Digital Transmission

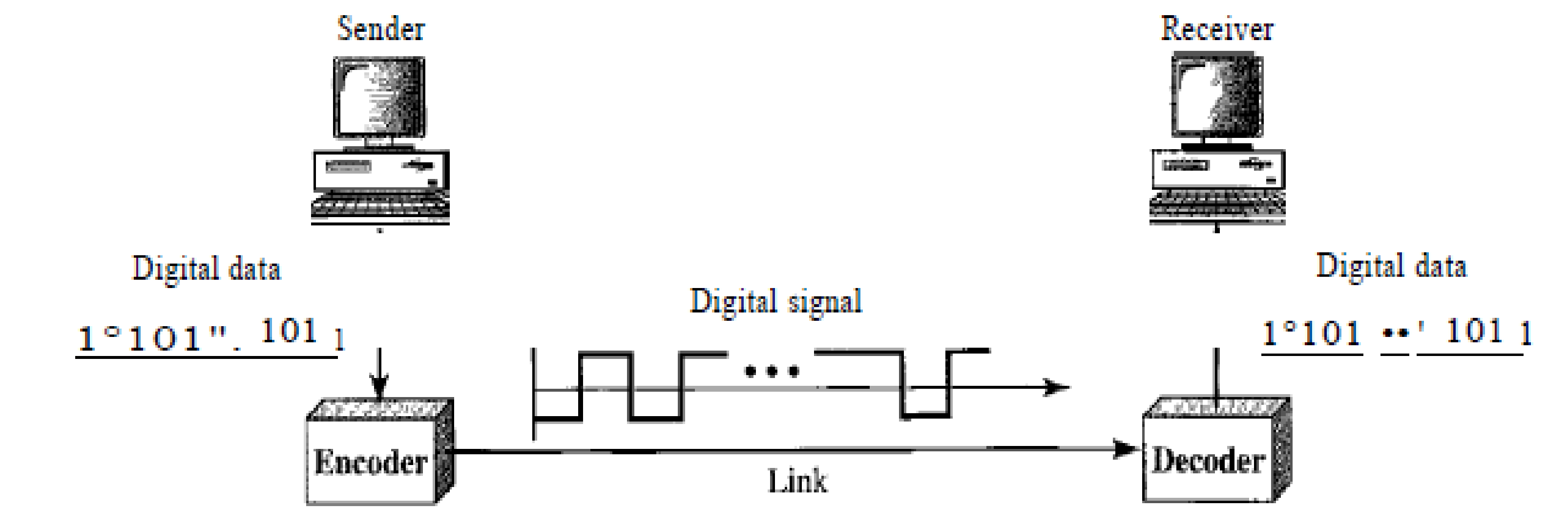
A computer network is designed to send information from one point to another. This information needs to be converted to either a digital signal or an analog signal for transmission. we will discuss the conversion to digital signals. first choice, we discussed data and signals. We said that data can be either digital or analog. We also said that signals that represent data can also be digital or analog.

In this section, we see how we can represent digital data by using digital signals. The conversion involves three techniques: line coding, block coding, and scrambling. Line coding is always needed, block coding and scrambling may or may not be needed.

Line Coding

Line coding is the process of converting digital data to digital signals. We assume that data, in the form of text, numbers, graphical images, audio, or video, are stored in computer memory as sequences of bits Line coding converts a sequence of bits to a digital signal. At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are recreated by decoding the digital signal. Figure 4.1 shows the process

Figure 4.1 *Line coding and decoding*



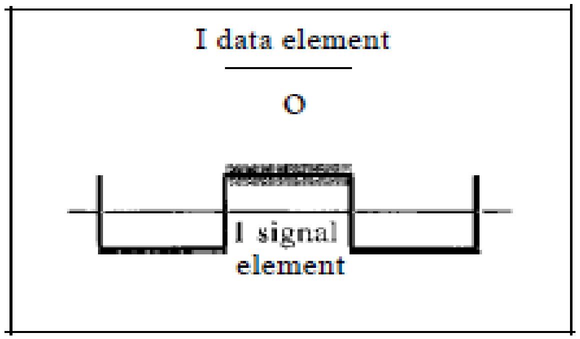
Before discussing different line coding schemes, we address their common characteristics.

Signal Element Versus Data Element

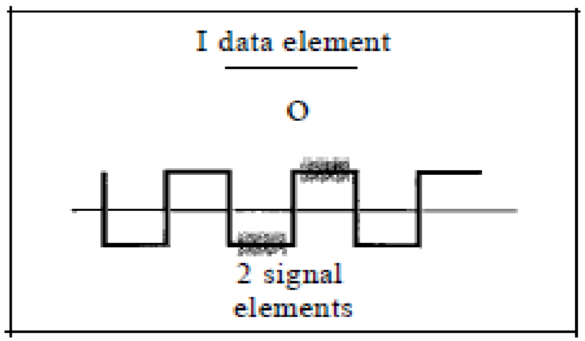
Let us distinguish between a data element and a signal element. In data communications, our goal is to send data elements. A data element is the smallest entity that can represent a piece of information: this is the bit. In digital data communications, a signal element carries data elements. A signal element is the shortest unit (time wise) of a digital signal. In other words, data elements are what we need to send; signal elements are what we can send. Data elements are being carried; signal elements are the carriers.

We define a ratio *r which is the number of data elements carried by each signal element*. Figure 4.2 shows several situations with different values of *r* .In part a of the figure, one data element is carried by one signal element *(r = 1)*. *In* part b of the figure, we need two signal elements (two transitions) to carry each data element (r=1/2)

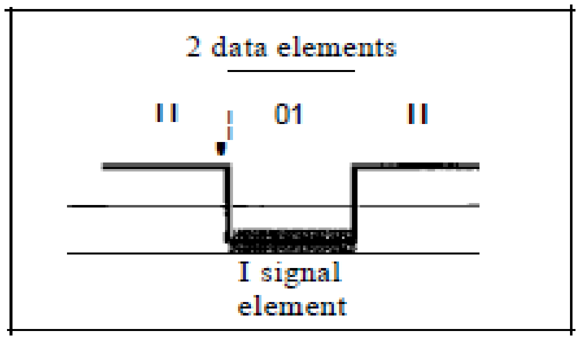
Figure 4.2 *Signal element versus data element*



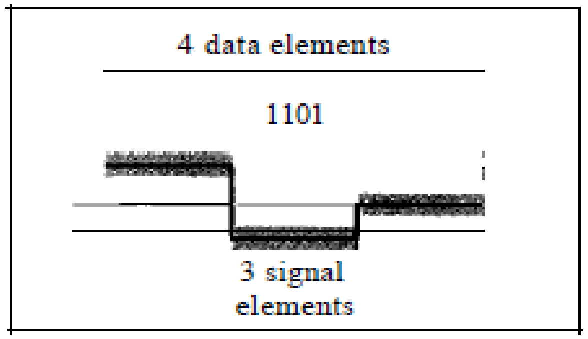
a. One data element per one signal element ($r = 1$)



b. One data element per two signal elements $\left(r = \frac{1}{2}\right)$



c. Two data elements per one signal element ($r = 2$)



d. Four data elements per three signal elements $\left(r = \frac{4}{3}\right)$

Data Rate Versus Signal Rate

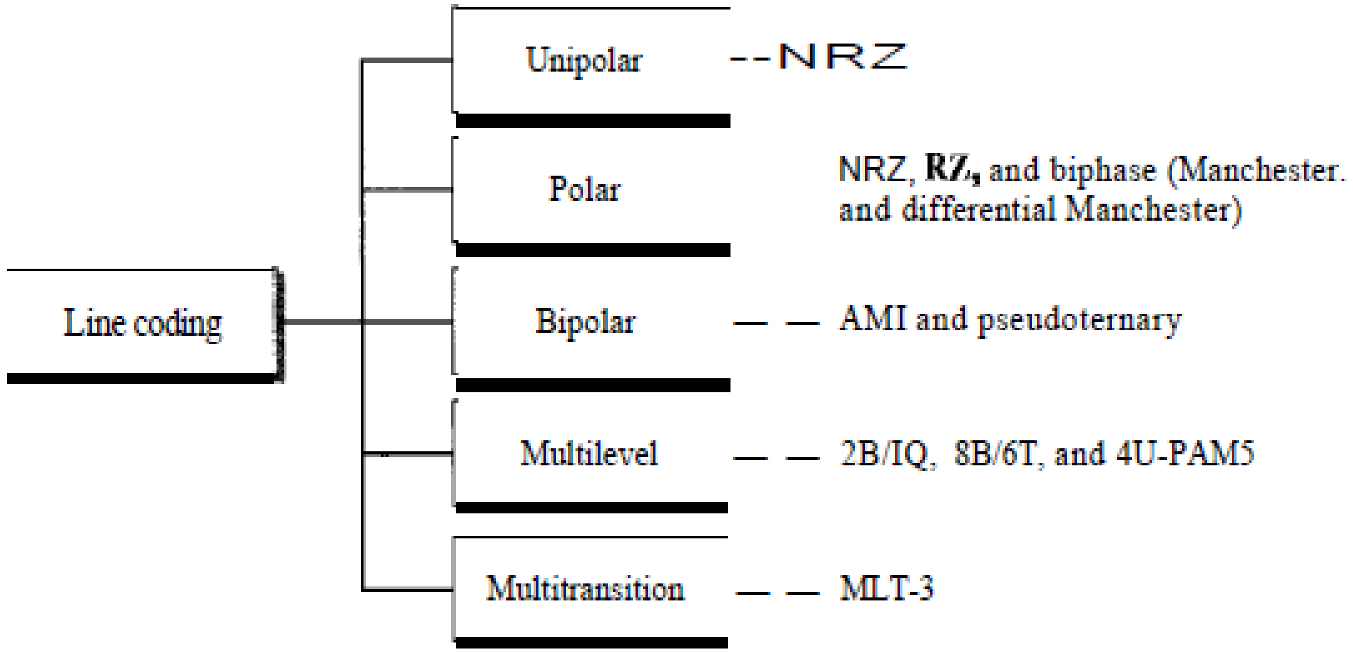
The data rate defines the number of data elements (bits) sent in Is. The unit is bits per second (bps). The signal rate is the number of signal elements sent in Is. The unit is the baud. There are several common terminologies used in the literature. The data rate is sometimes called the bit rate; the signal rate is sometimes called the pulse rate, the modulation rate, or the baud rate.

Line Coding Schemes

We can roughly divide line coding schemes into five broad categories, as shown in Figure 4.4.

Unipolar ,Polar ,Bipolar ,Multilevel ,Multitransmission

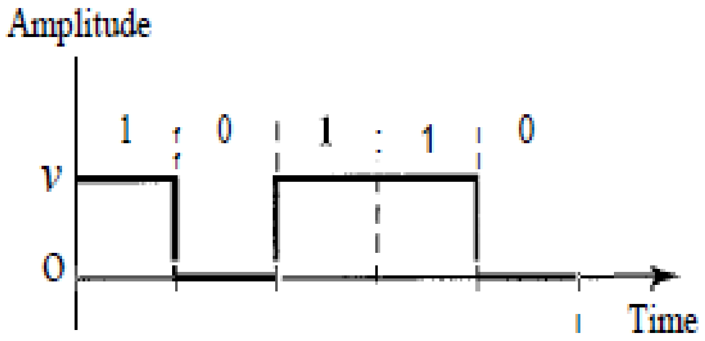
Figure 4.4 *Line coding schemes*


Line coding
Unipolar
--NRZ
Polar
NRZ, RZ, and biphase (Manchester. and differential Manchester)
Bipolar
— — AMI and pseudoternary
Multilevel
— — 2B/IQ, 8B/6T, and 4U-PAM5
Multitransition
— — MLT-3

Unipolar Scheme

In a unipolar scheme, all the signal levels are on one side of the time axis, either above or below. NRZ (Non-Return-to-Zero) Traditionally, a unipolar scheme was designed as a non-return-to-zero (NRZ) scheme in which the positive voltage defines bit I and the zero voltage defines bit O. It is called NRZ because the signal does not return to zero at the middle of the bit. Figure 4.5 show a unipolar NRZ scheme.

Figure 4.5 *Unipolar NRZ scheme*



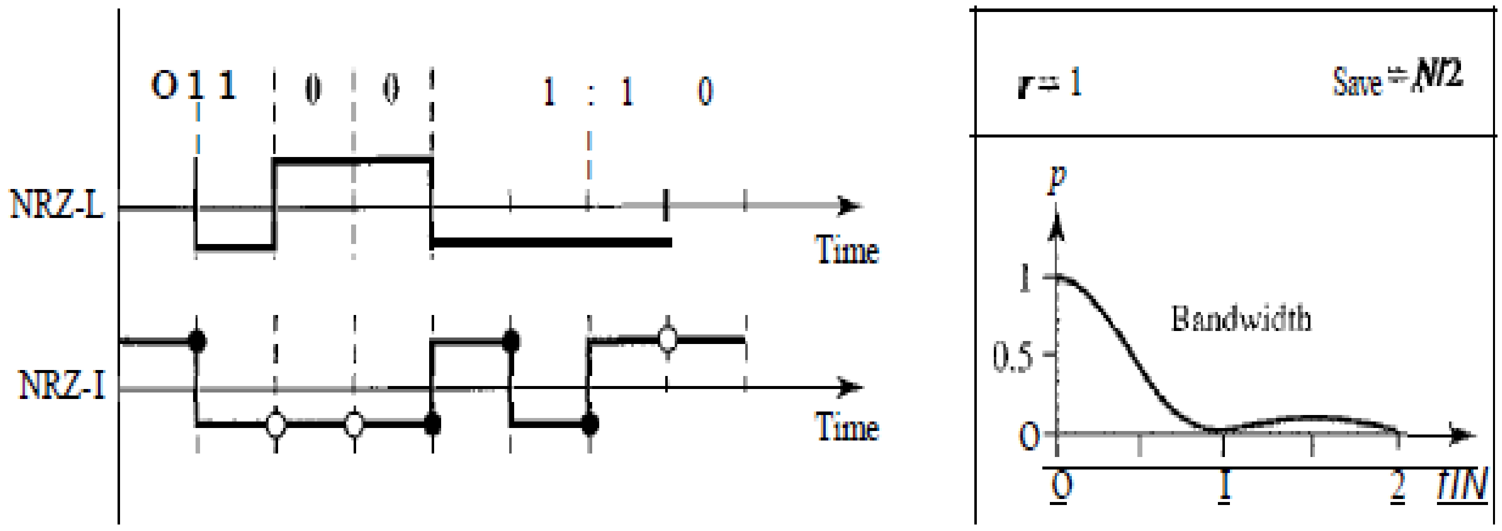
$$\boxed{\frac{1}{2}V^2 + \frac{1}{2}(0)^2 = \frac{1}{2}V^2}$$

Normalized power

Polar Schemes

In polar schemes, the voltages are on the both sides of the time axis. For example, the voltage level for 0 can be positive and the Non-Return-to-Zero (NRZ) In polar NRZ encoding, we use two levels of voltage amplitude. We can have two versions of polar NRZ: NRZ-Land NRZ-I, as shown in Figure 4.6. The figure also shows the value of *r, the average baud rate, and the bandwidth*. In the first variation, NRZ-L (NRZ-Level), the level of the voltage determines

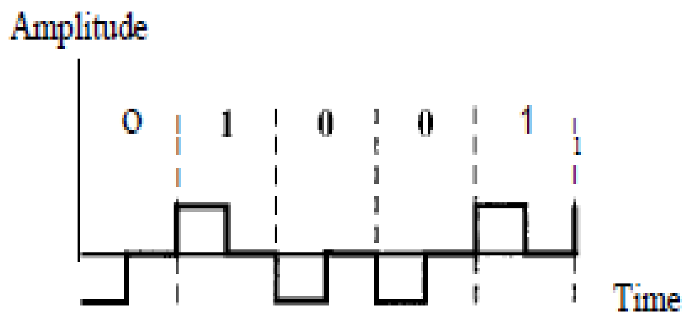
Figure 4.6 *Polar NRZ-L and NRZ-I schemes*

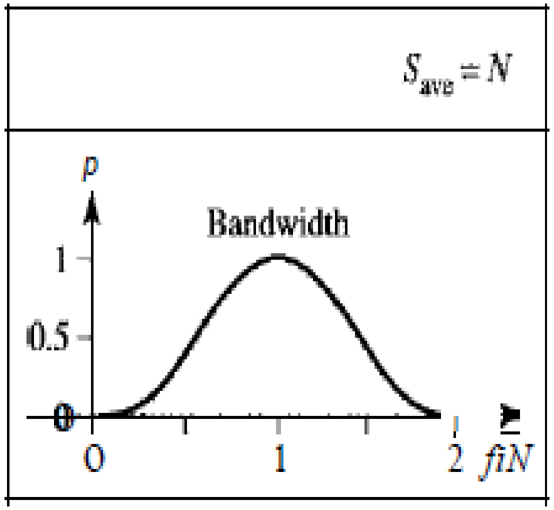


o No inversion: Next bit is 0 • Inversion: Next bit is 1 the value of the bit. In the second variation, NRZ-I (NRZ-Invert), the change or lack of change in the level of the voltage determines the value of the bit. If there is no change, the bit is 0; if there is a change, the bit is 1.voltage level for 1 can be negative.

Return to Zero (RZ) The main problem with NRZ encoding occurs when the sender and receiver clocks are not synchronized. The receiver does not know when one bit has ended and the next bit is starting. One solution is the return-to-zero (RZ) scheme, which uses three values: positive, negative, and zero. In RZ, the signal changes not between bits but during the bit.

Figure 4.7 *Polar RZ scheme*

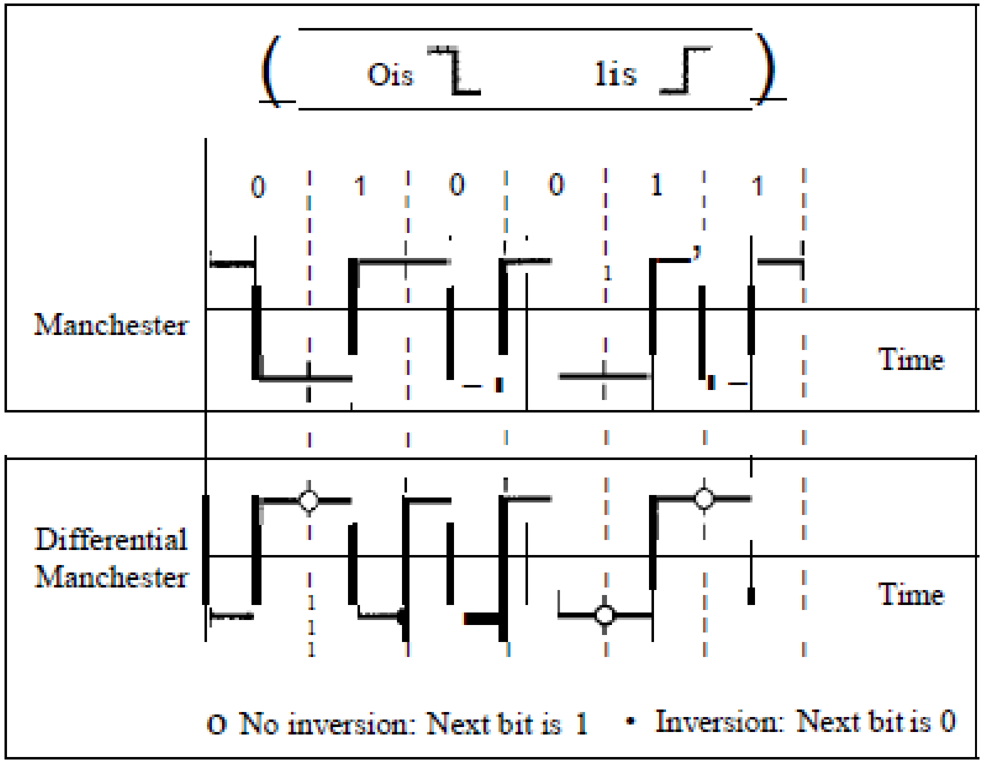


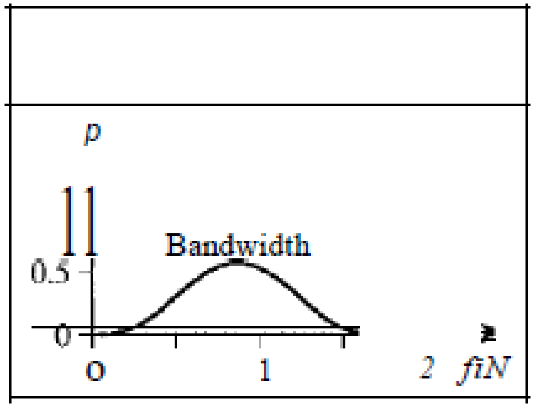


Biphase: Manchester and Differential Manchester The idea of RZ (transition at the middle of the bit) and the idea of NRZ-L are combined into the Manchester scheme. In Manchester encoding, the duration of the bit is divided into two halves. The voltage remains at one level during the first half and moves to the other level in the second half. The transition at the middle of the bit provides synchronization.

Differential Manchester, on the other hand, combines the ideas of RZ and NRZ-I. There is always a transition at the middle of the bit, but the bit values are determined at the beginning of the bit. If the next bit is 0, there is a transition; if the next bit is 1, there is none. Figure 4.8 shows both Manchester and differential Manchester encoding.

Figure 4.8 *Polar biphase: Manchester and differential Manchester schemes*





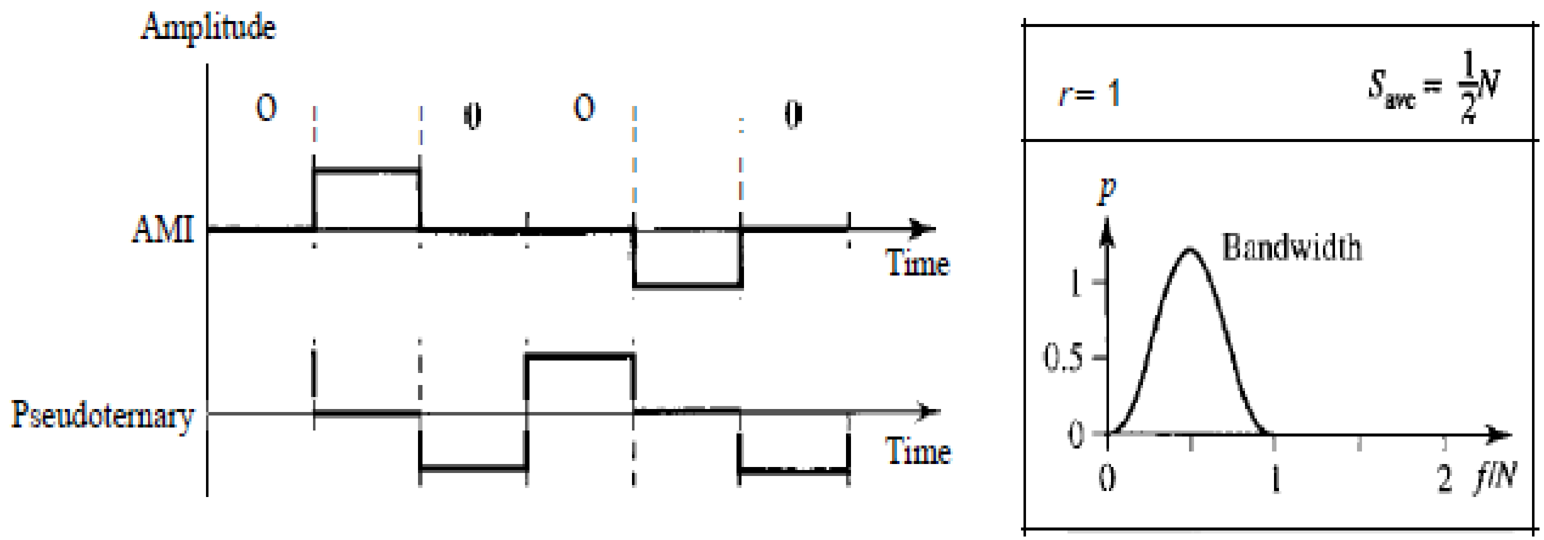
Bipolar Schemes

In bipolar encoding (sometimes called *multilevel binary), there are three voltage levels:*

positive, negative, and zero. The voltage level for one data element is at zero, while the voltage level for the other element alternates between positive and negative .AMI and Pseudoternary Figure 4.9 shows two variations of bipolar encoding: AMI

and pseudoternary.

Figure 4.9 *Bipolar schemes: AMI and pseudoternary*

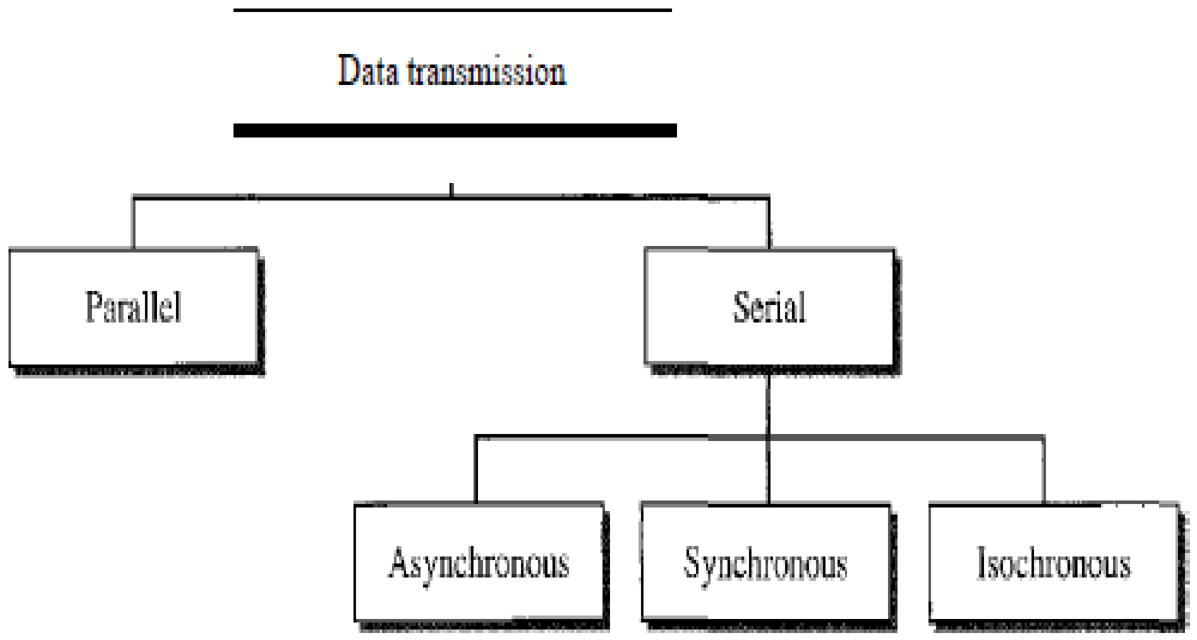


TRANSMISSION MODES

Of primary concern when we are considering the transmission of data from one device to another is the wiring, and of primary concern when we are considering the wiring is the data stream. Do we send 1 bit at a time; or do we group bits into larger groups and, if so, how?.

The transmission of binary data across a link can be accomplished in either parallel or serial mode. In parallel mode, multiple bits are sent with each clock tick. In serial mode, 1 bit is sent with each clock tick. While there is only one way to send parallel data, there are three subclasses of serial transmission: asynchronous, synchronous and isochronous (see Figure 4.31)

Figure 4.31 *Data transmission and modes*

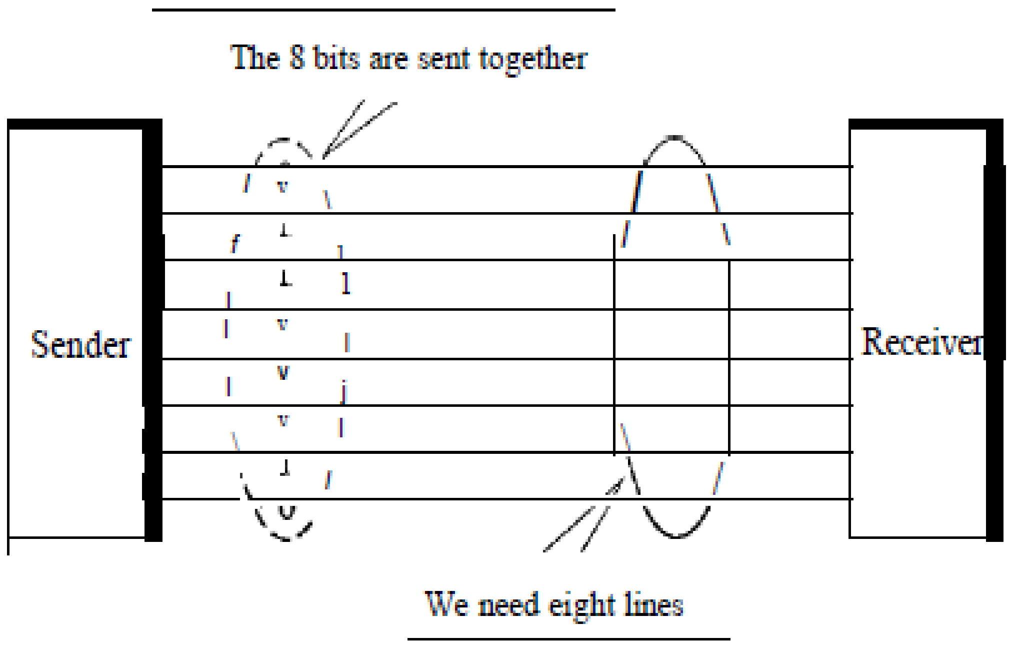


Parallel Transmission

Binary data, consisting of Is and Os, may be organized into groups of *n bits each*. Computers produce and consume data in groups of bits much as we conceive of and use spoken language in the form of words rather than letters. By grouping, we can send data *n bits at a time instead of 1. This is called parallel transmission.*

The mechanism for parallel transmission is a conceptually simple one: Use *n wires* to send *n bits at one time*. *That way each bit has its own wire, and all n bits of one* group can be transmitted with each clock tick from one device to another. Figure 4.32 shows how parallel transmission works for *n =8*. *Typically, the eight wires are bundled* in a cable with a connector at each end.

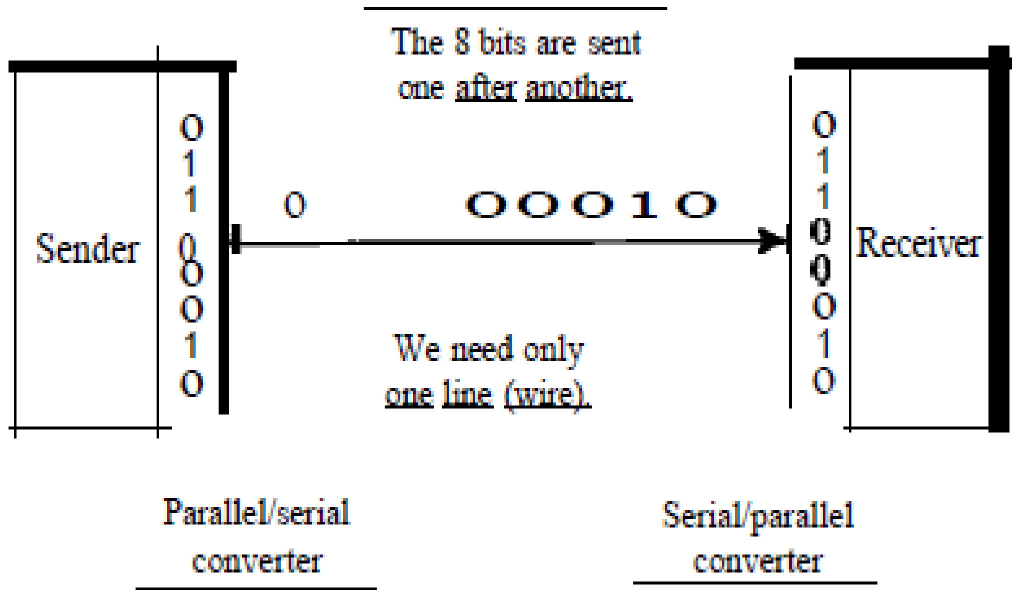
Figure 4.32 *Parallel transmission*



Serial Transmission

In serial transmission one bit follows another, so we need only one communication channel rather than *n to transmit data between two communicating devices (see* Figure 4.33) The advantage of serial over parallel transmission is that with only one communication channel, serial transmission reduces the cost of transmission over parallel by roughly a factor of *n*.

Figure 4.33 *Serial transmission*


The 8 bits are sent one after another.
Sender
0
1
1
0
0
0
1
0
0
00010
Receiver
0
1
1
0
0
0
1
0
We need only one line (wire).
Parallel/serial converter
Serial/parallel converter

Serial transmission occurs in one of three ways:

asynchronous, synchronous, and isochronous.

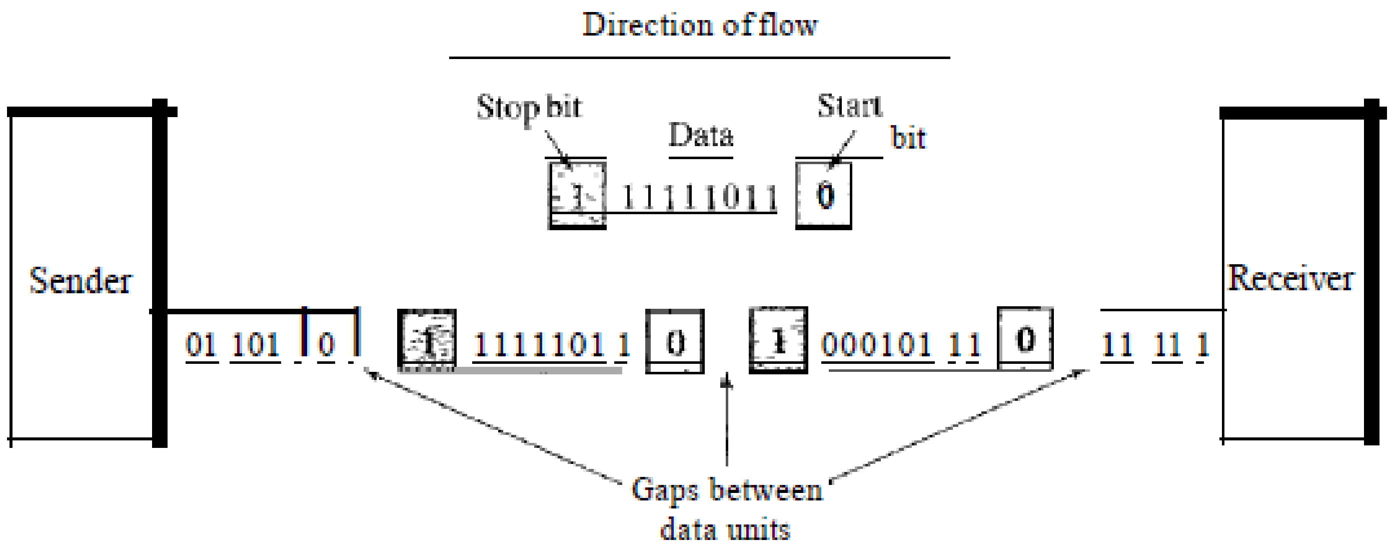
Asynchronous Transmission

Asynchronous transmission is so named because the timing of a signal is unimportant. Instead, information is received and translated by agreed upon patterns. As long as those patterns are followed, the receiving device can retrieve the information without regard to the rhythm in which it is sent. Patterns are based on grouping the bit stream into bytes. Each group, usually 8 bits, is sent along the link as a unit. The sending system handles each group independently, relaying it to the link whenever ready, without regard to a timer.

Without synchronization, the receiver cannot use timing to predict when the next group will arrive. To alert the receiver to the arrival of a new group, therefore, an extra bit is added to the beginning of each byte. This bit, usually a 0, is called the start bit. To let the receiver know that the byte is finished, 1 or more additional bits are appended

to the end of the byte. These bits, usually I s, are called stop bits. By this method, each byte is increased in size to at least 10 bits, of which 8 bits is information and 2 bits or more are signals to the receiver.

Figure 4.34 *Asynchronous transmission*



In addition, the transmission of each byte may then be followed by a gap of varying duration

Synchronous Transmission

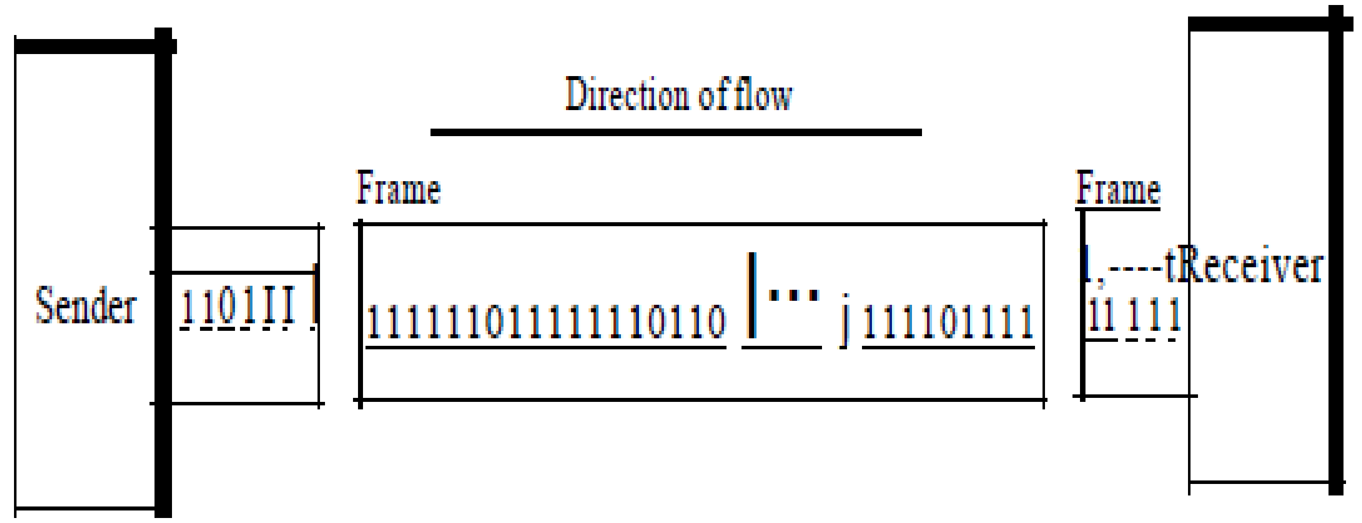
In synchronous transmission, the bit stream is combined into longer "frames," which may contain multiple bytes. Each byte, however, is introduced onto the transmission link without a gap between it and the next one. It is left to the receiver to separate the bit stream into bytes for decoding purposes. In other words, data are transmitted as an unbroken string of 1s and Os, and the receiver separates that string into the bytes, or characters, it needs to reconstruct the information.

The advantage of synchronous transmission is speed. With no extra bits or gaps to introduce at the sending end and remove at the receiving end, and, by extension, with

fewer bits to move across the link, synchronous transmission is faster than asynchronous

transmission. For this reason, it is more useful for high-speed applications such as the transmission of data from one computer to another.

Figure 4.35 *Synchronous transmission*



Isochronous

In real-time audio and video, in which uneven delays between frames are not acceptable, synchronous transmission fails. For example, TV images are broadcast at the rate of 30 images per second; they must be viewed at the same rate. If each image is sent by using one or more frames, there should be no delays between frames. For this type of application, synchronization between characters is not enough; the entire stream of bits must be synchronized. The isochronous transmission guarantees that the data arrive at a fixed rate