Divya Singh *et al*, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 11, November- 2014, pg. 150-154 **ISSN: 2321-8363**

Study of Data Transmission Using Sockets

Divya Singh[1] Pragun Arora[2] Rakshit[3] Saket Tomar[4]

Senior Lecturer, Department of Computer Science and Engineering, AMITY University Greater Noida [1]

Scholar, Department of Computer Engineering, AMITY University Greater Noida [2][3][4]

Abstract: As a java programmer one might face networking using socket programming. A network-based system contains 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. In a two-way communication link between two programs running on the network socket is the one end point. The connection between a client program and a server program is represented by socket classes. Such programming is known as socket programming.
To implement the client side of the connection and the server side of the connection, the java.net package of "JAVA" provides two classes—Socket and Server Socket, that , respectively.

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

1. Introduction

There are many codes developed for socket programming which were previously being implemented but some drawbacks like execution speed is slow, not user friendly, uneven flow of data, etc hindered their possible use and we have tried to patch them.
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, totally depends on the sockets and ports. .

1.2 Sockets

Socket is a programming abstraction through which java programs communicate . In two-way communication link between two computers (or programs) running on a network socket is a one end-point . A socket is assigned with a port number so that the TCP layer can identify the application that data is destined to be sent. Many applications doesn't care how data is actually transmitted in the net-work. 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 and write interface as sockets are just programming abstractions for network protocols, the other side of the connection doesn't use them. Sockets use communication mechanism which is provided by the encapsulated protocol .

1.3 Ports

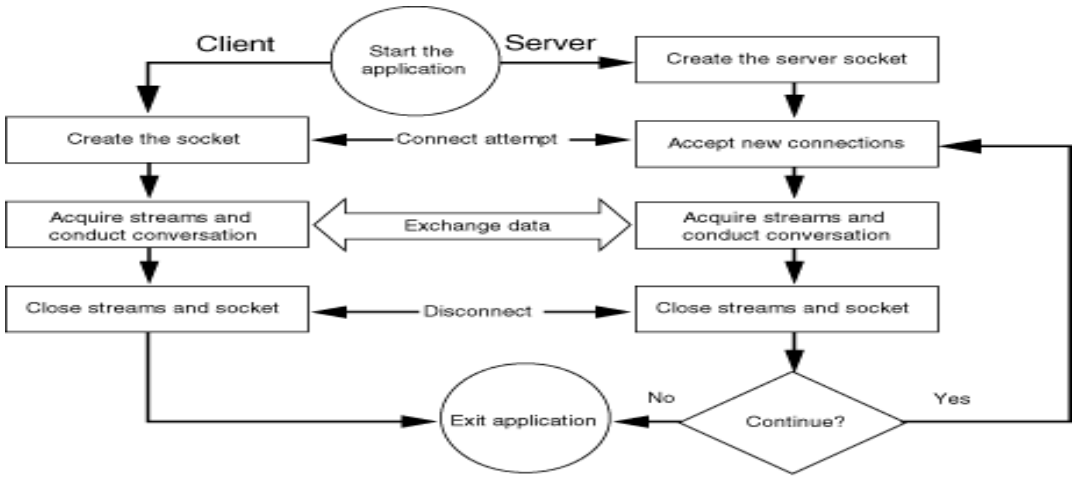
The mechanism, particularly by the Internet transport layer protocols, and commonly used by network protocols,was port addressing . For example,For FTP port number tcp:23 is assigned , for HTTP it is tcp:80, etc.
Typically, integer numbers are used to identify different ports. In order to contact network service, it is necessary to provide the IP address of its host, as well as port number which it is using.

1.4 Connection Establishment

TCP uses a 3 way handshake. By call bind( ) a server makes a passive open. By calling connect( ) the Client initiates an active open. The client sends an SYN packet to the server for the connection to be established. The Server replies with SYN or ACK packet and finally the client replies with the ACK packet .

1.5 Connection Termination

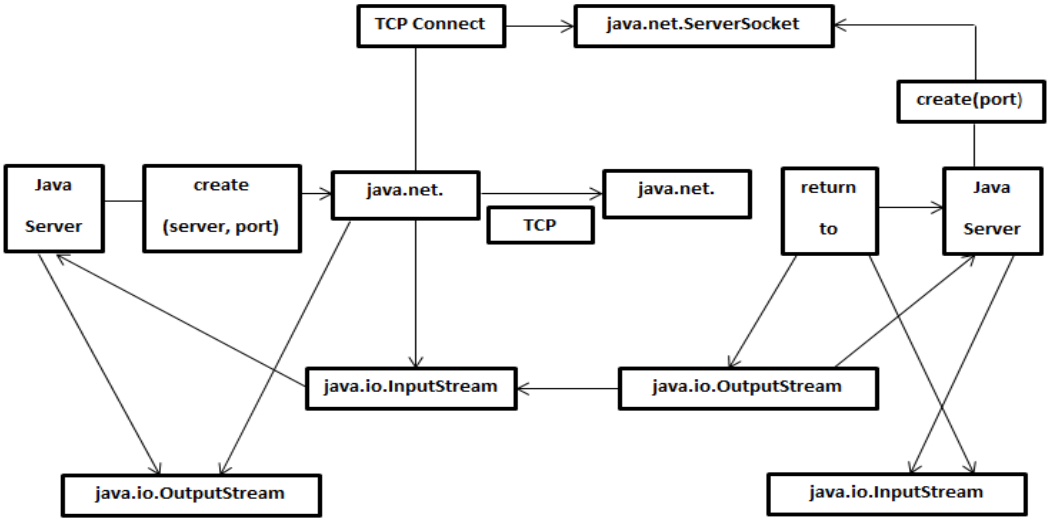
TCP uses a 4 way handshake to close the connection. , FIN packet is sends out when endpoint wants to close , the other side then replies with an ACK packet .
A connection is "half-open" when one side has close the connection and he other side is still free to send. A situation can occur whenever one side closes the connection and then reopens it immediately. So if 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 were allows a connection to remain open long enough for such packets to be removed from the network. This state was usually lasts for about 2 times the "round-trip", some implementation will be hardcode the default value to be anywhere from 30 to 120 seconds. We used the netstat utility to see TCP/IP states.



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

1.6 Package java.net

The Java language supports the TCP programming through the **java.net.Socket** and **java.net.ServerSocket** classes. Java clients were connect to TCP servers by creating objects of the **java.net.Socket** class. Similarly, Java servers were listen for java clients by creating **java.net.ServerSocket** class. Connections were establish through the methods of these two classes.Network communications were performed using the **java.io** package streaming classes.



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

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

2. Interpretation and Implementation

2.1 Sending Data Through A Socket

Once an application program has been establishes a socket, it uses the socket to transmit data.
While sending the data we can check that whether the data is null or not. We may proceed to send data to stream through socket at receiver's end only if the data is not null.

2.2 Receiving Data Through A Socket

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

2.3 Client-Side TCP Programming

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

2.3.1 Creating a Socket

In order to establish a connection with a network server one should have the address of the server's host, and port number to which the server is bounded. The **java.net.Socket** provides a constructor, which takes an port number and IP address as input and attempts to establish a connection. The signature of the constructor is –
Socket(intAddress, int port) throws IOException;
The constructor returns only after the connection has been established, i.e once the TCP three-way handshake which has been completed.

2.4 Server-side TCP Programming

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

2.4.1 Creating ServerSocket

The basic **ServerSocket** constructor takes a single argument as input, the TCP port number which is used in binding. If the constructor doesn't throw an exception then the server socket has successfully bound to the requested TCP port. The constructor may cause failure due to an Input/output error, or due to a security error. The signature of the constructor is –
ServerSocket(int port) throws IOException;

2.4.2 Accepting Sockets

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

2.5 Terminating Server Socket

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

Development of network application is possible in Java by the use of sockets, RMI, clustering,, Web services and threads. Developer can create efficient and portable Internet applications. In java.net package there are a set of classes for the implementation of network applications.
Typically, programs that run on client machines make requests for programs present on the server. This involve networking services which are provided by transport layer. The widely used transport protocols are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). TCP provides reliable data flow between two computers.
Sockets provide an interface for programming net-works at the transport layer. Use of sockets make network communication very similar to performing an file I/O. A socket acts as a endpoint for a two-way communication link between two programs running on the network.

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

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 Program-ming Interface) 5.0)

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

Books:

[6] Herb Schildt, *Java 2 The Complete Refer-ence* (4th edition, Osborne/MacGraw-Hill,2001)
[7] Kathy Sierra & Bert Bates, *Head First Java* (2nd edition, O"Reilly Media, 2005)

