COMPUTER NETWORKS– V SEM CSE VTU

MODULE-2: Data Link Layer

1. **Error Detection and Correction**
   - **1.1. Introduction**
   - **1.2. Block Coding**
   - **1.3. Cyclic Codes**
2. **Data link control**
   - **2.1. DLC Services**
   - **2.2. Data link layer protocols**
   - **2.3. High Level Data Link Control**
3. **Media Access Control**
   - **3.1. Random Access**
   - **3.2. Controlled Access**

1. Error Detection and Correction

1.1 Introduction

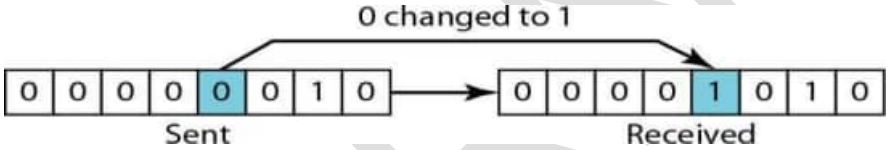
For many applications system must guarantee that the data received are same to the data transmitted. During transmission data may be corrupted because of many factors. Hence there should be some mechanism to detect and correct errors.

Types of Errors

There are two types of error: Single bit error and Burst error.

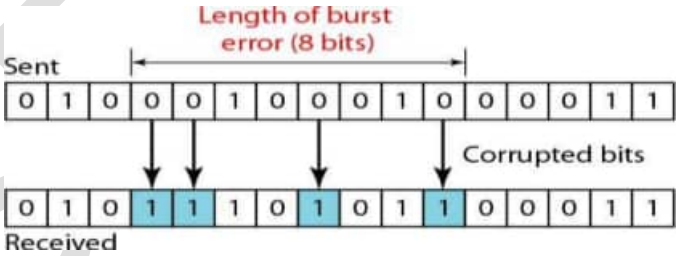
Single-Bit Error

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. Single-bit errors are the least likely type of error in serial data transmission.



Burst Error

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. In the below figure, 0100010001000011 was sent, but 0101110101100011 was received.



Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

Redundancy

To detect or correct errors some extra bits are sent with 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 looking only to see if any error has occurred.
- 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 the errors and the size of the message are important factors.
- If we need to correct one 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 possibilities.

Forward Error Correction versus Retransmission

There are two main methods of error correction.

- **Forward error correction** is the process in which the receiver tries to guess the message by using redundant bits. This is possible, if the number of errors is small.
- **Correction by retransmission** is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free.

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 or correct the errors.

- Coding schemes can be divided into two broad categories: block coding and convolution coding.

Modular Arithmetic

- In modular arithmetic only integers in the range 0 to N-1 is used. This is known as modulo-N arithmetic. For example, if the modulus is 12, we use only the integers 0 to 11, inclusive.

Modulo-2 Arithmetic

In this arithmetic, the modulus *N* is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.

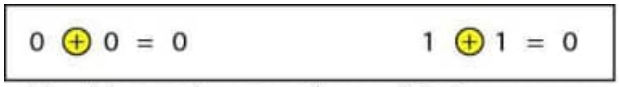
Adding:

0+0=0 0+1=1 1+0=1 1+1=0

Subtracting:

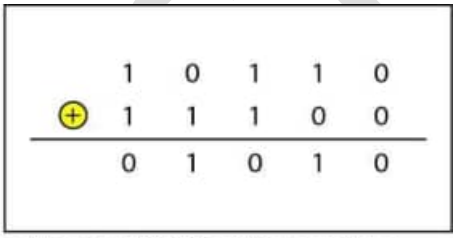
0-0=0 0-1=1 1-0=1 1-1=0

In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is I if two bits are different.



a. Two bits are the same, the result is 0.

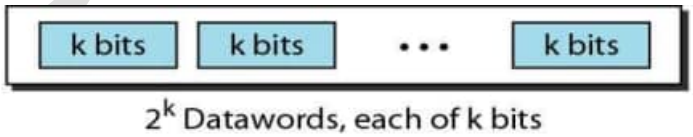
b. Two bits are different, the result is 1.



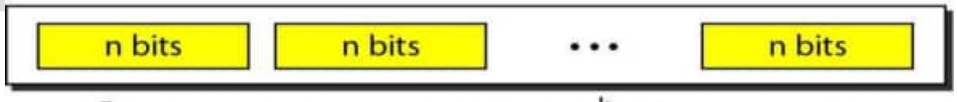
c. Result of XORing two patterns

1.2Block Coding

- In block coding message is divided into k bits blocks called **datawords**. Then *r* redundant bits are added to each block to make the length $n = k + r$. The resulting *n-bit* blocks are called **codewords**.
- With k bits, we can create a combination of 2k datawords; with n bits, we can create a combination of 2n codewords.
- Since n > k, the number of possible codewords is larger than the number of possible datawords.
- The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have 2n - 2k codewords that are not used. We call these codewords invalid or illegal.



2^k Datawords, each of k bits



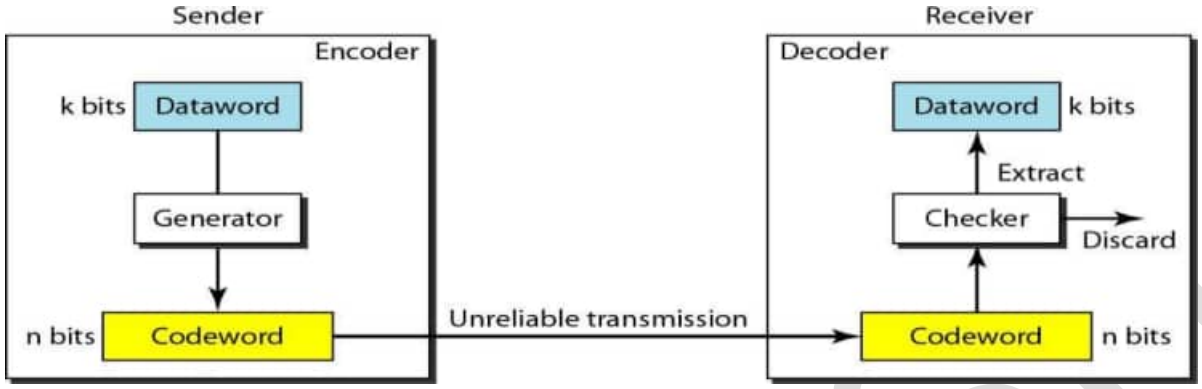
2^n Codewords, each of n bits (only 2^k of them are valid)

Error Detection

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

codeword.

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



Process of error detection in block coding

- The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding.
- 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.
- This type of coding can detect only single errors. Two or more errors may remain undetected.

Example:

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

Dataword	Codeword
00	000
01	011
10	101
11	110

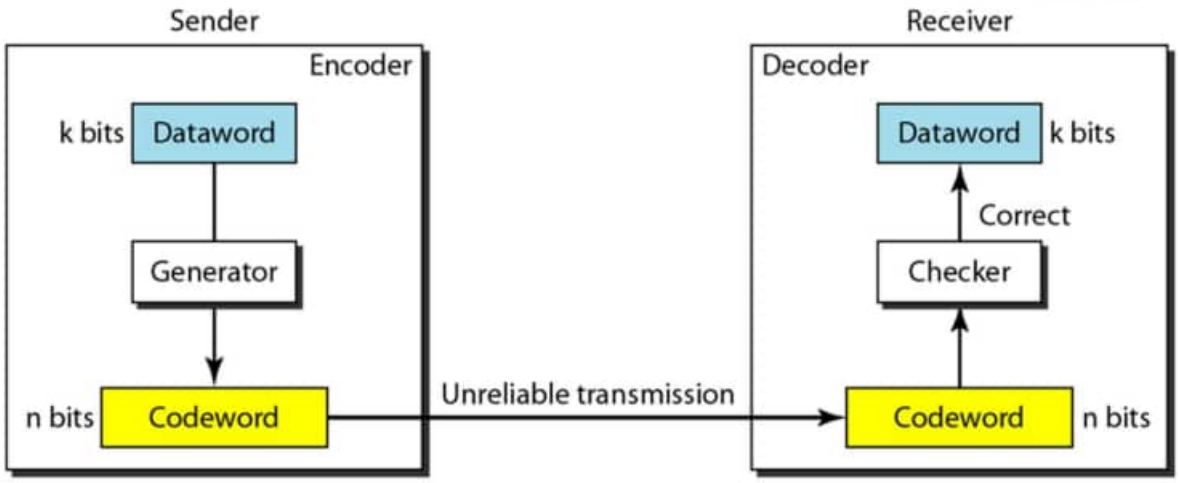
Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

- The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
- The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted).

This is not a valid codeword and is discarded.
The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

Error Correction

In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent.



Below Table shows the datawords and codewords.

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

Assume the dataword is 01. The sender consults the table (or uses an algorithm) to create the codeword 01011.

The codeword is corrupted during transmission, and 01001 is received (error in the second bit from the right).

1. First, the receiver finds that the received codeword is not in the table. This means an error has occurred. (Detection must come before correction.)
2. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.
3. Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.
4. By the same reasoning, the original codeword cannot be the third or fourth one in the table.

The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.

Hamming Distance

The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. Hamming distance between two words x and y is represented as d(x, y).The Hamming distance can be found by applying the XOR operation on the two words and counting the number of 1s in the result.

Example:

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: The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words. It is represented as dmin.

Find the minimum Hamming distance of the coding scheme in below table:

Dataword	Codeword
00	000
01	011
10	101
11	110

Solution

d(000,011) = 2, d(000,101) = 2, d(000,110) = 2, d(011,101) = 2, d(011,110) = 2,

d(101,110) = 2

The $\mathbf{d_{min}}$ in this case is 2.

Example 2

Find the minimum Hamming distance of the coding scheme in below table:

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

Solution

d(00000,01011) = 3, d(00000,10101) = 3, d(00000,11110) = 4 d(01011,10101) = 4, d(01011,11110) = 3, d(10101,11110) = 2

The $\mathbf{d_{min}}$ in this case is 3.

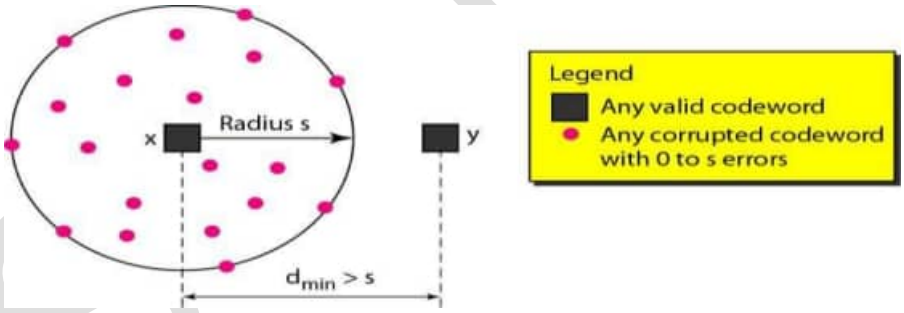
Coding scheme needs to have at least three parameters: the codeword size *n,* the dataword size *k,* and the minimum Hamming distance *dmin.* A coding scheme C is written as *C(n, k)* with a separate expression for *dmin.*
Ex: *C(5,* 2) with dmin = 3.

Hamming Distance and Error

- When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error.
- 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.

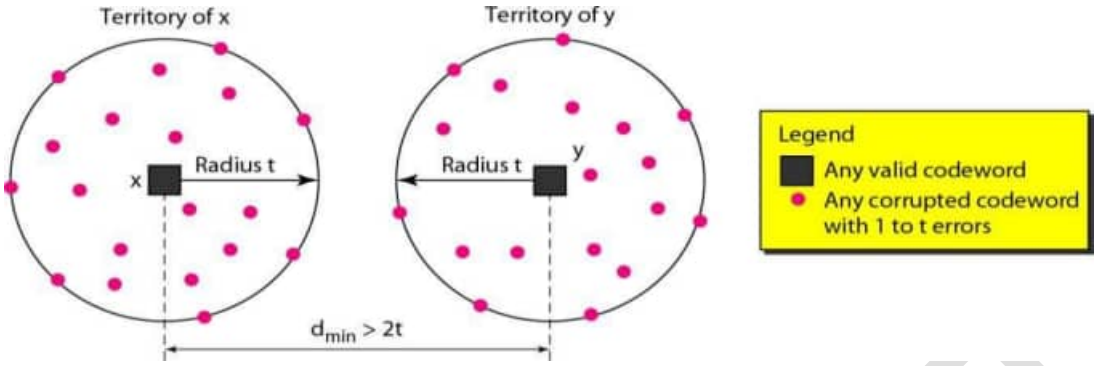
Minimum Distance for Error Detection

- If S errors occur during transmission, the Hamming distance between the sent codeword and received codeword is S.
- To guarantee the detection of up to S errors in all cases, the minimum Hamming distance in a block code must be dmin =*S* + 1.
- Let us assume that the sent codeword *x* is at the center of a circle with radius *S.* All other received codewords that are created by 1 to *S* errors are points inside the circle or on the perimeter of the circle. All other valid codewords must be outside the circle.



Minimum Distance for Error Correction

- When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword. Each valid codeword has its own territory.
- We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius of *t* and that the valid codeword is at the center.
- For example, suppose a codeword *x* is corrupted by *t* bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the center.
- To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be dmin == 2t + 1.



Linear Block Codes

Linear block code is a code in which the exclusive OR of two valid codewords creates another valid codeword.

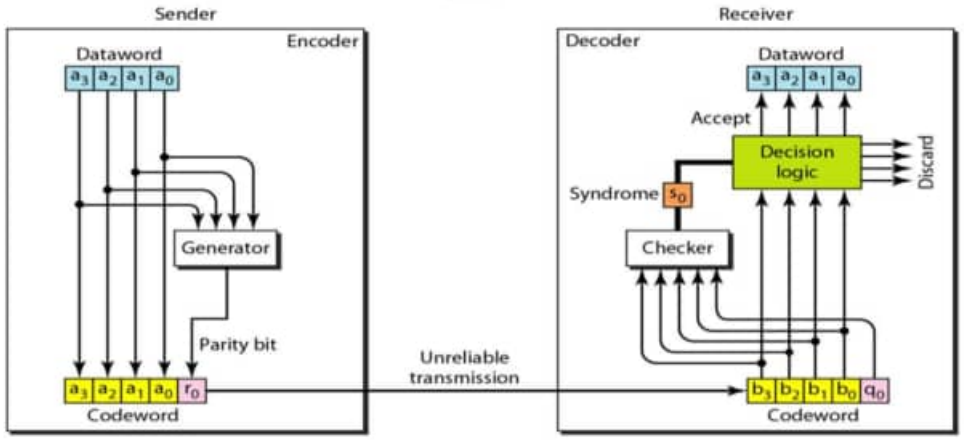
Minimum Distance for Linear Block Codes: The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Some Linear Block Codes

1) Simple Parity-Check 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.

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with dmin =2.



- The encoder uses a generator that takes a copy of a 4-bit dataword (a_0, a_1, a_2, and a_3) and generates a parity bit r_0.
- The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even.
  Example: Simple parity-check code C(5, 4)

Datawords	Codewords	Datawords	Codewords
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

- This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

  $r_0=a_3+a_2+a_1+a_0$ (modulo – 2)
- If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even.
- The sender sends the codeword which may be corrupted during transmission.
- The receiver receives a 5-bit word.
- The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits.
- The result, which is called the syndrome, is just 1 bit. The syndrome is 0 when the number of 1s in the received codeword is even; otherwise, it is 1.

  s0=b3+b2+b1+b0+q0 (modulo – 2)
- The syndrome is passed to the decision logic analyzer.
- If the syndrome is 0, there is no error in the received codeword; the data portion of the received codeword is accepted as the dataword.
- If the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver.

1. No error occurs; the received codeword is 10111. The syndrome is O. The dataword 1011 is created.
2. One single-bit error changes *a1* The received codeword is 10011. The syndrome is 1. No dataword is created.
3. One single-bit error changes *r0* The received codeword is 10110. The syndrome is 1. No dataword is created. Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.
4. An error changes *ro* and a second error changes *a3* The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value. The simple parity-check decoder cannot detect an

even number of errors. The errors cancel each other out and give the syndrome a value of 0.

5. Three bits-a_3, a_2, and a_1 are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Limitation: A simple parity-check code can detect an odd number of errors.

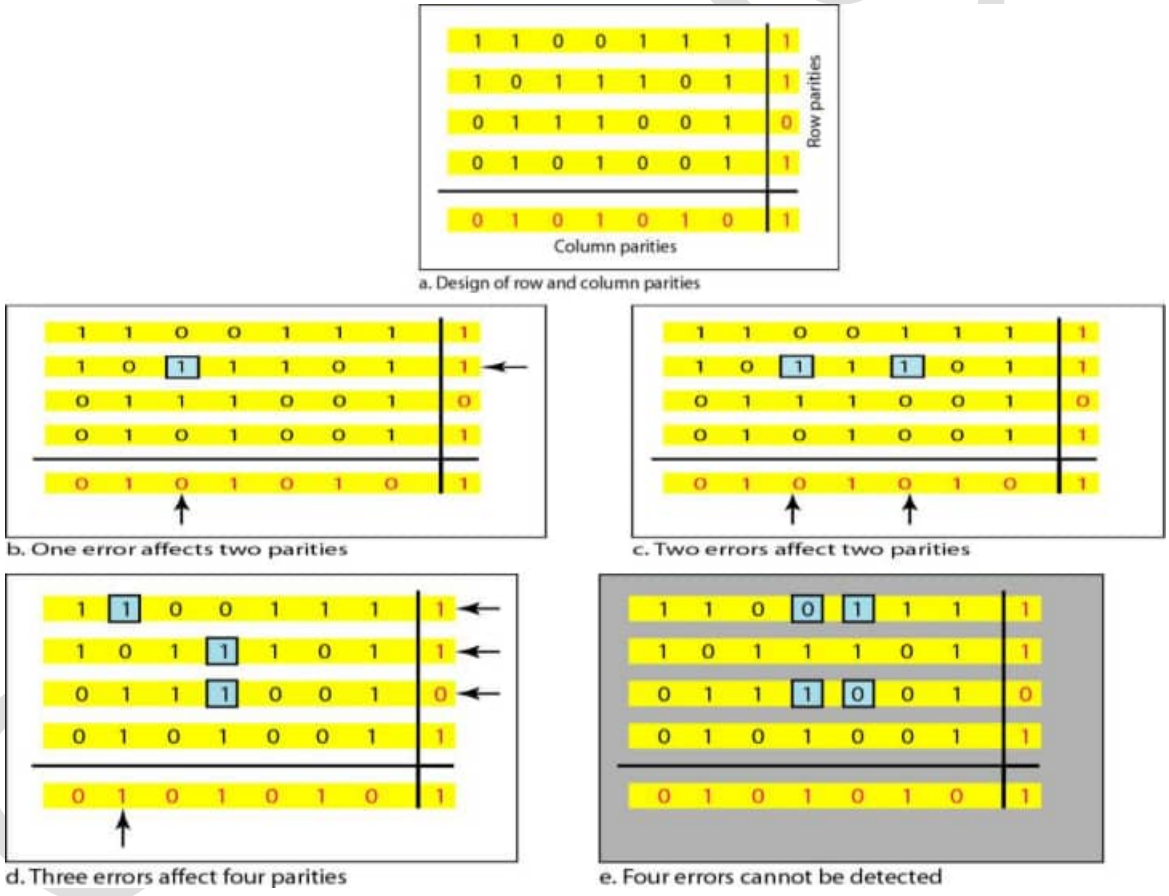
a. A better approach is the two-dimensional parity check. In this method, the dataword is organized in a table.

b. The data to be sent, five 7-bit bytes, are put in separate rows.

c. For each row and each column, 1 parity-check bit is calculated.

d. The whole table is then sent to the receiver, which finds the syndrome for each row and each column.

e. The two-dimensional parity check can detect up to three errors that occur anywhere in the table. However, errors affecting 4 bits may not be detected.



a. Design of row and column parities

b. One error affects two parities

c. Two errors affect two parities

d. Three errors affect four parities

e. Four errors cannot be detected

Hamming Codes

These codes were originally designed with *dmin* = 3, which means that they can detect up to two errors or correct one single error.

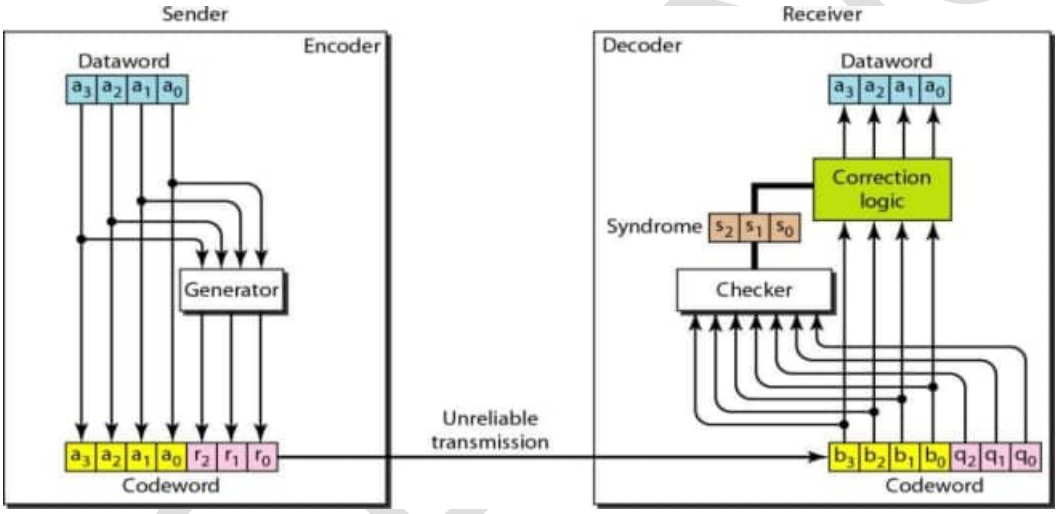
In hamming code we need to choose an integer m, say *m* >= 3. The values of n and *k* are then

calculated from m as $n = 2m - 1$ and $k = n - m$. The number of check bits $r = m$. Eg: if m = 3, n=7, k = 4

Hamming code C(7, 4) - n=7, k = 4:

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Below figure shows the structure of the encoder and decoder:



A copy of a 4-bit dataword is fed into the generator that creates three parity checks.

$r_0 = a_2 + a_1 + a_0$ modulo-2

$r_1 = a_3 + a_2 + a_1$ modulo-2

$r_2 = a_1 + a_0 + a_3$ modulo-2

The checker in the decoder creates a 3-bit syndrome (s2s1s0) in which each bit is the parity check for 4 out of the 7 bits in the received codeword:

$s_0 = b_2 + b_1 + b_0$ modulo-2

$s_1 = b_3 + b_2 + b_1$ modulo-2

$s_2 = b_1 + b_0 + b_3$ modulo-2

The 3-bit syndrome creates eight different bit patterns (000 to 111) that can represent eight different conditions. These conditions define a lack of error or an error in 1 of the 7 bits of the received codeword.

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

For example, if q_0 is in error, S_0 is the only bit affected; the syndrome, therefore, is 001. If b_2 is in error, S_0 and S_1 are the bits affected; the syndrome therefore is 011. Similarly, if b_1 is in error, all 3 syndrome bits are affected and the syndrome is 111.

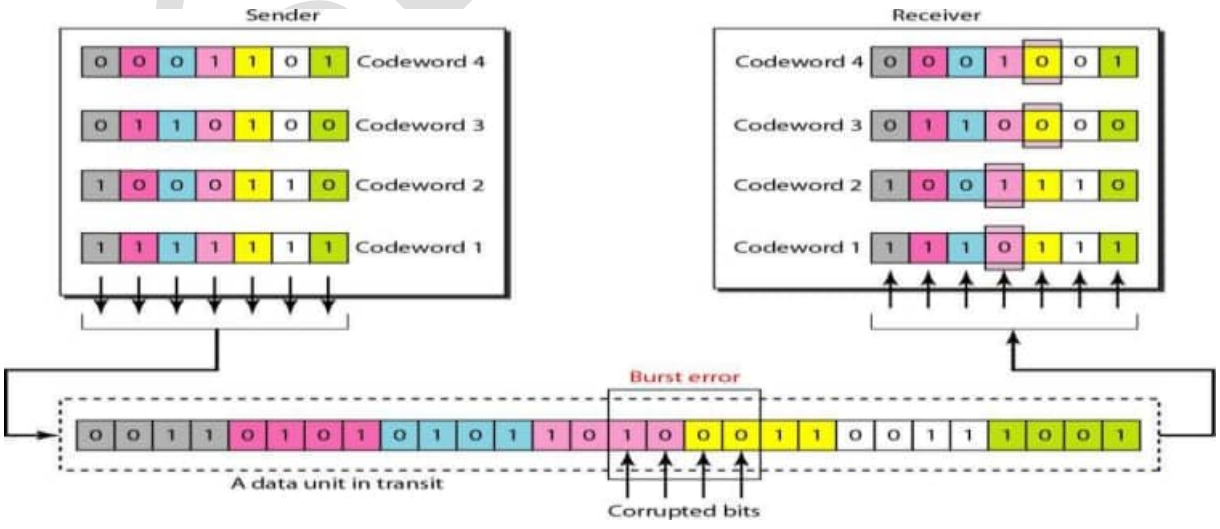
Example:

1. The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000 (no error), the final dataword is 0100.
2. The dataword 0111 becomes the codeword 0111001. The codeword 0011001 is received. The syndrome is 011. Therefore b_2 is in error. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.
3. The dataword 1101 becomes the codeword 1101000. The codeword 0001000 is received (two errors). The syndrome is 101, which means that b_0 is in error. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.

Performance

A Hamming code can only correct a single error or detect a double error. However, there is a way to make it detect a burst error.

The key is to split a burst error between several codewords, one error for each codeword.



To make the Hamming code respond to a burst error of size *N,* we need to make *N* codewords out of our frame. Then, instead of sending one codeword at a time, we arrange the codewords in a table and send the bits in the table a column at a time.

In the above Figure, the bits are sent column by column (from the left). In each column, the bits are sent from the bottom to the top. In this way, a frame is made out of the four codewords and sent to the receiver. It is shown in the figure that when a burst error of size 4 corrupts the frame, only 1 bit from each codeword is corrupted. The corrupted bit in each codeword can then easily be corrected at the receiver.

1.3 Cyclic Codes

Cyclic codes are special linear block codes in which, if a codeword is cyclically shifted (rotated), the result is another codeword.

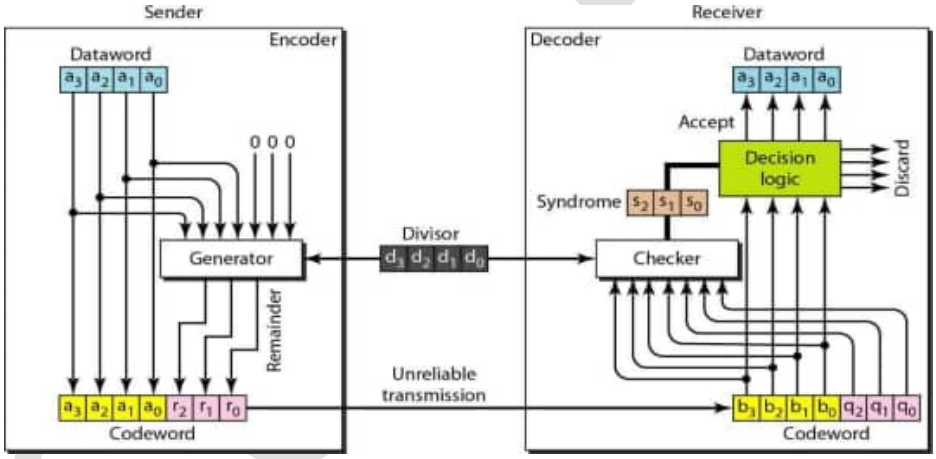
For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

In this case, if we call the bits in the first word *a0* to *a6* and the bits in the second word *b0* to *b6,*

we can shift the bits by using the following:

b1 = a0 b2 = a1 b3 = a2 b4 = a3 b5 = a4 b6 = a5 b0 = a6

Cyclic Redundancy Check



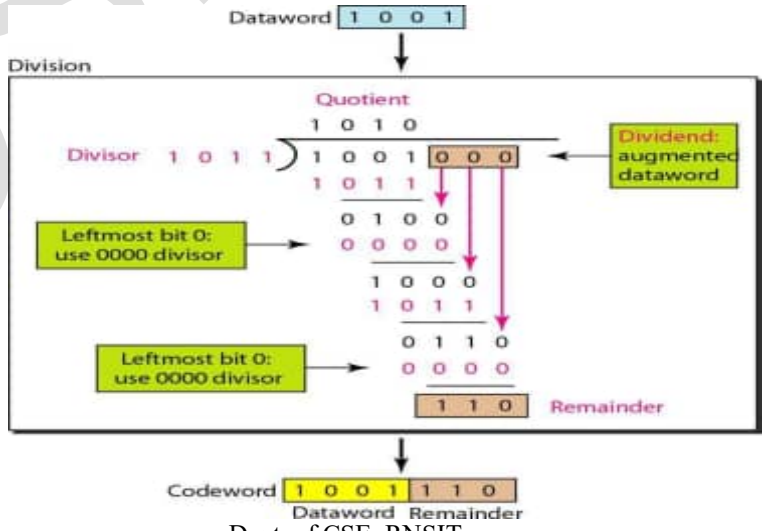
Below Table shows an example of a CRC code.

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

- 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 is appended to the dataword to create the codeword.
- The decoder receives the possibly corrupted codeword. 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 as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

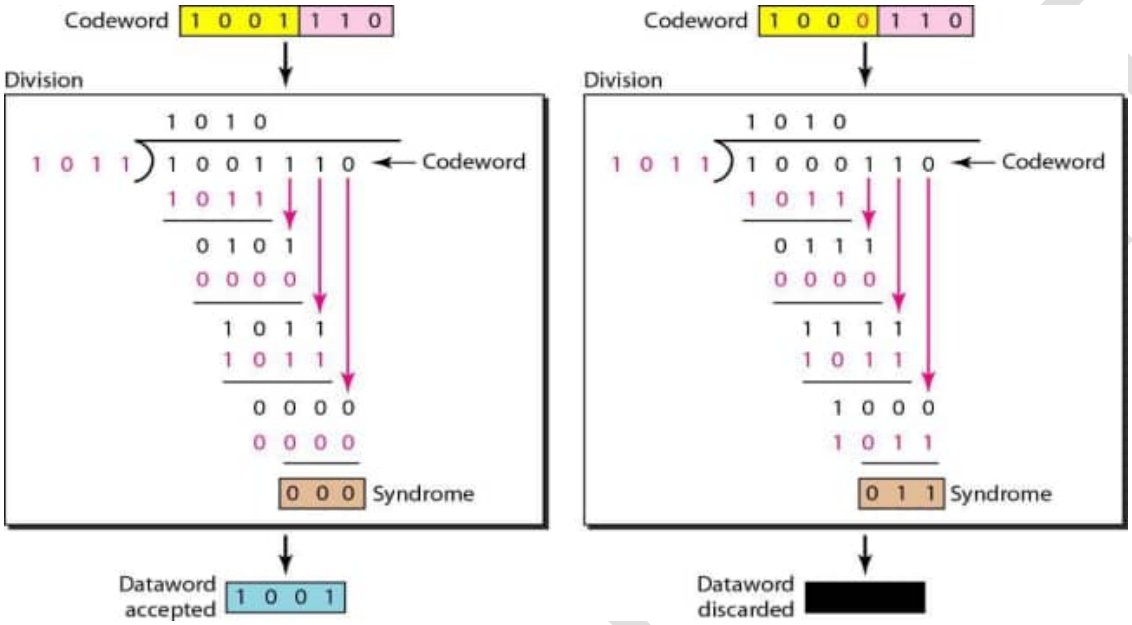
The encoder takes the dataword and augments it with n - k number of 0s. It then divides the augmented dataword by the divisor.



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; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded.

The left side figure shows the value of syndrome when no error has occurred; the syndrome is 0. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all 0s



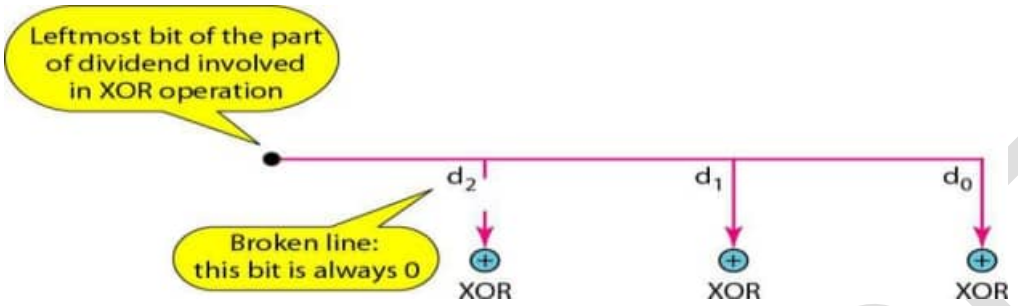
Hardware Implementation

One of the advantages of a cyclic code is that the encoder and decoder can easily and cheaply be implemented in hardware by using a handful of electronic devices. Also, a hardware implementation increases the rate of check bit and syndrome bit calculation.

Divisor:

1. The divisor is repeatedly XORed with part of the dividend.
2. The divisor has $n - k + 1$ bits which either are predefined or are all Os. In other words, the bits do not change from one dataword to another. In previous example, the divisor bits were either 1011 or 0000. The choice was based on the leftmost bit of the part of the augmented data bits that are active in the XOR operation.

3. A close look shows that only *n* - *k* bits of the divisor is needed in the XOR operation. The leftmost bit is not needed because the result of the operation is always 0, no matter what the value of this bit. The reason is that the inputs to this XOR operation are either both 0s or both 1s.



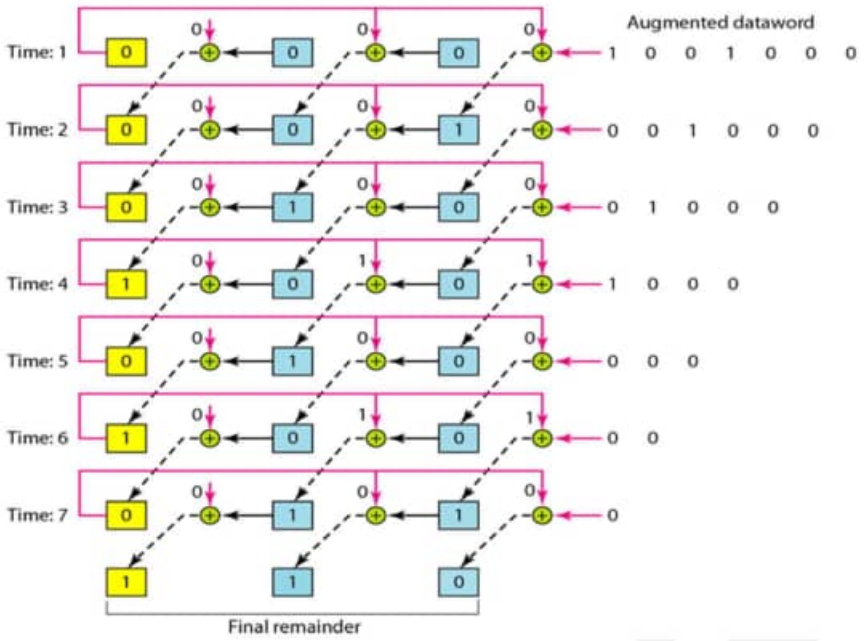
Steps:

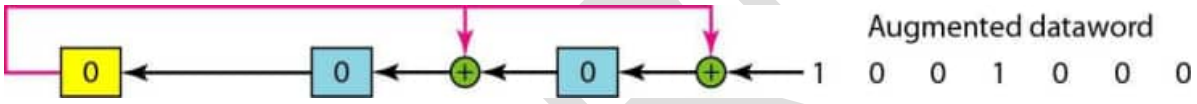
1. Assume that the remainder is originally all Os (000 in our example).
2. At each time click (arrival of 1 bit from an augmented dataword), repeat the following two actions:
   a. Use the leftmost bit to make a decision about the divisor (011 or 000).
   b. The other 2 bits of the remainder and the next bit from the augmented dataword (total of 3 bits) are XORed with the 3-bit divisor to create the next remainder.

   Below Figure shows this simulator, but note that this is not the final design; there will be more improvements.

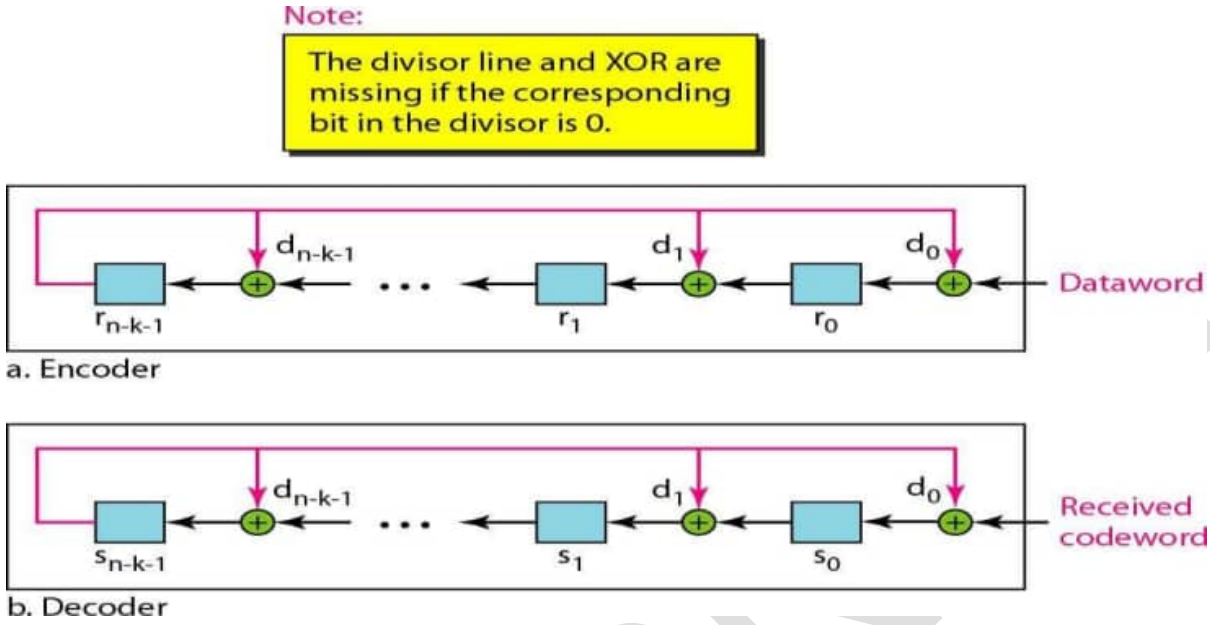
At each clock tick, shown as different times, one of the bits from the augmented dataword is used in the XOR process.

The above design is for demonstration purposes only. It needs simplification to be practical. First, we do not need to keep the intermediate values of the remainder bits; we need only the final bits. We therefore need only 3 registers instead of 24. After the XOR operations, we do not need the bit values of the previous remainder. Also, we do not need 21 XOR devices; two are enough because the output of an XOR operation in which one of the bits is 0 is simply the value of the other bit. This other bit can be used as the output. With these two modifications, the design becomes tremendously simpler and less expensive, as shown below


Augmented dataword
Time: 1
Time: 2
Time: 3
Time: 4
Time: 5
Time: 6
Time: 7
Final remainder

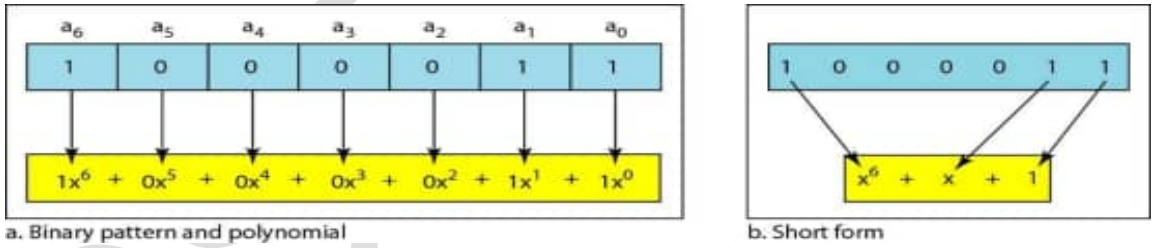

Augmented dataword
1 0 0 1 0 0 0

General Design



Polynomials

A pattern of 0s and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Figure shows a binary pattern and its polynomial representation.

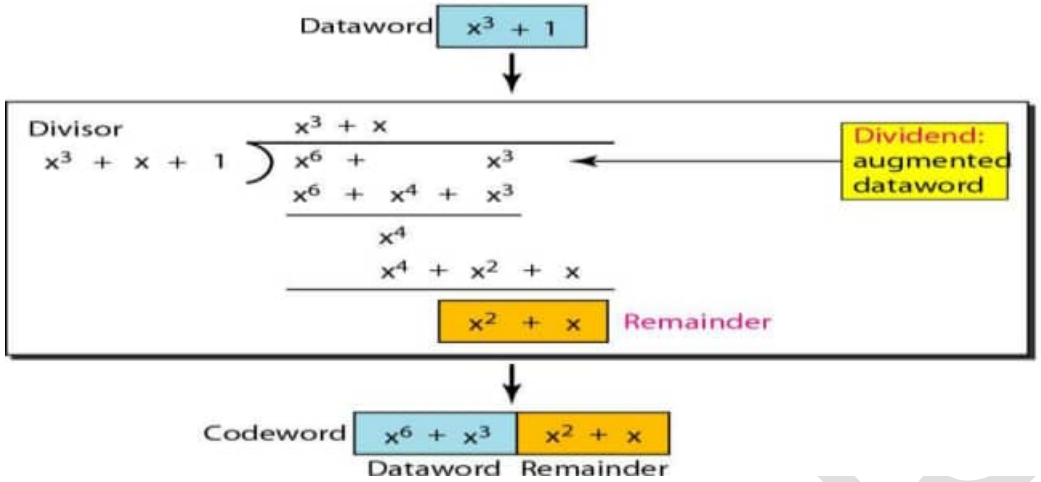


Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x6 + x + 1$ is 6. Note that the degree of a polynomial is 1 less that the number of bits in the pattern. The bit pattern in this case has 7 bits.

Shifting left 3 bits: 10011 becomes 10011000 $x^4 + x + 1$ becomes $x^7 + x^4 + x^3$

Shifting right 3 bits: 10011 becomes 10 $x^4 + x + 1$ becomes x



Cyclic Code Analysis

Following notations can be used in the cyclic codes:

Dataword: d(x) Error: e(x) Syndrome: s(x) Generator: g(x) Codeword: c(x)

In a cyclic code,

1. If s(x) != 0, one or more bits is corrupted.
2. If s(x) = 0, either
   a. No bit is corrupted. or
   b. Some bits are corrupted, but the decoder failed to detect them.

The received codeword is the sum of the sent codeword and the error. Received codeword =*c(x) + e(x)*

The receiver divides the received codeword by g (x) to get the syndrome.

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

The Right hand side of above equation is called as syndrome. If Syndrome does not have a remainder (syndrome =0), either *e(x)* is 0 or *e(x)* is divisible by *g(x).* In a cyclic code, those *e(x)* errors that are divisible by *g(x)* are not caught.

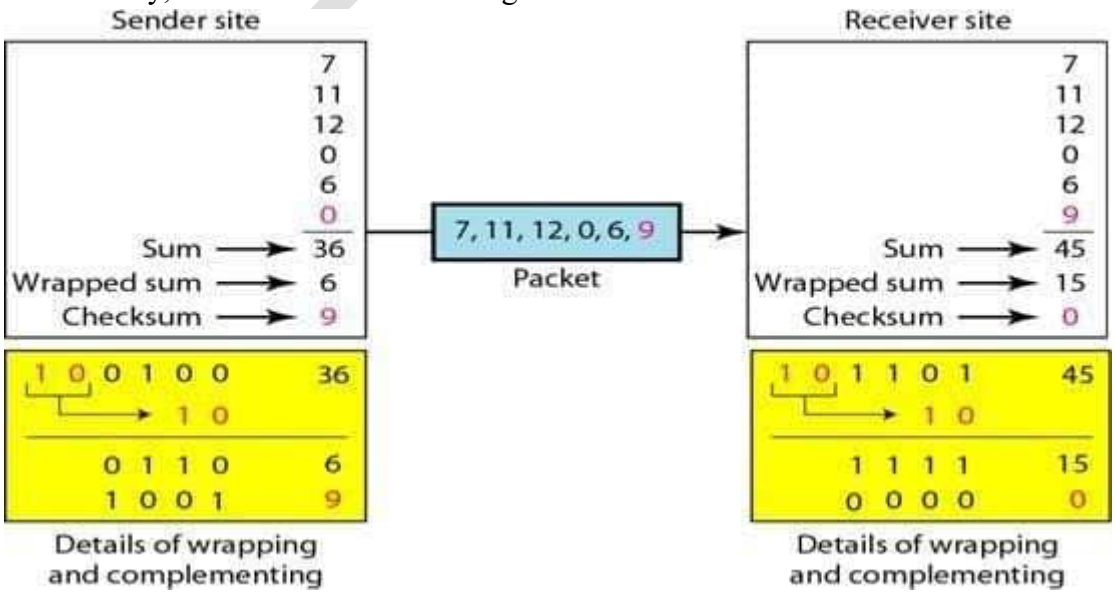
Checksum

The checksum is used in the Internet by several protocols. The checksum is based on the concept of redundancy.

Below Figure shows the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sum is then complemented, resulting in the checksum value 9 (15 - 6 = 9). The sender now sends six data items to the receiver including the checksum 9. The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Internet Checksum

Traditionally, the Internet has been using a 16-bit checksum.



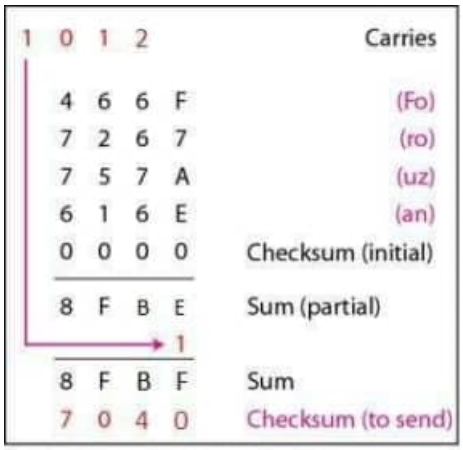
Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

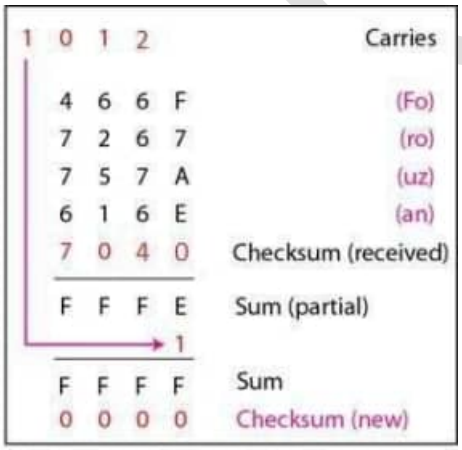
Receiver site:

1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Example:



a. Checksum at the sender site



a. Checksum at the receiver site

2.1 DLC SERVICES

- The data link control (DLC) deals with procedures for communication between two adjacent nodes i.e., node-to-node communication.
- Data link control functions include 1) Framing and 2) Flow control and 3) 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, or from other messages to other destinations, 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.
- Frames can be of fixed or variable size.

1) Fixed-Size Framing

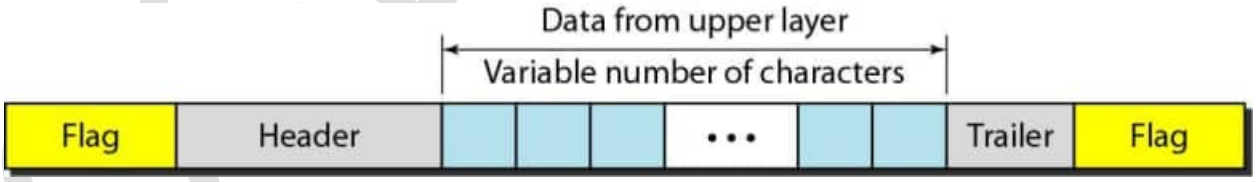
- In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter.

2) Variable-Size Framing

- In variable-size framing, we need a way to define the end of the 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 Protocols

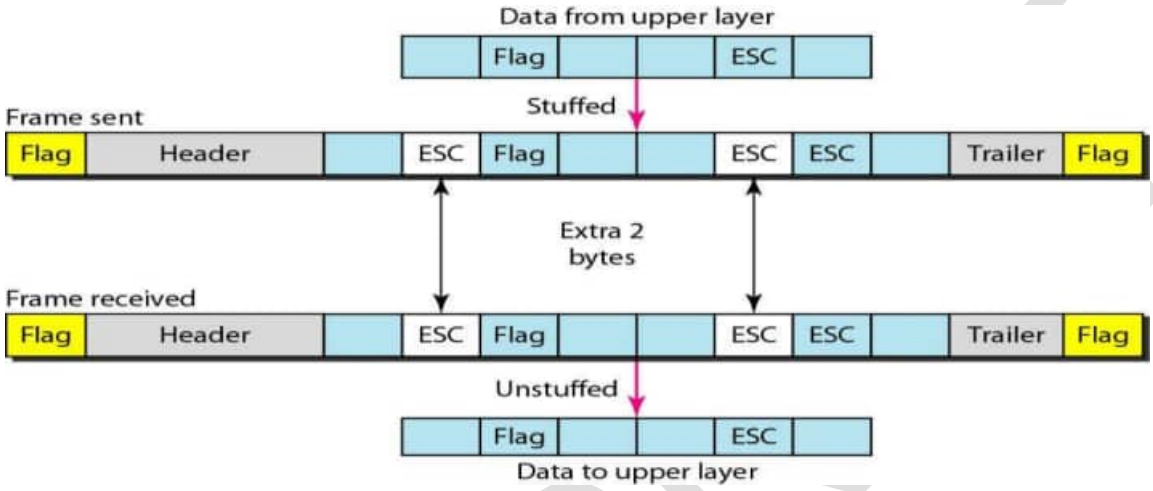
- In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII.
- The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are 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.



- Flag should be different from the text pattern used in the data section. Otherwise, when the receiver encounters this pattern in the middle of the data, it thinks that it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to character-oriented framing.

- **Byte stuffing** is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

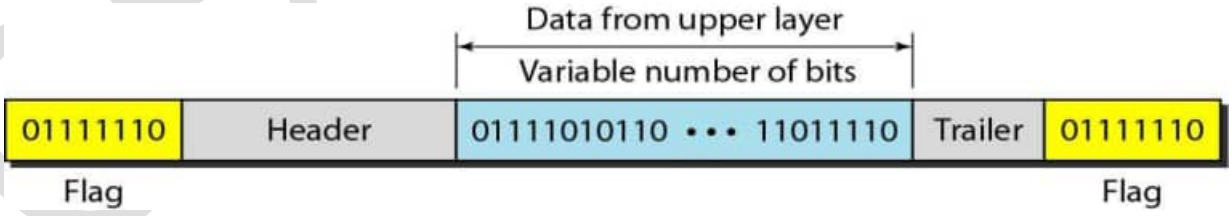
- The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.
- If even the escape character is also part of the text, an extra escape character is added to show that the second one is part of the text.



- Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters.

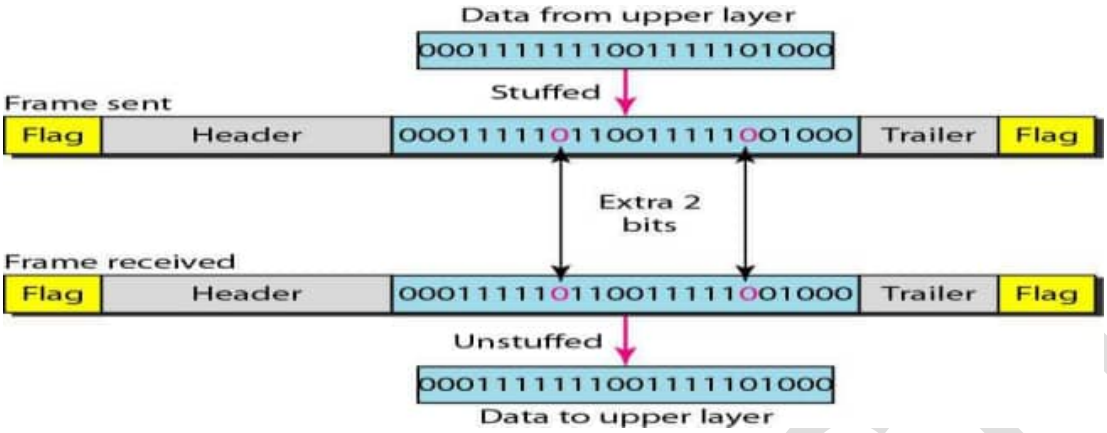
Bit-Oriented Protocols:

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.
- In addition to headers a special 8-bit pattern flag 01111110 is used as the delimiter to define the beginning and the end of the frame.



- If the flag pattern appears in the data, **bit stuffing** is used to differentiate the flag from information.
- 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.

- This extra stuffed bit is eventually removed from the data by the receiver.



Flow and Error Control

Flow Control

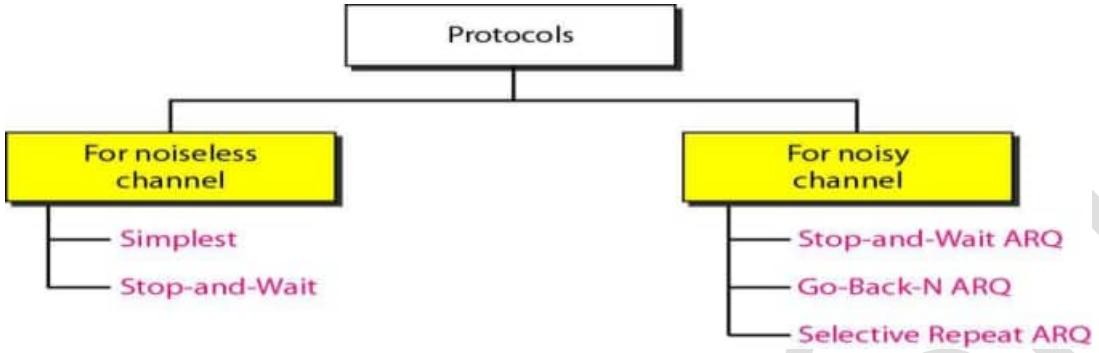
- Flow control coordinates the amount of data that can be sent before receiving an acknowledgment.
- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.
- The flow of data must not be allowed to overwhelm the receiver.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

Error Control

- Error control is both error detection and error correction.
- It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender.
- In the data link layer, the term *error control* refers primarily to methods of error detection and retransmission.
- Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

2.2 Protocols

The protocols are normally implemented in software by using one of the common programming languages.



- In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions. In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called **piggybacking**.

Simplest Protocol

- Simplest protocol is one that has no flow or error control.
- It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.

Design

- The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it.
- The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer.
- The data link layers of the sender and receiver provide transmission services for their network layers.
- The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits.
- The sender site cannot send a frame until its network layer has a data packet to send. The receiver site cannot deliver a data packet to its network layer until a frame arrives.

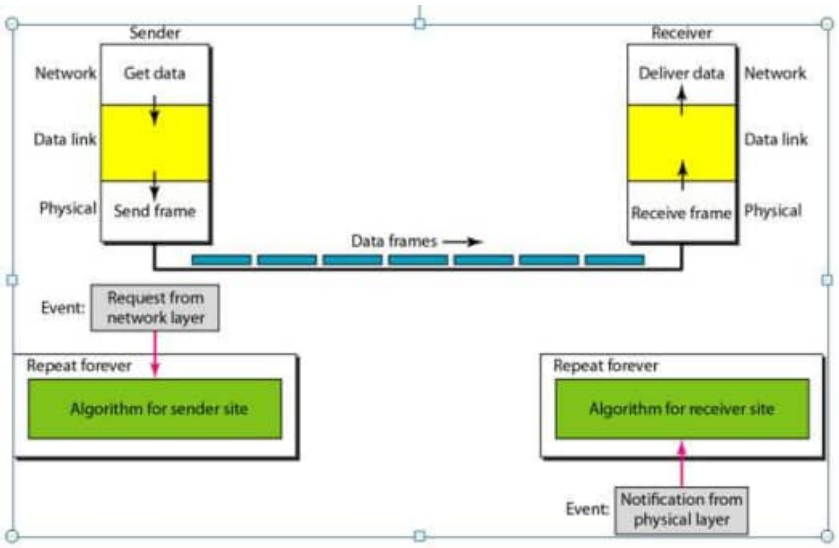


Fig: **The design of the simplest protocol with no flow or error control**

Figure below shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site.

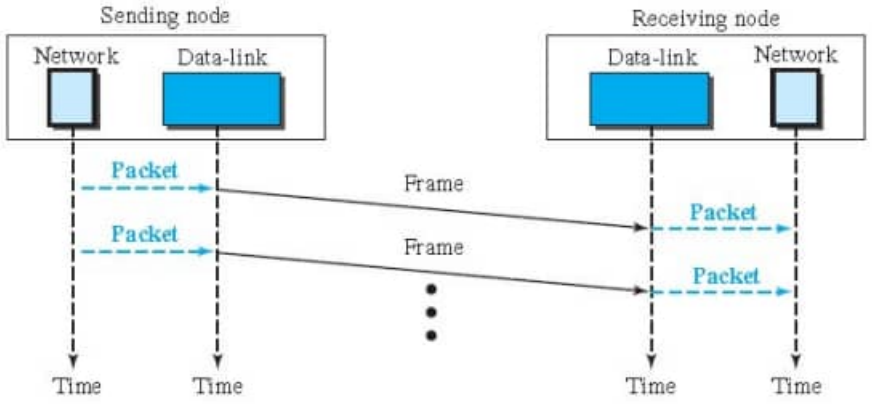
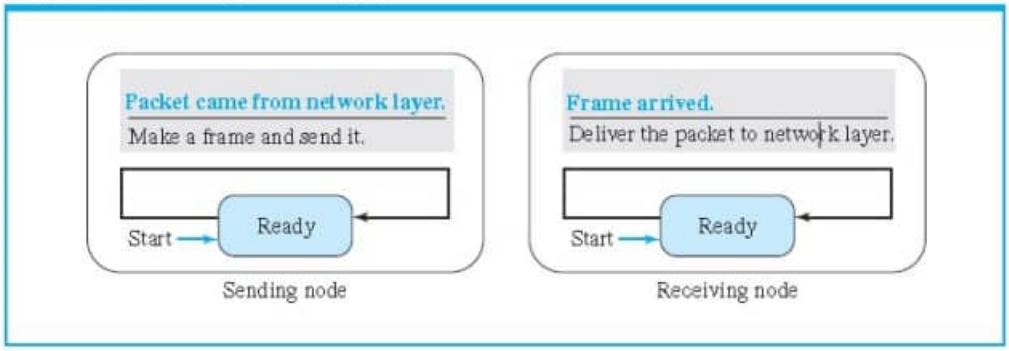


Fig: Flow diagram

FSMs

- Each FSM has only one state, the *ready state*. The sending machine remains in the ready state until a request comes from the process in the network layer. When this event occurs, the sending machine encapsulates the message in a frame and sends it to the receiving machine.
- The receiving machine remains in the ready state until a frame arrives from the sending machine. When this event occurs, the receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer.

Figure 11.8 *FSMs for the simple protocol*



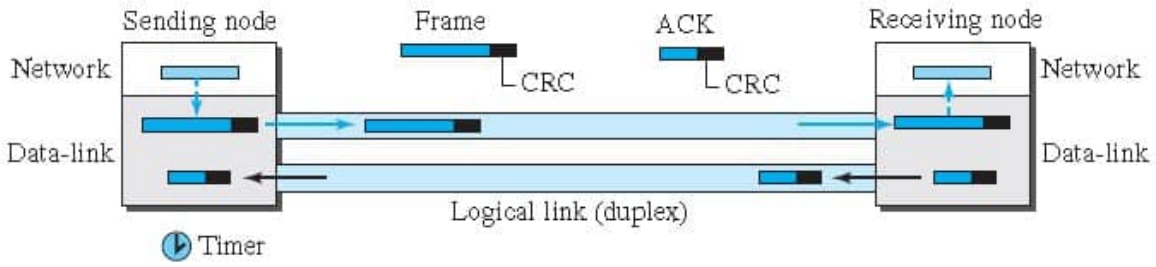
Stop-and-Wait Protocol

- If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use.
- Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service.
- To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.
- The protocol is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver, and then sends the next frame.

Design

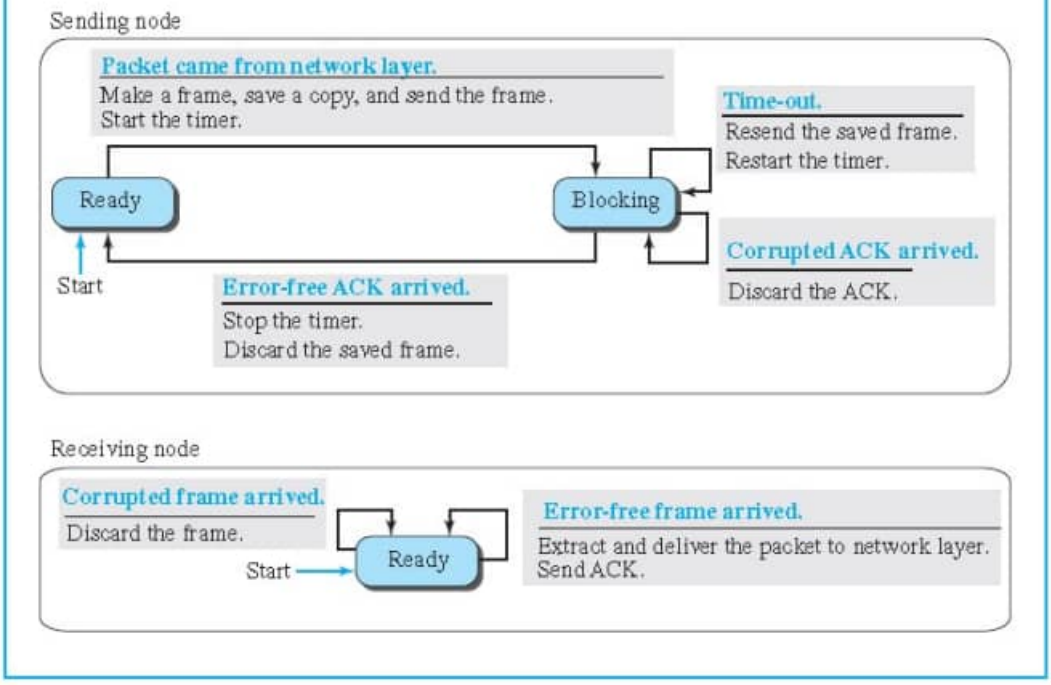
- At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. We therefore need a half-duplex link.

Figure 11.10 *Stop-and-Wait protocol*



- The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame.

Figure 11.11 *FSM for the Stop-and-Wait protocol*



Sender States

- The sender is initially in the ready state, but it can move between the ready and blocking state.

Ready State.

- When the sender is in this state, it is only waiting for a packet from the network layer. If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame.
- The sender then moves to the blocking state.

Blocking State.

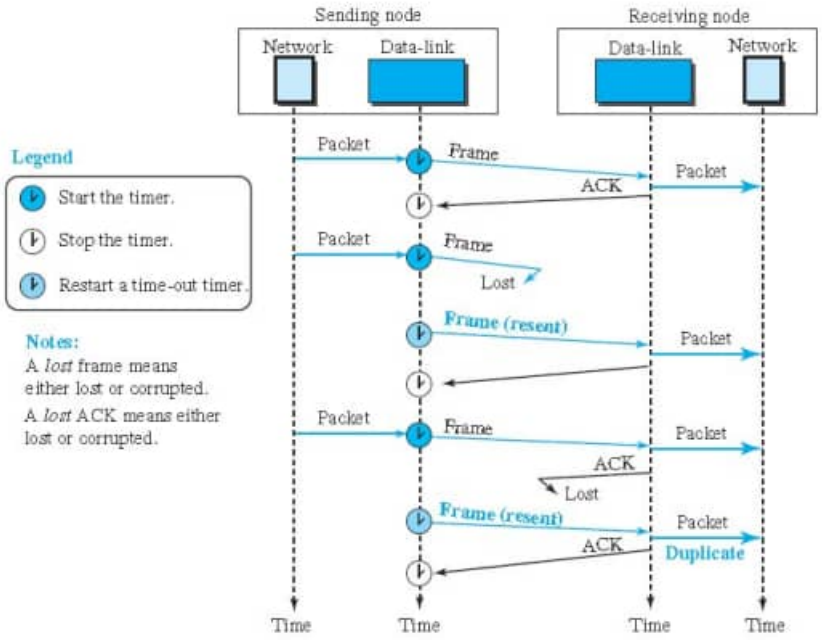
- When the sender is in this state, three events can occur:
  a. If a time-out occurs, the sender resends the saved copy of the frame and restarts the timer.

  b. If a corrupted ACK arrives, it is discarded.

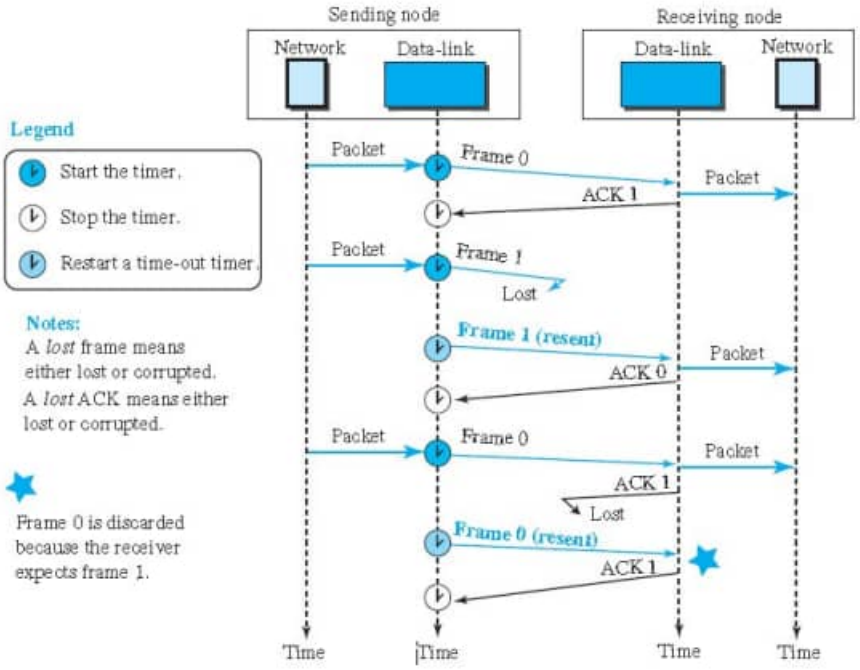
  c. If an error-free ACK arrives, the sender stops the timer and discards the saved copy of the frame. It then moves to the ready state.

Receiver

- The receiver is always in the ready state. Two events may occur:
  a. If an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.
  b. If a corrupted frame arrives, the frame is discarded.
- Figure below shows an example. The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent. However, there is a problem with this scheme. The network layer at the receiver site receives two copies of the third packet, which is not right.



- To correct the problem in the above Example , we need to add sequence numbers to the data frames and acknowledgment numbers to the ACK frames.
- However, numbering in this case is very simple. Sequence numbers are 0, 1, 0, 1, 0, 1, . . the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, … In other words, the sequence numbers start with 0, the acknowledgment numbers start with 1. An acknowledgment number always defines the sequence number of the next frame to receive.
- Figure below shows how adding sequence numbers and acknowledgment numbers can prevent duplicates. The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent.



2.3 HDLC

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

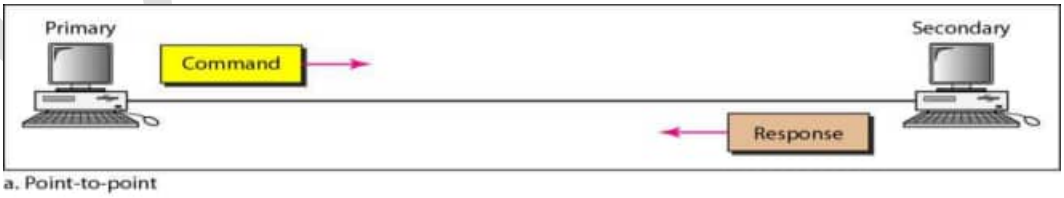
Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations:

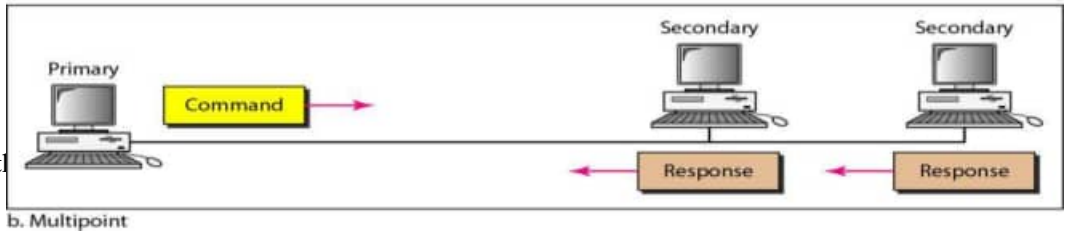
1. Normal response mode (NRM)
2. Asynchronous balanced mode (ABM)

Normal Response Mode:

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



a. Point-to-point



b. Multipoint

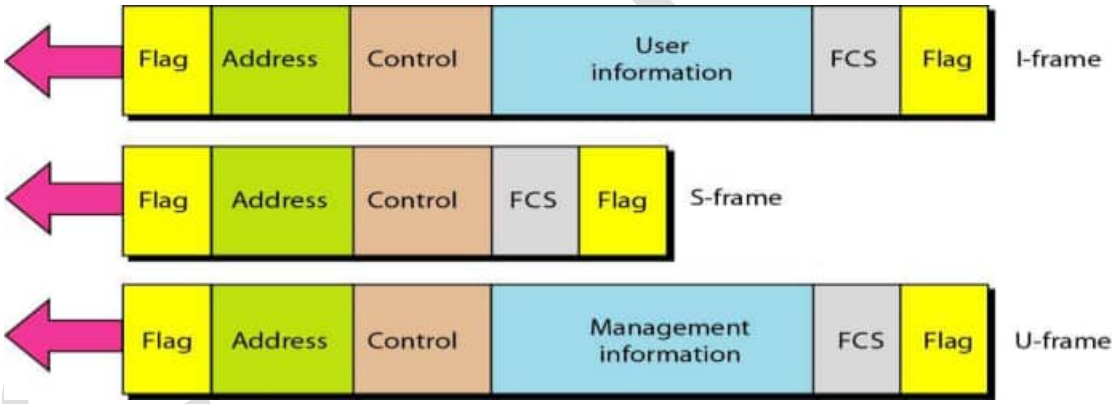
Asynchronous Balanced Mode

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

Frames

- HDLC defines three types of frames:
  1. Information frames (I-frames)
  2. Supervisory frames (S-frames)
  3. Unnumbered frames (U-frames)
- Each type of frame serves as an envelope for the transmission of a different type of message.
- I-frames are used to transport user data and control information relating to user data (piggybacking).
- S-frames are used only to transport control information.
- U-frames are reserved for system management. Information carried by U-frames is intended for managing the link itself.

Frame Format



Fields

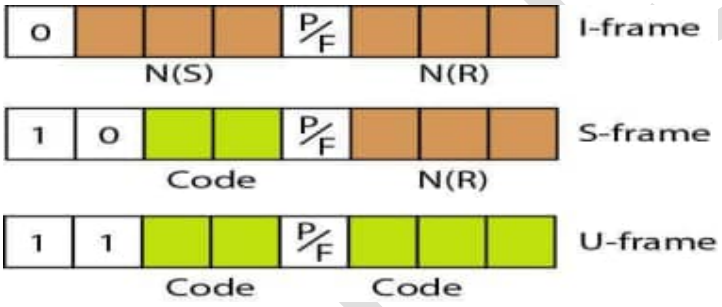
- **Flag field:** The flag field of an HDLC frame is an 8-bit sequence with the bit pattern 01111110 that identifies both the beginning and the end of a frame and serves as a synchronization pattern for the receiver.
- **Address field**: The second field of an HDLC frame contains the address of the secondary station. If a primary station created the frame, it contains a to address. If a secondary creates the frame, it contains a from address. An address field can be 1 byte or several bytes long,

depending on the needs of the network. One byte can identify up to 128 stations (l bit is used for another purpose). Larger networks require multiple-byte address fields. If the address field is only 1 byte, the last bit is always a 1. If the address is more than 1 byte, all bytes but the last one will end with 0; only the last will end with 1. Ending each intermediate byte with 0 indicates to the receiver that there are more address bytes to come.

- **Control field:** The control field is a 1- or 2-byte segment of the frame used for flow and error control. The interpretation of bits in this field depends on the frame type.
- **Information field**: The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- **FCS field:** The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte ITU-T CRC.

Control Field

The control field determines the type of frame and defines its functionality.



Control Field for I-Frames

- I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking).
- If the first bit of the control field is 0, this means the frame is an I-frame.
- The next 3 bits, called N(S), define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7; but in the extension format, in which the control field is 2 bytes, this field is larger.
- The last 3 bits, called N(R), correspond to the acknowledgment number when piggybacking is used.
- The single bit between N(S) and N(R) is called the P/F bit. The P/F field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means poll when the frame is sent by a primary station to a secondary. It means final when the frame is sent by a secondary to a primary.

Control Field for S-Frames

- Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate.
- S-frames do not have information fields.
- If the first 2 bits of the control field is 10, this means the frame is an S-frame.
- The last 3 bits, called N(R), corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame.
- The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

  Receive ready (RR): If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value N(R) field defines the acknowledgment number.

  Receive not ready (RNR): If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of NCR) is the acknowledgment number.

  Reject (REJ): If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-N ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of N(R) is the negative acknowledgment number.

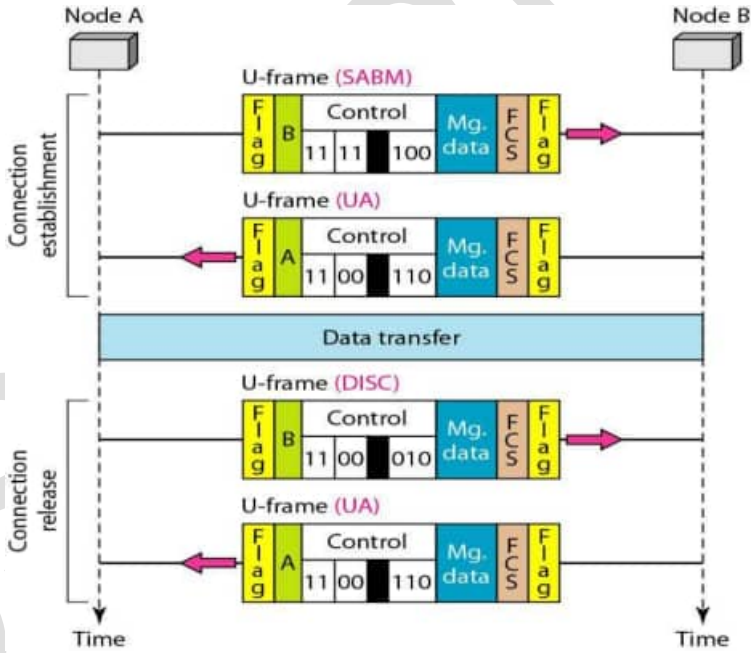
  Selective reject (SREJ): If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term selective reject instead of selective repeat. The value of N(R) is the negative acknowledgment number.

Control Field for V-Frames

- Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data.
- As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field.
- U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames. Some of the more common types are shown below.

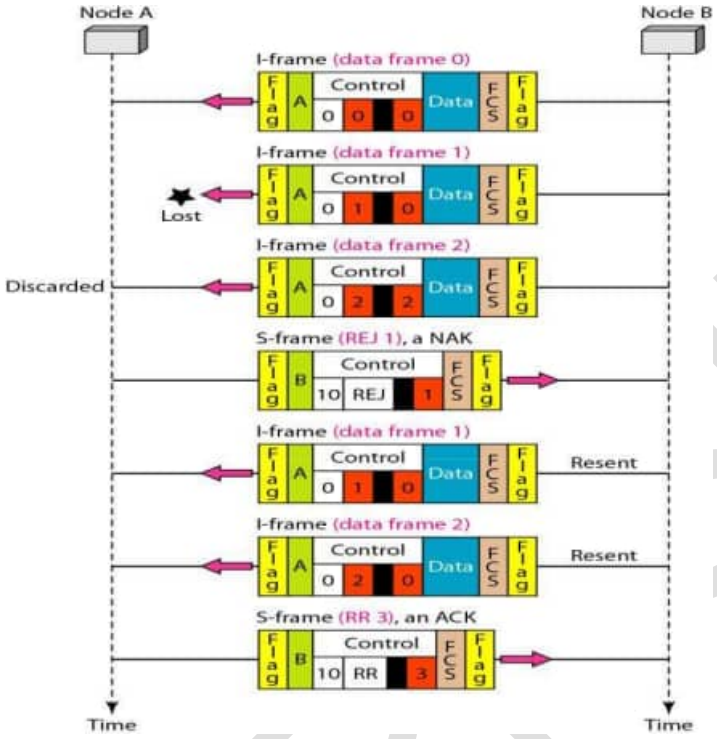
Code	Command	Response	Meaning
00 001	SNRM		Set normal response mode
11 011	SNRME		Set normal response mode, extended
11 100	SABM	DM	Set asynchronous balanced mode or disconnect mode
11 110	SABME		Set asynchronous balanced mode, extended
00 000	UI	UI	Unnumbered information
00 110		UA	Unnumbered acknowledgment
00 010	DISC	RD	Disconnect or request disconnect
10 000	SIM	RIM	Set initialization mode or request information mode
00 100	UP		Unnumbered poll
11 001	RSET		Reset
11 101	XID	XID	Exchange ID
10 001	FRMR	FRMR	Frame reject

Example: Connection/Disconnection



Example: Piggybacking without Error

Example: Piggybacking with error



Point-To-Point Protocol

- 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 provides several services:

1. PPP defines the format of the frame to be exchanged between devices.
2. PPP defines how two devices can negotiate the establishment of link and the exchange of data.
3. PPP defines how network layer data are encapsulated in the data link frame.
4. PPP defines how two devices can authenticate each other.
5. PPP provides multiple network layer services supporting a variety of network layer protocols.
6. PPP provides connections over multiple links.
7. PPP provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

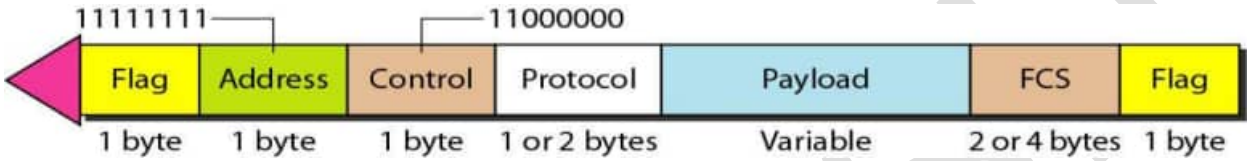
Limitations of PPP:

1. PPP does not provide flow control.

2. 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.
3. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

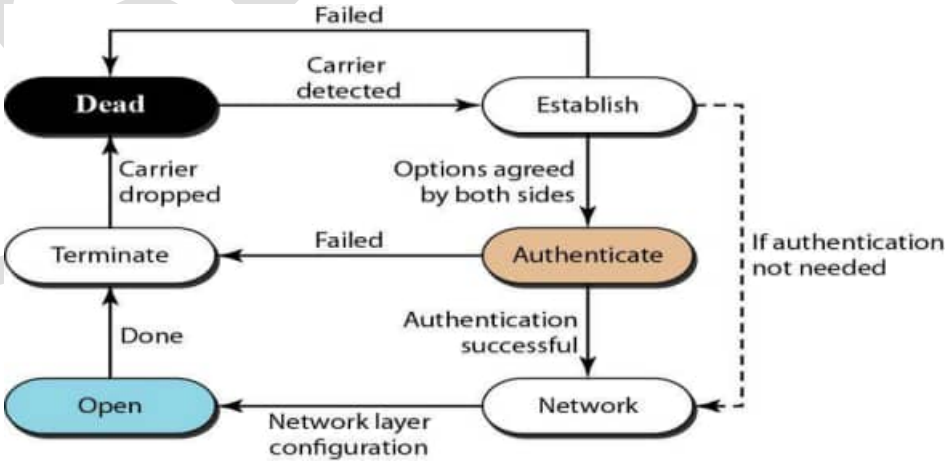
Framing

Frame Format



- **Flag:** A PPP frame starts and ends with 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 11000000
- **Protocol:** The protocol field defines what is being carried in the data field: either user data or other information.
- **Payload field:** This field carries either the user data or other information. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. 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.
- is a byte-oriented protocol using byte stuffing with the escape byte 01111101.

Transition Phases



- **Dead:** In the dead phase the link is not being used. There is no active carrier (at the physical layer) and the line is quiet.
- **Establish:** When one of the nodes starts the communication, the connection goes into this phase. In this phase, options are negotiated between the two parties. If the negotiation is successful, the system goes to the authentication phase (if authentication is required) or directly to the networking phase.
- **Authenticate:** The authentication phase is optional; the two nodes may decide, during the establishment phase, not to skip this phase. However, if they decide to proceed with authentication, they send several authentication packets. If the result is successful, the connection goes to the networking phase; otherwise, it goes to the termination phase.
- **Network:** In the network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at the network layer can be exchanged. The reason is that PPP supports multiple protocols at the network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive the data.
- **Open:** In the open phase, data transfer takes place. When a connection reaches this phase, the exchange of data packets can be started. The connection remains in this phase until one of the endpoints wants to terminate the connection.
- **Terminate:** In the termination phase the connection is terminated. Several packets are exchanged between the two ends for house cleaning and closing the link.

Multiplexing in PPP

- PPP: A link-layer protocol that uses various protocols to:
    1. Establish links
    2. Authenticate parties involved
    3. Carry network-layer data
- Three key protocols in PPP:
    1. Link Control Protocol (LCP)
    2. Authentication Protocols (APs)
    3. Network Control Protocols (NCPs)
- Allows carrying of data from multiple sources in the data field of PPP packets.

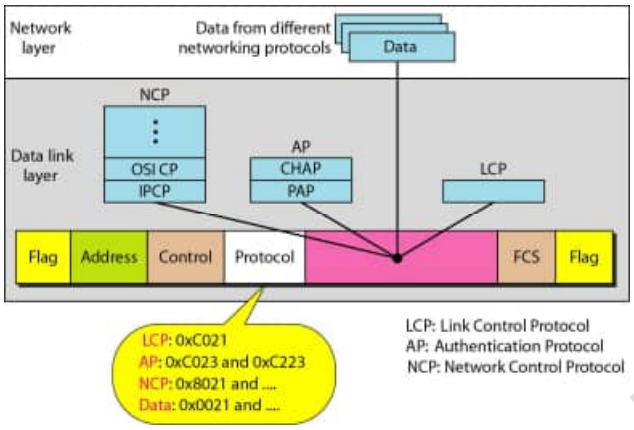


Figure: ***Multiplexing in PPP***

Link Control Protocol (LCP) - PPP

- Role of LCP:
  - **Establishes, configures, maintains, and terminates** links
  - Negotiates options between endpoints
  - Agreement on options required before link establishment
- LCP transitions through phases, illustrated by states like:
  - **Dead**
  - **Establish**
  - **Authenticate**
  - **Network**
  - **Terminate**

LCP packet encapsulated in a frame – PPP

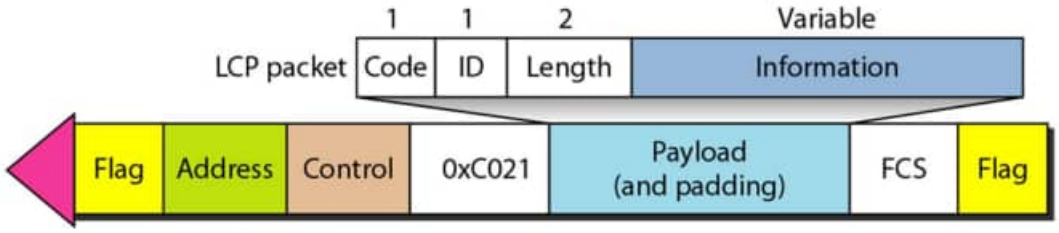


Figure : ***LCP packet encapsulated in a frame***

- **Fields in LCP Packet**:
  - **Code**: Packet type identifier (11 types of packets)
  - **ID**: Matches requests with replies
  - **Length**: Total length of the LCP packet
  - **Information**: Contains options, if needed
  - **LCP Packet Types**
- Three categories of LCP packets:
  - **Link Configuration** (e.g., Configure-request, Configure-ack)

- **Link Termination** (e.g., Terminate-request, Terminate-ack)
- **Link Monitoring/Debugging** (e.g., Echo-request, Echo-reply)

Code	*Packet Type*	*Description*
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

Table : *LCP packets*

Authentication Protocols in PPP

- Essential for user identity verification.
- Validates user identity for resource access.
- PPP Authentication Protocols:
    1. Password Authentication Protocol (PAP).
    2. Challenge Handshake Authentication Protocol (CHAP).

Password Authentication Protocol (PAP).

- Simple authentication process.
- Involves a two-step process between user and system.
- **PAP Authentication Steps**:
    1. **Step 1**: User sends an authentication ID (username) and password.
    2. **Step 2**: System verifies credentials and accepts or denies access.
- **PAP Packet Types**:
    1. **Authenticate-Request**: Sent by user (username and password).
    2. **Authenticate-Ack**: Sent by system to accept access.
    3. **Authenticate-Nak**: Sent by system to deny access.

PAP Packet Structure in PPP Frame

- Protocol Field Value: 0xC023 for PAP packets.
- Packet Types in Detail:
    1. Authenticate-Request: User credentials.
    2. Authenticate-Ack: Grant access.
    3. Authenticate-Nak: Deny access.
- PAP Usage Limitation:
    1. Vulnerable due to plaintext password transmission.

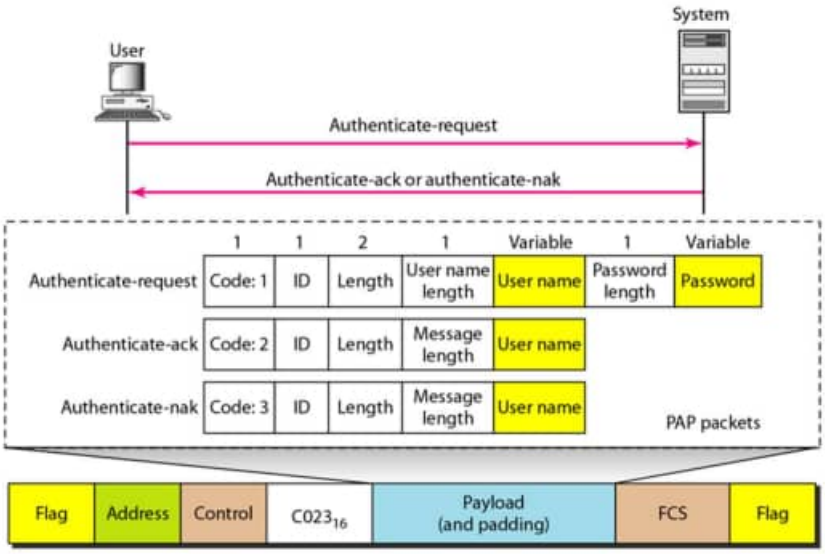


Figure: *PAP packets encapsulated in a PPP frame*

2.Challenge Handshake Authentication Protocol (CHAP)

- Secure, three-way handshake protocol.
- Password is never sent directly.
- CHAP Authentication Steps:
  - Step 1: System sends a challenge packet with a unique challenge value.
  - Step 2: User applies a function using challenge value + password, sends the result.
  - Step 3: System performs the same calculation and verifies the result.

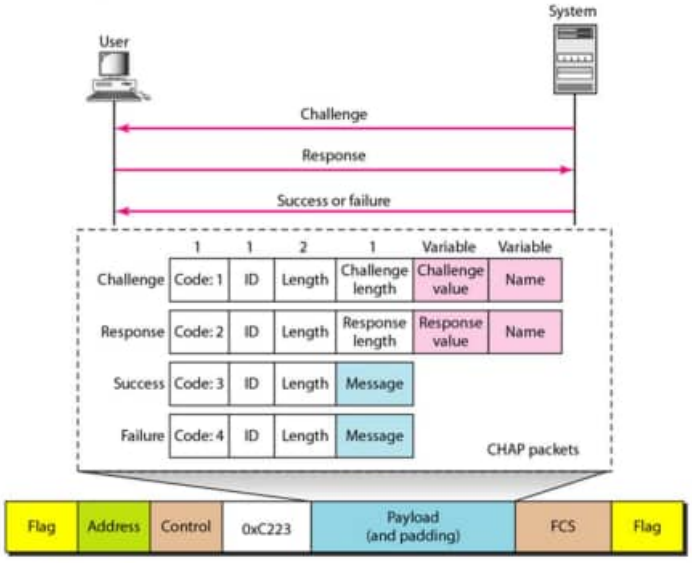


Figure: *CHAP packets encapsulated in a PPP frame*

- CHAP Packet Structure in PPP Frame
  - Protocol Field Value: 0xC223 for CHAP packets.

- CHAP Packet Types:
    - Challenge: System initiates authentication.
    - Response: User's calculated result.
    - Success: System grants access if results match.
    - Failure: System denies access if results don't match.

Comparison: PAP vs. CHAP

- PAP:
    - Simple two-step process.
    - Username and password sent directly.
    - Less secure due to plaintext transmission.
- CHAP:
    - Three-way handshake.
    - Password is never transmitted.
    - More secure due to dynamic challenge values.
- Best Practices:
    - Use CHAP for enhanced security in dial-up PPP links.

Network Control Protocols (NCP)

- Purpose: Configure link-layer settings for various network-layer protocols.
- Each network protocol has a specific NCP to handle configuration, e.g., IPCP for IP.

1. Internet Protocol Control Protocol (IPCP)
    - Configures links for IP data packets.
    - Encapsulated within a PPP frame, contains code, ID, length, and IPCP info.

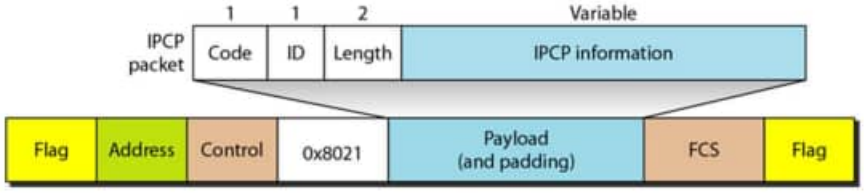


Figure: ***IPCP packet encapsulated in PPP frame***

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

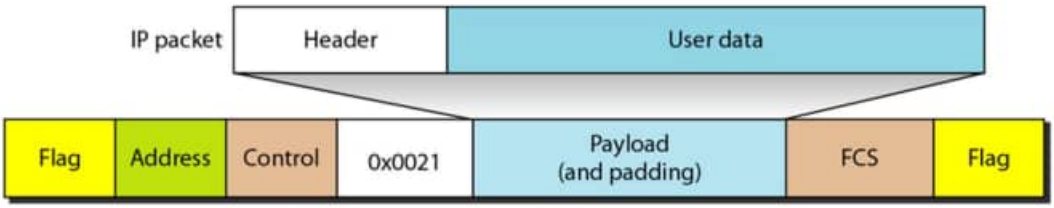


Figure: *IP datagram encapsulated in a PPP frame*

Multilink PPP (MLPPP)

- Original PPP Limitation: Single-channel point-to-point.
- Multilink PPP: Uses multiple channels to split a logical PPP frame across physical frames.
- Protocol Field in MLPPP: Set to 0x003d to indicate fragmentation.
- Additional Complexity: Sequence numbers to indicate fragment order.

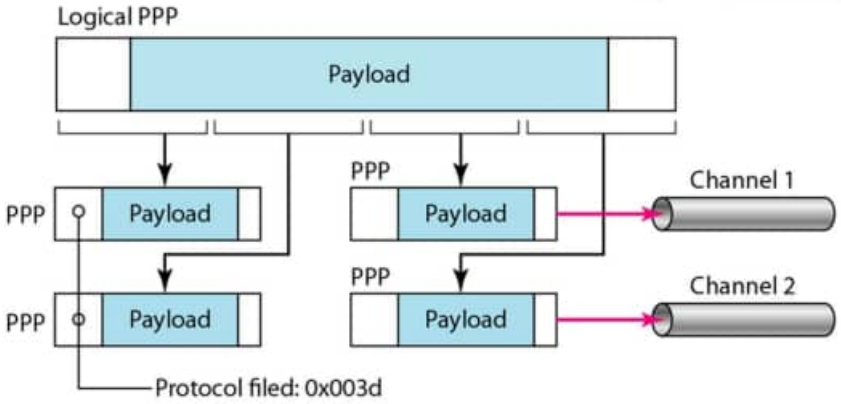
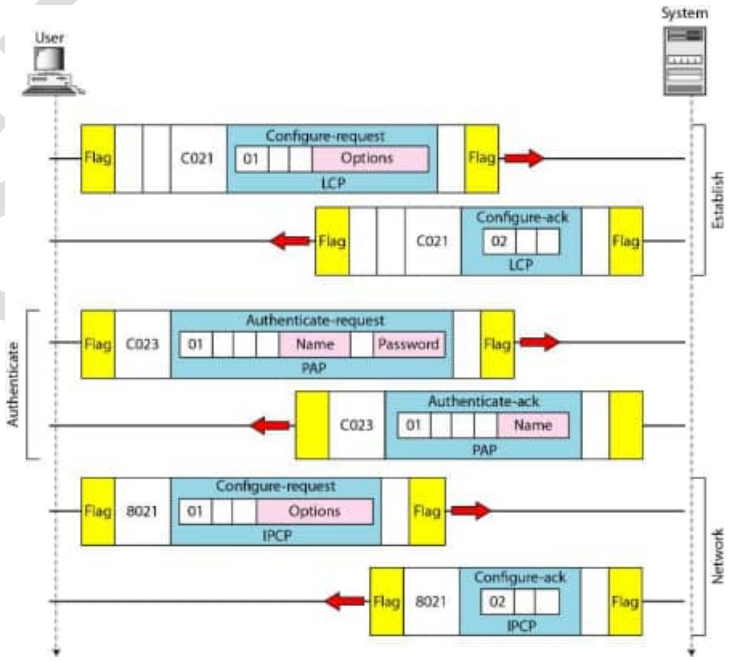


Figure: *Multilink PPP*

Phases of Network Layer Packet Transmission in PPP Connection



1. **PPP Transmission:**

Steps for sending data from user site to system site (e.g., sending an email through an ISP).
Unidirectional data flow assumed for simplicity.

2. **Link Establishment:**

Frames 1 and 2: Establish the link.

3. **Authentication Phase:**

Frames 3 and 4: Execute authentication with PAP.

4. **Network Layer Connection Setup:**

Frames 5 and 6: Establish Network Layer connection using IPCP (IP Control Protocol).

5. **Data Transfer:**

Encapsulation of IP packets within PPP frames.
System can recognize data for IP protocol as IPCP was used to establish the connection.

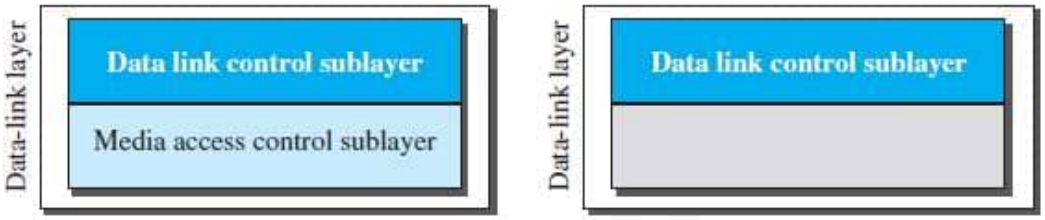
6. **Termination of Connection:**

User terminates Data Link Layer connection, which system acknowledges.
Optionally, user/system can terminate IPCP at the Network Layer while keeping the Data Link Layer active to initiate a new NCP protocol.

Media Access control

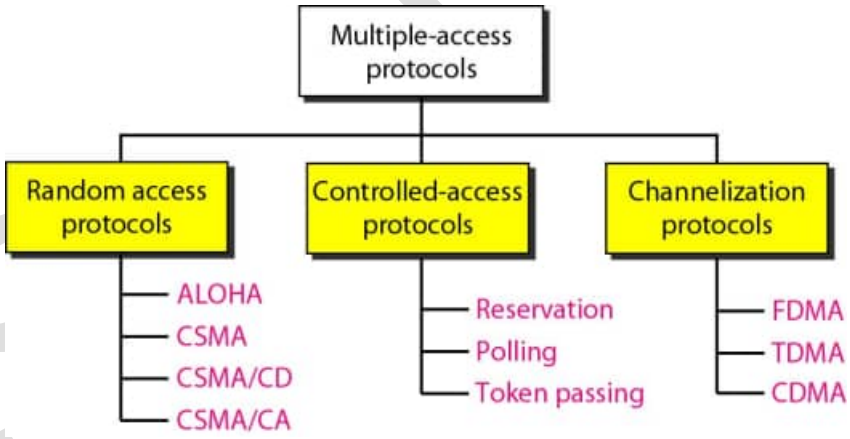
- **Data link layer is subdivided into two sublayers: data link control (DLC) and media access control (MAC).**
- The data link control sublayer deals with all issues common to both point-to-point and broadcast links. The media access control sublayer deals only with issues specific to broadcast links. In other words, we separate these two types of links at the data-link layer, as shown in Figure 9.4.

Figure 9.4 *Dividing the data-link layer into two sublayers*



a. Data-link layer of a broadcast link b. Data-link layer of a point-to-point link

- 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. Taxonomy of multiple access protocols is given below:



3.1 RANDOM ACCESS or CONTENTION methods

- In random access or contention methods, no station is superior to another station and none is assigned the control over another.
- A station that has data to send uses a procedure defined by the protocol to make a decision to send or not to send. This decision depends on the state of the medium (idle or busy).

Main features are:

1. There is no scheduled time for a station to transmit. Transmission is random among the stations, hence called *random access.*
2. No rules specify which station should send next. Stations compete with one another to access the medium, hence called *contention* methods.

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

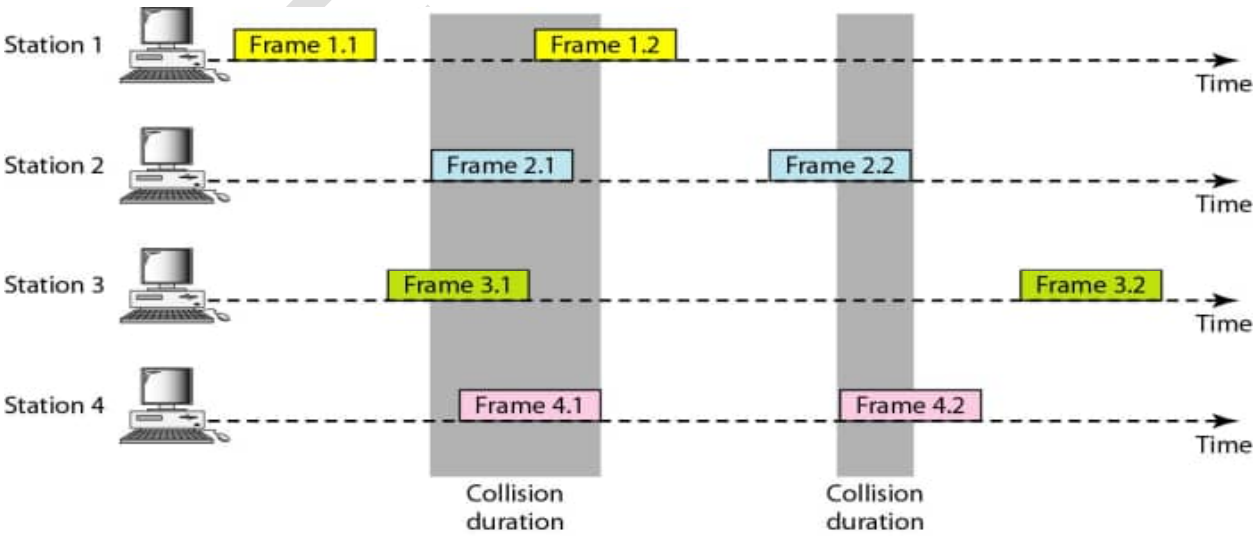
- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

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.

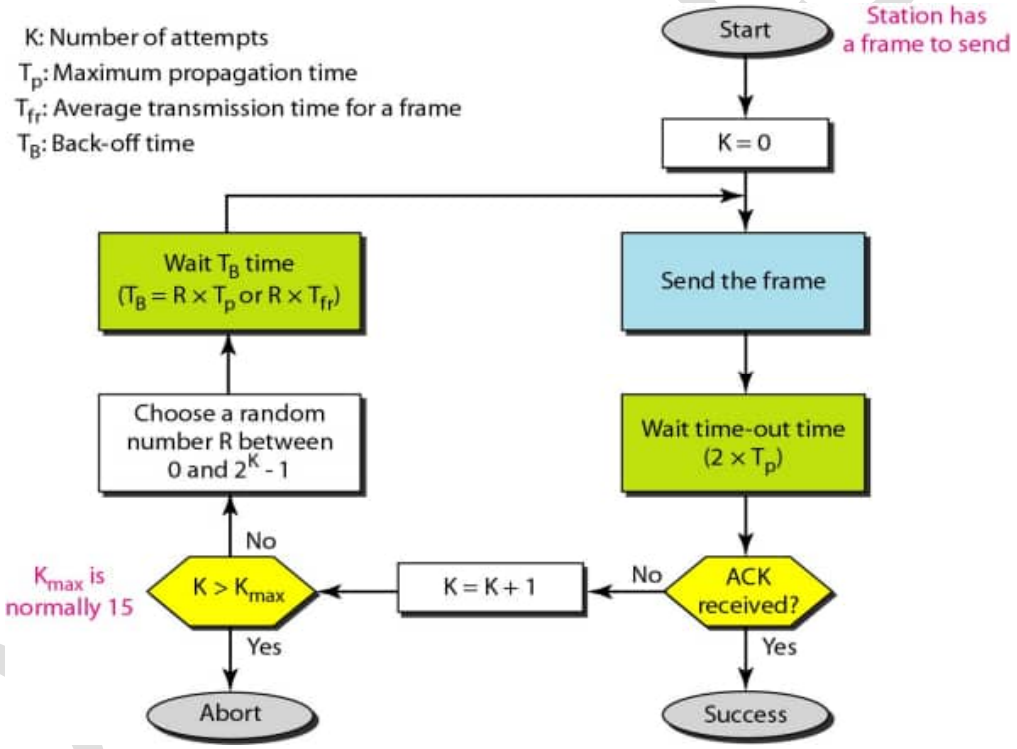
Pure ALOHA

- The original ALOHA protocol is called pure ALOHA.
- It supports multiple access. The idea is that each station sends a frame whenever it has a frame to send.
- However, since there is only one channel to share, there is the possibility of collision between frames from different stations.
- Figure below shows an example of frame collisions in pure ALOHA.



- Here four stations that contend with one another for access to the shared channel.
- Each station sends two frames, total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel.

- Figure above shows that only two frames survive, frame 1.1 from station 1 and frame 3.2 from station 3.
- If one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.
- 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 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. This randomness will help avoid more collisions.
- The time is back-off time T_B. *Below f*igure 12.4 shows the procedure for pure ALOHA based on the above strategy.



- The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations (2 x T_p).
- The back-off time T_B is a random value that normally depends on K (the number of attempted unsuccessful transmissions).
- Formula for T_B depends on the implementation. One common formula is the binary exponential back-off. In this method, for each retransmission, a multiplier in the range 0 to

2^K - 1 is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find T_B.

Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, Then

$Tp = (600 \times 10^3) / (3 \times 10^8) = 2$ ms.

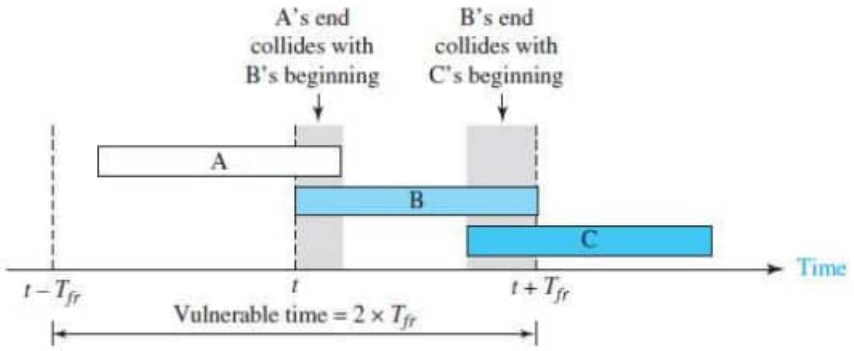
For K = 2, the range of R is {0, 1, 2, 3}

This means that TB can be 0, 2, 4, or 6 ms, based on the outcome of the random variable R.

Vulnerable time

- It is the Length of time, in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} Sec to send. Figure 12.5 shows the vulnerable time for station A.

Figure 12.4 *Vulnerable time for pure ALOHA protocol*



- Station A sends a frame at time *t*. Now imagine station B has already sent a frame between *t* - T_{fr} and *t*.
- This leads to a collision between the frames from station A and station B. The end of B's frame collides with the beginning of A's frame.
- Suppose that station C sends a frame between *t* and *t* + T_{fr} . Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.
- Figure 12.4, shows vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

Pure ALOHA vulnerable time = 2 x T_{fr}

Example 12.2

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

Solution:

- Average frame transmission time T_{fr} is 200 bits/200 kbps or 1 ms.
- The vulnerable time is 2 × 1 ms = 2 ms.
- This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the period (1 ms) that this station is sending.

Throughput

- Let G be the average number of frames generated by the system during one frame transmission time.
- **Then average number of successfully transmitted frames for pure ALOHA is S = G × e^{-2G}.**
- **The maximum throughput S_{max} is 0.184 for G = ½.**
- In other words, if one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully.

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

Solution

The frame transmission time is *200/200* kbps or 1 ms.

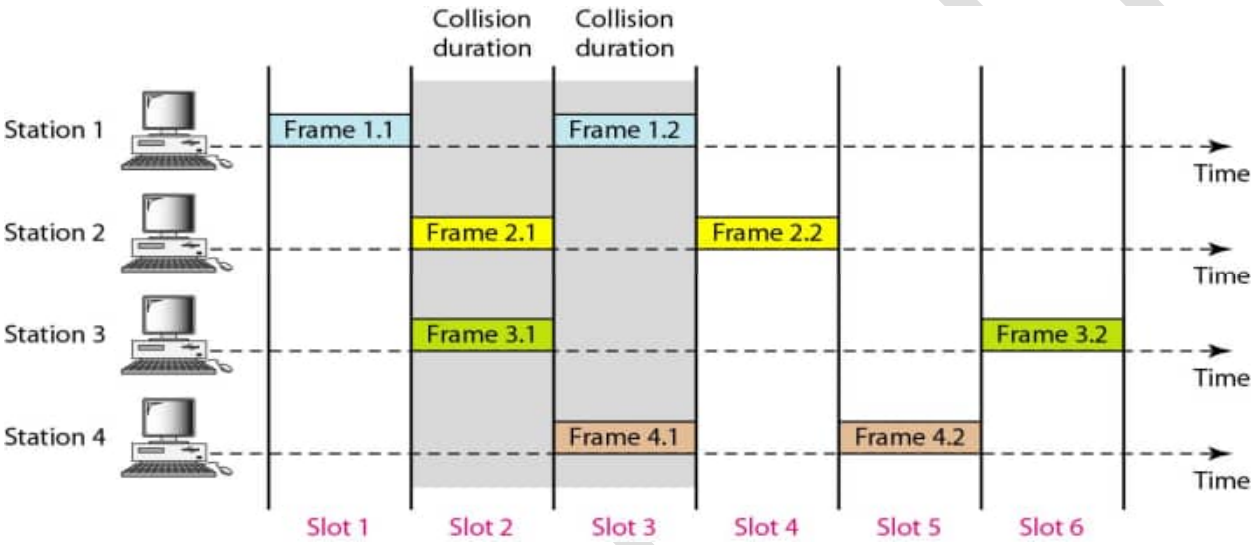
a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case $S = G \times e^{-2G}$ or $S = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.

b. If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is *(1/2)*. In this case $S = G \times e^{-2G}$ or $S = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentagewise.

c. If the system creates 250 frames per second, this is (1/4) frame per millisecond. The load is (1/4). In this case $S = G \times e^{-2G}$ or $S = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Slotted ALOHA

- Pure ALOHA has a vulnerable time of 2 x T_{fr} . This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
- In slotted ALOHA, we divide the time into slots of T_{fr}'s and force the station to send only at the beginning of the time slot. Figure below shows an example of frame collisions in slotted ALOHA.



- Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot.
- This means that the station which started at the beginning of this slot has already finished sending its frame.
- There is still the possibility of collision if two stations try to send at the beginning of the same time slot.
- The vulnerable time is now reduced to one-half, equal to T_{fr}. Figure below shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

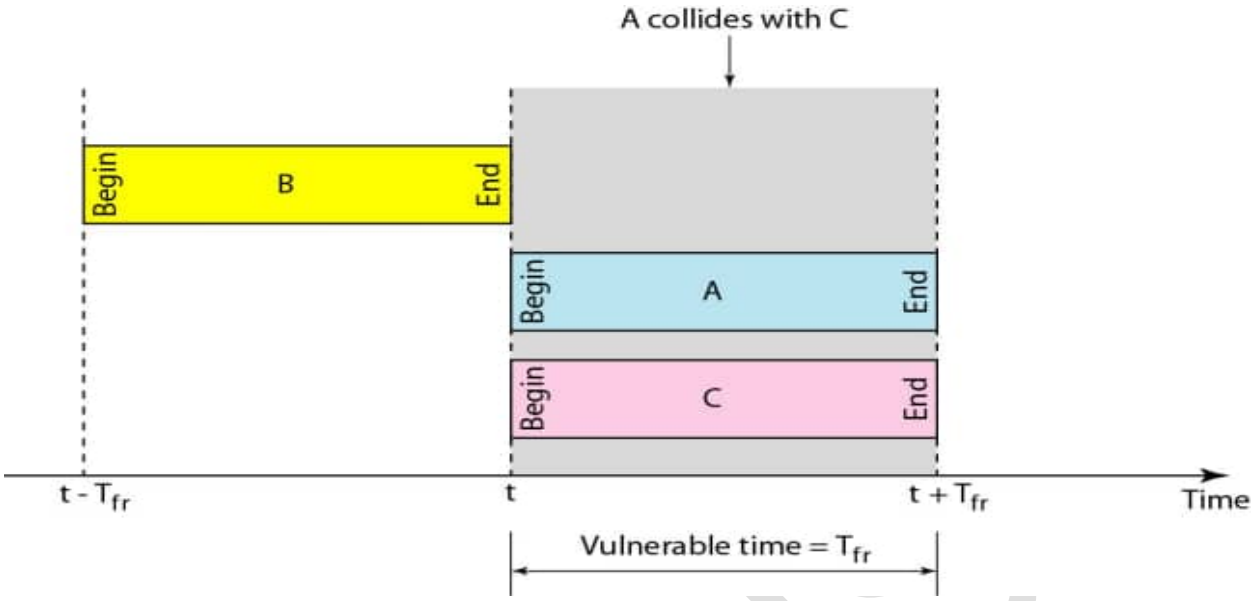
Slotted ALOHA vulnerable time = T_{fr}

Throughput

- It can be proved that the average number of successful transmissions for slotted ALOHA is

$$S = G \times e^{-G}.$$

The maximum throughput S_{max} is 0.368, when G = 1.



Example 12.4

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces
a. 1000 frames per second
b. 500 frames per second
c. 250 frames per second

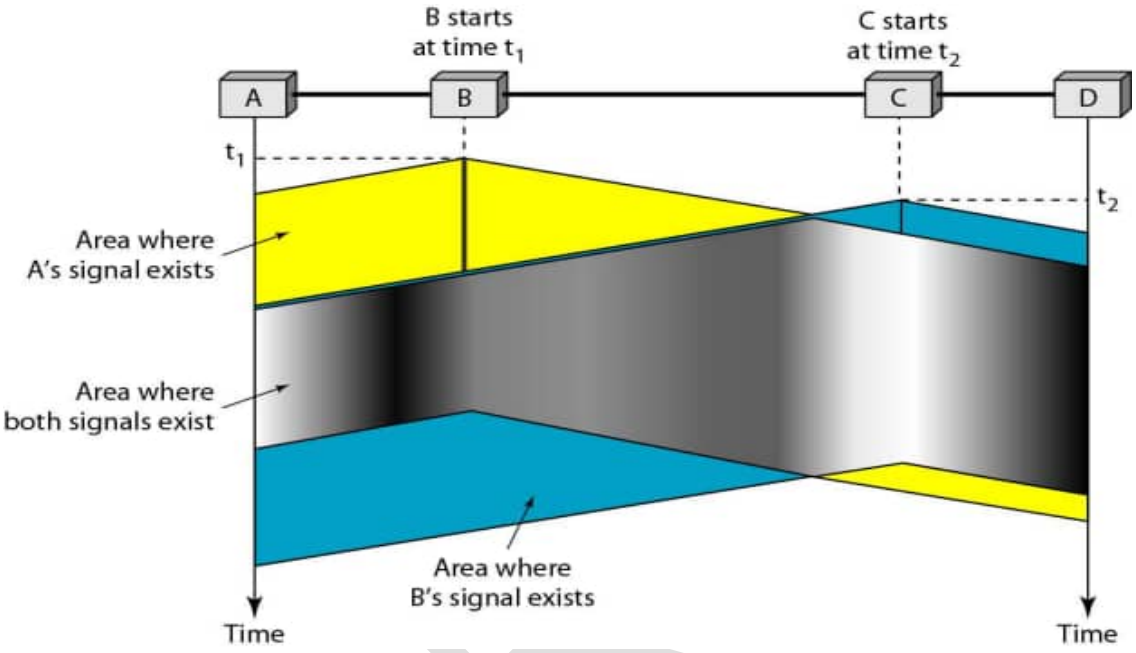
Solution

- The frame transmission time is *200/200* kbps or 1 ms.
- a. In this case G is 1. S =G x e^{-G} or *S* =0.368 (36.8 percent). This means that the throughput is 1000 x 0.0368 =368 frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
- b. Here G is1/2. In this case S =G x e^{-G} or S =0.303 (30.3 percent). This means that the throughput is 500 x 0.0303 =151. Only 151 frames out of 500 will probably survive.
- c. Now G is1/4. In this case S =G x e^{-G} or S =0.195 (19.5 percent). This means that the throughput is 250 x 0.195 = 49. Only 49 frames out of 250 will probably survive.

Carrier Sense Multiple Access (CSMA)

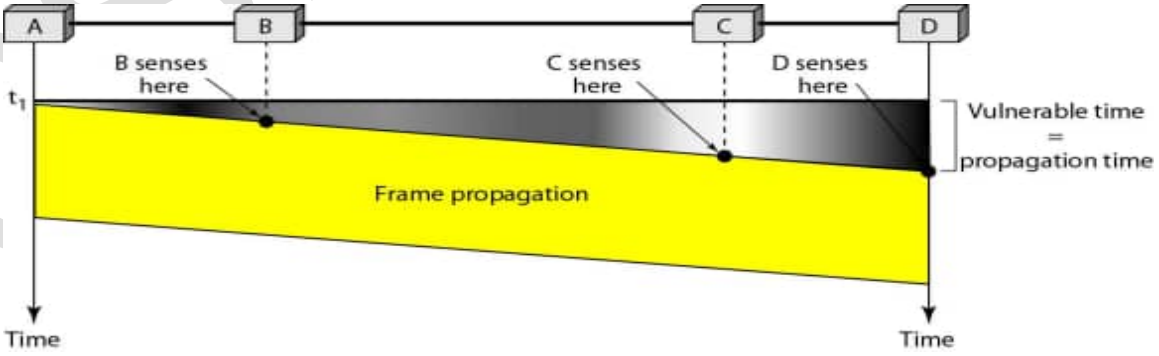
- Carrier sense multiple access (CSMA) requires that each station first listen to the medium before sending. The reason for this is shown in Figure below, a space and time model of a CSMA network.
- Stations are connected to a shared channel. The possibility of collision still exists because of propagation delay;

- When a station sends a frame, it still takes time for the first bit to reach every station and for every station to sense it. At time *t1* station B senses the medium and finds it idle, so it sends a frame. At time *t2 (t2> t1)* station C senses the medium and finds it idle because, at this time, the first bit from station B has not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.



Vulnerable Time

- **The vulnerable time for CSMA is the propagation time *Tp* . This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result**.
- Figure below shows the worst case. The leftmost station A sends a frame at time t1 which reaches the rightmost station D at time *t1 + Tp.*

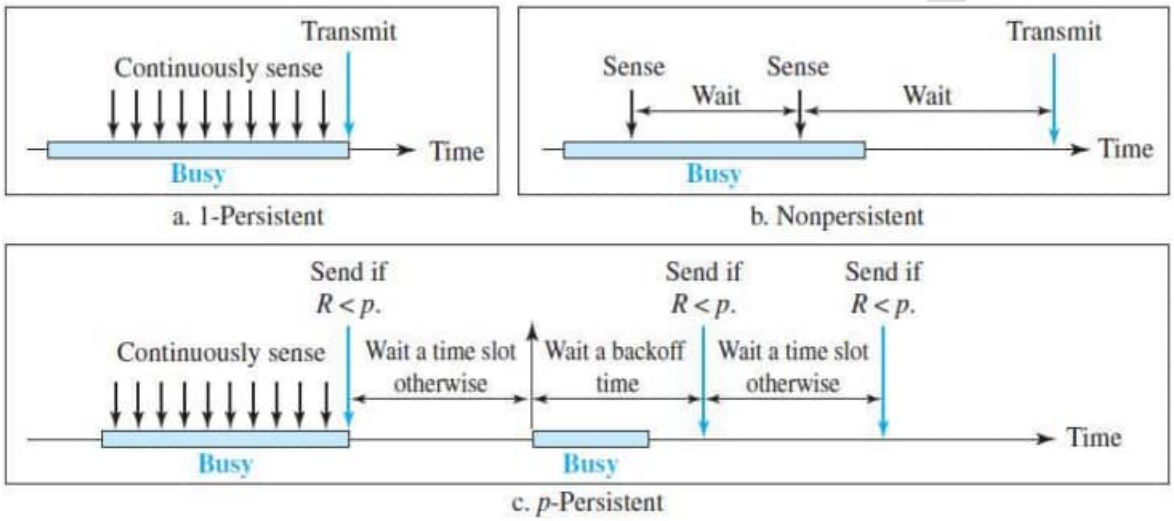


Persistence Methods

- What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions.

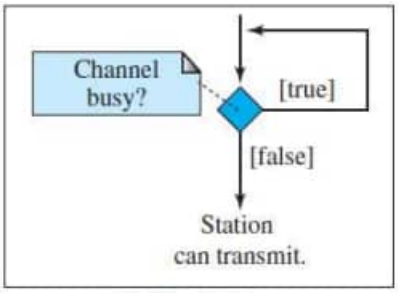
1. I-persistent method 2. nonpersistent method and 3. p-persistent method.

Figure below shows the behavior of three persistence methods when a station finds a channel busy.

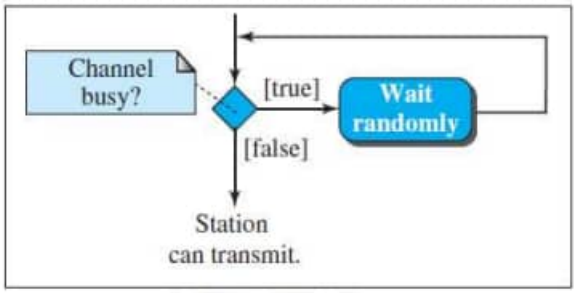


- **1-Persistent:** In this method, after the station finds the line idle, it sends its frame immediately. This method has the highest chance of collision because two or more stations may find the line idle at the same time and send their frames immediately.
- **Non-persistent:** A station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The non-persistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network.
- **P-Persistent:** Used if the channel has time slots with slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows steps:

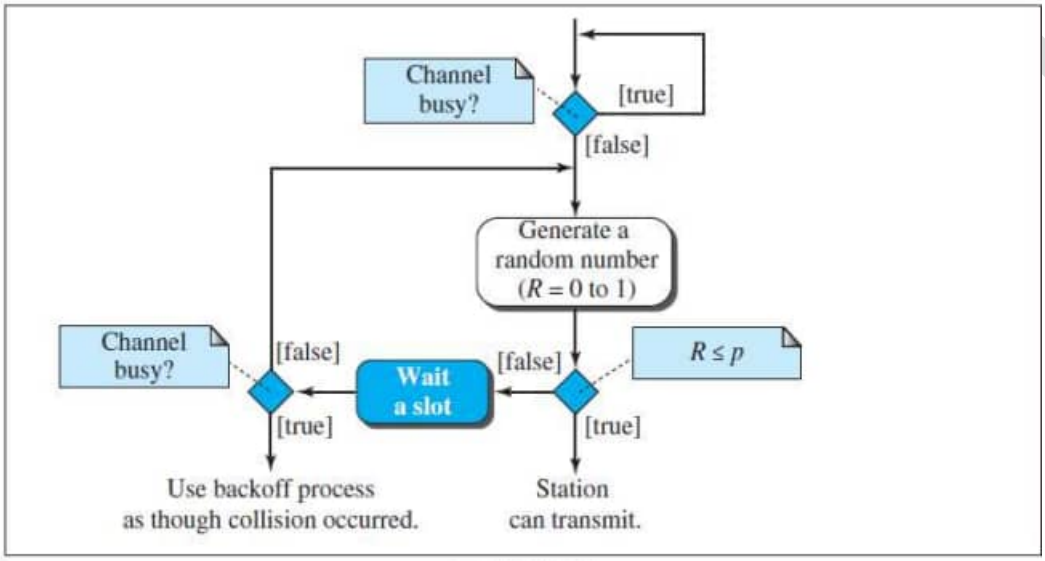
  1. With probability *p,* the station sends its frame.
  2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.
     a. If the line is idle, it goes to step 1.
     b. If the line is busy, it acts as though a collision has occurred and uses the back-off procedure.
- Flow diagram for three persistence methods are given below.



a. 1-Persistent



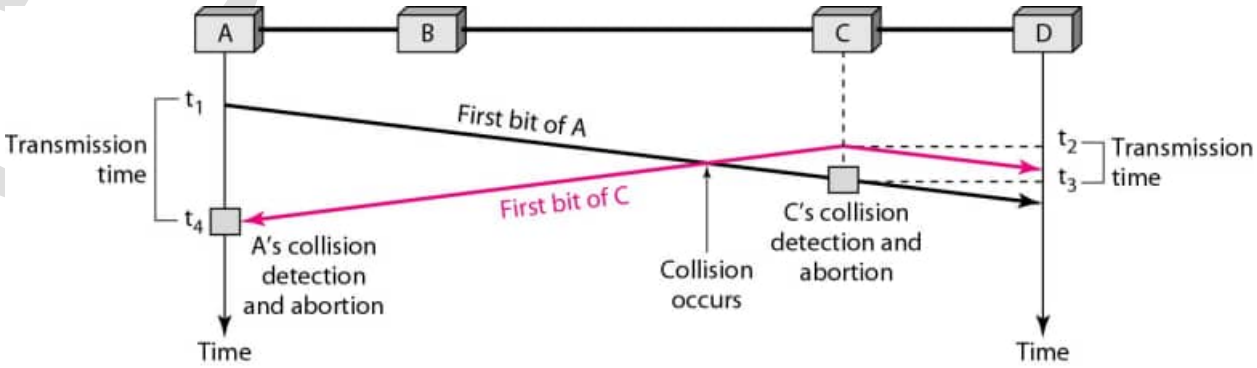
b. Nonpersistent



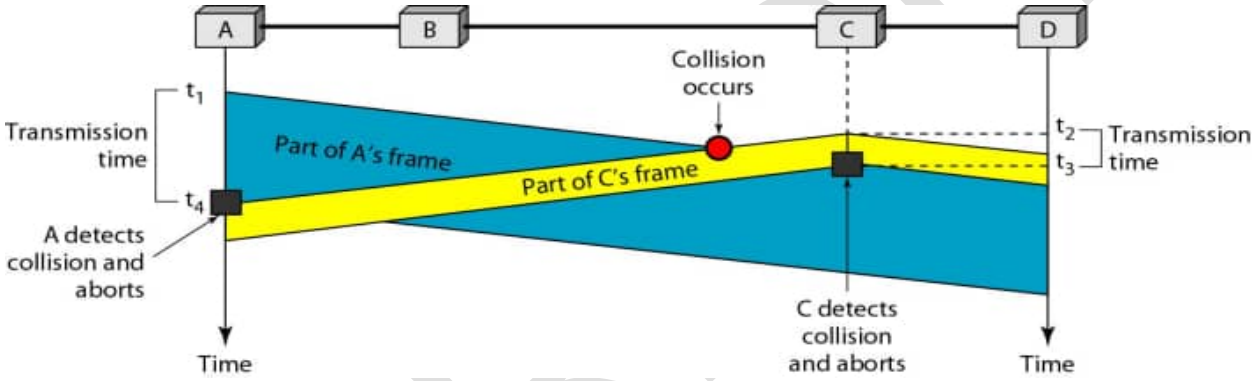
c. *p*-Persistent

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

- Carriers sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision.
- A **station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished else the frame is sent again**.
- Below Figure shows stations A and C are involved in the collision.



- At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which **propagate both to the left and to the right**.
- The collision occurs sometime after time t_2 Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately aborts transmission. Station A detects collision at time t_4 when it receives the first bit of C's frame, it also immediately aborts transmission.
- Station A transmits for the duration $t_4 - t_1$ *and* C is $t_3 - t_2$. At time t_4, station A aborts its transmission and time *t3 station C aborts* its transmission. Figure below shows complete graph.



Minimum Frame Size

- For *CSMA/CD* to work, there should be a restriction on the frame size.
- Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission.
- This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection.
- **Therefore the frame transmission time T_{fr} must be at least two times the maximum propagation time T_p.**

$$T_{fr} = 2 * T_p$$

Example 12.5

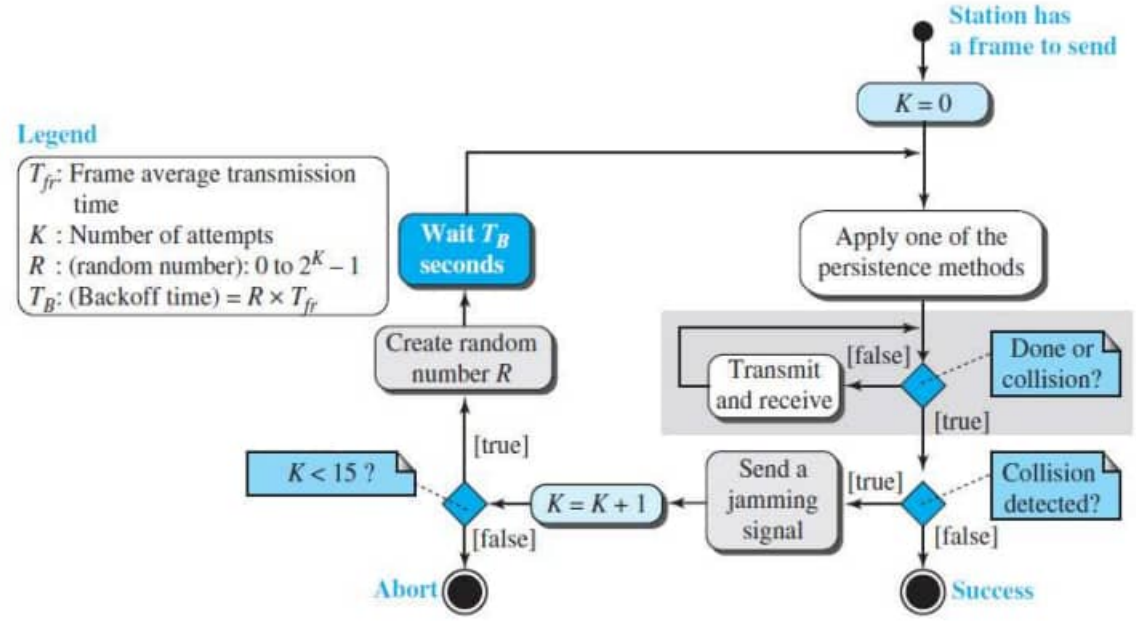
- A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

Solution

The minimum frame transmission time is $T_{fr} = 2 \times Tp = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision.
The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes.

Procedure

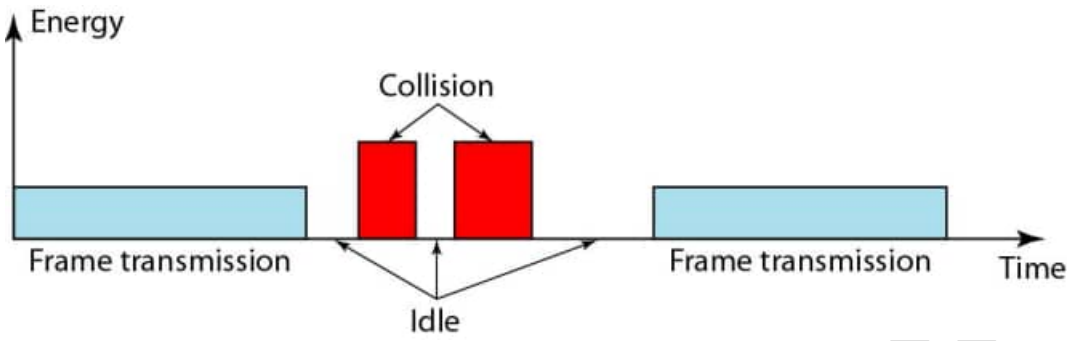
- Flow diagram for *CSMA/CD is as* shown below figure. It is similar to the one for the ALOHA protocol, **but there** are differences.



- **First difference** is the addition of the **persistence** process. The corresponding box can be replaced by one of the persistence processes.
- The **second difference is** the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In *CSMA/CD,* transmission and collision detection is a continuous process. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously.
- **The third difference** is the sending of a **short jamming signal** that enforces the collision in case other stations have not yet sensed the collision.

Energy Level

- The level of energy in a channel can have **three values: zero, normal, and abnormal**.
- At the zero level, the channel is idle.
- At the normal level, a station has successfully captured the channel and is sending its frame.
- At the abnormal level, there is a collision and the level of the energy is twice the normal level.
- A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode. Figure below shows the situation.

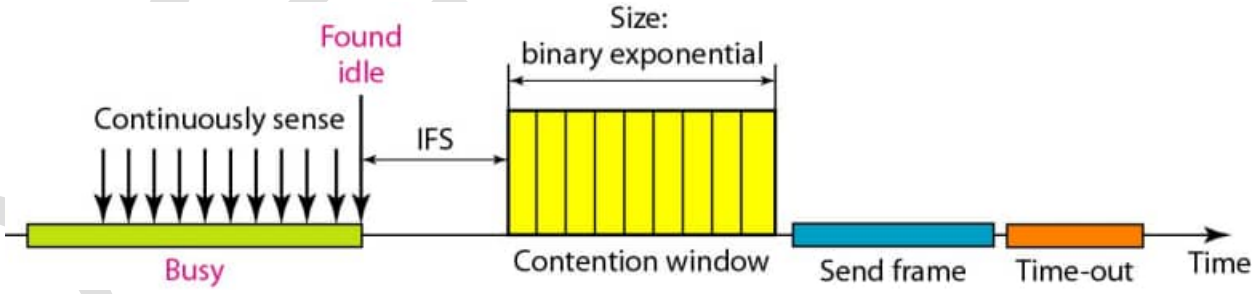


Throughput

- The throughput of *CSMA/CD* is greater than that of pure or slotted ALOHA.
- The maximum throughput occurs at a different value of G and is based on the persistence method and the value of *p* in the p-persistent approach.
- For 1-persistent method the maximum throughput is around 50 percent when G =1.
- For non-persistent method, the maximum throughput can go up to 90 percent when G is between 3 and 8.

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

- **Carrier sense multiple access with collision avoidance (CSMA/CA)** was invented for wireless networks.
- Collisions are avoided through the use of CSMA/CA's three strategies:
  1. **The interframe space(IFS)**
  2. **Contention window and**
  3. **Acknowledgments**



Interframe Space (IFS)

- **Collisions are avoided by deferring transmission even if the channel is found idle.** When an idle channel is found, the station does not send immediately. It waits for a period of time called the interframe space or IFS, because distant station may have already started transmitting. The distant station's signal has not yet reached this station. *The IFS time allows the front of the transmitted signal by the distant station to reach this station.* If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time

equal to the contention time. The IFS variable can also be used to prioritize stations or frame types.

Contention Window

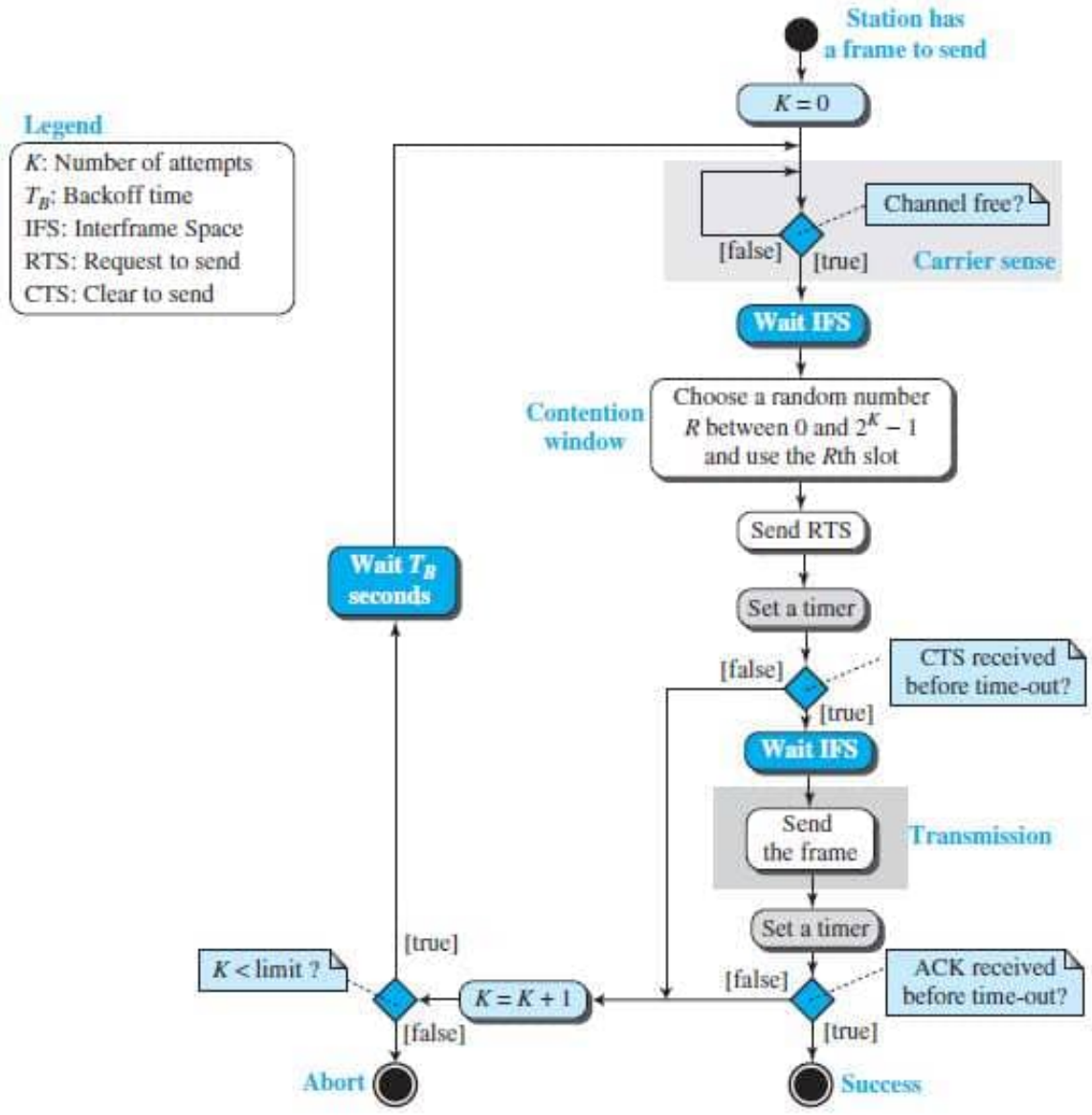
- **The contention window is an amount of time divided into slots**. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the **binary exponential back-off strategy**. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. One interesting point about the contention window is that the **station needs to sense the channel after each time slot.** However, if the station finds the channel busy, it does not restart the process. It just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

Acknowledgment

- With all these precautions, still there may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. **The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.**

- Figure below shows the flow chart of CSMA/CA.

Figure 12.15 *Flow diagram of CSMA/CA*



Frame Exchange Time Line

- Figure below shows the exchange of data and control frames in time.

1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.

a. The channel uses a persistence strategy with backoff until the channel is idle.

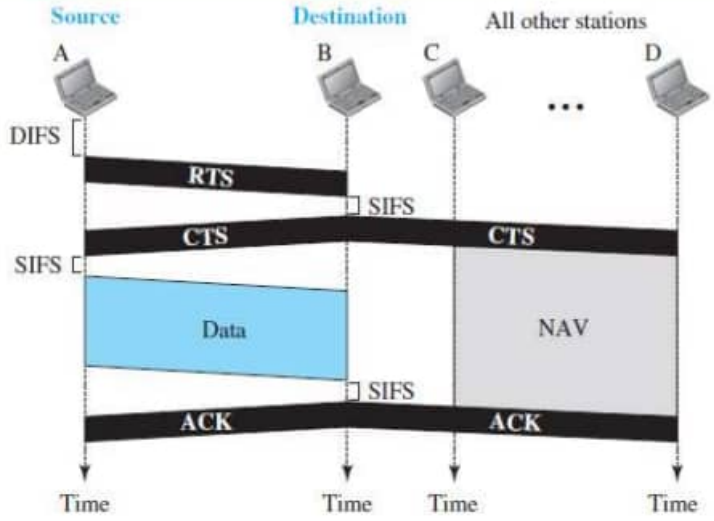
b. After the station is found to be idle, the station waits for a period of time called the ***DCF interframe space (DIFS);*** then the station sends a control frame called the *request to send (RTS)*.

2. After receiving the RTS and waiting a period of time called the ***short interframe space (SIFS),*** the destination station sends a control frame, called the *clear to send (CTS),* to the source station. This control frame indicates that the destination station is ready to receive data.

3. The source station sends data after waiting an amount of time equal to SIFS.

4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in CSMA/CD is a kind of indication to the source that data have arrived.

Figure 12.17 *CSMA/CA and NAV*



Network Allocation Vector

- How do other stations defer sending their data if one station acquires access? In other words, how is the collision avoidance aspect of this protocol accomplished? The key is a feature called NAV.
- When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a network allocation vector (NAV) that shows how much time must pass before these stations are allowed to check the channel for idleness.
- Each time a station accesses the system and sends an RTS frame, other stations start their NAV. In other words, each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired. Figure 12.17 shows the idea of NAV.

Collision During Handshaking

- Collision can happen during the time when RTS or CTS control frames are in transition, often called the handshaking period. Two or more stations may try to send RTS frames at the same time. These control frames may collide. However, because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The backoff strategy is employed, and the sender tries again.

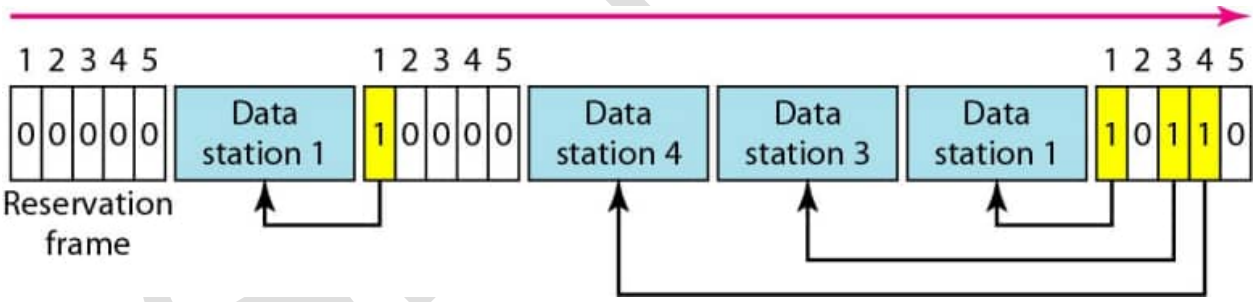
3.2 CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send.

1. Reservation

- **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 mini-slots in the reservation frame, each mini-slot belongs to a station.
- When a station needs to send a data frame, it makes a reservation in its own mini-slot. Figure 12.18 shows a situation with five stations and a five-mini-slot 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.**

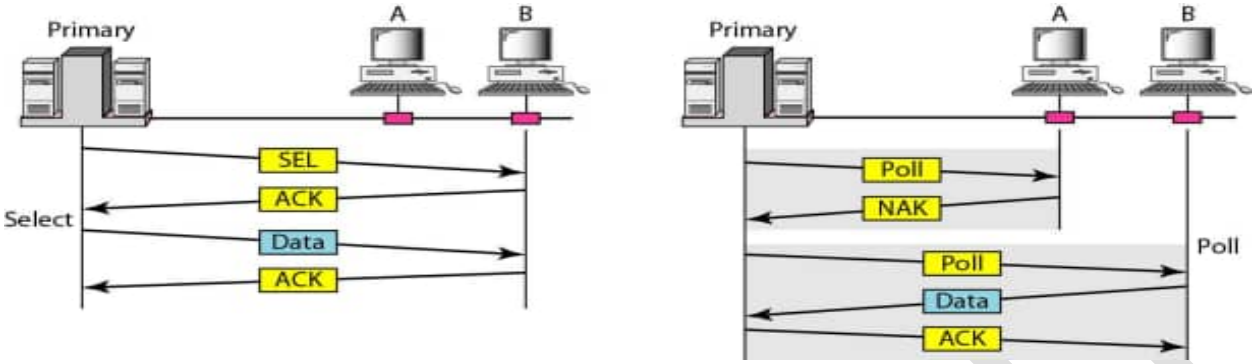
Figure 12.18



2. 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, as shown below f**igure 12.19.

Figure 12.19



- If the primary wants to receive data, it asks the secondary's if they have anything to send, this is called poll function. If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.

Select

- The *select* function is used whenever the primary device has something to send.
- 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

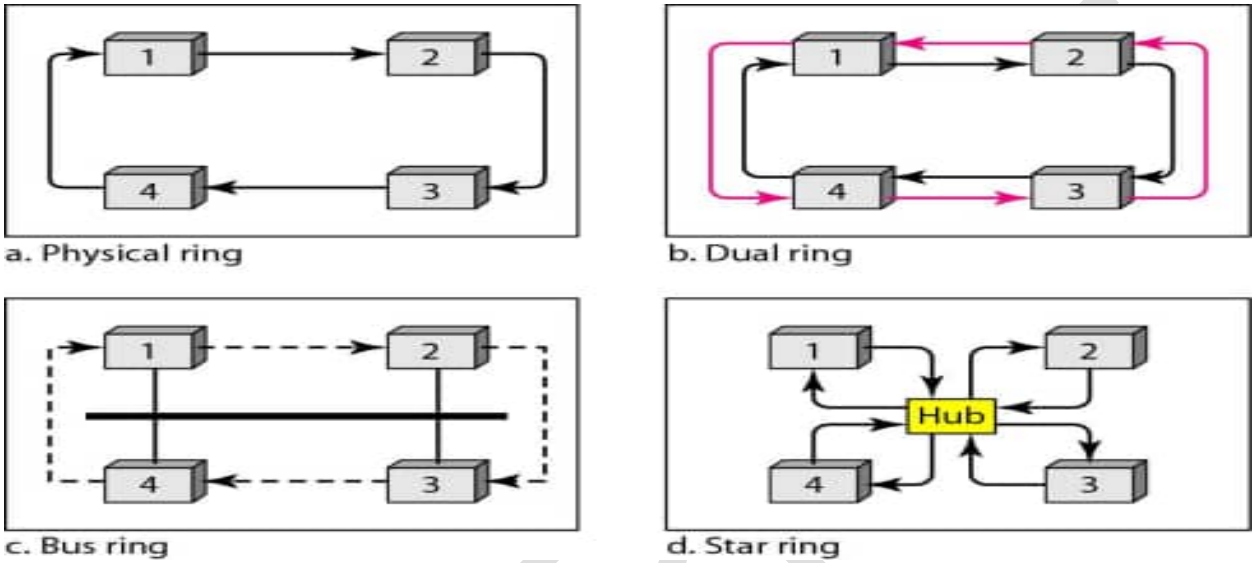
- The *poll* function is used by the primary device to solicit transmissions from the secondary devices.
- **When the primary is ready to receive data, it must ask (poll**) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

3. Token Passing

- The stations in a network are organized in a logical ring, 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.**
- Here special packet called a t**oken** circulates through the ring. The possession 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.

Logical Ring

- In a token-passing network, stations do not have to be physically connected in a ring, the ring can be a logical one. Figure below show four different physical topologies.



a. Physical ring b. Dual ring

c. Bus ring d. Star ring

- **In the physical ring topology**, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links-the medium between two adjacent stations fails, the whole system fails.
- **The dual ring topology uses a second** ring which operates in the reverse direction compared with the main ring. The **second ring is for emergencies only**. If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring, after the failed link is restored. Each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called FDDI (Fiber Distributed Data Interface) and CDDI (Copper Distributed Data Interface) use this topology.
- **In the bus ring** topology, also called a token bus, the stations are connected to a single cable called a bus. They, however, make a logical ring, because each station knows the address of its successor. When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology**.**
- **In a star ring topology**, there is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate.