Advanced Java: Networking with Java (Socket Programming)

Nipun Sharma[1] & NavitaDeswal[2]
Information Technology, Dronacharya College of Engineering,
nipun.14457@ggnindia.dronacharya.info; navita.14451@ggnindia.dronacharya.info

Abstract-
Network programming refers to the writing of programs that execute across multiple devices (computers), in which all the devices are connected to each other with a network. A socket is one endpoint of a two-way communication link between two programs running on the network. A client program creates a socket on its end and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end. The client and server can now communicate by writing to and reading from the socket. The Java Net. A socket class represents a socket, and the java.net. The server Socket class provides a mechanism for the server program to listen for clients and establish connections with them. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes--Socket and Server Socket--that implement the client side of the connection and the server side of the connection, respectively. The java.net package provides support for TCP and UDP network protocols.

Keywords – Network programming; socket; classes of socket

I. Introduction

Socket Overview
A network socket is like an electrical socket. The idea applies to network sockets, is that we consider TCP/IP packets and IP addresses rather than electrons and street addresses. Internet Protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination. Transmission Control Protocol (TCP) is a higher-level protocol that manages to robustly string together these packets, sorting and retransmitting them as necessary to reliably transmit your data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

Client or Server
You often hear the term client/server mentioned in the context of networking. A server is anything that has some resource that can be shared. A client is simply any other entity that wants to gain access to a particular server. The server is a permanently available resource, while the client is free to “unplug” after it is has been served. A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

Proxy Servers
A proxy server speaks the client side of a protocol to another server. This is often required when clients have certain restrictions on which servers they can connect to. Thus, a client would connect to a proxy server, which did not have such restrictions, and the proxy server would in

turn communicate to the client. A proxy server has the additional ability to filter certain requests or cache the results of those requests for future use. When a popular web site is being hit by hundreds of users, a proxy server can get the contents of the web server's popular pages once, saving expensive internetwork transfers while providing faster access to those pages to the clients.

II. Java & Network Programming

The Networking Classes & Interfaces

The classes contained in the java.net package are listed here:

Authenticator (Java 2) Inet SocketAddress (Java 2, v1. 4) Socket Impl
Content Handler Jar URL Connection (Java 2) Socket Permission
Datagram Packet Multicast Socket URI (Java 2, v1.4)
Datagram Socket Net Permission URL Class Loader (Java 2)
Datagram Socket Impl Network Interface (Java 2, v1.4) URL
Http URL Connection Password Authentication (Java 2) URL Connection InetAddress

Some of these are to support the new IPv6 addressing scheme. Others provide some added flexibility to the original java.net package. Java 2, version 1.4 also added functionality, such as support for the new I/O classes, to several of the pre-existing networking classes. The java.net package's interfaces are listed here:
Content Handler Factory SocketImpl Factory URL Stream Handler Factory
File Name Map Socket Options Datagram Socket ImplFactory
(Added by Java 2, v1. 3)

Server Socket Class Method

The java.net. Server Socket class is used by server applications to obtain a port and listen for client requests. The ServerSocket class has four constructors:

• *Public Server Socket (int port) throws IO Exception*
Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
• *Public Server Socket (int port, int backlog) throws IO Exception*
Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
• *Public ServerSocket (int port, int backlog, InetAddress address) throws IO Exception*
Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on
• *public Server Socket () throws IO Exception*
Creates an unbound server socket. When using this constructor, use the bind () method when you are ready to bind the server socket

If the Server Socket constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests. Here are some of the common methods of the Server Socket class:
• *publicintget LocalPort ()*
Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.
• *Public Socket accept () throws IO Exception*
Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the set So Timeout ()

method. Otherwise, this method blocks indefinitely

• *Public void set So Timeout (int timeout)*

Sets the time-out value for how long the server socket waits for a client during the accept ().

• *Public void bind (Socket Address host, int backlog)*

Binds the socket to the specified server and port in the Socket Address object. Use this method if you instantiated the Server Socket using the no-argument constructor.

When the Server Socket invokes accept (), the method does not return until a client connects. After a client does connect, the Server Socket creates a new Socket on an unspecified port and returns a reference to this new Socket. A TCP connection now exists between the client and server, and communication can begin.

III. Socket Class Method

The java.net. Socket class represents the socket that both the client and server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept () method. The Socket class has five constructors that a client uses to connect to a server:

• *Public Socket (String host, int port) throws Unknown Host Exception, IO Exception.*

This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.

• *Public Socket (InetAddress host, int port) throws IO Exception*

This method is identical to the previous constructor, except that the host is denoted by an Inet Address object.

• *Public Socket (String host, int port, InetAddress local Address, intlocalPort) throws IOException.*

Connects to the specified host and port, creating a socket on the local host at the specified address and port.

• *Public Socket (InetAddress host, int port, InetAddresslocalAddress, intlocalPort) throws IOException.*

This method is identical to the previous constructor, except that the host is denoted by an InetAddress object instead of a String

• *Public Socket ()*

Creates an unconnected socket. Use the connect () method to connect this socket to a server.

When the Socket constructor returns, it does not simply instantiate a Socket object, but it actually attempts to connect to the specified server and port. Some methods of interest in the Socket class are listed here. Notice that both the client and server have a Socket object, so these methods can be invoked by both the client and server.

• *Public void connect (Socke tAddress host, int timeout) throws IO Exception*

This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.

• *publicInet AddressgetInet Address ()*

This method returns the address of the other computer that this socket is connected to.

• *publicintget Port ()*

Returns the port the socket is bound to on the remote machine.

• *publicintget LocalPort ()*

Returns the port the socket is bound to on the local machine.

IV. InetAddress Class Method

This class represents an Internet Protocol (IP) address. Here are following useful methods which you would need while doing socket programming:

• *staticInet Addressge tBy Name (String host)*

Determines the IP address of a host, given the host's name.

• *String get Host Address ()*
Returns the IP address string in textual presentation.
• *String get Host Name ()*
Gets the host name for this IP address.
• *static InetAddress InetAddressget Local Host ()*
Returns the local host.
• *String To String ()*
Converts this IP address to a String.
• *booleanis Multi cast Address ()*
Returns true if this Internet address is a multicast address. Otherwise, it returns false.

Internet addresses are looked up in a series of hierarchically cached servers. That means that your local computer might know a particular name-to-IP-address mapping automatically, such as for itself and nearby servers. For other names, it may ask a local DNS server for IP address information. If that server doesn't have a particular address, it can go to a remote site and ask for it. This can continue all the way up to the root server, called Inter NIC (internic.net). This process might take a long time, so it is wise to structure your code so that you cache IP address information locally rather than look it up repeatedly.

Conclusion

This paper describes the details about sockets, ports, socket programming Network programming makes use of socket for in terprocess communication between hosts where sockets act as the endpoint of the in terprocess communication. Here sockets can also be termed as network socket or Internet socket for communication between computers is based on Internet protocol. So Network programming is also Socket Programming.

Acknowledgement

Much thanks to our guide for his constructive criticism, and assistance towards the successful completion of this research work.

References

1. MC Graw Hill "The Complete Reference: Java 2" Fifth Edition.
2. Brose, G., Vogel, A., & Duddy, K. (2001). *Java programming with CORBA: advanced techniques for building distributed applications* (Vol. 6). John Wiley & Sons.
3. Reilly, D., Reilly, M., Harold, E. R., Stevens, W. R., Fenner, B., & Rudoff, A. M. (2002). Java network programming and distributed computing. *Addison-Wesley*.
4. Steflik, D., & Sridharan, P. (2000). *advanced JAVA Networking*. Prentice Hall Professional.
5. Farris, R. D., Flaherty, S. J., & Goodman, W. D. (2000). *U.S. Patent No. 6,167,253*. Washington, DC: U.S. Patent and Trademark Office.