UNIT 1 DATA LINK LAYER FUNDAMENTALS

1.0 INTRODUCTION

Data Link Layer (DLL) is the second layer of the OSI model which takes services from the Physical layer and provides services to the network layer. DLL transfers the data from a host to a node or a node to another node.

The main task of the data link layer is to take a raw data from transmission facility and transform it into a line that appears free of transmission errors to the network layer. Data packets are encoded and decoded into bits. These are called frames. The **Functions of the Data Link Layer** are Framing, Frame synchronisation, Error Handling Flow Regulation, Addressing and Access Control. The data link layer is divided into two sublayers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sublayer controls how a computer on the network gains access to the link resources and grants permission to transmit it. The LLC layer controls frame synchronisation, flow control and error checking.

Frame is a data structure used in transmissions at DLL consisting of a header and a trailer bracketing a data frame. Packets are the fundamental unit of information transport in all modern computer networks, and increasingly, in other communication networks as well these are converted to frame at DLL. The destination host not receiving the data bits as transmitted by the source host is termed as error in data. The error needs to be detected or corrected, as the data on the network must be error free and reliable which is one of the key features of the DLL. There are various methods used for detection and correction of these errors viz.: FEC (Forward Error Correction) and CRC (Cyclic Redundancy Check). To transmit data from a source to a destination node, a node in a network should be uniquely identified. This identification is known as the address of the node. Access control is related to addressing the problem of "Which channel or node on a LAN or WAN would transmit data?" The data link layer also deals with the problem of difference in the speed of transmission of data by the sender and receiver. Very often the speed of the receiver is slower than the speed of the sender. To overcome the same, the DLL has many flow control mechanism as part of its functionality.

1.1 OBJECTIVES

After going through this unit, you should be able to understand:

• the functionality of the Data Link Layer;

• the concept of framing and how framing is done;

- the types of errors that can be generated in frames during transmission;
- error in a data frame;
- methods for detecting errors;
- methods for correcting errors;
- forwarding the error correction method for error correction;
- following the CRC method for error detection;
- the meaning of flow control, and
- the methods for Managing Flow Control and error control.

1.2   FRAMING

As you already know, the physical layer deals with raw transmission of data in the form of bits and gives services to the data link layer. The data link layer provides services to the network layer as shown in the *Figure 1*:
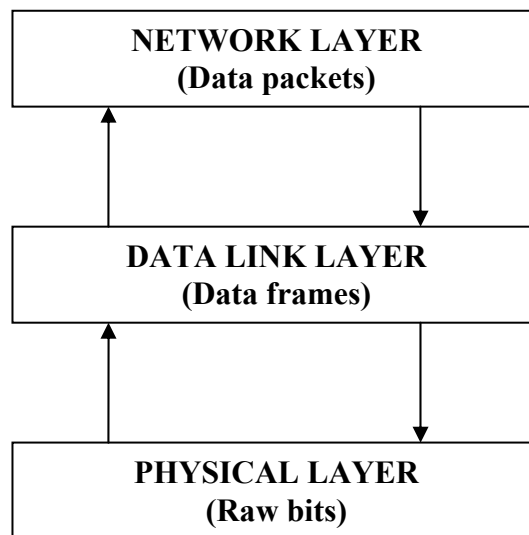

Figure 1: Data link layer providing services to network layer

The raw data coming from the Physical layer is converted into frames for forwarding to the network layer. This is done to ensure that the transmission of data is error free. Error detection and correction (if required) is done by the Data link layer, which is discussed in the following sections. The structure of the frame is shown in *Figure 2*.

Converting the bit stream into frames is a tedious process. The beginning and end of each frame should be explicitly marked. The easiest way to do so is to insert some time gap between frames.

Flag	Control Information	Data Value	Control Information	Flag

Figure 2: Frame format

But, it is difficult to keep track of counts on timing to mark the start and end time of each frame. So to overcome the same, we will discuss the following methods for framing.

- Character Count
- Character Patterns
- Bit Patterns
- Framing by Illegal code (code violation)

Character Count

The first framing method, Character count, uses a header field to specify the number of characters in the frame. The Data Link Layer at the destination checks the header field to know the size of the frame and hence, the end of frame. The process is shown in *Figure 3* for a four-frame of size 4, 5, 5 and 9 respectively.

--FRAME 1-- ------FRAME 2----   -----FRAME 3---   --------------FRAME 4---------

4	1	2	3	**5**	4	5	6	7	**5**	8	9	1	2	**9**	3	4	5	6	7	8	9	0

Figure 3: Character count

However, problems may arise due to changes in character count value during transmission. For example, in the first frame if the character count 4 changes to 6, the destination will receive data out of synchronisation and hence, it will not be able to identify the start of the next frame. This example is shown in *Figure. 4.* Even, after a request from the destination to the source for retransmission comes, it does not solve the problem because the destination does not know from where to start retransmission.

-------FRAME 1 ------   ----FRAME 2---

6	1	2	3	**5**	4	5	6	7	**5**	8	9	1	2	**9**	3	4	5	6	7	8	9	0

↑

ERROR-----DATA Received Out of Order--

Figure 4: Problems in character count

Character Patterns

The second framing method, Character stuffing, solves the problem of out of synchronisation. Here, each frame starts with special character set e.g., a special ASCII character sequence.

The frame after adding the start sequence and the end sequence is shown in *Figure 5*.

Begin each frame with DLE STX.

End each frame with DLE ETX.

DLE stands for Data Link Escape

STX stands for Start of Text

ETX stands for End of Text

DLE	STX	……………………………	DLE	ETX

Figure 5: Character patterns (Character stuffing)

If a DLE ETX occurs in the middle of the data and interferes with the data during framing then, insert an ASCII DLE character just before DLE character in the data. The Receiver interprets the single DLE as an escape indicating that the next character is a control character.

If two DLE's appear in succession at the receiver's end, the first is discarded and the second is regarded as data. Thus, framing DLE STX or DLE ETX is distinguished by whether DLE is present once or twice.

Here, the example is explained with the help of *Figures 6(a), 6(b) and 6(c).*

| DLE | STX | X | DLE | Y | DLE | ETX |

Figure 6(a) : Before stuffing

| DLE | STX | X | DLE | DLE | Y | DLE | ETX |

Figure 6(b) : After stuffing

| DLE | STX | X | DLE | DLE | Y | DLE | ETX |

Figure 6(c) : After destuffing

A problem with character stuffing is that not all bit streams are character oriented (e.g., Unicode is a 16-bit code). Hence, for arbitrary sized characters the process of character stuffing becomes more complex. So next we will discuss a new method known as the bit stuffing, which solves the problem of arbitrary sized character.

Bit Patterns

This method is similar to the one discussed above, except that, the method of bit stuffing allows insertion of bits instead of the entire character (8 bits). Bit pattern framing uses a particular sequence of bits called a ***flag*** for framing. The flag is set as the start and the end of the frame.

Use of ***bit patterns*** is to keep the sequence of data in the same order.

Flag = 0 1 1 1 1 1 1 0 (begins and ends frame.)

In this case the transmitter automatically inserts a 0 after 5 consecutive 1's in the data. This is called Bit stuffing. The receiver discards these stuffed 0's as soon as it encounters 5 consecutive 1's in the received data as shown with the help of an example described in *Figure 7 (a), (b) and (c).*

11111111110010011

Figure 7(a) : Original data ready to be sent

01111110111110111110001001101111110

Figure 7(b) : Data after adding Flag and stuffing

11111111110010011

Figure 7(c) : Data after destuffing

Framing by Illegal Code (Code violation)

A fourth method is based on any redundancy in the coding scheme. In this method, we simply identify an illegal bit pattern, and use it as a beginning or end marker, i.e., certain physical layers use a line code for timing reasons.

For example: Manchester Encoding

1 –It can be coded into two parts i.e., high to low ⌐⌐ ≡ 1 0

0 – It can be coded into two parts i.e., low to high ⌐⌐ ≡ 0 1

Codes of all low (000) or all high (111) aren't used for the data and therefore, can be used for framing.

☞ **Check Your Progress 1**

1)   Name different framing methods.

………………………………………………………………………………
………………………………………………………………………………
………………………………………………………………………………

2)   Write the bit sequence after bit stuffing for the data stream
        110001111111100001111100

………………………………………………………………………………
………………………………………………………………………………
………………………………………………………………………………

3)   Why bit stuffing is advantageous over character stuffing?

………………………………………………………………………………
………………………………………………………………………………
………………………………………………………………………………

1.3   BASICS OF ERROR DETECTION

The Network should ensure complete and accurate delivery of data from the source node to destination node. But many times data gets corrupted during transmission. As already discussed in the previous block, many factors can corrupt or alter the data that leads to an error. A reliable system should have methods to detect and correct the errors. Firstly, we will discuss what the error could be then, in the later section we will discuss the process of detecting and correcting them.

Types of Error

Several types of error may occur during transmission over the network:

- 1-bit error
- burst error
- lost message (frame)

1-bit error: 1-bit error/Single bit error means that only one bit is changed in the data during transmission from the source to the destination node i.e., either 0 is changed to 1 or 1 is changed to 0 as shown in *Figure 8*.
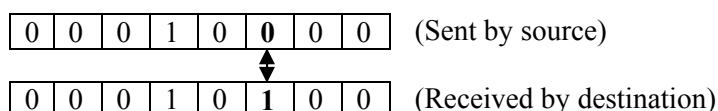
| 0 | 0 | 0 | 1 | 0 | **0** | 0 | 0 | (Sent by source)

| 0 | 0 | 0 | 1 | 0 | **1** | 0 | 0 | (Received by destination)

Figure 8: 1 bit error

This error will not appear generally in case of serial transmission. But it might appear in case of parallel transmission.

Burst error: Burst error means that 2 or more bits of data are altered during transmission from the source to the destination node. But, it is not necessary that error will appear in consecutive bits. Size of burst error is from the first corrupted bit to the last corrupted bit as shown in *Figure 9*.
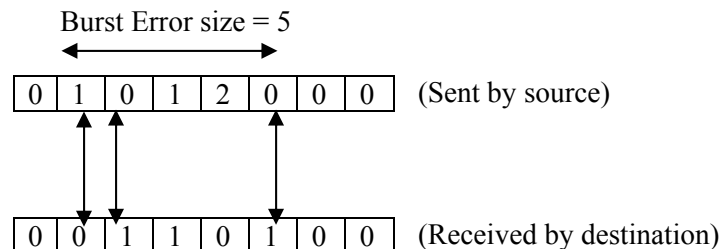
Burst Error size = 5

| 0 | 1 | 0 | 1 | 2 | 0 | 0 | 0 | (Sent by source)

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | (Received by destination)

Figure 9: Burst error

An *n*-bit burst error is a string of bits inverted during transmission. This error will hardly occur in case of parallel transmission. But, it is difficult to deal with all corrupted bits at one instance.

Lost Message (Frame): The sender has sent the frame but that is not received properly, this is known as loss of frame during transmission. To deal with this type of error, a retransmission of the sent frame is required by the sender. We will discuss retransmission strategies in the next unit.

Now we will discuss some methods for error detection.

Error Detection

As already discussed in the beginning of this section accurate delivery of data at the receiver's site is, one of the important goals of this layer. This implies that the receivers should get the data that is error free. However, due to some factors if, the data gets corrupted, we need to correct it using various techniques. So, we require error detection methods first to detect the errors in the data before correcting it. Error detection is an easy process.

For error detection the sender can send every data unit twice and the receiver will do bit by bit comparison between the two sets of information. Any alteration found after the comparison will, indicate an error and a suitable method can be applied to correct the error.

But, sending every data unit twice increases the transmission time as well as overhead in comparison. Hence, the basic strategy for dealing with errors is to include groups of bits as additional information in each transmitted frame, so that, the receiver can detect the presence of errors. This method is called Redundancy as extra bits appended in each frame are redundant. At the receiver end these extra bits will be discarded when the accuracy of data is confirmed.

For example:

Source node (Sender)

0	0	0	1	0	0	0	0

0	0	0	1	0	0	0	0	1	1	1

Destination node (Receiver)

0	0	0	1	0	0	0	0	1	1	1

0	0	0	1	0	0	0	0

Redundancy check methods commonly used in data transmission are:

- Parity check
- CRC
- Checksum

Parity Check

- The most common method used for detecting errors when the number of bits in the data is small, is the use of the ***parity bit.***

- A *parity bit* is an extra binary digit added to the group of data bits, so that, the total number of one's in the group is even or odd.

- Data bits in each frame is inspected prior to transmission and an extra bit (the parity bit) is computed and appended to the bit string to ensure even or odd parity, depending on the protocol being used.

- If odd parity is being used, the receiver expects to receive a block of data with an odd number of 1's.

- For even parity, the number of 1's should be even.

In the example below, even parity is used. The ninth column contains the parity bit.

```
0 1 0 1 0 1 0 1 0
0 1 1 1 1 0 0 1 1
1 1 1 1 0 0 1 1 0
```

- Use of parity bits a rather weak mechanism for detecting errors.
- A single parity bit can only detect single-bit errors.

Block Sum Check Method

- This is an extension to the single parity bit method. This can be used to detect up to two bit errors in a block of characters.
- Each byte in the frame is assigned a parity bit (*row parity*).
- An extra bit is computed for each bit position (*column parity*).
- The resulting set of parity bits for each column is called the *block sum check.* Each bit that makes up the character is the modulo-2 sum of all the bits in the corresponding column.

Block Sum Check, *example*

Sender's data: 0000100   0010101   0101011

```
1  0 0 0 0 1 0 0
1  0 0 1 0 1 0 1
0  0 1 0 1 0 1 1
   _____
0  0 1 1 1 0 1 0
```

Data after adding parity bits:

00001000   00101010   01010110   01110100

This method detects single bit errors as well as increases the probability of finding burst error.

CRC

Cyclic Redundancy Check (CRC) **is used to detect burst errors**

In this method, you would need to treat strings of bits as coefficients of a polynomial code that uses modulo 2 arithmetic. In modulo 2 arithmetic there are no carriers for addition and borrows for subtraction. Polynomial codes treat bit strings as representative of polynomials with coefficients of 0 and 1 only.

For example, the bit sequence 1 0 01 0 1 is represented by the polynomial $x^5 + x^2 + 1$ ($1.x^5 + 0.x^4 + 0.x^3 + 1.x^2 + 0.x^2 + 1.x^0$). When the polynomial method is employed, the sender and the receiver must agree upon a generator polynomial both the high and low order bits of the generator must be 1.

In this method the Sender divides frame (data string) by a predetermined Generator Polynomial and then appends the remainder (called checksum) onto the frame before starting the process of transmission. At the receiver end, the receiver divides the received frame by the same Generator polynomial. If the remainder obtained after the division is zero, it ensure that data received at the receiver's end is error free. All operations are done modulo 2.

An example that explains finding CRC (Cyclic Redundancy Check) will follow later.

Checksum

In this method the checksum generator divides the given input data into equal segments of k bits(8 or 16). The addition of these segments using ones complement arithmetic is complimented. This result is known as the checksum and it is appended with the data stream. This appended data stream is transmitted across the network on the transmission media. At the receiver end add all received segments. If the addition of segments at the receiver end is all 1's then, the data received is error free as, complement of the same will be all 0's. Then the data can be accepted, otherwise, data can be discarded.

For example:

Sender's data   0 0 0 0 0 0 1 0   0 1 0 1 0 0 0 0

```
              00000010
              01010000
              _____
   Sum        01010010
```

Checksum (Compliment)   10101101

Data with appended checksum: 00000010 01010000 10101101

Receiver's accept the data as

> **00000010 01010000 10101101**

At receiver's end

> **00000010**
> **01010000**
> **10101101**

sum   **11111111**

Complement   00000000

As complement is 0 it indicates that the data received at the receiver's end is error free so, it will be accepted by the receiver.

The checksum method detects all errors as it retains all its carries.

- Odd number of bits
- Most of even number of bits.

☞ **Check Your Progress 2**

1) Define parity bit? Also define even and odd parity?

   ………………………………………………………………………………….
   ………………………………………………………………………………….
   …………………………………………………………………………………

2) Name the method that can detect single bit error and burst error.

   ………………………………………………………………………………….
   ………………………………………………………………………………….
   ………………………………………………………………………………….

3) Define checksum.

   ………………………………………………………………………………….
   ………………………………………………………………………………….
   …………………………………………………………………………………

1.4   FORWARD ERROR CORRECTION

In the previous section, we have discussed detection of error that appears after the transmission of data over the network from sender to receiver. In this section, we will study how to perform correction of errors. Forward Error Correction (FEC) is a type of error correction method which improves on simple error detection schemes by enabling the receiver to correct errors once they are detected. This reduces the need for retransmissions of error frames.

Firstly we consider the simple case i.e., correcting single bit error. As we have discussed earlier that a single bit error can be detected by adding one additional bit (parity bit/ redundant bit). This additional bit can detect error in any bit stream by differentiating the two condition error or not error as a bit can have two states only i.e., 0 and 1.

For correction of detected single bit error two states are not sufficient. As an error occurs in bit stream indicates that one bit is altered from either 0 to 1 or 1 to 0. To correct the same, conversion of altered bit is required. For performing this conversion we must know the location of bit which is in error. Therefore, for error correction identification of location of error bit is required. For example, for applying error correction of single bit error in ASCII character we must find which of 7 bit is altered. For doing this we could have eight different states i.e., no error, error in bit position 1, error in bit position 2 up to error in bit position7. For this we need many redundant bits to represent all eight states.

Here, 3 bit redundancy code can represent all possible eight states because 3 bits can represent 8 states (000 to 111). But if an error occurs in redundancy bit then we need 3 Additional bits added with 7 ASCII character bits, it covers all possible error locations.

So we can generalise if we have n data bits and r redundancy bits then total n+r will be the transmittable bits and r bits must be able to represent at least n+r+1 different states, here plus 1 indicates no error state. Hence n+r+1 states are identifiable by r additional bits and r additional bits represents 2^r different states.

- $2^r >= n+r+1$

For example, in 7 bit ASCII character we need at least 4 redundant bits as $2^4>=7+4+1$.

Hamming code is one such error correcting code.

For the example discussed above for a 7 bit ASCII character and 4 redundant bit, we will have total bits as n+r i.e., 7+4 = 11. These 4 redundant bits can be inserted in 7 bit data stream in position 1,2,4 and 8 (in 11 bit sequence at $2^0,2^1,2^2,2^3$) named as r1,r2,r4 and r8 respectively.

In Hamming code each redundant bit is the combination of data bits where each data bit can be included in more than one combination as shown below.

r1 : 1,3,5,7,9,11
r2 : 2,3,6,7,10,11
r3 : 4,5,6,7
r8 : 8,9,10,11

Now we, will find the values of redundant bit r1,r2, r4 and r8 for the data bit sequence 1010101 as shown in following *Figure*.

11	10	9	8	7	6	5	4	3	2	1
1	0	1	r8	0	1	0	r4	1	r2	r1

For finding values of r1, r2, r4 and r8 we will find even parities for various bit combination. The parity bit will be the value of redundant bit.

1	0	1	r8	0	1	0	r4	1	r2	1

1	0	1	r8	0	1	0	r4	1	1	1

1	0	1	r8	0	1	0	1	1	1	1

1	0	1	0	0	1	0	1	1	1	1

Assume that during transmission the number 5 bit is altered from 0 to 1. So at receiver end for error detection and correction new parities will be calculated as

Error   new parity

1	0	1	0	0	1	1	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	0
1	0	1	0	0	1	1	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	0

0101 = 5. It implies 5th bit is the error bit. The binary number obtained from new parties will indicate the error bit location in the received data stream. Subsequently that bit can be altered and data can be corrected.

 If all the values in the new parity column are 0, we conclude that the data is error free. In the given example if no error it should be 0000.

☞ **Check Your Progress 3**

1)   Define Hamming distance.

……………………………………………………………………………………….
……...…………………………………………………………………………….
……..………………………………………………………………………….

2)   Generate the Hamming code for the following input data
       101111

……………………………………………………………………………………..
……...……………………………………………………………………………..
……..…………………………………………………………………………….

1.5   CYCLIC REDUNDANCY CHECK CODES FOR ERROR DETECTION

The most commonly used method for detecting burst error in the data stream is Cyclic Redundancy Check Method. This method is based on the use of *polynomial codes.* Polynomial codes are based on representing bit strings as polynomials with coefficients as 0 and 1 only.

For example, the bit string 1110011 can be represented by the following polynomial:

$$1.x^6 + 1.x^5 + 1.x^4 + 0.x^3 + 0.x^2 + 1.x^1 + 1.x^0$$

This is equivalent to:

$$x^6 + x^5 + x^4 + x^1 + 1.$$

- The polynomial is manipulated using *modulo 2* arithmetic (which is equivalent to Exclusive OR or XOR).

- Depending on the content of the frame a set of check digits is computed for each frame that is to be transmitted. Then the receiver performs the same arithmetic as the sender on the frame and check the digits. If the result after computation is the same then the data received is error free.

- A different answer after the computation by the receiver indicates that, some error is present in the data.

- The computed check digits are called the frame check sequence (FCS) or the cyclic redundancy check (CRC).

The CRC method requires that:

- The sender and receiver should agree upon a generator polynomial before the transmission process start.

- Both the high and low bits of the generator must be 1.

- To compute the checksum for a frame with *m* bits, the size of the frame must be longer than the generator polynomial.

Algorithm for Computing the Checksum is as follows as:

Let D(x) be the data and G(x) be the generating polynomial. Let r be the degree of generator polynomial G(x).

Step 1: Multiple the data D(x) by x^r, giving *r* zeros in the low-order end of the frame.

Step 2: Divide the result obtained in step1 by G(x), using modulo-2 division.

Step 3: Append the remainder from step 2 to D(x), thus, placing *r* terms in the *r* low-order positions.

The type of generating polynomial is important for finding the types of error that are detected.

- A generator polynomial of R bits will detect:
  - all single-bit errors
  - all double-bit errors
  - all odd-number of bit errors
  - all burst errors < R
  - most burst errors >= R

Example:

Data 1011101
Generator Polynomial G(x): $x^4+x^2+1 = 10101$
Here the size of the generator polynomial is 5 bit long, so, we will place $5 - 1 = 4$ 0's in the low order data stream. So the data will be:

Data 1011101 0000

Now using modulo2 division, (for getting data ready to be transmitted), we will divide the data by the generator polynomial as shown in the *Figure 10*.
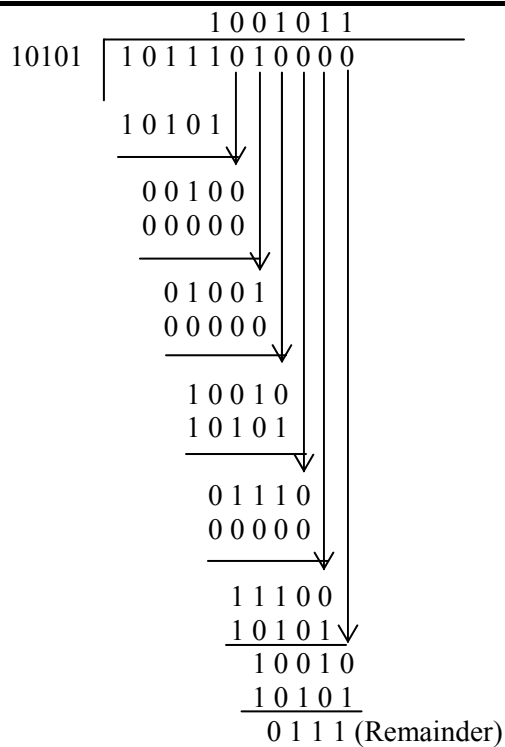
Division diagram at sender's site

```
                   1 0 0 1 0 1 1
    10101 | 1 0 1 1 1 0 1 0 0 0 0
             1 0 1 0 1
             0 0 1 0 0
             0 0 0 0 0
               0 1 0 0 1
               0 0 0 0 0
                 1 0 0 1 0
                 1 0 1 0 1
                   0 1 1 1 0
                   0 0 0 0 0
                     1 1 1 0 0
                     1 0 1 0 1
                       1 0 0 1 0
                       1 0 1 0 1
                         0 1 1 1 (Remainder)
```

Figure 10: Cyclic redundancy check

The remainder obtained after the division of 0111, will be placed on the low order bits of the data stream. So the Data that is transmitted over the network by the sender is $D(x) = 1011101\ 0111$

Now at the receiver's end, the receiver receives $D(x)$ as data which will be divided by the generator polynomial as shown in *Figure 11*.

Division diagram at receiver's site

```
                   1 0 0 1 0 1 1
    10101 | 1 0 1 1 1 0 1 0 1 1 1
             1 0 1 0 1
             0 0 1 0 0
             0 0 0 0 0
               0 1 0 0 1
               0 0 0 0 0
                 1 0 0 1 0
                 1 0 1 0 1
                   0 1 1 1 1
                   0 0 0 0 0
                     1 1 1 1 1
                     1 0 1 0 1
                       1 0 1 0 1
                       1 0 1 0 1
                         ─────────
                         0 0 0 0
```
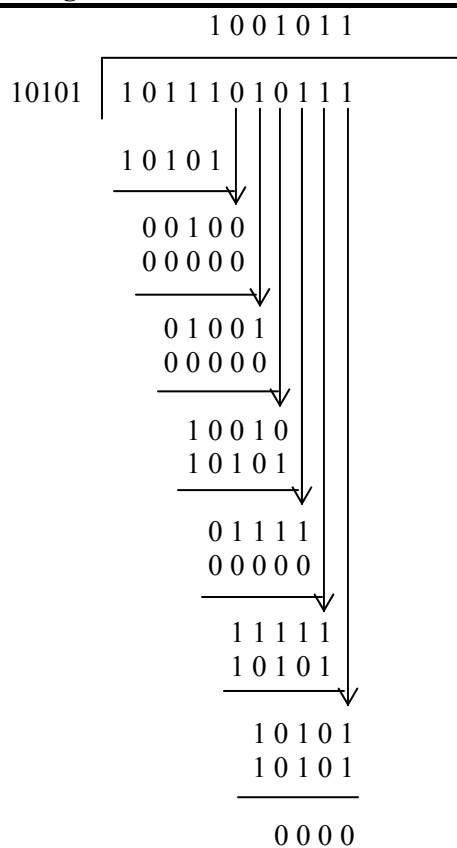
Figure 11: Cyclic redundancy check at the receiver side

Here the remainder obtained after division is 0000, so it ensure that data received at the receiver end is error free otherwise, it indicates that the data has some error in it. In this way the CRC method can detect whether the data has some error or is error free.

☞ **Check Your Progress 4**

1) Find the CRC for the data polynomial x^4+x^2+x+1 where generator polynomial is x^3+1.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

1.6 FLOW CONTROL

Another important issue for the data link layer is dealing with the situation which occurs when the sender transmits frames faster than the receiver can accept or process them. If the sender is working on a fast machine and the receiver is working on a slow machine this situation may occur in the network. In this process of transmission, some of the frames might be lost as they were not processed by the receiver due to it's low speed, while the sender might have through the transmission to be completely error free. To prevent this situation during transmission, a method is introduced called the *Flow Control.*

Flow control means using some feedback mechanism by which, the sender can be aware of when to the send next frame, and not at the usual speed of the sender. If the frame is accepted /processed by the receiver then only with the sender send the next frame. It may be said that the speed of the sender and the receiver should be compatible with each other, so that the receiver will receive or process all frames sent by the sender as every receiver has a limited block of memory called the buffer, reserved for storing incoming frames.

There are several methods available for deciding when a sender should send one frame or the next frame. Flow control ensures that the speed of sending the frame, by the sender, and the speed of processing the received frame by the receiver are compatible.

There are two basic strategies for flow control:

1) Stop-and-wait

2) Sliding window.

We shall start with the assumption that the transmission is error free and that we have an ideal channel.

Stop and Wait

Stop-and-Wait Flow Control (*Figure 12*) is the simplest form of flow control. In this method, the receiver indicates its readiness to receive data for each frame, the message is broken into multiple frames. The Sender waits for an ACK(acknowledgement) after every frame for a specified time (called time out). It is sent to ensure that the receiver has received the frame correctly. It will then send the next frame only after the ACK has been received.

Operations:

1) **Sender:** Transmits a single frame at a time.
2) **Receiver:** Transmits acknowledgement (ACK) as it receives a frame.
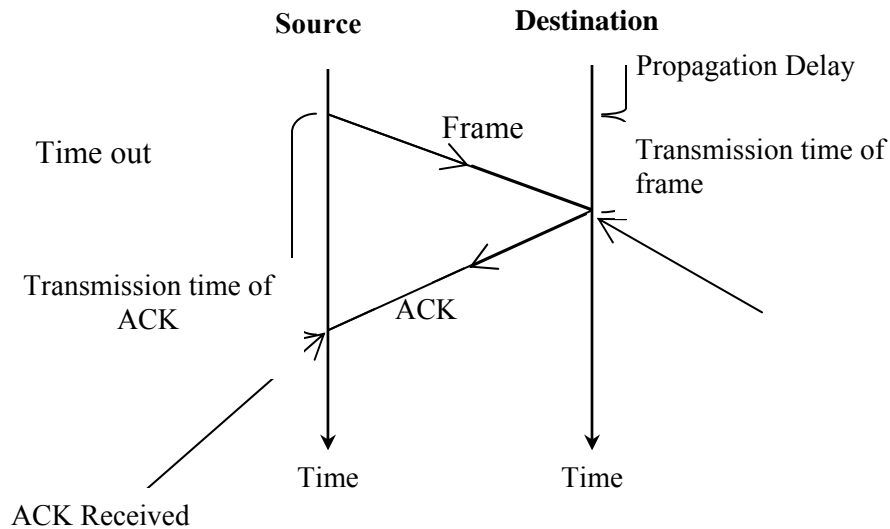3) Sender receives ACK within time out.
4) Go to step 1.

Figure 12: Stop and Wait

If a frame or ACK is lost during transmission then it has to be transmitted again by the sender. This retransmission process is known as ARQ (automatic repeat request). Stop and Wait ARQ is one of the simplest methods of flow control. It will be further discussed in detail in the next unit.

The problem with stop and wait is that only one frame can be transmitted at a time and that often leads to inefficient transmission channel till we get the acknowledgement the sender can not transmit any new packet. During this time both the sender and the channel are unutilised.

To deal with this problem, there is another flow control method i.e., sliding window protocol which is discussed below:

Sliding Window Protocol

In this flow control method, the receiver allocates buffer space for *n* frames in advance and allows transmission of multiple frames.This method allows the sender to transmit *n* frames without an ACK. A k-bit sequence number is assigned to each frame. The range of sequence number uses modulo-2 arithmetic. To keep track of the frames that have been acknowledged, each ACK has a *sequence* number. The receiver acknowledges a frame by sending an ACK that includes the Sequence number of the **next expected frame.** The sender sends the next *n* frames **starting with** the last received sequence number that has been transmitted by the receiver (ACK). Hence, a single ACK can acknowledge multiple frames as shown in the *Figure 13.*

The receiver receives frames 1, 2 and 3. Once frame 3 arrives ACK4 is sent to the sender. This ACK4 acknowledge the receipt of frame 1, 2 and 3 and informs the sender that the next expected frame is frame 4. Therefore, the sender can send multiple back-to-back frames, making efficient use of the channel.
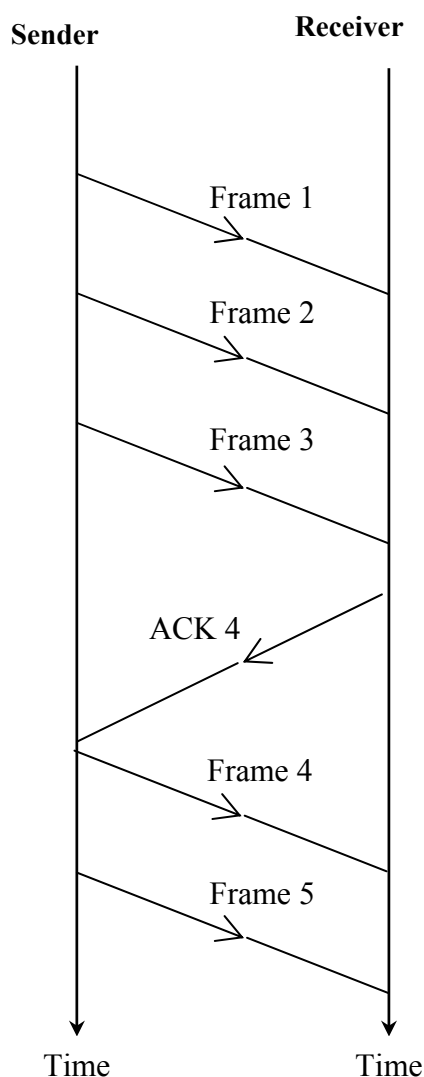
Normal Flow diagram of a sliding window

Sender · Receiver

Frame 1

Frame 2

Frame 3

ACK 4

Frame 4

Frame 5

Time · Time

Figure 13: Normal flow diagram of sliding

Sequence number is a field in the frame that is of finite size. If k bits are reserved for the sequence number, then the values of sequence number ranges from 0 to $2^k - 1$ (Modulo Arithmetic).

Operation of a Sliding Window

Sending Window:

In this mechanism we maintain two types of windows (buffer) sending window and receiving windows as shown in *Figure 14 & 15*.

At any instant, the sender is permitted to send frames with sequence numbers in a certain range (3-bit sending window) as shown in *Figure 14*.
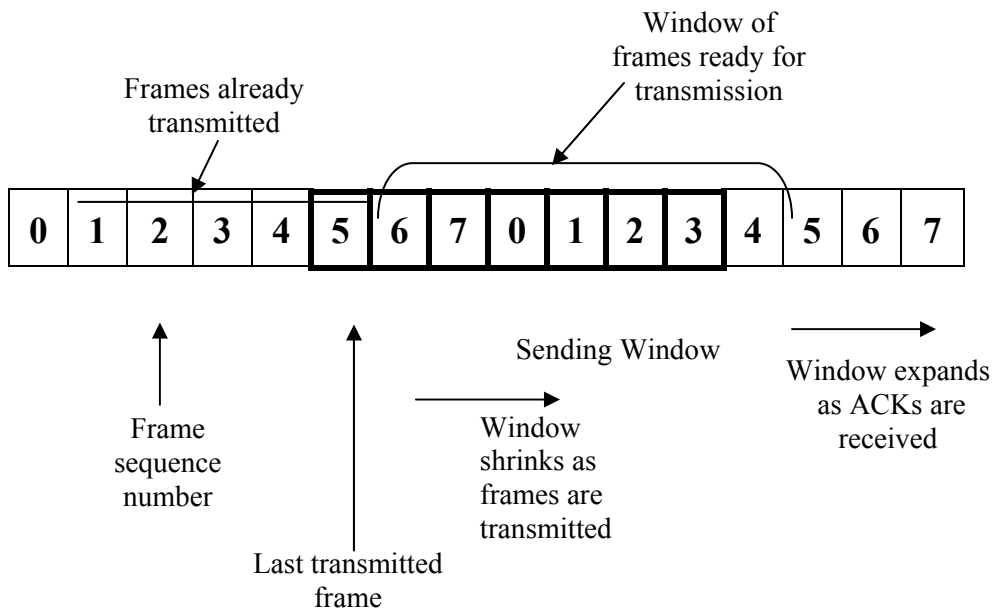
Frames already
transmitted

Window of
frames ready for
transmission

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

Frame
sequence
number

Sending Window

Window
shrinks as
frames are
transmitted

Window expands
as ACKs are
received

Last transmitted
frame

Figure 14: Sending window in a sliding window

Receiving Window

The receiver maintains a receiving window depending on the sequence numbers of
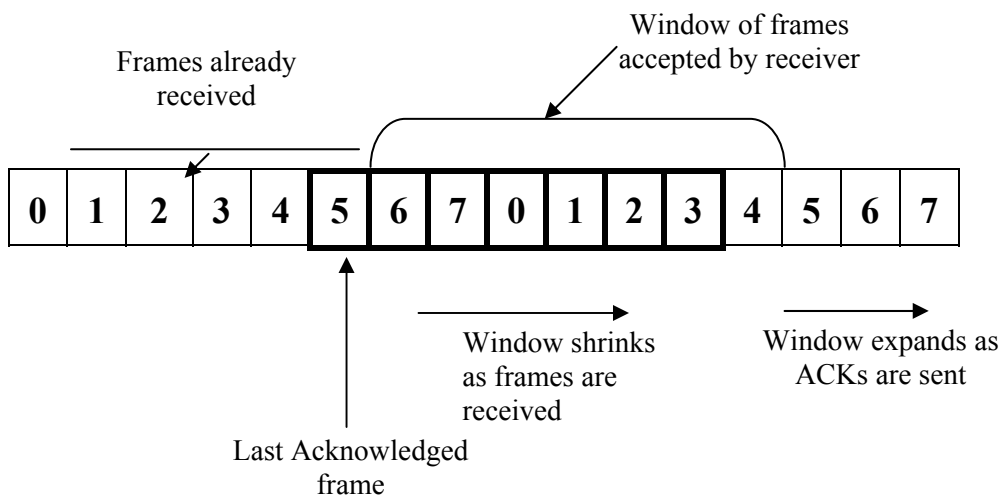frames that are accepted as shown in *Figure 15*.

Frames already
received

Window of frames
accepted by receiver

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

Window shrinks
as frames are
received

Window expands as
ACKs are sent

Last Acknowledged
frame

Figure 15: Receiving window in a sliding window

Functioning of Sliding Window

Flow control is achieved as the receiver can control the size of the sending window by
limiting the size of the sending window. Similarly, data flow from the sender to the
receiver can be limited, and that too can controls the size of receiving window as
explained with the help of an example in *Figure 16*.
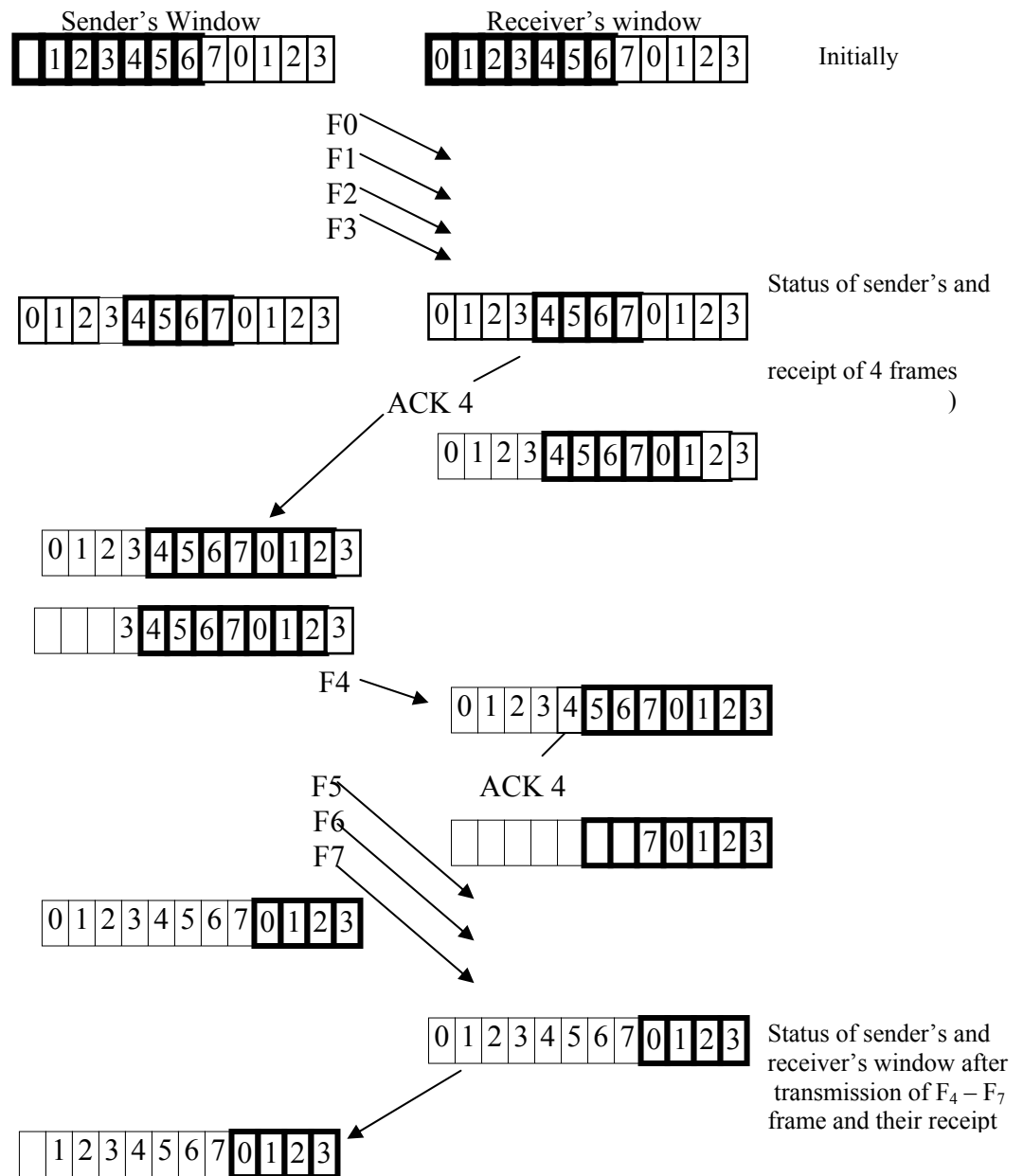
Example:

Figure 16: Function of a sliding window machine

Two types of errors are possible while transmission from source to destination takes place

1) Lost frame: Frame (data or control) fails to arrive.

2) Damaged frame: Frame is recognised, but some bits have been changed.

Two commonly used methods for sliding window are:

1) Go-back-n ARQ

2) Selective-repeat ARQ

Details of these two ARQ requests are discussed in the next unit.

☞ **Check Your Progress 5**

1)   How does the sliding window protocol increase the utilisation of bandwidth?

…………………………………………………………………………………………

…………………………………………………………………………………………

………………………………………………………………………………………….

2)   Define ARQ. How does ARQ facilitate the error control process?

…………………………………………………………………………………………

…………………………………………………………………………………………

………………………………………………………………………………………….

1.7   SUMMARY

This unit introduced the basic fundamental issues related to the Data Link Layer. The concept of framing and different methods of framing like Character Count, Character Stuffing, Bit stuffing and Framing by Illegal code has been focused up on. In the Data Link layer, data flow and error control is discussed. This error and flow control is required in the system for reliable delivery of data in the network. For the same, different types of error, different methods for error detection and correction are discussed. Among various methods, Block sum check method detects burst of error with high probability. CRC method is the one which detects the error with simplicity. Forward Error Correction is the error correction method that uses the parity concept. For flow control, stop and wait method tries to ensure that the speed of the sender and the receivers are matching to an extent. To overcome this sliding window protocol is introduced. Sliding window protocol can send many frames at one instance and that increases the efficiency of transmission channel. If some error occurs in the data then retransmission of the error frame is required and it is known as ARQ.

1.8   SOLUTIONS/ANSWERS

Check Your Progress 1

1)   Character Count, Character Stuffing, Bit Stuffing and Framing by Illegal Code

2)   11000111110111000011111000

3)   Only 1 bit used as stuffed bit in bit stuffing instead of character (8 bits) in character stuffing.

Check Your Progress 2

1)   The extra bit added to the data bits to make number of 1's even or odd.
     Even parity: The number of 1's in data bits including the parity bit
     should be even.

     Odd Parity: The number of 1's in data bits including the parity bit
     should be odd.

2)   Parity bit method is used for detecting single bit error. And CRC method
     is used to detect burst error.

3) Checksum is the value that is used for error detection. It is generated by adding data units using 1's compliment and then complementing the result (modulo 2 arithmetic).

Check Your Progress 3

1) A method that adds redundant bits to the data stream for error detection and correction is called Hamming Code.

2) $r_1 = 0$ $r_2 = 0$ $r_4 = 0$ $r_8 = 1$

Check Your Progress 4

1) CRC- 101

Check Your Progress 5

1) Sliding Window protocol increases the utilization of bandwidth as we can transmit many frames without waiting for an acknowledgement.

2) An error control method in which correction is made by retransmission of data by. ARQ facilitate error control process by using any of the following method:

Stop and Wait ARQ
Go-Back N ARQ
Selective Repeat ARQ

1.9   FURTHER READINGS

1) *Computer Networks,* Andrew S. Tanenbaum, 4th Edition, Prentice Hall of India, New Delhi.

2) *Data and Computer Communications,* William Stalling, 5th Edition, Pearson Education, New Delhi.

3) *Data Communications and Networking,* Behrouz A. Forouzan, 3rd Edition, Tata McGraw Hill, New Delhi.

4) *Communication Networks− Fundamental Concepts and Key Architectures,* Leon Garcia and Widjaja, 3rd Edition, Tata McGraw Hill, New Delhi.