UNIT 1 TCP/IP PROGRAMMING CONCEPTS

Structure **Page Nos.**

1.0 INTRODUCTION

We are moving in the world of IT where information plays important role in our daily life. But exchange of information is also becoming important issue in today's world due to which most organisations in India and outside world use a substantial number of'computers and communication tools. Computer networks (of an organisation or the Internet) allow the user to access programs and remote databases. The commonly used protocol suit, which provides these facilities, is called as TCPIIP. You might ask yourself, why should I learn about working of TCP/IP protocol or its programming? The answer is simple, as the Internet becomes more popular, more applications are developed. You should understand how TCPIIP these protocols work and also how you can develop your own application and protocols according to your need. At present TCP/IP has become an essential built-in component of most of the operating systems, including Unix, Windows 95, Windows 98, Windows NT, OS/2, and Linux etc. Before we can start writing network programs, we must understand how these systems (Internet or Intranet) actually communicate, which means that we have to learn a little about how the client and server communicate and work. First let's understand the client-server paradigm and next we will go in the details of socket and its structure which are used by network programs for communication.

1.1 OBJECTIVES

Our objective is to introduce you with basic concept of TCPIIP programming. On successful completion of this unit, you should be able to:

- have a reasonable understanding of the TCPIIP network architecture;
- describe the operation of simple client/server architecture;
- understand the role, meaning and structure of sockets;
- describe and understand how to use sockets, and
- demonstrate an understanding how to write client/server programs.

1.2 CLIENT SERVER COMMUNICATION

Computer Networks have revolutionised our lives. We have a lot of examples where we make use of computer network like, when we withdraw money from ATM machines or when we write e-mails or when we make computerized rail reservation (Figure I).

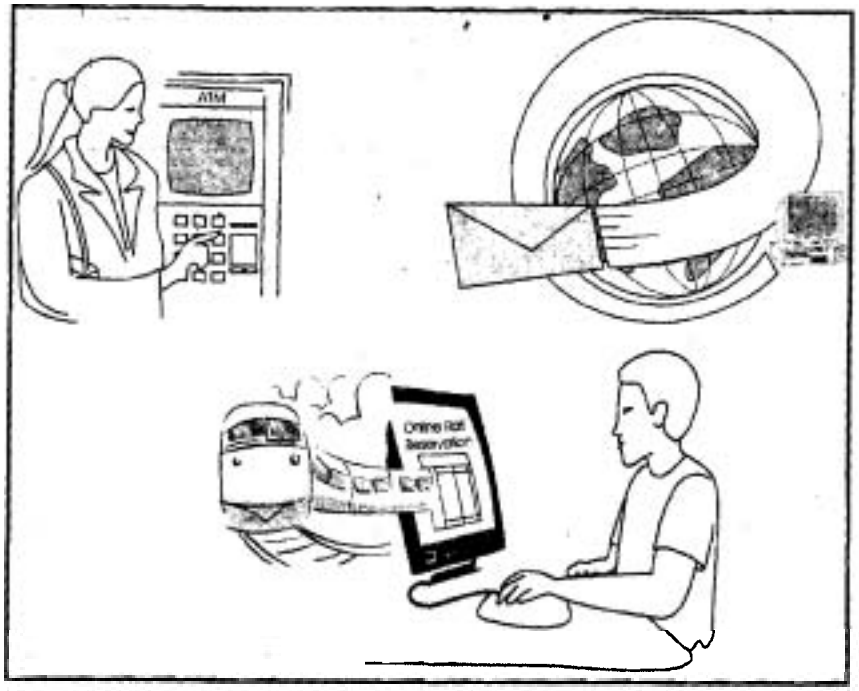


Figure 1: Use of Computer Network

All the above examples are nothing but computers communication. You must be wondering how these computers talk to each other and understand each other also? Because you are computer science students you may bethinking can 1 make them talk? Of course yes, this block is all about computer communication. But first of all try to understand how computers talk or network communication happens. Most of the computer networks use client/server model (Figure 2).

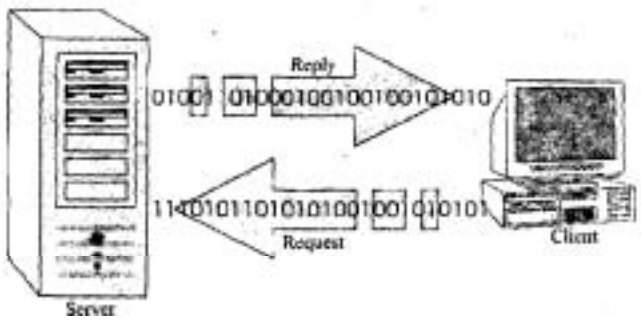


Figure 2: Client/server communications model

Client/server computing means one computer (or more) is acting as client computer one is behaving as serve which preferably of high configuration but can you think why we make a high configuration computer as a server? In simple words client is defined as requester of services and server is defined as provider of services. Client and Servers are nothing but programs, which generally run on different machines/computers.

When client wants some information or wants to perform some task by server it has to contact the server. Therefore, the client initially needs the address of the server, and tries to contact the server using that address. Server is a program which is always running and waiting to serve clients. Server welcomes the client (if free) as the client contacts it. After contacting the server, the server welcomes that client (if free), the client informs about its own address to servers so that server can reply the client also. When both clients and servers exchange important information to each other connection establishes. Once with connection or link established, both client and server could exchange any information with each other. Lets take an analogy of doctor and patients to understand the client server paradigm in our daily life. Doctor can be considered as a server and patients as clients, patients want to get cured or want some

information, so patient (s/he) should know the address of Doctor to whom s/he can contact. We assume doctor is always available for the patient, client inform about himself/herself. Now, doctor knows about patient details and history so they both can contact to each other according to their need.

We can say that a server is a program that is waiting for client so that it can do something for the clients and client program is defined as requester of services. Since server has to complete the task of many clients (may be thousands) that's why it should be of high configuration and have high speed, memory and disk space. Lets summarize the client/server communications.

A server is a process that is able to carry out some functions or services, like transferring files, translating host names to IP addresses, or any other task like finding inverse of a matrix.

1) Server is started on some computer system.
2) Client is started, either on same server computer or an different computer.
3) Client sends a request across the network to the server using server's network address which client should know before contacting.
4) Server listens the request and welcomes the client.
5) Client tells about its local address to server.
6) Connection established.
7) Both communicate to each other/servers serve the client like:

- Print the files for the client,
- Read or write a file present at the server etc.

3) When client's request is over and work is done, server goes back in waiting for other clients.

1.2.1 Designing of Client/Server Programs

Architecture of client and server program involves different issues, according to the system requirements. Here, in this section we have explained you about the characteristics and categories of client and server programs.

a) Characteristics of client program

Here we are explaining you about different characteristics of clients programs so that while writing programs you can always take care of these given characteristics. As you know, client program is an application program that becomes a client temporarily when remote access is needed.

- It is invoked directly by a user, and executes only for 1 session.
- Client program actively initiates contact with a server.
- Client program can access multiple services as needed, but actively contacts one remote server at a time.
  Client program does not require special hardware or sophisticated operating system.

b) Characteristics of sewer program

Server program is a special-purpose, program dedicated to provide generally one service, but server program can handle multiple remote clients at the same time. Server program has following characters:

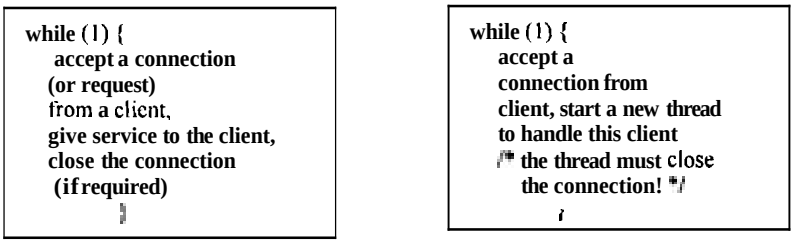
It is invoked automatically when a system boots, and continues to execute through many sessions. (inetd).

- Runs on a shared computer.
  Waits passively for contact request from arbitrary clients.
  Accepts contact from arbitrary clients, but offers a single service.

c) Iterative and concurrent programs

When server knows that client is going to take short amount of time, server program handles the client request one by one in linear manner. Such server are known as

interactive or sequential server programs. But think, if server does not know how much time client will take, server may get busy with only one long request and other poor clients will keep waiting. In case server does not know about the time clients will take, servers typically handles it concurrent fashion, and such server is called concurrent server program. Most client software achieves concurrent operation because the underlying Operating System allows users to execute client programs concurrently or because user can execute client software simultaneously. A concurrent server invokes another process to handle each client request so that original server is always free to serve the clients. The program structure of iterative and concurrent server are given in the *Figure3*.



Interactivc Concurrent

Figure 3: Program structure of interactive and concurrent servers

d) Standard and non-standard programs

Broadly client applications programs are categorised into standard and non-standard application programs. Standard application programs are those that use standard services like tile transfer, electronic mail etc. of TCP/IP application layer. The application program which uses the services definitely privately or locally by the organisation according to its need is called non-standard application. For example, a site of private database system of some institute. Remember, standard application services are assigned to well known (universly reorganised) port number. Like file transfer protocol is university assigned 21 port number. Non-standard applications are assigned to locally known (internally or privately recognised) port numbers. The site of Indira Gandhi National Open University is assigned to 3128 port number. System Administrator can assign these services to different port number according to its need and the options given by the authorities. It is important for network programmers to understand the differences between these standard and non-standard services and their port numbers.

e) Connection and connectionless programs

When you design client/server program you must select the type of communication mode. You can either choose connection oriented communication mode or connection less communication mode, which directly refer to the transport layer protocols of TCP/IP. As you know TCP (Transmission Control Protocol) provides connection oriented services and UDP (User Datagram Protocol) provides connectionless services. If TCP is used between client & server, we can say it is connect oriented communication and if UDP then its connectionless communication. Connection oriented and Connection less services are distinguished in the scale of reliability'-level. TCP provide reliability in communication between client and server, which means that whatever data is sent, the sender knows data is delivery properly or not. In connection-oriented communication underlying protocol (e.g., TCP) verifies the data delivery by acknowledgement and if not transmitted properly, data is automatically resent. It is also computes a checksum of data packets to provide the guarantee of non-corruption of data. TCP use the sequence numbers to maintain the proper sequencing and ordering of the data.

In contrast to this. connectionless services do not provide assurance about reliable delivery of data (no acknowledgement) or ensure that whether data will reach to the entitled process or not. For example, when client/server send some data to server it may be lost, resend, duplicated, can get corrupted, can be non-sequential or may be

received or delivered out of order. But, instead of these problems and disadvantages the connectionless services **(UDP)** provides best-effort delivery. Retransmission is not performed but application must detect lost data and retransmit if needed. It is having very less overheads because of its simplicity; it provides very high transmission speed even in the wide area networks.

f) Stateful and Stateless programs

State information is the information that is maintained by a server about its states (some information between requests) of the ongoing interaction with client programs. If a server program is maintaining the states of ongoing communication it is called as stateful server program. FTP is the stateful server, otherwise it is stateless (like, HTTP server). Maintaining these states of the interaction improve the efficiency of the server. Maintaining small amount of information about the states reduce the size of message that client and server exchange. It is also helps the server to understand on different requests quickly; in addition to that it may also add some security features in the communication. In contrast the stateless server avoids the possibility of wrong or incorrectly reply, because sometime state information can become incorrect, duplicated, delivered out of order or client reboot between the communications.

Generally, server programs are designed into four categories accordingly to their capability of providing concurrency and their use of communication mode. These are summarized here in the ***Figure*** 4.

	**Iterative**	**Concurrent**
Connectionless	Iterative connectionless	Concurrent Connectionless
Connection-Oriented	Iterative connection-oriented	Concurrent Connection-oriented

***Figure 4: Categories of sewer* program**

These four categories of server are further explained here:

i) **Concurrent Connectionless:** It is rarely used server design, where server create a new process to attend each request from the client, this concurrently provides the efficiency in the system but the time to create new process must be considerably less than the time required to compute the request. Generally, in this concurrent connectionless server design cost put for process creation become higher than the efficiency achieved with the concurrency. So, it is uncommonly used.

ii) **Concurrent Connection-oriented:** Most general design of server, it is suitable for service, which needs reliable transport of data in wide area networks and it can handle multiple clients simultaneously. It further has two different implementations:

- Concurrent process to handler connections.
- A single process and asynchronous I/O to handle multiple connections. (Process repeatedly waits for an I/O on any of the connections it has open and handles that request).

iii) **Iterative Connectionless:** Most commonly used server, this kind of server is suitable for services that need a less amount .of processing for each request. Iterative servers are generally stateless, simple and less vulnerable to failures.

iv) **Iterative Connection-oriented:** It is less commonly used server. It is used for the services that need less amount of processing for each request but the reliability in communication is a necessary. But these kinds of servers waste their time in connection establishment and termination.

Check Your Progress 1

1) Why the 'Server Computer' is high speed, memory and disk space?

.....

.....

2) Write the difference between interactive and concurrent programs.

.....

.....

3) List different categories of services.

.....

.....

1.2.2 Socket Concepts

As we have studied earlier in the Unit 1 that socket is basically a combination of IP address and port number, but socket structure contains many more things. To understand in detail let study the concept of connections and association in this section.

a) Connections and Associations

As you know when client wants to communicate to server it first establishes "Connection". Here connection is nothing but a communication link between client program and server program. As we have defined in our present section, when connecfion is established between client and server, they exchange a set of important information that is known as "Association". Lets see what exactly we mean by association.

The term association is used for the 5-tuple's that completely specify the two processes that comprise a connection.

Association {

protocol,
local address,
local process,
remote address,
remote process.
}

Let's understand the 'Association' with the help of one example. Machine 'A' is local host and wants to establish connection with Machine 'B' which is a remote host. Now Machine 'A' has to define which protocol it will use to establish connection like TCP or UDP (in-case of TCP/IP). Remote and local address of machines, are like 'IP address' of machine in TCP/IP, which actually identify the location of machine in network.

Local Process and Remote Process are used to identify the specific process in system that is involved in establishing connection. For example, Association needed for connection in TCP/IP is (UDP, 192.23.15.10, 1211, 192.64.10.12, 2102) where UDP is a protocol. The association with 5-tuple's is also known as full association but we also have half association.

Like:

{protocol, local address, local process)
&
{protocol, remote address, remote process)

Which define each half of the connection. This half Association is also called as **'Socket'** or 'transport address'. The term Socket's popularized by Berkley Unix networking system. In the next section we will discuss Socket in detail.

b) What is Socket?

The basic building block for computer communication is the socket, which is an end-point of communication. Sockets allow communication between two different processes on the same or different machines. Let's see the concept of sockets in general term. In term of electricity we use "Electrical socket" which use to get electrical power plug an appliance's power cord into a socket in the wall. When you want to talk to your friend through telephone, you can connect your telephone with different end-points (Telephone-Socket) available in your home as shown in the Figure 5. The same concept we can realize when we talk about the Socket for communication between machines.

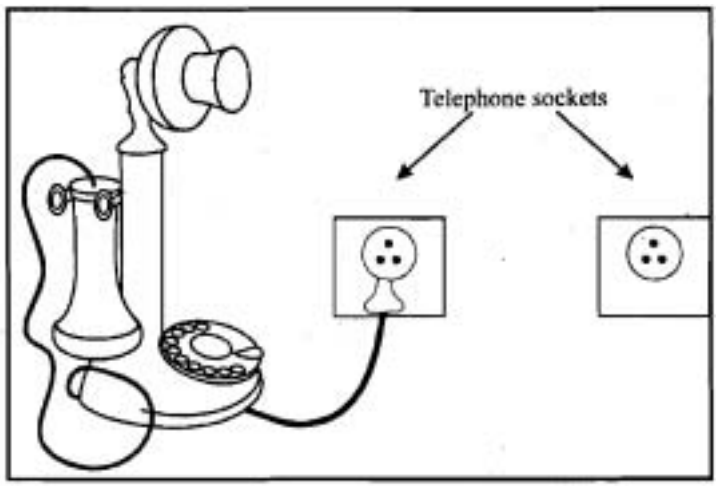


Figure 5: Socket is an end-point of communication

Why is it called a socket? Because of the analogy to plugging a wire into something that can understand what is carried by the wire. When two applications want to talk to each other so, one application tells the OS to open a socket and through socket it uses communication protocol, at receiver side one socket will be ready with the open socket. Once connection is established application can send and receive data.

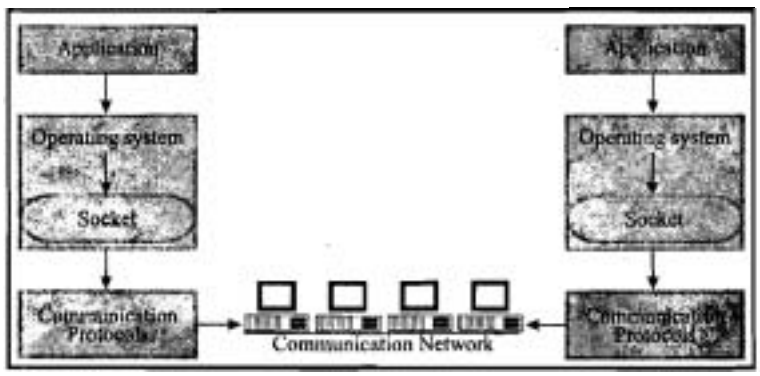


Figure 6: Sockets are basic building block of communication

1.2.3 IP Address and Port

An internet address or IP address is number made of 32 bit or 4 bytes (number between 0 and 255) written in the form of 'dotted notation' for example "192.168.10.10" is a valid IP address usually one computer has only one IP address but a computer might have more than one such address, if computer has more than one NIC (Network Interface Card) connected to internet. Therefore, when we mean host we should understand, IP address that identifies a host Interface not a host but usually we mean it to host.

When we talk about Network communication we say IP address identifies the host computer. However, usually more than one application program running on a computer use the network. Because of this reason TCP/IP use another level of address called port number, which identify the process running on computer.

Let's see an example, you want to call me, you know the IGNOU phone number is 29536207 but in IGNOU we have many teachers that is why you have to use my extension 2229, here in analogy to network communication, telephone number is IP address and extension number is as Port Number. Explain number is used, as another level of address is to resolve among teachers in IGNOU.

In networking, communication address is made up of an IP address and a Port Number. Port Number is a 16-bit identifier and each host has 65336 ports. In TCP/IP some ports are reserved for specific application for example, port number 21 is reserved for FTP, Port number 23 is for telnet, 80 for HTTP.

Name	Port Range	Use
The Well Known Ports	0 to 1023	Reserved for common services like telnet, ftp.
The Registered Ports	1024 to 49151	Registered for companies and organisation
The Dynamic and/or Private Ports	49152 to 65535	Available for the rest of world to use.

1.2.4 Byte Ordering

Some computers are "big endian". This refers to the representation of objects such as integers within a word. In big endian machine, the high byte of an integer is stored in low byte address. In little endian, the high byte of an integer is stored in high byte address as shown .in Figure 7. A Sun Sparc is big endian. So the number 2 + 3 • 256 would be stored as:

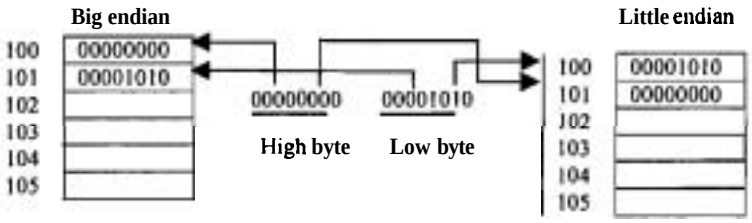
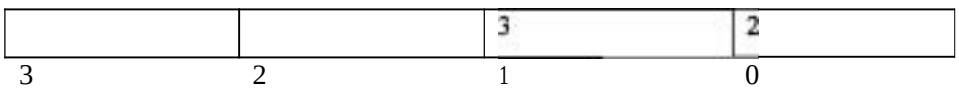
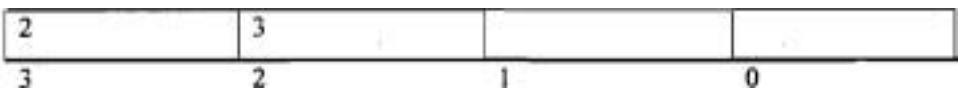


Figure 7: Data representationin big and little endian



A "little endian" machine stores them the other way. The 386 is little endian.



If a Sparc sends an integer to a 386, what happens? The 386 sees a data "2 + 3* 256" as "2 *16777216 + 3 • 65536".

1.2.5 Sketch of Networking Connection

The client/server model is used to divide the work of networks into two parts. One part knows how to do a certain task or how to provide certain service, this part is known as server program. Other part defines how to talk to user and how to connect to server program, known as client program. Server may give service to many different clients either simultaneously (concurrent services) or one after the other (Iterative services). On the other hand client talks to single user at a time. In network program we can have different possibilities depending on the needs. But first of all we will discuss only simple one. Let's see the following example where one client talks to one server. The main feature of client is to provide convenient user interface, hiding the details of how the server communicates from the user. The client needs to first establish a connection with server, on given address. Generally client does the following steps:

1) Create a socket.

2) Specify the address and service port of the server program.

3) Establish the connection with the server.

4) Send and receive information.

5) Close the socket when finished, terminating the conversation.

To establish and complete a connection the sewer-side would follow these steps:

1) Create a socket.

2) Listen for incoming connection requests from clients.

3) Accept the client connection.

4). Send and receive information.

5) Close the socket when finished, terminating the conversation.

1.2.6 Active and Passive Socket

The client application always starts the communication. It creates a socket and actively attempts to connect to the server program. This is known as active open or active socket. On the other hand the server application creates a socket and passively listens for incoming connection requests from clients. This socket is passive open, that's why this socket is called as passive socket. This is also same as the telephone system where one phone is always waiting for incoming calls.

When client initiates a connection, the server notices that same process is trying to connect with it. By accepting the connection the server establishes a logical communication path between two programs. (Generally in practical situation, the act of accepting a connection creates a new socket. The original socket remains unchanged so that it can continue its availability to clients for additional connections when the sewer no longer wishes to listen for connection).

☞ Check Your Progress 2

1) What are the types in association?

.....

.....

.....

2) What is a Socket?

.....

.....

.....

3) Write the difference between active and passive socket.

.....

.....

.....

1.3 SOCKET FUNDAMENTALS

To standardize network programming, Application Programs Interface's (API's) have been developed. An API is a set of declarations, definitions and procedure's followed by programmers to write clients server programs. For network programs commonly tleveloped API's are Socket Interfaces, Transport Level lnterface (TLI), Stream Interface, Thread Interface and Remote Procedural Call (RPC). Let's discuss socket interface in details but first see the data types defined in socket interface.

There are u_char u_short and u_long data type.

```
u_char    // character unsigned 8 bit
u–short   // short integer  unsigned 16 bit
u_long    // long integer unsigned  32 bit integers
```

From application point of view, the differences between network protocols are address schemes used. For TCPIIP, an ideal API would be one that understand IP addressing and port numbers. Since the socket library is designed to be used for multiple protocols, addresses are referenced by a generic structure as follows:

```
struct sockaddr {
u–char  sin_len;       /* length of structure */
u–short sa_family;     /* address family: AF_XXX value */
char sa_data[14];      /* up to 14 byte of protocol specific address*/
                    };
```

Originally sa_len was not there in the structure. The **sa_family** field specifies the type of protocol. But in case of TCPIIP, this field is always set to AF_INET (where INET stand for Internet). The remaining 14 bytes (sa_data) of this structure are always protocol dependent for TCPIIP, IP addresses and port numbers are placed in this field. The contents of the 14 bytes of protocol dependent addresses are interpreted according to the type of address. (In TCPIIP this is defined in < netinet/in.h >header file.)

Internet socket address structure: The structure sockaddr–in describing an address in the lnternet domain is defined in the file <netinet/in.h>. The application program that use the TCP/IP need a specific type of socket address structure, which can hold IP address, port number and protocol family. This structure is called as sockaddr–in., which have five fields.

```
struct sockaddr–in
    {
u–char      sin–len;       /* length of structure  */
u–short     sin–family;    /* for Internet it is AF_INET */
u–short     sin–port;      /* 16-bit port number */
struct in–addr  sin–addr;  /* 32-bit address */
char        sin_zero[8]; /* unused, reserved for future */
    };
```

In this structure first and last field are normally not used. A sockaddr–in structure contains an in–addr structure as a member field. In–addr has the following structure.

```
struct in–addr {
u_long  s_addr;        /* 32-bit address */
                  };
```

Note one thing here about the **sock–addr & sockddr–in** structures; these both structures are compatible with each other. Both structures are 16 bytes in size and first two bytes of both structures are the family field. Thus, struct sockaddr–in can always be cast to struct sock–addr. lt is important to remember that these fields always need to be set and interpreted in network byte order (this topic we will discuss in unit 3 of this Block).

A socket is defined in Operating System as structure, which has five fields. (Remember the five tuple's in association, same here in the structure of socket). Let's see these fields of structure in the Figure 8:

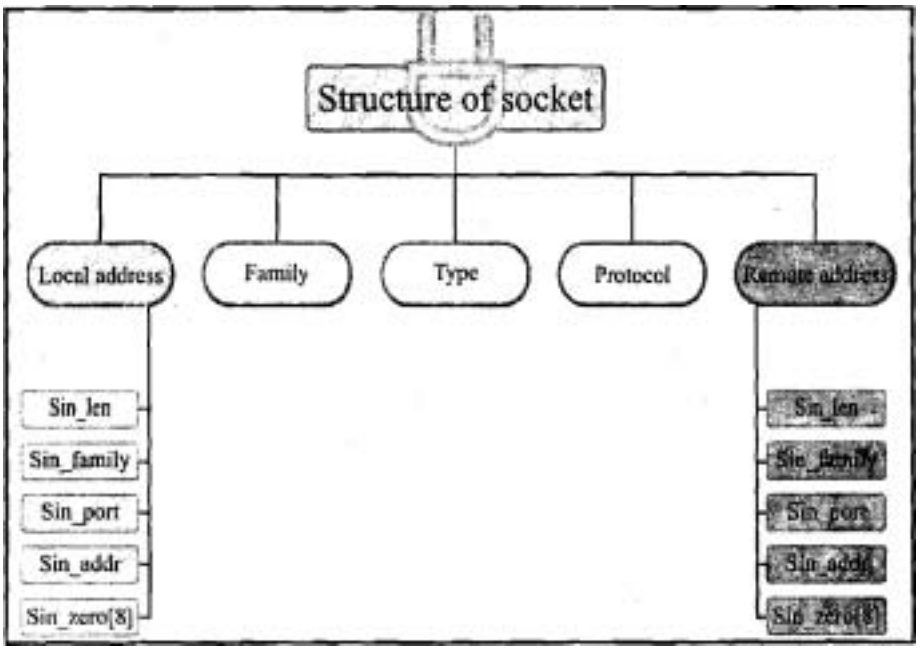


Figure 8: Structure of socket

1) **Family:** An application program specifying a socket family defines the protocol group needed for communication. It can be Unix domain protocols, Internet-domain protocol, Xerox NS protocols and IMP link layer (IMP is Interface Messaging Protocol) as given below, In case of TCP/IP this field is AF_INET.

2) **Type:** An application program specifying a socket type, which indicates the desired communication style for the socket. This field defines the type of socket like stream socket, data-gram socket and raw socket. Each socket has a type associated with it. The socket type determines the socket communication properties like reliability, ordering, and prevention of duplication of messages. The basic set of socket types is defined in the **sys/socket.h** header file. Total we have five types of Socket options available. Let's discuss in details what are these types of sockets:

   i) **Stream Socket:** Stream socket is used with connection oriented protocol. It provides stream model of communication. A stream socket provides for the bi-directional, reliable, sequenced, and unduplicated flow of data without record boundaries. This service is reliable and connection oriented. Transport protocols support this socket type and ensure that the delivery is in-order, without loss or error or duplication, if not, then abort the connection and report the error to the user. Message boundaries are not present in stream sockets. It can be used in both Unix and Internet domain. In case of Internet domain it uses TCP, which is connection-oriented protocol.

   ii) **Datagram Socket:** Datagram sockets are used with connection-less protocols. It provides datagram model of communication. It supports bl-directional flow of data, between sender and receiver and data is not promised to be in sequence, reliable, or unduplicated. That's why in case of datagram socket, a process receiving messages on Bdatagram socket may find messages duplicated, and, possibly, in an order different from the order in which it was sent. An important characteristic of a datagram socket is that record boundaries in data are preserved. This type of socket

is generally used for short messages, such as a name server or timeserver, since the order and reliability of message delivery is not guaranteed.

iii) **Raw Socket:** Raw Socket, as its name indicates, provides access to internal network protocols and interfaces. A raw socket allows an application direct access to lower-level communication protocols (like, IP layer in case of TCP/IP). Raw sockets are developed for advanced programmers who want to build new protocols or who are interested to use the low level feature, that are not directly accessible through a normal interface. Raw sockets are normally datagram-based, though their exact characteristics are dependent on the interface provided by the protocol.

iv) **Sequenced Packet Socket:** It provides sequenced, reliable, and unduplicated flow of information. A sequenced packet socket is similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as part of the NS socket abstraction, and is very important in most serious NS applications.

v) **Reliably Delivered Message Socket:** The reliably delivered message socket has similar properties to a datagram socket, but with reliable delivery. There is currently no support for this type of socket, but a reliably delivered message protocol similar to xerox's Packet Exchange Protocol (PEX) may be simulated at the user level.

3) **Protocol:** The protocol specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family. However, it is possible that many protocols may exist; in that case a particular protocol must be specified.

4) **Local Socket Address: A** pointer to a buffer of type struct sockaddr–in that contains the local address to which the socket is to be bound.

5) **Remote Socket Address: A** pointer to a buffer of type struct sockaddr–in that contains the remote address to which the socket is to be bound.

Now lets summarise the characteristics of socket:

- A socket is referenced by an integer. That integer is called a socket descriptor
- A socket exists as long as the process maintains an open link to the socket.
- You can name a socket and use it to communicate with other sockets.
- Sockets perform the communication when the server accepts connections from them, or when it exchanges messages with them.
- You can create sockets in pairs (only for sockets in the AF_UNIX address family).

1.4 NETWORKING EXAMPLE

As we have discussed earlier that client program can use the connection oriented reliable service or connection less unreliable services. So you should know the algorithms or steps for these client program.

Connection oriented and Concurrent client

Here we have given you an algorithm, where client opens the socket, sends a request and gets the response.

Step 1: Identify the IP address and port No. of a server.

Step 2: Open a socket.

Step 3: Choose TCP protocol (a connection oriented protocol) and an unused protocol port.

Step 4: Connect Socket to the server.

Step 5: Using application level protacol, communicate with server.

Step 6: Close the connection, when the job is over.

Connectionless Client

Step 1: Same as step 1 in connection oriented client

Step 2: Same as step 2 in connection oriented client

Step 3: Choose UDP Protocol (a connectionless protocol)

Step 4: Inform the server about the client address so that server can communicate the client.

Step 5: Same as step 5 connection oriented client

Step 6: Same as step 6 connection oriented client

Now you learnt the client program algorithm, but without server these are not useful so you must learn how to design different kinds of server programs. As we have early classified them into 4 types in the section, let's make there algorithms one by one.

Iterative and Connection oriented server

Step 1: Create a socket, and associate it with the well-known address (which is known to the client).

Step 2: Make the socket in passive mode.

Step 3: Welcome the connection request from the client, and create a new socket for the connection

Step 4: Read request from client repeatedly and give them reply according to application protocol.

Step 5: When the job of particular client is over close that connect and rather step 3.

Here at a time, a single process handles one connection from client, only one when it finish its request than only it accepts next connect request.

Iterative and connectionless server

Step 1: Same as step 1 iterative and communication Server

Step 2: Same as step 2 iterative and communication Server

Step 3: Repeatedly read the next request from the client and send the reply according to the protocol.

Connectionless and concurrent server

Master 1: Create socket and fix it to some address (well known to client) offered Socket must be left unconnected.

Master 2: Receive next request from client and create copy of the same (child process).

Slave 1: Receive the specific request

Slave 2: Reply accordingly to application protocol and give response to client

Slave 3: Terminate the child process after completion of the task.

Connection oriented concurrent server

Master 1: Create socket and fix it to the address, which is known to the client. Socket must be unconnected.

Master 2: Make the socket as passive socket

Master 3: Continuously receive the connection request from client and create slave server process to handle the communication with client.

Check Your Progress 3

1) What are the data types defined by the socket interface?

.....

.....

2) Write the structure of socket address.

.....

.....

3) Write the algorithm for interactive connection oriented server.

.....

.....

1.5 SUMMARY

In this unit we have discussed different issues related to client server communication so that you will have good understanding of different servers and their services like connection oriented and connectionless. The procedure of basic client and server was also explained in detail. After completing the theoretical conceptualization of client server communication we have defined the socket, which is considered as an end point of communication in network programming. The different components related to the sockets like, IP address, port, address family and socket types are also explained in this unit. The concept of byte ordering and its use in networking is explained briefly with the help of an example which you will again study in detail during the unit socket programming. Further the differences between Active and Passive Socket was explained and at the end of the unit we have given you some algorithms and procedure of different client server communication. In the next unit Socket interface we have given basic socket calls, which will provide you the knowledge of the functions and their characteristics.

1.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Server is a program that is waiting for client so that server can do something for the client and client program is defined as requester of services. Because server has to complete the task of many clients (may be thousands) that's why it should be of high configuration and have high speed, memory and disk space.

2) When server knows that client is going to take short amount of time then server program handles the client request one by one in linear manner, which is known as interactive or sequential server. In this, if server don't know how much time client will take, server will get busy with one request itself and other poor clients will wait. A concurrent server invokes another process to handle each client request so that original server can always be free to serve client. That's why we can say practically it is more efficient.

3) Server programs are designed into four categories accordingly to their capability of providing concurrency and their use of communication mode.

These are following:

1) Concurrent Connectionless
2) Concurrent Connection-oriented.
3) Interactive Connectionless
4) Interactive Connection-oriented

Check Your Progress 2

1) The term association is used for the 5-tuple's that completely specify the two processes that comprise a connection.

```
Association {
        Protocol,
        local Address,
        local process ,
        remote address,
        remote process.
    }
```

2) The basic building block for computer communication is the socket, which is an end-point of communication. Sockets allow communication between two different processes on the same or different machines.

3) The client application always starts the communication. It creates a socket and actively attempts to connect a server program. This is known as active open or active socket. On the other hand the server application creates a socket and passively listens for incoming connection from clients that is passive open situation. That's why this socket is called as passive socket.

Check Your Progress 3

1) The main dates types defined in this socket interface are u_chart u_shorts and u_long data type, it's meaning are defined below:

- u_chart : character unsigned 8 bit
- u_shorts : short integer unsigned 16 bit
- u_long : long integer unsigned 32 bit integers

2) The general socket address is given below:

```
struct sockaddr {
        u_char  sin–len;        /* length of structure */

        u_short sa_family; /* address family: AF_XXX value */

        char sa_data[14]; /* up to 14 byte of protocol specific address*/

        }
```

Where sa_len indicates the length of the address structure. The sa_family field specifies the type of protocol. The remaining 14 bytes (sa_data) of this structure are altvays protocol dependent for TCP/IP, IP addresses and port numbers are placed in this field. The contents of the 14 bytes of protocol dependent addresses are interpreted according to type of address.

3) The algorithm for Interactive and Connection oriented server is as given below.

Step 1: Create a socket, and associate it with the well-known address (which is known to the client).

Step 2: Make the socket in passive mode.

Step 3: Welcome the connection request from the client, and create a new socket for the connection.

Step 4: Read request from client repeatedly and give them reply according to application protocol.

Step 5: When the job of particular client is over close that connect and rather step 3.

Herr: at a time, a single process handles one connection from client. only one finishes its request than only it accepts next connect request