International Journal of Modern Engineering Research (IJMER)

www.ijmer.com Vol.1, Issue1, pp-162-165 ISSN: 2249-6645

Communication Enhancement Using Socket Programming

Monika Sharma [1], Prashant Soni [2]

* (Faculty, AIIT, Amity University, India)
*(Student, Amity University, India)

Abstract

As a java programmer one might face networking using socket programming. A network-based system consists of a server, client, and a media for communication and for such communication we use "Sockets".

Sockets are interfaces that can "plug into" each other over a network. Once so "plugged in", the programs so connected communicate. A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. Such programming is called socket programming.

The java.net package of "JAVA" provides two classes—Socket and ServerSocket, that implement the client side of the connection and the server side of the connection, respectively.

Keywords –**Client, Java.net, Sockets, Socket Programming, Socket (Class), Server and ServerSocket (Class).**

1. Introduction

There are many codes developed for socket programming which were previously being implemented but some drawbacks like slow speed of execution, they were not user friendly, uneven flow of data, etc hindered their best possible use and I have tried to patch them.

As far as security is concerned, the 'class' file of java plays its role and thus provide much security for code on being copied to any other operating system.

1.1 Java TCP Programming

The programming model of TCP communication in Java, rely completely on the sockets and ports. Because TCP is a stream protocol, it allows to send arbitrary amount of data rather rely on class to encapsulate data within TCP packets.

1.2 Sockets

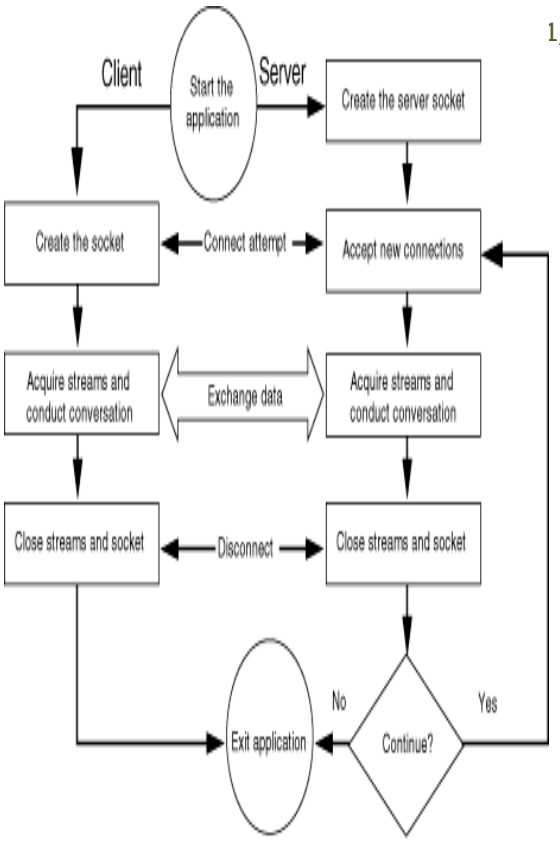
Java programs communicate through a programming abstraction called socket. A socket is one end-point of a two-way communication link between two computers (or programs) running on a network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. Mostly, applications do not care how data is actually transmitted in the network. Applications identify the address of the peer entity and then use sockets interface to read and write data from and to the peer.

Sockets include the implementation of network and transport layer protocols providing applications with a simple read/write interface. Because sockets are just programming abstractions for network protocols, the other side of the connection does not have to use them. Sockets don't use any additional communication mechanism other than that provided by the encapsulated protocol [1].

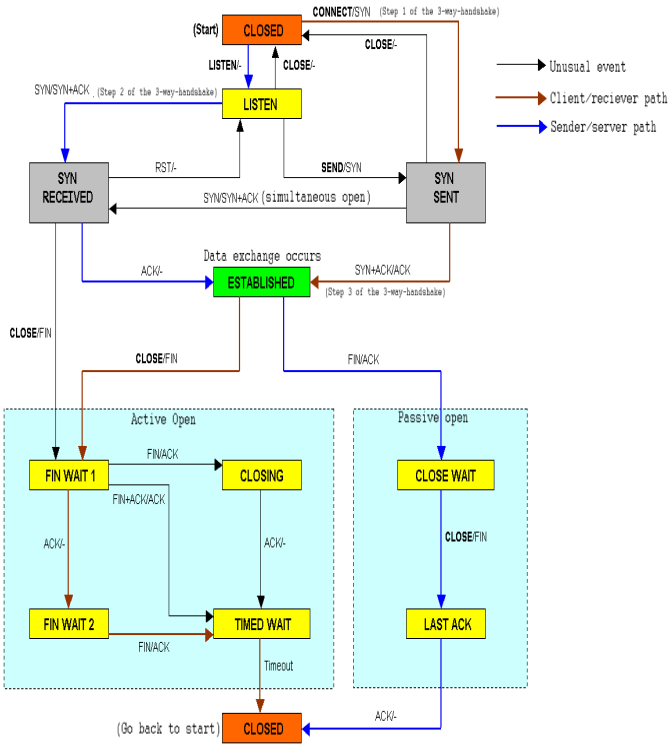
1.3 Ports

The mechanism, commonly used by network protocols, and particularly by the Internet transport layer protocols, is port addressing [2]. For example, tcp:23 port number is assigned for FTP, tcp:80 for HTTP, etc.

1, Issue1, pp-162-165 ISSN: 2249-6645



1.5.1 This figure illustrates how can client and server interact with each other



1.5.2 This figure illustrates how a three way and a four way handshake takes place.

Typically, integer numbers are used to identify different ports. In order to contact a network service, it is therefore necessary to provide both the IP address of its host, as well as port number it is using.

1.4 Connection Establishment

TCP uses a 3 way handshake. A Server makes a passive open by call bind( ). The Client initiates an active open by calling connect( ). For the connection to be established, the client sends a **SYN** packet to the server. The Server replies with **SYN/ACK** packet and finally the client replies with **ACK** packet [3].

1.5 Connection Termination

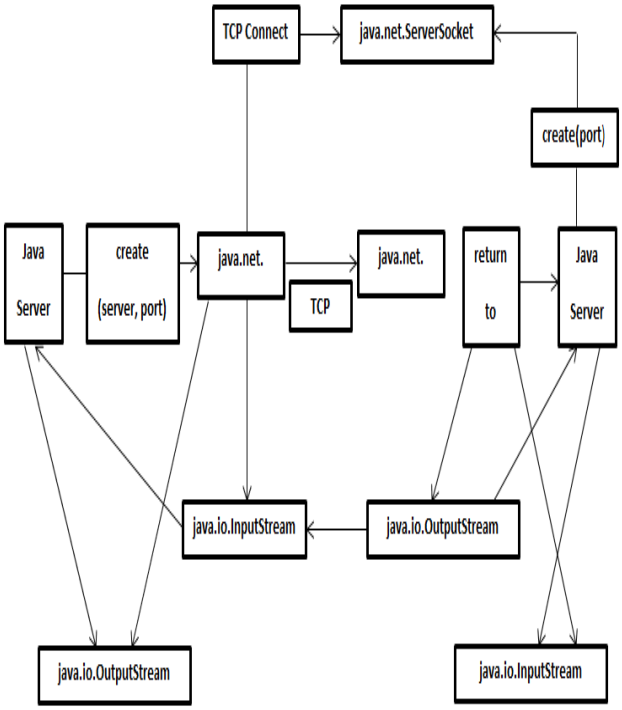
TCP uses a 4 way handshake to close the connection. When an endpoint wants to close, it sends out a **FIN** packet, the other side then replies with an **ACK** packet [3].

A connection is "half-open" when one side has close the connection. The other side is still free to send. A situation can occur where one side closes the connection and then reopens it immediately, so any lost packets that now arrive will not belong to this "new" connection and thus TCP we need to insure these packets do not get mixed in with the new packets. So the "TIME WAIT" state allows a connection to remain open long enough for such packets to be removed from the network. This state usually lasts for about 2 times the "round-trip", some implementation hardcode the default value to be anywhere from 30 to 120 seconds. We can use the **netstat** utility to see TCP/IP states.

1.6 Package java.net

The Java language supports TCP programming through the **java.net.Socket** and **java.net.ServerSocket** classes. Java clients connect to TCP servers by creating instances of the **java.net.Socket** class. Similarly, Java servers listen for java clients by creating **java.net.ServerSocket** class [3]. Connections are configured through the methods of these two classes. Actual network communications, however, are performed using the **java.io** [4] package streaming classes.

www.ijmer.com Vol.1, Issue1, pp-162-165 ISSN: 2249-6645



1.6.1 The figure illustrates the main operations of the Java streaming socket classes.

The Java TCP server, on the right, accepts TCP connections by creating a **java.net.ServerSocket** instance bound to a particular port.

When a ***ServerSocket*** establishes a connection with a TCP client, it creates a **java.net.Socket** instance encapsulating that connection and returns it to the server program. The **java.net.Socket** object returned is bound to an ephemeral ports number that is different from the one the **ServerSocket** is listening to. The server retrieves the socket's input and output streams and effects communication by implementing some protocol.

On the client side, TCP connections are established through instances of **java.net.Socket** associated with a TCP server on the given host and port. Once the client **Socket** has established a TCP connection, retrieves the socket's input and output stream, effects communication.

2 Interpretation and Implementation

2.1 Sending Data Through A Socket

Once an application program has established a socket, it can use the socket to transmit data.

While sending the data we check that whether the data is null or not. If the data is not null then we may proceed to send data to stream through socket at receiver's end [5].

2.2 Receiving Data through a Socket

For receiving the data also we shall use socket and in same way as data was sent to stream, we will receive it. The data received through socket is read and printed on output screen.

2.3 Client-Side TCP Programming

The **java.net.Socket** of a **java.net** package implements client side of a two-way connection between Java program and another program on the network. By using this class instead of relying on native code, Java program can communicate over network in platform independent fashion [6].

2.3.1 Creating a Socket

In order to establish a connection with a network server one must have the address of the server's host, and port number to which the server is bound. The **java.net.Socket** class provides a constructor, which takes an IP address and port number and attempts to establish a connection. The signature of this constructor is as follows –

Socket(InetAddress, int port) throws IOException;

The constructor returns only after the connection has been established, that is, once the TCP three-way handshake has been completed.

2.4 Server-side TCP Programming

In order to accept network connections a Java program must create an instance of **java.net.ServerSocket**. Server sockets are not directly used to perform any network communication. Instead, they act as factories that create a **java.net.Socket** object for every incoming TCP communication request. Programs create the server socket, bind to a specific port on one or more interfaces, and then invoke the blocking **accept()** method [7].

2.4.1 Creating ServerSocket

The basic **ServerSocket** constructor takes a single argument, the TCP port number used in binding. If the constructor returns without throwing an exception then the server socket has successfully bound

www.ijmer.com Vol.1, Issue1, pp-162-165 ISSN: 2249-6645

to the requested TCP port. The constructor may fail due to an I/O error, or due to a security error. The signature of the constructor is –

ServerSocket(int port) throws IOException,SecurityException;

2.4.2 Accepting Sockets

The main task of server socket is to receive incoming connection requests and generate a **java.net.Socket** object that encapsulates each request. Incoming connections are queued until the program retrieves them one at a time by invoking the accept() method. The accept() method takes no arguments, and returns the next connection in the queue.

2.5 Terminating Server Socket

A server socket may be terminated simply by invoking the no – argument close() method.

Closing the server socket will not affect connections that have already been returned by accept () invocation. If the accept () method is invoked on a closed socket then a **java.net.Socket** exception will be thrown with a message indicating that the socket has been closed. The signature is as follows –

void close() throws IO Exception;

3 Conclusion

Developing network applications is made possible in Java by using sockets, threads, RMI, clustering, and Web services. These technologies allow for the creation of portable, efficient, and maintainable large and complex Internet applications. The java.net package provides a powerful and flexible set of classes for implementing network applications.

Typically, programs running on client machines make requests to programs on a server machine. These involve networking services provided by the transport layer. The most widely used transport protocols on the Internet are TCP (Transmission control Protocol) and UDP (User Datagram Protocol). TCP is a connection-oriented protocol providing a reliable flow of data between two computers. It is used by applications such as the World Wide Web, e-mail, IP, and secure shell.

Sockets provide an interface for programming networks at the transport layer. Using sockets, network communication is very much similar to performing fi le I/O. A socket is an endpoint of a two-way communication link between two programs running on the network. The source and destination IP address, and the port numbers constitute a network socket.

Two key classes from java.net package used in creation of server and client programs are **ServerSocket**, which represents a server socket, and Socket, an instantiation of which performs the actual communication with the client.

4 Future Work

In future, for further optimization, I am trying to implement security constraints on my code like 'Cryptography' for enhanced two way communication.

References

Articles:

[1] http://www.buyya.com/java/Chapter13.pdf

[2] http://www.scribd.com/doc/29858538/Java-TCP-Programming

[3] http://www.pcvr.nl/tcpip/tcp_conn.htm

[4] http://download.oracle.com/javase/1,5.0/docs/api/ (Java API(Application Programming Interface) 5.0)

[5] http://edn.embarcadero.com/article/31995

Books:

[6] Herb Schildt, *Java 2 The Complete Reference* (4th edition, Osborne/MacGraw-Hill,2001)

[7] Kathy Sierra & Bert Bates, *Head First Java* (2nd edition, O'Reilly Media, 2005)