieh, David H.C.Du, Joseph P.Thomas and James A. MacDonald, ―Distributed Network Computing over Local ATM Networks,‖ IEEE Journal on Selected Areas in Communications, Vol. 13. No. 4, May 1995

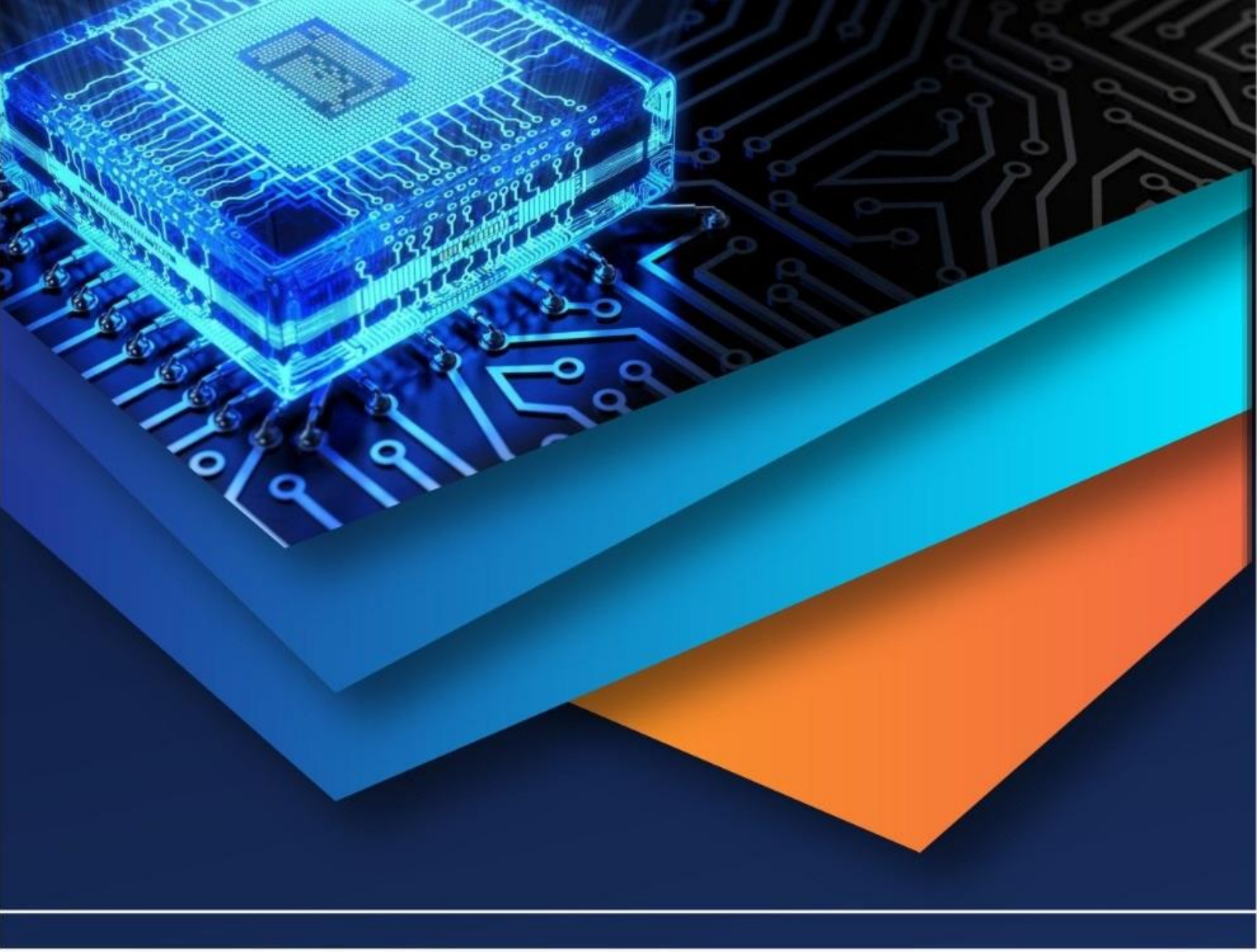
[2] David K. Y. Yau and Simon S. Lam,‖ Migrating Socket – End System Support for Networking with Quality of Service Guarantees,‖IEEE/ACM Transaction on Networking, Vol. 6,No. 6 , December 1998.

[3] Stefan Bocking,‖Socket++: A Uniform Application Programming Interface for Basic-Level Communication Services,‖ IEEE Communication Magazine December 1996.

[4] Mark A. Holliday,J. Traynham Houston and E. matthew Jones,‖From Socket and RMI to Web Services,‖SIGCSE'08 , Portland, Oregon,USA. March 12-15,2008.

[5] S.Kwong, K.T. Ng and W.N. Chau,‖ Design and Implementation of Remote Procedure Call Based on ISO/OSI Reference Model,‖IEEE TENCON Bejing,1993.

[6] Mattew Cook and Syed(shawon)M. Rahman,‖ Java and C/C++ language feature in terms of Network Programming,‖ 2005.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY