UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

INTRODUCTION TO DATALINK LAYER

INTRODUCTION:

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Figure 2.1 shows the same scenario.

Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

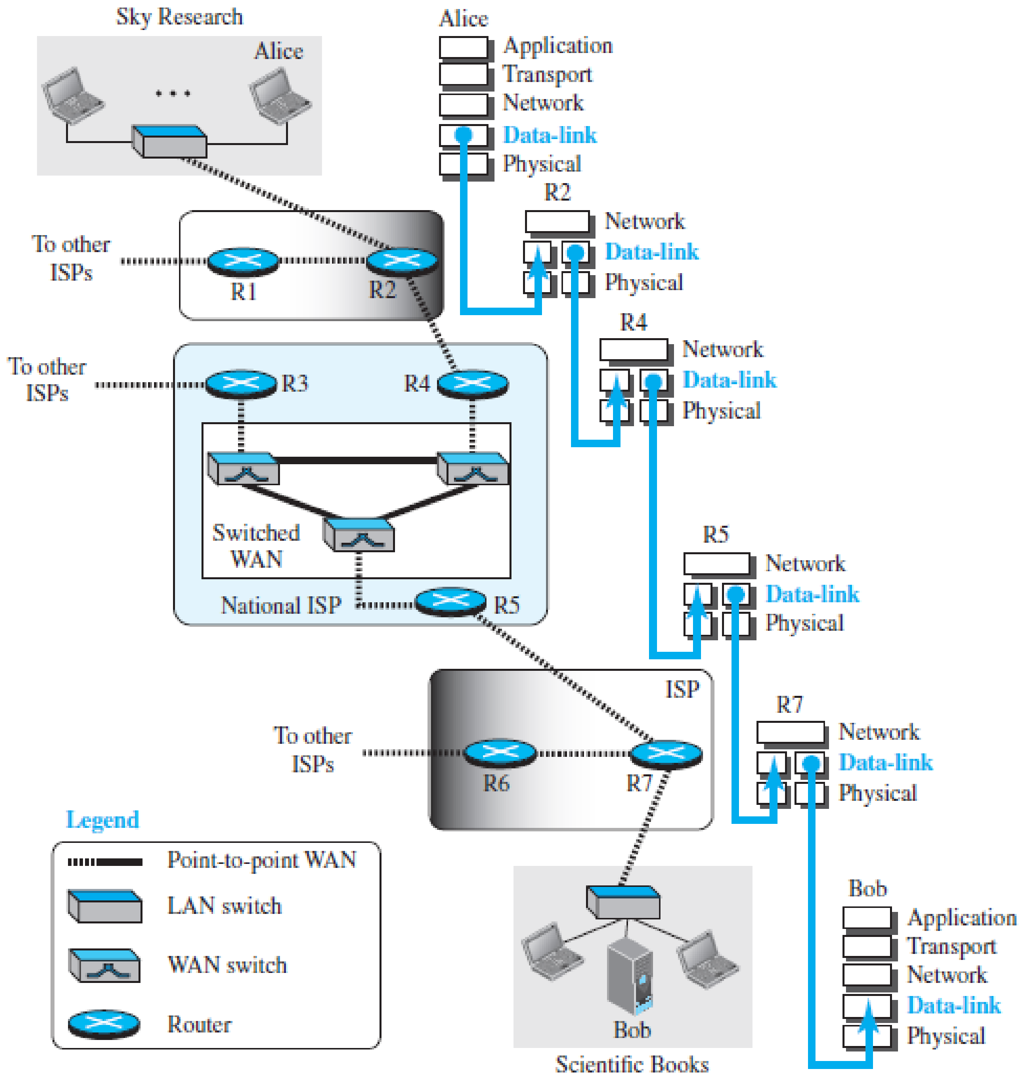


FIGURE 2.1: COMMUNICATION AT THE DATA-LINK LAYER

UNIT-II

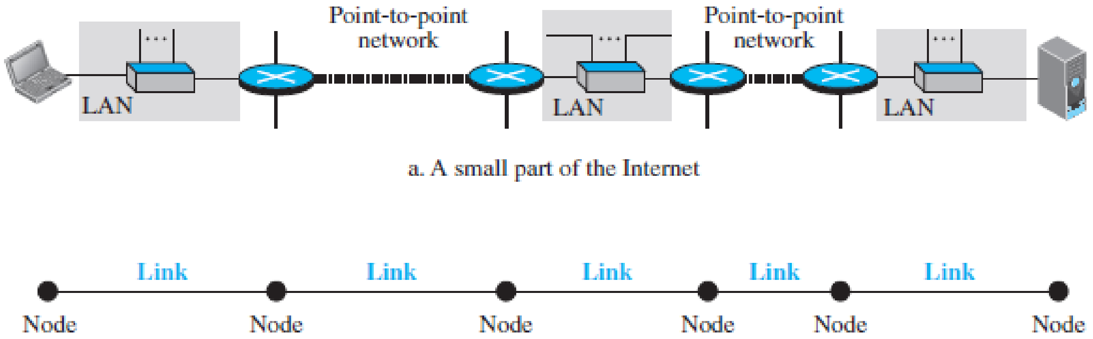
INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

The data-link layer at Alice's computer communicates with the data-link layer at router R2. The data-link layer at router R2 communicates with the data-link layer at router R4, and so on. Finally, the data-link layer at router R7 communicates with the data-link layer at Bob's computer. Only one data-link layer is involved at the source or the destination, but two data-link layers are involved at each router.

The reason is that Alice's and Bob's computers are each connected to a single network, but each router takes input from one network and sends output to another network. Note that although switches are also involved in the data-link-layer communication, for simplicity we have not shown them in the figure.

NODES AND LINKS:

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. Theses LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as ***nodes*** and the networks in between as ***links.*** Figure 2.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.



a. A small part of the Internet

b. Nodes and links

FIGURE 2.2: NODES AND LINKS

The first node is the source host; the last node is the destination host. The other four nodes are four routers. The first, the third, and the fifth links represent the three LANs; the second and the fourth links represent the two WANs.

SERVICES:

The data-link layer is located between the physical and the network layers. The data link layer provides services to the network layer; it receives services from the physical layer.

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL

CONNECTING DEVICES & VIRTUAL LANS

The duty scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path.

For this purpose, the data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame.

In other words, the data-link layer of the source host needs only to encapsulate, the data-link layer of the destination host needs to decapsulate, but each intermediate node needs to both encapsulate and decapsulate.

Figure 2.3 shows the encapsulation and decapsulation at the data-link layer. For simplicity, we have assumed that we have only one router between the source and destination. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router.

The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host. Note that, although we have shown only two data-link layers at the router, the router actually has three data-link layers because it is connected to three physical links.

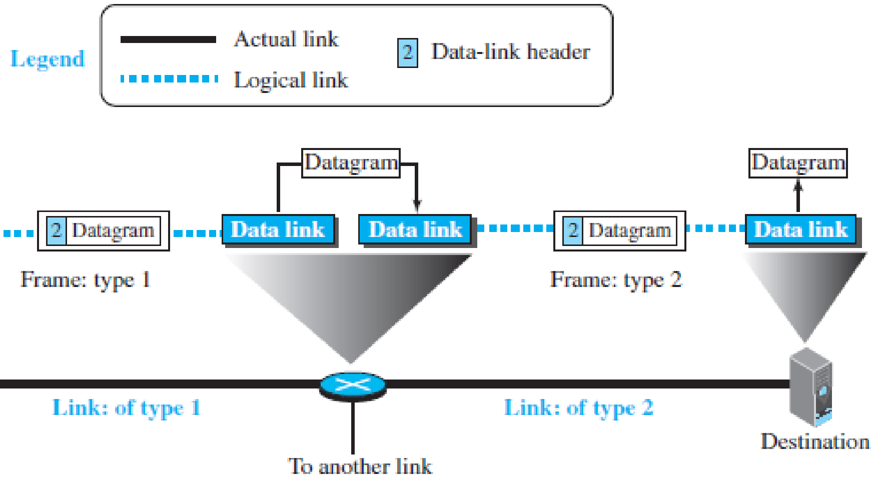


FIGURE 2.3: A COMMUNICATION WITH ONLY THREE NODES

With the contents of the above figure in mind, we can list the services provided by a data-link layer as shown below.

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL

CONNECTING DEVICES & VIRTUAL LANS

FRAMING: Definitely, the first service provided by the data-link layer is **framing**. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a **frame** before sending it to the next node.

The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer. Different data-link layers have different formats for framing.

FLOW CONTROL: Whenever we have a producer and a consumer, we need to think about flow control. If the producer produces items that cannot be consumed, accumulation of items occurs. The sending data-link layer at the end of a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer.

If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed). Definitely, we cannot have an unlimited buffer size at the receiving side. We have two choices. The first choice is to let the receiving data-link layer drop the frames if its buffer is full.

The second choice is to let the receiving data-link layer send a feedback to the sending data-link layer to ask it to stop or slow down. Different data-link-layer protocols use different strategies for flow control.

ERROR CONTROL: At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame.

Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node.

CONGESTION CONTROL: Although a link may be congested with frames, which may result in frame loss, most data-link-layer protocols do not directly use a congestion control to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature.

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL

CONNECTING DEVICES & VIRTUAL LANS

TWO CATEGORIES OF LINKS: Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium; we can also have a data-link layer that uses only part of the capacity of the link.

In other words, we can have a *point-to-point link* or a *broadcast link*. In a point-to-point link, the link is dedicated to the two devices; in a broadcast link, the link is shared between several pairs of devices.

Two Sub layers:

To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sub layers: **data link control (DLC)** and **media access control (MAC).**

The data link control sub layer deals with all issues common to both point-to-point and broadcast links; the media access control sub layer deals only with issues specific to broadcast links.

LINK-LAYER ADDRESSING:

A *link-layer address* is sometimes called a *link address*, sometimes a *physical address*, and sometimes a *MAC address*.

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another. Figure 2.4 demonstrates the concept in a small internet.

In the internet in Figure 2.4, we have three links and two routers. We also have shown only two hosts: Alice (source) and Bob (destination). For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L).

Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown three frames, one in each link. Each frame carries the same datagram with the same source and destination addresses (**N1** and **N8**), but the link-layer addresses of the frame change from link to link.

In link 1, the link-layer addresses are L1 and L2. In link 2, they are L4 and L5. In link 3, they are L7 and L8.

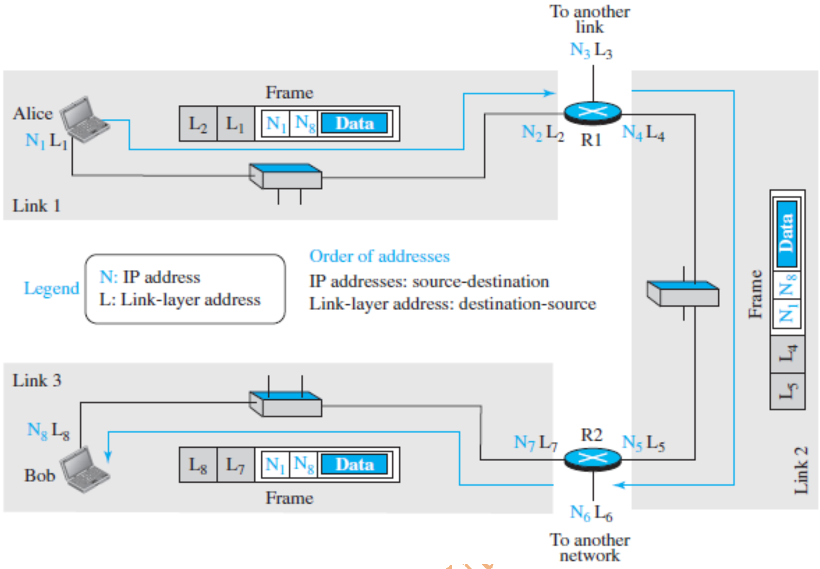


FIGURE 2.4: IP ADDRESSES AND LINK-LAYER ADDRESSES IN A SMALL INTERNET

Note that the IP addresses and the link-layer addresses are not in the same order. For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source.

THREE TYPES OF ADDRESSES:

Some link-layer protocols define three types of addresses: unicast, multicast, and broadcast.

UNICAST ADDRESS: Each host or each interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.

MULTICAST ADDRESS: Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).

BROADCAST ADDRESS: Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.

Address Resolution Protocol (ARP):

Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router.

Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host. However, the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the **Address Resolution Protocol (ARP)** becomes helpful.

The ARP protocol is one of the auxiliary protocols defined in the network layer, as shown in Figure 2.5. It belongs to the network layer, but we discuss it here because it maps an IP address to a logical-link address. ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

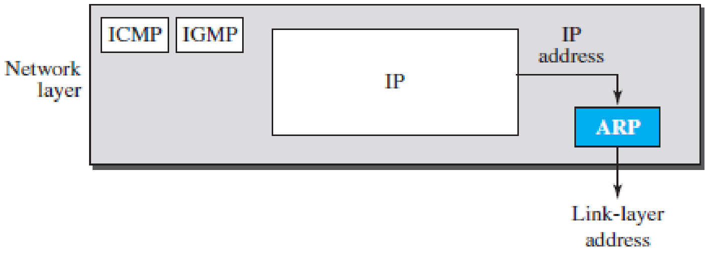


FIGURE 2.5: POSITION OF ARP IN TCP/IP PROTOCOL SUITE

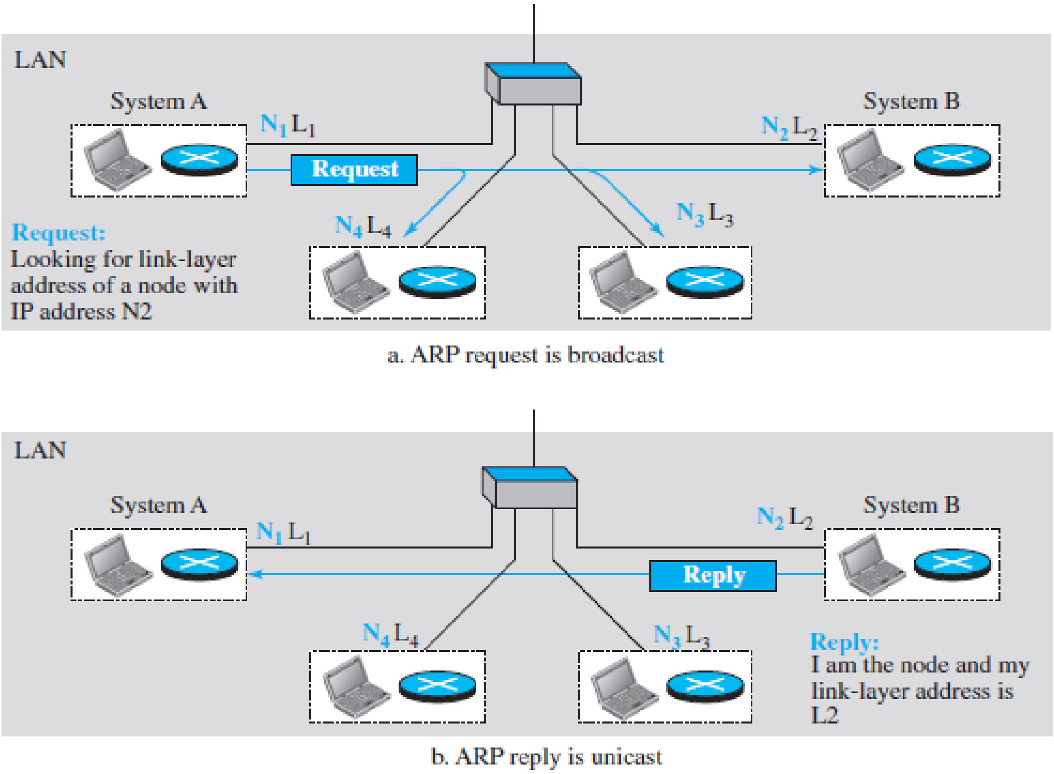
Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address (see Figure 2.6).

Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient’s IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.

In Figure 2.6a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address **N2**. System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient.

It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of **N2**. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 2.6b.

System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.



a. ARP request is broadcast

b. ARP reply is unicast

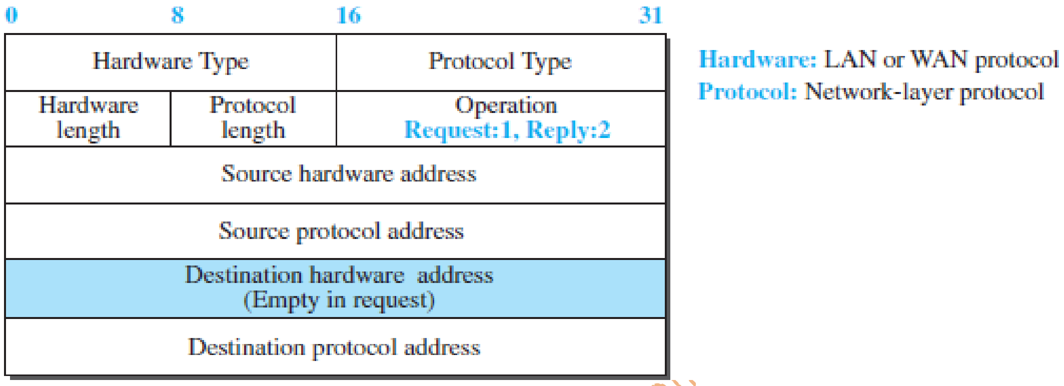
FIGURE 2.6: ARP OPERATION

Packet Format:

Figure 2.7 shows the format of an ARP packet. The names of the fields are self-explanatory. The *hardware type* field defines the type of the link-layer protocol; Ethernet is given the type 1.

The *protocol type* field defines the network-layer protocol: IPv4 protocol is (0800)16. The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender.

The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.



Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request:1, Reply:2
Source hardware address		
Source protocol address		
Destination hardware address (Empty in request)		
Destination protocol address		

Hardware: LAN or WAN protocol
Protocol: Network-layer protocol

FIGURE 2.7: ARP PACKET

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

ERROR DETECTION AND CORRECTION

Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted.

Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting **errors.**

Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy.

At the data-link layer, if a frame is corrupted between the two nodes, it needs to be corrected before it continues its journey to other nodes. However, most link-layer protocols simply discard the frame and let the upper-layer protocols handle the retransmission of the frame. Some multimedia applications, however, try to correct the corrupted frame.

INTRODUCTION:

Types of Errors:

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference.** This interference can change the shape of the signal. The term ***single-bit error*** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

The term ***burst error*** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1. Figure 2.8 shows the effect of a single-bit and a burst error on a data unit.

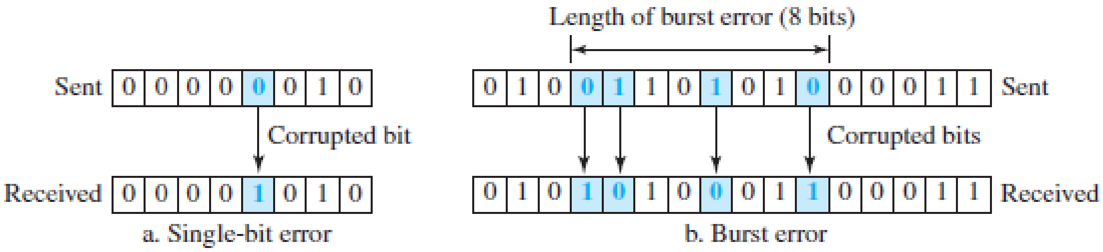


FIGURE 2.8: SINGLE-BIT AND BURST ERROR

A burst error is more likely to occur than a single-bit error because the duration of the noise signal is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits.

The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 second can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

Redundancy:

The central concept in detecting or correcting errors is **redundancy***.* To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

Detection versus Correction:

The correction of errors is more difficult than the detection. In **error detection***,* we are only looking to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of corrupted bits. A single-bit error is the same for us as a burst error.

In **error correction***,* we need to know the exact number of bits that are corrupted and, more importantly, their location in the message. The number of errors and the size of the message are important factors.

If we need to correct a single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 (permutation of 8 by 2) possibilities. You can imagine the receiver’s difficulty in finding 10 errors in a data unit of 1000 bits.

Coding:

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect errors.

The ratio of redundant bits to data bits and the robustness of the process are important factors in any coding scheme. We can divide coding schemes into two broad categories: **block coding** and **convolution coding***.*

NOTE: ***Convolution coding is more complex and beyond the scope of this book***

BLOCK CODING:

In block coding, we divide our message into blocks, each of *k* bits, called ***datawords*.** We add *r* redundant bits to each block to make the length *n* = *k* + *r*. The resulting *n*-bit blocks are called ***codewords*.**

Error Detection:

If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.
2. The original codeword has changed to an invalid one.

Figure 2.9 shows the role of block coding in error detection. The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding.

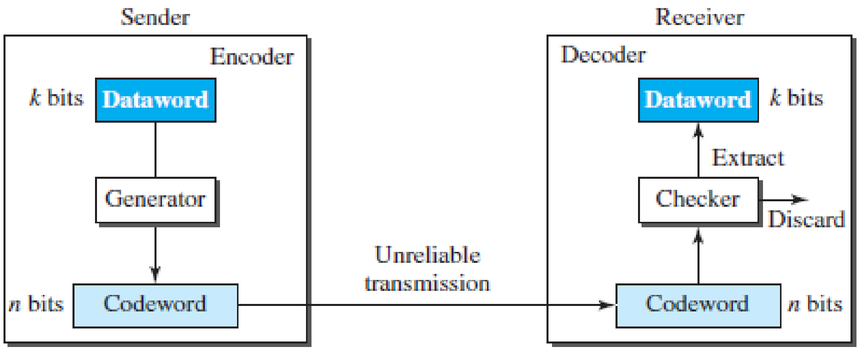
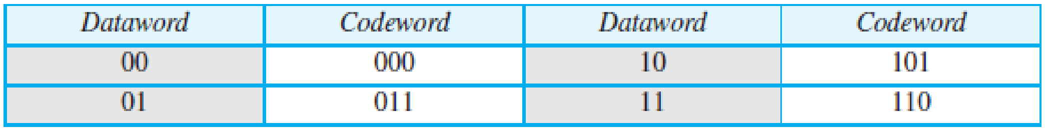


FIGURE 2.9: PROCESS OF ERROR DETECTION IN BLOCK CODING

Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.

Example 2.a:

Let us assume that *k* = 2 and *n* = 3. Table 2.1 shows the list of datawords and codewords.



Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

TABLE 2.1: A CODE FOR ERROR DETECTION IN EXAMPLE 2.a

Hamming Distance:

One of the central concepts in coding for error control is the idea of the Hamming distance. The **Hamming distance** between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words *x* and *y* as ***d***(*x*, *y*). We may wonder why Hamming distance is important for error detection.

The reason is that the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is ***d***(00000, 01101) = 3.

In other words, if the Hamming distance between the sent and the received codeword is not zero, the codeword has been corrupted during transmission.

The Hamming distance can easily be found if we apply the XOR operation (⊕) on the two words and count the number of 1s in the result. Note that the Hamming distance is a value greater than or equal to zero

Example 2.b:

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance ***d***(000, 011) is 2 because (000 ⊕ 011) is 011 (two 1s).
2. The Hamming distance ***d***(10101, 11110) is 3 because (10101 ⊕ 11110) is 01011 (three 1s).

Minimum Hamming Distance for Error Detection:

In a set of codewords, the **minimum Hamming distance** is the smallest Hamming distance between all possible pairs of codewords.

Now let us find the minimum Hamming distance in a code if we want to be able to detect up to *s* errors. If *s* errors occur during transmission, the Hamming distance between the sent codeword and received codeword is *s*.

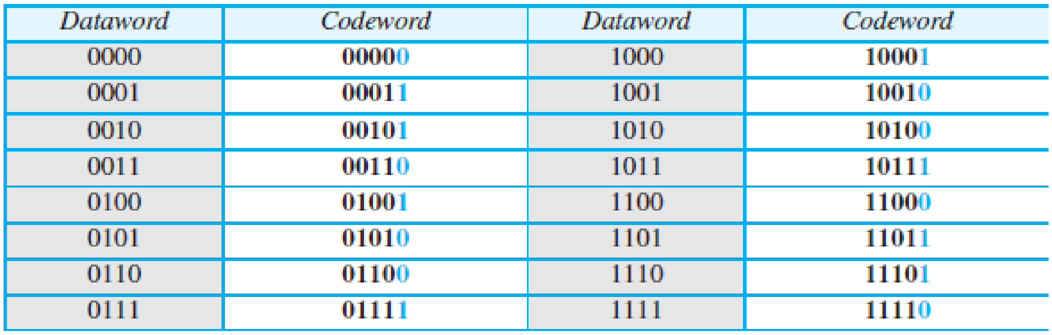
If our system is to detect up to *s* errors, the minimum distance between the valid codes must be (*s* + 1), so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is (*s* + 1), the received codeword cannot be erroneously mistaken for another codeword.

The error will be detected. We need to clarify a point here: Although a code with $d_{min} = s + 1$ may be able to detect more than *s* errors in some special cases, only *s* or fewer errors are guaranteed to be detected.

Parity-Check Code:

Perhaps the most familiar error-detecting code is the **parity-check code.** This code is a linear block code. In this code, a *k*-bit dataword is changed to an *n*-bit codeword where *n* = *k* + 1. The extra bit, called the *parity bit,* is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s, we discuss the even case.

The minimum Hamming distance for this category is *d*min = 2, which means that the code is a single-bit error-detecting code. Our first code (table 2.1) is a parity-check code (*k* = 2 and *n* = 3). The code in table 2.2 is also a parity-check code with *k* = 4 and *n* = 5.



Dataword	Codeword	Dataword	Codeword
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

TABLE 2.2: SIMPLE PARITY-CHECK CODE C(5, 4)

CYCLIC CODES:

Cyclic codes are special linear block codes with one extra property. In a **cyclic code,** if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

In this case, if we call the bits in the first word *a*0 to *a*6, and the bits in the second word *b*0 to *b*6, we can shift the bits by using the following:

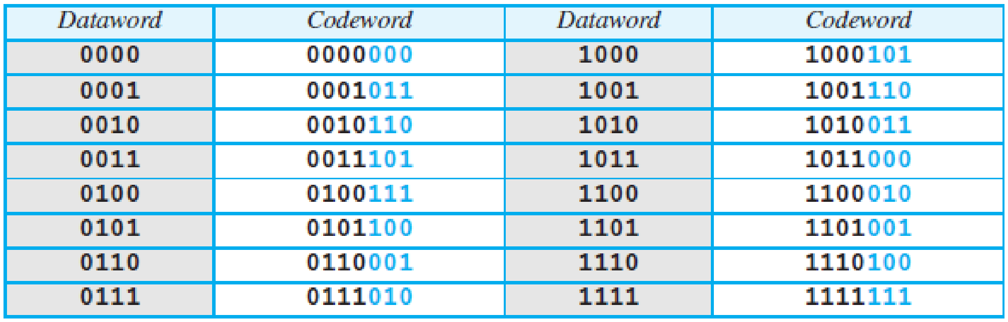
$$b_1 = a_0 \qquad b_2 = a_1 \qquad b_3 = a_2 \qquad b_4 = a_3 \qquad b_5 = a_4 \qquad b_6 = a_5 \qquad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check:

We can create cyclic codes to correct errors. In this section, we simply discuss a subset of cyclic codes called the **cyclic redundancy check (CRC)**, which is used in networks such as LANs and WANs.

Table 2.3 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.



Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

TABLE 2.3: A CRC CODE WITH C(7, 4)

Figure 2.10 shows one possible design for the encoder and decoder.

In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* − *k* (3 here) 0s to the right-hand side of the word. The *n*-bit result is fed into the generator.

The generator uses a divisor of size *n* − *k* + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (*r*2*r*1*r*0) is appended to the dataword to create the codeword.

The decoder receives the codeword (possibly corrupted in transition). A copy of all *n* bits is fed to the checker, which is a replica of the generator.

The remainder produced by the checker is a syndrome of *n* − *k* (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

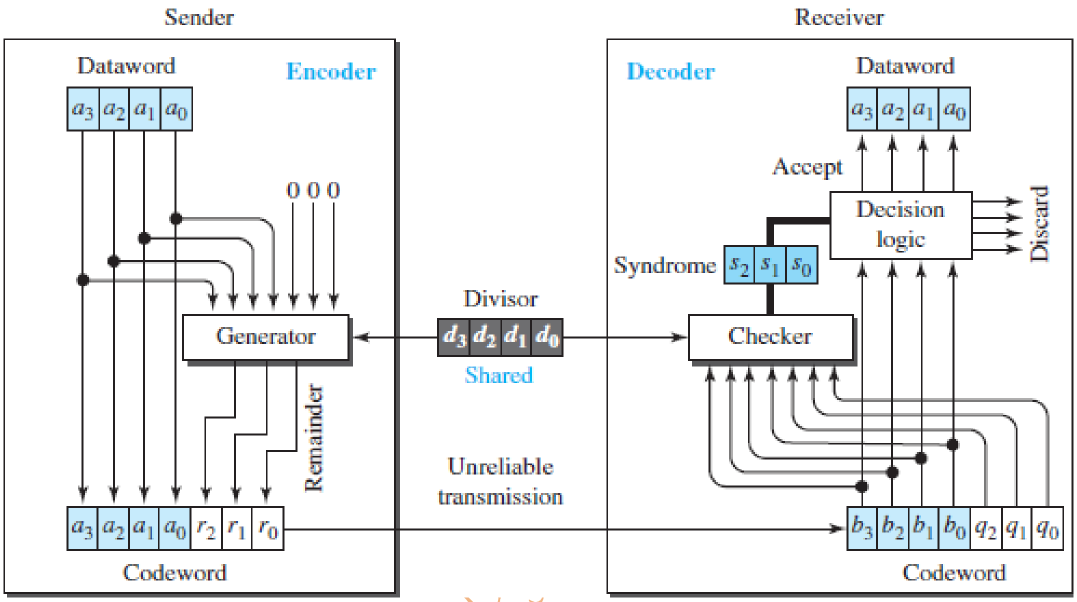


FIGURE 2.10: CRC ENCODER AND DECODER

Encoder:

Let us take a closer look at the encoder. The encoder takes a dataword and augments it with *n* − *k* number of 0s. It then divides the augmented dataword by the divisor, as shown in Figure 2.11.

Decoder:

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error with a high probability; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded.

Figure 2.12 shows two cases: The left-hand figure shows the value of the syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is a single error. The syndrome is not all 0s (it is 011).

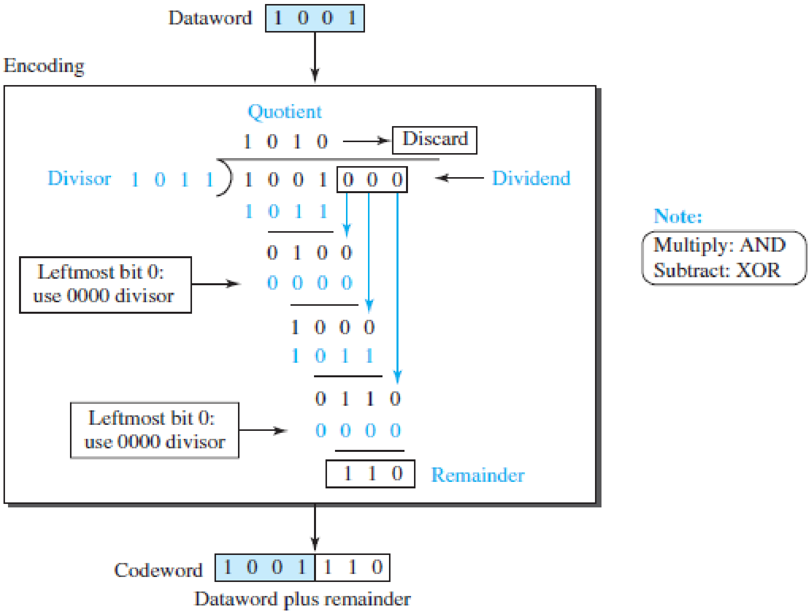


FIGURE 2.11: DIVISION IN CRC ENCODER

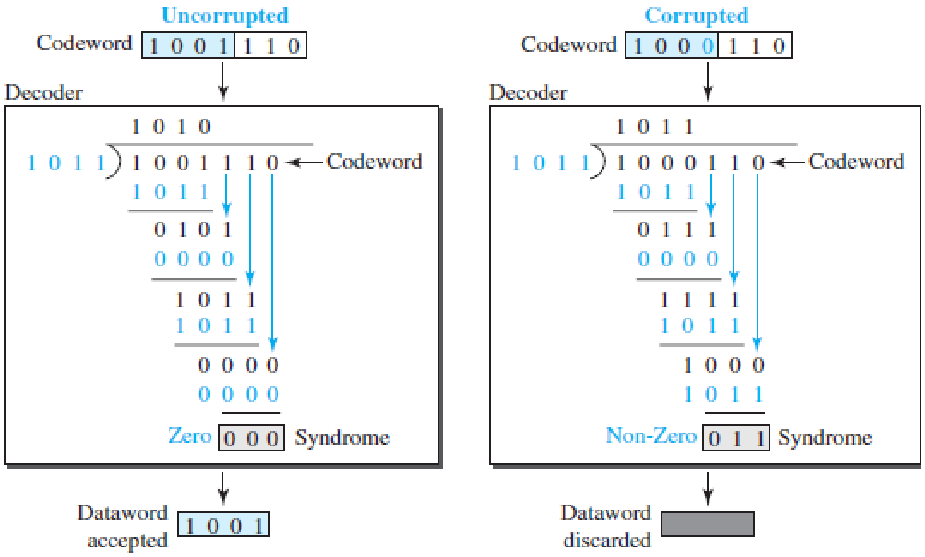


FIGURE 2.12: DIVISION IN THE CRC DECODER FOR TWO CASES

ADVANTAGES OF CYCLIC CODES:

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware. This has made cyclic codes a good candidate for many networks.

CHECKSUM:

Checksum is an error-detecting technique that can be applied to a message of any length. In the Internet, the checksum technique is mostly used at the network and transport layer rather than the data-link layer.

At the source, the message is first divided into *m*-bit units. The generator then creates an extra *m*-bit unit called the ***checksum,*** which is sent with the message. At the destination, the checker creates a new checksum from the combination of the message and sent checksum. If the new checksum is all 0s, the message is accepted; otherwise, the message is discarded (Figure 2.13). Note that in the real implementation, the checksum unit is not necessarily added at the end of the message; it can be inserted in the middle of the message.

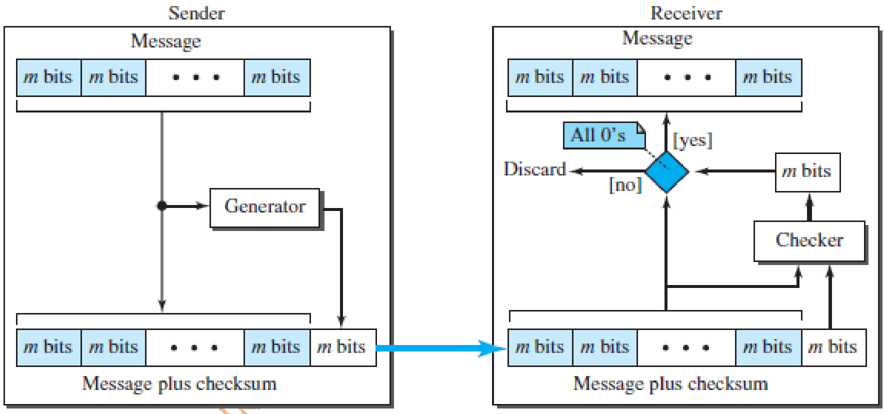


FIGURE 2.13: CHECKSUM

Concept

The idea of the traditional checksum is simple. We show this using a simple example.

Example 2.c:

Suppose the message is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers.

The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the message is not accepted.

One's Complement Addition:

The previous example has one major drawback. Each number can be written as a 4-bit word (each is less than 15) except for the sum. One solution is to use **one's complement** arithmetic.

In this arithmetic, we can represent unsigned numbers between 0 and $2^m - 1$ using only m bits. If the number has more than m bits, the extra leftmost bits need to be added to the m rightmost bits (wrapping).

Example 2.d:

In the previous example, the decimal number 36 in binary is (100100)2. To change it to a 4-bit number we add the extra leftmost bit to the right four bits as shown below.

$$\mathbf{(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}}$$

Instead of sending 36 as the sum, we can send 6 as the sum (7, 11, 12, 0, 6, **6**). The receiver can add the first five numbers in one's complement arithmetic. If the result is 6, the numbers are accepted; otherwise, they are rejected.

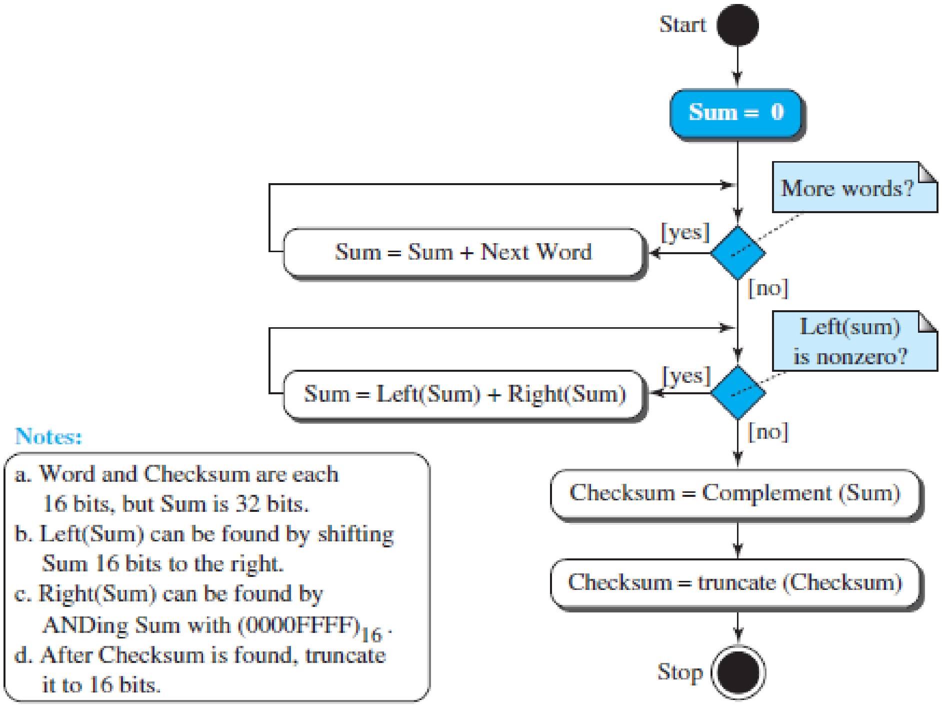
Algorithm

We can use the flow diagram of Figure 2.14 to show the algorithm for calculation of the checksum. A program in any language can easily be written based on the algorithm. Note that the first loop just calculates the sum of the data units in two's complement; the second loop wraps the extra bits created from the two's complement calculation to simulate the calculations in one's complement. This is needed because almost all computers today do calculation in two's complement.

FORWARD ERROR CORRECTION:

Retransmission of corrupted and lost packets is not useful for real-time multimedia transmission because it creates an unacceptable delay in reproducing: we need to wait until the lost or corrupted packet is resent.

We need to correct the error or reproduce the packet immediately. Several schemes have been designed and used in this case that is collectively referred to as **forward error correction** (**FEC**) techniques.



Notes:

a. Word and Checksum are each 16 bits, but Sum is 32 bits.
b. Left(Sum) can be found by shifting Sum 16 bits to the right.
c. Right(Sum) can be found by ANDing Sum with $(0000FFFF)_{16}$.
d. After Checksum is found, truncate it to 16 bits.

FIGURE 2.14: ALGORITHM TO CALCULATE A TRADITIONAL CHECKSUM

USING HAMMING DISTANCE:

We earlier discussed the Hamming distance for error detection. We said that to detect *s* errors, the minimum Hamming distance should be *d*min = *s* + 1. For error detection, we definitely need more distance.

It can be shown that to detect *t* errors, we need to have *d*min = 2*t* + 1. In other words, if we want to correct 10 bits in a packet, we need to make the minimum hamming distance 21 bits, which means a lot of redundant bits, need to be sent with the data.

To give an example, consider the famous BCH code. In this code, if data is 99 bits, we need to send 255 bits (extra 156 bits) to correct just 23 possible bit errors. Most of the time we cannot afford such a redundancy. We give some examples of how to calculate the required bits in the practice set.

USING XOR: Another recommendation is to use the property of the exclusive OR operation as shown below.

$$R = P_1 \oplus P_2 \oplus \ldots \oplus P_i \oplus \ldots \oplus P_N \quad \rightarrow \quad P_i = P_1 \oplus P_2 \oplus \ldots \oplus R \oplus \ldots \oplus P_N$$

In other words, if we apply the exclusive OR operation on *N* data items (P1 to P*N*), we can recreate any of the data items by exclusive-ORing all of the items, replacing the one to be created by the result of the previous operation (R).

This means that we can divide a packet into *N* chunks, create the exclusive OR of all the chunks and send *N* + 1 chunks. If any chunk is lost or corrupted, it can be created at the receiver site. Now the question is what the value of N should be. If *N* = 4, it means that we need to send 25 percent extra data and be able to correct the data if only one out of four chunks is lost.

CHUNK INTERLEAVING:

Another way to achieve FEC in multimedia is to allow some small chunks to be missing at the receiver. We cannot afford to let all the chunks belonging to the same packet be missing; however, we can afford to let one chunk be missing in each packet.

Figure 2.15 shows that we can divide each packet into 5 chunks (normally the number is much larger). We can then create data chunk by chunk (horizontally), but combine the chunks into packets vertically. In this case, each packet sent carries a chunk from several original packets. If the packet is lost, we miss only one chunk in each packet, which is normally acceptable in multimedia communication.

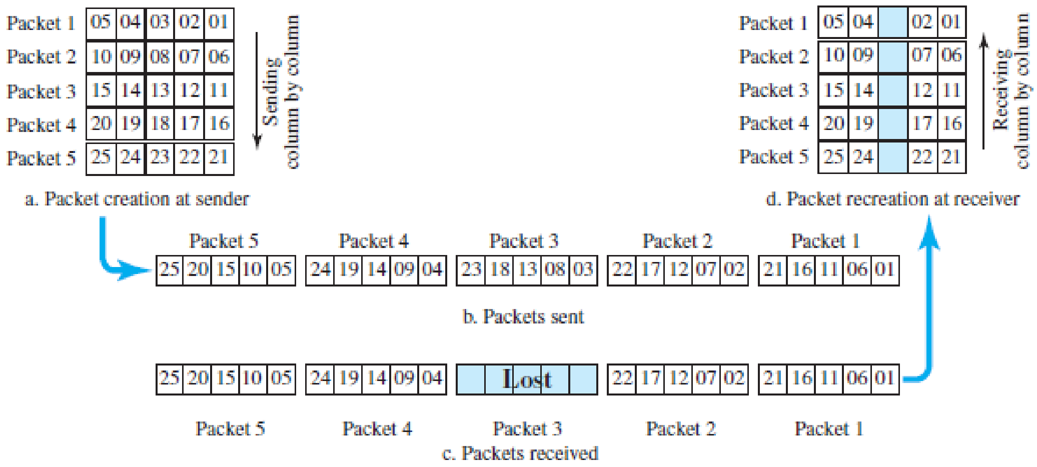


FIGURE 2.15: INTERLEAVING

COMBINING HAMMING DISTANCE AND INTERLEAVING:

Hamming distance and interleaving can be combined. We can first create *n*-bit packets that can correct *t*-bit errors. Then we interleave *m* rows and send the bits column by column. In this way, we can automatically correct burst errors up to *m* × *t*-bit errors.

COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS:

Still another solution is to create a duplicate of each packet with a low-resolution redundancy and combine the redundant version with the next packet. For example, we can create four low-resolution packets out of five high-resolution packets and send them as shown in Figure 2.16. If a packet is lost, we can use the low-resolution version from the next packet. Note that the low-resolution section in the first packet is empty.

In this method, if the last packet is lost, it cannot be recovered, but we use the low-resolution version of a packet if the lost packet is not the last one. The audio and video reproduction does not have the same quality, but the lack of quality is not recognized most of the time.

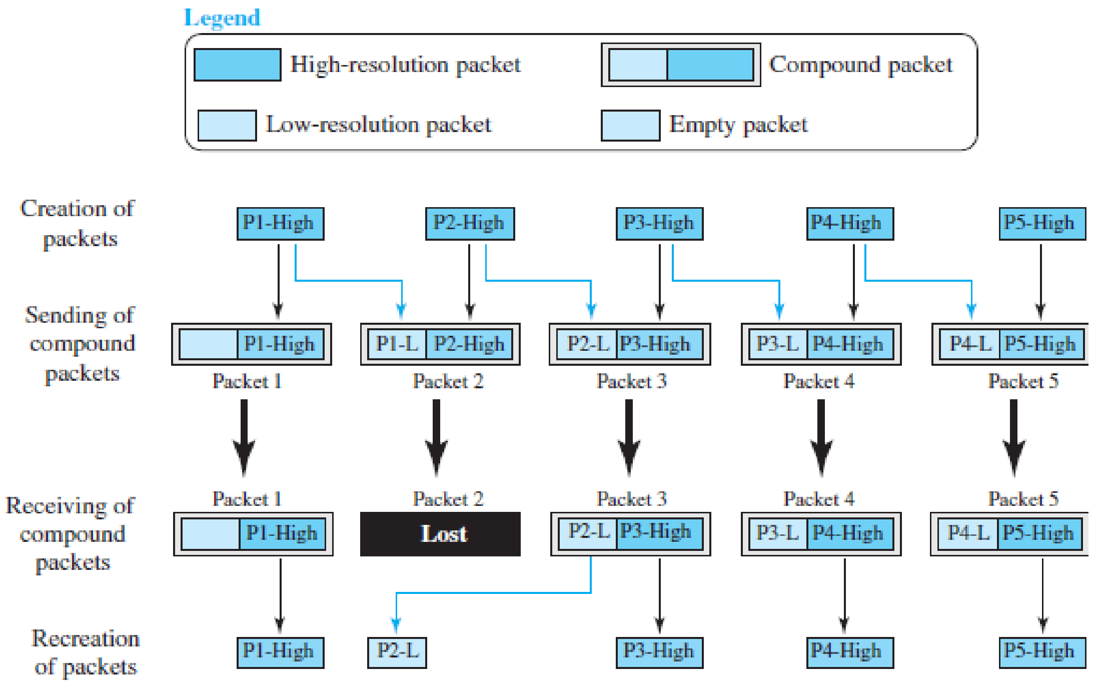


FIGURE 2.16: COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

DATA LINK CONTROL

DLC SERVICES:

The **data link control (DLC)** deals with procedures for communication between two adjacent nodes—node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include *framing* and *flow and error control*.

FRAMING:

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination. The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.

The data-link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another. *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, which 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 frame. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frame Size:

Frames can be of fixed or variable size. In *fixed-size framing,* there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM WAN, which uses frames of fixed size called *cells*.

In variable-size framing, we need a way to define the end of one frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

Character-Oriented Framing:

In *character-oriented (or byte-oriented) framing,* data to be carried are 8-bit characters from a coding system such as ASCII.

The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection redundant bits, are also multiples of 8 bits.

To separate one frame from the next, an 8-bit (1-byte) **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 2.17 shows the format of a frame in a character-oriented protocol.

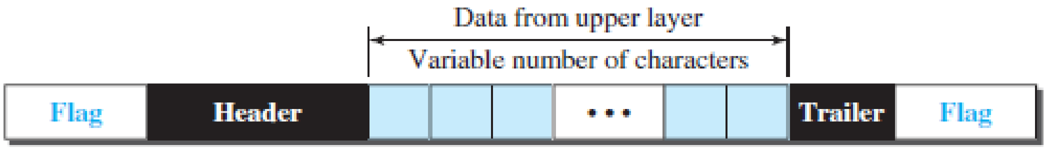


FIGURE 2.17: A FRAME IN A CHARACTER-ORIENTED PROTOCOL

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.

Now, however, we send other types of information such as graphs, audio, and video; any character used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame.

To fix this problem, a byte-stuffing strategy was added to character-oriented framing. In **byte stuffing** (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte.

This byte is usually called the *escape character (ESC)* and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag. Figure 2.18 shows the situation.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag?

To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

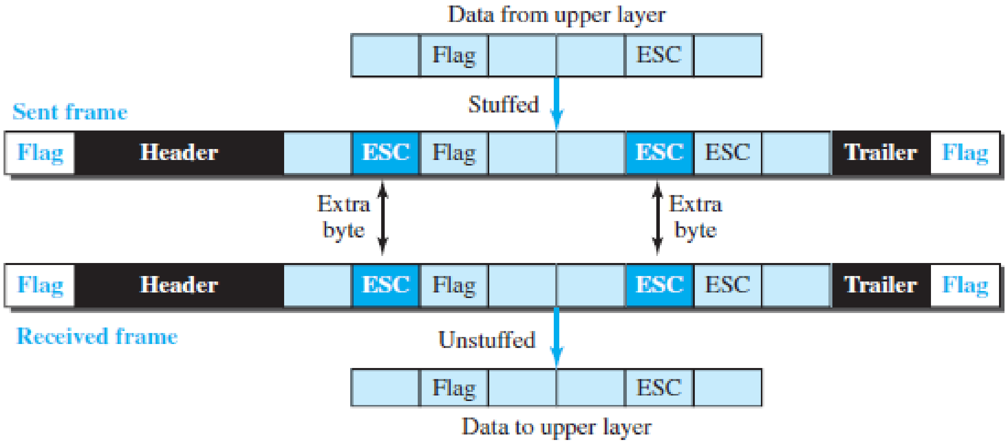


FIGURE 2.18: BYTE STUFFING AND UNSTUFFING

Bit-Oriented Framing:

In *bit-oriented framing,* 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 2.19.

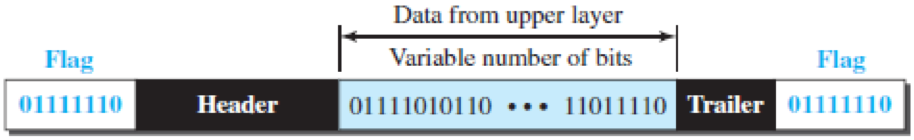


FIGURE 2.19: A FRAME IN A BIT-ORIENTED PROTOCOL

This flag can create the same type of problem we saw in the character-oriented protocols. That is, 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.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 2.20 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver. This means that if the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken for a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

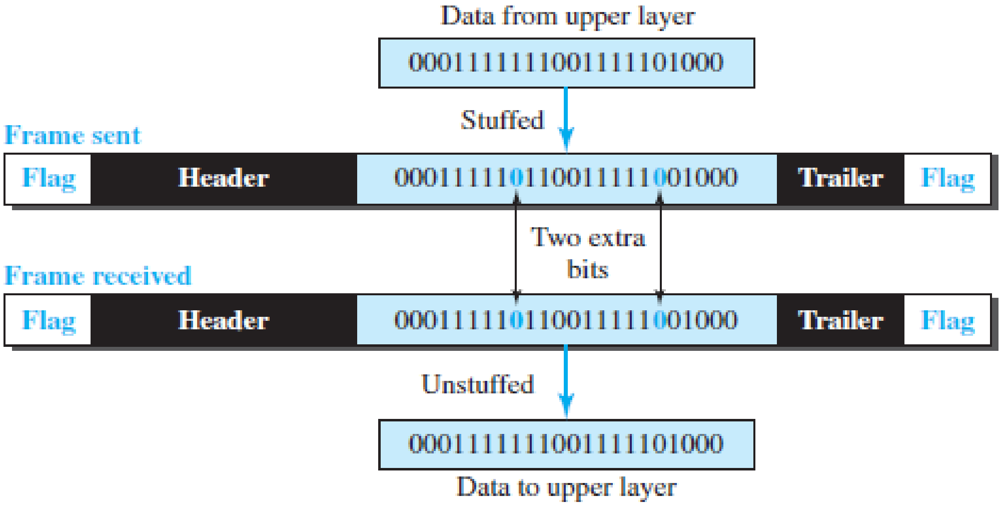


FIGURE 2.20: BIT STUFFING AND UNSTUFFING

FLOW AND ERROR CONTROL:

Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates. If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items.

If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.

In communication at the data-link layer, we are dealing with four entities: network and data-link layers at the sending node and network and data-link layers at the receiving node.

Although we can have a complex relationship with more than one producer and consumer, we ignore the relationships between networks and data-link layers and concentrate on the relationship between two data-link layers, as shown in Figure 2.21.

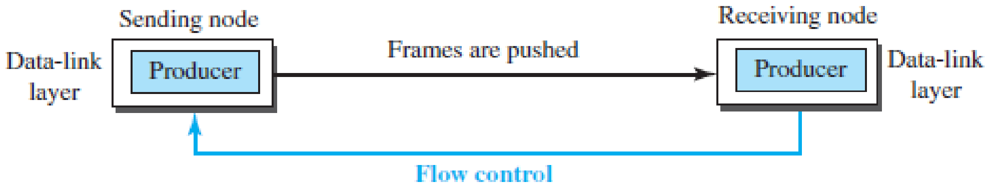


FIGURE 2.21: FLOW CONTROL AT THE DATA-LINK LAYER

Buffers:

Although flow control can be implemented in several ways, one of the solutions is normally to use two *buffers*; one at the sending data-link layer and the other at the receiving data-link layer.

A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control:

Since the underlying technology at the physical layer is not fully reliable, we need to implement error control at the data-link layer to prevent the receiving node from delivering corrupted packets to its network layer.

Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

- In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.
- In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

CONNECTIONLESS AND CONNECTION-ORIENTED:

A DLC protocol can be either connectionless or connection-oriented.

Connectionless Protocol:

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. Note that the term *connectionless* here does not mean that there is no physical connection (transmission medium) between the nodes; it means that there is no *connection* between frames. The frames are not numbered and there is no sense of ordering. Most of the data-link protocols for LANs are connectionless protocols.

Connection-Oriented Protocol:

In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase). After all frames that are somehow related to each other are transmitted (transfer phase), the logical connection is terminated (teardown phase). In this type of communication, the frames are numbered and sent in order.

If they are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer. Connection oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

DATA-LINK LAYER PROTOCOLS:

Traditionally four protocols have been defined for the data-link layer to deal with flow and error control: Simple, Stop-and-Wait, Go-Back-N, and Selective-Repeat. Although the first two protocols still are used at the data-link layer, the last two have disappeared.

The behavior of a data-link-layer protocol can be better shown as a **finite state machine (FSM).** An FSM is thought of as a machine with a finite number of states. The machine is always in one of the states until an *event* occurs.

Each event is associated with two reactions: defining the list (possibly empty) of actions to be performed and determining the next state (which can be the same as the current state). One of the states must be defined as the initial state, the state in which the machine starts when it turns on.

In Figure 2.22, we show an example of a machine using FSM. We have used rounded-corner rectangles to show states, colored text to show events, and regular black text to show actions.

A horizontal line is used to separate the event from the actions, although later we replace the horizontal line with a slash. The arrow shows the movement to the next state.

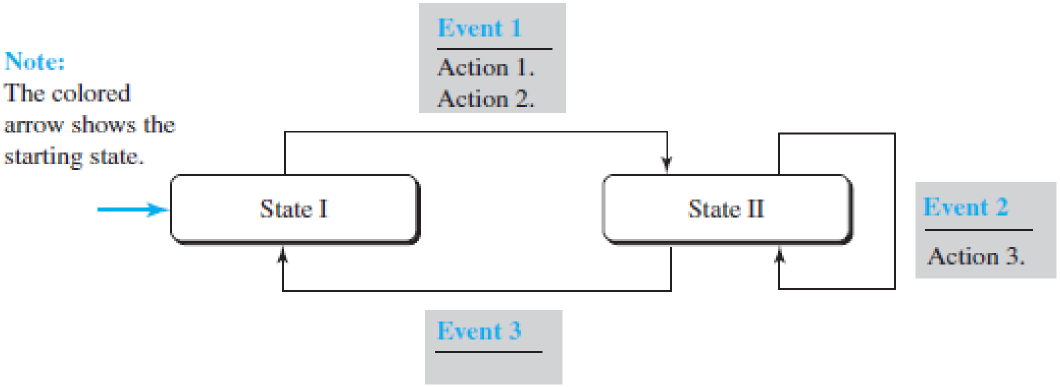


FIGURE 2.22: CONNECTIONLESS AND CONNECTION-ORIENTED SERVICE REPRESENTED AS FSMS

The figure shows a machine with three states. There are only three possible events and three possible actions. The machine starts in state I. If event 1 occurs, the machine performs actions 1 and 2 and moves to state II.

When the machine is in state II, two events may occur. If event 1 occurs, the machine performs action 3 and remains in the same state, state II. If event 3 occurs, the machine performs no action, but move to state I.

SIMPLE PROTOCOL:

Our first protocol is a **simple protocol** with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives. In other words, the receiver can never be overwhelmed with incoming frames. Figure 2.23 shows the layout for this protocol.

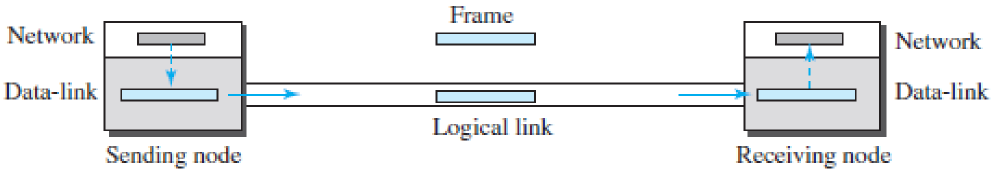


FIGURE 2.23: SIMPLE PROTOCOL

The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender and receiver provide transmission services for their network layers.

STOP-AND-WAIT PROTOCOL:

Our second protocol is called the **Stop-and-Wait protocol,** which uses both flow and error control. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame.

When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.

Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.

This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready.

Figure 2.24 shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

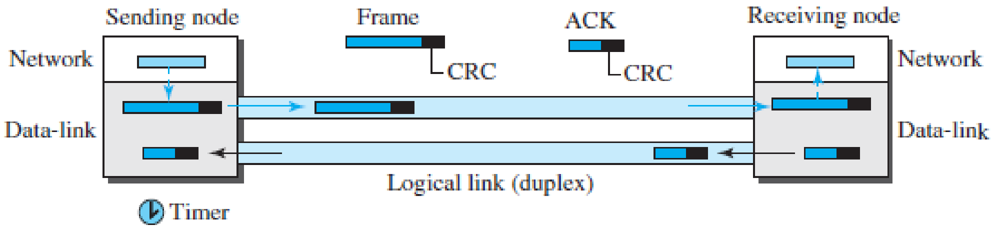


FIGURE 2.24: STOP-AND-WAIT PROTOCOL

Piggybacking:

The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction. Protocols have been designed in the past to allow data to flow in both directions. However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction.

In other words, when node A is sending data to node B, Node A also acknowledges the data received from node B. Because piggybacking makes communication at the data link layer more complicated, it is not a common practice.

HDLC:

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

Configurations and Transfer Modes:

HDLC provides two common transfer modes that can be used in different configurations: *normal response mode (NRM)* and *asynchronous balanced mode (ABM)*. 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 multipoint links, as shown in Figure 2.25.

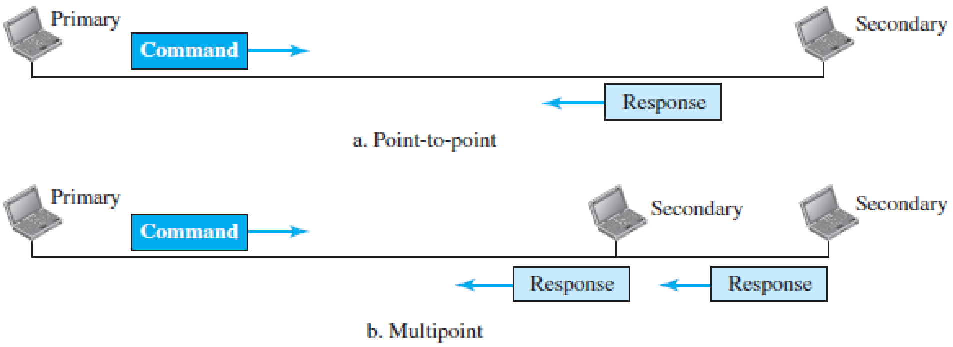


FIGURE 2.25: NORMAL RESPONSE MODE

In ABM, the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in Figure 2.26. This is the common mode today.

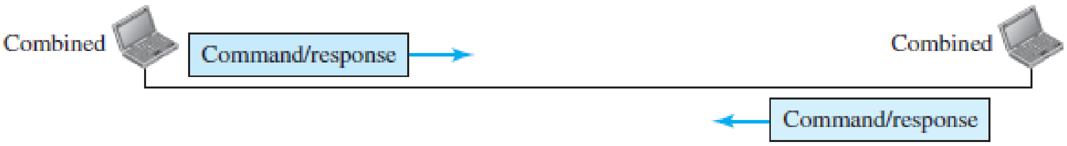


FIGURE 2.26: ASYNCHRONOUS BALANCED MODE

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

Framing:

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)*.

Each type of frame serves as an envelope for the transmission of a different type of message. I-frames are used to data-link 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. Each frame in HDLC may contain up to six fields, as shown in Figure 2.27: 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.

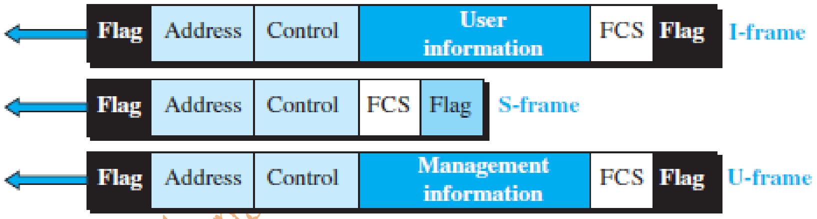


FIGURE 2.27: HDLC FRAMES

- ➢ ***Flag field.*** This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.
- ➢ ***Address field.*** This field contains the address of the secondary station. If a primary station created the frame, it contains *to* address. If a secondary station creates the frame, it contains *from* address. The address field can be one byte or several bytes long, depending on the needs of the network.
- ➢ ***Control field.*** The control field is one or two bytes used for flow and error control.
- ➢ ***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 CRC.

The control field determines the type of frame and defines its functionality. The format is specific for the type of frame, as shown in Figure 2.28.

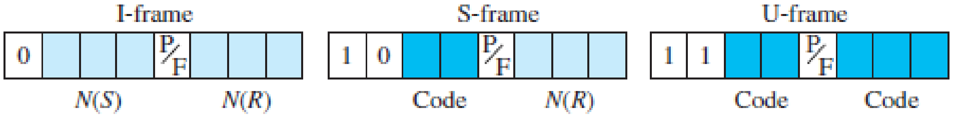


FIGURE 2.28: CONTROL FIELD FORMAT FOR THE DIFFERENT FRAME TYPES

POINT-TO-POINT PROTOCOL (PPP):

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 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.

Services:

The designers of PPP have included several services to make it suitable for a point-to-point protocol, but have ignored some traditional services to make it simple.

Services Provided by PPP:

PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data. PPP is designed to accept payloads from several network layers (not only IP).

Authentication is also provided in the protocol, but it is optional. The new version of PPP, called *Multilink PPP,* provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

Services Not Provided by PPP:

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.

Framing:

PPP uses a character-oriented (or byte-oriented) frame. Figure 2.29 shows the format of a PPP frame. The description of each field follows:

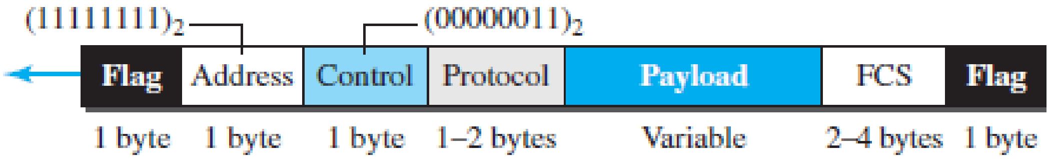


FIGURE 2.29: PPP FRAME FORMAT

- ***Flag.*** A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110.
- ***Address.*** The address field in this protocol is a constant value and set to 11111111 (broadcast address).
- ***Control.*** This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.
- ***Protocol.*** The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.
- ***Payload field.*** The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation.
  - The data field is byte-stuffed if the flag byte pattern appears in this field.
  - Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.
- ***FCS.*** The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

Byte Stuffing:

Since PPP is a byte-oriented protocol, the flag in PPP is a byte that 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 the next byte is not a flag. Obviously, the escape byte itself should be stuffed with another escape byte.

Multiplexing:

Although PPP is a link-layer protocol, it uses another set of protocols to establish the link, authenticate the parties involved, and carry the network-layer data. Three sets of protocols are defined to make PPP powerful: the Link Control Protocol (LCP), two Authentication Protocols (APs), and several Network Control Protocols (NCPs). At any moment, a PPP packet can carry data from one of these protocols in its data field.

Link Control Protocol:

The **Link Control Protocol (LCP)** is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established.

Authentication Protocols:

Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. *Authentication* means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP:

The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

a. The user who wants to access a system sends authentication identification (usually the user name) and a password.

b. The system checks the validity of the identification and password and either accepts or denies connection.

CHAP:

The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way handshaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

a. The system sends the user a challenge packet containing a challenge value, usually a few bytes.

b. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.

c. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret.

Network Control Protocols:

PPP is a multiple-network-layer protocol. It can carry a network-layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets.

IPCP:

One NCP protocol is the **Internet Protocol Control Protocol (IPCP).** This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 2.30. IPCP defines seven packets, distinguished by their code values, as shown in Table 2.4.

Other Protocols: There are other NCP protocols for other network-layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on.

Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject

TABLE 2.4: CODE VALUE FOR IPCP PACKETS

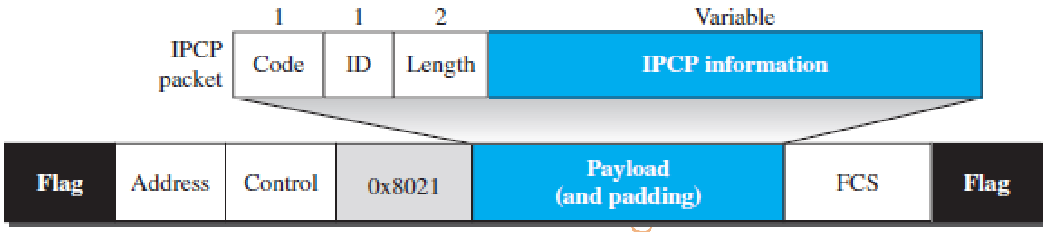


FIGURE 2.30: IPCP PACKET ENCAPSULATED IN PPP FRAME

Multilink PPP:

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 2.31.

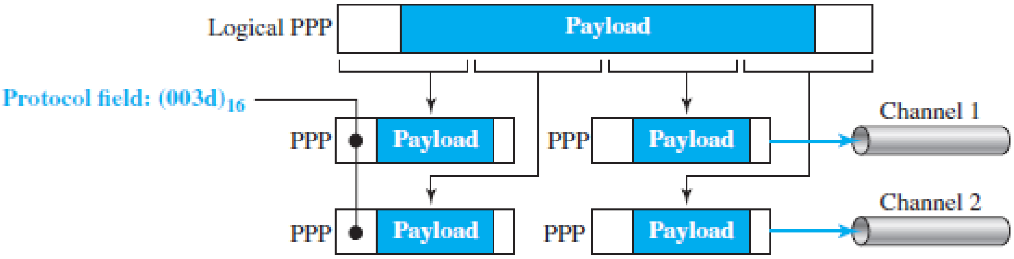


FIGURE 2.31: MULTILINK PPP

MEDIA ACCESS CONTROL (MAC)

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, do not monopolize the discussion, and so on. Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called *media access control (MAC).* We categorize them into three groups, as shown in Figure 2.32.

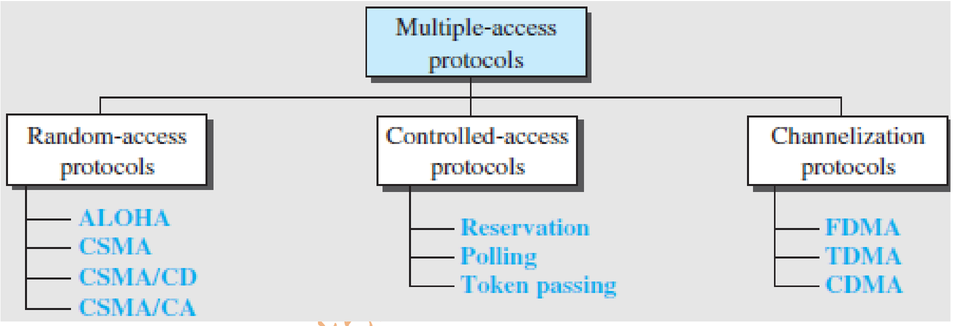


FIGURE 2.32: TAXONOMY OF MULTIPLE-ACCESS PROTOCOLS

RANDOM ACCESS:

In **random-access** or **contention** methods, no station is superior to another station and none is assigned control over another. 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 testing the state of the medium.

Two features give this method its name. First, 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*. Second, no rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

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. To avoid access conflict or to resolve it when it happens, each station follows a procedure that answers the following questions:

- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

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*). This method later evolved into two parallel methods: *carrier sense multiple access with collision detection (CSMA/CD),* which tells the station what to do when a collision is detected, and *carrier sense multiple access with collision avoidance (CSMA/CA),* which tries to avoid the collision.

ALOHA:

ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

Pure ALOHA:

The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations.

The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the *backoff time* T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}, a station must give up and try later.

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.

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.

CONTROLLED ACCESS:

In **controlled access,** the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations.

There are three controlled-access methods:

RESERVATION:

In the **reservation** method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot.

The stations that have made reservations can send their data frames after the reservation frame. Figure 2.33 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

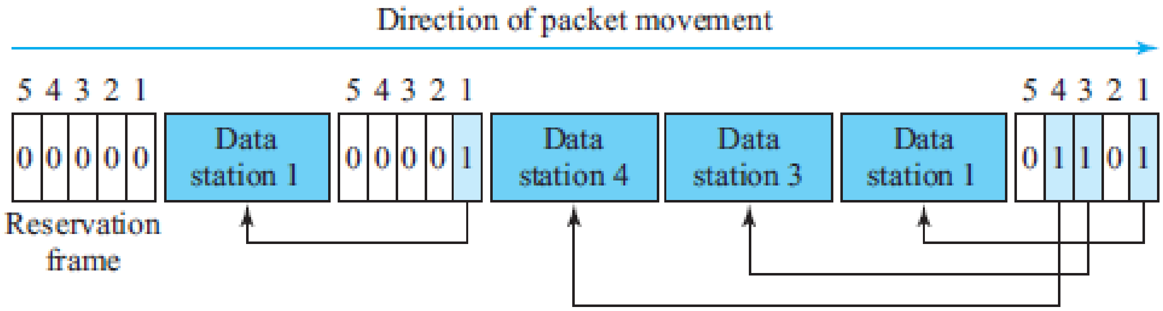


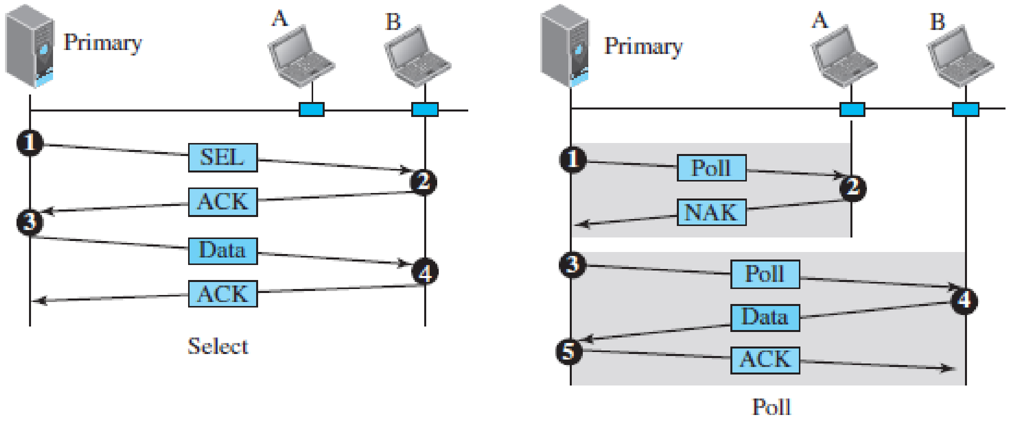
FIGURE 2.33: RESERVATION ACCESS METHOD

POLLING:

Polling works with topologies in which one device is designated as a ***primary station*** and the other devices are ***secondary stations*.** All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.

The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time.

The primary device, therefore, is always the initiator of a session (see Figure 2.34). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.



Select

Poll

FIGURE 2.34: SELECT AND POLL FUNCTIONS IN POLLING-ACCESS METHOD

Select:

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available. If it has something to send, the primary device sends it.

What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status.

Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll:

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send.

When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send.

When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

TOKEN PASSING:

In the **token-passing** method, the stations in a network are organized in a logical ring. In other words, for each station, there is a ***predecessor*** and a ***successor***.

The ***predecessor*** is the station which is logically before the station in the ring; the ***successor*** is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

But how is the right to access the channel passed from one station to another? In this method, a special packet called a ***token*** circulates through the ring. The possession (*meaning control*) of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor.

It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station.

CHANNELIZATION:

Channelization (or *channel partition,* as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations.

There are three channelization protocols: ***FDMA***, ***TDMA***, and ***CDMA***.

FDMA:

In **frequency-division multiple access (FDMA),** the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a band pass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small *guard bands.*

FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA.

We need to emphasize that although FDMA and frequency-division multiplexing (FDM) conceptually seem similar, there are differences between them.

FDM is a physical layer technique that combines the loads from low bandwidth channels and transmits them by using a high-bandwidth channel.

The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data-link layer. The datalink layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band.

There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

TDMA:

In **time-division multiple access (TDMA),** the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot.

The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area.

To compensate for the delays, we can insert *guard times*. Synchronization is normally accomplished by having some synchronization bits (normally referred to as *preamble bits*) at the beginning of each slot.

We also need to emphasize that although TDMA and time-division multiplexing (TDM) conceptually seem the same, there are differences between them.

TDM is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

TDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

CDMA:

Code-division multiple access (CDMA) was conceived (*meaning imagine/visualize*) several decades ago. Recent advances in electronic technology have finally made its implementation possible.

CDMA differs from FDMA in that only one channel occupies the entire bandwidth of the link. It differs from TDMA in that all stations can send data simultaneously; there is no timesharing.

In CDMA, one channel carries all transmissions simultaneously.

UNIT-II

INTRODUCTION TO DATALINK LAYER ERROR DETECTION & CORRECTION DATALINK CONTROL MEDIA ACCESS CONTROL CONNECTING DEVICES & VIRTUAL LANS

CONNECTING DEVICES AND VIRTUAL LAN'S

Hosts or LANs do not normally operate in isolation. They are connected to one another or to the Internet. To connect hosts or LANs, we use connecting devices. Connecting devices can operate in different layers of the Internet model.

CONNECTING DEVICES:

Hosts and networks do not normally operate in isolation. We use **connecting devices** to connect hosts together to make a network or to connect networks together to make an internet. Connecting devices can operate in different layers of the Internet model.

We discuss three kinds of *connecting devices:* ***hubs***, ***link-layer switches***, and ***routers***. Hubs today operate in the first layer of the Internet model. Link-layer switches operate in the first two layers. Routers operate in the first three layers. (See Figure 2.35)

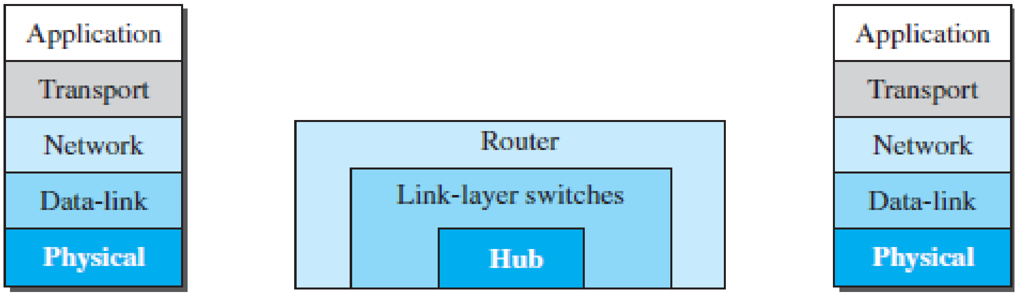


FIGURE 2.35: THREE CATEGORIES OF CONNECTING DEVICES

HUBS: A **hub** is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A **repeater** receives a signal and, before it becomes too weak or corrupted, *regenerates* and *retimes* the original bit pattern.

The repeater then sends the refreshed signal. In the past, when Ethernet LANs were using bus topology, a repeater was used to connect two segments of a LAN to overcome the length restriction of the coaxial cable. Today, however, Ethernet LANs use star topology.

In a star topology, a repeater is a multiport device, often called a *hub that* can be used to serve as the connecting point and at the same time function as a repeater. Figure 2.36 shows that when a packet from station A to station B arrives at the hub, the signal representing the frame is regenerated to remove any possible corrupting noise, but the hub forwards the packet from all outgoing ports except the one from which the signal was received.

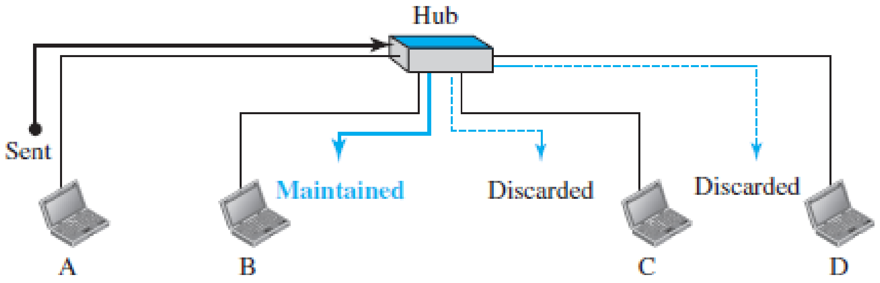


FIGURE 2.36: A HUB

In other words, the frame is broadcast. All stations in the LAN receive the frame, but only station B keeps it. The rest of the stations discard it. Figure 2.36 shows the role of a repeater or a hub in a switched LAN.

A repeater has no filtering capability.

A hub or a repeater is a physical-layer device. They do not have a link-layer address and they do not check the link-layer address of the received frame. They just regenerate the corrupted bits and send them out from every port.

LINK-LAYER SWITCHES: A **link-layer switch** (or ***switch***) operates in both the physical and the data-link layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the link-layer switch can check the MAC addresses (source and destination) contained in the frame.

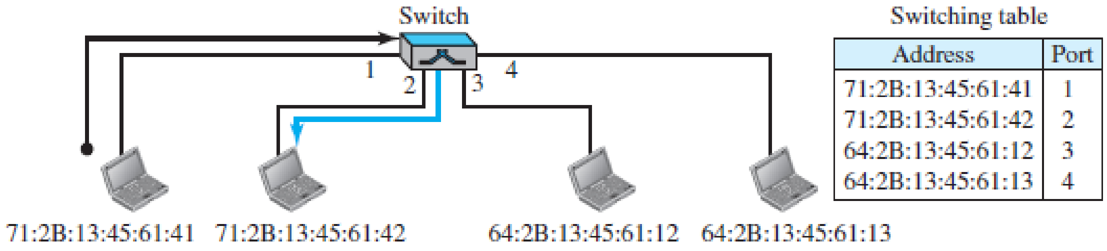
Filtering:

One may ask what the difference in functionality is between a link-layer switch and a hub. A link-layer switch has **filtering** capability. It can check the destination address of a frame and can decide from which outgoing port the frame should be sent.

A link-layer switch has a table used in filtering decisions.

Let us give an example. In Figure 2.37, we have a LAN with four stations that are connected to a link-layer switch. If a frame destined for station 71:2B:13:45:61:42 arrives at port 1, the link-layer switch consults its table to find the departing port.

According to its table, frames for 71:2B:13:45:61:42 should be sent out only through port 2; therefore, there is no need for forwarding the frame through other ports.



Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

FIGURE 2.37: LINK-LAYER SWITCH

A Link-Layer Switch does not change the Link-Layer (MAC) addresses in a frame.

Transparent Switches:

A **transparent switch** is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet three criteria:

- Frames must be forwarded from one station to another.
- The forwarding table is automatically made by learning frame movements in the network.
- Loops in the system must be prevented.

Advantages of Switches:

A link-layer switch has several advantages over a hub. We discuss only two of them here.

Collision Elimination:

This means increasing the average bandwidth available to a host in the network. In a switched LAN, there is no need for carrier sensing and collision detection; each host can transmit at any time.

Connecting Heterogeneous Devices:

A link-layer switch can connect devices that use different protocols at the physical layer (data rates) and different transmission media.

As long as the format of the frame at the data-link layer does not change, a switch can receive a frame from a device that uses twisted-pair cable and sends data at 10 Mbps and deliver the frame to another device that uses fiber-optic cable and can receive data at 100 Mbps.

ROUTERS:

A **router** is a three-layer device; it operates in the physical, data-link, and network layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network-layer device, a router checks the network-layer addresses.

A Router is a three-layer (Physical, Data-Link, and Network) device.

A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet.

There are three major differences between a router and a repeater or a switch.

1. A router has a physical and logical (IP) address for each of its interfaces.

2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.

3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Let us give an example. In Figure 2.38, assume an organization has two separate buildings with a Gigabit Ethernet LAN installed in each building. The organization uses switches in each LAN.

The two LANs can be connected to form a larger LAN using 10 Gigabit Ethernet technologies that speeds up the connection to the Ethernet and the connection to the organization server. A router then can connect the whole system to the Internet.

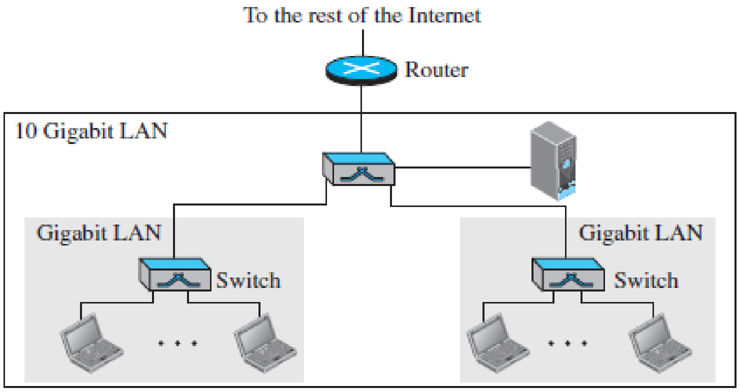


FIGURE 2.38: ROUTING EXAMPLE

A ROUTER CHANGES THE LINK-LAYER ADDRESSES IN A PACKET.

VIRTUAL LANS:

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a **virtual local area network (VLAN)** as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 2.39 shows a switched LAN in an engineering firm in which nine stations are grouped into three LANs that are connected by a switch.

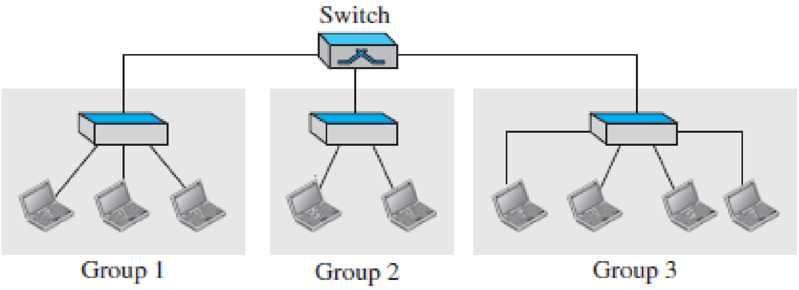


FIGURE 2.39: A SWITCH CONNECTING THREE LANS

The first three engineers work together as the first group, the next two engineers work together as the second group, and the last four engineers work together as the third group. The LAN is configured to allow this arrangement.

But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group?

The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

Figure 2.40 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs, called *VLANs.*

Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN.

All members belonging to a VLAN can receive broadcast messages sent to that particular VLAN. This means that if a station moves from VLAN 1 to VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

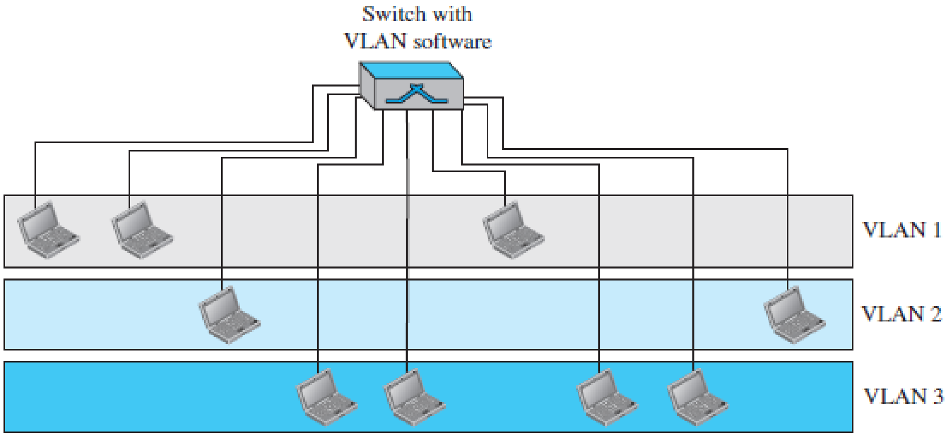


FIGURE 2.40: A SWITCH USING VLAN SOFTWARE

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 2.41 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

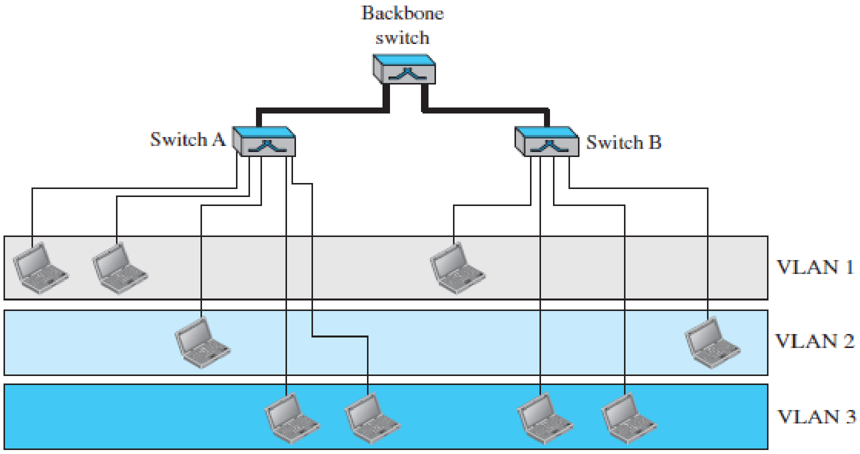


FIGURE 2.41: TWO SWITCHES IN A BACKBONE USING VLAN SOFTWARE

MEMBERSHIP:

Vendors use different characteristics such as ***interface numbers***, ***port numbers***, ***MAC addresses***, ***IP addresses***, ***IP multicast addresses***, or a combination of two or more of these to group stations in VLAN.

Interface Numbers:

Some VLAN vendor's uses switch interface numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1, stations connecting to ports 4, 10, and 12 belong to VLAN 2, and so on.

MAC Addresses:

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E2:13:42:A1:23:34 and F2:A1:23:BC:D3:41 belong to VLAN 1.

IP Addresses:

Some VLAN vendors use the 32-bit IP address as a membership characteristic. For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

Multicast IP Addresses:

Some VLAN vendors use the multicast IP address as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the datalink layer.

Combination:

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

ADVANTAGES:

There are several advantages to using VLANs.

Cost and Time Reduction:

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

Creating Virtual Work Groups:

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

Security:

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.