Republic of Iraq
Middle Technical University
Technical Engineering College of Artificial Intelligence
Cybersecurity Engineering Technology Department

Computer Network Fundamentals

Chapter Four

Physical Addressing & Error Detection

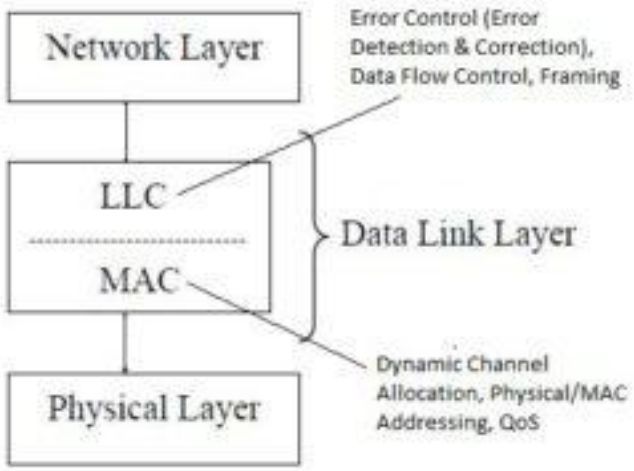
((Data Link Layer))

Prepared By:
Associate Prof. Dr. Mahmoud Shuker Mahmoud
mahmoud.shukur@mtu.edu.iq

4.1- Introduction

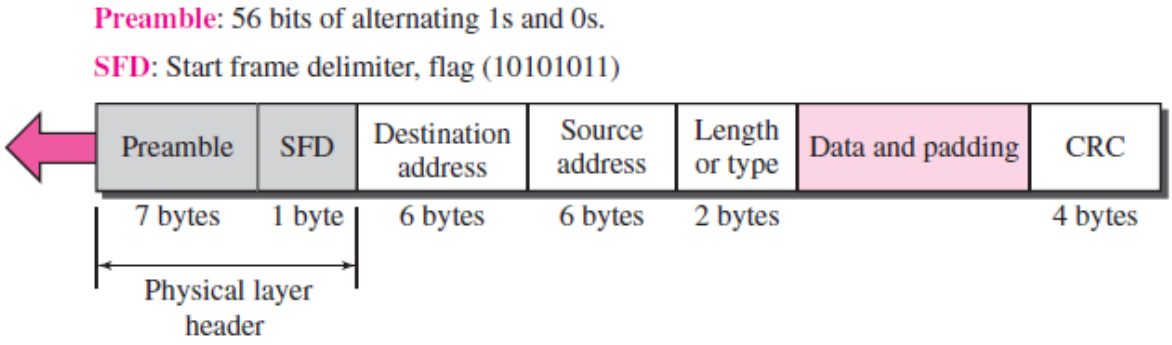
The data link layer is structured into two sublayers: the **Media Access Control (MAC) sublayer** and the **Logical Link Control (LLC) sublayer**.

The MAC sublayer controls access to the physical network medium using MAC addresses, handling data encapsulation and managing access to avoid collisions. The LLC sublayer provides flow control, error detection, and synchronization, and determines the protocol being used.



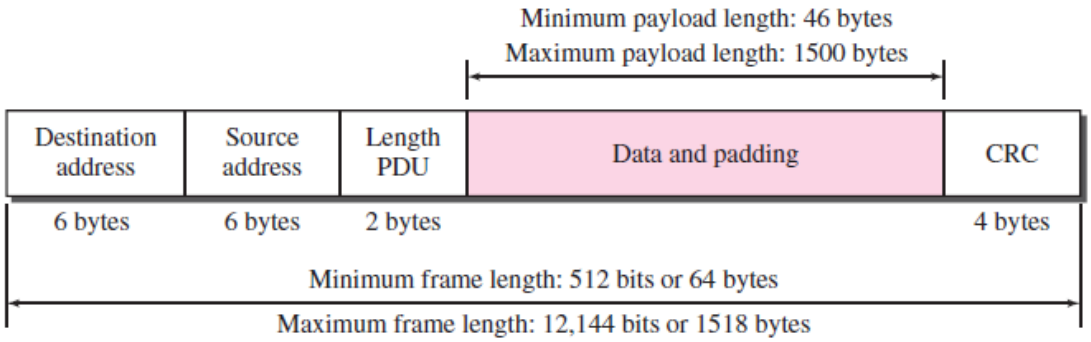
4.2- Frame Format

The Ethernet frame contains *seven fields*: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRC. The format of the MAC frame is shown in Figure below.



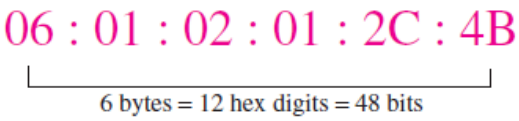
- **Preamble:** The first field of the frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The preamble is actually added at the physical layer and is not (formally) part of the frame.
- **Start frame delimiter (SFD):** The second field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.
- **Destination address (DA):** The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.
- **Source address (SA):** The SA field is also 6 bytes and contains the physical address of the sender of the packet.
- **Length or type:** This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.
- **Data:** This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.
- **CRC:** The last field contains error detection information.

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in Figure below.



4.3- Physical Addressing

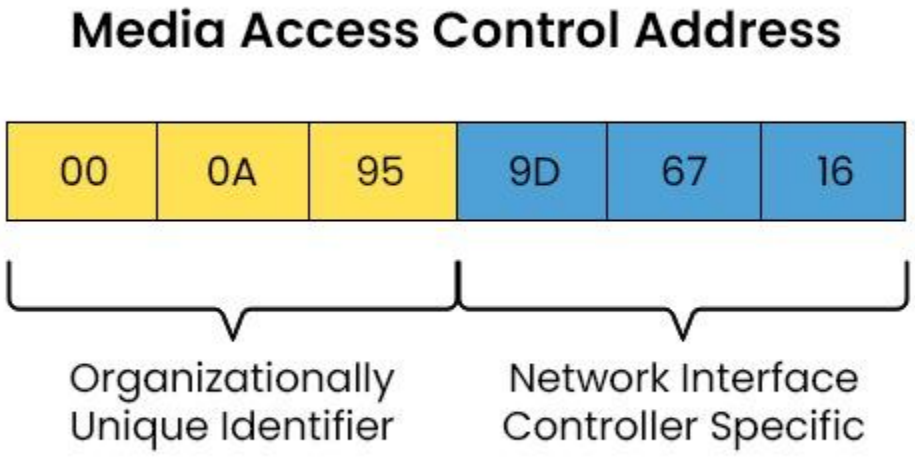
Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a 6-byte physical address (MAC), normally written in hexadecimal notation.



A MAC address is a 48-bit hardware identifier, represented as 12 hexadecimal digits, that is divided into two 24-bit halves.

- The first half, the **Organizationally Unique Identifier (OUI)**, identifies the manufacturer.
- The second half is a **unique device identifier** assigned by the manufacturer to the NIC.

These are typically displayed as six groups of two digits, separated by colons or hyphens (e.g., 00:0A:95:9D:67:16 **or** 00-0A-95-9D-67-16).



4.4- Error Handling

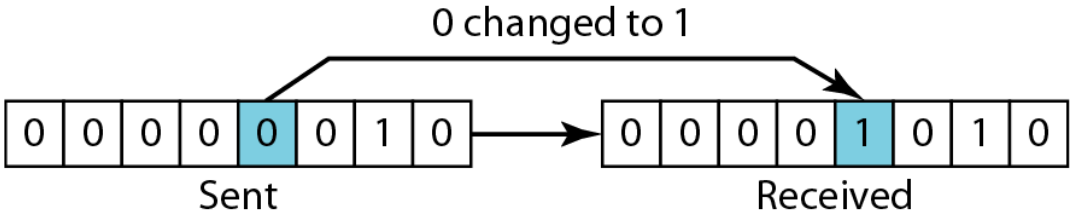
Networks must be able to transfer data from one device to another with *complete accuracy*. Yet any time data are transmitted from source to destination, they can become corrupted in passage. In fact, it is more likely that some part of a message will be altered in transit than that the entire contents will arrive intact. Reliable systems must have a mechanism for detection, and correcting such errors.

4.4.1- Types of Errors

Whenever an electromagnetic signal flows from one point to another, it is subject to *unpredictable interference from heat, magnetism, and other forms of electricity*. This interference can change the **shape** or **timing** of the signal. Such changes can alter the meaning of the data. Changing 0 to 1 or 1 to 0. Bits can be changed singly or in clumps. So it is important to understand three types of errors. These types are usually referred to as ***single-bit***, ***multiple-bit***, and ***burst errors***. Of the three, a single-bit, error is the most likely to occur and a burst error the least likely.

a) Single Bit Error

The term single-bit error means that only one bit of a given data unit (such as a byte, character, data unit or packet) is changed from 1 to 0 or from 0 to 1.

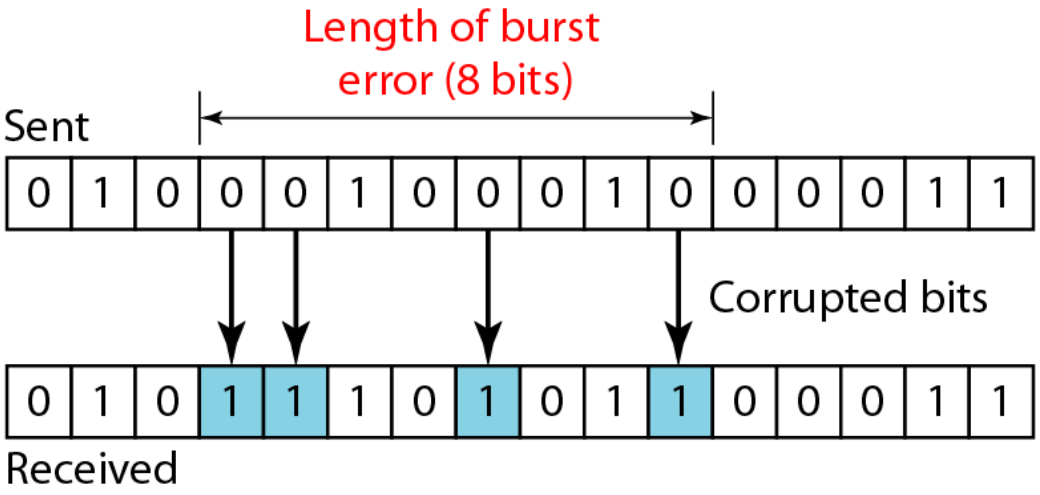


b) Multiple-Bit Error

The term multiple-bit error means that two or more non-consecutive bits in a data unit have changed from 1 to 0 or from 0 to1.

c) Burst Error

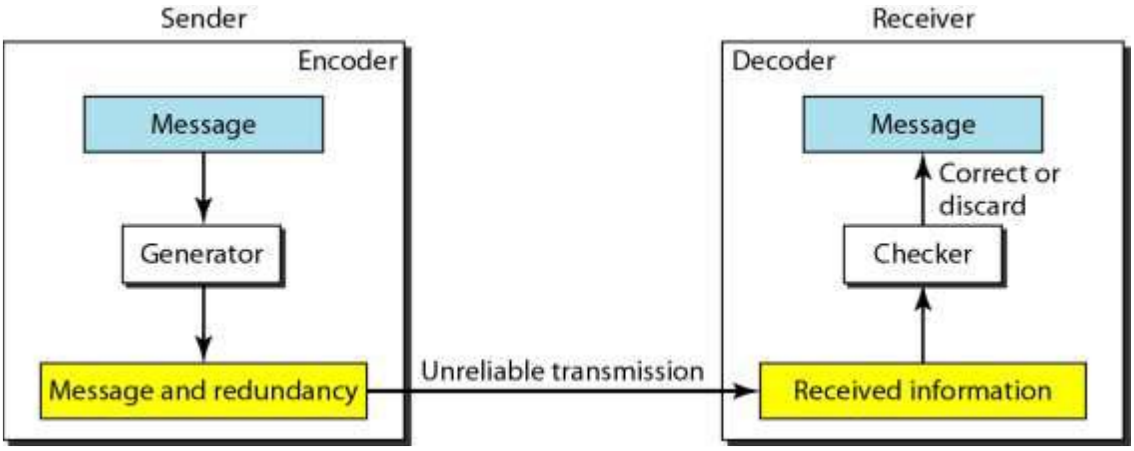
The term burst error means that two or more consecutive bits in the data unit have changed from 1 to 0 or from 0 to 1. In this case, 0100010001000011 was sent, but 0101101101000011, was received.



4.4.2- Error Detection

For a machine to check for errors this way would be slow, costly, and of questionable value. What we need is a mechanism that is simple and completely objective. One mechanism that would satisfy these requirements would be to *send every data unit twice*. The receiving device would then be able to do a bit for bit comparison between the two versions of the data. Any discrepancy would indicate an error and an appropriate correction mechanism could be set in place. This system would be completely accurate (the odds of errors being introduced onto exactly the same bits in both sets of data are infinitesimally small), but it would also be insupportably slow. Not only would the transmission time double, but the time it takes, to compare every unit bit by bit must be added.

The concept of including extra information solely for the purposes of comparison is a good one. But instead of repeating the entire data stream, a shorter group of bits may be appended to the end of each unit. This technique is called **redundancy** because the extra bits are redundant to the information; they are discarded as soon as the accuracy of the transmission has been determined.



Four types of redundancy checks are used in data communications:

a. Vertical Redundancy Check (VRC) (also called parity check).

b. Longitudinal Redundancy Check (LRC).

c. Cyclical Redundancy Check (CRC), and

d. Checksum.

Cyclical Redundancy Check (CRC)

The most powerful redundancy technique, CRC is based on binary division. The redundancy bits used by CRC are derived by dividing the data unit by the predetermined divisor and appending it to the end of the data string so that the resulting bit sequence must be exactly divisible by the divisor.

The CRC generator uses modulo-2 division. In the first stage, the four-bit divisor is subtracted from the first four bits of the dividend. Each bit of the divisor is subtracted from the corresponding bit of the dividend without disturbing the next higher bit.

The CRC generator is most often represented as an algebraic polynomial. This is useful because the code is short and can be easily mathematically understood.

$$x^7 + x^5 + x^2 + x + 1$$

Gives the binary:

1 0 1 0 0 1 1 1

Example: Determine the CRC code to be used to send message M(x) by using the divisor G(x) given:

M(x) = X5+X3 +X2+X

G(x) = X3+X

Form the polynomial M(x) =101110
And the G(x) =1010

Sender Side

```
      100111
1010 101110000
     1010
     ----
      0011
      0000
      ----
       0110
       0000
       ----
        1100
        1010
        ----
         1100
         1010
         ----
          1100
          1010
          ----
           110
```

Receiver Side

```
      100111
1010 101110110
     1010
     ----
      0011
      0000
      ----
       0110
       0000
       ----
        1101
        1010
        ----
         1111
         1010
         ----
          1010
          1010
          ----
          0000  message Accepted
```

The CRC code is=110
And
The message to be send= 101110110

Checksum

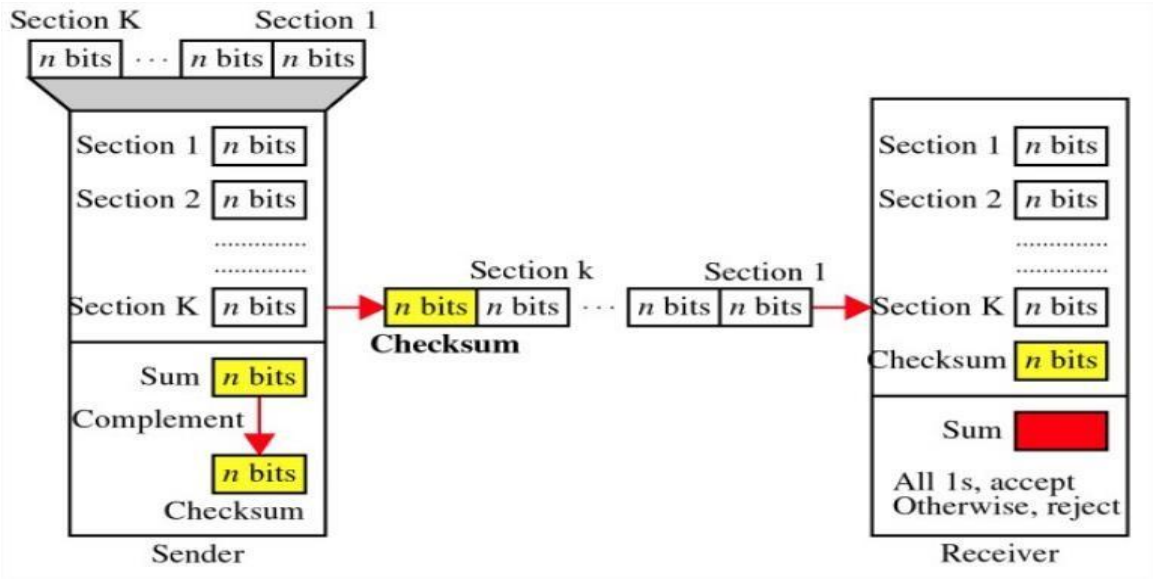
The error detection method used by the higher layer protocols. Like other methods, it depends on the concept of redundancy. In the sender, the checksum generator subdivided the data into equal segments of bits. These segments are added together using one's complement arithmetic in such a way that the total is also *n* bits long. That total is then complemented and appended to the end of the original data unit as redundancy bits called the checksum field. The data segment is referred to as *T* and the checksum as *–T*.

To create the checksum the sender does the following:

- The unit is divided into *k* sections, each of *n* bits.
- Sections 1 and 2 are added together using one's complement.
- Section 3 is added to the result of the previous step.

- Section 4 is added to the result of the previous step.
- The Process repeats until section *k* is added to the result of the previous step.
- The final result is complemented to make the checksum.

In the receiver, the coming data is subdivided into data and checksum. The total value of the data unit and the checksum must be zero for errorless transmission.



Example

Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using one's complement

	10101001
	00111001

Sum	11100010
Checksum	00011101

The pattern sent is 10101001 00111001 00011101

Now suppose the receiver receives the pattern sent and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

	10101001
	00111001
	00011101
Sum	11111111
Complement	00000000 means that the pattern is OK.

Now suppose there is a burst error of length 5 that affects 4 bits.

10101111 11111001 00011101

When the receiver adds the three sections, it gets

	10101111
	11111001
	00011101
Partial Sum	1 11000101
Carry	1
Sum	11000110
Complement	00111001 the pattern is corrupted.