Chapter Seven

Data Link Control and Protocols Data Link Layer

7.1- Introduction

- The two main functions of the data link layer are **data link control** and **media access control.** The first, *data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication. The second function of the data link layer is media access control, or how to share the link.*
- Data link control functions include ***framing, flow and error control,*** *and* ***software implemented protocols*** that provide smooth and reliable transmission of frames between nodes. To implement data link control, we need protocols. Each protocol is a set of rules that need to be implemented in software and run by the two nodes involved in data exchange at the data link layer.

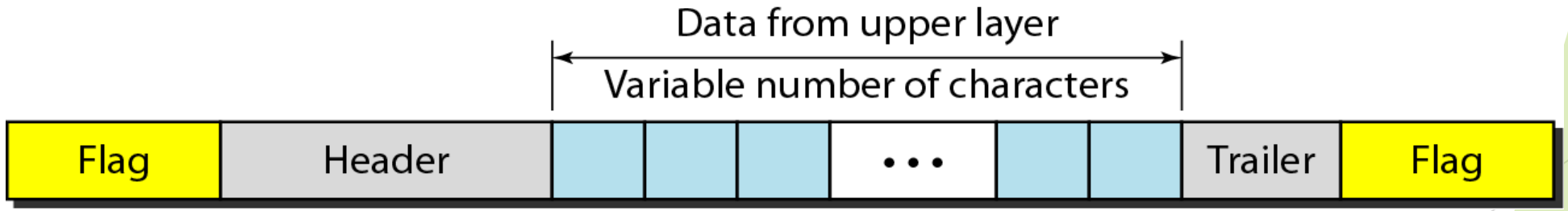
7.2- Framing

The data link layer pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. Framing in the data link layer separates a message from one source to a destination, *by adding* ***a sender address and a destination address***. *The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt*. Although the whole message could be packed in one frame; this is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message. When a message is divided into smaller frames, a single-bit error affects only that small frame.

- **Fixed-Size Framing**
- **Variable-Size Framing**

7.3 Character-Oriented Protocols

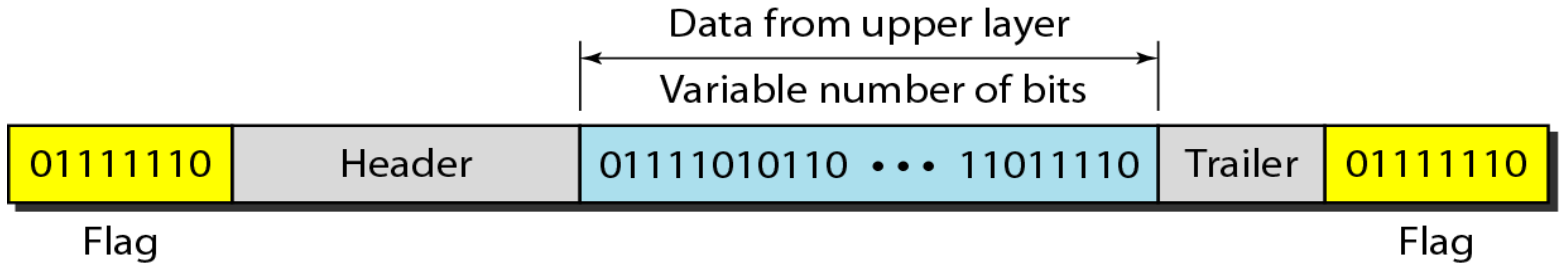
- In a character-oriented protocol, *data to be carried are 8-bit characters* from a coding system such as American Standard Code for Information Interchange (ASCII) code. *The **header**, which normally carries the source and destination addresses and other control information*, and *the **trailer**, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit **flag** is added at the beginning and the end of a frame*. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure (1) shows the format of a frame in a character-oriented protocol.



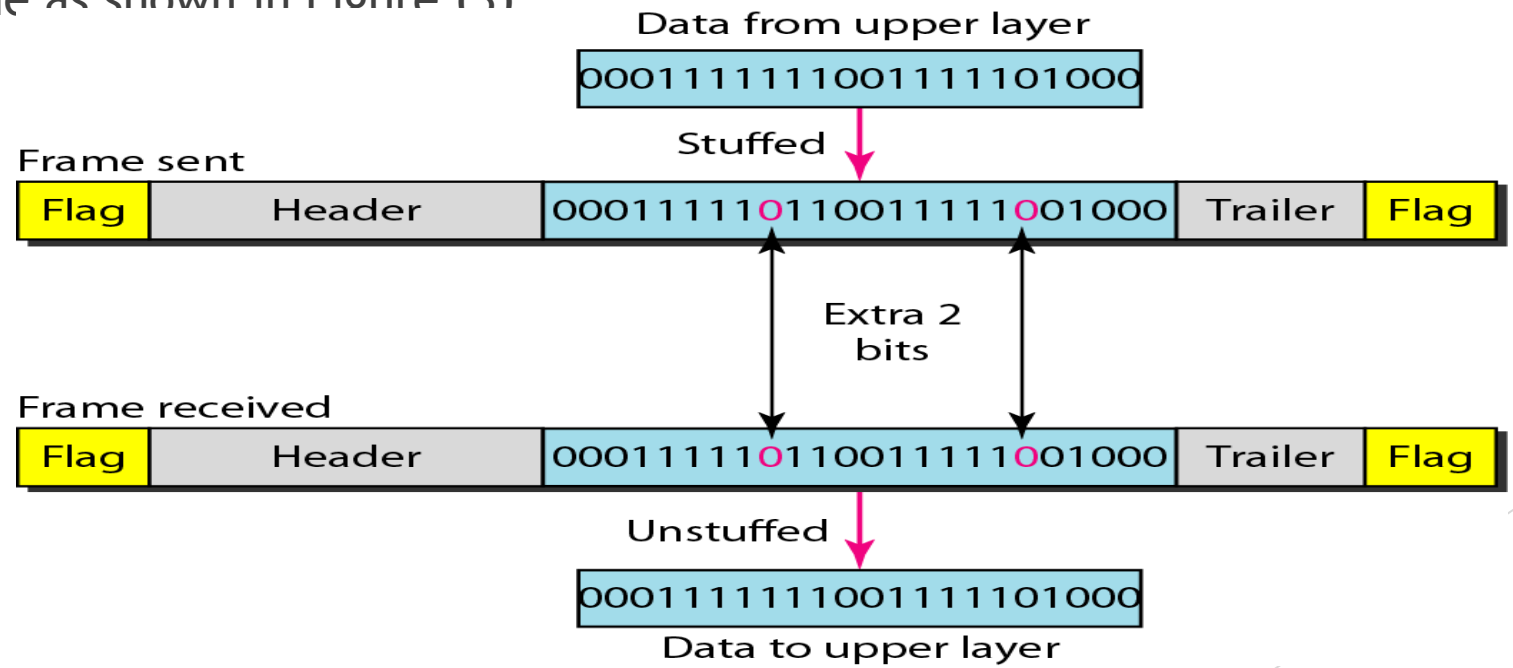
Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication.

7.4 Bit-Oriented Protocols

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame, as shown in Figure (2).



- If the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by ***stuffing*** *1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag*. The strategy is called ***bit stuffing***. *In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added*. This extra stuffed bit is eventually removed from the data by the receiver. *Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit*. This guarantees that the flag field sequence does not inadvertently appear in the frame as shown in Figure (3)



7.5- Flow and Error Control

- The most important responsibilities of the data link layer are ***flow control*** **and** ***error control***. Collectively, these functions are known as **data link control**.

7.5.1- Flow Control

- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before receiving an acknowledgment and is one of the most important duties of the data link layer. The flow of data must not be allowed to overwhelm the receiver. Any receiving device has a limited speed at which it can *process* incoming data and a limited amount of memory in which to *store* incoming data. *The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily*. Incoming data must be checked and processed before they can be used. *The rate of such processing is often slower than the rate of transmission.* For this reason, each receiving device has a block of memory, called a **buffer**, reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

7.5.2- Error Control

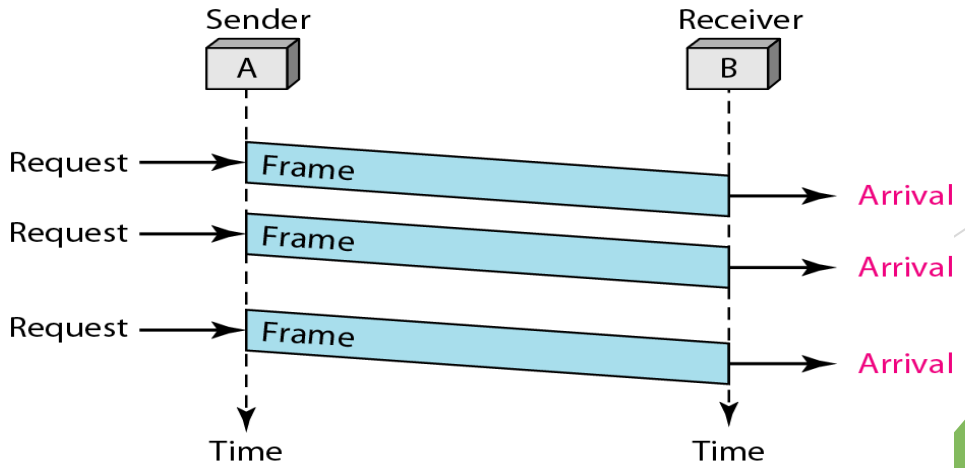
- Error control is both *error detection* and *error correction*. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. *Error control in the data link layer is often implemented simply. Any time an error is detected in an exchange, specified frames are retransmitted. This process is called* ***automatic repeat request (ARQ).***

7.6- Protocols

- Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are divided into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels.

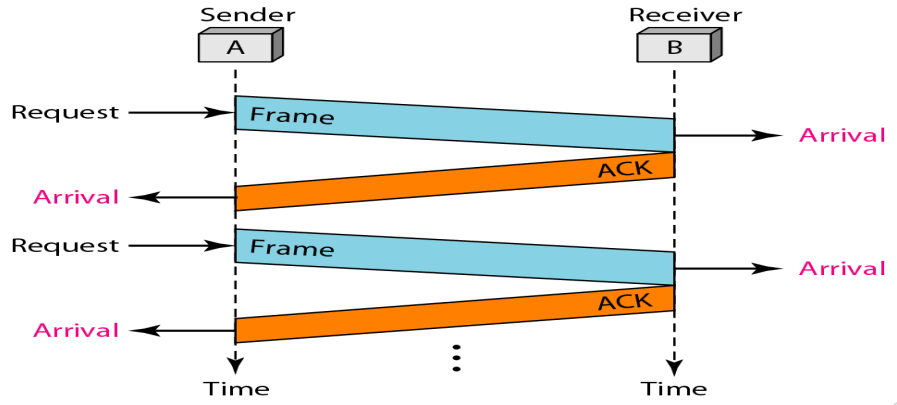
(A) Simplest Protocol

- It is a unidirectional protocol in which data frames are traveling in only one direction from the sender to receiver. We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.
- Figure (4) shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site.



(B) Stop-and-Wait Protocol

- If data frames arrive at the receiver site *faster than they can be processed*, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.
- The protocol we discuss now is called the **Stop-and-Wait Protocol** because *the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.* We add flow control to our previous protocol.
- Figure (5) illustrates the mechanism of this protocol. At any time, there is either one data frame on the forward channel or one **acknowledgment (ACK)** frame on the reverse channel. We therefore need a half-duplex link.



7.6.1- Noiseless Channels Protocols

- Noiseless channels are the ideal channel in which no frames are lost, duplicated, or corrupted. We will introduce two protocols for this type of channel. The first is a protocol that does not use flow control; the second is the one that does. Of course, neither has error control because we have assumed that the channel is a perfect noiseless channel.

7.6.2- Noisy Channels Protocols

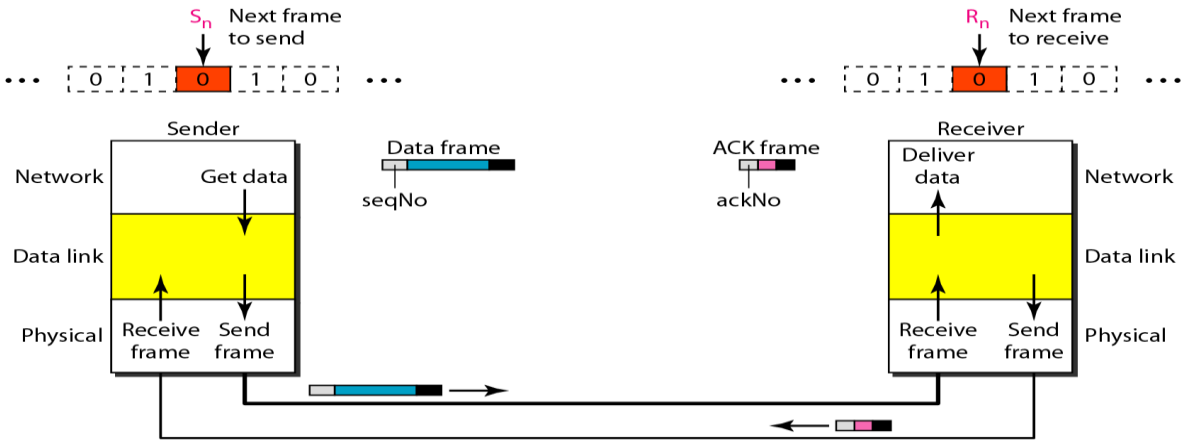
- Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent (i.e. the error control are ignored). We discuss three protocols in this section that use error control.

(A) Stop-and-Wait ARQ

Our first protocol, called the Stop-and-Wait ***Automatic Repeat Request*** (ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and at the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted.

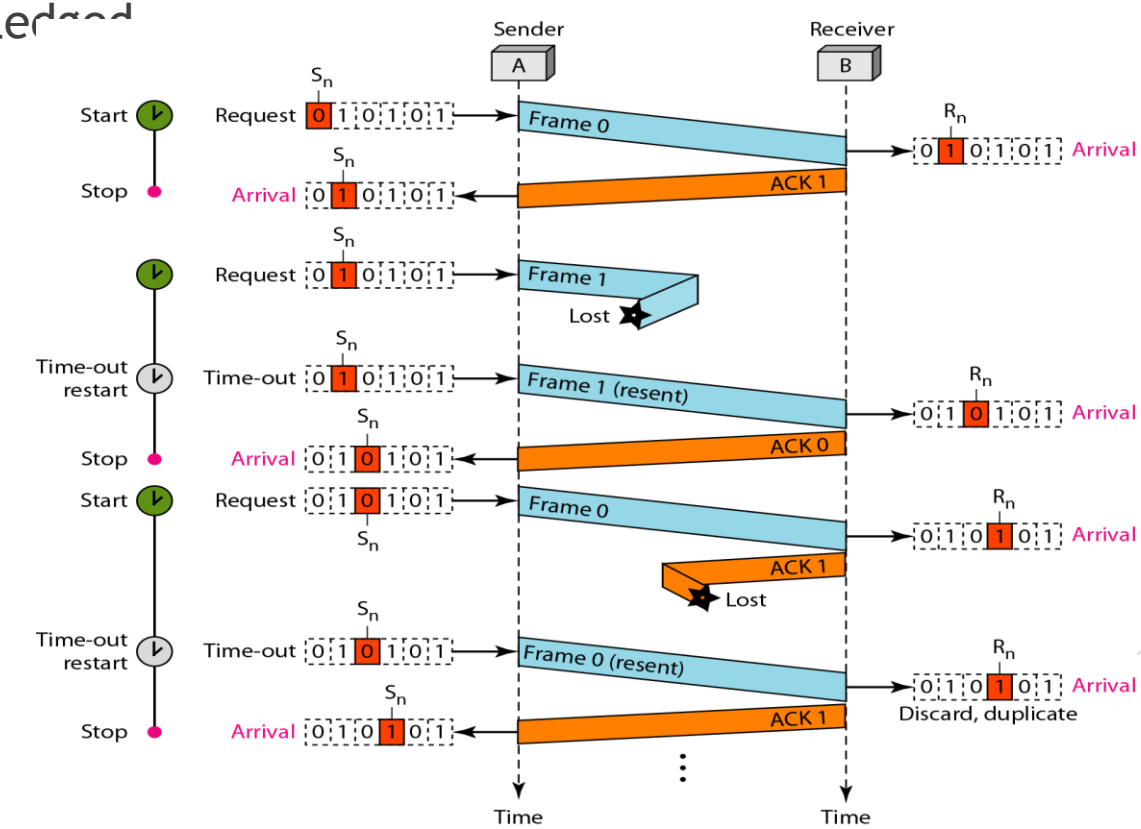
- ***Sequence Numbers***
- ***Operation***



Operation of the Stop-and-Wait ARQ Protocol

EXAMPLE (1)

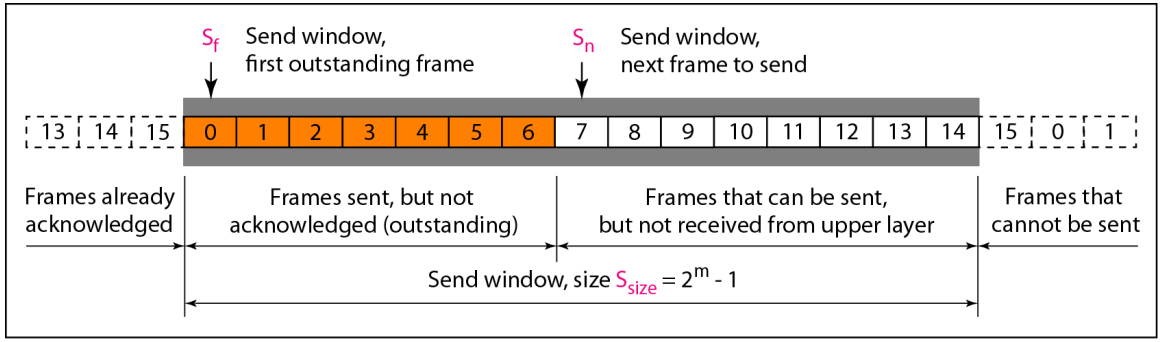
- Figure (7) shows an example of Stop-and-Wait ARQ. Frame (0) is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame (0) is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame (0), which is acknowledged



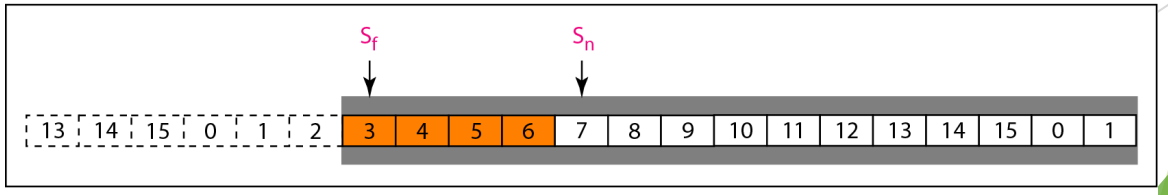
(B) Go-Back-N ARQ

To improve the efficiency of transmission, multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment. With Go-Back-N ARQ we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

- ***Sequence Numbers***
- ***Sender Sliding Window***



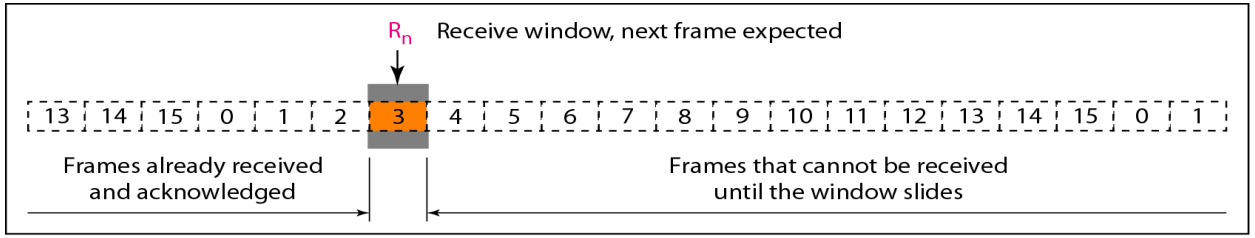
a. Send window before sliding



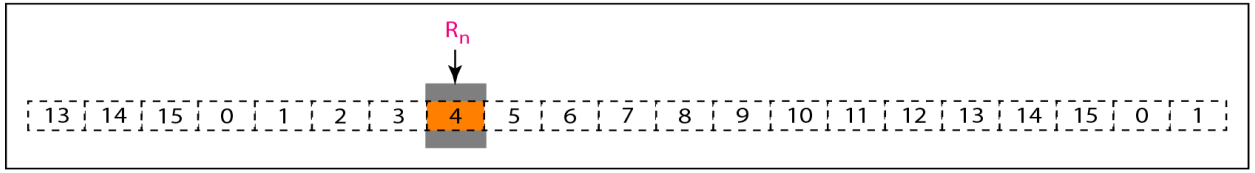
b. Send window after sliding

****Send window for Go-Back-NARQ***

- *Receiver Sliding Window*



a. Receive window

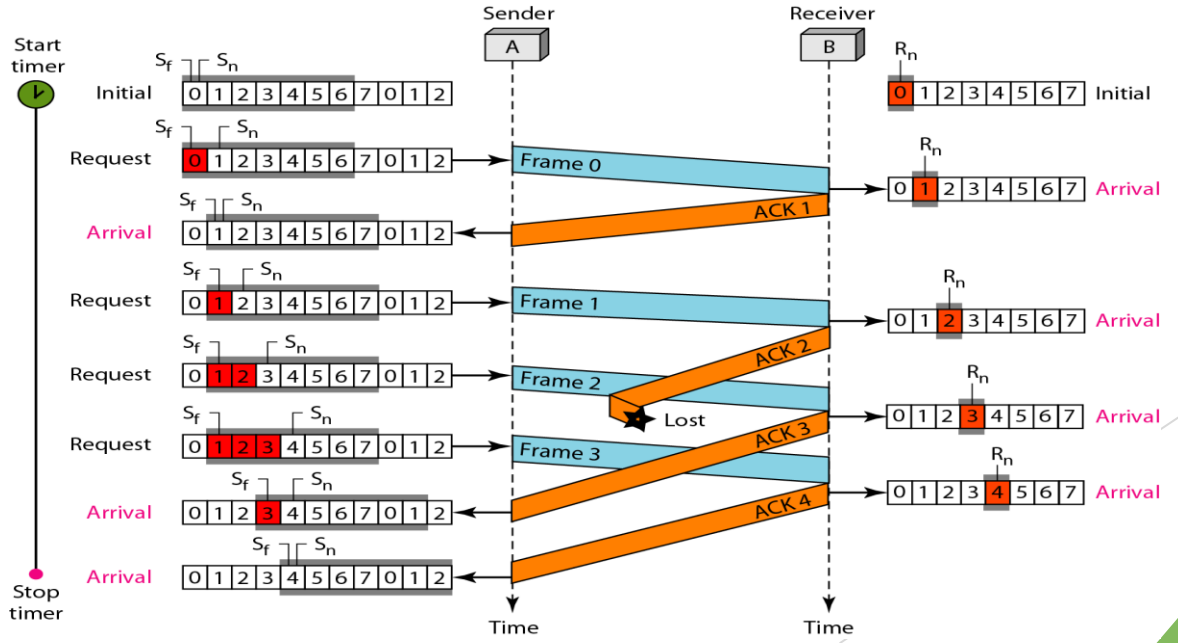


b. Window after sliding

- *Acknowledgment*
- *Resending a Frame*
- *Operation*

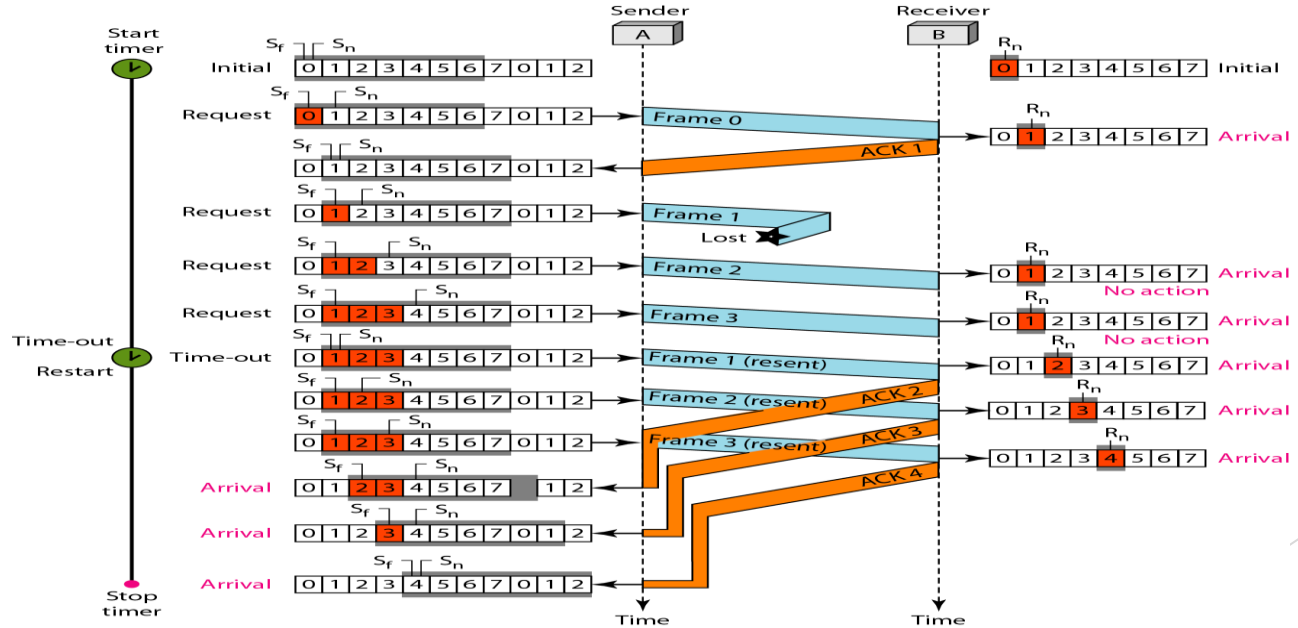
EXAMPLE (2)

- Figure (11) shows an example of Go-Back-*N*. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK2 is lost, ACK3 serves as both ACK2 and ACK3. There are four receiver events, all triggered by the arrival of frames from the physical layer.



EXAMPLE (3)

- Figure (12) shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order (frame 1 is expected). The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames l, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with ending frame 3. The physica1layer must wait until this event is completed and the data link layer goes back to its sleeping state. We have shown a vertical line to indicate the delay. It is the same story with ACK 3; but when ACK 3 arrives, the sender is busy responding to ACK 2. It happens again when ACK 4 arrives. Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.

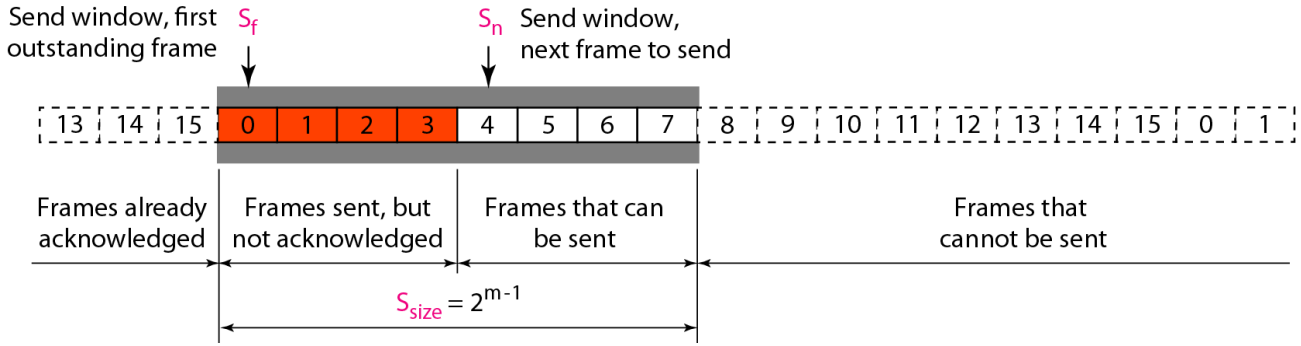


7.7 Selective Repeat ARQ

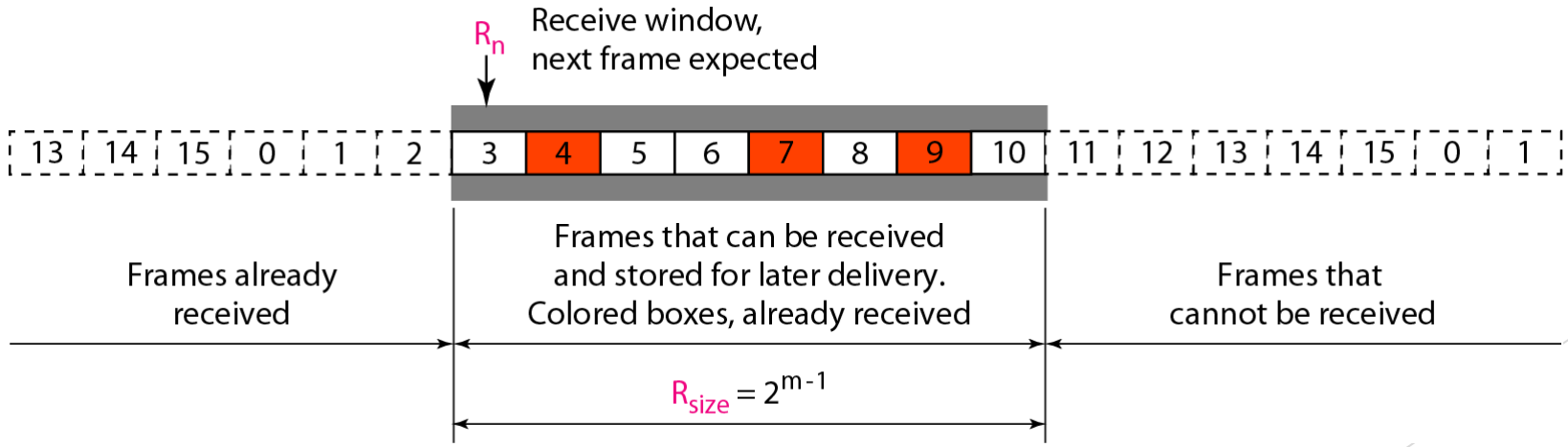
- *Go-Back-N* ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. *This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called* ***Selective Repeat ARQ****. It is more efficient for noisy links, but the processing at the receiver is more complex.*

7.7.1 Windows

- The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are *differences* between the windows in this protocol and the ones in Go-Back-N. ***First**, the size of the send window is much smaller; it is 2^{m-1}. **Second**, the receive window is the same size as the send window*. The send window maximum size can be 2^{m-1}. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the *Go-Back-N* Protocol). The protocol uses the same variables as we discussed for Go-Back-N. We show the Selective Repeat send window in Figure



- The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. We need, however, to mention that the receiver never delivers packets out of order to the network layer. Figure (14) shows the receive window in this protocol. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

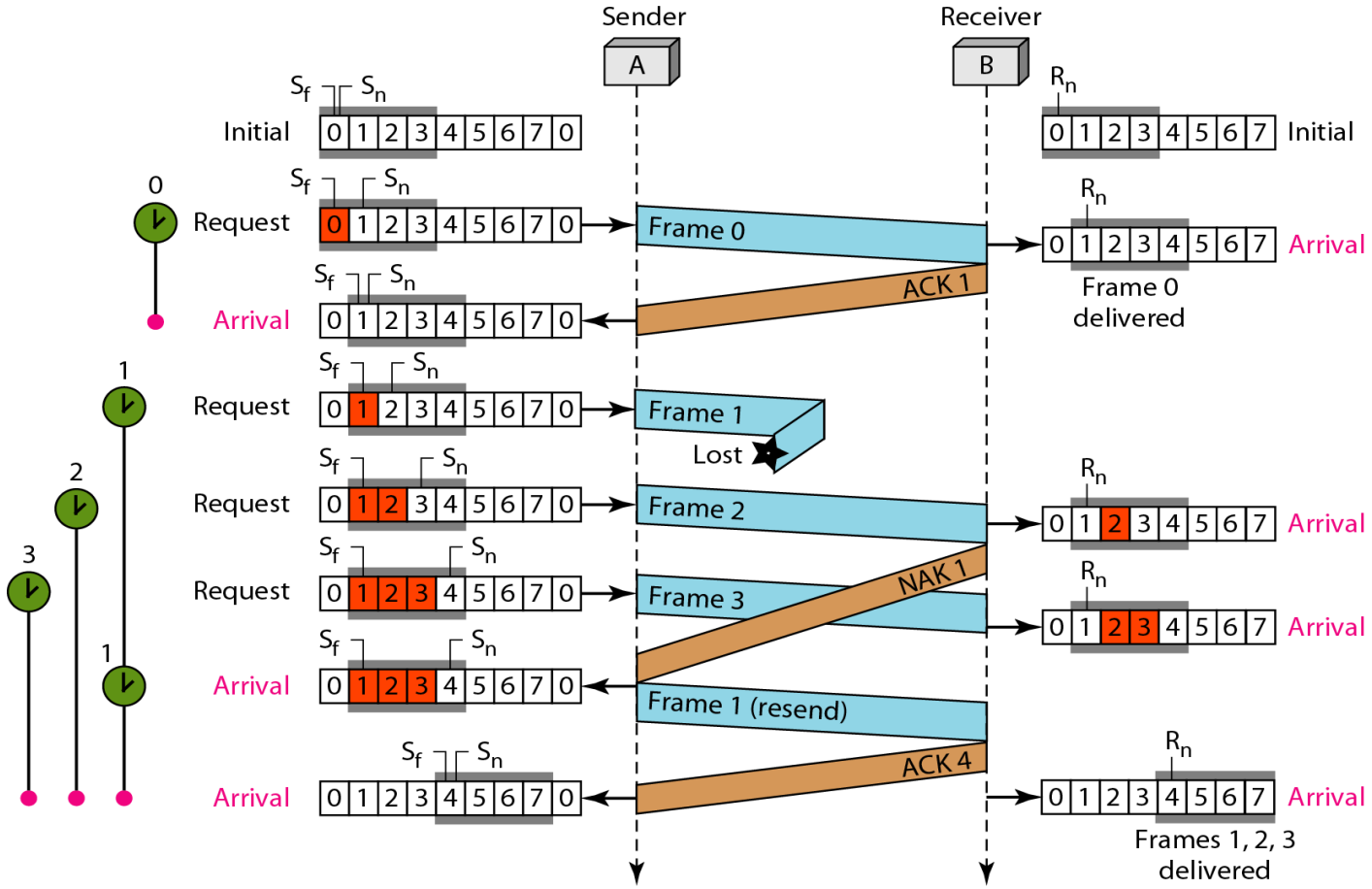


EXAMPLE (4)

This example is similar to Example (1) in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure (15) shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2 and 3).

The timer for frame starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked (colored slot), but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer.



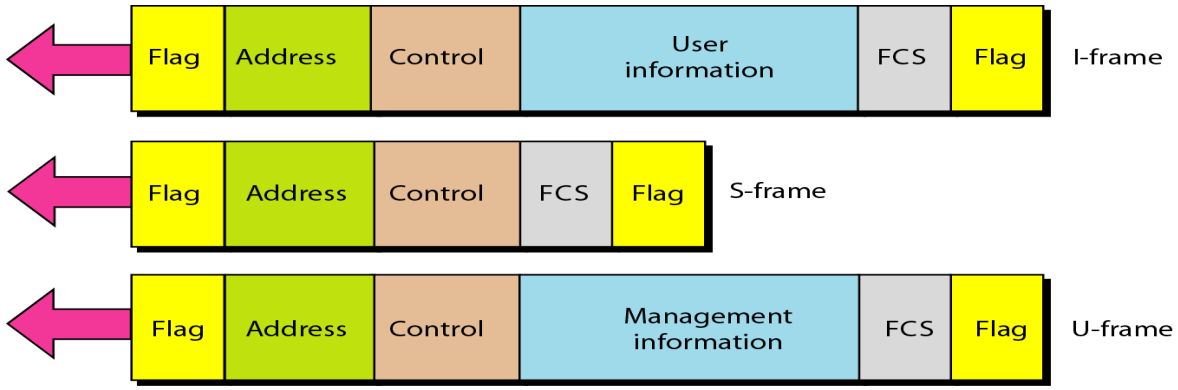
- Another important point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to *n* frames are delivered in one shot, only one ACK is sent for all of them.

7.8 HDLC

- High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms.

7.9- Frames

- To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: information frames **(I-frames)**, supervisory frames **(S-frames)**, and unnumbered frames **(U-frames)** as shown in Figure (16). Each type of frame serves as an envelope for the transmission of a different type of message.



- ***I-frames*** *are used to transport user data and control information relating to user data*. ***S-frames*** *are used only to transport control information.* ***U-frames*** *are reserved for system management*. *Information carried by U-frames is intended for managing the link itself.*
-
- ***Frame Format***
- *Each frame in HDLC may contain up to six fields:* ***a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field.*** In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.
-
- ***Fields***
- Let us now discuss the fields and their use in different frame types:
- **Flag field:** The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.
- **Address field:** The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a ***to*** **address**. If a secondary creates the frame, it contains *a* ***from*** **address**. An address field can be 1 byte or several bytes long, depending on the network size.
- **Control field:** The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type.
- **Information field:** The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- **FCS field:** The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

7.10 Point-to-Point_Protocol

- Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, one of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP)**. Today, millions of Internet users who need to connect their home computers to the server of an ***Internet Service Provider*** **(ISP)** use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point-to-point protocol at the data link layer. PPP is by far the most common.
- PPP provides several services:
- PPP defines the format of the frame to be exchanged between devices.
- PPP defines how two devices can negotiate the establishment of the link and the exchange of data.
- PPP defines how network layer data are encapsulated in the data link frame.
- PPP defines how two devices can authenticate each other.
- PPP provides multiple network layer services supporting a variety of network layer protocols.
- PPP provides connections over multiple links.
- PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

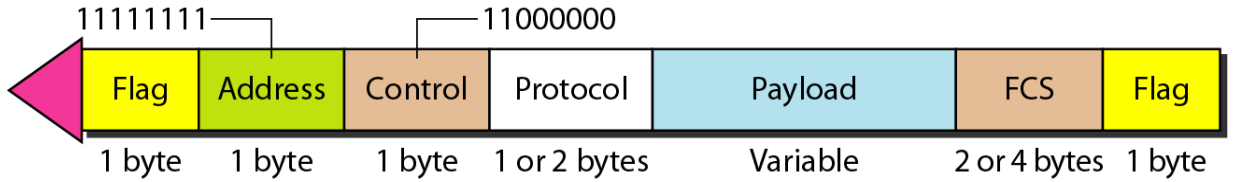
- On the other hand, to keep PPP simple, several services are missing:
- PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.
- PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order.
- PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

7.10.1- Framing

PPP is a byte-oriented protocol. Framing is done according to the discussion of byte oriented protocols at the previous chapter.

- ***Frame Format***

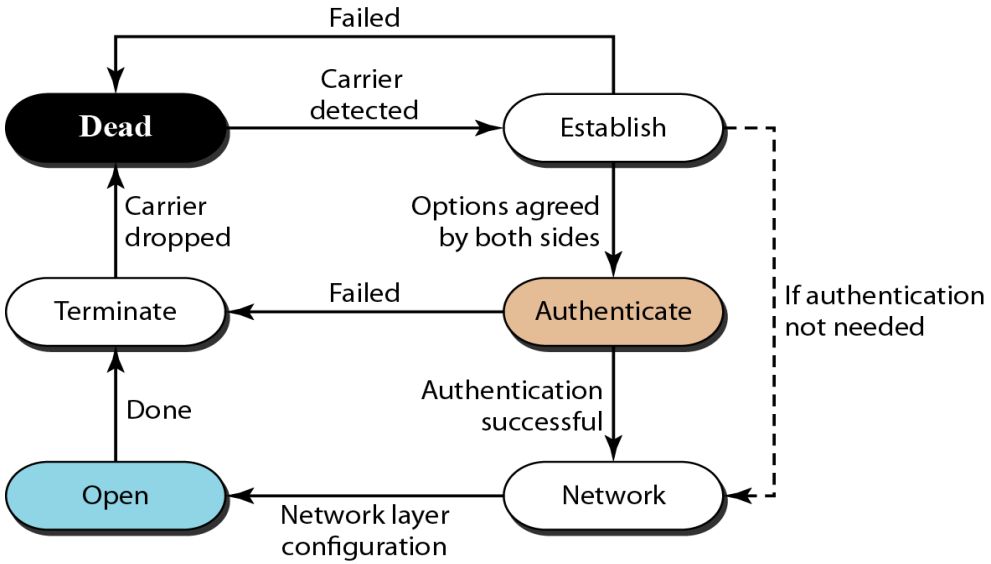
Figure (6.1) shows the format of a PPP frame. The description of each field follows:



- Flag
- Address
- Control
- Protocol
- Payload field
- FCS

7.10.2 - Transition Phases

- A PPP connection goes through phases which can be shown in a transition phase diagram as shown in Figure (6.2)



7.11- Multiple Accesses Protocols

- When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link.
- The problem of controlling the access to the medium is similar to the rules of speaking in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, and do not monopolize the discussion, and so on.
- The situation is similar for multipoint networks. Many formal protocols have been devised to handle access to a shared link.

7.11.1- Random Access

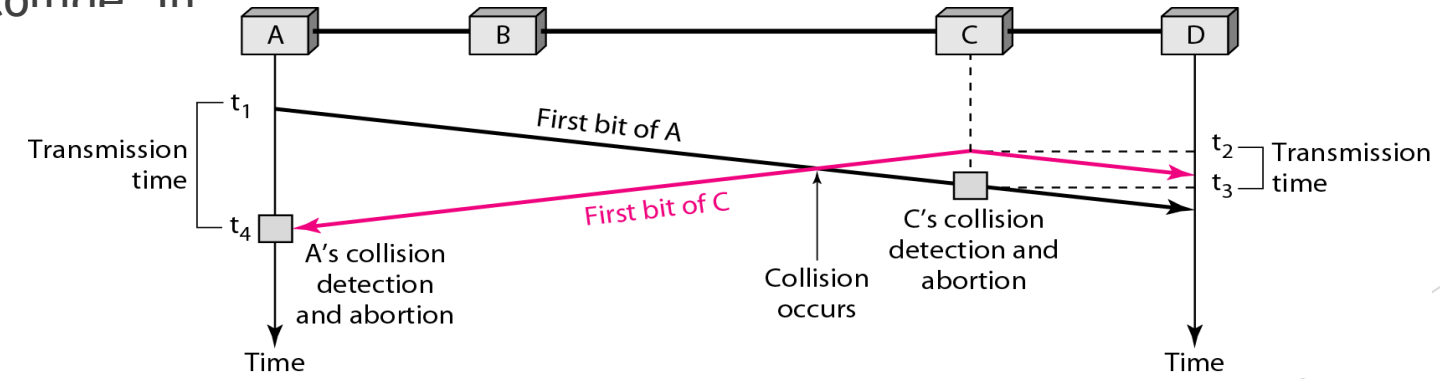
- In random access, *no station is superior to another station and none is assigned the control over another*. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including the testing of the state of the medium. There is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*.
- In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict-*collision*-and the frames will be either destroyed or modified.
- The random access methods have evolved from a very interesting protocol known as **ALOHA**, which used a very simple procedure called **Multiple Access** (MA). The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called **Carrier Sense Multiple Access** (CSMA).

7.11.1.1 Carrier Sense Multiple Access (CSMA)

- To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "*sense before transmit*" or *"listen before talk*".
- CSMA can *reduce* the possibility of collision, but it *cannot eliminate* it. The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

7.11.1.2 Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- The CSMA method does not specify the procedure following a collision. Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision. In this method, *a station monitors the medium after it sends a frame to see if the transmission was successful*. If so, the station is finished. If, however, there is a collision,the frame is sent again.
- To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In



stations A and C are involved in the collision.

7.11.1.3 Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

- The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: *its own signal*. When there is a collision, the station receives two signals: *its own signal* and *the signal transmitted by a second station*.
- In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles. However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection. Therefore, we need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance *(CSMA/CA)* was invented for this network.