**UNIT 04: Socket and Network**

In today’s computing world, business data is an expensive resource that needs to be shared. To facilitate this, robust networks have come into existence. Networks allow expensive resources to be shared.

Any programming environment must provide standard techniques to permit its applications to communicate across a network. Java communication is built on standard Client/Server architecture.

A Server must have a port number on which some software is a listener/ talker. This software is constantly listening for Client requests. When the software hears a Client request it becomes a talker and replies to the Client appropriately. Imagine multiple servers on a network. For this system to work properly, each listener software must clearly and unambiguously recognize that a Client is making a specific request to it. Additionally, when the listener becomes a talker it must clearly and unambiguously speak back only to the Client that spoke to it in the first place.

This necessitates that both the Clients and the Servers on the network must be uniquely or identifiable. This is where TCP/IP comes into the picture. Java provides a Socket based API that shields a programmer from a lot of low level code writing while making it pretty simple to create network based applications.

**Socket**

A network socket is a lot like an electric socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands standard networking protocol can plug-in to the socket and communicate.

Internet Protocol [IP] is a low level routing protocol that breaks data into small packets and sends them to an address across a network. This does not guarantee the delivery of the packets to the destination.

Transmission Control Protocol [TCP] is a higher-level protocol that manages to string rar these packets, sorting and retransmitting them as necessary, to transmit robustly and data to an address across a network. A third protocol, User Datagram Protocol (UDP) does a similar job as TCP and can be used directly to support fast transport of packets from source to destination.

**Client/Server Networking**

A Server is an entity me resource that can be shared. A Client is simply another entity that wants to gain access that has those resources. In à networking environment, a Socket on the server allows a client to plug-in and access a server's resources. Server Sockets allow computers to single handedly serve different clients different kinds of information.

This feat is managed by the introduction of a port, which is a numbered socket on a particular machine. A Server process is said to listen at 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, the server's listening process must be multi-thread.

**Proxy Servers**

A proxy server is a software that runs on Server that speaks the language of Clients. This software hides the actual Server from the Clients that communicate with it. This is required when Clients have 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 service the client while communicating back to back with a Server on which the resources required by the Client reside.

A proxy server has the additional ability to filter Client requests or cache the results of those requests for future use. A caching proxy HTTP server can help reduce the bandwidth demands on a local network's connection to the Internet. When a popular website is being hit by hundreds of users, a proxy server can get the contents of the web server's popular pages once and save these in its cache, thus saving expensive internet data transfers while providing fast access to the same pages to its client.

**Internet Addressing**

The Internet is really the world's largest network and it is growing bigger every day. Every computer on the Internet has a unique IP address.

An Internet address is a number that uniquely identifies each computer on the internet like any other network. There are 32 bits in an IP address and they are often referred to

sequence of four numbers between 0 and 255 separated by periods . [remember the XXX.XXX. XXX. XXX

**Domain Naming Service [DNS]**

addresses as a series of numbers separated by a period. For example: It is hard to imagine seeing http://192.9.9.1/ at the bottom of an advertisement. Thankfully, a clearing house exists for a parallel hierarchy of names bound to each of these unique numbers. It is called the Domain Naming Service [DNS]. Just as the four numbers of an lP address describe a network hierarchy from left to right, the name of an Internet address, called its [domain name] describes a machine's location in a namespace, from right to left.

For example: www.starwave.com is in the COM domain which is reserved for U.S. based commercial sites, is called starwave and www is the name of the specific computer that is Starwave's physical Web server. The www corresponds to the rightmost number in the underlying [and equivalent] IP address.

**Using UDP Connection**

To make the concepts of Network programming fall into place a simple example of network programming using the UDP protocol will help. In this example there are two .class files created. One will run on a Client and the other will run on a Server. The Client and Server should be connected via an Ethernet network.

The example is really a simple one, the Server broadcasts a message at one second intervals to a computer on which the Client application is running. The Client listens on a specific port and receives the Server's broadcast. Then displays what the Server has sent on its VDU. For this exchange to take place the Server application will have to know the Name of the Client computer.

The Client application will have to know the:

* Port number on which to listen for the Server's broadcasts.
* Create a ket and lin to the Server
* A Accept the packets sent from the Server
* Process the packets sent from the server
* Display the contents of the packets sent from the Server on the Client's VDU

In this case, since the physical existence of a network is not the Server known, both the Client .class file and .class file will be run on the same computer. Hence, the Server application the IP address of the Client as localhost [i.e. 127.0.0.1], which internationally points to aral cOmputer. What this means is the Client and Server applications both run on the sane computer but in different Root Consoles.

Using TCP Connection

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

java.net.Socket represents a socket and java.net.ServerSocket provides a mechanism for the server program to listen for clients and establish connections with them.

The following is what happens while establishing a TCP connection between two computers using sockets:

* The server instantiates a ServerSocket object, denoting which port number communication is to occur on
* The server invokes ServerSocket.accept), which waits until a client connects to the server on the given port
* After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to
* The constructor of Socket attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server
* On the server side, accept() returns a reference to a new socket on the server that is connected to the client's socket
* After the connections are established, communication can occur using l/O streams. Eachn socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream and the client’s InputStream is connected to the server’s OutputStream.
* TCP is a two way communication protocol, at the same time. so data can be sent across both the streams at the