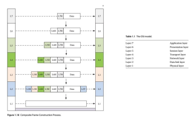
3

Data Link Layer Protocols

3.1 Introduction

As discussed in Chapter 1, the data link layer is responsible for organizing data in frames and for detecting errors that occur in a frame. It is also responsible for hop-by-hop (or link-by-link) flow control. In this chapter, we examine the structure of a frame, the data link layer flow control schemes, and two of the protocols defined for this layer, namely the high-level data link control (HDLC) and the point-to-point protocol (PPP).



3.2 Framing

The data link layer breaks up the bitstream it receives from the network layer into discrete *frames* and computes the checksum for each frame. In the past, many schemes were used to create frames, especially when character-oriented transmission was still used. Today, the common method is the bit-oriented scheme in which a frame consists of the following:

- A leading *flag*
- A header
- The data to be transmitted (or payload)
- The checksum
- A trailing flag.

The format of a frame is shown in Figure 3.1. The exact structure of the header depends on the type of data link control (DLC) protocol in use. The checksum is used for error detection in the frame at the receiver.

Fundamentals of Data Communication Networks, First Edition. Oliver C. Ibe.
© 2018 John Wiley & Sons, Inc. Published 2018 by John Wiley & Sons, Inc.

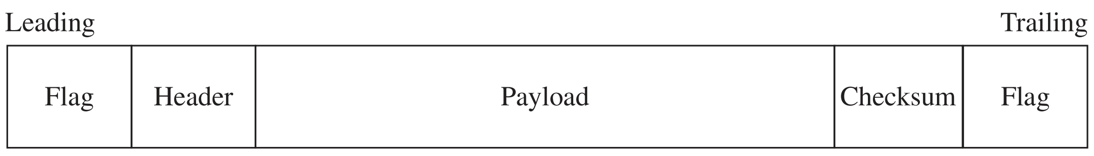


Figure 3.1 Format of Data Link Layer Frame.

3.3 Bit Stuffing

The flag is a special bit pattern 01111110, which is 1 byte (or octet) long. A process called *bit stuffing* is used to preserve the uniqueness of the flag in a frame. Thus, bit stuffing ensures that the flag's bit pattern never appears anywhere else in a frame. This is carried out in the following way. When the transmitter encounters five consecutive 1s following a 0 in the data, it automatically stuffs (i.e., inserts) a 0 into the outgoing bit stream. At the receiving end, when the receiver sees five consecutive 1s followed by a 0 bit, it automatically destuffs (or deletes) the 0 bit.

For example, consider the bit stream 011011101111100110 that is to be transmitted. We have the following:

(a) Data to be transmitted: 011011101111100110
(b) Data actually transmitted: 0110111011111**0**00110
(c) Data received after destuffing: 011011101111100110.

The 0 in bold is the stuffed bit that is inserted because there are five 1s after a 0 before the stuffed bit. At the receiving end, the receiver strips the zero and recovers the original bitstream.

As another example, assume that we want to send the bitstream:

$$m = 0110111011111111111100110$$

What is actually sent is m_1 = 0110111011111**0**11111**0**100110. In this case two bits are stuffed, as shown in bold. The first bit is stuffed after the five 1s following a 0. The second bit is stuffed as a result of the first stuffed bit. Introducing the first stuffed bit restarts the counter, which means that if there are five consecutive 1s after that 0, then another bit needs to be inserted. At the destination, the two bits in bold are discarded.

3.4 Flow Control

Flow control is a technique used to ensure that a sender transmits data at a rate that the receiver can accept. Thus, it is used to ensure that the transmitter does not overwhelm the receiver. Flow control can be exercised at the data link layer on a hop-by-hop basis and at the transport layer on an end-to-end basis. In this chapter, we consider flow control protocols that are used in the data link layer.

These include the *stop-and-wait* protocol and the *sliding window flow control* protocol.

3.4.1 The Stop-and-Wait Protocol

In the stop-and-wait protocol, the sender sends one frame and then waits for an acknowledgment (or ACK) from the receiver before sending another frame. Since the sender cannot send a new frame until the receiver has issued an ACK for the previous frame, it is obvious that the protocol ensures that the sender is transmitting at a rate that the receiver can handle. Figure 3.2 is an illustration of the protocol.

One of the drawbacks of the scheme is its poor channel utilization. A single frame travels from the source to the destination, and the associated acknowledgment travels from the destination to the source. This round trip delay, which is the time for a frame to go from the source to the destination and for the ACK to be received at the source, can be appreciable, particularly if the distance between the source and the destination is long.

3.4.2 The Sliding Window Flow Control

In the sliding window flow control, each outgoing frame contains a sequence number that ranges from zero to a predefined maximum number. The maximum number is usually $2^n - 1$, which means that the sequence numbers are represented by an n-bit field in the frame. The essence of the window scheme is that at any time instant, the sender maintains a set of sequence numbers corresponding to the frames that it is allowed to send. These frames are said to fall within the *sending window*. Similarly, the receiver maintains a *receiving window*, which corresponds to the set of frames it is permitted to receive.

The sequence numbers within the sending window represent the frames that have been sent or can be sent but are yet unacknowledged. The sequence numbers within the receiving window represent the frames that the receiver may accept; any frame falling outside the window is silently discarded. Note that the sending and receiving windows need not have the same lower and upper limits. Also, the window size need not be equal to the maximum number of frames that the sequence number field can support. Thus, even though the maximum number of frames that an n-bit sequence number can support is 2^n, the window size need not be 2^n; however, it cannot exceed 2^n. For example, consider a system

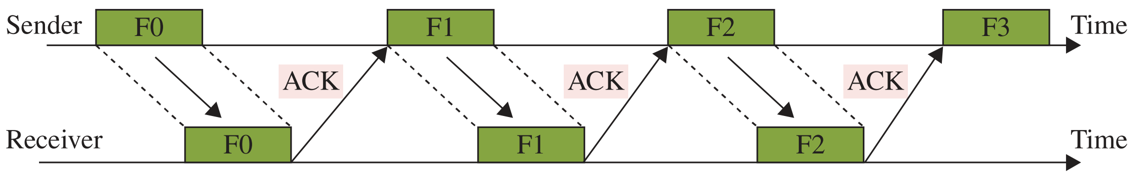


Figure 3.2 Example of Stop-and-Wait Protocol.

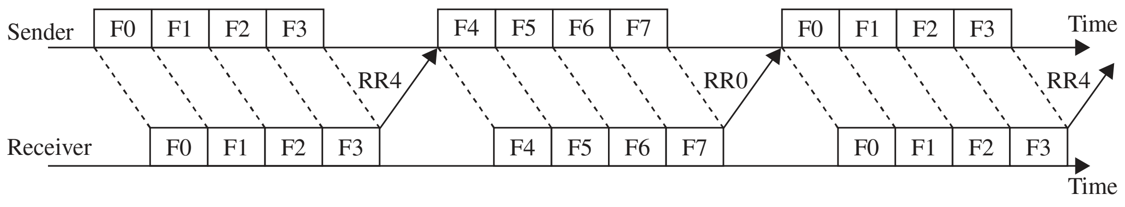


Figure 3.3 Example of the Window Flow Control.

with $n = 3$ and window size $W = 4$, which is shown in Figure 3.3 where the frames are numbered modulo-8 (since $2^3 = 8$). The sender usually does group acknowledgment with the RR (or receive ready) message, where RRk means "I have received frames up to and including frame $k - 1$; I am ready to receive frame k."

The sliding window flow control is a major improvement over the stop-and-wait protocol in terms of the channel utilization. The channel utilization increases as the window size increases.

3.5 Error Detection

Every practical communication channel is prone to noise that can cause errors to occur in the received information. Two techniques that are used to control transmission errors are error correction coding and error detection coding. Error correction codes add enough redundant bits, called *error correction bits*, to the information bit stream to enable the receiver to determine which bits have been corrupted by noise. Error detecting codes are concerned with detecting that errors have occurred without knowing precisely which bits are in error. Thus, while they also add redundant bits to the data bitstream that is to be transmitted, the overhead in terms of the number of redundant bits they use for this function is much smaller than that of error correcting codes.

In this chapter, we consider only error detection that usually calls for a retransmission of the information. We consider two error detection schemes:

- Parity checking
- Cyclic redundancy checking (CRC).

3.5.1 Parity Checking

A simple method of error detection is by adding redundant bits called *parity bits* to each character. This method is called parity checking and is commonly used for ASCII characters where seven bits are used for actual character encoding and the eighth bit is for parity. The value of the parity bit (i.e., the eighth bit) is chosen to make the number of 1s an even number (for even parity) or an odd number (for odd parity).

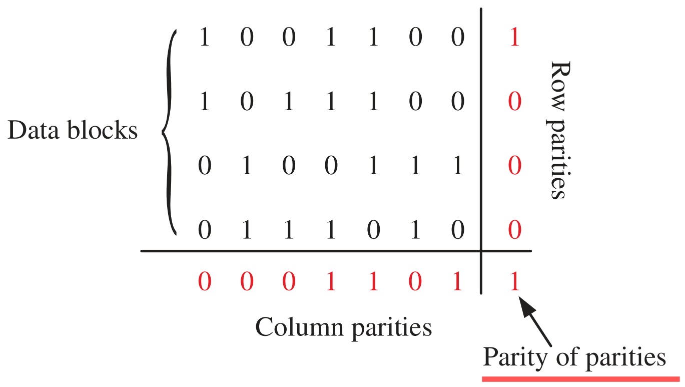
As an example of even parity, assume that the bit sequence 1001101 is to be transmitted; the message actually transmitted is 10011010 – the number of 1s is 4 (even), so the bit 0 is appended after the original sequence. The problem with the scheme is that if, for example in even parity, an even number of errors occur, the parity check scheme will fail to detect the fact the frame should be discarded. For example, if we transmitted 10011010 and received 10011000, the parity check will fail because the second to the last bit was corrupted, which is fine. But if the bit sequence 11011000 is received, the parity check will pass, even though the second bit and second to the last bit were corrupted in transit.

3.5.2 Two-Dimensional Parity

In the two-dimensional parity check, blocks of data are organized as a two-dimensional array. Specifically, each row of the array is a data block that is to be transmitted. A parity bit is appended to each row based on if even or odd parity is used. The parity bit for each column is similarly calculated. This is illustrated in Figure 3.4, which has four blocks of data and even parity is used in both row and column parities.

As shown in Figure 3.5(a), when exactly one error occurs the scheme can detect the location by the failure of the horizontal and parity checks associated with that bit. Then it can flip the bit thereby correcting the error. It can detect certain configurations of errors, particularly if they do not occur on the same

Figure 3.4 Example of Two-Dimensional Parity Check.



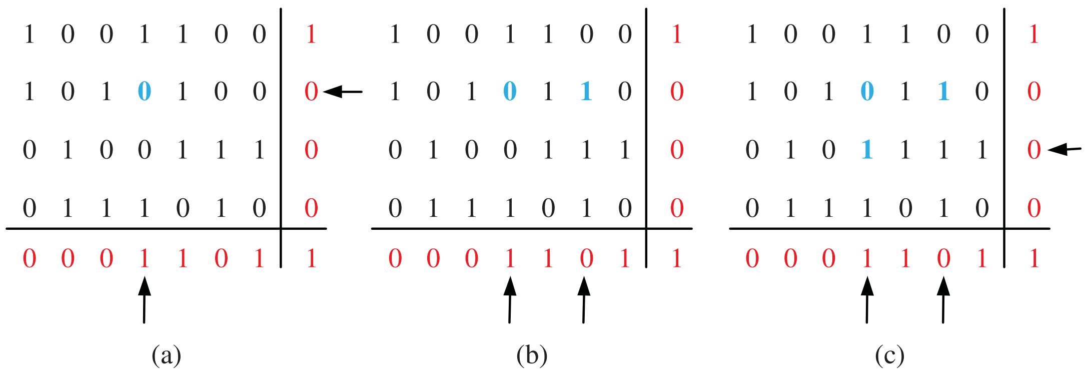


Figure 3.5 Some Possible Error Configurations. (a) 1 Error; (b) 2 Errors; and (c) 3 Errors.

row or column which it cannot detect. However, it cannot correct such errors as Figure 3.5(b) illustrates. Similarly, it can detect three errors, as illustrated in Figure 3.5(c). Finally, it cannot detect all the four error configurations.

3.5.3 Cyclic Redundancy Checking

A more sophisticated error detection scheme used by almost all communication networks is the CRC. CRC detects the presence of transmission errors by adding a few bits called CRC bits or *checksum* bits or *frame check sequence* (FCS) bits to each frame.

Let m be a string of $n+1$ bits, where n is usually a large number. m is the message to be transmitted and is represented by the binary string $m = b_n, b_{n-1}, \dots, b_2, b_1, b_0$, where $b_i = 0$ or 1. The message is sometimes represented as the polynomial $M(x)$ in which the bits of the message are considered to be the coefficients of the polynomial; that is, for the above message m we have the following polynomial of order n (which implies that the highest power of x is n and thus the number of bits in the bit stream m is $n+1$):

$$M(x) = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_2x^2 + b_1x + b_0$$

For example, the bit stream $m = 101101$ is represented as the polynomial $M(x) = x^5 + x^3 + x^2 + 1$, which is a polynomial of order 5.

Before it is transmitted, the message polynomial is known only to the transmitter. Another polynomial $G(x)$ called the *generator polynomial* of order k is known to both the transmitter and the receiver. Typical values of k are 12, 16, and 32.

The sender and receiver use an agreed-upon *generator polynomial* $G(x)$. The high and low order bits of $G(x)$ are 1. To compute a *checksum* for a frame with n bits (whose polynomial is $M(x)$), the frame must be longer than the generator polynomial.

CRC operates as follows:

(a) $M(x)$ is multiplied by x^k, which is equivalent to shifting the bit stream k bits to the left (recall that $k+1$ is the number of bits in g).
(b) The new polynomial $x^kM(x)$ is divided by $G(x)$ to generate a remainder $R(x)$ whose degree is at most k.
(c) The division is not done in the usual manner; in the subtraction portion of division, the remainder is obtained through an XOR operation rather than by traditional subtraction.
(d) The quotient is discarded, and the message is concatenated with the remainder, and the concatenated bit stream is transmitted.
(e) Upon receiving the bit stream, the receiver divides it by $G(x)$. If a nonzero remainder is found, the message is discarded because it is in error; otherwise, it is accepted.

The reason is the following: Let

$$\frac{x^k M(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

where $Q(x)$ is the quotient. Then, using modulo-2 arithmetic,

$$\frac{x^k M(x) + R(x)}{G(x)} = Q(x)$$

That is, $x^k M(x) + R(x)$ is divisible by $G(x)$, where in modulo-2 arithmetic, addition and subtraction operations produce identical results. Thus, if we discard $Q(x)$ and add the remainder bitstream r to the shifted version of the message (equivalently, if we append the remainder bitstream to the original message bitstream), the cascaded sequence obtained is divisible by g.

Note: In the remainder of this discussion, we use the upper case for polynomials and the lower case for bitstreams. For example, $G(x)$ is the generator polynomial and g is the associated bitstream.

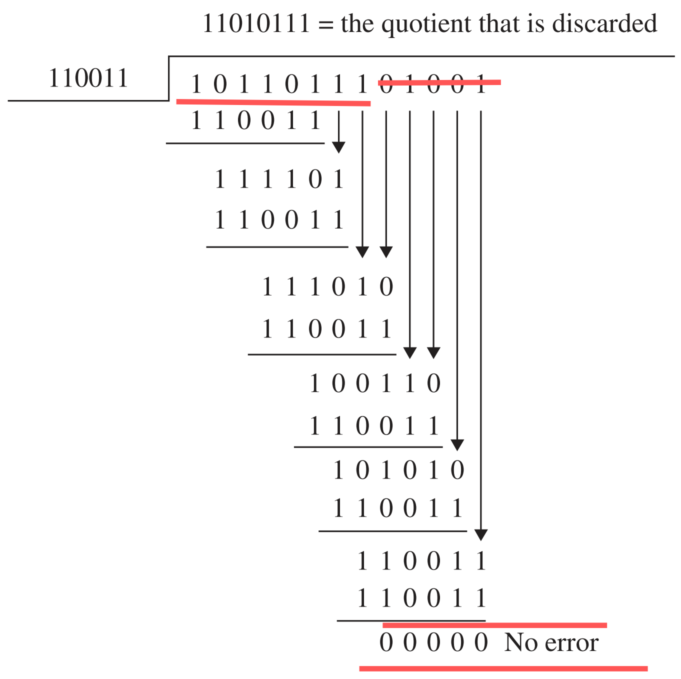
Example 3.1 Assume that $M(x) = x^7 + x^5 + x^4 + x^2 + x + 1$, which means that m = 10110111. Let $G(x) = x^5 + x^4 + x + 1$, which means that g = 110011 and $k = 5$. Let

$$A(x) = x^k M(x) = x^5 M(x) = x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5$$

This corresponds to the bitstream a = 1011011100000, which is the original bitstream m shifted five places to the left. Next we divide a by g to generate the remainder, which is the FCS:

```
                11010111 = the quotient that is discarded
         ___________________________________
 110011 | 1 0 1 1 0 1 1 1 0 0 0 0 0
          1 1 0 0 1 1
          -----------
            1 1 1 1 0 1
            1 1 0 0 1 1
            -----------
                1 1 1 0 1 0
                1 1 0 0 1 1
                -----------
                    1 0 0 1 0 0
                    1 1 0 0 1 1
                    -----------
                      1 0 1 1 1 0
                      1 1 0 0 1 1
                      -----------
                        1 1 1 0 1 0
                        1 1 0 0 1 1
                        -----------
                          0 1 0 0 1  = r (or FCS)
```

The FCS is appended to the message to generate the bitstream b, which is then transmitted, where $b = 1011011101001$. At the receiver, b is divided by g, and if the remainder of this division is zero, the last $k = 5$ bits of b are discarded to generate the message bitstream m:



Different generator polynomials are used in CRC computations for different applications. These include the following:

(a) CRC-8: $x^8 + x^2 + x + 1$
(b) CRC-10: $x^{10} + x^9 + x^5 + x^4 + x + 1$
(c) CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$
(d) CRC-16: $x^{16} + x^{15} + x^2 + 1$
(e) CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
(f) CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

3.6 Error Control Protocols

We now consider mechanisms that deal with errors that occur in the transmission of a frame. These mechanisms deal with *error correction by retransmission*; that is, when an error occurs, the receiver will detect it and inform the source, and the source will have to retransmit the frame either on account of an explicit negative acknowledgment (NAK) from the receiver or after a timeout period with no positive ACK. Three error control schemes based on the automatic repeat request (ARQ) are used depending on how retransmission is done when errors occur:

(a) Stop-and-wait ARQ
(b) Go-back-*N* ARQ
(c) Selective repeat ARQ.

The stop-and-wait ARQ is based on the stop-and-wait flow control protocol, while the other two are based on the sliding window flow control protocol.

3.6.1 Stop-and-Wait ARQ

As stated earlier, the stop-and-wait ARQ is used to deal with errors that occur when the stop-and-wait flow control protocol is used. Errors can occur in four ways:

(a) The frame was corrupted in transit when going from source to sink.
(b) The frame was OK, but the ACK was corrupted in transit.
(c) The frame was lost in transit.
(d) The ACK was lost in transit.

Under the stop-and-wait ARQ, the source sends a frame and waits for a response, which can be an ACK or a NAK that can also be in the form of a timeout. When a NAK is received, the source resends the frame and keeps resending it until it receives an ACK after the frame has been correctly received at the destination. Some protocols permit a fixed maximum number of retransmissions after which the link is declared unusable. Figure 3.6 is an illustration of the scheme. In the figure, frame 1 is received in error and is retransmitted after the sender received the NAK.

3.6.2 Go-Back-*N* ARQ

The go-back-*N* ARQ deals with errors that occur when the sliding window protocol is used. When a NAK is received for a particular frame, the source resends that frame and all the frames that have been transmitted since that frame was sent as well as any new frames, provided the total number of frames being sent does not exceed *N*. Consider the example in Figure 3.7 where $N = 4$, which is essentially the window size. Frame F4 is received in error, but the sender got this information after it had already sent frames F5 and F6. Thus, these frames will be ignored by the receiver and the sender resends frames F4, F5, F6, and additionally frame F7 since its window size is 4.

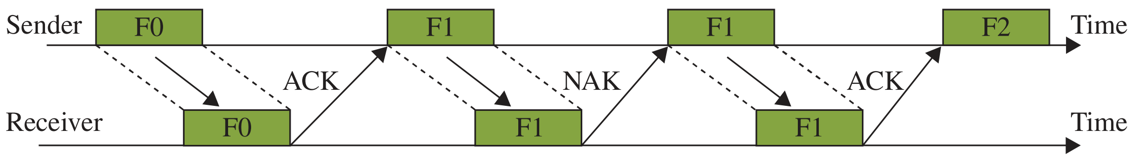


Figure 3.6 Illustration of the Stop-and-Wait ARQ.

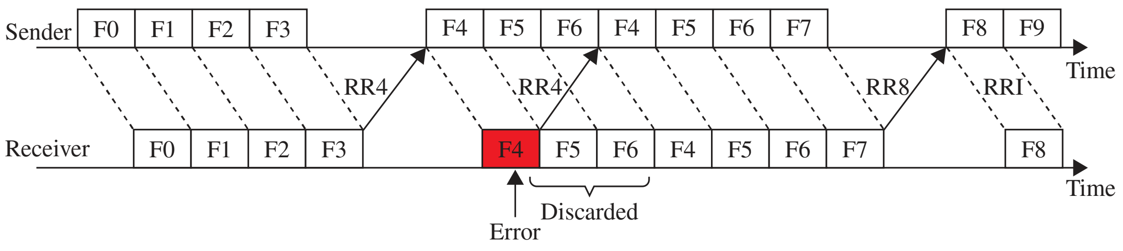


Figure 3.7 Example of Go-Back-4 ARQ.

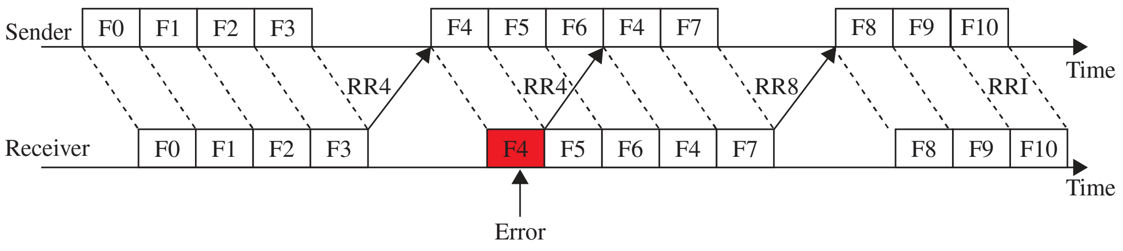


Figure 3.8 Illustration of the Selective Repeat ARQ.

3.6.3 Selective Repeat ARQ

One of the problems with the go-back-N scheme is that once a frame is found to be in error, all subsequent frames, including those that were not in error, are retransmitted. When the round-trip delay is high, as in satellite communication, the throughput of the channel becomes low. The selective repeat ARQ addresses this problem. Under this scheme, only the frame in error is retransmitted. The drawback is that the receiver must provide enough buffer to store the frames that were transmitted after the erroneous frame until the frame has been retransmitted. The reason for this is that the destination must resequence the frames and deliver them in the same order that they appear at the source. Thus, until the errored frame has been retransmitted and correctly received, the frames that are not errored must be stored in the buffer at the receiver.

Consider the operation of the scheme in the previous example assuming $W = 4$. This is illustrated in Figure 3.8 where unlike the go-back-4 scheme, only frame F4 is retransmitted instead of F4, F5, and F6 as in the go-back-4 scheme.

3.7 Data Link Control Protocols

DLC protocols supervise the retransmission of corrupted frames and regulate the transfer of frames. Thus, the job of DLC protocols is to ensure that data passed up to the next layer has been received *exactly as transmitted* (i.e., *error free, without loss, and in the correct order*), and also practice *flow control*, which ensures that data is transmitted only as fast as the receiver can receive it. As stated earlier, frames have a specific format and carry sequence numbers that enable the receiver to acknowledge them either individually or in groups. Different DLC protocols have been defined and they include the following:

(a) HDLC protocol
(b) PPP
(c) Frame relay.

The most commonly used DLC protocols are based on the same principles, namely, as follows:

(a) They are all bit-oriented and use bit stuffing for data transparency.
(b) They have the same frame structure where each frame has a leading and a trailing flag; there is an address field, a control field, and a data field.

They differ only in minor ways.

3.7.1 High-level Data Link Control

HDLC is a DLC protocol defined by the ISO for use in both point-to-point (P2P) and point-to-multipoint (or multidrop) data links. It supports full-duplex operation and is widely used in computer networks. HDLC operates in three modes:

(a) Normal response mode (NRM)
(b) Asynchronous response mode (ARM)
(c) Asynchronous balanced mode (ABM).

NRM is an unbalanced or asymmetric mode in which one end of the line is the master and the other end is the slave. It is used mainly in terminal-based networks where the slaves (or secondary stations) can transmit only when polled by the master (or primary). Thus, the primary station initiates transfers to the secondary station; the secondary station can only transmit a response when, and only when, it is instructed to do so by the primary station. In other words, the secondary station must receive explicit permission from the primary station to transfer a response. After receiving permission from the primary station, the secondary station initiates its transmission. This transmission from the secondary station to the primary station may be much more than just an acknowledgment of a frame. After each transmission, the secondary station must wait once again for explicit permission from the primary station to transfer anything. A *command frame* is a frame sent from the primary to the secondary, and a *response frame* is a frame sent from the secondary to the primary.

ARM is a scheme in which the secondary station does not have to wait to receive explicit permission from the primary station to transfer any frames. That is, the primary station does not need to initiate transfers by a secondary station. The frames may be more than just acknowledgment frames; they may contain data or control information regarding the status of the secondary station. This mode can reduce overhead on the link as no frames need to be transferred in order to give the secondary station permission to initiate a transfer. However, some limitations do exist. Due to the fact that this mode is asynchronous, the secondary station must wait until it detects an idle channel before it can transfer any frames. This is when the ARM link is operating at half duplex. If the ARM link is operating at full duplex, the secondary station can transmit at any time. In this mode, the primary station still retains responsibility for error recovery, link setup, and link disconnection.

ABM is used mainly on P2P links and each station has equal status and performs both primary and secondary functions. This mode uses combined stations. There is no need for permission on the part of any station in this mode because combined stations do not require any sort of instructions to perform any task on the link.

The NRM is used most frequently in multipoint lines, where the primary station controls the link. The ARM is better for point-to-point links as it reduces overhead. The ABM is not used widely today. The "asynchronous" in both ARM and ABM does not refer to the format of the data on the link; it refers to the fact that any given station can transfer frames without explicit permission or instruction from any other station.

3.7.1.1 HDLC Frame Format

The HDLC frame format is shown in Figure 3.9.

The frame consists of the following fields:

- *Flag*: Every frame on the link must begin and end with a flag sequence field containing the sequence 01111110.
- *Address*: The address field identifies the primary or secondary station involved in the frame transmission or reception. Each station on the link has a unique address. In an unbalanced configuration, the address field in both command and response frames refers to the secondary station. In a balanced configuration, the command frame contains the destination station's address and the response frame has the sending station's address.
- *Control*: HDLC uses the control field to determine how to control the communication process. This field contains the commands, responses, and sequence numbers used to maintain the data flow accountability of the link,

Bytes	1	1	1	Variable	2	1
	Flag	Address	Control	Data	Checksum	Flag

Figure 3.9 HDLC Frame Format.

defines the functions of the frame, and initiates the logic to control the movement of traffic between sending and receiving stations.

- *Information/data*: This field is not in all HDLC frames; it is only present when the information format is being used in the control field. The information field contains the actual data the sender is transmitting to the receiver.
- *FCS*: This field contains a 16-bit CRC that is used for error detection.

3.7.1.2 Control Field Format

The control field has three formats:

(a) *Information format*, which is used to indicate that the frame is used to transmit end-user data between two devices.
(b) *Supervisory format*, which is used to indicate that the control field is performing control functions such as acknowledgment of frames, requests for retransmission, and requests for temporary suspension of frames being transmitted. Its use depends on the operational mode being used.
(c) *Unnumbered format*, which is used to indicate that a frame is being used for control purposes. It is used to perform link initialization, link disconnection, and other link control functions.

These formats are shown in Figure 3.10.

- *Seq* is the sequence number of the current frame.
- *Next* is the piggybacked acknowledgment and the value is the sequence number of the next frame expected by the station transmitting this frame.
- *Poll/final bit* (*P*/*F*) is called the poll/final (or *P*/*F*) bit, which is ignored when it is set to 0 but is used to provide dialogue between the primary and the secondary stations. The primary station uses $P = 1$ to acquire a status response from the secondary station; the *P* bit signifies a poll. The secondary station responds to the *P* bit by transmitting a data or status frame to the primary

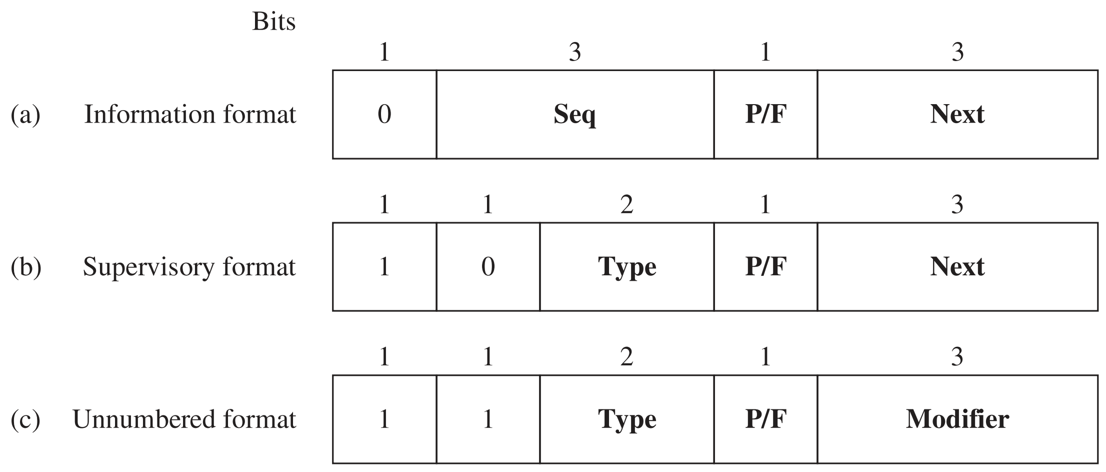


Figure 3.10 Control Filed Formats.

station with the *P*/*F* bit set to $F = 1$. The *F* bit can also be used to signal the end of a multiframe transmission from the secondary station under NRM.

In supervisory frame, different *types* are defined:

(a) Type 0 is *RR*, used by the primary or secondary station to indicate that it is ready to receive an information frame and/or acknowledge previously received frames.
(b) Type 1 is *reject* (REJ), which denotes negative acknowledgment due to transmission error and thus is used to request the retransmission of frames.
(c) Type 2 is *receive not ready* (RNR), which is used to indicate that the primary or secondary station is not ready to receive any information frames or acknowledgments.
(d) Type 3 is *selective reject* (SREJ), which is used by a station to request retransmission of specific frames.

The unnumbered format frames have five bits (two from type and three from modifier) that provide up to 32 additional commands and 32 additional response functions. These include the following:

- *Set normal response mode* (SNRM) places the secondary station into NRM. NRM does not allow the secondary station to send any unsolicited frames. Hence the primary station has control of the link.
- *Set asynchronous response mode* (SARM) allows a secondary station to transmit frames without a poll from the primary station.
- *Set asynchronous balanced mode* (SABM) sets the operational mode of the link to ABM.
- *Disconnect* (*DISC*) places the secondary station into a disconnected mode.
- *Set normal response mode extended* (SNRME) increases the size of the control field to two octets instead of one in NRM. This is used for extended sequencing. The same applies for *SARME* and *SABME*.
- *Set initialization mode* (SIM) is used to cause the secondary station to initiate a station-specific procedure(s) to initialize its data link level control functions.
- *Unnumbered information* (UI) is used to send information to a secondary station.
- *Unnumbered acknowledgment* (UA) is used by the secondary station to acknowledge the receipt and acceptance of an *SNRM, SARM, SABM, SNRME, SARME, SABME, RSET, SIM,* or *DISC* commands.

3.7.2 Point-to-Point Protocol

PPP was originally designed as an encapsulation protocol for transporting IP traffic over P2P links. It also established a standard for the assignment and management of IP addresses, asynchronous (start/stop) and bit-oriented synchronous encapsulation, network protocol multiplexing, link configuration, link quality testing, error detection, and option negotiation for such capabilities as network layer address negotiation and data-compression negotiation. PPP supports these functions by providing an extensible link control protocol (LCP) and a family of network control protocols (NCPs) to negotiate optional configuration parameters and facilities. In addition to IP, PPP supports other network layer protocols, such as the Novell's internetwork packet exchange (IPX) and DECnet.

3.7.2.1 PPP Components

PPP provides a method for transmitting datagrams over serial P2P links and contains three main components:

(a) A method for encapsulating datagrams over serial links. PPP uses the HDLC protocol as a basis for encapsulating datagrams over P2P links.
(b) An extensible LCP to establish, configure, and test the data link connection.
(c) A family of NCPs for establishing and configuring different network layer protocols; PPP is designed to allow the simultaneous use of multiple network layer protocols.

To establish communications over a P2P link, the originating PPP first sends LCP frames to configure and (optionally) test the data link. After the link has been established and optional facilities have been negotiated as needed by the LCP, the originating PPP sends NCP frames to choose and configure one or more network layer protocols. When each of the chosen network layer protocols has been configured, packets from each network layer protocol can be sent over the link. The link will remain configured for communications until explicit LCP or NCP frames close the link, or until some external event occurs (e.g., an inactivity timer expires or a user intervenes). PPP uses the principles, terminology, and frame structure of the HDLC procedures.

3.7.2.2 PPP Frame Format

The format of PPP is shown in Figure 3.11.

The fields are as follows:

- *Flag*: A single byte consisting of the binary sequence 01111110 that indicates the beginning or end of a frame.

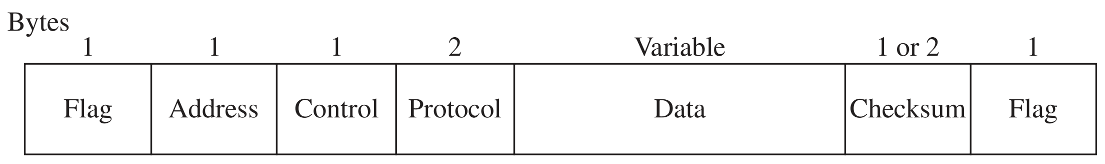


Figure 3.11 PPP Frame Format.

- *Address*: A single byte that contains the binary sequence 11111111, the standard broadcast address; PPP does not assign individual station addresses.
- *Control*: A single byte that contains the binary sequence 00000011, (hexadecimal 0x03), which is the unnumbered information (UI) command in HDLC with the *P*/*F* bit set to zero. Frames with other control field values are silently discarded.
- *Protocol*: Two bytes that identify the protocol encapsulated in the information field of the frame and content is written in hexadecimal format. The protocol field values in the "cxxx" range identify datagrams as belonging to the LCP or associated protocols. Values in the "8xxx" range identify datagrams belonging to the family of NCPs. Finally, values in the "0xxx" range identify the network protocol of specific datagrams.
- *Data* consists of zero or more bytes that contain the datagram for the protocol specified in the protocol field. The end of the information field is found by locating the closing flag sequence and allowing two bytes for the FCS field. The default maximum length of the information field is 1500 bytes. By prior agreement, consenting PPP implementations can use other values for the maximum information field length.
- FCS is normally 16 bits (2 bytes), but by prior agreement, consenting PPP implementations can use a 32-bit (4-byte) FCS for improved error detection.

The LCP can negotiate modifications to the standard PPP frame structure, but modified frames will always be clearly distinguishable from standard frames.

3.7.2.3 PPP Link Control

The PPP LCP provides a method of establishing, configuring, maintaining, and terminating the P2P connection. LCP goes through four distinct phases. First, link establishment and configuration negotiation occur. Before any network layer datagrams (e.g., IP) can be exchanged, LCP first must open the connection and negotiate configuration parameters. This phase is complete when a configuration-acknowledgment frame has been both sent and received. This is followed by link quality determination. LCP allows an optional link quality determination phase following the link-establishment and configuration-negotiation phase. In this phase, the link is tested to determine whether the link quality is sufficient to bring up network layer protocols. (This phase is optional.) LCP can delay transmission of network layer protocol information until this phase is complete. At this point, network layer protocol configuration negotiation occurs. After LCP has finished the link quality determination phase, network layer protocols can be configured separately by the appropriate NCP and can be brought up and taken down at any time. If LCP closes the link, it informs the network layer protocols so that they can take appropriate action. Finally, link termination occurs; LCP can terminate the link at any time. This usually is done at the request of a user but can happen because of a physical event, such as the loss of carrier or the expiration of an idle-period timer.

Three classes of LCP frames exist:

(a) Link-establishment frames are used to establish and configure a link.
(b) Link-termination frames are used to terminate a link, and
(c) Link-maintenance frames are used to manage and debug a link.

These frames are used to accomplish the work of each of the LCP phases.

3.8 Summary

The data link layer is responsible for organizing data in frames and for detecting errors that occur in a frame. It is also responsible for hop-by-hop (or link-by-link) flow control. In this chapter, we have discussed different data link layer flow control schemes as well as error control schemes. We have also discussed the HDLC protocol that is the basis on which many other data link layer protocols were defined and the PPP.

Exercises

1 Assume that the following bitstream is to be transmitted as the payload of a system that uses the HDLC protocol:

011111001101111101111101110111110.

What bitstream is actually transmitted, given that bit stuffing is practiced?

2 Consider the following bit-stuffed bitstream that is received at the destination:

11001111101100111011111011111001111O

What is the bitstream that was presented to the source where the bit stuffing took place? (Note that this is asking for the original bitstream before bit stuffing was practiced at the source.)

3 Assume that the following block of messages is to be transmitted together:

0011011, 1100110, 1010110, 0101110, 1110001

Construct a two-dimensional parity check matrix for the block of messages using even parity.

4 Consider a sender–receiver arrangement for the following go-back-3 ARQ system:

- Frames 0, 1, 2, and 3 are sent, and the four frames are acknowledged.
- Frames 4, 5, 6, and 7 are sent, and NAK is received for frame 5.

(a) Draw the diagram for this sequence of transmissions.

(b) Assuming the sender has frames 8 and 9 to send after receiving the NAK for frame 5, show the next frame transmission sequence after frame 5 is retransmitted.

5 Consider a system with the generator bitstream $g = 1011$. We are interested in transmitting the data (or message) $m = 1001001$.

(a) Write down the generator polynomial $G(x)$.

(b) What is the order of the generator polynomial?

(c) Write down the message polynomial $M(x)$.

(d) Perform the cyclic redundancy check (CRC) computation on the message and indicate the bitstream that is transmitted.

(e) Assume that the bitstream 1001001011 is received at the destination. Show why it should or should not be accepted as being error-free.

6 A data link layer protocol uses the cyclic redundancy check (CRC) error detection technique with the generator polynomial $G(x) = x^5 + x^4 + x^2 + 1$.

(a) If the codeword 1010001101 is to be transmitted in this system, what codeword is actually sent?

(b) If the codeword 101000101001111 is received in this system, should it be accepted or rejected?

7 What is the difference between the normal response mode of operation of the high-level data link control (HDLC) protocol and the asynchronous response mode of operation of the HDLC protocol?