BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2104
BASICS OF DATA COMMUNICATION

Unit 11

Data Link Control Protocols

Table of Contents

1. INTRODUCTION

This unit introduces the concept of data link control protocols. Two main functions of the data link layer are data link control and media access control. Data link control deals with the design and procedures for communication between two adjacent nodes. Data link control protocols provide reliable and smooth transmission of frames between nodes. Protocols are 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.

In this unit, we are going to discuss five data link layer protocols, two for noiseless channels and three for noisy channels. We will discuss the simplest protocol and stop-and-wait protocol for noiseless channels. Then we will discuss protocols for noisy channels such as stop-and-wait automatic repeat request, go-back-N-automatic repeat request and selective repeat automatic repeat request protocols. In this unit, we will also discuss how a bit-oriented protocol is implemented by using high-level data link control (HDLC) protocol. In the last section, you will see one such popular byte-oriented protocol known as point-to-point protocol (PPP).

1.1 Objectives

After studying this unit, you should be able to:

- *Describe protocols for noiseless channels*
- *Explain protocols for noisy channels*
- *Describe high-level data link control protocol (hdlc)*
- *Explain point-to-point protocol (PPP)*

2. PROTOCOLS FOR NOISELESS CHANNELS

Let us assume that we have an ideal channel in which no frames are lost, duplicated or corrupted. In this section, we discuss two protocols for this noiseless channel. The first protocol known as simplest protocol which does not use flow control and the second is stop and wait protocol which uses flow control. Neither has error control because we assumed that the channel is a perfect noiseless channel.

Assume that a machine 'A' wants to send a long stream of data to machine B using a reliable, connection oriented service. B also wants to send data to A simultaneously. A data link layer frame comprisean embedded packet, some control information (in the header) and a checksum (in the trailer). The frame is then transmitted to the data link layer on the other machine. We also assume that there exist suitable procedure to send (*to_physical_layer*) and receive (*from_physical_layer*) frame.

Initially the receiver just waits for the procedure call, *wait_for_event*(&event). This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable event tells what happened. The set of possible event differs based on protocols. When a frame arrives at the receiver, checksum will be recomputed at the receiver side. If the checksum is incorrect, the data link layer is so informed (*so event=chksum_err*). If the inbound frame arrived undamaged, the data link layer is also informed (*event = frame arrival*) so that it can acquire the frame for inspection using from physical layer procedure. When the receiver side data link layer acquires an undamaged frame, it checks the control information in the header, and if everything is correct then it passes the packet portion to the network layer.

Five data structures are defined there. They are: *Boolean, seq nr, packet, frame kind,* and *frame.* A Boolean is an enumerated data type and can take on the values such as true and false. A seq nr is a small integer used to number the frames. These sequence numbers run from 0 up to *MAX SEQ*, which is defined in each protocol needing it. A packet is the unit of information exchanged between the network layer and the data link layer.

A frame is composed of four fields. They are: kind, seq, ack, and info. The first three fields contain control information and the last may contain actual data to be transferred. These control fields are collectively called the frame header. The kind field tells whether there are any data in the frame, because some of the protocols distinguish frames containing only control information from those containing data as well. The seq and ack fields are used for sequence numbers and acknowledgements, respectively. The info field of a data frame contains a single packet.

The procedure wait_for_event wait for something to happen. The procedures *to_network_* layer and *from_network_layer* are used by the data link layer to pass packets to the network layer and accept packets from the network layer. *From_physical_layer* and *to_physical_layer* pass frames between the data link layer and the physical layer. The procedures *start_timer* and *stop_timer* turn the timer on and off, respectively. The procedures *start_ack_timer* and *stop_ack_timer* control an auxiliary timer used to generate acknowledgements under certain conditions.

2.1 Simplest Protocol

Simplest protocol is the one with no flow or error control. It is a unidirectional protocol in which data frames are travelling in one direction from sender to receiver. As the name specifies, this protocol is as simple as it can be because it does not worry about the possibility of anything going wrong. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This protocol consist of two procedures, sender () and receiver (). The sender runs in the data link layer of the source machine, and the receiver runs in the data link layer of the destination machine. No sequence numbers or acknowledgements are used.

Here, sender is in an infinite while loop just pumping data out onto the line as fast as it can. The body of the loop consists of three actions, they are: fetch a packet from network layer, construct an outbound frame using the variable s, and send the frame. The receiver is equally simple. Initially, it waits for something to happen, the only possibility being the arrival of an undamaged frame. Once the frame arrives, the procedure wait_for_event returns, with event set to frame_arrival. The call to from_physical_layer removes the newly arrived frame from the hardware buffer and puts it in the variable r, where the receiver code can get at it. Finally, the data portion is passed on to the network layer, and the data link layer settles back to wait for the next frame. Algorithm shown in table 11.1 shows the procedure at the sender and receiver side.

Table 11.1: Simplex protocol

```
Void sender1()
{
Frame s;                        /*buffer for an outbound frame*/
Packet buffer;                  /*buffer for an outbound packet*/
While(true)
{
From_network_lyer(&buffer);     /* get something to send*/
s.info=buffer;                  /* copy it to s for transmission */
to_physical_layer(&s);          /* send it on its way */
}
}
Void receiver1()
{
Frame r;
Event_type event;
While(true)
{
Wait_for_event(&event);         /* only possibility is frame_arrival*/
From_physical_layer(&r);        /* get the inbound frame*/
To_network_layer(&r.info);      /* pass the data to the network layer */
}
}
```

Simplex protocol is an unrealistic protocol because it doesn't handle flow control or error correction.

2.2 Stop-and-Wait Protocol

Another type of data link protocol is stop and wait protocol. This protocol implements flow control mechanisms and prevent the sender from overloading the receiver. Flow control is the mechanism to prevent the sender to send frames faster so that receiver is unable to process them. This protocol assumes that communication channel is error free. Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called stop-and-wait. In this protocol, receiver sends feedback to the sender. After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to send the next frame. After sending a frame, sender will wait for the little dummy frame to arrive. Here, actual data traffic is from sender to receiver only, but frames travels in both directions. In this protocol, first sender sends a frame, then the receiver sends a frame (dummy frame for acknowledgment), then the sender sends another frame and so on. Algorithm shown in table 11.2 shows the procedure for stop and wait protocol for an error-free channel.

Table 11.2: stop and wait protocol for an error-free channel

```
Typedef enum (frame_arrival} event_type;
#include “protocol.h”
Void sender2(void)
{
  frame s;                    /* buffer for an outbound frame*/
  packet buffer;              /* buffer for an outbound packet*/
event_type event;             /* frame_arrival is the only possibility */
while(true)
{
  from_network_layer(&buffer);       /* get something to send*/
 s.info = buffer;                    /* copy it into s for transmission */
 to_physical_layer(&s);              /*sending frame*/
wait_for_event(&event);              /* waiting for acknowledgement*/
}
}
Void receiver2(void)
{
 frame r, s;                         /* buffers for frames*/
event_type event;                    /* frame_arrival is the only possibility */
while(true)

{
   wait_for_event(&event);           /* only possibility is frame_arrival*/
   from_physical_layer(&r);          /* get the inbound frame*/
  to_network_layer(&r.info);         /* pass the data to the network layer*/
to_physical_layer(&s);               /*send acknowledgement to sender */
}
}
```

Similar to simplex protocol, the sender starts out by fetching a packet from the network layer, create a frame using it and sending it to the destination. In this protocol, the sender must wait until an acknowledgement frame arrives before looping back and fetching the next packet from the network layer. In the receiver side, after delivering packet to the network layer, receiver sends an acknowledgement frame back to the sender before entering the wait loop again.

Self-Assessment Questions – 1

1. Which layer provides functions such as data link control and mediaaccess control?
   a) Physical layer
   b) Datalink layer
   c) Network layer
   d) Transport layer
2. Protocol for noiseless channel which uses flow control but nomechanism for error control are known as__________________________.
3. ______________ protocol does not use any flow control and errorcontrol.
4. A_____________is composed of four fields such as: *kind*, *seq*, *ack*, and *info*.

3. PROTOCOLS FOR NOISY CHANNELS

Although the stop-and-wait protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are non-existent. We can ignore the error or we need to add error control to our protocols.

3.1 Stop-and-Wait Automatic Repeat Request

Our first protocol, called the stop and wait automatic repeat request adds a simple error control mechanism to the stop and wait protocol. This protocol deals with the normal situation of a communication channel that make errors. Frames can be either damaged or lost completely in the communication channel. If the frame is damaged in transit, the receiver hardware will detect this while computing the checksum. When compared to the protocol for error free channel, here we are using an additional timer and the sender could send a frame, but the receiver would only send an acknowledgement frame if the data were correctly received. If a damaged frame arrived at the receiver, it would be discarded. After a while the sender would time out and send the frame again. This process would be repeated until the frame finally arrived is intact.

Another trouble in this scenario is that, consider the situation where a sender A sends a packet 1 to the receiver B, it is correctly received at B and B sends an acknowledgement frame back to A. But the acknowledgment frame gets lost completely and the timer on A eventually times out. Since sender A has not received the acknowledgment, it assumes that the data frame was lost or damaged and sends the frame containing packet 1 again. This duplicate frame also arrives at B and is passed to the network layer there. Here, duplication of the frame occurs.

So, some way is required to identify the duplicate frame. The obvious way to achieve this is to have the sender put a sequence number in the header of each frame it sends. Then the receiver can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded. A one bit sequence number (0 or 1) is sufficient. At each instant of time, the receiver expects a particular sequence number next. When a frame containing the correct sequence number arrives, it is accepted and passed to the network layer, then acknowledged. Then the expected sequence number is incremented modulo 2 (i.e., 0 becomes 1 and 1 becomes 0). Any arriving frame containing the wrong sequence number is rejected as a duplicate. Algorithm 11.3 shows the procedure for stop and wait protocol for a noisy channel.

Table 11.3: Stop and wait protocol for a noisy channel

```
#define MAX_SEQ 1
Typedef enum { frame_arrival, cksum_err, timeout}
event_type; #include "protocol.h"
Void sender3()
{
seq_nr next_frame_to_send;          /*seq number of next outgoing
frame */ frame s;                   /* scratch variable */
packet buffer;                      /* buffer for an outbound
packet */ event_type event;
next_frame_to_send =0;              /* Initialize outbound sequence
 numbers */ from_network_layer(&buffer);              /* fetch first
 packet */
while(true)
{
s.info=buffer;                          /* construct a frame for
transmission */ s.seq=next_frame_to_send;       /* insert sequence
number in frame */ to_physical_layer(&s);       /* send it on its way*/
start_timer(s.seq);                     /* if answer takes too long, time
out */ wait_for_event(&event);         /*   frame_arrival   ,   cksum_err,
timeout */ if(event==frame_arrival)
{
from_physical_layer(&s);               /* get the
acknowledgement*/ if(s.ack==next_frame_to_send)
{
stop_timers(s.ack);                    /* turn the timer off*/
from_network_layer(&buffer);           /* get the next one to
send*/ inc(next_frame_to_send);        /* invert
next_frame_to_send*/
}
}
}
}


Void receiver3()
{
seq_nr
frame_expected;
frame r,s;
event_type event;
frame_expected=0;
while(true)
{
 wait_for_event(&event);          /*possibilities: frame_arrival, cksum_err*/
 if(event==frame_arrival)         /* a valid frame has arrived */
 {
 from_physical_layer(&r);         /* get the newly arrived frame*/
 if(r.seq==frame_expected)
 {
 to_network_layer(&r.info);       /* pass the data to network layer*/
 inc(frame_expected);             /* next time expect the other sequence nr */
 }
 s.ack= 1- frame_expected;        /* tell which frame is being acknowledge*/
 to_physical_layer(&s);           /* send acknowledgement */
 }
 }}
```

Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called *ARQ (Automatic Repeat reQuest) or PAR (Positive Acknowledgement with Retransmission*). The sender remembers the sequence number of the next frame to send in *next_frame_to_send*; the receiver remembers the sequence number of the next frame expected in *frame_expected*. Each protocol has a short initialization phase before entering the infinite loop.

After transmitting a frame, the sender starts the timer. If timer was already running, it will be reset to allow another full timer interval. The interval should be chosen in such a way that it should allow enough time for the frame to get to the receiver, process at the receiver, and for the acknowledgement frame to propagate back to the sender. Only when that interval has elapsed is it safe to assume that either the transmitted frame or its acknowledgement has been lost, and to send a duplicate.

The sender waits for some time for an event after transmitting the frame and start the timer. Any of the three possible events can happen: an acknowledgment frame arrives correctly, a damaged acknowledgement frame arrives, or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet. It advances the sequence number.

Instead, if a damaged frame arrives, or the timer expires, sender will send a duplicate frame without changing the buffer or the sequence number.

When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated. Duplicate and damaged frames are not accepted at the receiver, but it results the last correctly received frame to be acknowledged to inform the sender to advance to the next frame or retransmit a damaged frame.

3.2 Go-Back-N-Automatic Repeat Request

The problem we have faced in the previous algorithm is that the sender has to wait for an acknowledgment before sending another frame. If we avoid that constraint, much better efficiency can be achieved. The solution lies in allowing the sender to transmit up to w frames before blocking, instead of 1. With a large enough choice of w the sender will be able to continuously transmit frames since the acknowledgements will arrive for previous frames before the window becomes full, preventing the sender from blocking.

The value of w is determined by the number of frames that can fit inside the channel as they propagate from sender to receiver. This capacity is determined by the bandwidth in bits/sec multiplied by the one-way transit time, or the bandwidth-delay product of the link. The technique of keeping multiple frames in trajectory is an example of pipelining. Pipelining frames over an unreliable communication channel raises some serious issues. If a frame in the middle of a long stream is damaged or lost, succeeding frames will arrive at the receiver before the sender even finds out that anything is wrong. Two basic approaches are available for dealing with errors in the presence of pipelining, which are shown in below figure 11.1.

One approach is called ***go-back-n.*** In this, the receiver discard subsequent frames, sending no acknowledgements for the discarded frames. We can see that this approach corresponds to receive window of size 1. That is, here the data link layer refuses to accept any frame except the next one it must give to the network layer. If the sender's window fills up before the timer runs out, the pipeline will be empty. The sender will time out and retransmit all unacknowledged frames in order, starting with the damaged or lost one. This approach can waste bandwidth if the error rate is high.

In figure 11.1 (b), we see ***go-back-n*** for the case in which the receiver's window is large. Frames 0 and 1 are correctly received and acknowledged. Frame 2, however, is damaged or lost. The sender, unaware of this problem, continues to send frames until the timer for frame 2 expires. Then it backs up to frame 2 and starts over with it, sending 2, 3, 4, etc. all over again.

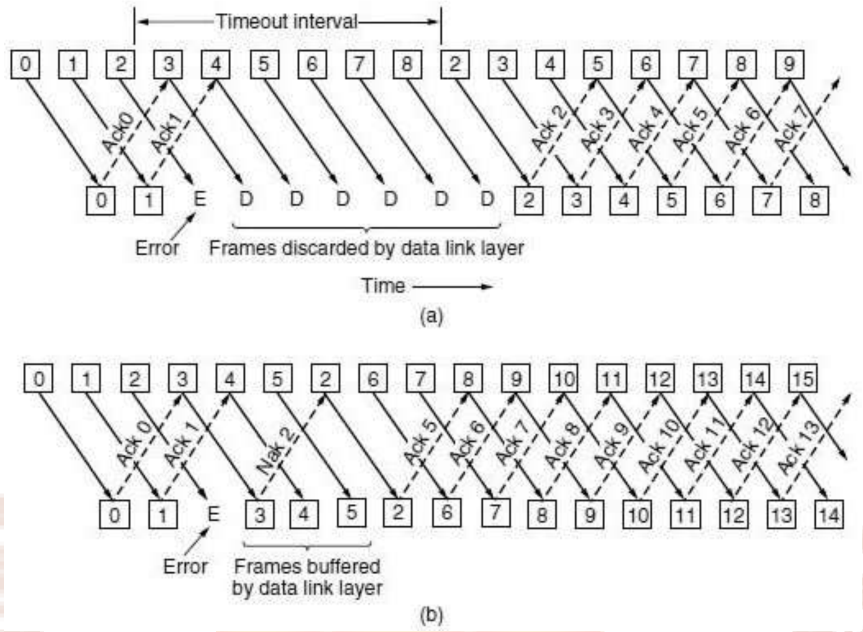
Timeout interval
Ack0
Ack1
E
D
Error
Frames discarded by data link layer
Time
(a)
Ack 0
Ack 1
Nak 2
Error
Frames buffered by data link layer
(b)

Fig 11.1: Pipelining and error recovery. Effect of an error when (a) receiver's window size is 1 and (b) receiver's window size is large.

Second approach for handling errors when frames are pipelined is called selective repeat. When this is used, a bad frame that is received is discarded, but any good frames received after it are accepted and buffered. When the sender times out, only the oldest unacknowledged frame is retransmitted. If that frame arrives correctly, the receiver can deliver to the network layer, in sequence, all the frames it has buffered. Selective repeat corresponds to a receiver window larger than 1. This approach can require large amounts of data link layer memory if the window is large.

Mostly, selective repeat is combined with having the receiver send a negative acknowledgement (NAK) when it detects an error. NAKs stimulate retransmission before the corresponding timer expires and thus improve performance. In figure 11.1 (b), frames 0 and 1 are correctly received and acknowledged and frame 2 is lost. When frame 3 arrives at the receiver, the data link layer there notices it has missed a frame, so it sends back a NAK for 2 but buffers 3. When frames 4 and 5 arrive, they, too, are buffered by the data link layer instead of being passed to the network layer. Eventually, the NAK 2 gets back to the sender, which immediately resend frame 2. When that arrives, the data link layer now has 2, 3, 4, and 5 and can pass all of them to the network layer in the correct order. It can also acknowledge all frames up to and including 5.

In a ***go-back-n*** protocol, the receiving data link layer only accepts frames in order. Frames following an error are discarded. Although ***go-back-n*** protocol does not buffer the frames arriving after an error, it does not escape the problem of buffering altogether. Since a sender may have to retransmit all the unacknowledged frames at a future time, it must keep all transmitted frames until it knows that they have been accepted by the receiver. When an acknowledgement comes in for frame n, frames n − 1, n−2, and so on are also automatically acknowledged. This acknowledgement is called a ***cumulative acknowledgement.*** This property is very important because when some of the previous acknowledgement- bearing frames were lost or garbled. Whenever any acknowledgement comes in, the data link layer checks to see if any buffers can now be released. If buffers can be released, a previously blocked network layer can be available to cause more network layer ready events.

3.3 Selective Repeat Automatic Repeat Request

In case of go-back-n protocol if there are more errors then, it wastes a lot of bandwidths on retransmitted frames. This protocol is best suited if errors are rare. An alternative strategy is selective repeat protocol. Here, the receiver can accept and buffer the frames following a damaged or lost one. In this protocol, both sender and receiver maintain a window of outstanding and acceptable sequence numbers respectively.

Sender's window size starts at 0 and grows to some pre-defined maximum. But receiver's window is always fixed in size and equal to the pre-defined maximum. The receiver has a buffer reserved for each sequence number within its fixed window. Associated with each buffer is a bit telling whether the buffer is full or empty. Whenever a frame arrives, its sequence number

is checked to see if it falls within the window. If so and if it has not already been received, accepted and stored. This action is taken regardless of whether the frame contains the next packet expected by the network layer. It must be kept within the data link layer and not passed to the network layer until all the lower-numbered frames have already been delivered to the network layer in the correct order.

A problem associated with this nonsequential receive is illustrated here with an example. Suppose that we have a 3-bit sequence number, so that the sender may transmit up to seven frames before being required to wait for an acknowledgement. Initially, the sender's and receiver's windows are as shown in figure 11.2 (a). The sender now transmits frames 0 through 6. The receiver's window allows it to accept any frame with a sequence number between 0 and 6 inclusive. All seven frames arrive correctly, so the receiver acknowledges them and advances its window to allow receipt of 7, 0, 1, 2, 3, 4, or 5, as shown in figure 11.2 (b). All seven buffers are marked empty. Suppose at this point all the acknowledgements lost, the protocol should operate correctly regardless of this trouble. The sender eventually times out and retransmits frame 0. When this frame arrives at the receiver, a check is made to see if it falls within the receiver's window. Unfortunately, in figure

11.2 (b) frame 0 is within the new window, so accepted as a new frame. The receiver also sends a (piggybacked) acknowledgement for frame 6 since 0 through 6 have been received.

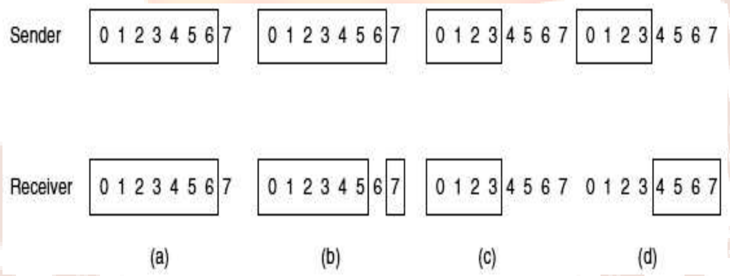

Fig 11.2: (a) Initial situation with a window of size 7 (b) After 7 frames have been sent and received but not acknowledged. (c) Initial situation with a window size of 4. (d) After 4 frames have been sent and received but not acknowledged.

The sender is happy to learn that all its transmitted frames arrived correctly, so it advances its window and immediately sends frames 7, 0, 1, 2, 3, 4, and 1. Frame 7 will be accepted by the receiver and its packet will be passed directly to the network layer. Thereafter, the receiving data link layer checks to see if it has a valid frame 0 already, discovers that it does, and passes the old buffered packet to the network layer as if it were a new packet. The network layer gets an incorrect packet, and the protocol fails.

We can say, the essence of this problem is that after the receiver advanced its window, the new range of valid sequence numbers overlapped the old one. Therefore, the following batch of frames might be duplicates or new one. The receiver has no way to distinguish these two cases.

This problem can be resolved by making sure that after the receiver has advanced its window there is no overlap with the original window. So, in order to ensure this, the maximum window size should be at most half the range of the sequence numbers. This is shown in figure 11.2 (c) and Fig

11.2 (d). With 3 bits, the sequence numbers range from 0 to 7. Only four unacknowledged frames should be outstanding at any instant. That way, if the receiver has just accepted frames 0 through 3 and advanced its window to permit acceptance of frames 4 through 7, it can unambiguously tell if subsequent frames are retransmissions (0 through 3) or new ones (4 through 7).

Self-Assessment Questions – 2

5. The protocol known as_____________________adds a simple error control mechanism to the stop and wait protocol.
6. Protocols in which the sender waits for a positive acknowledgementbefore advancing to the next data item are often called ______________.
7. In_approach, the receiver discard subsequentframes, sending no acknowledgements for the discarded frames.
8. In__________protocol, the receiver can accept and bufferthe frames following a damaged or lost one.

4. HIGH-LEVEL DATA LINK CONTROL PROTOCOL (HDLC)

A bit-oriented protocol views the frame as a collection of bits. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. HDLC provides two common transfer modes that can be used in different configurations. They are normal response mode (NRM) and asynchronous balanced mode (ABM).

Normal Response Mode

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands. A secondary station can only respond. The NRM is used for both point to point and multiple-point links. Figure 11.3 shows normal respond mode

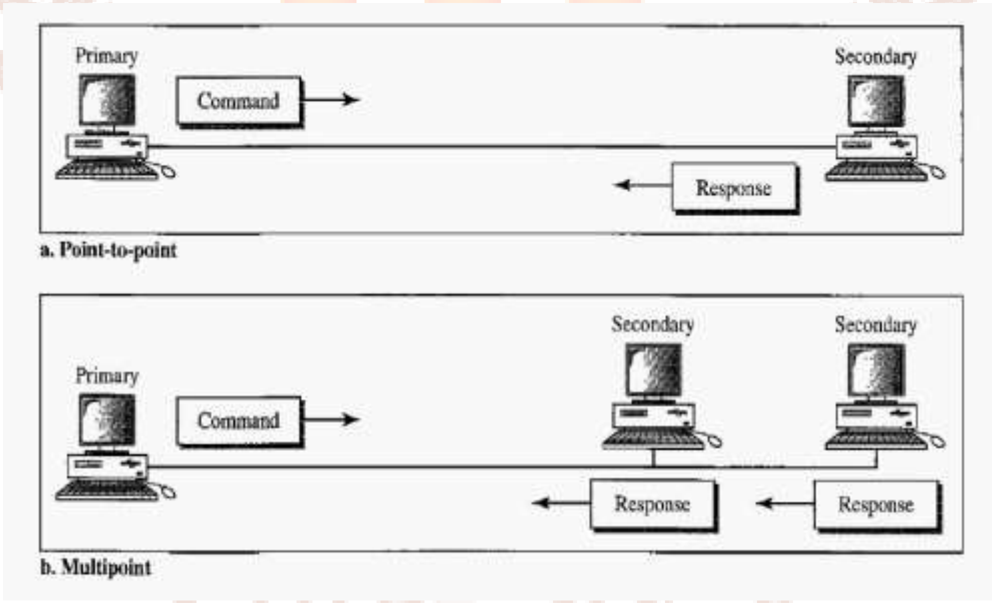

Fig 11.3: Normal Response Mode

Asynchronous Balanced Mode

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point and each station can function as a primary and secondary. Figure 11.4 shows asynchronous balanced mode.

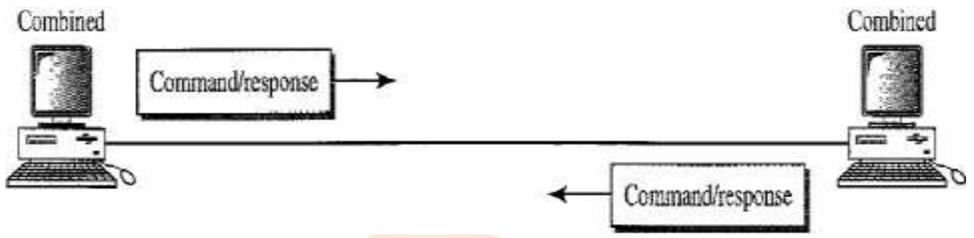

Fig 11.4: Asynchronous balanced mode

Frames

To provide flexibility necessary to support all options possible in the modes, HDLC defines three types of frames. Information frames (I-frames), supervisory frames (S-frames) and unnumbered frames (U-frames). 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 (piggybacking). 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 contain up to six fields as shown in figure 11.5. The fields are: beginning flag field, address filed, control field, information field, 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.

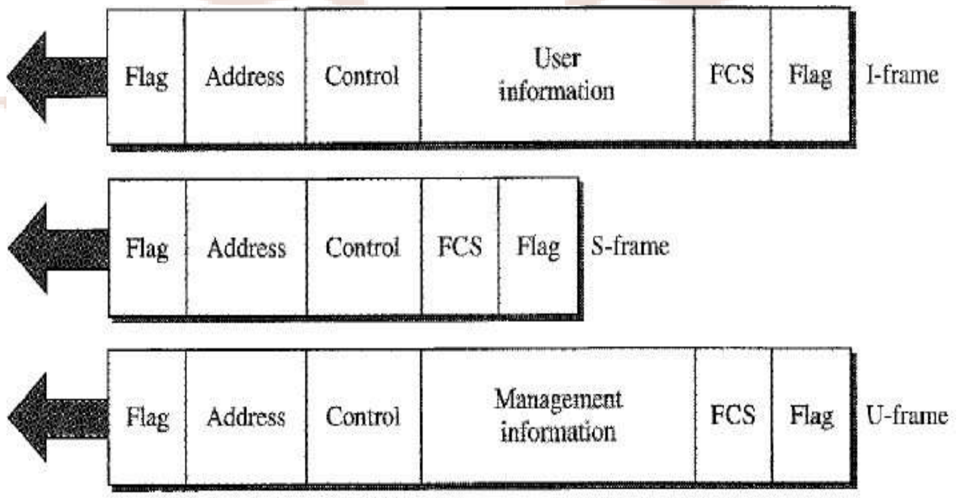

Fig 11.5: HDLC frames

The flag field of an HDLC frame is an 8-bit sequence with bit pattern 01111110 that identifies both beginning and end of a frame and serves as a synchronization pattern for the receiver.

The second filed 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 a frame, it contains a from address. An address field can be 1 byte or several bytes long, depending on the needs of the network. One byte can identify up to 128 stations. Larger networks require multiple byte address fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0, only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come. 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. The information field contains the user's data from the network layer or management information. Its length can vary from one network to another. The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2 or 4-byte CRC (cyclic redundancy check).

Self-Assessment Questions – 3

9. The Synchronous Data Link Control (SDLC) protocol developed by IBMwas later standardized by the ISO as_____protocol.
10. Two common transfer modes that can be used in different configurations are____ ___________and_________________________________.
11. HDLC defines three types of frames, they are_________, _____________ and______.

5. POINT-TO-POINT PROTOCOL (PPP)

One of the most common protocols for point-to-point access is point to point protocol (PPP). Today many users who want to connect their home computers to the server of an internet service provider use PPP. The majority of users have a traditional modem, they are connected to the internet through a telephone line which provides the service of the physical layer. To control and manage the transfer of data, there is a need for a point to point protocol at the data link layer. Services provided by point to point protocol are:

- Point to point protocol defines the format of the frame to be exchanged between devices.
- Point to point protocol defines how two devices can negotiate the establishment of the link and the exchange of data.
- Point to point protocol defines how network layer data are encapsulated in the data link frame.
- Point to point protocol defines how to 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 useful when a home user needs a temporary network address to connect to the internet.

Point to point protocol does not provide flow control. It has a very simple mechanism for error control. PPP does not provide an advanced addressing mechanism to handle frames in a multipoint configuration.

Framing

PPP is a byte oriented protocol. Figure 11.6 shows the format of a PPP frame.

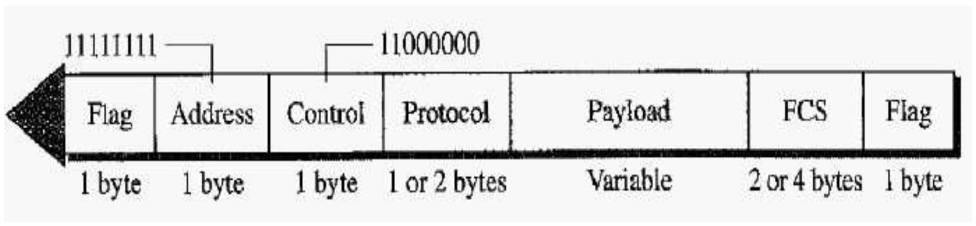

Fig 11.6: Point to Point Protocol (PPP) frame format

A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110. PPP is a byte oriented protocol, HDLC is a bit oriented protocol. The flag is treated as a byte. The address field in this protocol is a constant value and set to 11111111. Control field is set to the constant value 11000000. PPP does not provide any flow control. Error control is also limited to error detection. This means that this field is not needed at all and two parties can agree during negotiation to omit this byte. The protocol field defines what is being carried in the data field, either user data or other information. Payload field carries either the user data or other information. The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC (cyclic redundancy check).

Byte stuffing

The similarity between PPP and HDLC ends at the frame format. As a byte oriented protocol, the flag in PPP is a byte and needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that every time the flag like pattern appears in the data, this extra byte is stuffed to tell the receiver that next byte is not a flag.

Transition Phases

A PPP connection goes through phases which can be shown in a transition phase diagram which shows in figure 11.7.

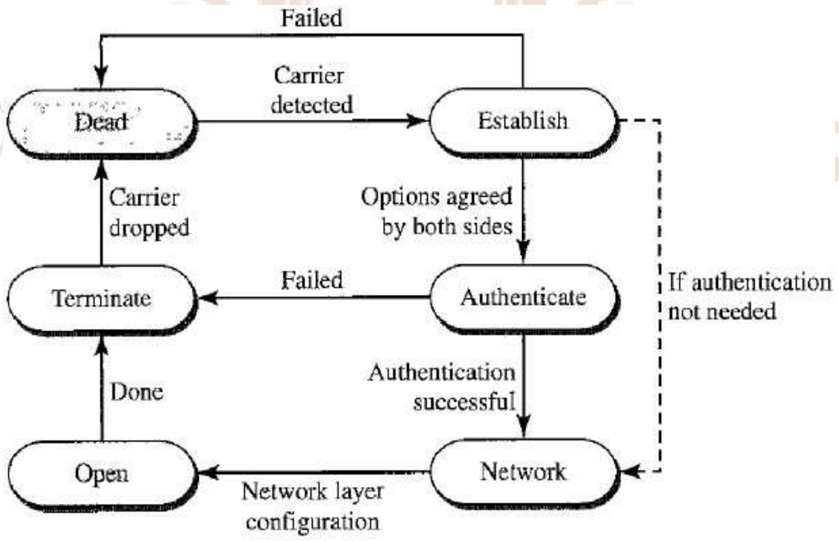
Failed
Carrier detected
Dead
Establish
Carrier dropped
Options agreed by both sides
Terminate
Failed
Authenticate
If authentication not needed
Done
Authentication successful
Open
Network
Network layer configuration

Fig 11.7: Transition phases

In the dead phase the link is not being used. There is no active carrier and the line is quiet. When one of the nodes starts the communication, the connection goes into establish phase.

In this phase, options are negotiated between the two parties. If the negotiation is successful, the system goes to the authentication phase or directly to the networking phase. The authentication phase is optional. The two nodes may decide during the establishment phase, not to skip this phase. However, if they decide to proceed with authentication, they send several authentication packets. If the result is successful, the connection goes to the networking phase. Otherwise it goes to the termination phase. In network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at the network layer can be exchanged. The reason is that PPP supports multiple protocols at the network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive data. In the open phase, data transfer takes place. When a connection reaches this phase, the exchange of data packets can be started. The connection remains in this phase until one of the endpoints wants to terminate the connection. In the termination phase, the connection is terminated. Several packets are exchanged between the two ends for house cleaning and closing the link.

Self-Assessment Questions – 4

12. The most common protocols for point-to-point access is______________.
13. Point to point protocol is a_________________protocol.

6. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Protocols are 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.
- Simplest protocol and stop and wait protocol are the protocols for noiseless channel.
- Simplest protocol is the one with no flow or error control. It is a unidirectional protocol in which data frames are travelling in one direction from sender to receiver.
- Stop and wait protocol implements flow control mechanisms and prevent the sender from overloading the receiver.
- Stop and wait automatic repeat request, Go-back-N-automatic repeat request and selective repeat automatic repeat request are the protocols for noisy channels.
- The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol.
- HDLC provides two common transfer modes that can be used in different configurations. They are normal response mode (NRM) and asynchronous balanced mode (ABM).
- One of the most common protocols for point-to-point access is point to point protocol (PPP).
- Point to point protocol is a byte oriented protocol.

7. TERMINAL QUESTIONS

1. Explain the protocols for noiseless channels.
2. Describe Stop-and-wait automatic repeat request.
3. Differentiate between go-back-N-automatic repeat request and selective repeat automatic repeat request.
4. Explain high level data link control protocol (HDLC)
5. Describe point-to-point protocol.

8. ANSWERS

Self-Assessment Questions

1. (b) datalink layer
2. Stop and wait protocol
3. Simplest
4. Frame
5. Stop and wait automatic repeat request
6. Automatic repeat request (ARQ)
7. Go-back-n
8. Selective repeat protocol
9. HDLC (High-level data link protocol)
10. Normal response mode (NRM), asynchronous balanced mode (ABM)
11. Information frame (I-frame), supervisory frame (S-frame), unnumbered frame (u-frames)
12. Point to point protocol
13. Byte oriented protocol

Terminal Questions

1. The first protocol known as simplest protocol which does not use flow control and the second is stop and wait protocol which uses flow control. Neither has error control because we assumed that the channel is a perfect noiseless channel. (Refer section 2 for detail).
2. Our first protocol, called the stop and wait automatic repeat request adds a simple error control mechanism to the stop and wait protocol. This protocol deals with the normal situation of a communication channel that make errors. Frames can be either damaged or lost completely in the communication channel. (Refer section 3.1 for detail).
3. In go-back-n, the receiver discard subsequent frames, sending no acknowledgements for the discarded frames. We can see that this approach corresponds to receive window of size 1. In case of go-back-n protocol if there are more errors then, it wastes a lot of bandwidths on retransmitted frames. This protocol is best suited if errors are rare. An alternative strategy is selective repeat protocol. Here, the receiver can accept and buffer the frames following a damaged or lost one. (Refer section 3.2 and 3.3 for detail).

4. A bit-oriented protocol views the frame as a collection of bits. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. (Refer section 4 for detail).
5. One of the most common protocols for point-to-point access is point to point protocol (PPP). Today many users who want to connect their home computers to the server of an internet service provider use PPP. (Refer section 5 for detail)

References:

1. Behrouz A. Forouzan, Sophia Chung Fegan, “Data Communications and Networking”, Fourth edition.
2. William Stallings, “Data and Computer Communications”, Sixth edition, Pearson Education, Delhi, 2002.
3. Taub and Schilling, “Principles of Communication Systems”, Tata Mc Graw Hill, Delhi, 2002.
4. S. Tanenbaum, ”Computer Networks”, Pearson Education, Fourth Edition.
5. N. Olifer, V. Olifer, ”Computer Networks: Principles, technologies and Protocols for Network design”, Wiley India Edition, First edition.
6. Simon Poulton (2003), packet switching and X.25 Networking, Pitman publishing.
7. Walrand, P. Varaiya, “high performance communication networks”, Morgan kaufmann.