B.E.

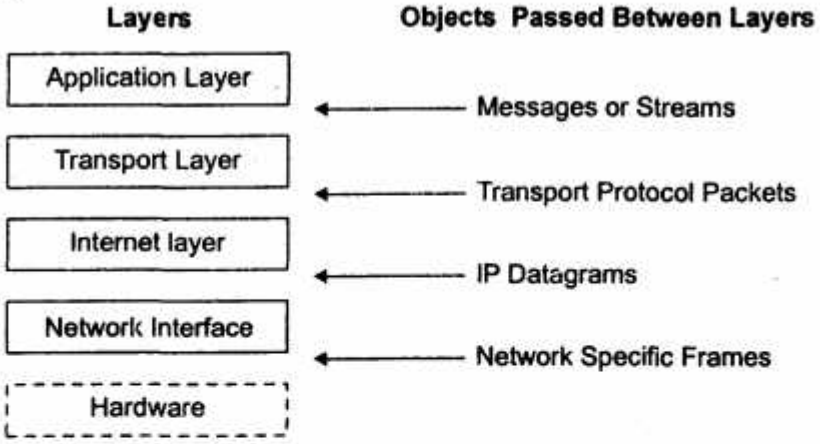
Sixth Semester Examination, May-2008

Network Programming (IT-302-E)

Note : Attempt any **five** questions. All questions carry **equal** marks.

Q. 1. (a) Explain TCP/IP protocol Architecture. 7

Ans. TCP/IP software is organized into five conceptual layers-four software layers that build on a fifth layer of hardware.



Application Layer : At the highest layer, users invoke application programs that access services available across a TCP/IP internet. An application interacts with one of the transport layer protocols to send or receive data.

Transport Layer : The primary duty of the transport layer is to provide communication from one application program to another, such communication is often called end-to-end. The transport layer may regulate flow of information. It may also provide reliable transport, ensuring that data arrives without error and in sequence.

Internet Layer : As Internet layer handles communication from one machine to another. It accepts a request to send a packet from the transport layer along with an identification of the machine to which the packet should be sent. It encapsulates the packet in an IP datagram, fills in the datagram directly or send it to a router and passes the datagram to the appropriate network interface for transmission. The Internet layer also handles incoming datagrams.

Network Interface Layer : The lowest layer TCP/IP software comprises a network interface layer, responsible for accepting IP datagrams and transmitting them over a specific network. A network interface may consists of a device drive or complex subsystem that uses its own data link protocol.

Q. 1. (b) Explain ICMP with its purpose. 7

Ans. ICMP : To allow routers in an internet to report errors or provide information about unexpected circumstances, the designers added a special-purpose message mechanism to the TCP/IP protocols. The mechanism, known as the Internet Control Message Protocol {ICMP}, is considered a required part of IP and

must be included in every IP implementation. Like all other traffic, ICMP messages travel across the internet in the data portion of IP datagrams. The ultimate destination of an ICMP message is not an application program or user on the destination machine, however, but the internet protocol software on that machine. That is, when an ICMP error message arrives, the ICMP software module handles it. If ICMP determines that a particular higher-level protocol or application program caused a problem, it will inform the appropriate module. ICMP is an error reporting mechanism. It provides a way for routers that encounters an error to report the error to the original source. ICMP does not fully specify the action to be taken for each possible error.

Q. 1. (c) List out various advantages and disadvantages of computer networking. **6**

Ans. Advantages of Computer Networking :

(i) Resource Sharing : It allows all programs, equipment and data available to anyone on the network irrespective of the physical location of the resource and the user.

(ii) High Reliability : It provides high reliability by having alternative sources of data.

(iii) Money Saving : Computer networking is an important financial aspect for organizations because it saves money. Organizations can use separate personal computer one per user instead of using mainframe computer which are expensive.

(iv) Powerful Communication Medium : A computer network provides a powerful communication medium among widely spread employees.

Disadvantages :

(i) No Security : Computer network is a threat for security of data as any unauthorized user can also access some important data.

(ii) Virus Threat : If one computer in a network get infected by virus there is a possibility of virus infecting all the computers in the network.

(iii) Network Halt : Any fault in connecting devices like switches, hubs, etc. and lead to a whole network halt.

Q. 2. Explain the following in context of socket (TCP and UDP) : **20**

(a) Elementary node

(b) Address Conversions

(c) Echo services

Ans. (a) Elementary Node : Elementary socket functions are required to write a complete TCP client and server. To perform network I/O, the first thing a process must do is call the socket function, specifying the type of communication protocol desired family specifies the protocol family and is one of the constants. This argument is often refferred to as domain instesd of family. The socket type is one of the constants. The protocol argument to the socket function should be set to the specific protocol type found or 0 to select the system's default for the given combination of family and type. Not all combinations of socket family and type are valid. On success, the socket function returns a small non-negative integer value, similar to a file descriptor.

(b) Address Conversions : The DNS {Domain Name System} is used primarily to map between host names and IP addresses. A hostname can be either a simple name, such as solaris or freebsd, or a fully qualified domain name, such as solaris.unpbook.com

It is possible to obtain name and address information without using the DNS. Common alternatives are static host files, the Network Information System {NIS} or Lightweight Directory Access Protocol {LDAP}. We can resolve the address by calling resolver functions such as **gethostbyname** and **gethostbyaddr.**

Host computers are normally known by human-readable names. The most basic function that looks up a hostname is **gethostbyname.** If successful, it returns a pointer to a hostent structure that contains all the IP_{V4} addresses. The function **gethostbyaddr** takes a binary IP_{V4} address and tries to find the hostname corresponding to that address. This is the reverse of gethostbyname.

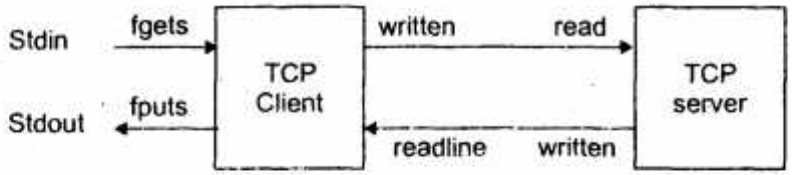
The function returns a pointer to the same hostent structure that we described with gethostbyname. The field of interest in this structure is normally h_name, the canonical hostname.

(c) Echo Services : An echo server performs the following steps :

(i) The client reads a line of text from its standard input and writes the line to the server.

(ii) The server reads the line from its network input and echoes the line back to the client.

(iii) The client reads the echoed line and prints it on its standard output.



Two arrows between the client and server, but this is really one full-duplex TCP connection. The fgets and fputs functions are from the standard I/O library and the written and readline functions.

While we will develop our own implementation of an echo server, most TCP/IP implementations provide such a server using both TCP and UDP. A client/server that echoes input lines is a valid, yet simple, example of a network application.

Q. 3. (a) Explain various debugging techniques with examples. **10**

Ans. Following are various debugging techniques :

(a) System Call Tracing : Many versions of Unix provide a system call tracing facility. This can often provide a valuable debugging technique :

(i) BSD Kernel Sockets : We start with FreeBSD, a Berkeley derived kernel in which all the socket functions are system calls. The **Ktrace** program is provided by Free BSD to run a program and trace the system calls that are executed.

(ii) S9 Kernel Sockets : Solaris 2.x based on SVR4 and all releases before 2.6 have implemented this socket for tracing system calls.

(b) Standard Internet Services : Many sites now prevent access to these services through their firewalls because of some denial-of-service attacks using these services in 1996. Nevertheless, we can hopefully use these services within our own network.

(c) Sock Program : Steven's sock program first appeared in TCP_{V1}, where it was frequently used to generate special case conditions, most of which were then examined in the text using tcpdump. tcpdump program reads packets from a network and prints lot of information about the packets.

(d) Netstat Program : This program serves multiple purposes :

(i) It shows the status of networking endpoints.

(ii) It shows the multicast groups that a host belongs to on each interface.

(iii) It shows the per-protocol statistics with the -s option.

(iv) It displays the routing table with the -r option and the interface information with the -i option.

(e) lsof Program : The name lsof stands for "list open files", like tcpdump, it is a publicly available tool that is handy for debugging and has been ported to many versions of Unix.

Q. 3. (b) List out various difference between TCP and UDP datagrams. **10**

Ans.

S. No.	Characteristic/ Description	UDP	TCP
(i)	General Description	Simple, high-speed, low-functionality "Wrapper" that interfaces Applications to the network layer and does little else.	Full-featured protocol that allows application to send data reliably without worrying about network layer issues.
(ii)	Protocol Connection setup	Connectionless; data is sent without setup.	Connection-oriented, connection must be established prior to transmission.
(iii)	Data Interface To Application	Message-based; data is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure.
(iv)	Reliability and Acknowledgements	Unreliable, best-effort delivery without acknowlegement.	Reliable delivery of messages: all data is acknowledged.
(v)	Retransmissions	Not performed. Application must detect lost data and retransmit.	Delivery of all data is managed, and lost data is retransmitted automatically.
(vi)	Features Provided to Manage Flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms.
(vii)	Overhead	Very low	Low, but higher than UDP
(viii)	Transmission Speed	Very high	High, but not as high as UDP
(ix)	Data Quantity Suitability	Small to moderate amount of data (up to a few hundred bytes).	Small to very large amounts of data (up to gigabytes).
(x)	Types of Applications that use The Protocol	Application where data delivery speed matters more than completeness, where small amount of data are sent; or where multicast/broadcast are used.	Most protocols and applications sending data that must be received reliably, including most file and message transfer protocols.
(xi)	Well-known Application and Protocols	Multimedia application, DNS, BOOTP, DACP, TFTP, SNMP, RIP, NFS (early versions).	FTP, Telnet, SMTP, DNP, HTTP, POP NNTP, IMAP, BGP, IRC, NFS (later versions).

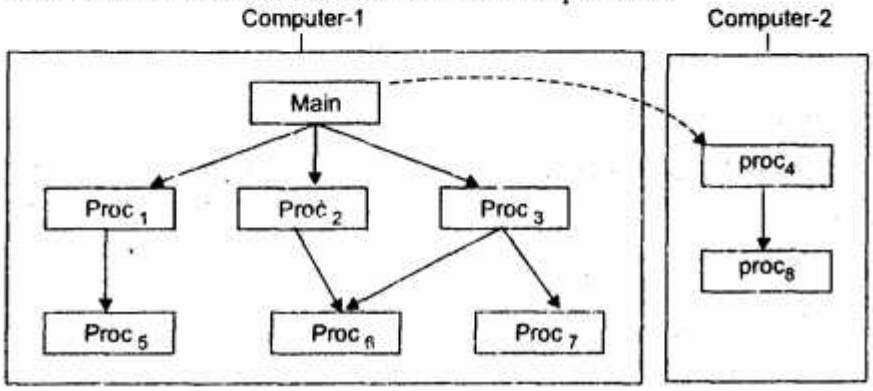
Q. 4. Explain the following : **20**

(a) RPC Transmission

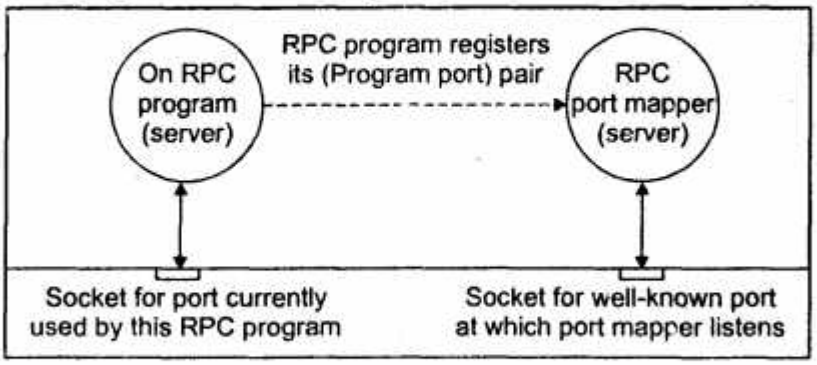
(b) Dynamic Port mapping

(c) Authentication in RPC

Ans. (a) RPC Transmission : The remote procedure call paradigm for programming focuses on the application. It allows a programmer to concentrate on devising a conventional program that solves the problem before attempting to divide the program into pieces that operate on multiple computers. The remote procedure call model uses the same procedural abstraction as a conventional program, but allows a procedure call to span the boundary between two computers. Although, a conventional procedure call cannot pass from one computer to another. Before a program can use remote procedures calls, it must be augmented with protocol software that allows it to communicate with the remote procedure.



(b) Dynamic Port Mapping : To solve the port identification problem, a client must be able to map from an RPC program number and a machine address to the protocol port that the server obtained on the destination machine when it started. The mapping must be dynamic because it can change if the machine reboots or if the RPC program starts execution again.



To allow clients to contact remote programs, the sun RPC mechanism includes a dynamic mapping service. Each machine that offers an RPC program {i.e., that runs a server} maintains a database of port mappings and provides a mechanism that allows a caller to map RPC program numbers to protocol ports. Called the RPC port mapper or sometimes simply the port mapper, the RPC port mapping mechanism uses a

server to maintain a small database on each machine. The diagram below illustrates that the port mapper operates as a separate server process.

Each RPC program registers its program number and protocol port number with the port mapper on its local machine. A caller always contacts the port mapper on a machine to find the protocol port to use for a given RPC program on that machine.

(c) Authentication in RPC : RPC defines several possible forms of authentication, including a simple authentication scheme that relies on UNIX and a more complex scheme that uses the Data Encryption Standard (DES) originally published by the National Bureau of Standards. Authentication information can have one of the four types shown below :

```
enum auth_type {
        AUTH NULL = 0; /* no authentication */
        AUTH_UNIX = 1; /* UNIX machine name authentic */
        AUTH_SHORT = 2; /* used for short form auth. in */
        AUTH_DES = 3;
        };
```

In each case, RPC leaves the format and interpretation of the authentication information upto the authentication subsystem. Therefore, the declaration of the authentication structure in an RPC message uses the keyword opaque to indicate that it appears in the message without any interpretation :

```
struct opaque_auth{
        auth_type atype;
        opaque body<400>;
};
```

Each authentication method uses a specific format for encoding data. UNIX authentication defines the structure of its authentication information to contain several fields :

```
struct auth_unix {
        unsigned int timestamp;
        string smachine <255>;
        unsigned int userid;
        unsigned int grpid;
        unsigned int grpids <10>;
};
```

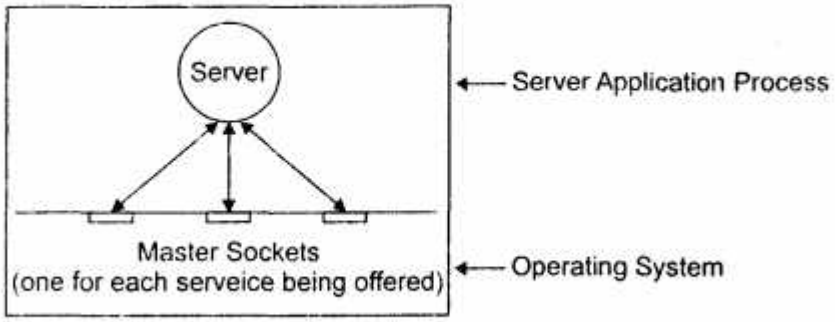
UNIX authentication relies on the client machine to supply its name in field machine and the numeric identifier of the user making the request in field userid. The client also specifies its local time in field timestamp, which can be used to sequence requests. Finally; the client sends the main numeric group identifier and secondary group identifiers of the sending user in fields grpid and grpids.

Q. 5. Explain the following :

(i) Multi service servers for TCP and UDP.

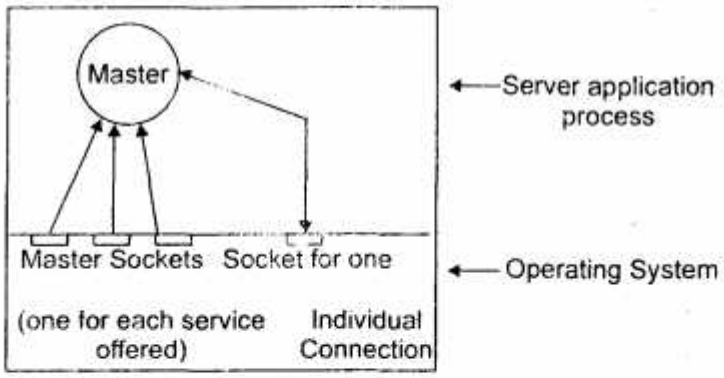
(ii) Concurrent serves for TCP and UDP. **10**

Ans. (i) A Connectionless Multiservice Server : Multi service servers can use either connectionless or connection-oriented transport protocols. Given below is process structure for a connectionless multiservice server.



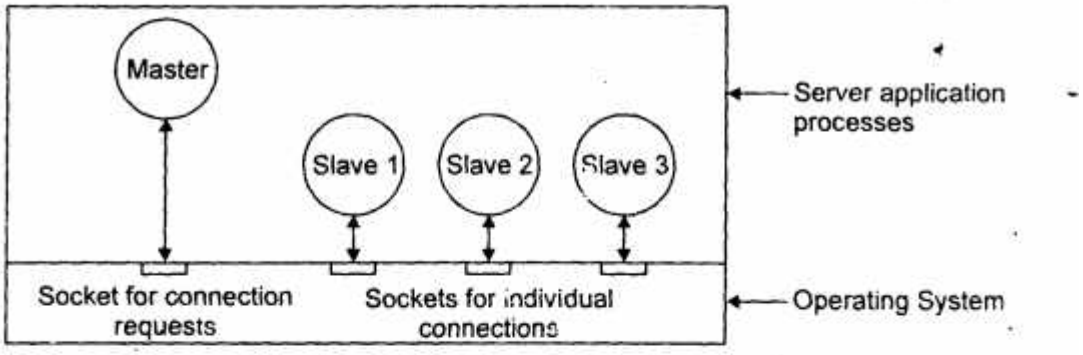
An iterative, connectionless, multiservice server usually consists of a single process that contains all the code needed for the services it supplies. The server opens a set of UDP sockets and binds each to a well-known port for one of the services being offered. It uses a small table of map sockets to services. For each socket descriptor, the table records the address of a procedure that handles the service offered on that socket. The server uses the select system call to wait for a datagram to arrive on any of the sockets. When a datagram arrives, the server calls the appropriate procedure to compute a response and send a reply. Because the mapping table records the service offered by each socket, the server can easily map the socket descriptor to the procedure that handles the service.

A Connection-Oriented, Multiservice Server : A connection-oriented, multiservice server can also follow an iterative algorithm. In principle, such a server performs the same tasks as a set of iterative, connection-oriented servers. To be more precise, the single process in a multiservice server replaces the master server processes in a set of connection-oriented servers. At the top level, the multiservices server uses asynchronous I/O to handle its duties. The following figure shows the process structure.



When it begins execution, the multiservice server creates one socket for each service it offers, binds each socket to the well-known port for the service, and uses select to wait for an incoming connection request on any of them. When one of the sockets becomes ready, the server calls accept to obtain the new connection that has arrived. Accept creates a new socket for the incoming connection. The server uses the new socket to interact with a client and then closes it. Thus, besides one master socket for each service, the server has at most one additional socket open at anytime.

(ii) Concurrent Servers : The master server process does not communicate with client directly. Instead, it merely waits at the well-known port for the next connection request. Once a request has arrived, the system returns the socket descriptor of the new socket to use for that connect. Process structure is given below



The master server process creates a slave process to create a slave process to handle the connection and allows the slave to operate concurrently. At anytime, the server consists of one master process and zero or more slave processes. To avoid using CPU resources while it waits for connections, the master server uses a block call of accept to obtain the next connection from the well-known port. The master server process in a concurrent server spends most of its time blocked in a call to accept. When a connection request arrives, the call to accept returns, allow the master process to execute. The master creates a slave to handle the request and reissues the call to accept. The call blocks the server again until another connection request arrives.

Q. 6. (a) Explain connectionless and connection oriented servers. **10**

Ans. Connectionless and Connection-Oriented Servers : Servers that use TCP are called connection-oriented servers. TCP provides all the reliability needed to communicate across the network. It verifies that data arrives and automatically retransmits segments that are not correct. It computes a checksum over the data to guarantee that it is not corrupted during transmission. It uses sequence numbers to ensure that the data arrives in order and automatically eliminates duplicate packets. It provides flow control to ensure that the sender does not transmit data faster than the receiver can consume it.

Servers that use UDP are called connectionsless servers. UPD do not guarantees reliable delivery. When a client sends a request, the request may be lost, duplicated, delayed or delivered out of order, similarly, a response the server sends back to a client may be lost, duplicated, delayed or delivered out of order. The server application programs must take appropriate actions to detect and correct such errors.

Another consideration in choosing connectionless versus connection-oriented design focuses on whether the services require broadcast or multicast communication. Because TCP offers point-to-point communication, it cannot supply broadcast or multicast communication, such services require UDP. Thus, any server that accepts or responds to multicast communication must be connectionless.

Q. 6. (b) How Routing sockets are different from originary sockets ? Explain. **10**

Ans. Traditionally, the UNIX routing table within the kernel has be accessed using ioctl commands. There is no command to dump the entire routing table and instead program such as netstat read the kernel memory to obtain the contents of the routing table. One additional piece to this hodgepadge is that routing daemons such as gated need to monitor ICMP redirect messages that are received by the kernel and they often do this by creating a raw ICMP socket and listening on this socket to all received ICMP messages.

BSD Reno cleaned up the interface to the kernel's routing subsystem by creating the AF_ROUTE domain. The only type of socket supported in the route domain is a raw socket. Three types of operations are supported in the route domain is a raw socket. Three types of operations are supported on a routing socket :

(i) A process can send a message to the kernel by writing to a routing socket. For example, this is how routes are added and deleted.

(ii) A process can read a message from the kernel on a routing socket. This is how the kernel notifies a process that an ICMP redirect has been received and processed, or how it requests a route resolution from an external routing process.

(iii) A process can use the sysctl function to either dump the routing table or list all configured interfaces.

Q. 7. Explain the following Addresses scheme : **20**

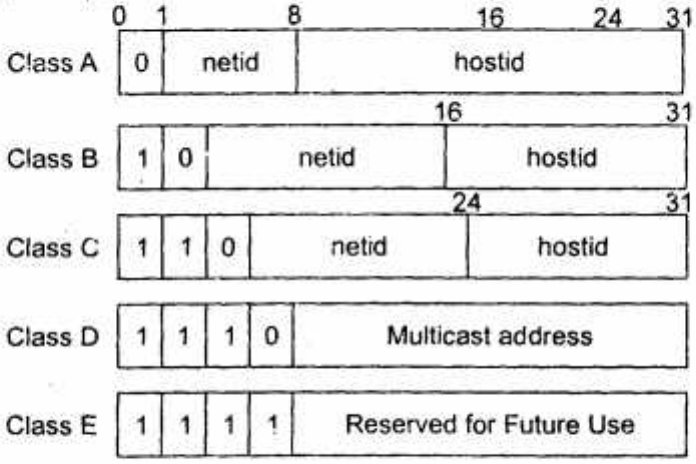
(a) Internet adobesses

(b) Subnet adobesses

(c) Super net adobessing

How ARP works ? Explain.

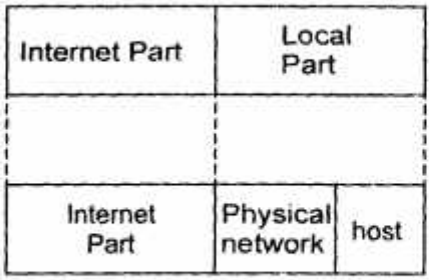
Ans. (a) Internet Addresses : Each host on a TCP/IP internet is assigned a unique 32-bit internet address that is used in all communication with that host. A prefix of an IP address or internet address identifies a network. That is, the IP addresses in all hosts on a given network share a common prefix. Conceptually, each address is a pair (netid, hostid), where netid identifies a network and hostid identifies the host on that network.



(b) Subnet Addresses : Subnet addressing is used to allow a single network address to span multiple physical networks. Subnetting is most widely used because it is most general and standardized. The easiest way to understand subnet addressing is to imagine that a site has a single class B IP network address assigned to it, but it has two or more physical networks. Only local routers know that there are mutliple physical nets and how to route traffic among them; routers in other autonomous system route all traffic as if there were a single physical network.

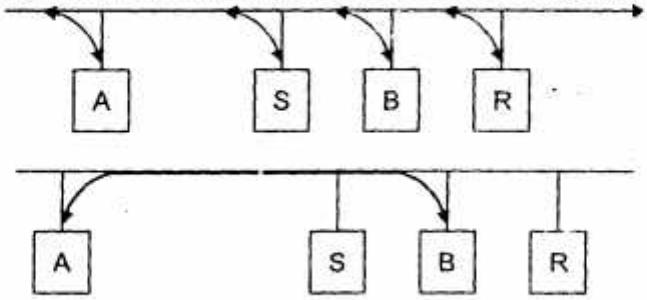
Conceptually, adding subnets only changes the interpretation of IP addresses slightly. Instead of dividing the 32 bit IP address into a network prefix and host suffix, subnetting divides the address into a

network portion and a local portion. The interpretation of the network portion remains the same as for networks that do not use subnetting.



(c) Supernet Addresses : Classless addressing, supernet addressing or supernetting is the scheme that takes an approach that is complementary to subnet addressing. Instead of using a single IP network prefix for multiple physical network at a given organization, supernetting allows the addresses assigned to a single organization to span multiple classed prefixes.

ARP : ARP stands for Address Resolution Protocol. ARP helps to solve the address resolution problem. ARP protocol provides a mechanism that is both efficient and easy to maintain.



When host A wants to resolve IP address I_B, it broadcasts a special packet that asks the host with IP address I_B to respond with its physical address, P_B. All hosts, including B receive the request, but only host B recognizes its IP address and sends a reply that contains its physical address. When A receives the reply it uses the physical address to send the packet directly to B.

Q. 8. Write short notes : **5 × 4 = 20**

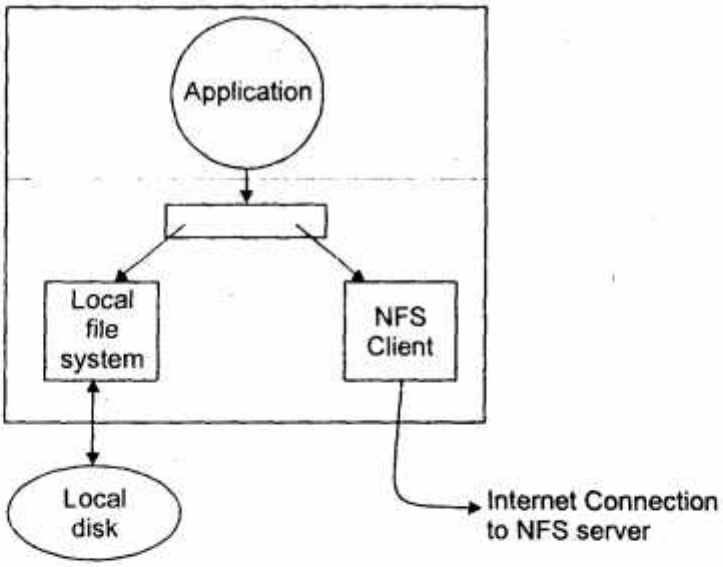
(a) Network File System

(b) TRACERT

(c) RPC Model

(d) Select and Poll

Ans. (a) Network File System : When an application program executes, it calls the operating system to open a file, or to store and retrieve data in files. The file access mechanism accepts the request and automatically passes it to either the local file system software or to the NFS client, depending on whether the file is on the local disk or a remote machine. When it receives a request, the client software uses the NFS protocol to contact the appropriate server on a remote machine and perform the requested operation. When the remote server replies, the client software returns the results to the application program.



(b) TRACERT : TRACERT stands for **Traceroute**. Traceroute lets us to determine the path that IP datagrams follow from our host to some other destination. Traceroute uses the IP_{V4} TTL field or IP_{V6} hop limit field and two ICMP messages. It starts by sending a UDP datagram to the destination with a TTL of 1. This datagram causes the first-hop router to return an ICMP "time exceeded in transit" error. The TTL is then increased by one and another UDP datagram is sent, which locates the next router in the path when the UDP datagram reaches the final destination, the goal is to have that host return an ICMP "port unreachable" error. This is done by sending the UDP datagram to a random port that is not in use on that host.

(c) RPC Model : When programmers build a clinet-server application, they cannot focus exclusively on one component at a time. Instead, they must consider how the entire system will function and how the two components will interact. To help programmers design and understand client-server interaction, researchers have devised a conceptual framework for building distributed programs. Known as the remote procedure call model or RPC model, the framework uses familiar concepts from conventional programs as the basis for the design of distributed applications.

There are two paradigms for building distributed programs :

(i) Communication-Oriented Design : Begin with the communication protocol. Design a message format and syntax. Design the client and server components by specifying how each reacts to incoming messages and how each generates outgoing messages.

(ii) Application-Oriented Design : Begin with the application design a conventional application program to solve the problem. Build and test a working version of the conventional program that operates on a single machine. Divide the program into two or more pieces and add communication protocols that allow each piece to execute on a separate computer.

(d) Select and Poll : Select and poll are I/O multiplexing functions.

Select function allows the process to instruct the kernel to wait for any one of multiple events to occur and to wake up the process only when one or more or these events occurs or when a specified amount of time has passed. That is, we tell the kernel what descriptors we are interested in {for reading, writing, or an

exception condition} and how long to wait. The descriptors in which we are interested are not restricted to sockets; any descriptor can be tested using select.

There are three possibilities :

(i) Wait Forever : Return only when one of the specified descriptors is ready for I/O. For this, we specify the timeout argument as a null pointer.

(ii) Wait upto a Fixed Amount of Time : Return when one of the specified descriptors is ready for I/O, but do not wait beyond the number of seconds and microseconds specified in the timeval structure pointed to by the timeout argument.

(iii) Do not Wait at All : Return immediately after checking the descriptors. This is called polling. To specify this, the timeout argument must point to a timeval structure and timer value must be o.

Poll function originated with SVR3 {System V Release 3} and was originally limited to STREAMs devices. SVR4 removed this limitation, allowing poll to work with any descriptor. Poll provides functionality that is similar to select, but poll provides additional information when dealing with STREAMs devices. The first argument is a pointer to the first element of an array of structures. Each element of the array is a pollfd structure that specifies the conditions to be tested for a given descriptor, fd.

```
struct pollfd{
        int fd;
        short events;
        short revents;
```