Network Software:

A telecommunications network contains hardware and software components that need to work together to transmit information. Most networks are organized as a series of layers or levels, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer differ from network to network, however in all networks the purpose of each layer is:

1- Offer services to higher layer.

2- Shielding higher layers from how offered services are implemented.

Layer (n) on one machine carries a conversation with layer (n) on another machine. The rules used in this conversation are known as the layer (n) protocol.

Protocol: is an agreement between the communicating parties on how communication is to proceed.

Protocols usually exist in two forms. First, they exist in a textual form for humans to understand. Second, they exist as programming code for computers to understand. Both forms should ultimately specify the precise interpretation of every bit of every message exchanged across a network.

Protocols exist at every point where logical program flow crosses between hosts. In other words, we need protocols every time we want to do something on another computer. Every time we want to print something on a network printer we need protocols. Every time we want to download a file we need protocols.

Usually multiple protocols will be in use simultaneously. For one thing, computers usually do several things at once, and often for several people at once. Therefore, most protocols support multitasking. Also, one operation can involve several protocols. For example, consider the NFS (Network File System) protocol. A write to a file is done with an NFS operation, that uses another protocol (RPC) to perform a function call on a remote host, that uses another protocol (UDP) to deliver a datagram to a port on a remote host, that uses

another protocol to deliver a datagram on an Ethernet, and so on. Along the way we made need to lookup host names (using the DNS protocol), convert data to a network standard form (using the XDR protocol), find a routing path to the host (using one or many of numerous protocols).

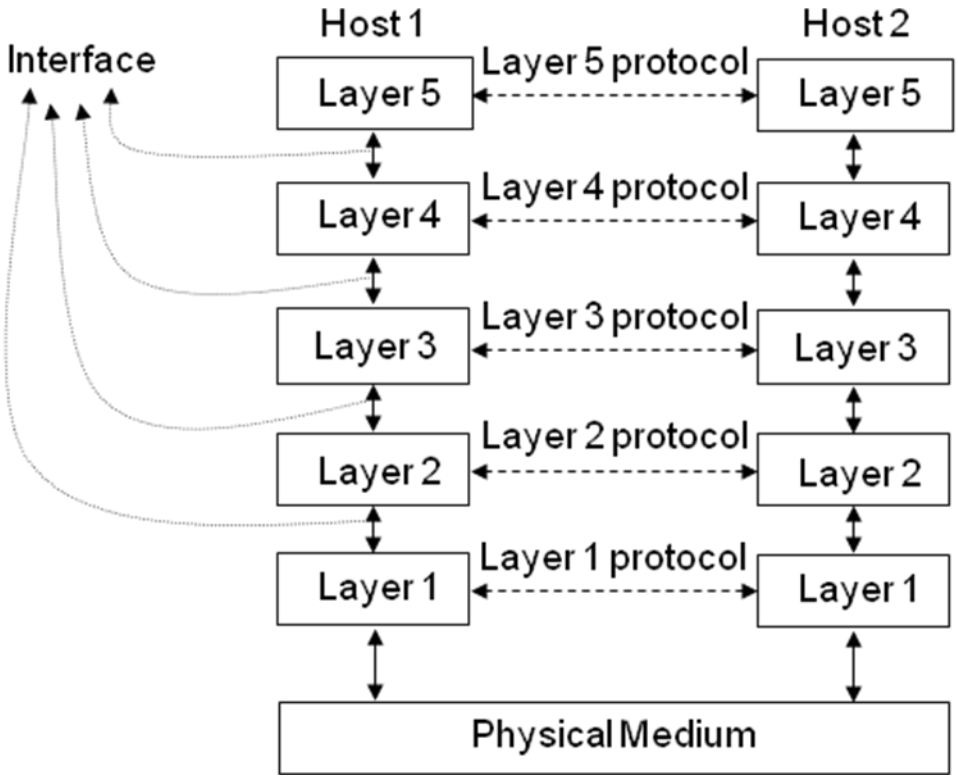
Protocol Layering:

Protocol layering is a common technique to simplify networking designs by dividing them into functional layers, and assigning protocols to perform each layer's task. For example, it is common to separate the functions of data delivery and connection management into separate layers, and therefore separate protocols. Thus, one protocol is designed to perform data delivery, and another protocol, layered above the first, performs connection management. The data delivery protocol is fairly simple and knows nothing of connection management. The connection management protocol is also fairly simple, since it doesn't need to concern itself with data delivery.

Protocol layering produces simple protocols, each with a few well-defined tasks. These protocols can then be assembled into a useful whole. Individual protocols can also be removed or replaced.

The most important layered protocol designs are the Internet's original DoD model, and the OSI Seven Layer Model. The modern Internet represents a fusion of both models.

The figure shown below, will illustrate layers, protocols, and interfaces:



Between each pair of adjacent layers there is an interface.

Interface: Primitive operations and services the lower layer offers to upper. The purpose of interface:

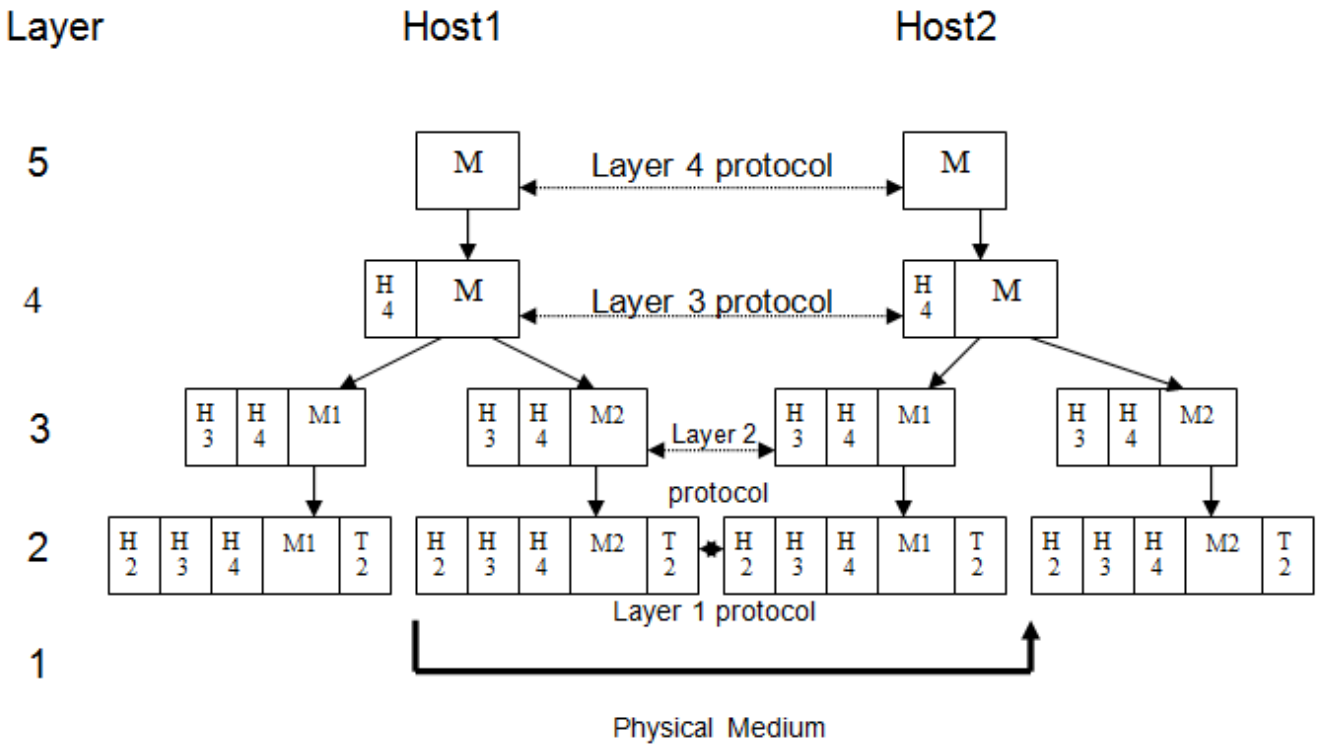
1-Minimizing amount of information that must be passed between layers.

2-Simpler to replace the implementation of one layer with completely different implementation (clean cut).

Note:

In reality, no data are directly transferred from layer (n) on one machine to layer (n) on another machine, instead each layer passes data and control information to the layer immediately below it, until the lowest layer is reached below layer1 is the physical medium through which actual communication occurs. So in previous figure the virtual communication is shown by dotted lines and physical communication by solid lines.

Technical Example:



Note: A set of layers and protocols are called network architecture.

Types of Services:

Layer can offer two different types of services:

1- Connection-Oriented Service:

It is modeled after telephone system. The service user first establishes a connection, uses the connection, and then release the connection.

2- Connectionless Service:

It is modeled after postal system. Each message carries the full destination address, and each one is routed through the system independent of all the others. If two messages are sent to same destination the first one will be the first one arrive but may be the first message delayed and the second message be the first, This is impossible in connection-oriented service.

Quality of Service (QoS):

Service can be characterized by quality of service. The service is reliable if the data are never lost.

Reliable: The sender be sure that the message is received by receiver, this is done id the sender take the receiver acknowledge of reception on each message.

Typical situation for reliable connection-oriented service is file transfer (the owner of the file wants to be sure that all the bits arrive correctly).

Service Primitives (Operations):

A service is specified by a set of primitives (operations) available to a user or other entity to access the service. Service primitives are divided into four classes as shown in table below:

Primitive	Meaning
Request	An entity wants the service to do some work
Indication	An entity is to be informed about an event
Response	An entity wants to respond to an event
Confirm	The response to an earlier request has come back

These four classes illustrate how connection is established and released. The initiating entity does a (Connect.request) primitive which results in a packet being sent. The receiver then gets a (Connect. Indication) primitive announcing that an entity somewhere wants to setup a connection to it. The entity getting the (Connect. Indication) primitive then uses the (Connect. response) primitive to tell whether it wants to accept or reject proposed connection. The entity issuing the initial (Connect. request) primitive finds out what happened via (Connect. confirm) primitive.

Primitives have parameters, such as:

(Connect. request) has the following parameters:

1- Specify the machine to connect to.
2- Type of service desired.
3- Maximum message size.

(Connect. Indication) has the following parameters:

1- Caller's identity.
2- Type of service desired.
3- Proposed maximum message size.

Note: The details of negotiation (for example about maximum message size) are part of protocol. If the called entity did not agree to the proposed maximum size, it could make proposal in response primitive which would be made available in confirm primitive. If conflicting proposals about maximum message size the protocol might specify the smaller value is always chosen.

Service can be either confirmed or unconfirmed. Confirmed service has the primitives: request, indication, response, and confirm. Unconfirmed only has request and indication primitives. Connect is always confirmed service because remote peer must agree to establish the connection. Data transfer can be either confirmed or unconfirmed depending on or not the sender needs an acknowledgement.

Data Link Control

Introduction:

Communication requires at least two devices working together, one to send and one to receive. Even such a basic arrangement requires a great deal of coordination an exchange to occur. For example, in half duplex transmission, it is essential that only one device transmit at a time. If both ends of the link put signals on the line simultaneously, they collide, leaving nothing on the line but noise. The coordination of half duplex transmission is part of a procedure called line discipline. In addition to line discipline, data link control includes the functions: flow control and error control.

Line Discipline:

Line discipline answers the question, who should send now?

Line discipline can be done in two ways:

1- ENQuiry/ACKnowledgment (**ENQ/ACK**):

ENQ/ACK is used primarily in systems where there is no question of the wrong receiver getting the transmission, that is, when there is a dedicated link between two devices so that the only device capable of receiving the transmission is the intended one.

ENQ/ACK coordinates which device may start a transmission and whether or not the intended recipient is ready and enabled.

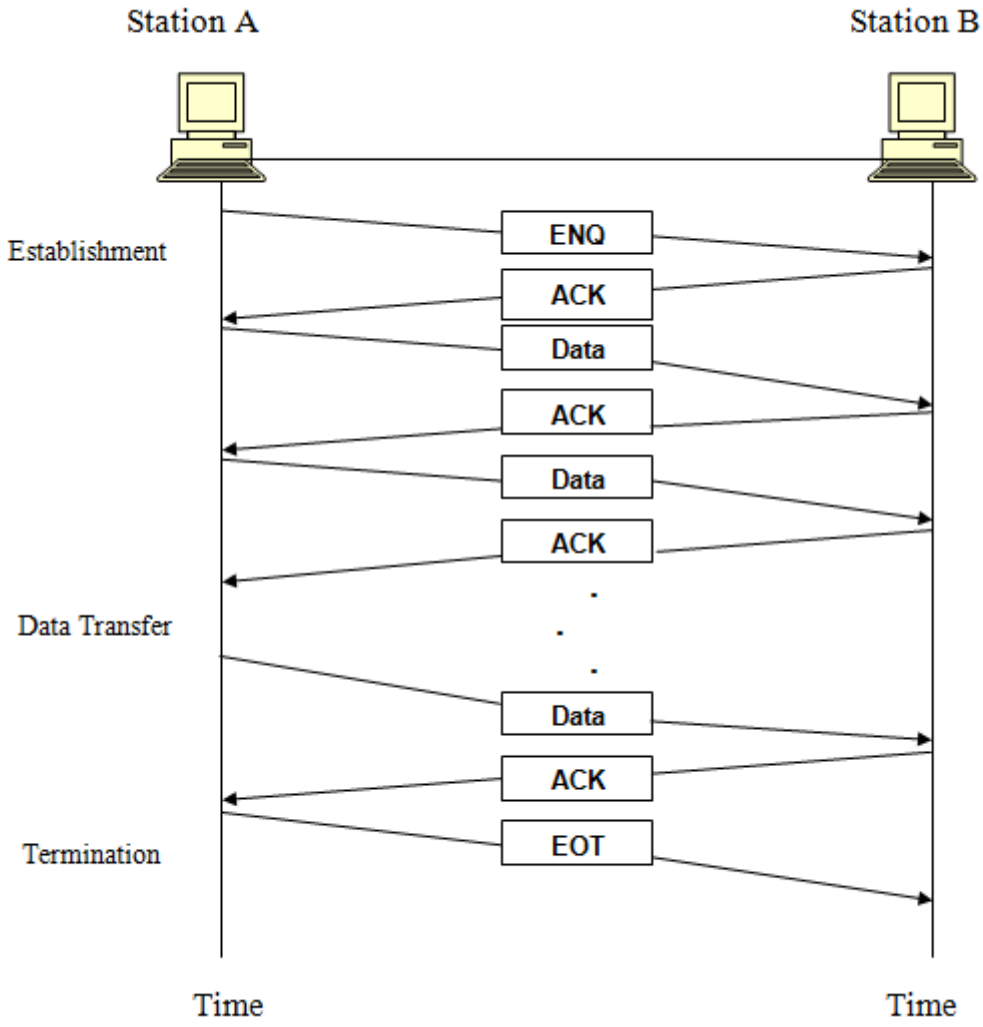
How it Works:

The initiator first transmits a frame called an enquiry (ENQ) asking if the receiver is available to receive data. The receiver must answer either with an acknowledgement (ACK) frame if it is ready to receive or with a negative acknowledgment (NAK) frame if it is not.

By requiring a response even if the answer is negative, the initiator knows that its enquiry was in fact received even if the receiver is currently unable to accept a transmission. If neither an ACK nor a NAK is received within a specified time limit, the initiator assumes

that the ENQ frame was lost in transmit, disconnects, and sends a replacement. An initiating system ordinarily makes three such attempts to establish a link before giving up.

If the response to the ENQ is negative for three attempts, the initiator disconnects and begins the process again at another time. If the response is positive, the initiator is free to send its data. Once all of its data have been transmitted, the sending system finishes with an end of transmission (EOT) frame. This process is illustrated in following figure.

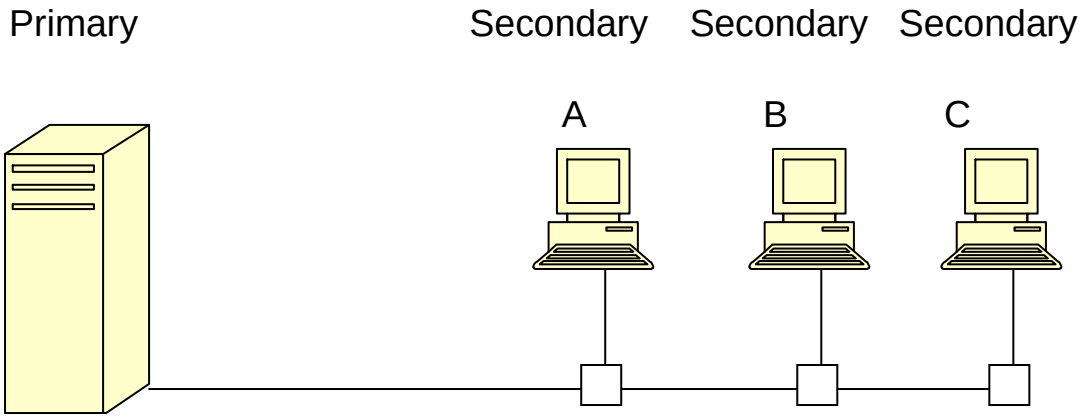


2- Poll/Select:

The poll/select method of line discipline works with topologies where one device is designed as a primary station and the other devices are secondary stations. Multipoint systems must coordinate several nodes, not just two. The question to be determined in these cases, therefore, is more than just, Are you ready? It is also, which of the several nodes has the right to use the channel?

How it Works:

Whenever a multipoint link consists of a primary device and multiple secondary devices using a single transmission line, all exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary to determine which device is allowed to use the channel at a given time as shown in following figure.



Who has the right to the channel?

The primary, therefore, is always the initiator of a session. If the primary wants to receive data, it asks the secondaries if they have anything to send; this function is called polling. If the primary wants to send data, it tells the target secondary to get ready to receive; this function is called selecting.

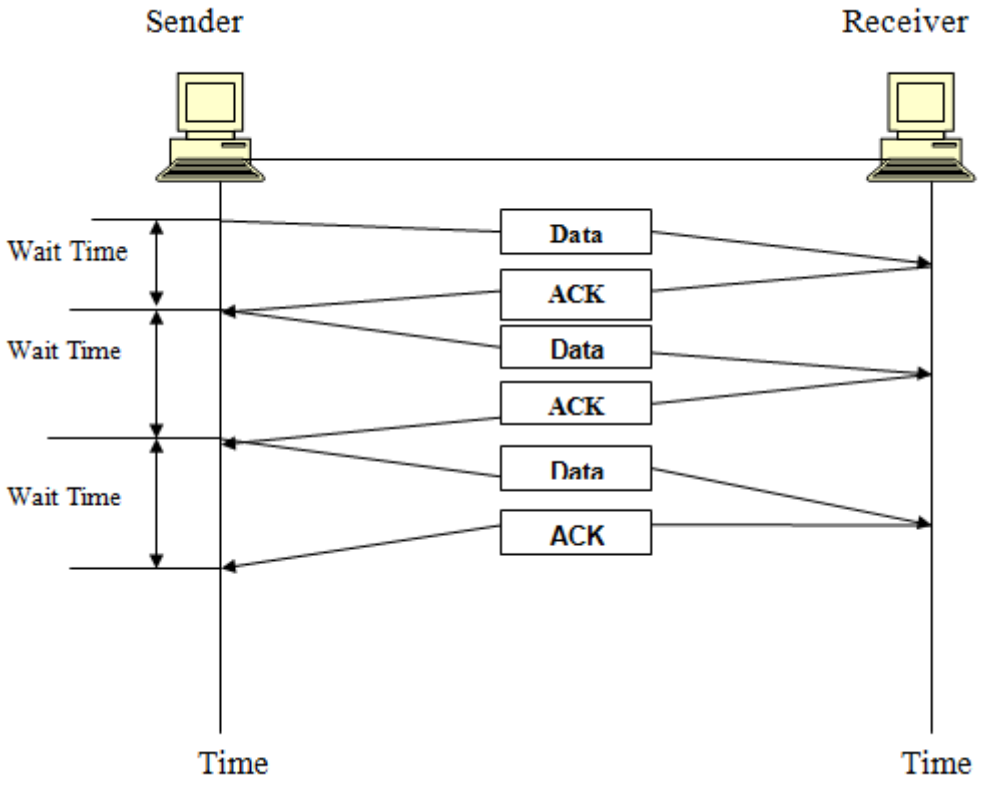
Every device on a link has an address that can be used for identification. Poll/select protocols identify each frame as being either to or from a specific device on the link. Each secondary device has an address that differentiate it from the others.

Flow Control:

Flow control refers to a set of procedures used to restrict the amount of data the sender can send waiting for acknowledgment. Two methods have been developed to control the flow of data across communications links:

1- Stop and Wait:

In the stop and wait method of flow control, the sender sends one frame and waits for an acknowledgment before sending the next frame, as shown in following figure.



The advantage of stop and wait is simplicity: each frame is checked and acknowledged before the next frame is sent. The disadvantage is inefficiency: stop and wait is slow. Each frame is alone on the line. If the distance between devices is long, the time spent waiting for ACKs between each frame can add significantly to the total transmission time.

2- Sliding Window:

In the sliding window method of flow control, several frames can be in transmit at a time. The sliding window refers to imaginary boxes at both the sender and receiver. This window can hold frames at either end and provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment. Frames may be acknowledged at any point frames at either end and provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment. Frames may be acknowledged at any point without waiting for the window to fill up and may be transmitted as long as the window is not yet full. To keep track of which frames have been transmitted and which received, sliding window introduces an identification scheme based on the size of the window. The frames are numbered modulo-n, which means they are numbered from 0 to n-1. For example, if n=8, the frames are numbered 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, ….. The size of the window is n-1 (in this case 7), In other words, the window cannot cover the whole modulo (8 frames); it covers one frame less.

When the receiver sends an ACK, it includes the number of the next frame it expects to receive. In other words, to acknowledge the receipt of a string of frames ending in frame 4, the receiver sends an ACK containing the number 5. When the sender an ACK with the number 5, it knows that all frames up through number 4 have been received.

The window can hold n-1 frames at either end; therefore, a maximum of n-1 frames may be sent before an acknowledgement is required.

Conceptually, the sliding window of the sender shrinks from the left when frames of data are sent. The sliding window of the sender expands to the right when acknowledgment are received.

Conceptually, the sliding window of the receiver shrinks from the left when frames of data are received. The sliding window of the receiver expands to the right when acknowledgment are sent.