

Q1.Describe the following:

1)ARP

2) ELECTRONIC MAIL

3) ERROR DETECTION & RECOVERY

4)FLOW CONTROL

5)SNMP

ANS:1

Address Resolution Protocol (arp)

The address resolution protocol (arp) is a protocol used by the [Internet Protocol \(IP\) \[RFC826\]](#), specifically IPv4, to map [IP network addresses](#) to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when [IPv4 is used over Ethernet](#).

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

An [Ethernet](#) network uses two hardware addresses which identify the source and destination of each frame sent by the [Ethernet](#). The destination address (all 1's) may also identify a [broadcast](#) packet (to be sent to all connected computers). The hardware address is also known as the [Medium Access Control \(MAC\) address](#), in reference to the standards which define [Ethernet](#). Each computer [network interface card](#) is allocated a globally unique 6 byte link address when the factory manufactures the card (stored in a PROM). This is the normal link source address used by an interface. A computer sends all packets which it creates with its own hardware source link address, and receives all packets which match the same hardware address in the destination field or one (or more) pre-selected broadcast/multicast addresses.

Electronic Mail

One of the most prominent uses of networking since the first networks were devised, has been electronic mail. It started as a simple service that copied a file from one machine to another, and appended it to the recipient's *mailbox* file. Basically, this is still what email is all about, although an ever growing net with its complex routing requirements and its ever increasing load of messages has made a more elaborate scheme necessary.

Various standards of mail exchange have been devised. Sites on the Internet adhere to one laid out in RFC-822, augmented by some RFCs that describe a machine-independent way of transferring special characters, and the like. Much thought has also been given recently to "multi-media mail", which deals with including pictures and sound in mail messages. Another standard, X.400, has been defined by CCITT.

Quite a number of mail transport programs have been implemented for systems. One of the best-known is the University of Berkeley's sendmail, which is used on a number of platforms. The original author was Eric Allman, who is now actively working on the sendmail team again. There are two ports of sendmail-5.56c available, one of which will be described in chapter-11. The sendmail version currently being developed is 8.6.5.

The mail agent most commonly used with is smail-3.1.28, written and copyrighted by Curt Landon Noll and Ronald S.-Karr. This is the one included in most distributions. In the following, we will refer to it simply as smail, although there are other versions of it which are entirely different, and which we don't describe here.

Compared to sendmail, smail is rather young. When handling mail for a small site without complicated routing requirements, their capabilities are pretty close. For large sites, however, sendmail always wins, because its configuration scheme is much more flexible.

Both smail and sendmail support a set of configuration files that have to be customized. Apart from the information that is required to make the mail subsystem run (such as the local hostname), there are many more parameters that may be tuned. sendmail's main configuration file is very hard to understand at first. It looks as if your cat had taken a nap on your keyboard with the shift key pressed. smail configuration files are more structured and easier to understand than sendmail's, but don't give the user as much power in tuning the mailer's behavior. However, for small UUCP or Internet sites the work required in setting up any of them is roughly the same.

Error detection and Recovery

The Run Control system is responsible for error logging and it is the central system which should cope in an expeditious manner with various types of failures, including failures attributable to other systems such as the trigger, readout or slow control system, etc. Depending on the type of error, emergency instructions or dedicated recovery procedures can be invoked either automatically or by requesting the operator intervention, for instance, to ignore errors, to try again,

after a relatively brief period of coexistence, SNMP Version 2 (SNMPv2) will largely replace SNMP Version 1 (SNMPv1). SNMP is part of a larger architecture called the Internet *Network Management Framework* (NMF), which is defined in Internet documents called *requests for comments* (RFCs). The SNMPv1 NMF is defined in RFCs 1155, 1157, and 1212, and the SNMPv2 NMF is defined by RFCs 1441 through 1452.

Today, SNMP is the most popular protocol for managing diverse commercial internetworks as well as those used in universities and research organizations. SNMP-related standardization activity continues even as vendors develop and release state-of-the-art, SNMP-based management applications. SNMP is a relatively simple protocol, yet its feature set is sufficiently powerful to handle the difficult problems presented in trying to manage today's heterogeneous networks.

SNMP Technology

SNMP is part of the Internet network management architecture. This architecture is based on the interaction of many entities, as described in the following section.

The Internet Management Model

As specified in Internet RFCs and other documents, a network management system comprises:

- *Network elements* -- Sometimes called *managed devices*, network elements are hardware devices such as computers, routers, and terminal servers that are connected to networks.
- *Agents* -- Agents are software modules that reside in network elements. They collect and store management information such as the number of error packets received by a network element.
- *Managed object* -- A managed object is a characteristic of something that can be managed. For example, a list of currently active TCP circuits in a particular host computer is a managed object. Managed objects differ from variables, which are particular object instances. Using our example, an object instance is a single active TCP circuit in a particular host computer. Managed objects can be scalar (defining a single object instance) or tabular (defining multiple, related instances).
- *Management information base* (MIB) -- A MIB is a collection of managed objects residing in a virtual information store. Collections of related managed objects are defined in specific MIB modules.
- *Syntax notation* -- A syntax notation is a language used to describe a MIB's managed objects in a machine-independent format. Consistent use of a syntax notation allows different types of computers to share information. Internet management systems use a subset of the International Organization for Standardization's (ISO's) *Open System Interconnection* (OSI) *Abstract Syntax Notation 1* (ASN.1) to define both the packets exchanged by the management protocol and the objects that are to be managed.
- *Structure of Management Information* (SMI) -- The SMI defines the rules for describing management information. The SMI is defined using ASN.1.
- *Network management stations* (NMSs) -- Sometimes called *consoles*, these devices execute management applications that monitor and control network elements. Physically, NMSs are usually engineering workstation-caliber computers with fast CPUs, megapixel color

displays, substantial memory, and abundant disk space. At least one NMS must be present in each managed environment.

Q2. Outline the generation of three-way handshaking?

Ans: TCP Three Way Handshake

The TCP three way handshakes is the process for establishing a TCP connection. A TCP connection is established as shown in the below example. In this example, we assume a client computer is contacting a server to send it some information.

1. The client sends a packet with the SYN bit set and a sequence number of N.
2. The server sends a packet with an ACK number of N+1, the SYN bit set and a sequence number of X.
3. The client sends a packet with an ACK number of X+1 and the connection is established.
4. The client sends the data.

The first three steps in the above process is called the three way handshake which is used to establish a TCP connection.

Q3. Explain the two main reasons for using layered protocol architecture for network communication?

Ans: Protocol architecture or TCP/IP protocol architecture also referred to as the TCP/IP protocol suite is a simple fundamental underlying design which obeys a set of rules and conventions in which communication tasks are to be performed.

The communication tasks of the protocol architecture are organized into five relatively independent layers:

- Physical
- Network Access Layer
- Internet layer
- Host-to-host, or transport layer
- Application Layer

In my research to this question it was interested to find out that there were some who consider the protocol architecture to be comprised of only four relative independent layers which are:

- Network Access Layer
- Internet layer
- Host-to-host, or transport layer
- Application Layer

Q4. Write a small report to list the features of different application protocols of the TCP/IP protocol suite?

Ans: *Data Encapsulation and the TCP/IP Protocol Stack*

The packet is the basic unit of information transferred across a network, consisting, at minimum, of a header with the sending and receiving hosts' addresses, and a body with the data to be transferred. As the packet travels through the TCP/IP protocol stack, the protocols at each layer either add or remove fields from the basic header. When a protocol on the sending host adds data to the packet header, the process is called *data encapsulation*. Moreover, each layer has a different term for the altered packet. This section summarizes the life cycle of a packet from the time the user issues a command or sends a message to the time it is received by the appropriate application on the receiving host.

Application Layer--The User Initiates Communication

The packet's history begins when a user on one host sends a message or issues a command that must access a remote host. The application protocol associated with the command or message formats the packet so that it can be handled by the appropriate transport layer protocol, TCP or UDP.

Suppose the user issues an `rlogin` command to log in to the remote host. The `rlogin` command uses the TCP transport layer protocol. TCP expects to receive data in the form of a stream of bytes containing the information in the command. Therefore, `rlogin` sends this data as a *TCP stream*.

Not all application layer protocols use TCP, however. Suppose a user wants to mount a file system on a remote host, thus initiating the NFS application layer protocol. NFS uses the UDP transport layer protocol. Therefore, the packet containing the command must be formatted in a manner that UDP expects. This type of packet is referred to as a *message*.

Transport Layer --Data Encapsulation Begins

When the data arrives at the transport layer, the protocols at the layer start the process of data encapsulation. The end result depends on whether TCP or UDP has handled the information.

TCP Segmentation TCP is often called a "connection-oriented" protocol because it ensures the successful delivery of data to the receiving host. How the TCP protocol receives the stream from the `rlogin` command. TCP divides the data received from the application layer into *segments* and attaches a header to each segment.

Segment headers contain sender and recipient ports, segment ordering information, and a data field known as a *checksum*. The TCP protocols on both hosts use the checksum data to determine whether data has transferred without error.

Establishing a TCP Connection In addition, TCP uses segments to determine whether the receiving host is ready to receive the data. When the sending TCP wants to establish connections, it sends a segment called a SYN to the peer TCP protocol running on the receiving host. The receiving TCP returns a segment called an ACK to acknowledge the successful receipt of the segment. The sending TCP sends another ACK segment and then proceeds to send the data. This exchange of control information is referred to as a *three-way handshake*.

UDP Packets UDP is a "connectionless" protocol. Unlike TCP, it does not check to make sure that data arrived at the receiving host. Instead, UDP takes the message received from the application layer and formats it into **UDP packets**. UDP attaches a header to each packet, which contains the sending and receiving host ports, a field with the length of the packet, and a checksum.

The sending UDP attempts to send the packet to its peer UDP process on the receiving host. The receiving UDP may send a response as an acknowledgment that it received the data. If the sending UDP doesn't get a response, it sends the packet again. UDP does not use the three-way handshake concept.

Internet Layer

As shown in both TCP and UDP pass their segments and packets down to the Internet layer, where they are handled by the IP protocol. IP prepares them for delivery by formatting them into IP datagram's. Then IP determines the IP addresses for the datagram's, so they can effectively be delivered to the receiving host.

IP Datagram's IP attaches an **IP header** to the segment or packet's header in addition to the information added by TCP or UDP. Information in the IP header includes the IP addresses of the sending and receiving hosts, datagram length, and datagram sequence order. This is provided in case the datagram exceeds the allowable byte size for network packets and must be fragmented.

Data Link Layer--Framing Takes Place

Data link layer protocols such as PPP format the IP datagram into a **frame**. They attach a third header and a footer to "frame" the datagram. The frame header includes a cyclical redundancy check (**CRC**) field that checks for errors as the frame travels over the network media. Then the data link layer passes the frame to the physical layer.

Physical Network Layer--Preparing the Frame for Transmission

The physical network layer on the sending host receives the frames and converts the IP addresses into the hardware addresses appropriate to the network media. The physical network layer then sends the frame out over the network media.

How the Receiving Host Handles the Packet

When the packet arrives on the receiving host, it travels through the TCP/IP protocol stack in the reverse order from that which it took on the sender illustrates this path. Moreover, each protocol on the receiving host strips off header information attached to the packet by its peer on the sending host. Here is what happens.

1. Physical Network Layer

Receives the packet in its "frame" form. It converts the hardware addresses of the sender and recipient into IP addresses and then sends the frame to the data link layer.

2. Data Link Layer

Verifies that the CRC for the frame is correct and strips off the frame header and CRC. Finally, the data link protocol sends the frame to the Internet layer.

3. Internet Layer

Reads information in the header to identify the transmission and determine if it is a fragment. If the transmission was fragmented, IP reassembles the fragments into the original datagram. It then strips off the IP header and passes the datagram on to transport layer protocols

4. Transport Layer (TCP and UDP)

Reads the header to determine which application layer protocol must receive the data. Then TCP or UDP strips off its related header and sends the message or stream up to the receiving application.

5. Application Layer

Receives the message and performs the operation requested by the sending host.

Q5. Draw tcp header structure and describe its fields?

Ans: the structure and content of TCP/IP e-mail messages. The structure is intentionally designed to be very simple and easy to both create and understand. Each message begins with a set of headers that describe the message and its contents. An empty line marks the end of the headers, and then the message body follows.

The message body contains the actual text that the sender is trying to communicate to the recipient(s), while the message header contains various types of information that serve various purposes. The headers help control how the message is processed, by specifying who the recipients are, describing the contents of the message, and providing information to a recipient of a message about processing done on the message as it was delivered.

Header Field Structure

Each header field follows the simple text structure we saw in the preceding topic:

<header name>: <header value>

The <header name> is of course the name of the header, and the <header value> is the value associated with that header, which depends on the header type. Like all RFC 822 lines, headers must be no more than 998 characters long and are recommended to be no more than 78 characters in length, for easier readability. The RFC 822 and 2822 standards [support](#) a special syntax for allowing headers to be “folded” onto multiple lines if they are very lengthy. This is done by simply continuing a header value onto a new line,

which *must* begin with at least one “white space” character, such as a space or <Tab> character, like this:

<header name>: <header value part 1>

<white space> <header value part 2>

<white space> <header value part 3>