**TCP/IP socket in Java**

1. **Networks, Packets and Protocols:**

* Computer network include interconnected machines called hosts and routers. A communication channel is a way of sending data from one computer to another through a wired (Ethernet) wireless (WiFi).
* Routers are essential since connecting a host directly to every other host is impractical. Rather, a few hosts link to a router, which in turn connects to other routers, and so on, to form the network. Routers are not explicitly interacted with by programs that exchange data across the network, and they are normally unaware of their presence.
* By facts, we mean byte sequences that are created and interpreted by software. These byte sequences are known as packets in the form of computer networks. A packet provides control information that the network uses to perform its functions, as well as user data in some cases. Such control information is used by routers to determine how to forward each packet.
* A protocol is a set of rules for how communications systems share packets and what they say. A protocol specifies how packets are organized. A protocol is normally created to address a complex problem with limited resources.
* Implementing a useful network necessitates the resolution of several issues. Different protocols are designed to address different sets of problems and keep things simple and modular. TCP/IP is an example of a protocol set, which is a series of solutions. That is the collection of protocols that are used on the Internet, but it can also be used in standalone private networks.
* The Internet Protocol (IP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP) are the three major protocols in the TCP/IP suite.
* Layers are used to arrange protocols; TCP/IP and nearly all other protocol suites are arranged this way. The sockets API allows applications to use the resources offered by UDP and TCP. The arrow represents data flow from the program, through TCP and IP implementations, through the network, and back through IP and TCP implementations at the other end.
* The underlying networking networks make up the bottom layer of TCP/IP. The Internet Protocol is the only network layer protocol in the TCP/IP suite, and it addresses the dilemma of making any two hosts' sequence of channels and routers appear as a single host-to-host channel.
* The Internet Protocol uses a datagram service, in which each packet is managed and distributed separately by the network, much as letters or parcels sent by the postal system. To make this work, each IP packet must have the destination's address, much as any package you send is addressed to someone.
* The transmission layer is the layer above IP. There are two protocols available: TCP and UDP. Each builds on IP's service, but in various ways to provide different types of transport that are used by application protocols with different requirements. TCP and UDP share a single function: addressing.
* To get a packet to a specific application program, maybe one of many on the same host, a finer granularity of addressing is needed. TCP and UDP both use addresses, known as port numbers, to label programs on hosts. TCP and UDP are known as end-to-end transport protocols because they transport data from one application to the next.
* TCP is designed to track and recover from IP's host-to-host channel losses, duplications, and other faults. TCP offers a secure byte-stream channel, removing the need for programs to deal with these issues. It is a link-oriented protocol, which means that before two programs can communicate, they must first create a TCP connection, which entails exchanging handshake messages between the two communicating computers' TCP implementations. In several cases, using TCP is equivalent to using file input/output (I/O).
* A fair model of communication over a TCP network is a file that is written by one program and read by another. UDP does not try to recover from IP errors; instead, it expands IP's best-effort datagram support to run between application programs rather than between hosts. As a result, UDP implementations must be equipped to cope with losses and reordering…

1. **About Address:**

* Before a program may connect with another program, it must first locate the other program to the network. In TCP/IP, a program is identified by two pieces of information: an Internet address, which is used by IP, and a port number, which is an additional address interpreted by the transport protocol (TCP or UDP).
* Binary numbers are used to represent internet addresses. They're available in two flavors, which correspond to the two uniform implementations of the Internet Protocol. Version 4 (IPv4) is the most common; version 6 is the other (IPv6). IPv4 addresses are 32 bits long; because this is only enough to identify about 4 billion distinct destinations. IPv6 addresses are 128 bits long.
* The dotted-quad notation is used to write IPv4 addresses, which are composed of four decimal numbers separated by intervals (e.g., 10.1.2.3). In a dotted-quad string, the four numbers reflect the four bytes of an Internet address, each of which is a number between 0 and 255.
* The sixteen bytes of an IPv6 address, on the other hand, are divided by colons and represented as groups of hexadecimal digits. Each digit group contains two bytes of the address; leading zeros may be omitted. Additionally, successive classes containing only zeros can be skipped entirely.
* Every Internet address refers to the link between a host and an underlying communication channel, or network interface, in technical terms. There are many interfaces that a host may provide.
* In TCP or UDP, the port number is often represented in relation to an Internet address. Each port number is a 16-bit unsigned binary number in the range of 1 to 65,535. (0 is a reserved number).
* Certain special-purpose addresses are specified in each edition of IP. The loopback address, which is often allocated to a special loopback interface, a virtual system that merely echoes sent packets back to the sender, is one of these that is worth learning. It can be used even though a device has no other interfaces because it is available on any host.
* The loopback address for IPv4 is 127.0.0.1; for IPv6, it is 0:0:0:0:0:0:0:1.
* IPv4 addresses designated for "private use" are another subset of IPv4 addresses set aside for a specific reason. All IPv4 addresses that begin with 10 or 192.168, as well as those with a first number of 172 and a second number between 16 and 31, fall into this category.
* These addresses were designed to be used in private networks that aren't connected to the Internet. They are now often found in households and small offices that use a network address translation (NAT) system to link to the Internet. This system functions similarly to a router in that it converts (rewrites) the addresses and ports in packets as they are forwarded. It maps pairs in packets on one of its interfaces to pairs on the other interface, to be more exact. This simply allows a select group of hosts to "share" a single IP address. These addresses are significant because they are not accessible through the global Internet.
* The link-local, or "auto-configuration" addresses are stored in a similar class. 169.254 is the start of such an IPv4 address. A link-local address is one whose first 16-bit chunk begins with FE8 in IPv6. Routers can not forward those addresses; they can only be used for correspondence with hosts on the same network.
* Multicast addresses are a different kind of address. Multicast addresses may refer to an infinite number of destinations, while standard IP (sometimes referred to as "unicast") addresses refer to a single destination. In IPv4, dotted-quad multicast addresses have a first number in the range 224 to 239. Multicast addresses in IPv6 begin with FF.

1. **About Names:**

* You're also used to addressing hosts by their first names. Addresses (binary numbers) are dealt with by Internet protocols, not names. The use of names rather than addresses is a convenience function that is separate from TCP/standard IP's operation. Humans are clearly better at remembering names than dotted-quads. Names offer a layer of abstraction that shields users from IP addresses.
* The name-resolution service has access to data from several different outlets. The Domain Name System (DNS) and local configuration databases are two of the most important outlets. The DNS protocol helps hosts connecting to the Internet to access information from the network via TCP or UDP. The DNS is a distributed database that maps domain names to Internet addresses and other information.

1. **Clients and Servers:**

* These roles are referred to as client and server: the client program initiates communication, while the server program waits passively for and then responds to clients who approach it. The program is made up of the client and the server working together. The terms client and server describe the usual scenario in which the server performs a certain function.
* The difference between client and server is crucial since the client must first know the server's address and port, but not the other way around. When the server gets the initial message from the client, it may use the sockets API to learn the client's address information.
* Typically, the client knows the name of the server it wants—for example, from a URL—and uses a name-resolution service to figure out the corresponding Internet address.
* Servers can use any port in theory, but clients must be able to figure out what it is. There is a convention on the Internet for assigning well-known port numbers to specific applications. The Internet Assigned Numbers Authority (IANA) is in charge of this task. The File Transfer Protocol, for example, has been allocated port number 21. (FTP).

1. **What is a Socket:**

* A socket is a data communication abstraction that allows an application to transmit and receive data. A socket enables an application to connect to the network and collaborate with other applications on the same network. Information written to the socket by one machine's application can be read by another machine's application, and vice versa.
* Various types of sockets refer to various underlying protocol suites and protocol stacks within a suite. Stream sockets and datagram sockets are the two most common types of sockets in TCP/IP today. Stream sockets provide a secure byte-stream service by using TCP as the end-to-end protocol (with IP underneath). One end of a TCP connection is represented by a TCP/IP stream socket.
* Datagram sockets use UDP (again, with IP underneath) to provide a best-effort datagram service that allows applications to send individual messages of up to 65,500 bytes. An Internet address, an end-to-end protocol (TCP or UDP), and a port number are used to identify a TCP/IP socket.