6 Digital communications basics

6.1 Introduction

As we showed in Figure 5.1 in the last chapter, associated with all the networks that are used to support multimedia applications is a standard network interface to which all the end systems/terminal equipments that are attached to the network must adhere. Hence in each end system is a **network interface card** (**NIC**) – consisting of hardware controlled by associated software – that performs the related network interface functions. In the following chapters we describe the interface associated with the different types of network that we identified in Figure 5.1 and also we present an insight into the operation both of the networks and, where appropriate, the internal (network) protocols that are used.

In the last chapter we also identified a selection of the application standards relating to multimedia and found that, in general, the integrated information stream generated by the various applications is a series of blocks of binary data of varying size. In some instances, the application involves a dialog using these blocks while in others it involves the transfer of a stream of blocks. In terms of the network interface, however, the information relating to the different applications that is to be transferred over the network is

treated simply as a string of one or more blocks containing what is referred to as (network) user data.

Although the various networks that are used operate in different ways, the physical and link layers of both the network interface standards and the internal network standards have a number of common features associated with them. So before we describe the operation of the various networks and their interfaces, we shall discuss in this chapter the basic principles associated with digital communications which are common for all networks.

The access lines (and the internal transmission lines used within the various networks) all use bit-serial transmission. In general, therefore, the signal output by the NIC simply varies between two voltage levels ($+V$ and $-V$ for example) at a rate determined by the transmitted bit rate. This mode of transmission is known as **baseband transmission** and is illustrated in Figure 6.1(a). Thus with networks that provide a digital interface, such as a LAN and an ISDN, baseband transmission is also used over the access lines to the network. With networks such as a PSTN, however, analog transmission is used over the access lines since, as we shall expand upon in Section 7.2.1, the presence of a transformer in the speech path means a DC signal – for example a long string of binary 0s or 1s – will not be discernible. As we saw in Section 2.5.1, the bandwidth used over these lines is that of a bandlimited speech signal and is from 200 Hz through to 3400 Hz. Hence as we show in Figure 6.1(b), prior to transmitting the baseband signal output by the NIC, it is necessary first to convert this signal into an analog signal within the same bandwidth that is used for speech. This is done, for example, by modulating – also known as mixing or multiplying – a single-frequency audio signal/tone – chosen from within the bandwidth used for a speech signal and known as the **carrier signal** – by the binary signal to be transmitted. This mode of transmission is known as **modulated transmission** and the unit that performs the modulation and demodulation functions is a modem.

We shall describe modems and this mode of transmission further in Section 7.2.2. It should be noted, however, that even though modulated transmission is sometimes used over the access lines to the network – and also within the various types of broadcast entertainment networks – normally, the output from the NIC within each end system is a baseband signal.

When transmitting any type of electrical signal over a transmission line, the signal is **attenuated** (decreased in amplitude) and **distorted** (misshapen) by the transmission medium. Also, present with all types of transmission medium is an electrical signal known as **noise**. The amplitude of the noise signal varies randomly with time and adds itself to the electrical signal being transmitted over the line. The combined effect is that at some stage, the receiver is unable to determine from the attenuated received signal with noise whether the transmitted signal was a binary 1 or 0. An example showing the combined effect of attenuation, distortion, and noise on a baseband signal is shown in Figure 6.2.

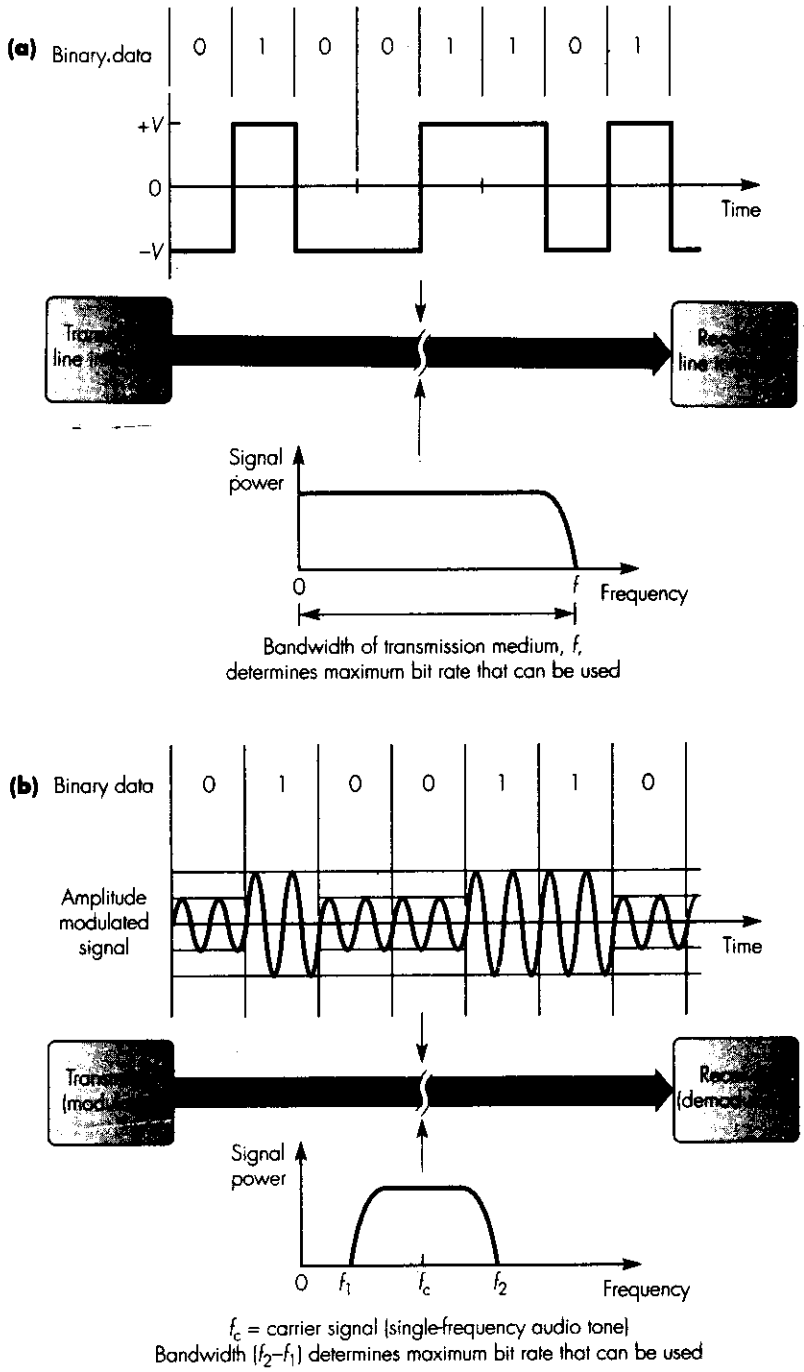


Figure 6.1 Modes of transmission: (a) baseband transmission; (b) modulated transmission.

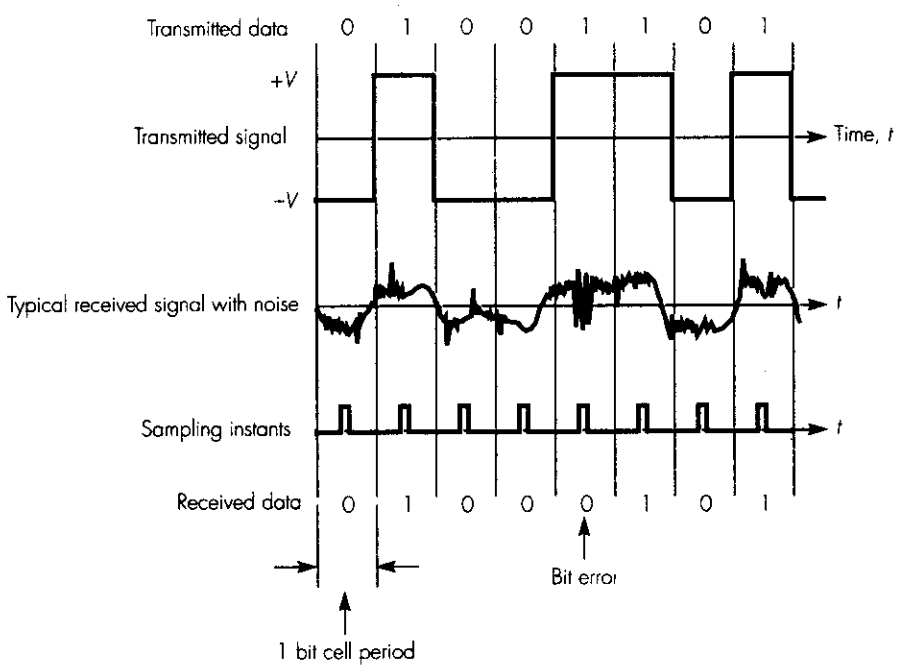


Figure 6.2 Effect of attenuation, distortion, and noise on transmitted signal.

In practice, the level of signal impairment is determined by:

- the type of transmission medium,
- the length of the transmission medium,
- the bandwidth of the medium,
- the bit rate of the data being transmitted.

We shall discuss the characteristics of the different types of transmission media in Section 6.2.

As we show in Figure 6.2, the received signal is at its peak amplitude in the center of each **bit cell period**. Hence in order for the receiving electronics to determine the signal level (and hence bit) present on the line during each bit cell period, it endeavors to sample the received signal at the center of each bit cell. When the receiver is doing this, it is said to be in **bit synchronism** with the incoming bitstream. However, although the receiver knows the nominal bit rate being used by the transmitter to transmit the bitstream, the receiver electronics operates quite independently of the transmitter with the effect that the receiver clock used to sample the signal will be slightly different from that used at the transmitter. In practice, therefore, sampling the signal present on the line at the correct time instant is a non-trivial task.

Achieving bit synchronism is only the initial step in achieving the reliable transfer of information over a transmission line. Most of the bit synchronization methods that are used take a variable number of bits before bit synchronism is achieved. Hence in order to interpret the received bitstream on the correct character/byte boundaries, the receiver electronics must also be able to determine the start of each new character/byte and, where appropriate, the start and end of each block of characters/bytes. Again, when the receiver is doing this, it is said to be in **character/byte synchronism** and **block/frame synchronism** respectively. In practice, there are two alternative types of baseband transmission – asynchronous and synchronous – and each uses different methods to achieve the three levels of synchronization. We describe the methods used with asynchronous transmission in Section 6.4 and those used with synchronous transmission in Section 6.5.

As we showed in Figure 6.2, if the amplitude of the received signal falls below the noise signal level, then the received signal may be incorrectly interpreted and, if it is, a transmission/bit error will result. To allow for this possibility, additional bits are added to each transmitted block to enable the receiver to determine – to high probability – when transmission errors are present in a received block. We describe a selection of the methods that are used to detect the presence of transmission errors in Section 6.6.

In some applications, the loss of occasional blocks of information from the received bitstream can be tolerated and hence blocks that are found to contain transmission errors are simply discarded. In other applications, however, the loss of a block is unacceptable and it then becomes necessary for the receiver to request another copy of those blocks that contain errors. This involves the receiver sending what are called error control messages back to the transmitter. This is done in one of two ways and depends on whether the network interface offers a best-effort service or a reliable service. If the service offered is best-effort, both the network and link layers simply discard blocks in error and the error recovery procedure is carried out as part of the transport protocol in the two communicating end systems. If a reliable network service is offered, then the error recovery is part of the link protocol. We discuss link protocols in Section 6.7 and a practical example in Section 6.8. We shall defer the discussion of transport protocols until Chapter 12.

6.2 Transmission media

The transmission of an electrical signal requires a transmission medium which, normally, takes the form of a transmission line. In some cases, this consists of a pair of conductors or wires. Common alternatives are a beam of light guided by a glass fiber and electromagnetic waves propagating through free space. The type of transmission medium is important, since the various types of media have different bandwidths associated with them which, in turn, determines the maximum bit rate that can be used. We discuss the more common types of transmission media in the following subsections.

6.2.1 Two-wire open lines

A two-wire open line is the simplest transmission medium. Each wire is insulated from the other and both are open to free space. This type of line is adequate for connecting equipment that is up to 50 m apart using moderate bit rates (less than, say, 19.2 kbps). The signal, typically a voltage or current level relative to some ground reference, is applied to one wire while the ground reference is applied to the other.

Although a two-wire open line can be used to connect two computers (data terminal equipments or DTEs) directly, it is used mainly for connecting a DTE to local data circuit-terminating equipment (DCE) – for example a modem. Such connections usually utilize multiple lines, the most common arrangement being a separate insulated wire for each signal and a single wire for the common ground reference. The complete set of wires is then either enclosed in a single protected **multicore cable** or molded into a **flat ribbon cable** as shown in Figure 6.3(a).

With this type of line, care must be taken to avoid cross-coupling of electrical signals between adjacent wires in the same cable. This is known as **crosstalk** and is caused by **capacitive coupling** between the two wires. In addition, the open structure makes it susceptible to the pick-up of spurious **noise signals** from other electrical signal sources caused by **electromagnetic radiation**. The main problem with such signals is that they may be picked up in just one wire – for example the signal wire – creating an additional difference signal between the two wires. Since the receiver normally operates using the difference signal between the two wires, this can give rise to an erroneous interpretation of the combined (signal plus noise) received signal. These factors all contribute to the limited lengths of line and bit rates that can be used reliably.

6.2.2 Twisted-pair lines

We can achieve much better immunity to spurious noise signals by employing a twisted-pair line in which a pair of wires are twisted together. The proximity of the signal and ground reference wires means that any interference signal is picked up by both wires reducing its effect on the difference signal. Furthermore, if multiple twisted pairs are enclosed within the same cable, the twisting of each pair within the cable reduces crosstalk. A schematic of a twisted-pair line is shown in Figure 6.3(b).

Twisted-pair lines are suitable, with appropriate line driver and receiver circuits that exploit the potential advantages gained by using such a geometry, for bit rates in order of 1 Mbps over short distances (less than 100 m) and lower bit rates over longer distances. More sophisticated driver and receiver circuits enable bit rates of tens of Mbps to be achieved over similar distances. Such lines, known as **unshielded twisted pairs (UTPs)**, are used extensively in telephone networks and (with special integrated circuits) in many local area networks. With **shielded twisted pairs (STPs)**, a protective screen or shield is used to reduce further the effects of interference signals. This is shown in Figure 6.3(c) and is now preferred to UTP in networking applications.

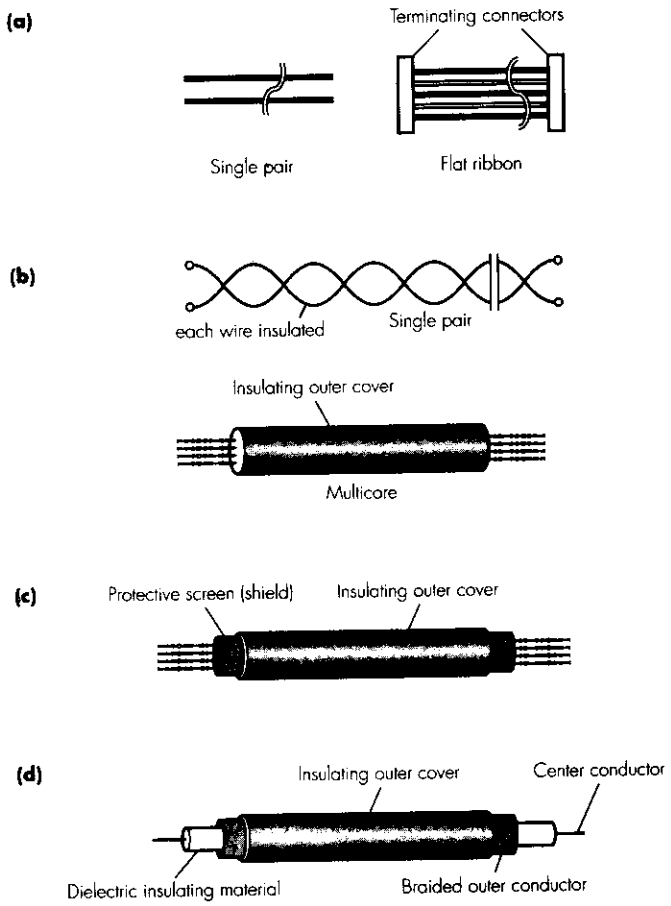


Figure 6.3 Copper wire transmission media: (a) two-wire and multiwire open lines; (b) unshielded twisted pair; (c) shielded twisted pair; (d) coaxial cable.

6.2.3 Coaxial cable

The main limiting factors of a twisted-pair line are its capacity and a phenomenon known as the **skin effect**. As the bit rate (and hence frequency) of the transmitted signal increases, the current flowing in the wires tends to flow only on the outer surface of the wire, thus using less of the available cross-section. This increases the electrical resistance of the wires for higher-frequency signals, leading to higher attenuation. In addition, at higher frequencies, more signal power is lost as a result of radiation effects. Hence, for applications that demand a high bit rate over long distances, coaxial cable is often used as the transmission medium.

Coaxial cable minimizes both these effects. Figure 6.3(d) shows the signal and ground reference wires as a solid center conductor running concentrically (coaxially) inside a solid (or braided) outer circular conductor. Ideally the space between the two conductors should be filled with air, but in practice it is normally filled with a dielectric insulating material with a solid or honeycomb structure.

The center conductor is effectively shielded from external interference signals by the outer conductor. Also only minimal losses occur as a result of electromagnetic radiation and the skin effect because of the presence of the outer conductor. Coaxial cable can be used with either baseband or modulated transmission, but typically 10 Mbps over several hundred meters – or higher with modulation – is perfectly feasible. As we shall see in Section 11.2, coaxial cable is used extensively in cable television (CATV) networks.

6.2.4 Optical fiber

While the geometry of coaxial cable significantly reduces the various limiting effects, the maximum signal frequency, and hence the bit rate that can be transmitted using a solid (normally copper) conductor, although very high, is limited. This is also the case for twisted-pair cable. **Optical fiber cable** differs from both these transmission media in that it carries the transmitted bitstream in the form of a fluctuating beam of light in a glass fiber, rather than as an electrical signal on a wire. Light waves have a much wider bandwidth than electrical waves, enabling optical fiber cable to achieve transmission rates of hundreds of Mbps. It is used extensively in the core transmission network of PSTNs and LANs and also CATV networks.

Light waves are also immune to electromagnetic interference and crosstalk. Hence optical fiber cable is extremely useful for the transmission of lower bit rate signals in electrically noisy environments, in steel plants, for example, which employ much high-voltage and current-switching equipments. It is also being used increasingly where security is important, since it is difficult physically to tap.

An optical fiber cable consists of a single glass fiber for each signal to be transmitted, contained within the cable's protective coating which also shields the fiber from any external light sources. See Figure 6.4(a). The light signal is generated by an optical transmitter, which performs the conversion from a normal electrical signal as used in a DTE. An optical receiver is used to perform the reverse function at the receiving end. Typically, the transmitter uses a **light-emitting diode (LED)** or **laser diode (LD)** to perform the conversion operation while the receiver uses a light-sensitive **photodiode** or **photo transistor**.

The fiber itself consists of two parts: an optical core and an optical cladding with a lower refractive index. Light propagates along the optical fiber core in one of three ways depending on the type and width of core material used. These transmission modes are shown in Figure 6.4(b).

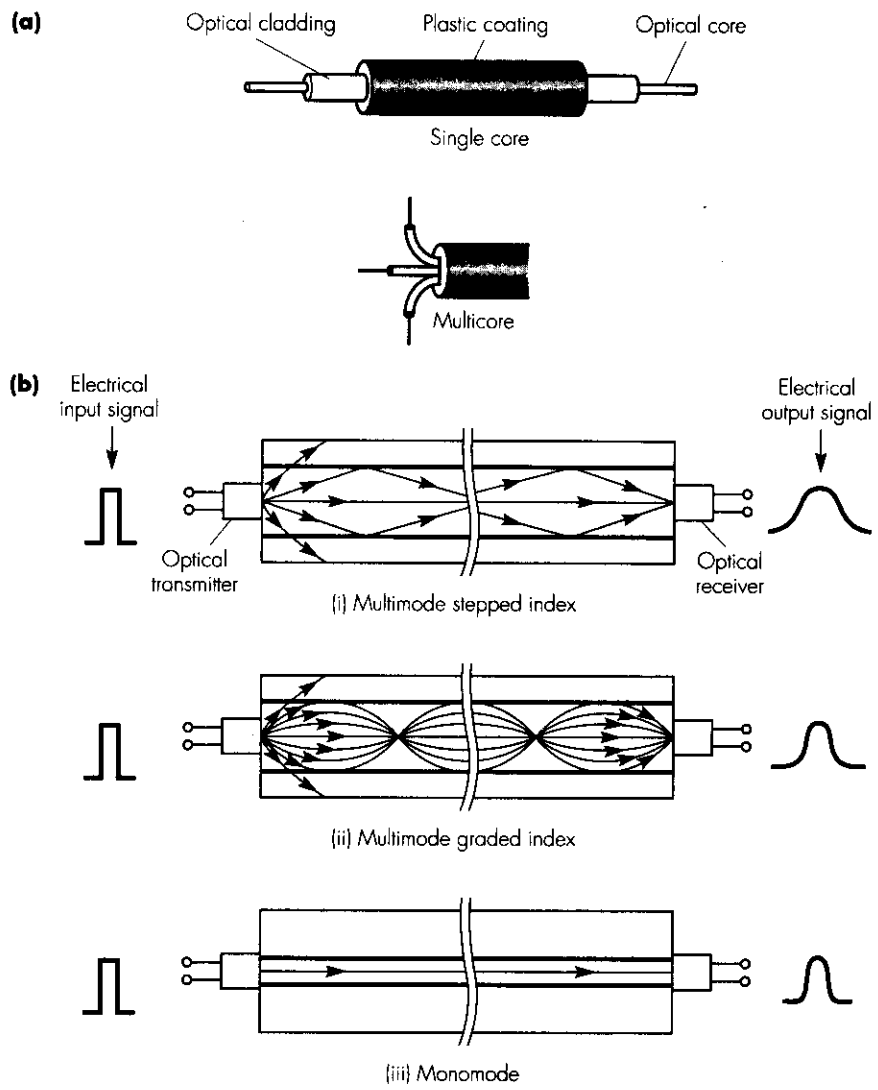


Figure 6.4 Optical fiber transmission media: (a) cable structures; (b) transmission modes.

In a **multimode stepped index fiber** the cladding and core material each has a different but uniform refractive index. All the light emitted by the diode at an angle less than the critical angle is reflected at the cladding interface and propagates along the core by means of multiple (internal) reflections. Depending on the angle at which it is emitted by the diode, the light will take a variable amount of time to propagate along the cable. Therefore the received signal has a wider pulse width than the input signal

with a corresponding decrease in the maximum permissible bit rate. This effect is known as **dispersion** and means this type of cable is used primarily for modest bit rates with relatively inexpensive LEDs compared to laser diodes.

Dispersion can be reduced by using a core material that has a variable (rather than constant) refractive index. As we show in Figure 6.4(b), in a **multimode graded index fiber** light is refracted by an increasing amount as it moves away from the core. This has the effect of narrowing the pulse width of the received signal compared with stepped index fiber, allowing a corresponding increase in maximum bit rate.

Further improvements can be obtained by reducing the core diameter to that of a single wavelength (3–10μm) so that all the emitted light propagates along a single (dispersionless) path. Consequently, the received signal is of a comparable width to the input signal and is called **monomode fiber**. It is normally used with LDs and can operate at hundreds of Mbps.

Alternatively, multiple high bit rate transmission channels can be derived from the same fiber by using different portions of the optical bandwidth for each channel. This mode of operation is known as **wave-division multiplexing** (**WDM**) and, when using this, bit rates in excess of tens of Gbps can be achieved.

6.2.5 Satellites

All the transmission media we have discussed so far have used a physical line to carry the transmitted information. However, data can also be transmitted using electromagnetic (radio) waves through free space as in **satellite** systems. A collimated **microwave beam**, onto which the data is modulated, is transmitted to the satellite from the ground. This beam is received and retransmitted (relayed) to the predetermined destination(s) using an on-board circuit known as a **transponder**. A single satellite has many transponders, each covering a particular band of frequencies. A typical satellite channel has an extremely high bandwidth (500 MHz) and can provide many hundreds of high bit rate data links using a technique known as **time division multiplexing** (**TDM**). We shall describe this in Section 7.2.3 but, essentially, the total available capacity of the channel is divided into a number of subchannels, each of which can support a high bit rate link.

Satellites used for communication purposes are normally **geostationary**, which means that the satellite orbits the earth once every 24 hours in synchronism with the earth's rotation and hence appears stationary from the ground. The orbit of the satellite is chosen so that it provides a line-of-sight communication path to the transmitting station(s) and receiving station(s). The degree of the collimation of the microwave beam retransmitted by the satellite can be either coarse, so that the signal can be picked up over a wide geographical area, or finely focussed, so that it can be picked up only over a limited area. In the second case the signal power is higher allowing smaller-diameter receivers called **antennas** or **dishes** to be used. Satellites are widely used for data

transmission applications ranging from interconnecting different national computer communication networks to providing high bit rate paths to link communication networks in different parts of the same country. They are also used in entertainment applications for the broadcast of TV programs.

A typical satellite system used for TV broadcast applications is shown in Figure 6.5(a). Only a unidirectional transmission path is used with the up and down channels operating at different frequencies. For data communication applications, however, a more common configuration involves a central hub ground station that communicates with a number of ground stations distributed around the country. Each ground station has a small antenna associated with it – typically 1 meter in diameter – which, in addition to receiving the signal transmitted by the hub station, allows the station to transmit back to the hub. Such ground stations are known as **very small aperture terminals** or **VSATs**. Typically, as we show in Figure 6.5(b), the computer associated with each VSAT communicates with a central computer connected to the hub. Normally, the central site broadcasts to all VSATs at a high bit rate of 0.5–2 Mbps while in the reverse direction each VSAT transmits at a lower bit rate of up to 64 kbps.

To communicate with a particular VSAT, the central site broadcasts the message with the identity of the intended VSAT at the head of the message. For applications that require VSAT-to-VSAT communication, all messages are first sent to the central site – via the satellite – which then broadcasts them to the intended recipients. With the next generation of higher-powered satellites it will be possible for the routing to be carried out on board the satellite without passing through a central site. Direct VSAT-to-VSAT communication is then possible.

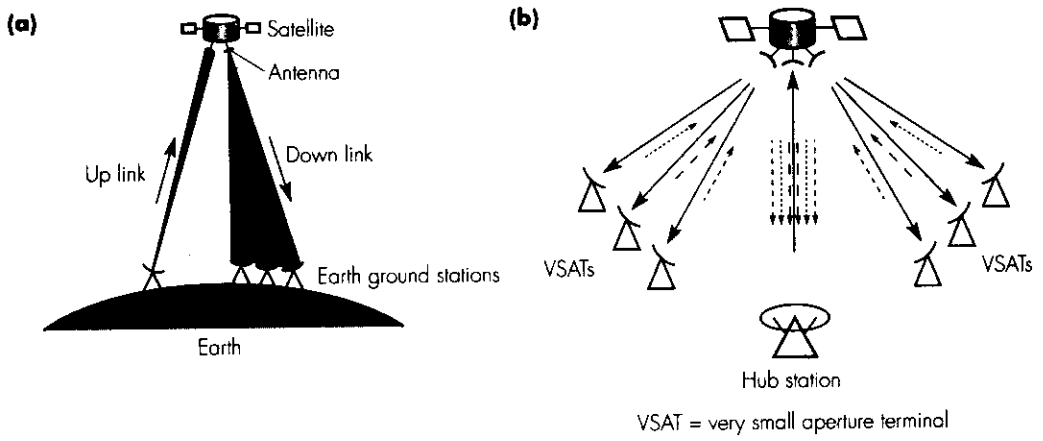


Figure 6.5 Satellite systems: (a) broadcast television; (b) data communications.

6.2.6 Terrestrial microwave

Terrestrial microwave links are widely used to provide communication links when it is impractical or too expensive to install physical transmission media, for example across a river or perhaps a swamp or desert. As the collimated microwave beam travels through the earth's atmosphere, it can be disturbed by such factors as manufactured structures and adverse weather conditions. With a satellite link, on the other hand, the beam travels mainly through free space and is therefore less prone to such effects. Nevertheless, line-of-sight microwave communication through the earth's atmosphere can be used reliably for the transmission of relatively high bit rates over distances in excess of 50 km.

6.2.7 Radio

Radio transmission using lower-frequency radio waves (c.f. microwaves) is also used for the transmission of digital information in place of fixed-wire links over distances up to several kilometers.

Modulated transmission is used and example applications include mobile telephony and more general mobile data applications. A radio transmitter – known as a **base station** – is located at a fixed-wire termination point as shown in Figure 6.6(a). This provides a cordless link to the fixed-wire termination point for any handset/terminal that is within the (radio) field of coverage of the base station.

Multiple base stations must be used for applications such as mobile telephony that require a wider coverage area or a higher density of users. The coverage area of each base station is restricted – by limiting its power output – so that it provides only sufficient channels to support the total number of calls in that area. Wider coverage is achieved by arranging multiple base stations in a cell structure as shown in Figure 6.6(b). In practice, the size of each cell varies and is determined by such factors as the handset/terminal density and local terrain.

Each base station operates using a different band of frequencies from its neighbors. However, since the field of coverage of each base station is limited, it is possible to reuse its frequency band in other parts of the network. All the base stations within a region are connected by fixed-wire lines to a mobile switching center (MSC) which, in turn, is connected both to MSCs in other regions and the fixed telephone network. In this way, a handset can make and receive calls both to other handsets and also to telephones that are connected to the fixed network.

A similar arrangement can be utilized within a building to provide cordless links to computer-based equipment within each office. In such cases one or more base stations are located on each floor of the building and connected to the fixed network. Each base station provides cordless links to the fixed network for all the computers in its field of coverage. This avoids rewiring whenever a new computer is installed or moved, but at the cost of providing a radio unit to convert the data into and from a radio signal. The usable data rate is often much lower than that achieved with fixed wiring.

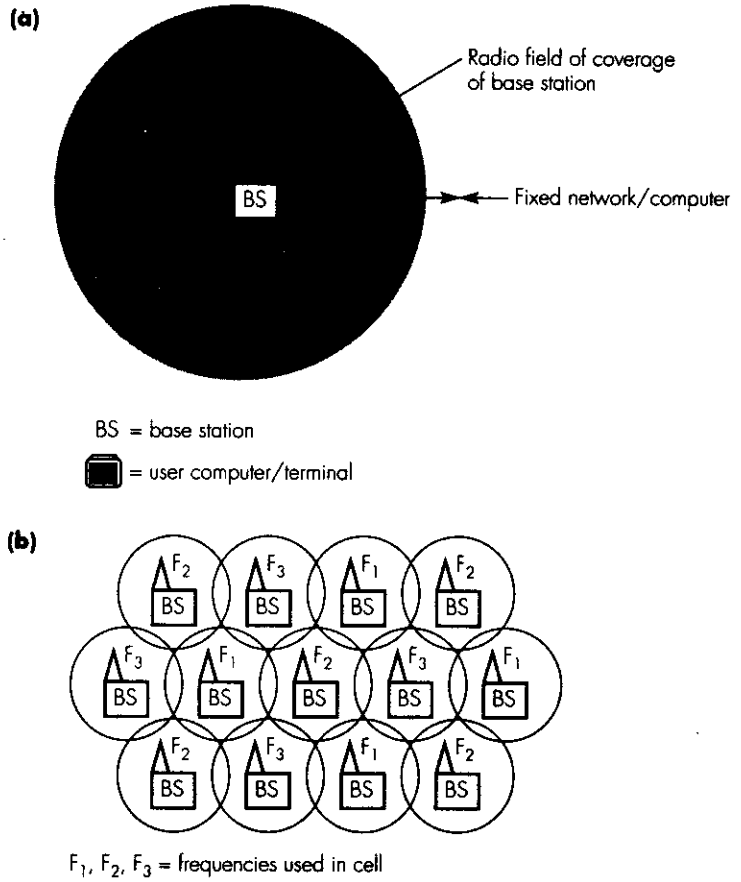


Figure 6.6 Ground-based radio transmission: (a) single cell; (b) multiple cells.

6.2.8 Signal propagation delay

As we saw in Section 1.5.5, there is always a short but finite time delay for a signal (electrical, optical, or radio) to propagate (travel) from one end of a transmission medium to the other. This is known as the **propagation delay**, T_p, of the medium. At best, signals propagate (radiate) through the medium at the speed of light ($3 \times 10^8 \text{ms}^{-1}$). The speed of propagation for twisted-pair wire or coaxial cable is a fraction of this figure. Typically, it is in the region of $2 \times 10^8 \text{ms}^{-1}$, that is, a signal will take 0.5×10^{-8}s to travel 1 m through the medium. Although this may seem insignificant, in some situations the resulting delay is important.

As we explain later in Section 6.7.1, in many instances, on receipt of each block of data/information, an acknowledgment of correct (or otherwise)

receipt is returned to the sender. An important parameter of a transmission link, therefore, is the **round-trip delay** associated with the link, that is, the time delay between the first bit of a block being transmitted by the sender and the last bit of its associated acknowledgement being received. Clearly, this is a function not only of the time taken to transmit the frame at the link bit rate – known as the transmission delay, T_x – but also of the propagation delay of the link, T_p. The relative weighting of the two times varies for different types of link and hence the two times are often expressed as a ratio a such that:

$$a = \frac{T_p}{T_x}$$

where $$T_p = \frac{\text{physical separation } S \text{ in meters}}{\text{velocity of propagation } V \text{ in meters per second}}$$

and $$T_x = \frac{\text{number of bits to be transmitted } N}{\text{link bit rate } R \text{ in bits per second}}$$

Example 6.1

A 1000-bit block of data is to be transmitted between two computers. Determine the ratio of the propagation delay to the transmission delay, a, for the following types of data link:

(i) 100 m of twisted-pair wire and a transmission rate of 10 kbps,
(ii) 10 km of coaxial cable and a transmission rate of 1 Mbps,
(iii) 50 000 km of free space (satellite link) and a transmission rate of 10 Mbps.

Assume that the velocity of propagation of an electrical signal within each type of cable is 2×10^8 ms^{-1}, and that of free space 3×10^8 ms^{-1}.

Answer:

(i) $T_p = \frac{S}{V} = \frac{100}{2 \times 10^8} = 5 \times 10^{-7}$ s

$T_x = \frac{N}{R} = \frac{1000}{10 \times 10^3} = 0.1$ s

$a = \frac{T_p}{T_x} = \frac{5 \times 10^{-7}}{0.1} = 5 \times 10^{-6}$

6.1 Continued

We can conclude from this example:

- If a is less than 1, then the round-trip delay is determined primarily by the transmission delay.
- If a is equal to 1, then both delays have equal effect.
- If a is greater than 1, then the propagation delay dominates.

Furthermore, in case (c) it is interesting to note that, providing blocks are transmitted contiguously, there will be:

$$10 \times 10^6 \times 1.67 \times 10^{-1} = 1.67 \times 10^6 \text{ bits}$$

in transit between the two end systems at any one time, that is, the sending system will have transmitted 1.67×10^6 bits before the first bit arrives at the receiving system. Such links are said, therefore, to have a large **bandwidth/delay product**, where bandwidth relates to the bit rate of the link and delay the propagation delay of the link. We shall discuss these implications further in Section 6.7.1.

6.3 Sources of signal impairment

A selection of the various attenuation and distortion effects that can degrade a signal during transmission are shown in Figure 6.7. Any signal carried on a transmission medium is affected by attenuation, limited bandwidth, delay distortion, and noise. Although all are present and produce a combined effect, we shall consider the source of each impairment separately.

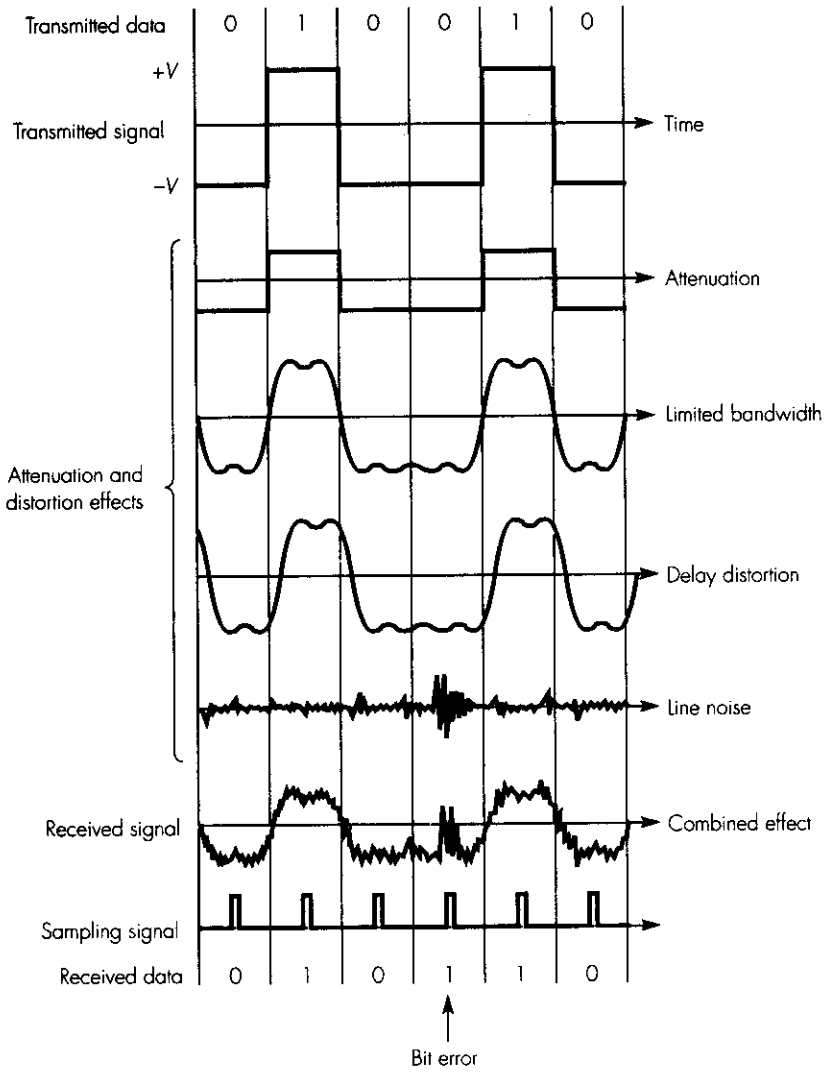


Figure 6.7 Sources of signal impairment.

6.3.1 Attenuation

As a signal propagates along a transmission medium (line) its amplitude decreases. This is known as **signal attenuation**. Normally, to allow for attenuation, a limit is set on the length of the cable that can be used to ensure that the receiver circuitry can reliably detect and interpret the received attenuated signal. If the cable is longer, one or more **amplifiers** – also known as **repeaters** – are inserted at intervals along the cable to restore the received signal to its original level.

Because there is a small but finite *capacitance* between the two wires making up a transmission line, signal attenuation increases as a function of frequency. Hence, since a signal comprises a range of frequencies, the signal is also distorted. To overcome this effect, the amplifiers are designed to amplify different frequency signals by varying amounts. Alternatively, devices known as **equalizers** are used to equalize the attenuation across a defined band of frequencies.

We measure both attenuation and amplification – also known as **gain** – in **decibels** (**dB**). If we denote the transmitted signal power level by P_1 and the received power by P_2, then

$$\text{Attenuation} = 10 \log_{10} \frac{P_1}{P_2} \text{dB}$$

and

$$\text{Amplification} = 10 \log_{10} \frac{P_2}{P_1} \text{dB}$$

Since both P_1 and P_2 have the same unit of **watts**, then decibels are dimensionless and simply a measure of the relative magnitude of the two power levels. The use of logarithms means that the overall attenuation/ amplification of a multisection transmission channel can be determined simply by summing together the attenuation/amplification of the individual sections.

More generally, as we shall expand upon in Section 6.5.1, when we transmit binary information over a limited-bandwidth channel such as PSTN, we can often use more than two signal levels. This means that each signal element can represent more than a single binary digit. In general, if the number of signal levels is M, the number of bits per signal element m, is given by:

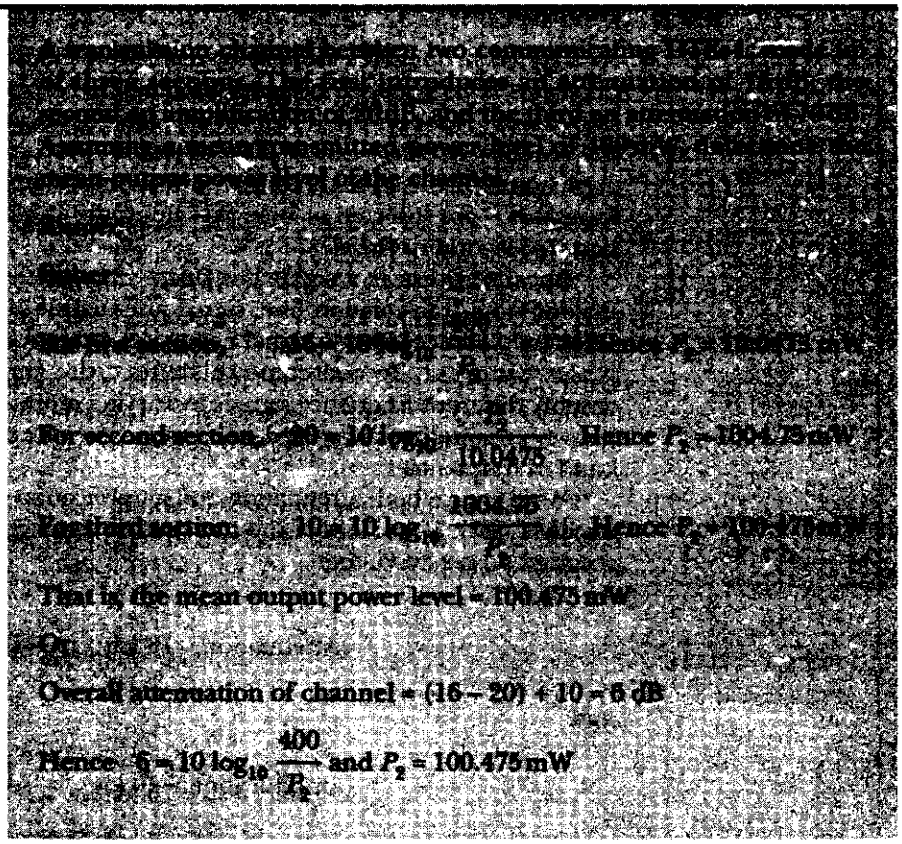
$$m = \log_2 M$$

For example, if four signal levels are used to transmit a data bitstream, then each signal element can be used to transmit two binary digits.

The rate of change of the signal is known as the **signaling rate** (R_s) and is measured in **baud**. It is related to the data bit rate, R, by the following expression:

$$R = R_s \log_2 M$$

Example 6.2



6.3.2 Limited bandwidth

Any communications channel/transmission medium – twisted-pair wire, coaxial cable, radio, and so on – has a defined bandwidth associated with it which specifies the band of sinusoidal frequency components that will be transmitted by the channel unattenuated. Hence when transmitting data over a channel, we need to quantify the effect the bandwidth of the channel will have on the transmitted data signal.

We can use a mathematical technique known as **Fourier analysis** to show that any periodic signal – that is, a signal which repeats itself at equal time intervals (known as the period) – is made up of an infinite series of sinusoidal frequency components. The period of the signal determines the **fundamental frequency** component: the reciprocal of the period in seconds yields the frequency in **cycles per second** (**Hz**). The other components have frequencies which are multiples of this and these are known as the **harmonics** of the fundamental.

In terms of data transmission, the signals of interest are binary sequences and, although in practice a transmitted binary message may be made up of

randomly varying sequences, for analysis purposes we shall consider selected periodic sequences such as 101010..., 110110..., 11101110..., and so on. In the first example, the sequence 10 repeats with a period of two bit-cell intervals. in the second, the sequence 110 with a period of three intervals, and so on. We can deduce from this that the sequence 101010... has the shortest period and will yield the highest fundamental frequency component. This means that other sequences will yield frequencies less than this and hence, for analysis purposes, the sequence with the shortest period is often referred to as the **worst-case sequence.**

For transmission purposes there are two basic binary signal types: **unipolar** and **bipolar**. An example of each is shown in Figure 6.8(a). With a unipolar signal, the signal amplitude varies between a positive voltage level – say, $+V$ – and 0 volts. We call such signals **return-to-zero (RZ)** signals. With a bipolar signal, the signal amplitude varies between a positive and a negative voltage level – say $+V$ and $-V$. These signals are **non-return-to-zero (NRZ)** signals. A unipolar signal has a mean signal level of $V/2$ while a bipolar signal has a mean of zero. The amplitude variation of a unipolar signal is V while for a bipolar signal it is $2V$. These differences yield slightly different Fourier series which, for the two signal types, are as follows:

$$\text{Unipolar} \quad v(t) = \frac{V}{2} + \frac{2V}{\pi}\left\{\cos \omega_0 t - \frac{1}{3}\cos 3\omega_0 t + \frac{1}{5}\cos 5\omega_0 t - \ldots\right\}$$

$$\text{Bipolar} \quad v(t) = \frac{4V}{\pi}\left\{\cos \omega_0 t - \frac{1}{3}\cos 3\omega_0 t + \frac{1}{5}\cos 5\omega_0 t - \ldots\right\}$$

where: $v(t)$ is the voltage signal representation as a function of time,
ω_0 is the fundamental frequency component in radians per second,
$f_0 = \omega_0/2\pi$ is the fundamental frequency in Hz,
$T = 1/f_0$ is the period of the fundamental frequency in seconds.

We can deduce from these expressions that any periodic binary sequence is made up of an infinite series of sinusoidal signals including the fundamental frequency component, f_0, a third harmonic component, $3f_0$, a fifth harmonic, $5f_0$, and so on.

Note that for binary sequences only the odd harmonic components are present and that their amplitude diminishes with increasing frequency. To illustrate the effect of this, the fundamental together with the third and fifth harmonics of a bipolar signal are shown in diagrammatic form in Figure 6.8(b).

Since a communications channel has a limited bandwidth, as we can deduce from the above, when a binary data signal is transmitted over a channel, only those frequency components that are within the channel bandwidth will be received. The effect of this on the example binary signal 101010... is

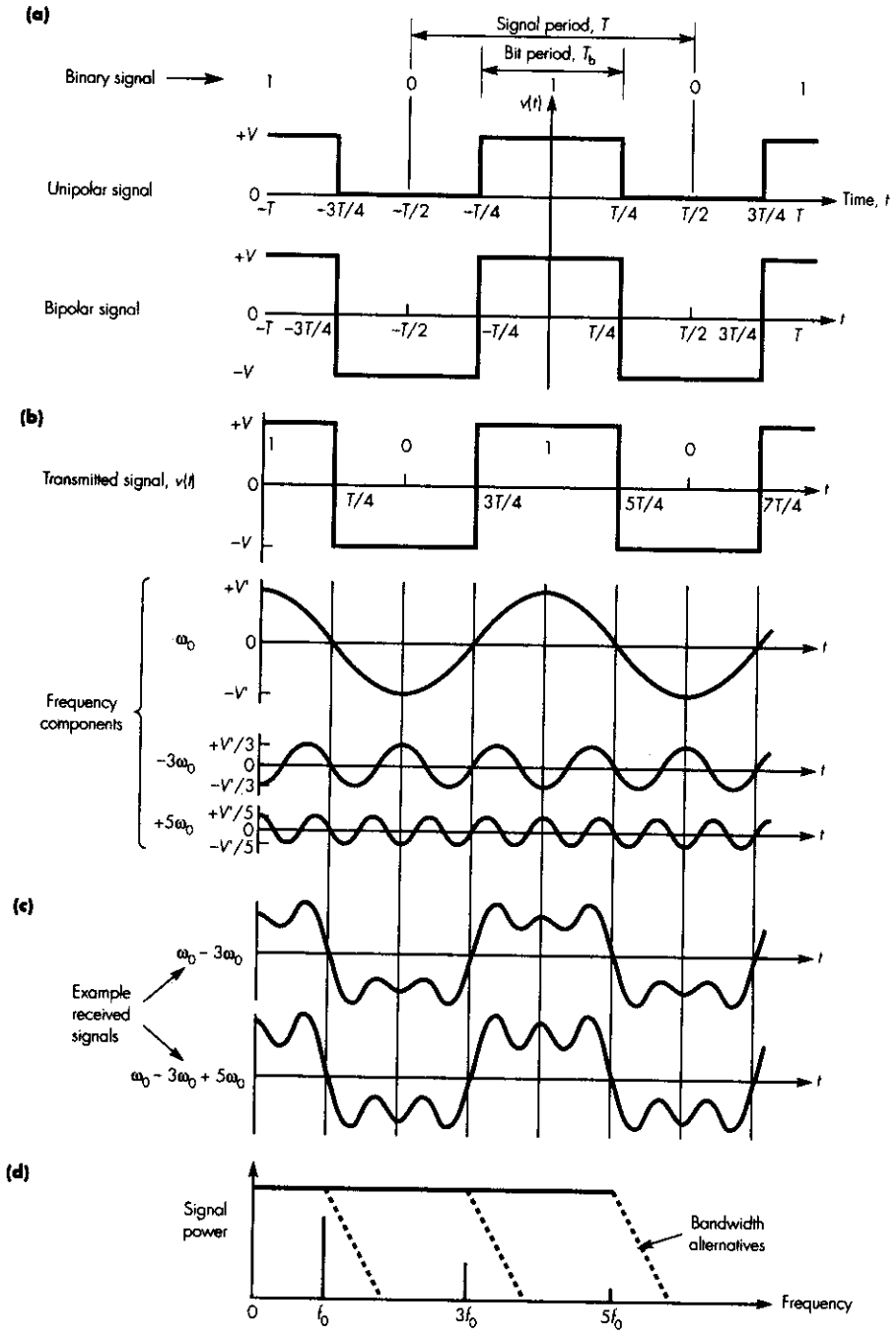
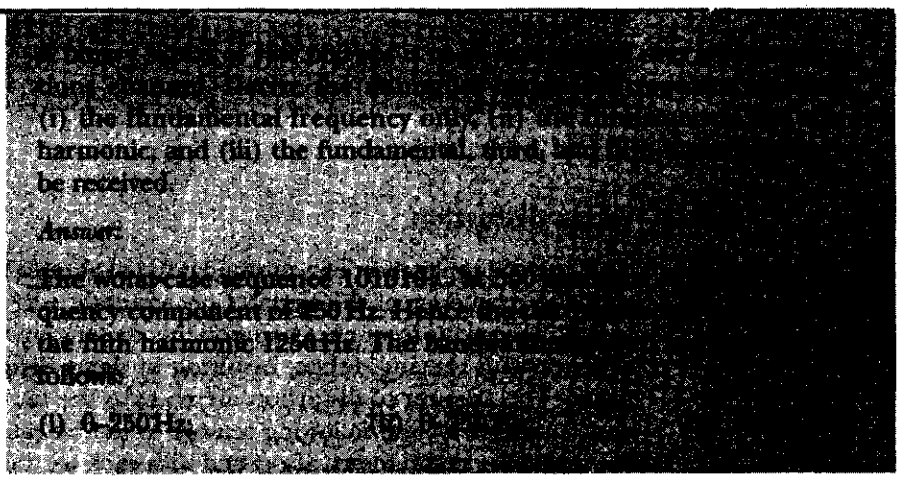


Figure 6.8 Effect of limited bandwidth: (a) alternative binary signals; (b) frequency components of a periodic binary sequence; (c) examples of received signals; (d) bandwidth representations.

shown in Figure 6.8(c). As we can see, the larger the channel bandwidth, the more higher-frequency components are received and hence the closer is the received signal to the original (transmitted) signal.

As the bandwidth of a channel is measured in Hz, normally, it is shown as a function of frequency as illustrated in Figure 6.8(d). In the figure, there are three alternative bandwidths: the first allows sinusoidal signals of frequencies up to f_0 to pass unattenuated, the second up to $3f_0$, and the third up to $5f_0$. In practice, however, when only a two-level (binary) signal is being transmitted, the receiver simply samples the received signal at the center of each bit-cell interval. This means that the receiver need only discriminate between the binary 1 and 0 levels at the sampling instant and the exact shape of the signal outside these instances is less important. As we discussed earlier, the sequence 101010... generates the highest-frequency components while a sequence of all 1s or all 0s is equivalent to a zero frequency of the appropriate amplitude. Hence a channel with a bandwidth from 0 Hz to a frequency (in Hz) equal to half the bit rate (in bps) – that is, a channel that passes only frequency components up to the fundamental frequency of the binary sequence 101010 – can often give a satisfactory performance.

Example 6.3



As we mentioned earlier, it is also possible to transmit more than one bit with each change in signal amplitude thereby increasing the data bit rate. Nevertheless, the bandwidth of the channel always limits the maximum data rate that can be obtained. A formula derived by **Nyquist** for determining the maximum information transfer rate of a noiseless transmission channel, C, is given by the expression:

$$C = 2W\log_2 M$$

where W is the bandwidth of the channel in Hz and M is the number of levels per signaling element. In practice, as we shall see in Sections 6.4 and 6.5, additional bits are added for transmission control purposes and hence the useful data rate is often less than the transmitted bit rate. So when we transmit information over a communications channel, three rates are involved – the signaling rate, the bit rate, and the data rate – all of which may be the same or different.

Example 6.4

6.3.3 Delay distortion

The rate of propagation of a sinusoidal signal along a transmission line varies with the frequency of the signal. Consequently, when we transmit a digital signal the various frequency components making up the signal arrive at the receiver with varying delays, resulting in **delay distortion** of the received signal. The amount of distortion increases as the bit rate of the transmitted data increases for the following reason: as the bit rate increases, so some of the frequency components associated with each bit transition are delayed and start to interfere with the frequency components associated with a later bit. Delay distortion is also known as **intersymbol interference**; its effect is to vary the bit transition instants of the received signal. Since the received signal is normally sampled at the nominal center of each bit cell, this can lead to incorrect interpretation of the received signal as the bit rate increases.

We can best observe the level of intersymbol interference associated with a transmission channel by means of an **eye diagram**, an example of which is shown in Figure 6.9. This diagram is obtained by displaying the signal received from the channel on an oscilloscope which is triggered by the transitions in the signal. Hence, assuming the received signal contains random

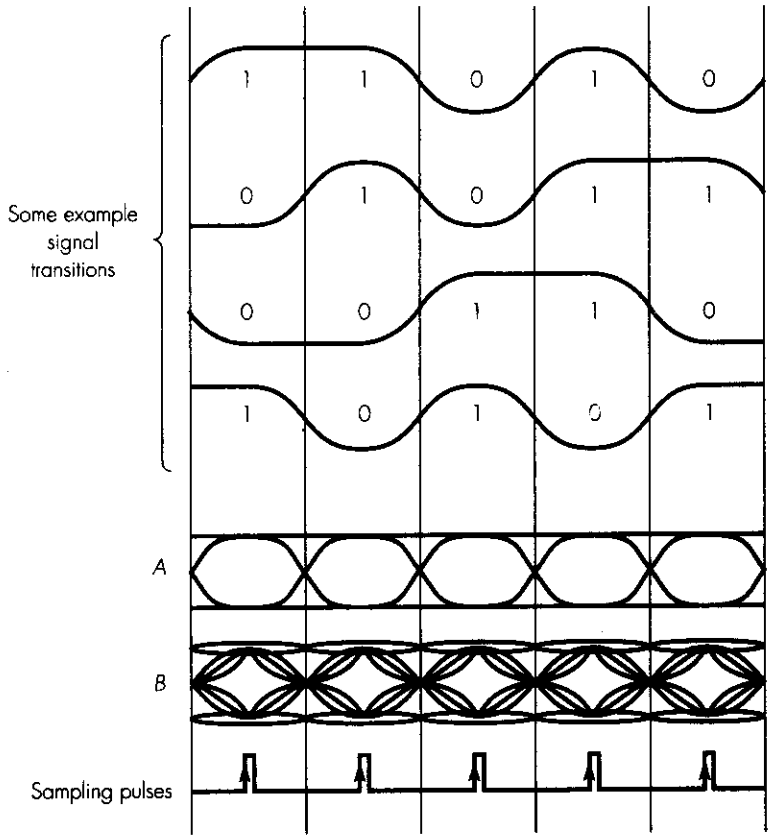


Figure 6.9 Examples of (binary) eye diagrams resulting from intersymbol interference: A, ideal; B, typical.

binary 1 and 0 signal transitions, the oscilloscope will display all the possible signals superimposed on one another. Two examples are shown in the figure. In the absence of intersymbol interference the signal will be of the form shown as *A*, while with interference the signal will be of the form shown as *B*. We can deduce that the higher the level of interference, the smaller the central section – known as the **eye** – becomes.

6.3.4 Noise

In the absence of a signal, a transmission line or channel ideally has zero electrical signal present. In practice, however, there are random perturbations on the line even when no signal is being transmitted. These are caused by what is known as **thermal noise**, which is present in all types of transmission media. It is called **line noise** and, in the limit, as a transmitted signal becomes

attenuated, its level is reduced to that of the line (background) noise. An important parameter associated with the received signal, therefore, is the ratio of the average power in the received signal, S, to the power in the noise level, N. The ratio S/N is called the **signal-to-noise ratio** (**SNR**) and is normally expressed in decibels, that is:

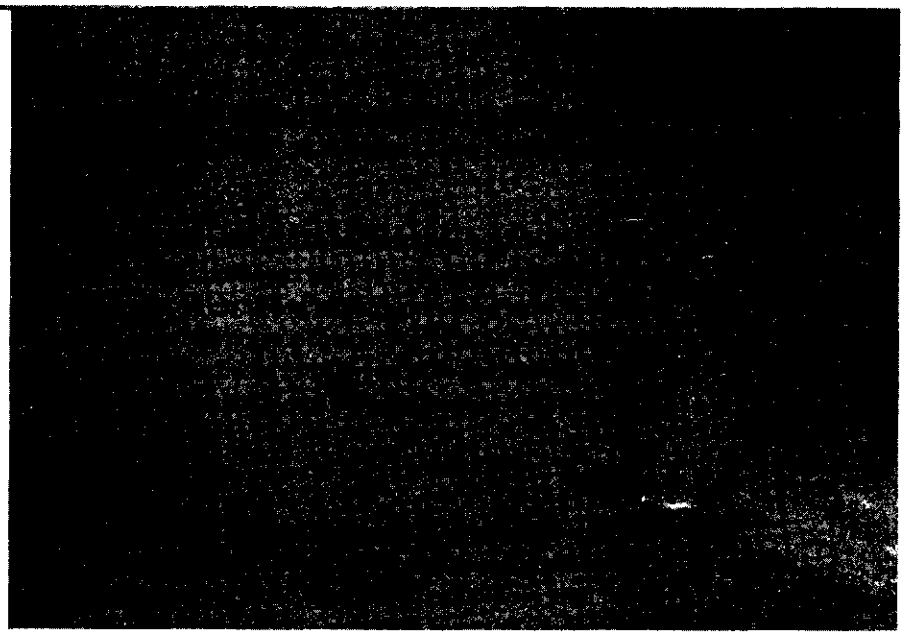
$$\text{SNR} = 10 \log_{10} \frac{S}{N} \text{ dB}$$

Clearly, a high SNR means a high power signal relative to the prevailing noise level, resulting in a good-quality signal. Conversely, a low SNR means a low-quality signal. The theoretical maximum information (data) rate of a transmission medium is related to the SNR and we can determine this rate using a formula attributed to Shannon and Hartley. This is known as the **Shannon–Hartley law**, which states:

$$C = W \log_2 \left(1 + \frac{S}{N}\right) \text{ bps}$$

where C is the information (data) rate in bps, W is the bandwidth of the line/channel in Hz, S is the average signal power in watts, and N is the random noise power in watts.

Example 6.5



As we have explained, thermal noise is present in a line even when nothing is being transmitted. In addition, there are other noise signals present that are caused by electrical activity external to the transmission line. We identified one source of this, crosstalk, in Section 6.2.1 when we discussed open-wire and twisted pair transmission lines. Crosstalk is caused by unwanted electrical coupling between adjacent lines. This coupling results in a signal that is being transmitted in one line being picked up by adjacent lines as a small but finite (noise) signal.

There are several types of crosstalk but in most cases the most limiting impairment is **near-end crosstalk** or **NEXT**. This is also known as **self-crosstalk** since it is caused by the strong signal output by a transmitter circuit being coupled (and hence interfering) with the much weaker signal at the input to the local receiver circuit. As we showed in Figure 6.2, the received signal is normally significantly attenuated and distorted and hence the amplitude of the coupled signal from the transmit section can be comparable with the amplitude of the received signal.

Special integrated circuits known as **adaptive NEXT cancelers** are now used to overcome this type of impairment. A typical arrangement is shown in Figure 6.10. The canceler circuit adaptively forms an attenuated replica of the crosstalk signal that is coupled into the receive line from the local transmitter and this is subtracted from the received signal. Such circuits are now used in many applications involving UTP cable, for example, to transmit data at high bit rates.

6.4 Asynchronous transmission

With asynchronous transmission, each character or byte that makes up a block/message is treated independently for transmission purposes. Hence it can be used both for the transfer of, say, single characters that are entered at a keyboard, or for the transfer of blocks of characters/bytes across a low bit rate transmission line/channel.

Within end systems, all information is transferred between the various circuits and subsystems in a word or byte parallel mode. Consequently, since all transfers that are external to the system are carried out bit-serially, the transmission control circuit on the network interface card (NIC) must perform the following functions:

- parallel-to-serial conversion of each character or byte in preparation for its transmission on the line;
- serial-to-parallel conversion of each received character or byte in preparation for its storage and processing in the receiving end system;
- a means for the receiver to achieve bit, character, and frame synchronization;
- the generation of suitable error check digits for error detection and the detection of such errors at the receiver should they occur.

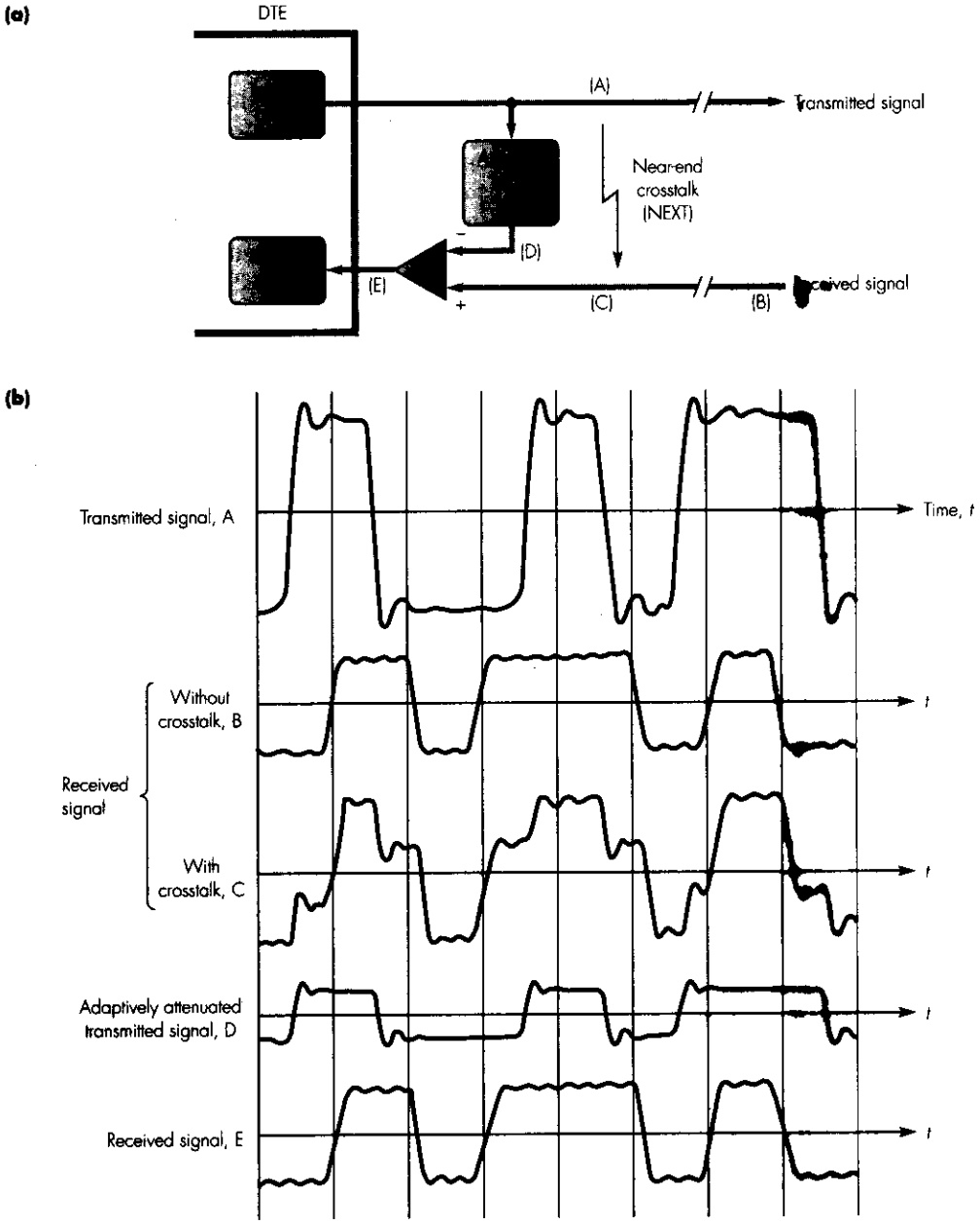


Figure 6.10 Adaptive NEXT cancelers: (a) circuit schematic; (b) example waveforms.

As we show in Figure 6.11(a), parallel-to-serial conversion is performed by a **parallel-in, serial-out (PISO) shift register**. This, as its name implies, allows a complete character or byte to be loaded in parallel and then shifted out bit-serially. Similarly, serial-to-parallel conversion is performed by a **serial-in, parallel-out (SIPO) shift register** which executes the reverse function.

To achieve bit and character synchronization, we must set the receiving transmission control circuit (which is normally programmable) to operate with the same characteristics as the transmitter in terms of the number of bits per character and the bit rate being used.

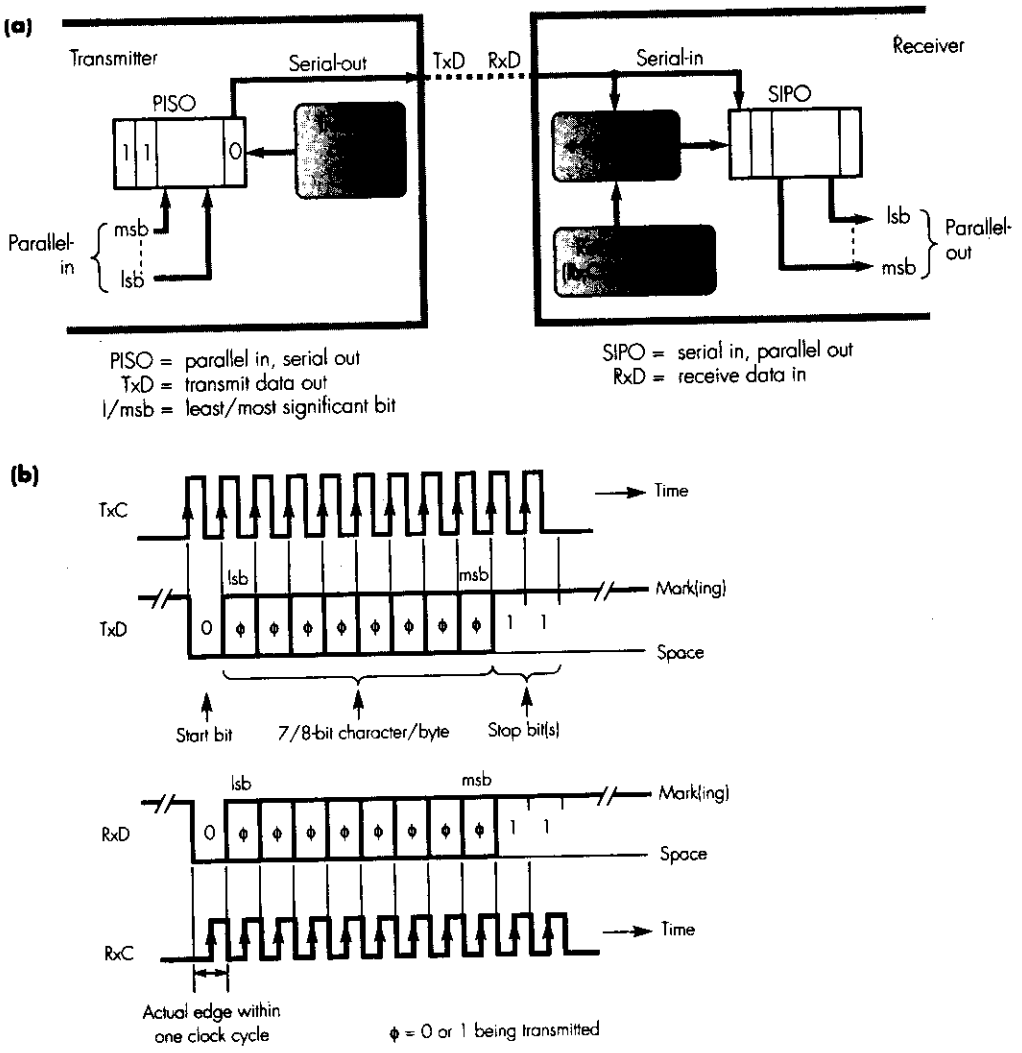


Figure 6.11 Asynchronous transmission: (a) principle of operation; (b) timing principles.

6.4.1 Bit synchronization

In asynchronous transmission, the receiver clock (which is used to sample and shift the incoming signal into the SIPO shift register) runs asynchronously with respect to the incoming signal. In order for the reception process to work reliably, we must devise a scheme whereby the local (asynchronous) receiver clock samples (and hence shifts into the SIPO shift register) the incoming signal as near to the center of the bit cell as possible.

To achieve this, each character/byte to be transmitted is encapsulated between a *start bit* and one or two *stop bits*. As we show in Figure 6.11(b), the start bit is a binary 0 (and known as a *space*) and the stop bit a binary 1 (and known as a *mark*). When transmitting a block comprising a string of characters/bytes, the start bit of each character/byte immediately follows the stop bit(s) of the previous character/byte. When transmitting random characters, however, the line stays at the stop/1 level until the next character is to be transmitted. Hence between blocks (or after each character), the line is said to be *marking* (time).

The use of a start and stop bit per character/byte of different polarities means that, irrespective of the binary contents of each character/byte, there is a guaranteed $1 \rightarrow 0$ transition at the start of each new character/byte. A local receiver clock of N times the transmitted bit rate ($N = 16$ is common) is used and each new bit is shifted into the SIPO shift register after N cycles of this clock. The first $1 \rightarrow 0$ transition associated with the start bit of each character is used to start the counting process. Each bit (including the start bit) is sampled at (approximately) the center of each bit cell. After the first transition is detected, the signal (the start bit) is sampled after $N/2$ clock cycles and then subsequently after N clock cycles for each bit in the character and also the stop bit(s). Three examples of different clock rate ratios are shown in Figure 6.12.

Remembering that the receiver clock (RxC) is running asynchronously with respect to the incoming signal (RxD), the relative positions of the two signals can be anywhere within a single cycle of the receiver clock. Those shown in the figure are just arbitrary positions. Nevertheless, we can deduce from these examples that the higher the clock rate ratio, the nearer the sampling instant will be to the nominal bit cell center. Because of this mode of operation, the maximum bit rate normally used with asynchronous transmission is 19.2 kbps.

6.4.2 Character synchronization

As we indicated in Section 6.4 above, the receiving transmission control circuit is programmed to operate with the same number of bits per character and the same number of stop bits as the transmitter. After the start bit has been detected and received, the receiver achieves character synchronization simply by counting the programmed number of bits. It then transfers the received character/byte into a local **buffer register** and signals to the control-

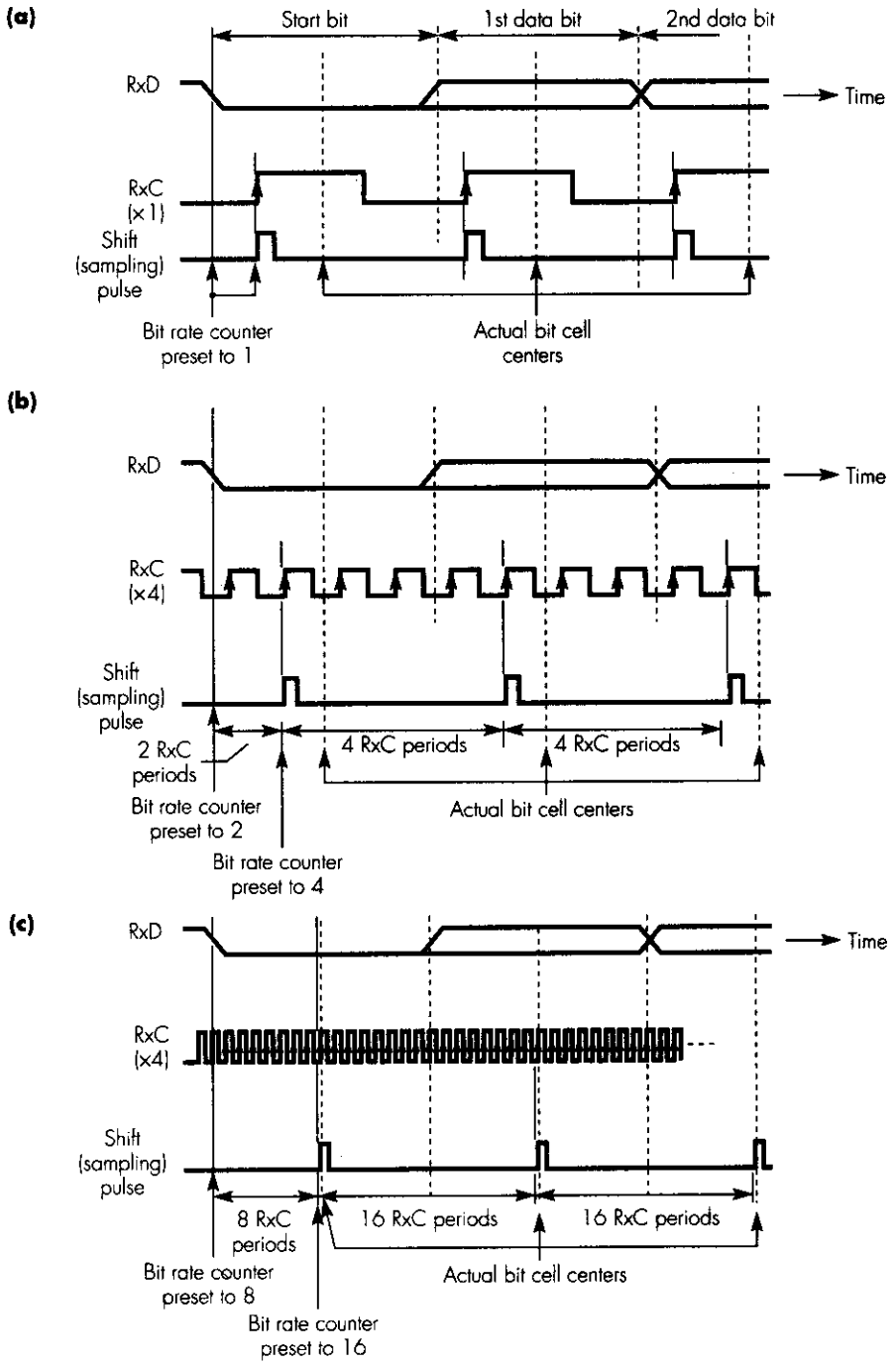
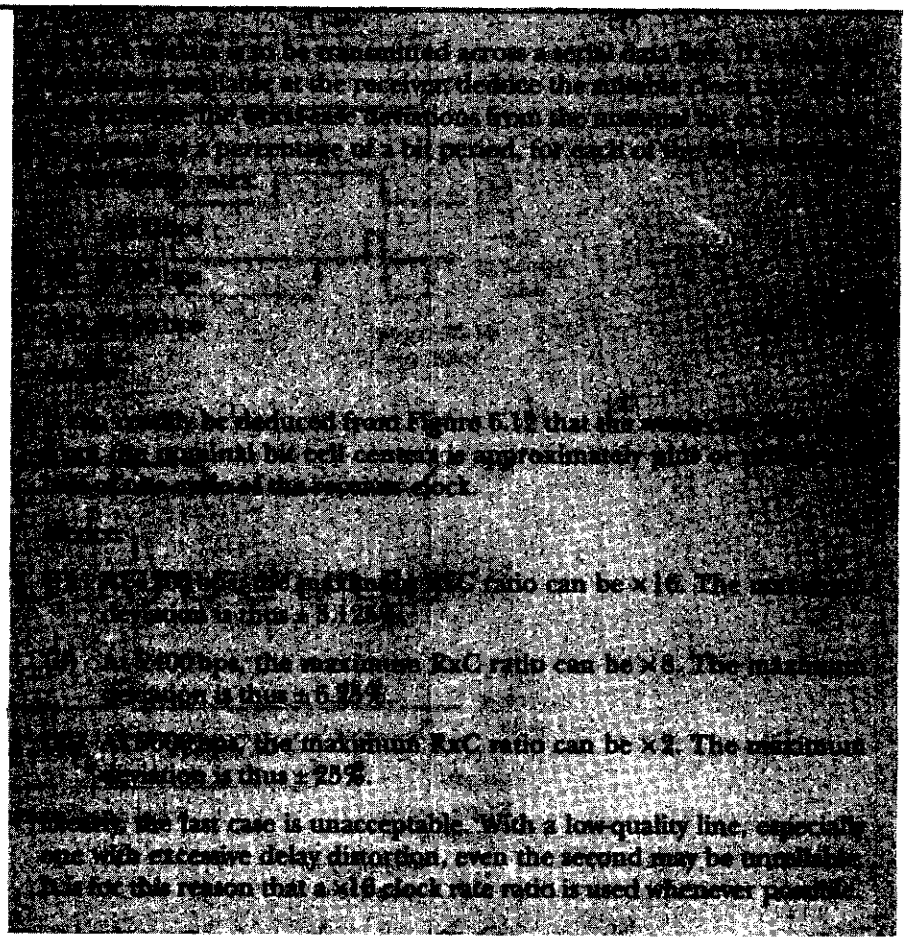


Figure 6.12 Examples of three different receiver clock rate ratios: (a) ×1; (b) ×4; (c) ×16.

Example 6.6



ling device (a microprocessor, for example) on the NIC that a new character/byte has been received. It then awaits the next line signal transition that indicates a new start bit (and hence character) is being received.

6.4.3 Frame synchronization

When messages comprising blocks of characters or bytes – normally referred to as **information frames** – are being transmitted, in addition to bit and character synchronization, the receiver must be able to determine the start and end of each frame. This is known as frame synchronization.

The simplest method of transmitting blocks of printable characters is to encapsulate the complete block between two special (nonprintable) transmission control characters: STX (start-of-text) which indicates the start of a new frame after an idle period, and ETX (end-of-text) which indicates the end of

the frame. As the frame contents consist only of printable characters, the receiver can interpret the receipt of an STX character as signaling the start of a new frame and an ETX character as signaling the end of the frame. This is shown in Figure 6.13(a).

Although the scheme shown is satisfactory for the transmission of blocks of printable characters, when transmitting blocks that comprise strings of bytes (for example, the contents of a file containing compressed speech or video), the use of a single ETX character to indicate the end of a frame is not sufficient. In the case of a string of bytes, one of the bytes might be the same as an ETX character, which would cause the receiver to terminate the reception process abnormally.

To overcome this problem, when transmitting this type of data the two transmission control characters STX and ETX are each preceded by a third

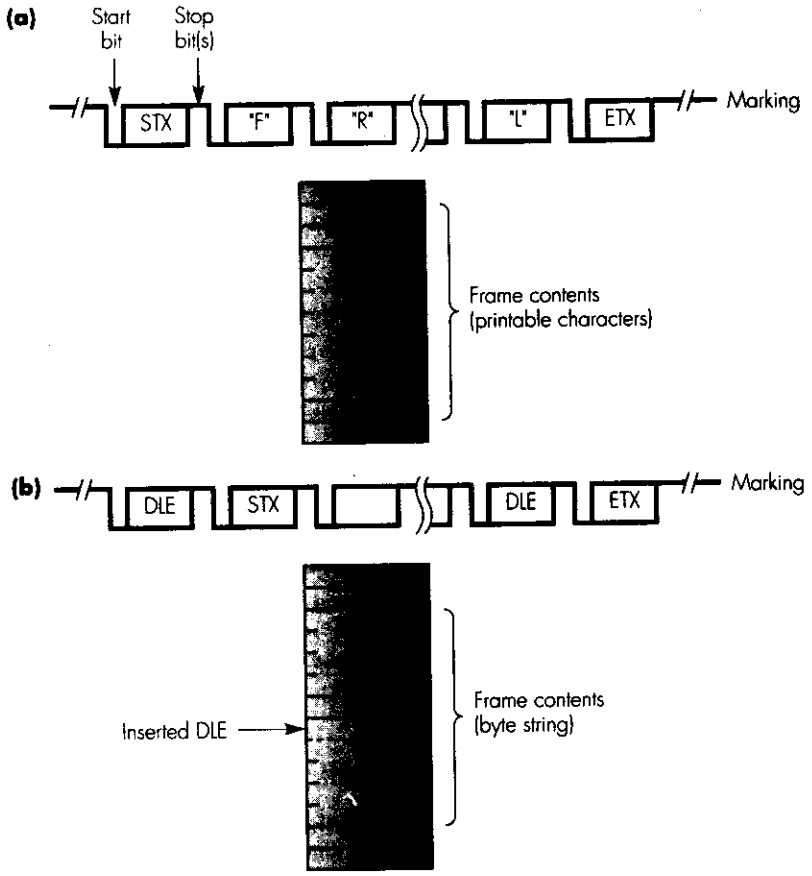


Figure 6.13 Frame synchronization with different frame contents: (a) printable characters; (b) string of bytes.

transmission control character known as **data link escape** (**DLE**). The modified format of a frame is then as shown in Figure 6.13(b).

Remember that the transmitter knows the number of bytes in each frame to be transmitted. After transmitting the start-of-frame sequence (DLE-STX), the transmitter inspects each byte in the frame prior to transmission to determine if it is the same as the DLE character pattern. If it is, irrespective of the next byte, a second DLE character (byte) is transmitted before the next byte. This procedure is repeated until the appropriate number of bytes in the frame have been transmitted. The transmitter then signals the end of the frame by transmitting the unique DLE-STX sequence.

This procedure is known as **character** or **byte stuffing**. On receipt of each byte after the DLE-STX start-of-frame sequence, the receiver determines whether it is a DLE character (byte). If it is, the receiver then processes the next byte to determine whether that is another DLE or an ETX. If it is a DLE, the receiver discards it and awaits the next byte. If it is an ETX, this can reliably be taken as being the end of the frame.

6.5 Synchronous transmission

The use of an additional start bit and one or more stop bits per character or byte means that asynchronous transmission is relatively inefficient in its use of transmission capacity, especially when transmitting messages that comprise large blocks of characters or bytes. Also, the bit (clock) synchronization method used with asynchronous transmission becomes less reliable as the bit rate increases. This results, firstly, from the fact that the detection of the first start bit transition is only approximate and, secondly, although the receiver clock operates at N times the nominal transmit clock rate, because of tolerances there are small differences between the two which can cause the sampling instant to drift during the reception of a character or byte. We normally use synchronous transmission to overcome these problems. As with asynchronous transmission, however, we must adopt a suitable method to enable the receiver to achieve bit (clock) character (byte) and frame (block) synchronization. In practice, there are two synchronous transmission control schemes: character-oriented and bit-oriented. We shall discuss each separately but, since they both use the same bit synchronization methods, we shall discuss these methods first.

6.5.1 Bit synchronization

Although we often use the presence of a start bit and stop bit(s) with each character to discriminate between asynchronous and synchronization transmission, the fundamental difference between the two methods is that with asynchronous transmission the receiver clock runs asynchronously (unsynchronized) with respect to the incoming (received) signal, whereas with synchronous transmission the receiver clock operates in synchronism with the received signal.

As we have just indicated, start and stop bits are not used with synchronous transmission. Instead each frame is transmitted as a contiguous stream of binary digits. The receiver then obtains (and maintains) bit synchronization in one of two ways. Either the clock (timing) information is embedded into the transmitted signal and subsequently extracted by the receiver, or the receiver has a local clock (as with asynchronous transmission) but this time it is kept in synchronism with the received signal by a device known as a **digital phase-lock-loop** (**DPLL**). As we shall see, the DPLL exploits the $1 \rightarrow 0$ or $0 \rightarrow 1$ bit transitions in the received signal to maintain bit (clock) synchronism over an acceptably long period. Hybrid schemes that exploit both methods are also used. The principles of operation of both schemes are shown in Figure 6.14.

Clock encoding and extraction

The alternative methods of embedding timing (clock) information into a transmitted bitstream are shown in Figure 6.15. The scheme shown in part (a) is called **Manchester encoding** and that in part (b) **differential Manchester encoding**.

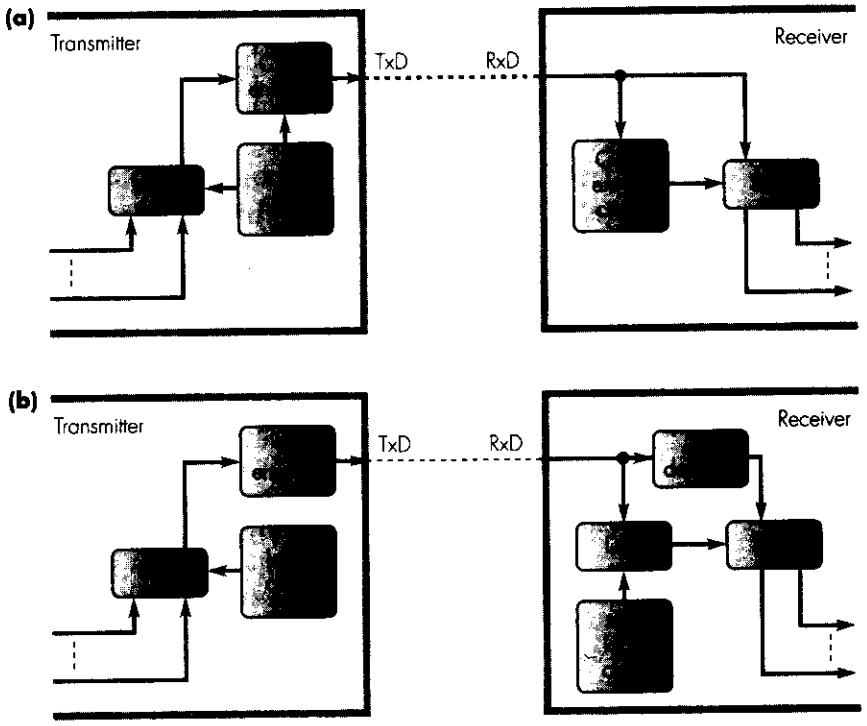


Figure 6.14 Alternative bit/clock synchronization methods with synchronuous transmission: (a) clock encoding; (b) digital phase-lock-loop (DPLL).

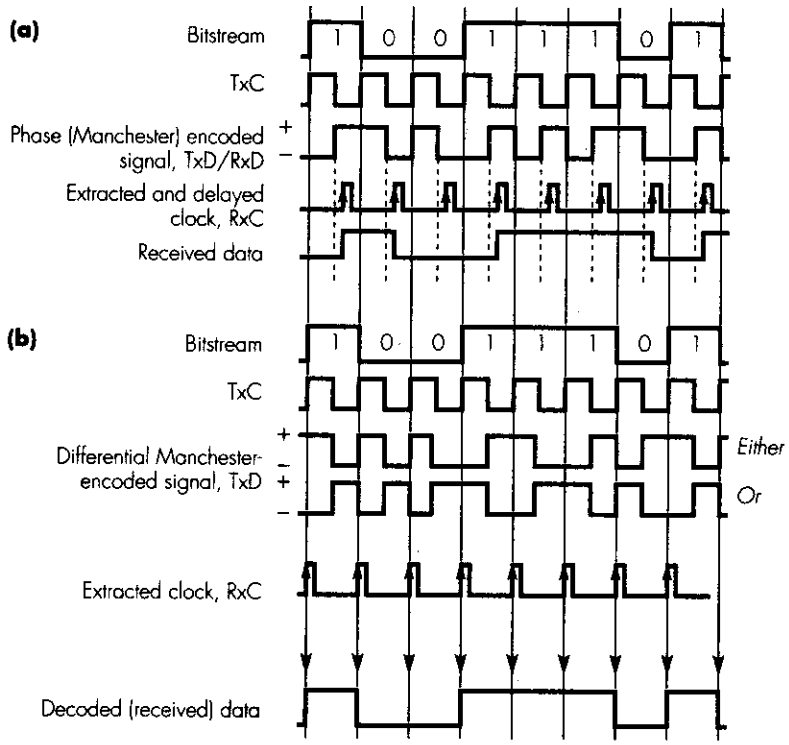


Figure 6.15 Synchronous transmission clock encoding methods: (a) Manchester; (b) differential Manchester.

As we can see, with Manchester encoding each bit is encoded as either a low-high signal (binary 1) or a high-low signal (binary 0), both occupying a single bit-cell period. Also, there is always a transition (high-low or low-high) at the center of each bit cell. It is this that is used by the clock extraction circuit to produce a clock pulse which is then delayed to the center of the second half of the bit cell. At this point the received (encoded) signal is either high (for binary 1) or low (for binary 0) and hence the correct bit is sampled and shifted into the SIPO shift register.

The scheme shown in Figure 6.15(b) is differential Manchester encoding. This differs from Manchester encoding in that although there is still a transition at the center of each bit cell, a transition at the start of the bit cell occurs only if the next bit to be encoded is a 0. This has the effect that the encoded output signal may take on one of two forms depending on the assumed start level (high or low). As we can see, however, one is simply an inverted version of the other and this is the origin of the term "differential". As we show later in Figure 7.12(c), and explain in the accompanying text, a differential driver circuit produces a pair of differential signals and the differ-

ential receiver operates by determining the difference between these two signals. For example, if the two signals each vary between $+V$ and $-V$, then the difference would be $+2V$ and $-2V$. The extracted clock is generated at the start of each bit cell. At this point the received (encoded) signal either changes – for example, from $+2V$ to $-2V$ or $-2V$ to $+2V$ in which case a binary 0 is shifted into the SIPO – or remains at the same level, in which case a binary 1 is shifted into the SIPO.

The two Manchester encoding schemes are **balanced codes** which means there is no mean (DC) value associated with them. This is so since a string of binary 1s (or 0s) will always have transitions associated with them rather than a constant (DC) level. This is an important feature since it means that the received signal can be **AC coupled** to the receiver electronics using a transformer. The receiver electronics can then operate using its own power supply since this is effectively isolated from the power supply of the transmitter.

Digital phase-lock-loop

An alternative approach to encoding the clock in the transmitted bit stream is to utilize a stable clock source at the receiver which is kept in time synchronism with the incoming bit stream. However, as there are no start and stop bits with a synchronous transmission scheme, we must encode the information in such a way that there are always sufficient bit transitions ($1 \rightarrow 0$ or $0 \rightarrow 1$) in the transmitted waveform to enable the receiver clock to be resynchronized at frequent intervals. One approach is to pass the data to be transmitted through a **scrambler** which randomizes the transmitted bitstream so removing contiguous strings of 1s or 0s. Alternatively, the data may be encoded in such a way that suitable transitions will always be present.

The bit pattern to be transmitted is first differentially encoded as shown in Figure 6.16(a). We refer to the resulting encoded signal as a **non-return-to-zero-inverted (NRZI)** waveform. With NRZI encoding the signal level (1 or 0) does not change for the transmission of a binary 1, whereas a binary 0 causes a change. This means that there will always be bit transitions in the incoming signal of an NRZI waveform, providing there are no contiguous streams of binary 1s. On the surface, this may seem no different from the normal NRZ waveform but, as we shall describe in Section 6.5.3, if a bit-oriented scheme with zero bit insertion is used, an active line will always have a binary 0 in the transmitted bitstream at least every five bit cells. Consequently, the resulting waveform will contain a guaranteed number of transitions, since long strings of 0s cause a transition every bit cell. This enables the receiver to adjust its clock so that it is in synchronism with the incoming bitstream.

The circuit used to maintain bit synchronism is known as a digital phase-lock-loop and is shown in Figure 6.16(b). A crystal-controlled oscillator (clock source), which can hold its frequency sufficiently constant to require only very small adjustments at irregular intervals, is connected to the DPLL. Typically, the frequency of the clock is 32 times the bit rate used on the data

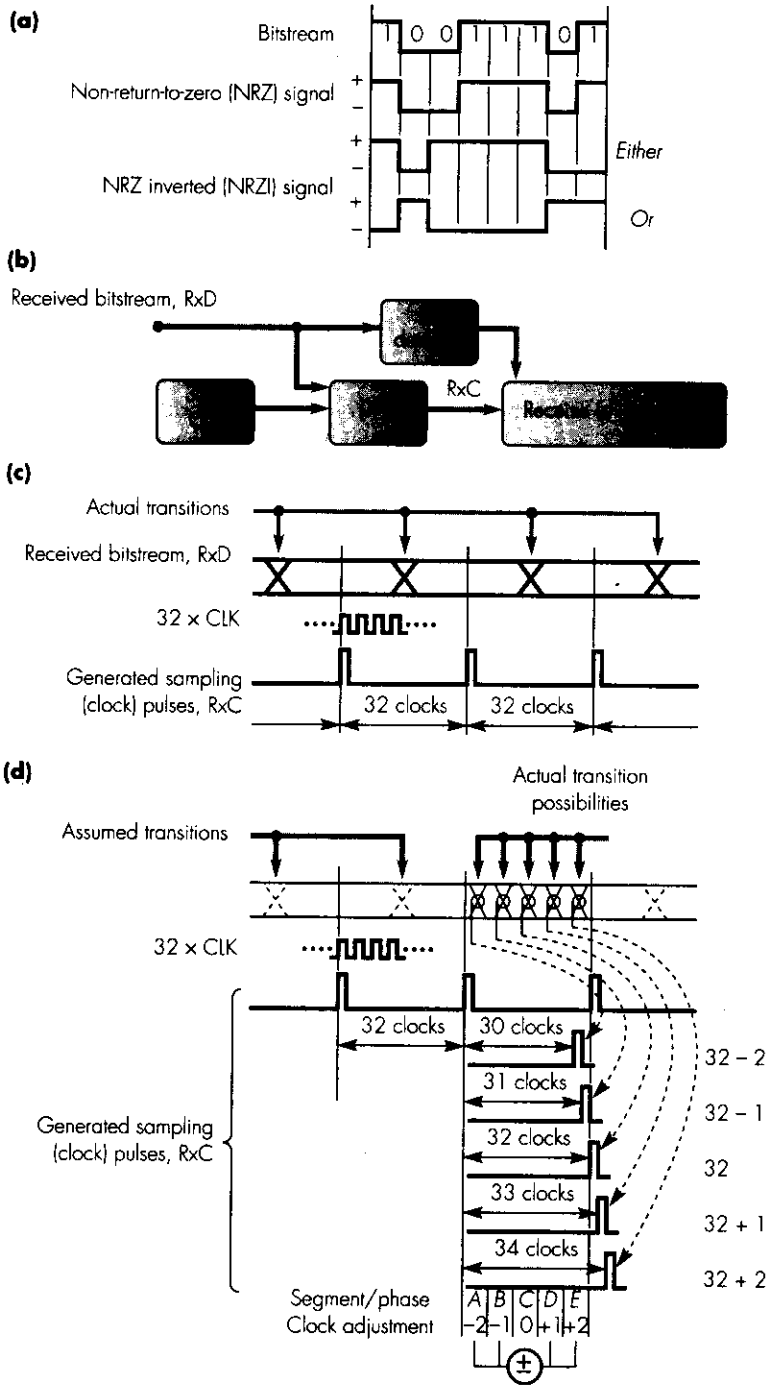


Figure 6.16 DPLL operation: (a) bit encoding; (b) circuit schematic; (c) in phase; (d) clock adjustment rules.

link and is used by the DPLL to derive the timing interval between successive samples of the received bitstream.

Assuming the incoming bitstream and the local clock are in synchronism, the state (1 or 0) of the incoming signal on the line will be sampled (and hence clocked into the SIPO shift register) at the center of each bit cell with exactly 32 clock periods between each sample. This is shown in Figure 6.16(c).

Now assume that the incoming bitstream and local clock drift out of synchronism because of small variations in the latter. The sampling instant is adjusted in discrete increments as shown in Figure 6.16(d). If there are no transitions on the line, the DPLL simply generates a sampling pulse every 32 clock periods after the previous one. Whenever a transition ($1 \rightarrow 0$ or $0 \rightarrow 1$) is detected, the time interval between the previously generated sampling pulse and the next is determined according to the position of the transition relative to where the DPLL thought it should occur. To achieve this, each bit period is divided into five segments, shown as *A, B, C, D,* and *E* in the figure. For example, a transition occurring during segment *A* indicates that the last sampling pulse was too close to the next transition and hence late. The time period to the next pulse is therefore shortened to 30 clock periods. Similarly, a transition occurring in segment *E* indicates that the previous sampling pulse was too early relative to the transition. The time period to the next pulse is therefore lengthened to 34 clock periods. Transitions in segments *B* and *D* are clearly nearer to the assumed transition and hence the relative adjustments are less (–1 and +1 respectively). Finally a transition in segment *C* is deemed to be close enough to the assumed transition to warrant no adjustment.

In this way, successive adjustments keep the generated sampling pulses close to the center of each bit cell. In practice, the widths of each segment (in terms of clock periods) are not equal. The outer segments (*A* and *E*) being further away from the nominal center, are made longer than the three inner segments. For the circuit shown, a typical division might be $A = E = 10$, $B = D = 5$, and $C = 2$. We can readily deduce that in the worst case the DPLL requires 10 bit transitions to converge to the nominal bit center of a waveform: 5 bit periods of coarse adjustments (±2) and 5 bit periods of fine adjustments (±1). Hence when using a DPLL, it is usual before transmitting the first frame on a line, or following an idle period between frames, to transmit a number of characters/bytes to provide a minimum of 10 bit transitions. Two characters/bytes each composed of all 0s, for example, provide 16 transitions with NRZI encoding. This ensures that the DPLL generates sampling pulses at the nominal center of each bit cell by the time the opening character or byte of a frame is received. We must stress, however, that once in synchronism (lock) only minor adjustments normally take place during the reception of a frame.

We can deduce from Figure 6.16(a), that with NRZI encoding the maximum rate at which the encoded signal changes polarity is one half that of Manchester encoding. If the bit period is *T*, with NRZI encoding the maximum rate is $1/T$, whereas with Manchester encoding it is $2/T$. The maximum rate is known as the **modulation rate**. As we described in Section 6.3.2, the

highest fundamental frequency component of each scheme is $1/T$ and $2/T$ respectively. This means that, for the same data rate, Manchester encoding requires twice the transmission bandwidth of an NRZI encoded signal, that is, the higher the modulation rate, the wider is the required bandwidth.

The effect of this is that Manchester and differential Manchester encoding are both used extensively in applications such as LANs. As we shall expand upon in Chapter 8, LANs operate in a single office or building and hence use relatively short cable runs. This means that even though they operate at high bit rates – for example 10 Mbps and higher – the attenuation and bandwidth of the transmission medium are not generally a problem. In contrast, as we shall expand upon in Chapter 7, in networks such as an ISDN twisted-pair cable is often used with relatively high bit rates and over distances of several kilometers. Hence encoding schemes such as NRZI are used with each bit represented by a full-width pulse. We shall describe a number of examples of each scheme in later chapters.

6.5.2 Character-oriented

As we indicated at the beginning of Section 6.5, there are two types of synchronous transmission control scheme: character-oriented and bit-oriented. Both use the same bit synchronization methods. The major difference between the two schemes is the method used to achieve character and frame synchronization.

Character-oriented transmission is used primarily for the transmission of blocks of characters, such as files of ASCII characters. Since there are no start or stop bits with synchronous transmission, an alternative way of achieving character synchroniztion must be used. To achieve this the transmitter adds two or more transmission control characters, known as **synchronous idle** or **SYN** characters, before each block of characters. These control characters have two functions. Firstly, they allow the receiver to obtain (or maintain) bit synchronization. Secondly, once this has been done, they allow the receiver to start to interpret the received bitstream on the correct character boundaries – **character synchronization**. The general scheme is shown in Figure 6.17.

Part (a) shows that frame synchronization (with character-oriented synchronous transmission) is achieved in just the same way as for asynchronous transmission by encapsulating the block of characters – the frame contents – between an STX-ETX pair of transmission control characters. The SYN control characters used to enable the receiver to achieve character synchronization precede the STX start-of-frame character. Once the receiver has obtained bit synchronization it enters what is known as the **hunt mode**. This is shown in Figure 6.17(b).

When the receiver enters the hunt mode, it starts to interpret the received bitstream in a window of eight bits as each new bit is received. In this way, as each bit is received, it checks whether the last eight bits were equal to the known SYN character. If they are not, it receives the next bit and repeats

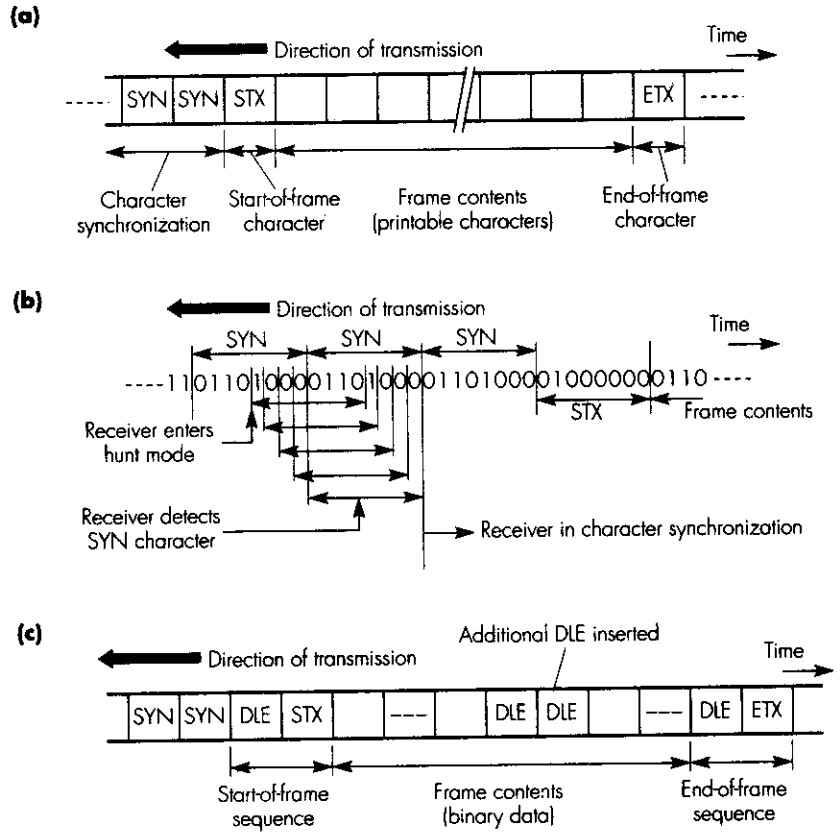


Figure 6.17 Character-oriented synchronous transmission: (a) frame format; (b) character synchronization; (c) data transparency (character stuffing).

the check. If they are, then this indicates it has found the correct character boundary and hence the following characters are then read after each subsequent eight bits have been received.

Once in character synchronization (and hence reading each character on the correct bit boundary), the receiver starts to process each subsequently received character in search of the STX character indicating the start of the frame. On receipt of the STX character, the receiver proceeds to receive the frame contents and terminates this process when it detects the ETX character. On a point-to-point link, the transmitter normally then reverts to sending SYN characters to allow the receiver to maintain synchronism. Alternatively, the above procedure must be repeated each time a new frame is transmitted.

Finally, as we can see in Figure 6.17(c), when binary data is being transmitted, data transparency is achieved in the same way as described previously

by preceding the STX and ETX characters by a DLE (data link escape) character and inserting (stuffing) an additional DLE character whenever it detects a DLE in the frame contents. In this case, therefore, the SYN characters precede the first DLE character.

6.5.3 Bit-oriented

The need for a pair of characters at the start and end of each frame for frame synchronization, coupled with the additional DLE characters to achieve data transparency, means that a character-oriented transmission control scheme is relatively inefficient for the transmission of binary data. Moreover, the format of the transmission control characters varies for different character sets, so the scheme can be used only with a single type of character set, even though the frame contents may be pure binary data. To overcome these problems, a more universal scheme known as **bit-oriented transmission** is now the preferred control scheme as it can be used for the transmission of frames comprising either printable characters or binary data. The main features of the scheme are shown in Figure 6.18(a). It differs mainly in the way the start and end of each frame is signaled.

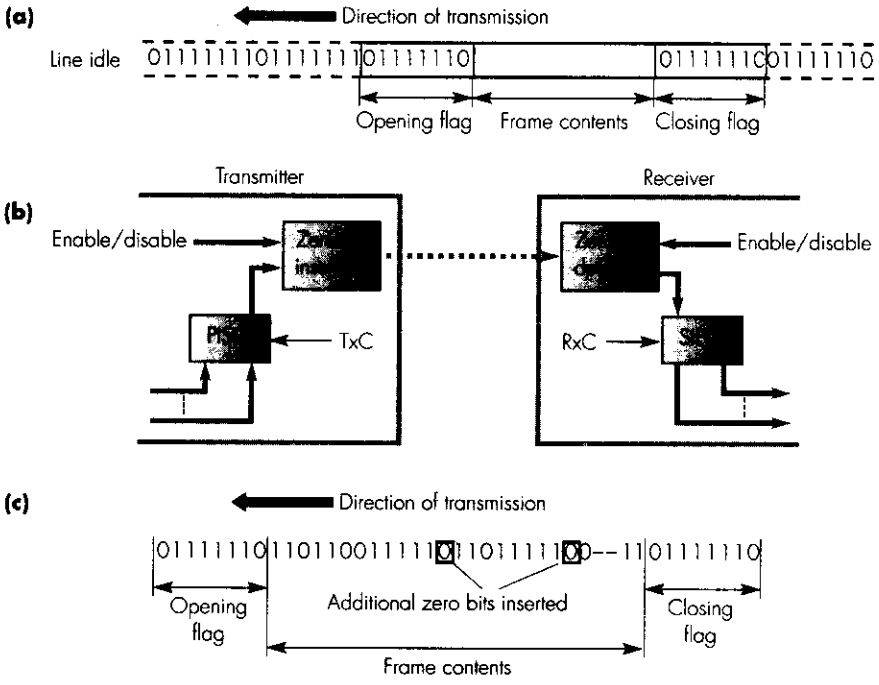


Figure 6.18 Bit-oriented synchronous transmission: (a) framing structure; (b) zero bit insertion circuit location; (c) example transmitted frame contents.

The start and end of a frame are both signaled by the same unique 8-bit pattern 01111110, known as the **flag byte** or **flag pattern**. We use the term "bit-oriented" because the received bitstream is searched by the receiver on a bit-by-bit basis for both the start-of-frame flag and, during reception of the frame contents, for the end-of-frame flag. Thus in principle the frame contents need not necessarily comprise multiples of 8 bits.

To enable the receiver to obtain and maintain bit synchronism, the transmitter sends a string of **idle bytes** (each comprising 01111111) preceding the start-of-frame flag. Recall that with NRZI encoding the 0 in the idle byte enables the DPLL at the receiver to obtain and maintain clock synchronization. On receipt of the opening flag, the received frame contents are read and interpreted on 8-bit (byte) boundaries until the closing flag is detected. The reception process is then terminated.

To achieve data transparency with this scheme, we must ensure that the flag pattern is not present in the frame contents. We do this by using a technique known as **zero bit insertion** or **bit stuffing**. The circuit that performs this function is located at the output of the PISO register, as shown in Figure 6.18(b). It is enabled by the transmitter only during transmission of the frame contents. When enabled, the circuit detects whenever it has transmitted a sequence of five contiguous binary 1 digits, then automatically inserts an additional binary 0 digit. In this way, the flag pattern 01111110 can never be present in the frame contents between the opening and closing flags.

A similar circuit at the receiver located prior to the input of the SIPO shift receiver performs the reverse function. Whenever a zero is detected after five contiguous 1 digits, the circuit automatically removes (deletes) it from the frame contents. Normally the frame also contains additional error detection digits preceding the closing flag which are subjected to the same bit stuffing operation as the frame contents. An example stuffed bit pattern is shown in Figure 6.18(c).

6.6 Error detection methods

As we indicated in Section 6.1, when transmitting a bitstream over a transmission line/channel a scheme is normally incorporated into the transmission control circuit of the NIC to enable the presence of bit/transmission errors in a received block to be detected. In general, this is done by the transmitter computing a set of (additional) bits based on the contents of the block of bits to be transmitted. These are known as error detection bits and are transmitted together with the original bits in the block. The receiver then uses the complete set of received bits to determine (to a high probability) whether the block contains any errors.

The two factors that determine the type of error detection scheme used are the **bit error rate** (**BER**) probability of the line and the type of errors, that is whether the errors occur as random single-bit errors or as groups of

contiguous strings of bit errors. The latter are referred to as **burst errors**. The BER is the probability P of a single bit being corrupted in a defined time interval. Thus a BER of 10^{-3} means that, on average, 1 bit in 10^{-3} will be corrupted during a defined time period.

If we are transmitting single characters using asynchronous transmission (say 8 bits per character plus 1 start and 1 stop bit), the probability of a character being corrupted is $1 - (1 - P)^{10}$ which, if we assume a BER of 10^{-3}, is approximately 10^{-2}. Alternatively, if we are transmitting blocks of say 125 bytes using synchronous transmission, then the probability of a block (frame) containing an error is approximately 1. This means that on average every block will contain an error. Clearly, therefore, this length of frame is too long for this type of line and must be reduced to obtain an acceptable throughput.

The type of errors present is important since, as we shall see, the different types of error detection scheme detect different types of error. Also, the number of bits used in some schemes determines the burst lengths that are detected. The three most widely used schemes are parity, block sum check, and cyclic redundancy check. We shall consider each separately.

6.6.1 Parity

The most common method used for detecting bit errors with asynchronous and character-oriented synchronous transmission is the **parity bit method**. With this scheme the transmitter adds an additional bit – the parity bit – to each transmitted character prior to transmission. The parity bit used is a function of the bits that make up the character being transmitted. On receipt of each character, the receiver then performs the same function on the received character and compares the result with the received parity bit. If they are equal, no error is assumed, but if they are different, then a transmission error is assumed.

To compute the parity bit for a character, the number of 1 bits in the code for the character are added together (modulo 2) and the parity bit is then chosen so that the total number of 1 bits (including the parity bit itself) is either even – **even parity** – or odd – **odd parity**. The principles of the scheme are shown in Figure 6.19.

The circuitry used to compute the parity bit for each character comprises a set of **exclusive-OR (XOR)** gates connected as shown in Figure 6.19(c). The XOR gate is also known as a **modulo-2 adder** since, as shown by the **truth table** in part (b) of the figure, the output of the exclusive-OR operation between two binary digits is the same as the addition of the two digits without a carry bit. The least significant pair of bits are first XORed together and the output of this gate is then XORed with the next (more significant) bit, and so on. The output of the final gate is the required parity bit which is loaded into the transmit PISO register prior to transmission of the character. Similarly, on receipt, the recomputed parity bit is compared with the received parity bit. If it is different, this indicates that a transmission error has been detected.

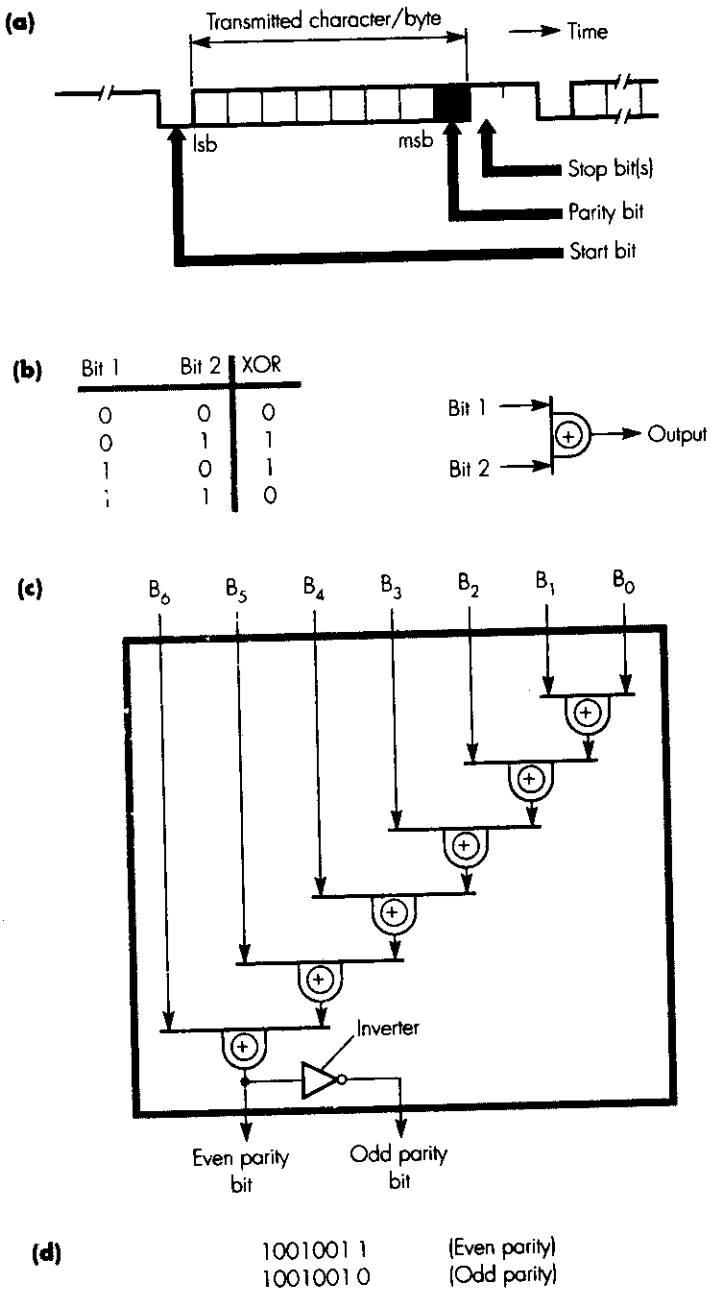


Figure 6.19 Parity bit method: (a) position in character; (b) XOR gate truth table and symbol; (c) parity bit generation circuit; (d) two examples.

The term used in coding theory to describe the combined message unit, comprising the useful data bits and the additional error detection bits, is codeword. The minimum number of bit positions in which two valid codewords differ is known as the **Hamming distance** of the code. As an example, consider a coding scheme that has seven data bits and a single parity bit per codeword. If we assume even parity is being used, consecutive codewords in this scheme will be:

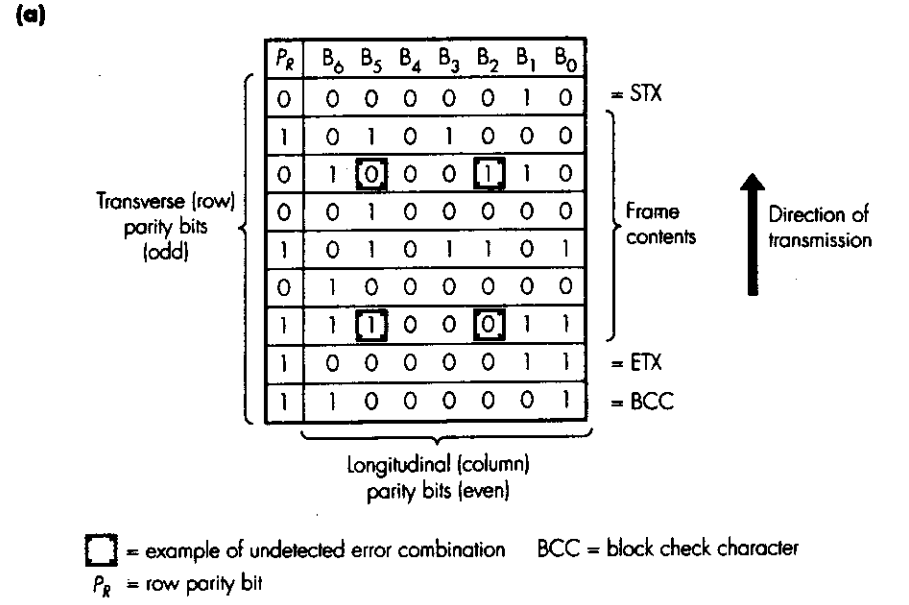
0000000 0
0000001 1
0000010 1
0000011 0

We can deduce from this list that such a scheme has a Hamming distance of 2 since each valid codeword differs in at least two bit positions. This means that the scheme does not detect 2-bit errors since the resulting (corrupted) bit pattern will be a different but valid codeword. It does, however, detect all single-bit errors (and all odd numbers of bit errors) since, if a single bit in a codeword is corrupted, an invalid codeword will result.

6.6.2 Block sum check

When blocks of characters (or bytes) are being transmitted, there is an increased probability that a character (and hence the block) will contain a bit error. The probability of a block containing an error is known as the **block error rate**. When blocks of characters (frames) are being transmitted, we can achieve an extension to the error detecting capabilities obtained from a single parity bit per character (byte) by using an additional set of parity bits computed from the complete block of characters (bytes) in the frame. With this method, each character (byte) in the frame is assigned a parity bit as before (**transverse** or **row parity**). In addition, an extra bit is computed for each bit position (**longitudinal** or **column parity**) in the complete frame. The resulting set of parity bits for each column is referred to as the **block (sum) check character** since each bit making up the character is the modulo-2 sum of all the bits in the corresponding column. The example in Figure 6.20(a) uses odd parity for the row parity bits and even parity for the column parity bits, and assumes that the frame contains printable characters only.

We can deduce from this example that although two bit errors in a character will escape the row parity check, they will be detected by the corresponding column parity check. This is true, of course, only if no two bit errors occur in the same column at the same time. Clearly, the probability of this occurring is much less than the probability of two bit errors in a single character occurring. The use of a block sum check significantly improves the error detection properties of the scheme.



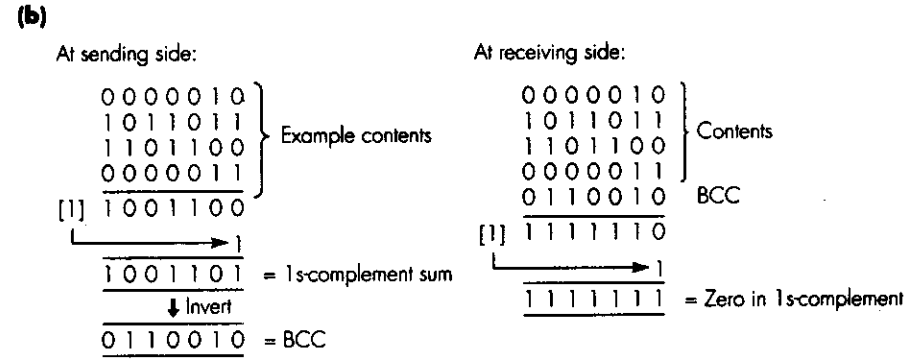


Figure 6.20 Block sum check method: (a) row and column parity bits; (b) 1s complement sum.

A variation of the scheme is to use the 1s-complement sum as the basis of the block sum check instead of the modulo-2 sum. The principle of the scheme is shown in Figure 6.20(b).

In this scheme, the characters (or bytes) in the block to be transmitted are treated as unsigned binary numbers. These are first added together using 1s-complement arithmetic. All the bits in the resulting sum are then inverted and this is used as the block check character (BCC). At the receiver, the 1s-complement sum of all the characters in the block – including the block check character – is computed and, if no errors are present, the result should

be zero. Remember that with 1s-complement arithmetic, end-around-carry is used, that is, any carry out from the most significant bit position is added to the existing binary sum. Also, zero in 1s-complement arithmetic is represented by either all binary 0s or all binary 1s.

As we shall see in later chapters, since the 1s-complement sum is readily computed, it is used as the error-detection method in a number of applications which require the error detection operation to be performed in software only.

6.6.3 Cyclic redundancy check

The previous two schemes are best suited to applications in which random single-bit errors are present. When bursts of errors are present, however, we must use a more rigorous method. An error burst begins and ends with an erroneous bit, although the bits in between may or may not be corrupted. Thus, an error burst is defined as the number of bits between two successive erroneous bits including the incorrect two bits. Furthermore, when determining the length of an error burst, the last erroneous bit in a burst and the first erroneous bit in the following burst must be separated by B or more correct bits, where B is the length of the error burst. An example of two different error burst lengths is shown in Figure 6.21. Notice that the first and third bit errors could not be used to define a single 11-bit error burst since an error occurs within the next 11 bits.

The most reliable detection scheme against error bursts is based on the use of **polynomial codes**. Polynomial codes are used with frame (or block) transmission schemes. A single set of check digits is generated (computed) for each frame transmitted, based on the contents of the frame, and is appended by the transmitter to the tail of the frame. The receiver then performs a similar computation on the complete frame plus check digits. If no

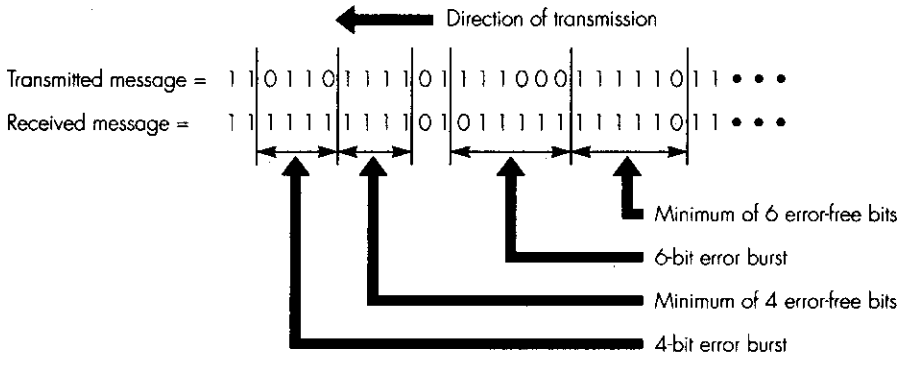


Figure 6.21 Error burst examples.

errors have been induced, a known result should always be obtained; if a different answer is found, this indicates an error.

The number of check digits per frame is selected to suit the type of transmission errors anticipated, although 16 and 32 bits are the most common. The computed check digits are referred to as the **frame check sequence** (**FCS**) or the **cyclic redundancy check** (**CRC**) digits.

The underlying mathematical theory of polynomial codes is beyond the scope of this book but, essentially, the method exploits the following property of binary numbers if modulo-2 arithmetic is used. Let:

$M(x)$ be a k-bit number (the message to be transmitted)

$G(x)$ be an $(n+1)$-bit number (the divisor or generator)

$R(x)$ be an n-bit number such that $k > n$ (the remainder)

Then if:

$$\frac{M(x) \times 2^n}{G(x)} = Q(x) + \frac{R(x)}{G(x)}, \text{ where } Q(x) \text{ is the quotient,}$$

$$\frac{M(x) \times 2^n + R(x)}{G(x)} = Q(x), \text{ assuming modulo -2 arithmetic.}$$

We can readily confirm this result by substituting the expression for $M(x) \times 2^n / G(x)$ into the second equation, giving:

$$\frac{M(x) \times 2^n + R(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)} + \frac{R(x)}{G(x)}$$

which is equal to $Q(x)$ since any number added to itself modulo 2 will result in zero, that is, the remainder is zero.

To exploit this, the complete frame contents, $M(x)$, together with an appended set of zeros equal in number to the number of FCS digits to be generated (which is equivalent to multiplying the message by 2^n, where n is the number of FCS digits) are divided modulo 2 by a second binary number, $G(x)$, the **generator polynomial** containing one more digit than the FCS. The division operation is equivalent to performing the exclusive-OR operation bit by bit in parallel as each bit in the frame is processed. The remainder $R(x)$ is then the FCS which is transmitted at the tail of the information digits. Similarly, on receipt, the received bitstream including the FCS digits is again divided by the same generator polynomial – that is, $(M(x) \times 2^n + R(x))/G(x)$ – and, if no errors are present, the remainder is all zeros. If an error is present, however, the remainder is nonzero.

Example 6.7

A series of 8-bit message blocks (frames) is to be transmitted across a data link using a CRC for error detection. A generator polynomial of 11001 is to be used. Use an example to illustrate the following:

(a) the FCS generation process,

(b) the FCS checking process.

Answer:

Generation of the FCS for the message 11100110 is shown in Figure 6.22(a). Firstly, four zeros are appended to the message, which is equivalent to multiplying the message by 2^4, since the FCS will be four bits. This is then divided (modulo 2) by the generator polynomial (binary number). The modulo-2 division operation is equivalent to performing the exclusive-OR operation bit by bit in parallel as each bit in the dividend is processed. Also, with modulo-2 arithmetic, we can perform a division into each partial remainder, providing the two numbers are of the same length, that is, the most significant bits are both 1s. We do not consider the relative magnitude of both numbers. The resulting 4-bit remainder (0110) is the FCS, which is then appended at the tail of the original message when it is transmitted. The quotient is not used.

At the receiver, the complete received bit sequence is divided by the same generator polynomial as used at the transmitter. Two examples are shown in Figure 6.22(b). In the first, no errors are assumed to be present, so that the remainder is zero – the quotient again not used. In the second, however, an error burst of four bits at the tail of the transmitted bit sequence is assumed. Consequently, the resulting remainder is nonzero, indicating that a transmission error has occurred.

The choice of generator polynomial is important since it determines the types of error that are detected. Assume the transmitted frame, $T(x)$ is:

110101100110

and the error pattern, $E(x)$ is:

00000000 1001

that is, a 1 in a bit position indicates an error. Hence, with modulo-2 arithmetic:

Received frame = $T(x) + E(x)$

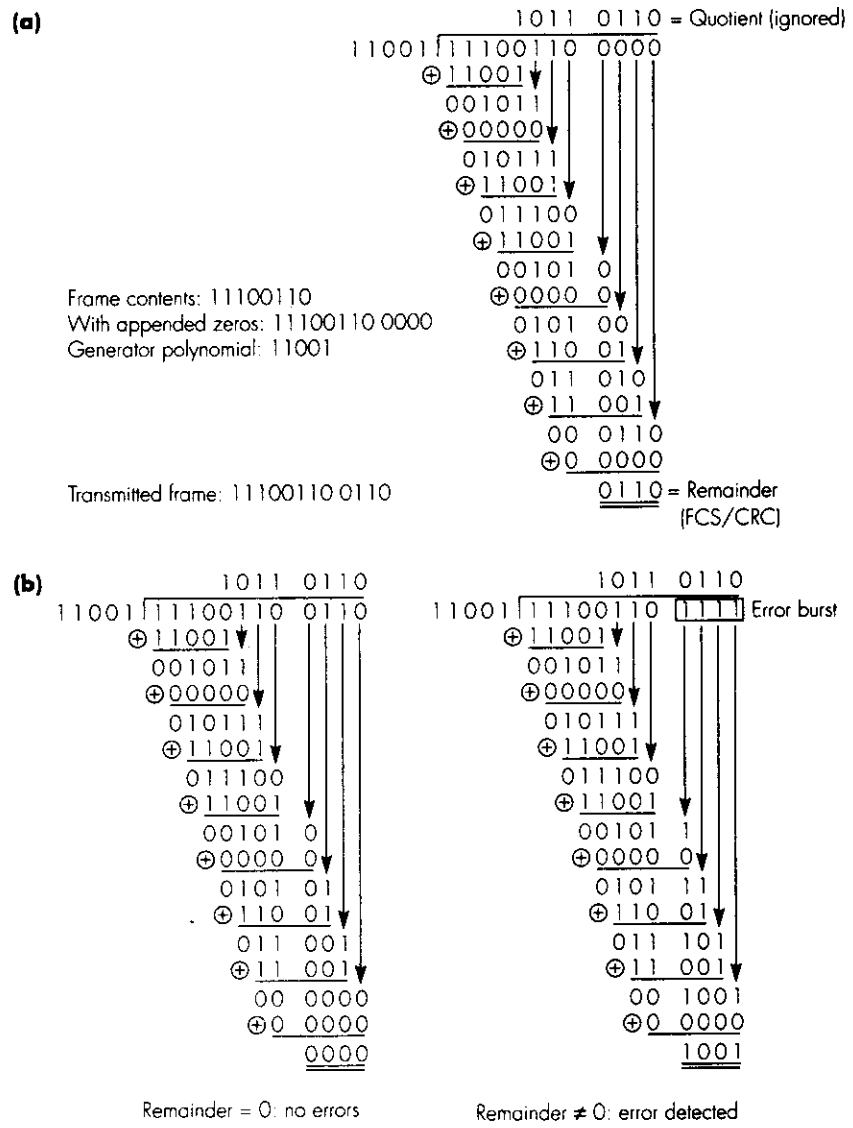


Figure 6.22 CRC error detection example: (a) FCS generation; (b) two error detection examples.

Now:

$$\frac{T(x) + E(x)}{G(x)} = \frac{T(x)}{G(x)} + \frac{E(x)}{G(x)}$$

but $T(x)/G(x)$ produces no remainder. Hence an error is detected only if $E(x)/G(x)$ produces a remainder.

For example, all $G(x)$ have at least three terms (1 bits) and $E(x)/G(x)$ will yield a remainder for all single-bit and all double-bit errors with modulo-2 arithmetic and hence be detected. Conversely, an error burst of the same length as $G(x)$ may be a multiple of $G(x)$ and hence yield no remainder and go undetected.

In summary, a generator polynomial of R bits will detect:

- all single-bit errors,
- all double-bit errors,
- all odd number of bit errors,
- all error bursts $0 < R$,
- most error bursts $\geq R$.

The standard way of representing a generator polynomial is to show those positions that are binary 1 as powers of X. Examples of CRCs used in practice are thus:

$$\begin{aligned}
&\text{CRC-6} && = X^{16} + X^{15} + X^2 + 1 \\
&\text{CRC-CCITT} && = X^{16} + X^{12} + X^5 + 1 \\
&\text{CRC-32} && = X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 \\
& && \quad + X^5 + X^4 + X^2 + X + 1
\end{aligned}$$

Hence CRC-16 is equivalent in binary form to:

1 1000 0000 0000 0101

With such a generator polynomial, 16 zeros would be appended to the frame contents before generation of the FCS. The latter would then be the 16-bit remainder. CRC-16 will detect all error bursts of less than 16 bits and most error bursts greater than or equal to 16 bits. CRC-16 and CRC-CCITT are both used extensively with networks such as an ISDN, while CRC-32 is used in most LANs. Also, although the requirement to perform multiple (modulo-2) divisions may appear to be relatively complicated, as we show in Appendix B, it can be done readily in hardware (and software) and, in practice, this is integrated into the transmission control circuit.

6.7 Protocol basics

In Section 6.6 we described the different methods that are used to detect the presence of transmission errors. Also, as we explained in Section 6.1, in most cases any blocks/frames that are received containing errors are simply discarded by the link layer. It is then left to the transport layer in each of the two communicating end systems to detect any missing blocks and, if necessary, to

request that another copy of these is retransmitted. However, in a small number of cases the error recovery procedure is performed in the link layer; for example, in the link layer associated with a PSTN. In this section we describe the basic principles associated with the error control procedure that is used and we then use this to explain how a protocol is specified.

6.7.1 Error control

The transmission control circuit associated with most network interfaces performs both the transmission control and error detection functions we explained in Sections 6.5 and 6.6. The link layer protocol then builds on these basic functions to provide the required link layer service. Typically, the receiving link protocol checks the received frame for possible transmission errors and then returns a short control message/frame either to acknowledge its correct receipt or to request that another copy of the frame is sent. This type of error control is known as **automatic repeat request** (**ARQ**) and the aim of all ARQ schemes is to provide a *reliable* link layer service. In this context, "reliable" means that the two peer link layer protocols will communicate with each other in order to deliver a sequence of blocks that is submitted to the sending link layer protocol. Thus

- the blocks will be delivered by the receiving link layer protocol in the same sequence as they were submitted and with no duplicate copies of any of the blocks;
- to a high probability, each block will be free of any bit errors.

The most basic type of ARQ scheme is **idle RQ** and so we shall start by explaining this. We then identify the limitations of idle RQ and explain how these are overcome in the **continuous RQ** scheme. In practice, there are two types of continuous RQ: selective repeat and go-back-N, both of which we describe. We then return to the idle RQ scheme to explain how the error control part of a protocol is specified.

6.7.2 Idle RQ

In order to discriminate between the sender (source) and receiver (destination) of data frames – more generally referred to as information or **I-frames** – the terms **primary** (**P**) and **secondary** (**S**) are used respectively. The idle RQ protocol operates in a half-duplex mode since the primary, after sending an I-frame, waits until it receives a response from the secondary as to whether the frame was correctly received or not. The primary then either sends the next frame, if the previous frame was correctly received, or retransmits a copy of the previous frame if it was not.

The secondary informs the primary of a correctly received frame by returning a (positive) **acknowledgment** or **ACK-frame**. Similarly, if the

secondary receives an I-frame containing errors, then it returns a **negative acknowledgment** or **NAK-frame**. Three example frame sequences illustrating various aspects of this basic procedure are shown in Figure 6.23. The following points should be noted when interpreting the sequences:

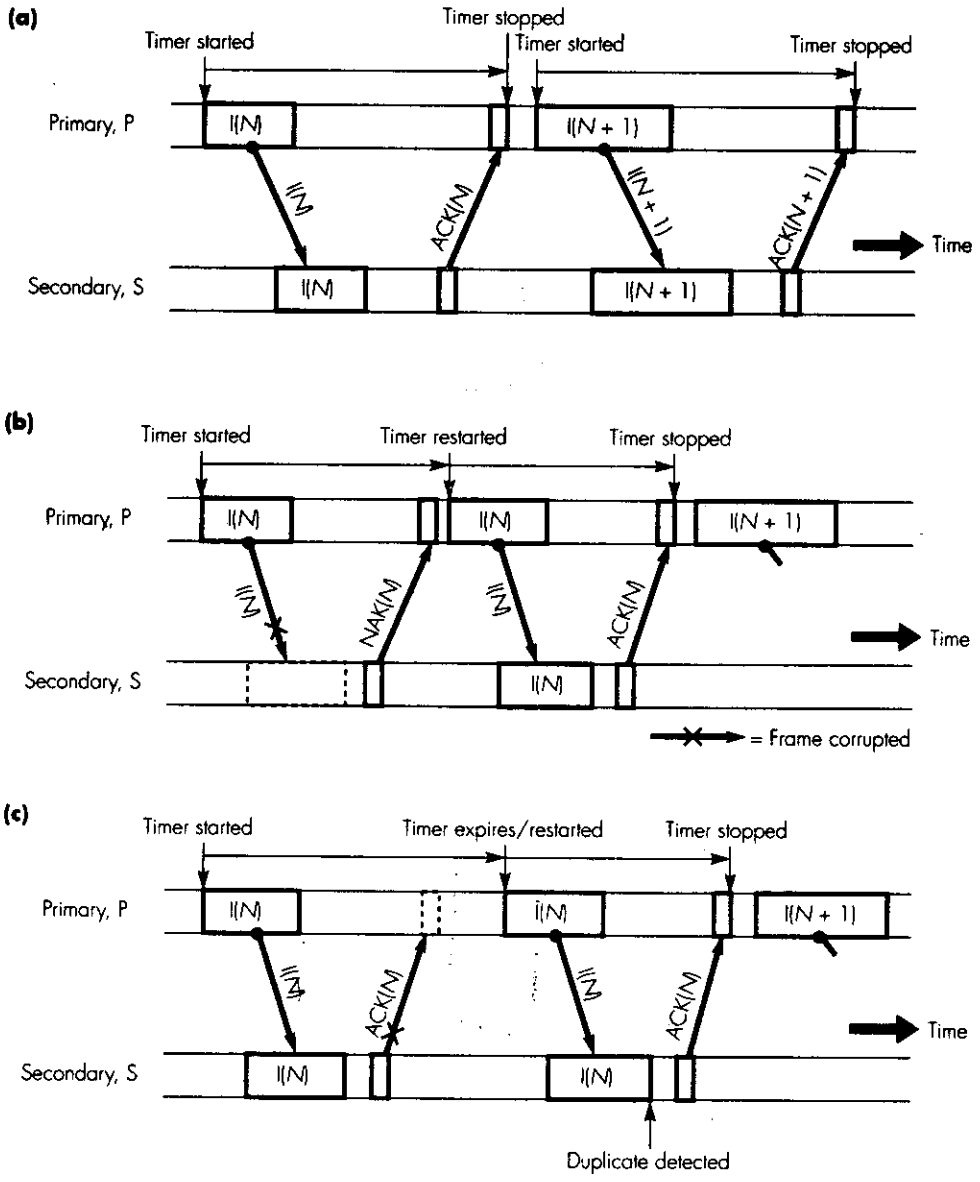


Figure 6.23 Idle RQ error control scheme: (a) error free; (b) corrupted I-frame; (c) corrupted ACK-frame.

- P can have only one I-frame outstanding (awaiting an ACKNAK-frame) at a time.
- When P initiates the transmission of an I-frame it starts a timer.
- On receipt of an error-free I-frame, S returns an ACK-frame to P and, on receipt of this, P stops the timer for this frame and proceeds to send the next frame – part (a).
- On receipt of an I-frame containing transmission errors, S discards the frame and returns a NAK-frame to P which then sends another copy of the frame and restarts the timer – part (b).
- If P does not receive an ACK- (or NAK-) frame within the timeout interval, P retransmits the I-frame currently waiting acknowledgment – part (c). However, since in this example it is an ACK-frame that is corrupted, S detects that the next frame it receives is a duplicate copy of the previous error-free frame it received rather than a new frame. Hence S discards the duplicate and, to enable P to resynchronize, returns a second ACK-frame for it. This procedure repeats until either an error-free copy of the frame is received or a defined number of retries is reached in which case the network layer in P would be informed of this.

As we show in the figure, in order for S to determine when a duplicate is received, each frame transmitted by P contains a unique identifier known as the **send sequence number N(S)** (N, $N + 1$, and so on) within it. Also, S retains a record of the sequence number contained within the last I-frame it received without errors and, if the two are the same, this indicates a duplicate. The sequence number in each ACK- and NAK-frame is known as the **receive sequence number N(R)** and, since P must wait for an ACK- or NAK-frame after sending each I-frame, the scheme is known also as **send-and-wait** or sometimes **stop-and-wait**.

Link utilization

Before considering the error procedures associated with the two types of continuous RQ scheme, we shall first quantify the efficiency of utilization of the available link capacity with the idle RQ scheme. The efficiency of utilization U is a ratio of two times, each measured from the point in time the transmitter starts to send a frame. It is defined as:

$$U = \frac{T_{ix}}{T_t}$$

where T_{ix} is the time for the transmitter to transmit a frame and T_t equals T_{ix} plus any time the transmitter spends waiting for an acknowledgment.

To quantify the link utilization with idle RQ, a frame sequence diagram with the various component times identified is given in Figure 6.24. In practice, in most cases for which the idle RQ protocol is adequate, the time to

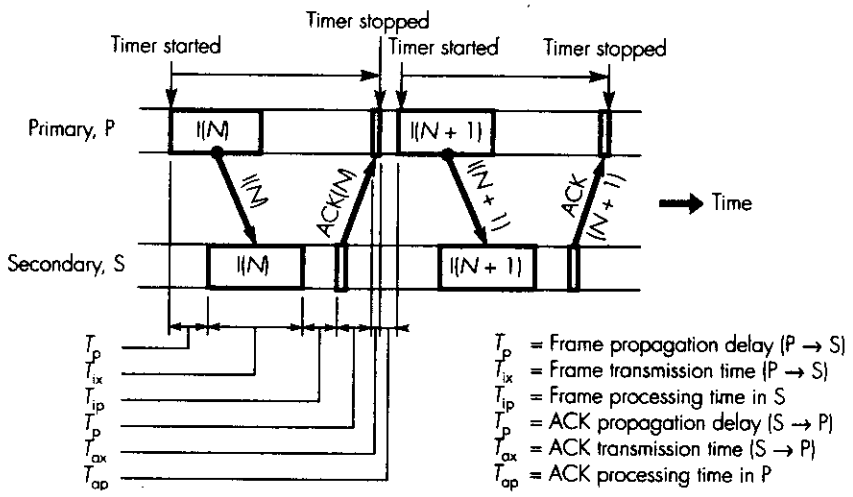


Figure 6.24 Idle RQ link utilization.

process an I-frame T_{ip} and its associated ACK-frame T_{ap} are both short compared with their transmission times T_{ix} and T_{ax}. Also, since an ACK-frame is much shorter than an I-frame, T_{ax} is negligible compared with T_{ix}. Hence the minimum total time before the next frame can be transmitted is often approximated to $T_{ix} + 2T_p$ where T_p is the signal propagation delay of the link. An approximate expression for U is thus:

$$U = \frac{T_{ix}}{T_{ix} + 2T_p}$$

or:

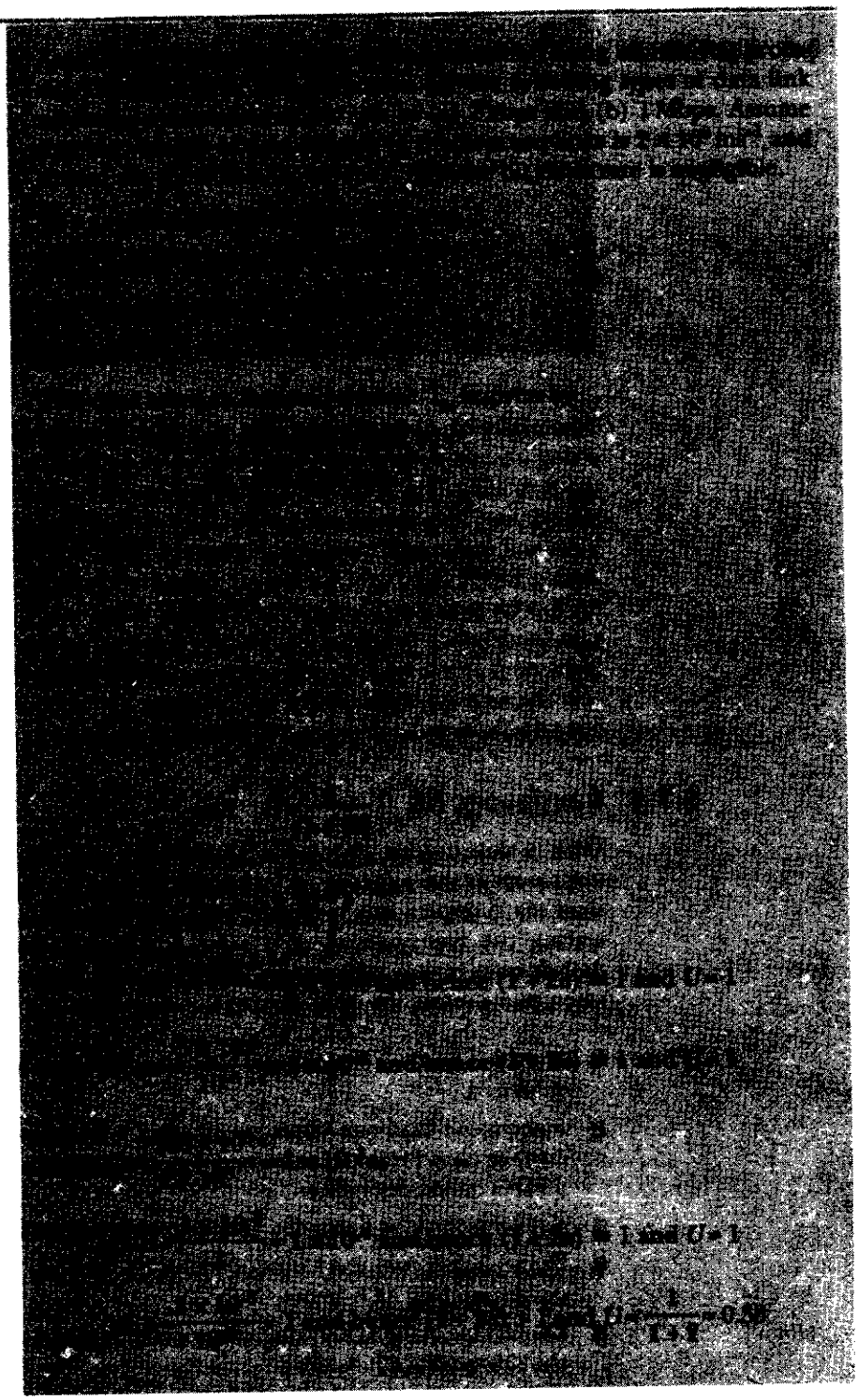
$$U = \frac{1}{1 + 2T_p/T_{ix}}$$

As we described earlier in Section 6.2.8, the ratio T_p/T_{ix} is often given the symbol a and hence:

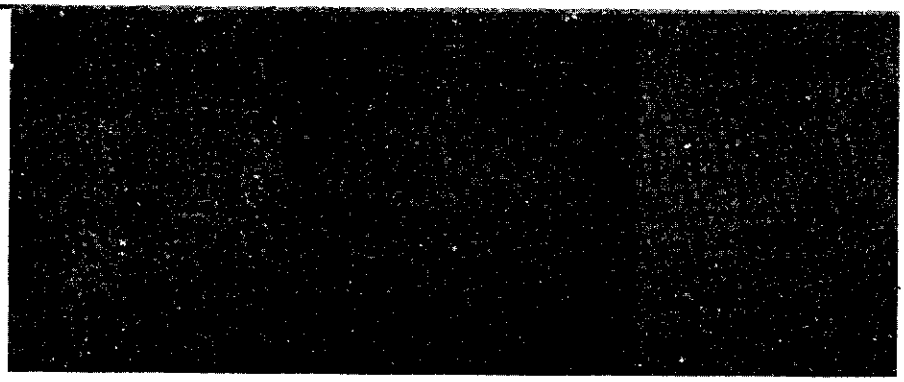
$$U = \frac{1}{1 + 2a}$$

In Example 6.1 we saw that a can range from a small fraction for low bit rate links of modest length to a large integer value for long links and high bit rates. For these two extremes, U varies between a small fraction and near unity (100%).

Example 6.8



6.8 Continued



The results are summarized in Figure 6.25 from which we can make some interesting observations. Firstly, for relatively short links for which a is less than 1, the link utilization is (to a good approximation) 100% and is independent of the bit rate. This means that an idle RQ protocol is perfectly adequate for short links and modest bit rates. Examples are networks based on modems and an analog PSTN. Secondly, for longer terrestrial links, the link utilization is high for low bit rates (and hence low values of a) but falls off significantly as the bit rate (and hence a) increases. Thirdly, the link utilization is poor for satellite links, even at low bit rates. We can conclude that an idle RQ protocol is unsuitable for such applications and also for those that involve high bit rate terrestrial links which include all of the networks that are used for multimedia.

6.7.3 Continuous RQ

With a continuous RQ error control scheme, link utilization is much improved at the expense of increased buffer storage requirements. As we shall see, a duplex link is required for its implementation. An example illustrating the transmission of a sequence of I-frames and their returned ACK-frames is shown in Figure 6.26. You should note the following points when interpreting the operation of the scheme:

- P sends I-frames continuously without waiting for an ACK-frame to be returned.
- Since more than one I-frame is awaiting acknowledgment, P retains a copy of each I-frame transmitted in a **retransmission list** that operates on a FIFO queue discipline.
- S returns an ACK-frame for each correctly received I-frame.
- Each I-frame contains a unique identifier which is returned in the corresponding ACK-frame.
- On receipt of an ACK-frame, the corresponding I-frame is removed from the retransmission list by P.

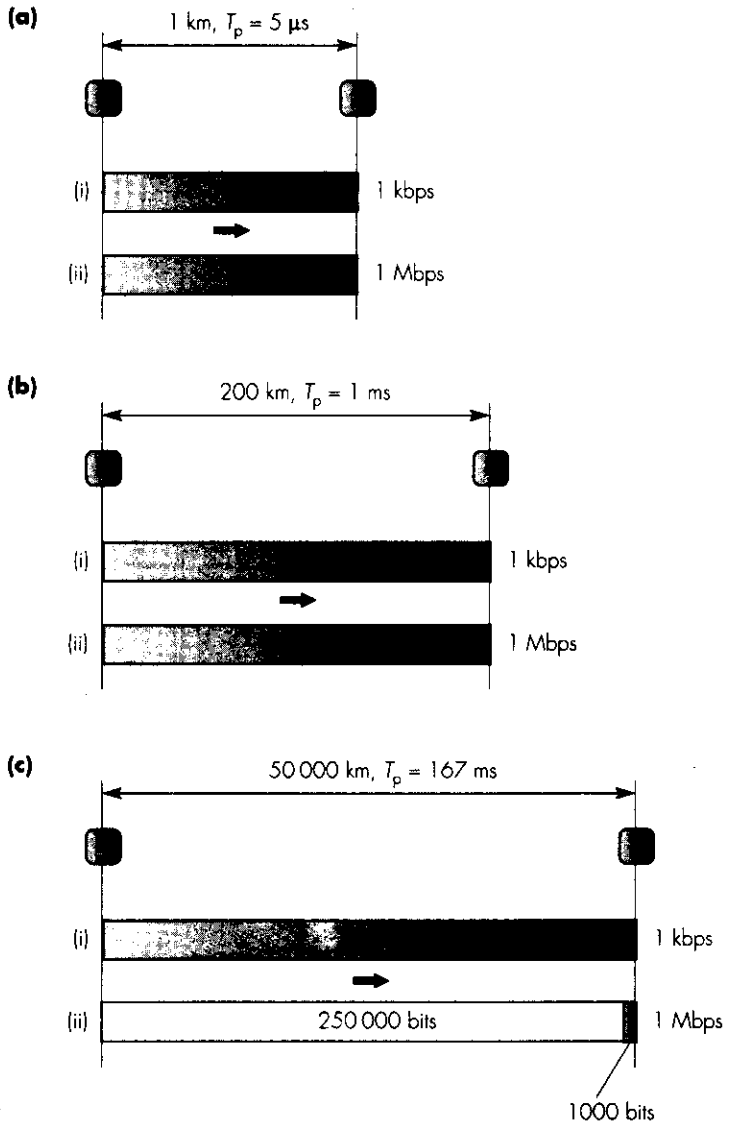
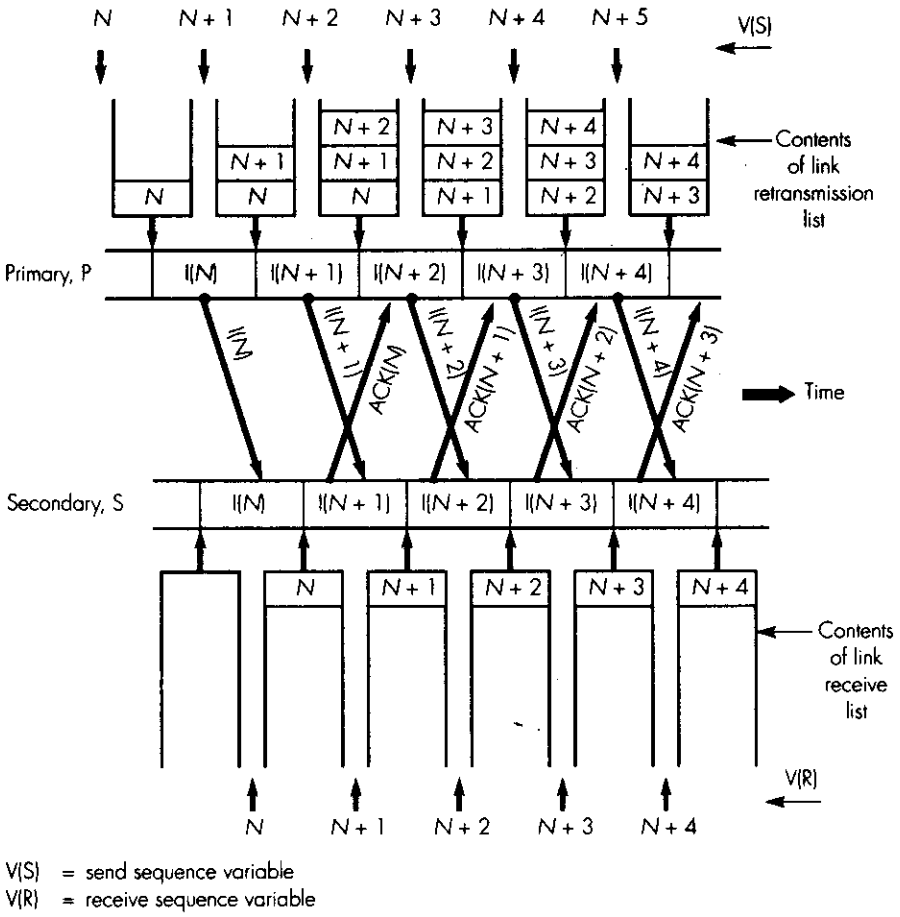


Figure 6.25 Effect of propagation delay as a function of data transmission rate; parts correspond to Example 6.8.

- Frames received free of errors are placed in the **link receive** list to await processing.
- On receipt of the next in-sequence I-frame expected, S delivers the information content within the frame to the upper network layer immediately it has processed the frame.



V(S) = send sequence variable
V(R) = receive sequence variable

Figure 6.26 Continuous RQ frame sequence without transmission errors.

To implement the scheme, P must retain a send sequence variable V(S), which indicates the send sequence number N(S) to be allocated to the next I-frame to be transmitted. Also, S must maintain a receive sequence variable V(R), which indicates the next in-sequence I-frame it is waiting for.

The frame sequence shown in Figure 6.26 assumes that no transmission errors occur. When an error does occur, one of two retransmission strategies may be followed:

- S detects and requests the retransmission of just those frames in the sequence that are corrupted – selective repeat.
- S detects the receipt of an out-of-sequence I-frame and requests P to retransmit all outstanding unacknowledged I-frames from the last correctly received, and hence acknowledged, I-frame – go-back-N.

Note that with both continuous RQ schemes, corrupted frames are discarded and retransmission requests are triggered only after the next error-free frame is received. Hence, as with the idle RQ scheme, a timeout is applied to each frame transmitted to overcome the possibility of a corrupted frame being the last in a sequence of new frames.

Selective repeat

Two example frame sequence diagrams that illustrate the operation of the selective repeat retransmission control scheme are shown in Figure 6.27. The sequence shown in part(a) shows the effect of a corrupted I-frame being received by S. The following points should be noted when interpreting the sequence:

- An ACK-frame acknowledges all frames in the retransmission list up to and including the I-frame with the sequence number the ACK contains.
- Assume I-frame $N+1$ is corrupted.
- S returns an ACK-frame for I-frame N.
- When S receives I-frame $N+2$ it detects I-frame $N+1$ is missing from V(R) and hence returns a NAK-frame containing the identifier of the missing I-frame $N+1$.
- On receipt of NAK $N+1$, P interprets this as S is still awaiting I-frame $N+1$ and hence retransmits it.
- When P retransmits I-frame $N+1$ it enters the **retransmission state**.
- When P is in the retransmission state, it suspends sending any new frames and sets a timeout for the receipt of ACK $N+1$.
- If the timeout expires, another copy of I-frame $(N+1)$ is sent.
- On receipt of ACK $N+1$ P leaves the retransmission state and resumes sending new frames.
- When S returns a NAK-frame it enters the retransmission state.
- When S is in the retransmission state, the return of ACK-frames is suspended.
- On receipt of I-frame $N+1$, S leaves the retransmission state and resumes returning ACK-frames.
- ACK $N+1$ acknowledges all frames up to and including frame $N+4$.
- A timer is used with each NAK-frame to ensure that if it is corrupted (and hence NAK $N+1$ is not received), it is retransmitted.

The sequence shown in Figure 6.27(b) shows the effect of a corrupted ACK-frame. The following points should be noted:

- Assume ACK N is corrupted.
- On receipt of ACK-frame $N+1$, P detects that I-frame N is still awaiting acknowledgment and hence retransmits it.

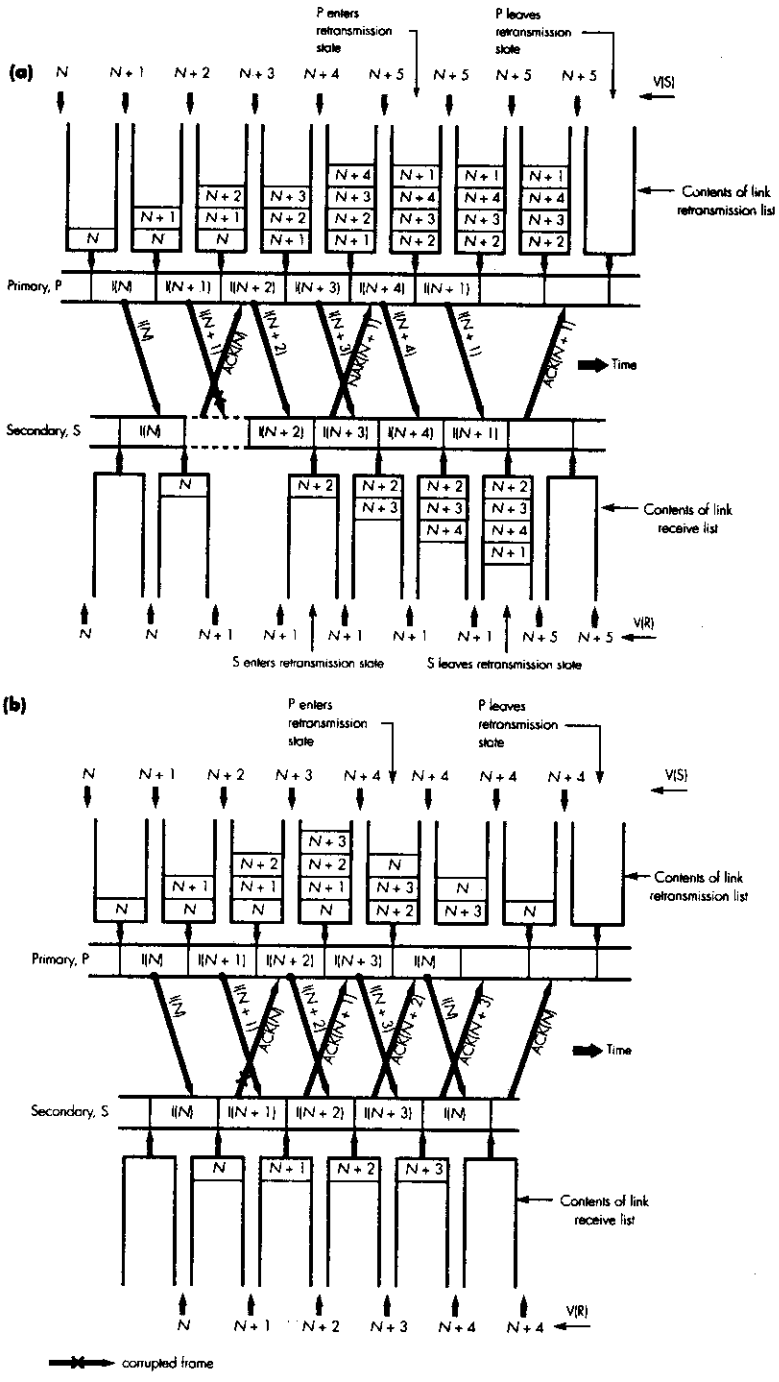


Figure 6.27 Selective repeat: (a) effect of corrupted I-frame; (b) effect of corrupted ACK-frame.

- On receipt of the retransmitted I-frame N, S determines from its received sequence variable that this has already been received correctly and is therefore a duplicate.
- S discards the frame but returns an ACK-frame to ensure P removes the frame from the retransmission list.

Go-back-N

Two example frame sequence diagrams that illustrate the operation of the go-back-N retransmission control scheme are shown in Figure 6.28. The sequence shown in part(a) shows the effect of a corrupted I-frame being received by S. The following points should be noted:

- Assume I-frame $N+1$ is corrupted.
- S receives I-frame $N+2$ out of sequence.
- On receipt of I-frame $N+2$, S returns NAK $N+1$ informing P to go back and start to retransmit from I-frame $N+1$.
- On receipt of NAK $N+1$, P enters the retransmission state. When in this state, it suspends sending new frames and commences to retransmit the frames waiting acknowledgment in the retransmission list.
- S discards frames until it receives I-frame $N+1$.
- On receipt of I-frame $N+1$, S resumes accepting frames and returning acknowledgments.
- A timeout is applied to NAK-frames by S and a second NAK is returned if the correct in-sequence I-frame is not received in the timeout interval.

The frame sequence shown in Figure 6.28(b) shows the effect of a corrupted ACK-frame. Note that:

- S receives each transmitted I-frame correctly.
- Assume ACK-frames N and $N+1$ are both corrupted.
- On receipt of ACK-frame $N+2$, P detects that there are two outstanding I-frames in the retransmission list (N and $N+1$).
- Since it is an ACK-frame rather than a NAK-frame, P assumes that the two ACK-frames for I-frames N and $N+1$ have both been corrupted and hence accepts ACK-frame $N+2$ as an acknowledgment for the outstanding frames.

In order to discriminate between the NAK-frames used in the two schemes, in the selective repeat scheme a NAK is known as a **selective reject** and in the go-back-N scheme a **reject**.

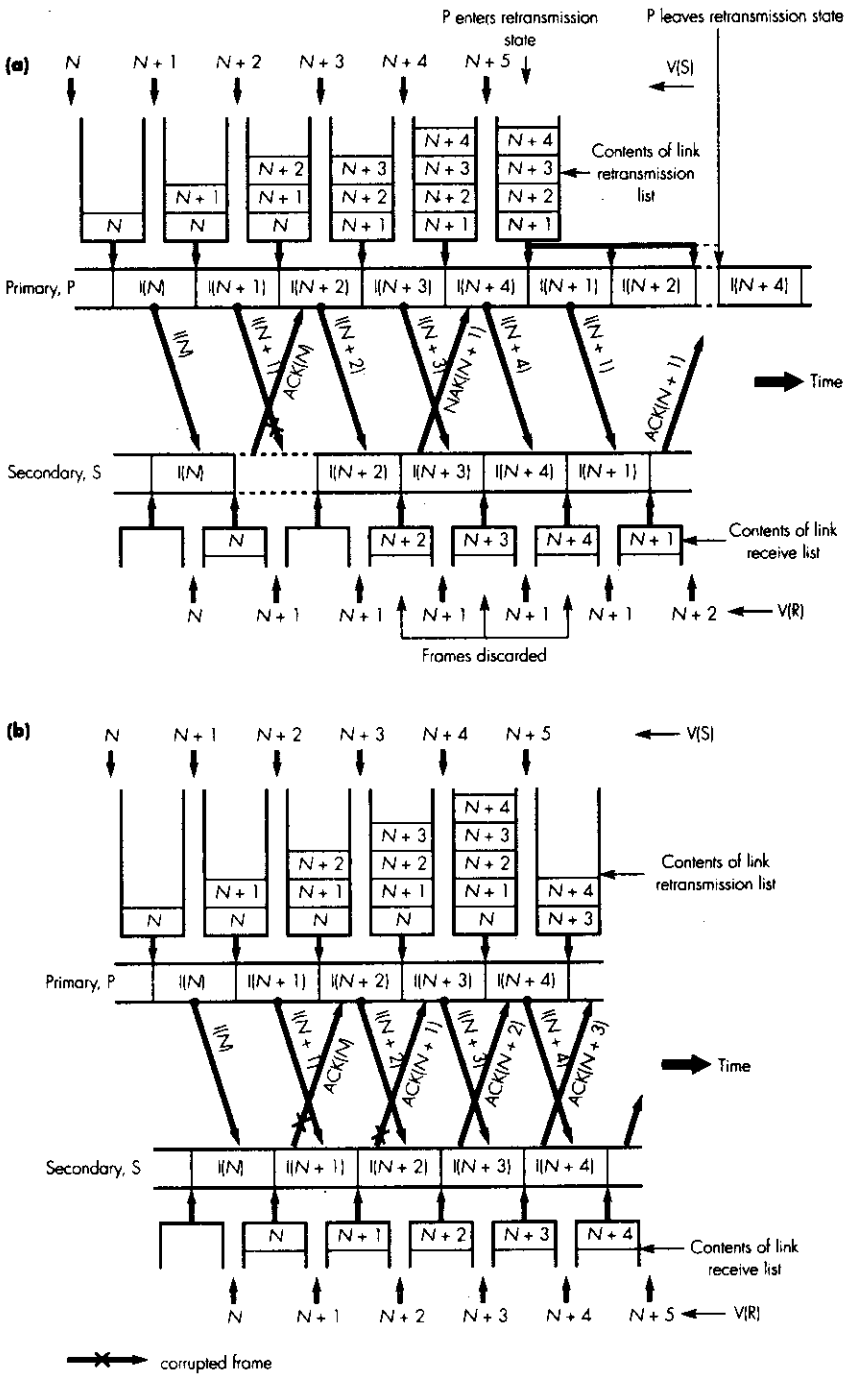


Figure 6.28 Go-back-N retransmission strategy: (a) corrupted I-frame; (b) corrupted ACK-frame.

6.7.4 Flow control

Error control is only one component of a data link protocol. Another important and related component is flow control. As the name implies, it is concerned with controlling the rate of transmission of frames on a link so that the receiver always has sufficient buffer storage resources to accept them prior to processing.

To control the flow of frames across a link, a mechanism known as a **sliding window** is used. The approach is similar to the idle RQ control scheme in that it essentially sets a limit on the number of I-frames that P may send before receiving an acknowledgment. P monitors the number of outstanding (unacknowledged) I-frames currently held in the retransmission list. If the destination side of the link is unable to pass on the frames sent to it, S stops returning acknowledgment frames, the retransmission list at P builds up and this in turn can be interpreted as a signal for P to stop transmitting further frames until acknowledgments start to flow again.

To implement this scheme, a maximum limit is set on the number of I-frames that can be awaiting acknowledgment and hence are outstanding in the retransmission list. This limit is the **send window**, K for the link. If this is set to 1, the retransmission control scheme reverts to idle RQ with a consequent drop in transmission efficiency. The limit is normally selected so that, providing the destination is able to pass on or absorb all frames it receives, the send window does not impair the flow of I-frames across the link. Factors such as the maximum frame size, available buffer storage, link propagation delay, and transmission bit rate must all be considered when selecting the send window.

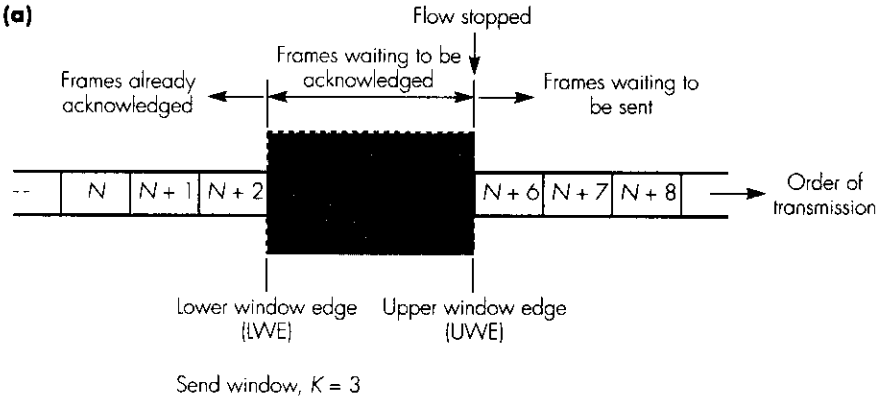
The operation of the scheme is shown in Figure 6.29. As each I-frame is transmitted, the **upper window edge** (**UWE**) is incremented by unity. Similarly, as each I-frame is acknowledged, the **lower window edge** (**LWE**) is incremented by unity. The acceptance of any new message blocks, and hence the flow of I-frames, is stopped if the difference between UWE and LWE becomes equal to the send window K. Assuming error-free transmission, K is a fixed window that moves (slides) over the complete set of frames being transmitted. The technique is thus known as "sliding window".

The maximum number of frame buffers required at S is known as the **receive window**. We can deduce from the earlier frame sequence diagrams that with the idle RQ and go-back-N schemes only one buffer is required. With selective repeat, however, K frames are required to ensure frames are delivered in the correct sequence.

6.7.5 Sequence numbers

Until now, we have assumed that the sequence number inserted into each frame by P is simply the previous sequence number plus one and that the range of numbers available is infinite. Defining a maximum limit on the number of I-frames being transferred across a link not only limits the size of the link retransmission and receive lists, but also makes it possible to limit the

(a)



(b)

Protocol	Send window	Receive window
Idle RQ	1	1
Selective repeat	K	K
Go-back-N	K	1

Figure 6.29 Flow control principle: (a) sliding window example; (b) send and receive window limits.

range of sequence numbers required to identify each transmitted frame uniquely. The number of identifiers is a function of both the retransmission control scheme and the size of the send and receive windows.

For example, with an idle RQ control scheme, the send and receive windows are both 1 and hence only two identifiers are required to allow S to determine whether a particular I-frame received is a new frame or a duplicate. Typically, the two identifiers are 0 and 1; the send sequence variable is incremented modulo 2 by P.

With a go-back-N control scheme and a send window of K, the number of identifiers must be at least $K+1$. We can deduce this by considering the case when P sends a full window of K frames but all the ACK-frames relating to them are corrupted. If only K identifiers were used, S would not be able to determine whether the next frame received is a new frame – as it expects – or a duplicate of a previous frame.

With a selective repeat scheme and a send and receive window of K, the number of identifiers must not be less than $2K$. Again, we can deduce this by considering the case when P sends a full window of K frames and all subsequent acknowledgments are corrupted. S must be able to determine whether any of the next K frames are new frames. The only way of ensuring that S can deduce this is to assign a completely new set of K identifiers to the next window of I-frames transmitted, which requires at least $2K$ identifiers. The limits for each scheme are summarized in Figure 6.30(a).

(a)

Protocol	Maximum number of frame identifiers
Idle RQ	2
Selective repeat	$2K + 1$
Go-back-N	$K + 1$

(b)

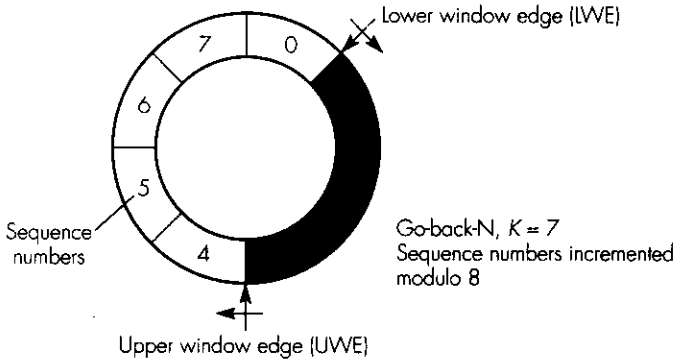


Figure 6.30 Sequence numbers: (a) maximum number for each protocol; (b) example assuming eight sequence numbers.

In practice, since the identifier of a frame is in binary form, a set number of binary digits must be reserved for its use. For example, with a send window of, say, 7 and a go-back-N control scheme, three binary digits are required for the send and receive sequence numbers yielding eight possible identifiers: 0 through 7. The send and receive sequence variables are then incremented modulo 8 by P and S respectively. This is illustrated in Figure 6.30(b).

6.7.6 Layered architecture

The frame sequence diagrams that we showed earlier provide a qualitative description of the error control (and flow control) components of a link layer protocol that is based on an idle RQ scheme – Figure 6.23 – and a continuous RQ scheme – Figures 6.26–28. In practice, however, it is not possible to describe fully the operation of all aspects of a protocol using just this method. Normally, therefore, the operation of a protocol is specified in a more formal way and, in order to gain an insight into how this is done, we shall specify the error control component of the idle RQ error control scheme.

Before we do this, we revisit the subject of layering which we first introduced in Section 5.2.2. As we showed in Figure 5.3(a), this involves decoupling each protocol layer one from another and defining a formal interface between them. Assuming an idle RQ scheme, a suitable layered architecture for the link layer is as shown in Figure 6.31. As we have described, the service provided to the network layer in the source is to transfer in a reliable way a series of blocks of information to the network layer in

the destination. Also, depending on the BER probability of the line, a maximum block size will be specified that ensures a good percentage of I-frames transmitted will be free of errors.

As we show in Figure 6.31, in order to decouple the network and link layers in each system, we introduce a queue between them. Each queue is simply a data structure that implements a first-in, first-out (FIFO) queuing discipline. Elements are added to the tail of the queue and are removed from the head.

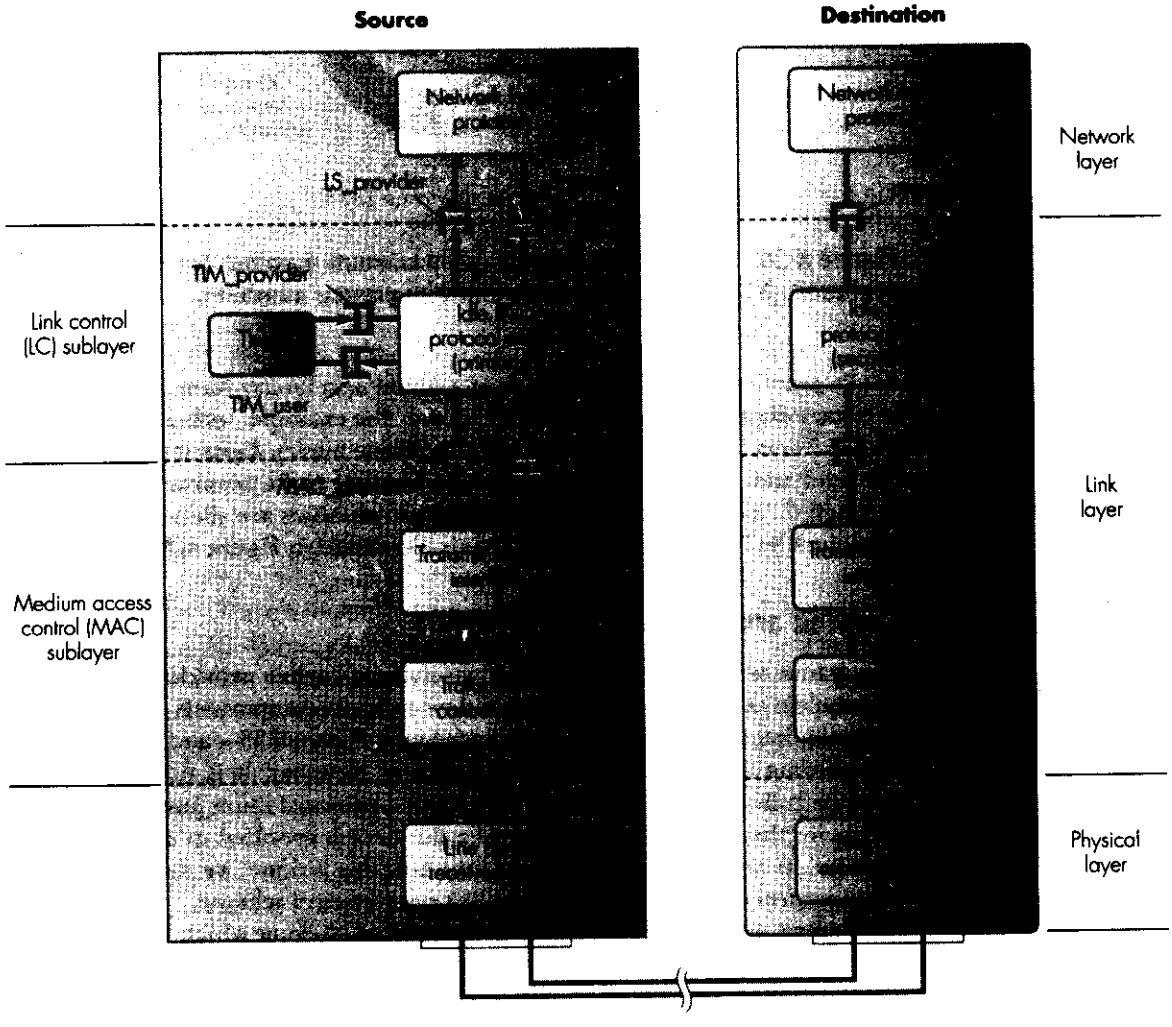


Figure 6.31 Example layered architecture showing the layer and sublayer interfaces associated with the idle RQ protocol.

Normally, the user service primitive(s) associated with a layer is(are) passed between layers using a data structure known as an **event control block** (**ECB**). This has the primitive type in the first field and an array containing the user data in a second field. For example, whenever the network layer protocol wishes to send a message block – the contents of a data structure – it first obtains a free ECB, sets the primitive type field to L_DATA.request, writes the address pointer of the data structure in the user data field, and inserts the ECB at the tail of the link layer (LS_user) input queue ready for reading by the idle RQ primary.

When the idle RQ protocol software is next run, it detects the presence of an entry (ECB) in the LS_user queue, reads the entry from the head of the queue, and proceeds to create an I-frame containing the send sequence number and the contents of the user data field. It then initiates the transmission of the frame to the secondary protocol using the services provided by the transmission control circuit. Normally, as we show in the figure, associated with the latter is a piece of low-level software which, in practice, is part of the **basic input-output software** (**BIOS**) of the computer. Assuming bit-oriented transmission, as the frame contents are being output, the transmission control circuit generates the CRC for the frame and adds the start and end flags. As we can deduce from this, therefore, the link layer comprises two sublayers: the **link control** (**LC**) – which is concerned with the implementation of the error and flow control procedures that are being used and is independent of the type of transmission control mode – and the **medium access control** (**MAC**) **sublayer** – which is concerned with the transmission of preformatted blocks using a particular transmission control mode which may vary for different networks. The physical layer comprises suitable bit/clock encoding circuits, line driver and receiver circuits, and the plug and socket pin definitions.

At the destination, assuming the received frame is error free, the MAC sublayer passes the frame contents to the LC sublayer using a MAC_DATA.indication primitive in an ECB and the MAC_provider queue. The LC sublayer then uses the send sequence number at the head of the frame to confirm it is not a duplicate and passes the frame contents – the message block – up to the network layer in an ECB using the LS_provider output queue with the primitive type set to L_DATA.indication. It then creates and returns an ACK-frame to P using a MAC_DATA.request primitive in an ECB and the MAC_user queue.

When the destination network layer protocol is next run, it detects and reads the ECB from the LS_provider queue and proceeds to process the contents of the message block it contains according to the defined network layer protocol. At the sending side, assuming the ACK-frame is received free of errors, the MAC sublayer passes the frame to the LC sublayer primary which frees the memory buffer containing the acknowledged I-frame and checks the LS_user input queue for another waiting ECB. If there is one, the procedure is repeated until all queued blocks have been transferred. Note that the LS_user queue and the retransmission list are quite separate. The first is used

to hold new message blocks waiting to be transmitted and the second to hold frames – containing blocks – that have already been sent and are waiting to be acknowledged.

We can conclude that the adoption of a layered architecture means that each layer performs its own well-defined function in relation to the overall communications task. Each layer provides a defined service to the layer immediately above it. The service primitives associated with the service are each implemented by the layer protocol communicating with a peer layer protocol in the remote system. Associated with the protocol are protocol data units (PDUs) – for example, I-frame, ACK-frame, and so on in the case of the link layer – and these are physically transferred using the services provided by the layer immediately below it.

6.7.7 Protocol specification

Irrespective of the specification method that is used, we model a protocol as a **finite state machine** or **automaton**. This means that the protocol – or, more accurately, the **protocol entity** – can be in just one of a finite number of defined **states** at any instant. For example, it might be idle waiting for a message to send, or waiting to receive an acknowledgment. Transitions between states take place as a result of an incoming event, for example, a message becomes ready to send, or an ACK-frame is received. As a result of an incoming event, an associated **outgoing event** is normally generated, for example, on receipt of a message, format and send the created I-frame on the link, or on receipt of a NAK-frame, retransmit the waiting I-frame.

Some incoming events may lead to a number of possible outgoing events. The particular outgoing event selected is determined by the computed state of one or more **predicates** (boolean variables). As an example, predicate P1 may be true if the N(R) in a received ACK-frame is the same as the N(S) in the I-frame waiting to be acknowledged. Hence, if P1 is true, then free the memory buffer in which the I-frame is being held; if it is false, initiate retransmission of the frame.

An incoming event, in addition to generating an outgoing event (and possibly a change of state), may also have one or more associated **local** or **specific actions**. Examples include *start a timer* and *increment the send sequence variable.*

We shall now expand upon all of these aspects of the specification of a protocol by considering the specification of the error control procedure associated with the idle RQ protocol. To simplify the description, we shall consider only a unidirectional flow of I-frames – from the source to the destination. In most applications, however, a two-way flow is needed and both sides require a primary and a secondary.

All finite state machines – and hence protocol entities – operate in an atomic way. This means that once an incoming event has started to be processed, all processing functions associated with the event, including the

generation of any outgoing event(s), local (specific) actions, and a possible change in state, are all carried out in their entirety (that is, in an indivisible way) before another incoming event is accepted.

To ensure this happens, the various incoming (and outgoing) event interfaces are decoupled from the protocol entity itself by means of queues. As we showed earlier in Figure 6.31, there is an additional pair of queues between the protocol entity and the transmit–receive procedure that controls the particular transmission control circuit being used. Similarly, there is a pair of queues between the protocol entity and the timer procedure. Normally, the latter is run at regular (tick) intervals by means of an **interrupt** and, if a timer is currently running, its current value is decremented by the tick value. If the value goes to zero, a **timer expired** message is returned to the protocol entity via the appropriate queue.

The role of the transmit–receive procedure is simply to transmit a preformatted frame passed to it or to receive a frame from the link and queue the frame for processing by the protocol entity. This procedure may also be run as a result of an interrupt, but this time from the transmission control circuit. Also, although in principle only a single input and output queue is necessary to interface the primary and secondary to their respective network layers, in practice a pair of queues is necessary at each interface in order to handle the duplex flows of primitives.

To simplify the specification procedure, we give each of the various incoming events, outgoing events, predicates, specific actions, and states associated with each protocol entity an abbreviated name. Prior to specifying the protocol, the various abbreviated names are listed and all subsequent references are made using these names. For the error control component of the idle RQ protocol, the list of abbreviated names for the primary is as shown in Figure 6.32.

Since each protocol entity is essentially a sequential system, we must retain information that may vary as different incoming events are received. This information is held in a number of **state variables**. Examples, for the primary, are the send sequence variable V(S) – Vs in the specification – which holds the sequence number to be allocated to the next I-frame to be transmitted, the PresentState variable which holds the present state of the protocol entity, RetxCount which is a count of the number of erroneous frames received. Typically, if either RetxCount or ErrorCount reaches its maximum limit then the frame is discarded, an error message is sent to the network layer above and the protocol (entity) reinitializes.

The three most common methods that are used for specifying a communication protocol are **state transition diagrams**, **extended event–state tables**, and high-level structured programs. In many instances, we define a protocol as a combination of these coupled with time sequence diagrams to illustrate the user service primitives associated with the protocol.

The formal specification of the primary is shown in Figure 6.33. In part (a) a state transition diagram is used, in part (b) an extended event–state table, and in part (c) structured pseudocode.

Incoming events

Name	*Interface*	*Meaning*
LDATAreq	LS_user	L_DATA.request service primitive received
ACKRCVD	MAC_provider	ACK-frame received from S
TEXP	TIM_provider	Wait-ACK timer expires
NAKRCVD	MAC_provider	NAK-frame received from S

States

Name	*Meaning*
IDLE	Idle, no message transfer in progress
WTACK	Waiting an acknowledgment

Outgoing events

Name	*Interface*	*Meaning*
TxFrame	MAC_user	Format and transmit an I-frame
RetxFrame	MAC_user	Retransmit I-frame waiting acknowledgment
LERRORind	LS_provider	Error message: frame discarded for reason specified

Predicates

Name	*Meaning*
P0	N(S) in waiting I-frame = N(R) in ACK-frame
P1	CRC in ACK/NAK-frame correct

Specific actions

[1] = Start_timer using TIM_user queue
[2] = Increment Vs
[3] = Stop_timer using TIM_user queue
[4] = Increment RetxCount
[5] = Increment ErrorCount
[6] = Reset RetxCount to zero

State variables

Vs = Send sequence variable
PresentState = Present state of protocol entity
ErrorCount = Number of erroneous frames received
RetxCount = Number of retransmissions for this frame

Figure 6.32 Abbreviated names used in the specification of the idle RQ primary.

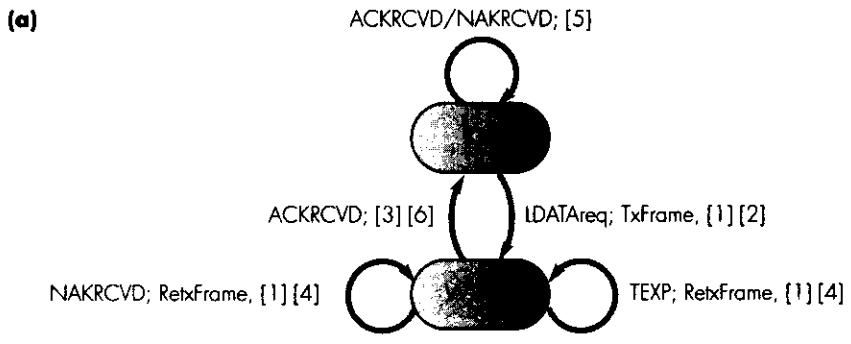


Figure 6.33 Specification of idle RQ primary in the form of: (a) a state transition diagram; (b) an extended event–state table; (c) pseudocode.

(b)

Present state \ Incoming event	LDATAreq	ACKRCVD	TEXP	NAKRCVD
IDLE	1	0	0	0
WTACK	4	2	3	3

0 = [5], IDLE (error condition)
1 = TxFrame, [1] [2], WTACK
2 = P0 and P1: [3] [6], IDLE
P0 and NOT P1: RetxFrame, [1] [4], WTACK
NOT P0 and NOT P1: [5], IDLE
3 = RetxFrame, [1] [4], WTACK
4 = NoAction, WTACK

(c)

```
program  IdleRQ_Primary;
const    MaxErrCount;
         MaxRetxCount;
type     Events = (LDATAreq, ACKRCVD, TEXP, NAKRCVD);
         States = (IDLE, WTACK);
var      EventStateTable = array [Events, States] of 0..4;
         PresentState : States;
         Vs, ErrorCount, RetxCount : integer;
         EventType : Events;
procedure Initialize;   } Initializes state variables and contents of EventStateTable
procedure TxFrame;      }
procedure RetxFrame;    } Outgoing event procedures
procedure LERRORind;    }
procedure Start_timer;  } Specific action procedures
procedure Stop_timer;   }
function  P0 : boolean; } Predicate functions
function  P1 : boolean; }

begin    Initialize;
         repeat Wait receipt of an incoming event
                EventType := type of event
                case EventStateTable [EventType, PresentState] of
                  0 : beginErrorCount := ErrorCount + 1; PresentState = IDLE;
                        if(ErrorCount = MaxErrCount) thenLERRORind end;
                  1 : beginTxFrame; Start_timer; Vs := Vs + 1; PresentState := WTACK end;
                  2 : beginif(P0 and P1) then begin Stop_timer; RetxCount := 0; PresentState := IDLE end;
                        else if (P0 and NOTP1) then beginRetxFrame; Start_timer;
                                                        RetxCount := RetxCount + 1;
                                                        PresentState := WTACK end;
                        else if (NOTP0 and NOTP1) then begin PresentState := IDLE; ErrorCount := ErrorCount + 1
                                                        if (ErrorCount = MaxErrorCount) then begin LERRORind; Initialize; end;
                      end;
                  3 : begin RetxFrame; Start_timer; RetxCount := RetxCount + 1; PresentState := WTACK;
                        if (RetxCount = MaxRetxCount) then begin LERRORind; Initialize; end;
                      end;
                  4 : begin NoAction end;
         until Forever;
end.
```

Figure 6.33 Continued.

Using the state transition diagram method, the possible states of the protocol entity are shown in ovals with the particular states written within them. **Directional arrows** (also known as **arcs**) indicate the possible transitions between the states, with the incoming event causing the transition and any resulting outgoing event and specific actions, written alongside. If, for example, an L_DATA.request (LDATAreq) is received from LS_user interface, then the frame is formatted and output to the MAC_user interface (TxFrame), a timer is started for the frame [1], the send sequence variable incremented [2], and the WTACK state entered. Similarly, if an ACK-frame is received with an N(R) equal to the N(S) in the waiting frame and the CRC is correct, then the timer is stopped [3] and the transitions can be interpreted in a similar way.

Although state transition diagrams are useful for showing the correct operation of a protocol, because of space limitations it is not always practicable to show all possible incoming event possibilities including error conditions. Hence most state transition diagrams are incomplete specifications. Moreover, with all but the simplest of protocols, we need many such diagrams to define even the correct operation of a protocol. It is for these reasons that we use the extended event–state table and the structured program code methods.

Using the extended event–state table method – as we see in part (b) of the figure – we can show all the possible incoming events and protocol (present) states in the form of a table. For each state, the table entry defines the outgoing event, any specific action(s), and the new state for all possible incoming events. Also, if predicates are involved, the alternative set of action(s). Clearly, the extended event–state table is a far more rigorous method since it allows for all possible incoming-event, present-state combinations. A basic event–state table has only one possible action and next-state for each incoming-event/present-state combination. It is the presence of predicates – and hence possible alternative actions/next states – that gives rise to the use of the term "extended" event–state table.

When we are interpreting the actions to be followed if predicates are involved, we must note that these are shown in order. Hence the action to be followed if the primary is in the WTACK state and an ACK-frame is received (ACKRCVD), is first to determine if P0 and P1 are both true. If they are, then carry out specific action [3] and [6] and enter the IDLE state. Else, determine if {P0 and NOTP1} is true, and so on. If neither condition is true then an error is suspected and the actions are shown.

A feature of the extended event–state table is that it lends itself more readily to implementation in program code than a state transition diagram. We can see this by considering the pseudocode specification of the idle RQ primary in Figure 6.33(c). In the figure this is shown as a program but in practice it is implemented in the form of a procedure or function so it can be included with the other protocol layers in a single program. However, this does not affect the basic operation of the program shown.

When each program (layer) is first run, the Initialize procedure is invoked. This performs such functions as initializing all state variables to their initial values and the contents of the EventStateTable array to those in the extended event–state table. The program then enters an infinite loop waiting for an incoming event to arrive at one of its input queues.

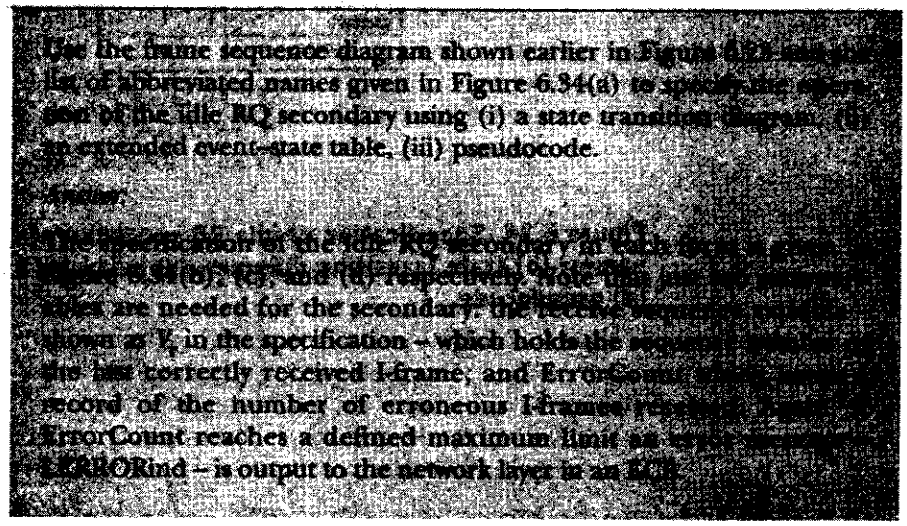
The incoming event which causes the program to run is first assigned to EventType. The current contents of PresentState and EventType are then used as indices to the EventStateTable array to determine the integer – 0, 1, 2, 3, or 4 – that defines the processing actions associated with that event. For example, if the accessed integer is 2, this results in the predicate functions P0 and P1 being invoked and, depending on their computed state (true or false), the invocation of the appropriate outgoing event procedure, coupled with any specific action procedures(s) as defined in the specification; for example, starting or resetting the timer, and updating PresentState.

We have simplified the pseudocode to highlight the structure of each program and hence the implementation methodology. No code is shown for the various outgoing event procedures nor for the predicate functions. In practice, these must be implemented in an unambiguous way using the necessary steps listed in the specification.

6.7.8 User service primitives

As we explained in the last section, to initiate the transfer of a block of information across the transmission line/link, the source network layer uses an ECB with a L_DATA.request primitive and the block of information within it. Similarly, the destination link layer (protocol), on receipt of an error-free I-frame containing the block of information, also uses an ECB to pass the block to the network layer with a L_DATA.indication primitive within it.

Example 6.9



(a) **Incoming events**

Name	*Interface*	*Meaning*
IRCVD	MAC_provider	I-frame received from P

States

Name	*Meaning*
WTIFM	Waiting a new I-frame from P

Outgoing events

Name	*Interface*	*Meaning*
LDATAind	LS_provider	Pass contents of received I-frame to user AP with an L_DATA.indication primitive
TxACK(X)	MAC_user	Format and transmit an ACK-frame with N(R) = X
TxNAK(X)	MAC_user	Format and transmit a NAK-frame with N(R) = X
LERRORind	LS_provider	Issue error message for reason specified

Predicates

Name	*Meaning*
P0	N(S) in I-frame = Vr
P1	CRC in I-frame correct
P2	N(S) in I-frame = Vr − 1

Specific actions

[1] = Increment Vr
[2] = Increment ErrorCount

State variables

Vr = Receive sequence variable
ErrorCount = Number of erroneous frames received

(b)

(c)

Present state \ Incoming event	IRCVD
WTIFM	1

1 = NOT P1: TxNAK, [2]
P1 and P2: TxACK
P0 and P1: LDATAind, TxACK, [1]

Figure 6.34 Specification of idle RQ secondary: (a) abbreviated names; (b) state transition diagram; (c) extended event–state table; (d) pseudocode.

(d)

```
program  IdleRQ_Secondary;
const.   MaxErrorCount;
type     Events = IRCVD;
         States = WTIFM;
var      EventStateTable = array [Events, States] of 1;
         EventType : Events;
         PresentState : States;
         Vr, X, ErrorCount : integer;
procedure Initialize;          } Initializes state variables and contents of EventStateTable
procedure LDATAind;            }
procedure TxACK(X);            } Outgoing event procedures
procedure TxNAK(X);            }
procedure LERRORind;           }
function  P0 : boolean;        }
function  P1 : boolean;        } Predicate functions
function  P2 : boolean;        }

begin    Initialize;
         repeat Wait receipt of incoming event; EventType := type of event;
                case EventStateTable[EventType, PresentState] of
                   1 : X := N(S) from Hframe;
                      if (NOTP1) then TxNAK(X);
                      else if(P1 and P2) then TxACK(X);
                      else if(P0 and P1) then begin LDATAind; TxACK(X); Vr := Vr + 1; end;
                      else begin ErrorCount := ErrorCount + 1; if (ErrorCount = MaxErrorCount) then
                                 begin LERRORind; Initialize; end;
                           end;
         until Forever;
end.
```

Figure 6.34 Continued.

For the error and flow control schemes we have outlined in the previous sections to function correctly, we have assumed that both communicating link protocols have been initialized so that they are ready to exchange information. For example, both sides of the link must start with the same send and receive sequence variables before any information frames are transmitted. In general, this is known as the initialization or **link setup** phase and, after all data has been exchanged across a link, there is a **link disconnection** phase. Since the link setup and disconnection phases are not concerned with the actual transfer of user data, they are collectively referred to as **link management**. The two link management functions are also initiated by the network layer (protocol) using an ECB and the set of primitives that we show in Figure 6.35(a). Since the primitives shown are in the same sequence as they are issued, this form of representing the various user service primitives associated with a protocol is known as a **time sequence diagram**. Note that to avoid the diagram becoming too cluttered, we have left off the two error indication primitives.

On receipt of an **L_CONNECT.request** primitive, the link protocol entity at the source initializes all state variables and then creates a **link SETUP** frame (PDU). This is sent to the correspondent (peer) link protocol entity in the destination using the selected transmission mode. On receipt of the SETUP frame, the destination initializes its own state variables and proceeds by sending an **L_CONNECT.indication** primitive to the correspondent LS_user and an acknowledgment frame back to the source.

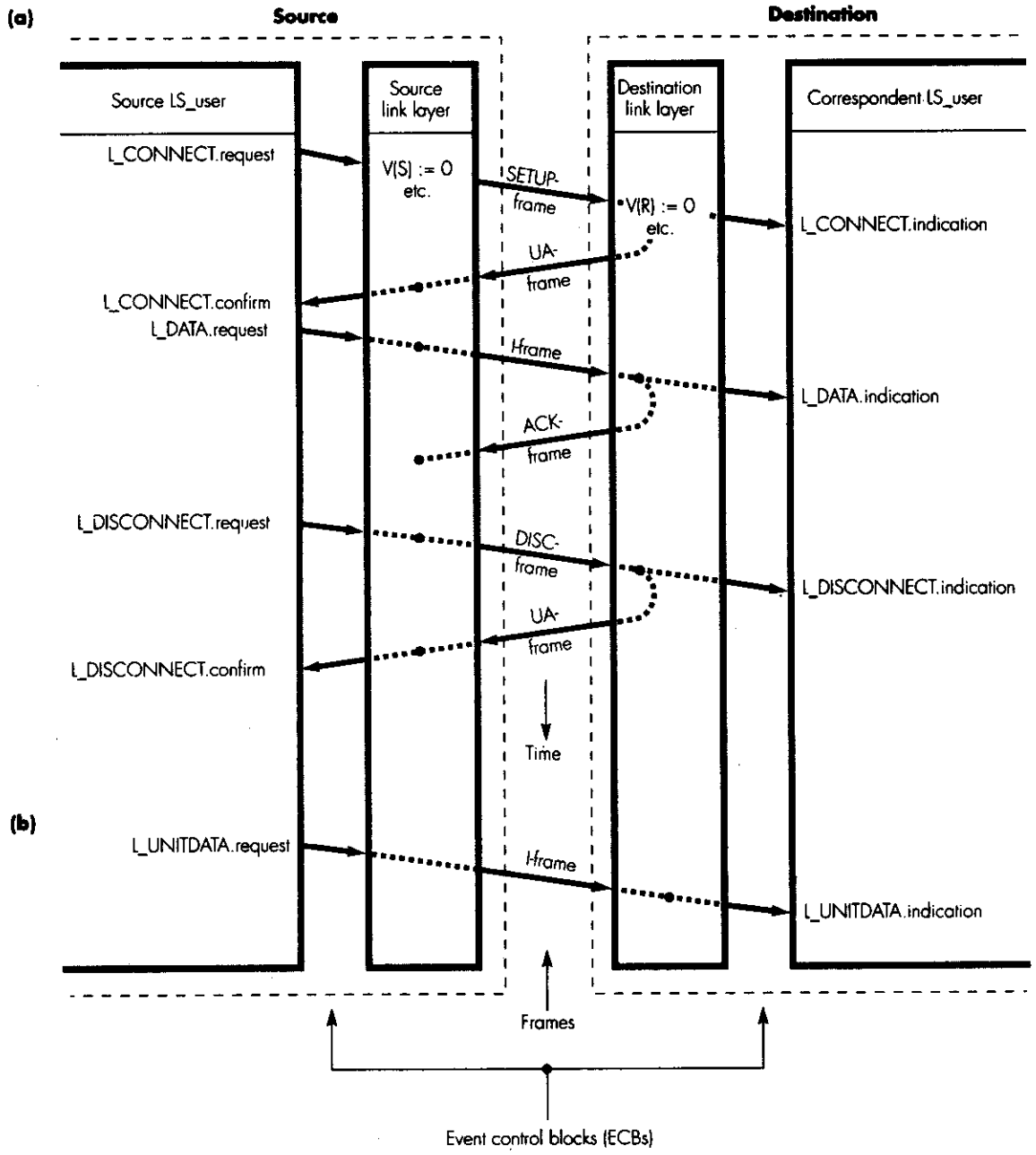


Figure 6.35 Time sequence diagram showing the link layer service primitives: (a) connection-oriented (reliable) mode; (b) connectionless (best-effort) mode.

Since this acknowledgment does not relate to an I-frame, it does not contain a sequence number. It is known, therefore, as an **unnumbered acknowledgment** or **UA-frame**. On receipt of this UA-frame, the source protocol entity issues the **L_CONNECT.confirm** primitive to the LS_user and the link is now ready for the transfer of data using the L_DATA service. Finally, after all data has been transferred, the setup link is released using the L_DISCONNECT service, which is also a confirmed service. The corresponding frame, known as a **disconnect** or **DISC frame**, is acknowledged using a UA-frame.

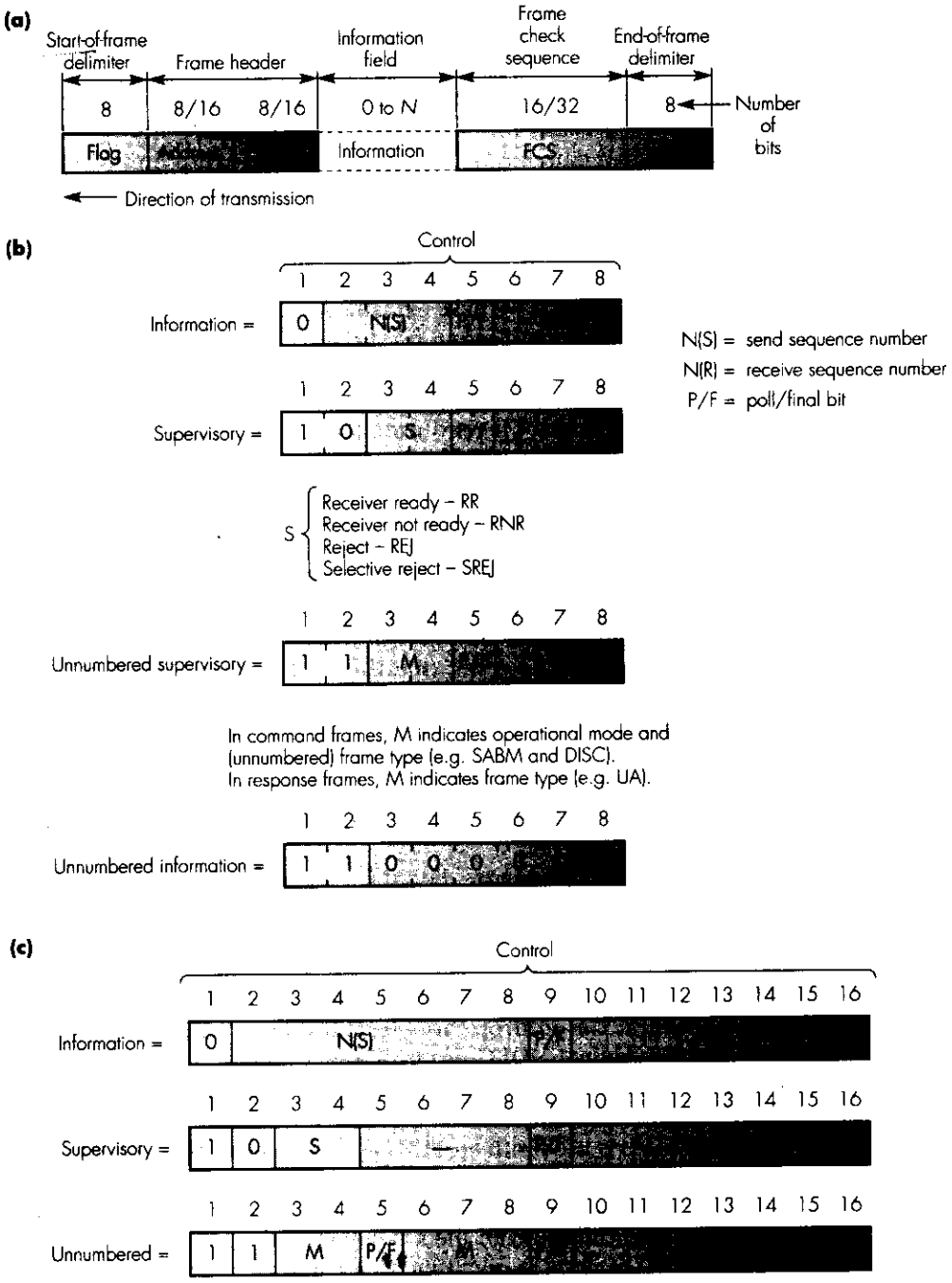
This mode of operation of the link layer is known as the connection-oriented mode and, as we have explained, it provides a reliable service. As we indicated at the start of Section 6.7, however, in many applications this mode of operation is not used and instead the simpler best-effort service is used in which the link layer protocol at the destination simply discards any frames received with errors. The two user service primitives associated with this mode are shown in Figure 6.35(b).

Since it is not necessary to set up a logical connection prior to sending blocks of information, this mode of operation is known as the connectionless mode and, in order to discriminate between the data transfer service associated with the two modes, as we show in the figure, the two service primitives used are **L_UNITDATA.request** and **L_UNITDATA.indication**. Also, since in the connectionless mode no retransmissions are used, no sequence numbers are required. Note, however, that the differences between the two modes occur only at the link control sublayer as both modes use the same medium access control sublayer.

6.8 The HDLC protocol

To conclude this chapter, we describe selected aspects of a practical example of a link layer protocol known as the **high-level data link control** (**HDLC**) protocol. This is an international standard that has been defined for use with a number of different network configurations and types. These include duplex point-to-point links as used over the access circuits associated with an ISDN, and half-duplex multidrop/broadcast links as used in some LANs. Hence there is the original HDLC protocol and a number of variants of it, each of which uses slightly different fields in the frame header and also a different MAC sublayer. Examples include the **link access procedure D-channel** (**LAPD**) which is used with an ISDN and the **logical link control** (**LLC**) which is used with LANs.

In HDLC, the frames sent by the primary to the secondary are known as **commands**, and those from the secondary to the primary as **responses**. Also, when the LC sublayer is operating in a connection-oriented (reliable) mode, all error and flow control frames are known as **supervisory frames** and the various frames that are used to set up and disconnect a logical link **unnumbered frames**. A standard format is used for all frames, however, and this is shown in Figure 6.36(a).



Note: With the indicated direction of transmission, all control field types are transmitted bit 8/16 first.

Figure 6.36 HDLC frame format and types: (a) standard/extended format; (b) standard control field bit definitions; (c) extended control field bit definitions.

As we can see, HDLC is based on a bit-oriented transmission control scheme with flags to indicate the start and end of each frame together with zero bit insertion and deletion to ensure the flag pattern (01111110) does not occur within the bitstream between the flags. The frame check sequence (FCS) is a 16-bit CRC that is computed using the generator polynomial:

$$x^{16} + x^{12} + x^5 + 1$$

The CRC is first generated using the procedure we described in Section 6.6.2 but an additional step is taken to make the check more robust. This involves adding sixteen 1s to the tail of the dividend (instead of zeros) and inverting the remainder. This has the effect that the remainder computed by the receiver is not all zeros but the bit pattern 0001 1101 0000 1111.

The various control field bit definitions are shown in Figure 6.36(b). The S-field in supervisory frames and the M-field in unnumbered frames are used to define the specific frame type. The send and receive sequence numbers – N(S) and N(R) – are used in conjunction with the error and flow control procedures.

The **P/F bit** is known as the **poll/final bit**. A frame of any type is called a **command frame** if it is sent by the primary station and a **response frame** if it is sent by a secondary station. The P/F bit is called the poll bit when used in a command frame and, if set, indicates that the receiver must acknowledge this frame. The receiver acknowledges this frame by returning an appropriate response frame with the P/F bit set; it is then known as the final bit.

The use of 3 bits for each of N(S) and N(R) means that sequence numbers can range from 0 through 7. This, in turn, means that a maximum send window of 7 can be selected. Although this is large enough for many applications, those involving very long links (satellite links, for example) or very high bit rates, require a larger send window if a high link utilization is to be achieved. The extended format uses 7 bits (0 through 127), thereby increasing the maximum send window to 127. This is shown in Figure 6.36(c).

The address field identifies the secondary station that sent the frame, and is not needed with point-to-point links. However, with multipoint links, the address field can be either eight bits – normal mode – or multiples of eight bits – extended mode. In the latter case, bit 1 of the least significant address octet(s) is(are) set to 0 and bit 1 is set to 1 in the last octet. The remaining bits form the address. In both modes, an address of all 1s is used as an all-stations broadcast address.

Unnumbered frames are used both to set up a logical link between the primary and secondary and to inform the secondary of the mode of operation to be used. For example, the set asynchronous balanced mode (SABM) command frame – indicated by the M-bits in the control field – is used to set up a logical link in both directions when a duplex point-to-point link is being used. Other examples are the DISC-frame (which is used to disconnect the

link) and the UA-frame, which is a response frame and is sent to acknowledge receipt of the other (command) frames in this class. Also, when operating in the connectionless (best-effort) mode, no acknowledgment information is required. Hence all information is transmitted in **unnumbered information** (**UI**) frames with the control field set to 11000000.

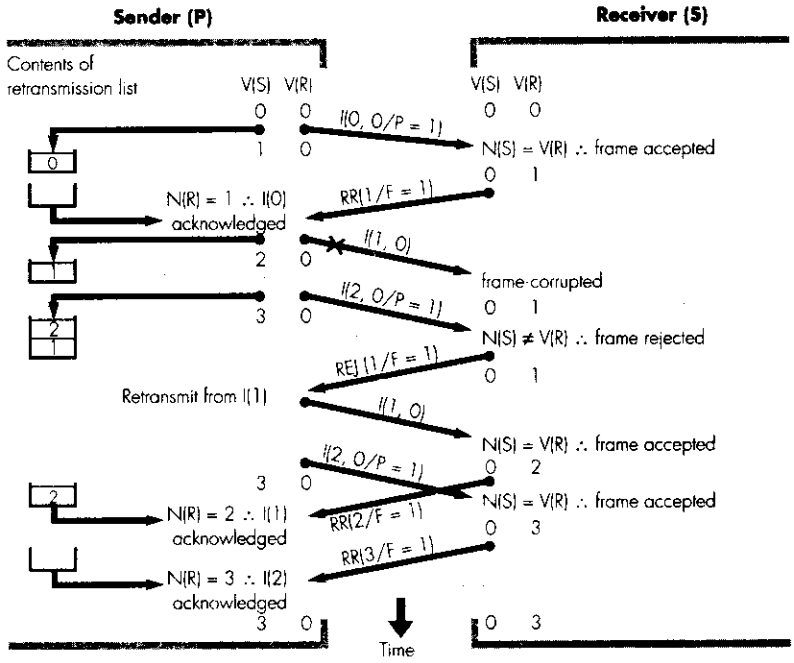
The four supervisory frames are used to implement a continuous RQ error control scheme and have the following functions:

- receiver ready (RR): this has the same function as the ACK-frame in Figures 6.27 and 6.28;
- reject (REJ): this has the same function as the NAK-frame in the go-back-N scheme;
- selective reject (SREJ): this has the same function as the NAK-frame in the selective repeat scheme;
- receiver not ready (RNR): this can be used by the secondary to ask the primary to suspend sending any new I-frames.

Each RR (ACK) frame contains a receive sequence number – N(R) – which acknowledges correct receipt of all I-frames awaiting acknowledgment up to and including that with a N(S) equal to N(R) – 1. This is slightly different from what we used earlier in the frame sequence diagrams in Figures 6.27 and 6.28 in which N(R) acknowledged I-frames up to N(S). This is because in HDLC the receive sequence variable V(R) is incremented before the ACK-frame is returned rather than after as we used in the earlier figures. An example frame sequence diagram showing the use of the RR (ACK) and REJ (NAK) frames and the P/F bit is given in Figure 6.37. The example relates to a go-back-N error control scheme and, for clarity, just a unidirectional flow of I-frames which means the link is operating in the **normal response mode** (**NRM**).

When interpreting the figure, remember that each I-frame contains an N(S) and an N(R), the latter acknowledging frames flowing in the reverse direction and known as a **piggyback acknowledgment**. Hence, since in this example there are no I-frames flowing in the reverse direction, the N(R) field in the I-frames is always 0. Also, since an I-frame is a command, when the P-bit is set, this means that S must acknowledge the frame immediately using a RR (ACK) frame with the appropriate N(R) within it and, since it is being returned in response to a command (P = 1), with the F-bit set to 1.

The example also shows the use of REJ (NAK) frames. When S detects that I-frame I(2,0/P = 1) is out of sequence, it returns an REJ (1/F = 1). P then retransmits frames I(1,0) and I(2,0) with the P-bit again set to 1 in frame I(2,0). The receiver acknowledges correct receipt of each frame with the F-bit set to 1 in the last RR-frame. If selective retransmission is being used, then frame I(2,0/P = 1) will be accepted and an SREJ-frame returned to request that frame I(1,0) is retransmitted.



Note: I (2, 0/P = 1) means N(S) = 2, N(R) = 0, P = 1
RR (1/F = 1) and REJ (1/F = 1) means N(R) = 1, F = 1

Figure 6.37 HDLC normal response mode: example frame sequence diagram with single primary and secondary (i.e. no piggyback acknowledgments).

An example showing the use of piggyback acknowledgments is given in Figure 6.38. As we can see, in this example there is a flow of I-frames in both directions simultaneously and hence each side contains a primary and a secondary shown as combined P/S in the figure. For clarity, no transmission errors are shown.

As each I-frame is received, the N(S) and N(R) contained in the header are both read. N(S) is first compared with the receiver's V(R). If they are equal, the frame is in the correct sequence and is accepted; if they are not equal, the frame is discarded and an REJ or SREJ-frame returned. N(R) is examined and used to acknowledge any outstanding frames in the retransmission list. Finally, as no further I-frames are awaiting transmission, an RR-frame is used to acknowledge the outstanding unacknowledged frames in each retransmission list.

Flow control is particularly important when two-way simultaneous working is used; that is, the link is being operated in the **asynchronous balanced mode (ABM)**. As we can see in the example shown in Figure 6.38, the send and receive sequence variables – and hence numbers – are incremented

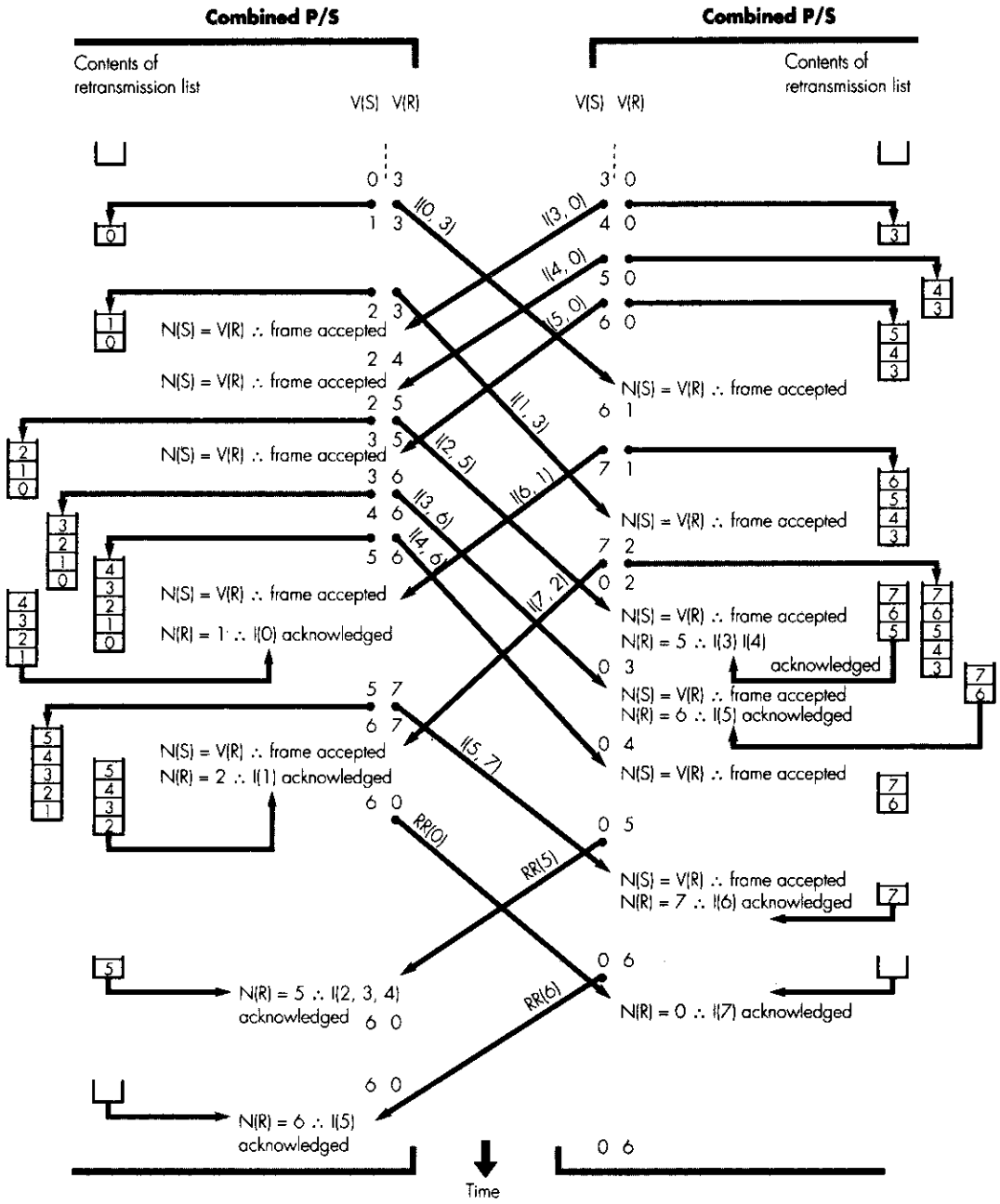


Figure 6.38 HDLC asynchronous balanced mode: piggyback acknowledgment procedure.

modulo 8 so the maximum send window K that can be used is 7. Thus a maximum of 7 I-frames can be awaiting acknowledgment in the retransmission list at any time. Each side of the link maintains a separate variable known as the retransmission count (RetxCount) which is initialized to zero when the logical link is set up. It is incremented each time an I-frame is transmitted, and hence each time a frame is placed in the retransmission list, and is decremented whenever a positive acknowledgment is received, and hence each time a frame is removed from the retransmission list. The primary stops sending I-frames when the retransmission count reaches K and does not resume until a positive acknowledgment is received either as a separate RR supervisory frame or piggybacked in an I-frame flowing in the reverse direction. We can conclude that transmission of I-frames is stopped when:

$$V(S) = \text{last } N(R) \text{ received} + K$$

Note that the window mechanism controls the flow of I-frames in only one direction and that supervisory and unnumbered frames are not affected by the mechanism. Hence, these frames can still be transmitted when the window is operating. An example with $K = 3$ is shown in Figure 6.39; for clarity, only the flow of I-frames in one direction is affected.

The use of a window mechanism means that the sequence numbers in all incoming frames must lie within certain boundaries. On receipt of each frame the secondary must check to establish that this is the case and, if not, take corrective action. Therefore each received N(S) and N(R) must satisfy the following conditions:

(1) V(R) is less than or equal to N(S) which is less than V(R) + K.
(2) V(S) is greater than N(R) which is greater than or equal to V(S) – RetxCount.

If N(S) equals V(R), then all is well and the frame is accepted. If N(S) is not equal to V(R) but is still within the range, then a frame has simply been corrupted and a REJ (go-back-N) or a SREJ (selective repeat) frame is returned, indicating to the primary that a sequence error has occurred and from which frame to start retransmission.

If N(S) or N(R) is outside the range, then the sequence numbers at both ends of the link have become unsynchronized and the link must be reinitialized (set up). This is accomplished when the secondary, on detecting an out-of-range sequence number, discards the received frame and returns a **frame reject** (**FRMR**) – ABM – or a **command reject** (**CMDR**) – NRM – frame to the primary. The primary discards all waiting frames and proceeds to set up the link again by sending an SABM/SNRM and waiting for a UA response. On receipt of this response, both sides of the link reset their sequence and window variables enabling the flow of I-frames to be resumed. In fact, this is only one reason why a link may be reset; another is the receipt of an unnumbered frame, such as a UA, during the data transfer phase which indicates

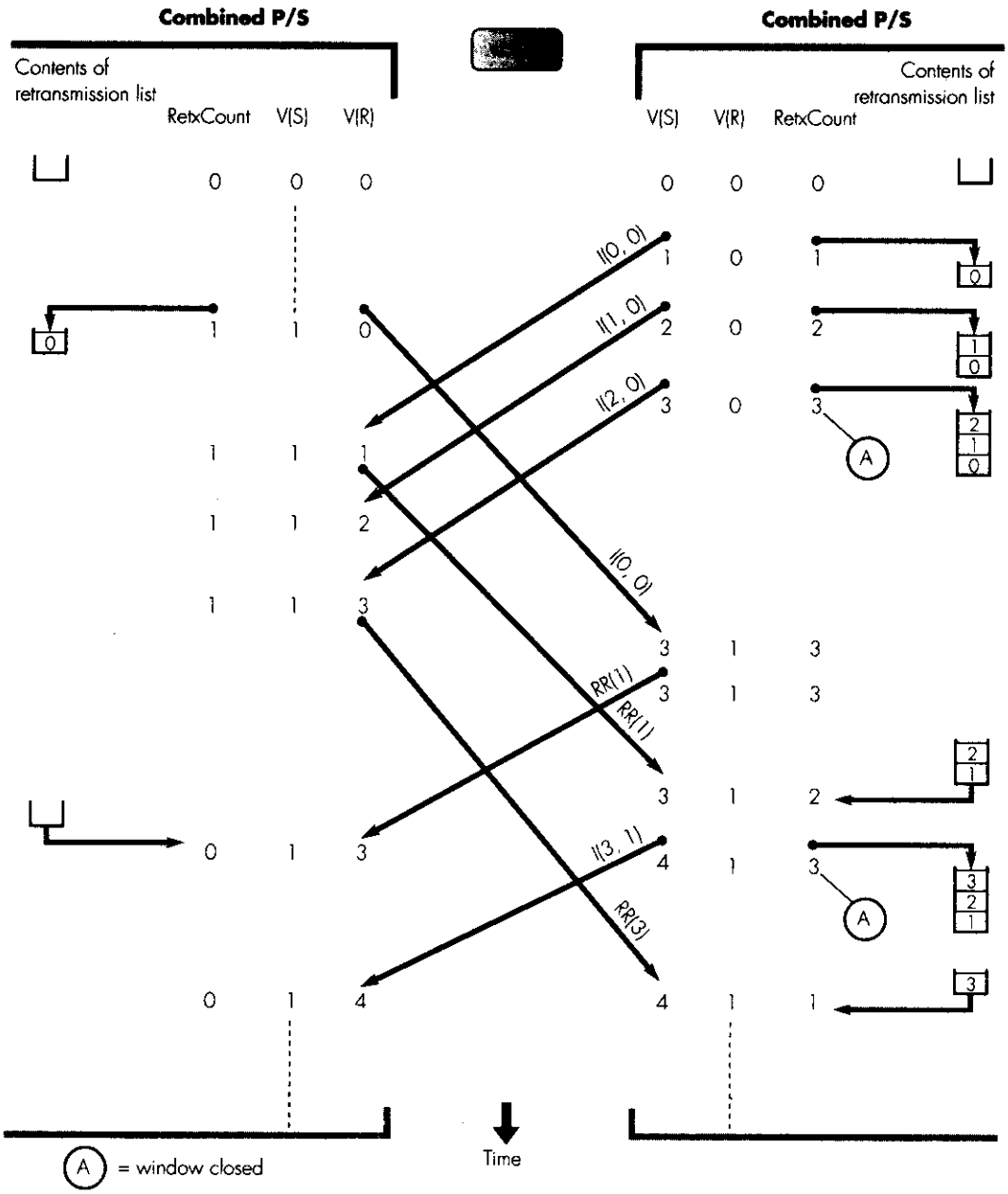


Figure 6.39 HDLC window flow control procedure.

that the primary and secondary have become unsynchronized.

The flow control procedure just outlined is controlled by the primary side of the link controlling the flow of I-frames according to the send window. In addition, it may be necessary for the secondary to stop the flow of I-frames as a result of some event occurring at its side of the link. For example, with a go-back-N retransmission strategy the receive window is 1 and it is reasonably straightforward to ensure that there are sufficient memory buffers available at the receiver. However, if selective retransmission is used, it is possible for the secondary to run out of free buffers to store any new frames. Hence, when the secondary approaches a point at which all its buffers are likely to become full, it returns an RNR supervisory frame to the primary to instruct it to stop sending I-frames. Acknowledgment frames are not affected, of course. When the number of full buffers drops below another preset limit, the secondary returns an RR-frame to the primary with a N(R) indicating from which frame to restart transmission.

A summary of the various service primitives and frame types (protocol data units) associated with HDLC is given in Figure 6.40(a). In practice, there are more unnumbered frames associated with HDLC than shown in the figure but, as mentioned earlier, the aim is simply to highlight selected aspects of HDLC operation. To reinforce understanding further, a (simplified) state transition diagram for HDLC is given in Figure 6.40(b). The first entry alongside each arc is the incoming event causing the transition (if any); the second entry is the resulting action. Note that a state transition diagram shows only the correct operation of the protocol entity; normally it is accompanied by a more complete definition in the form of an extended event–state table and/or pseudocode.

Summary

In this chapter we have discussed a range of topics relating to digital communications and these are summarized in Figure 6.41. As we have explained, the topics relate primarily to the way blocks of information/data are transmitted over a transmission line/channel reliably; that is, the received blocks are free of transmission/bit errors, are in the same sequence as they were submitted, and contain no duplicates. In the following chapters we shall show how many of the schemes we have described are utilized in the various types of network used to support multimedia applications.

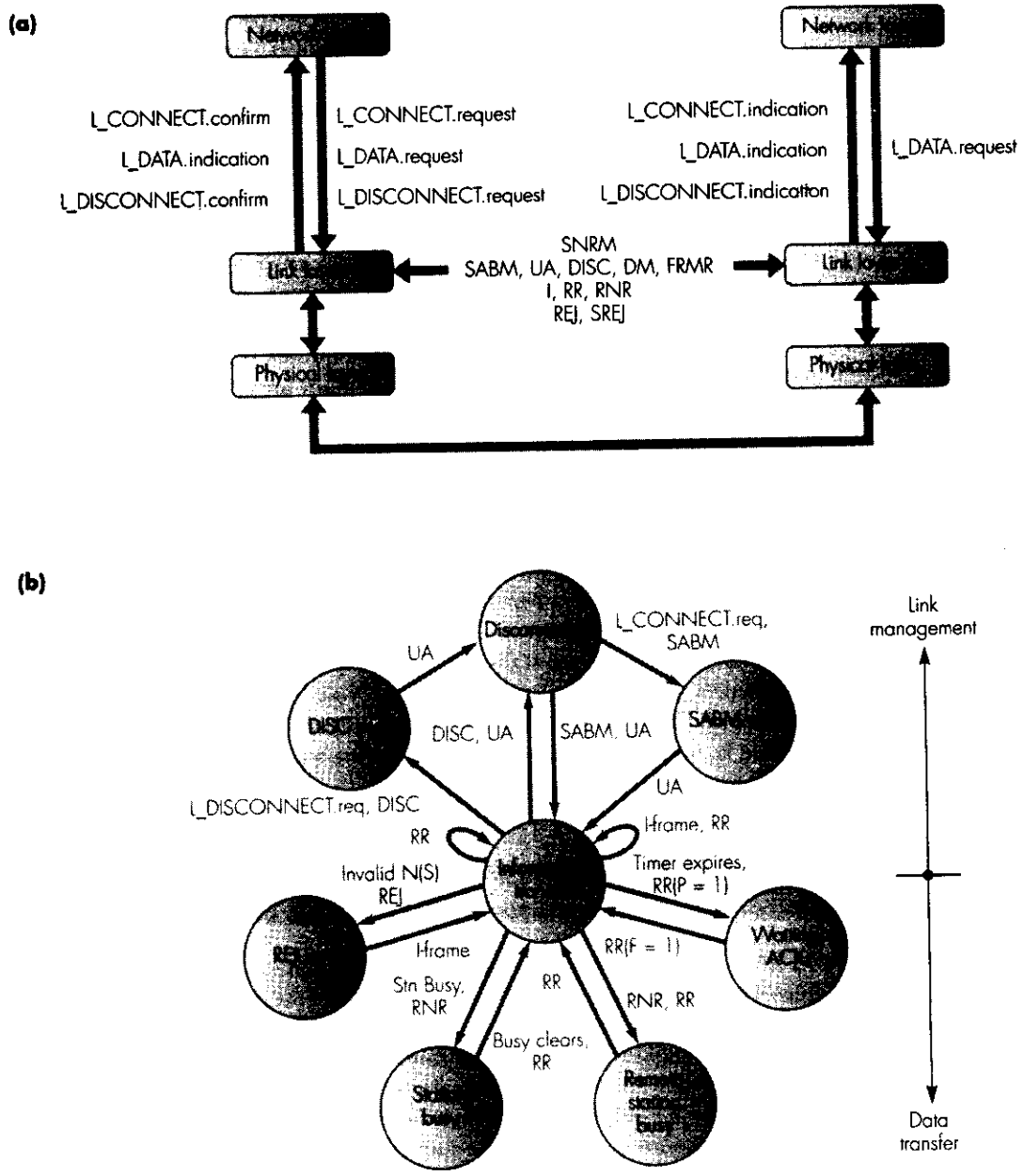


Figure 6.40 HDLC summary: (a) service primitives; (b) state transition diagram (ABM).

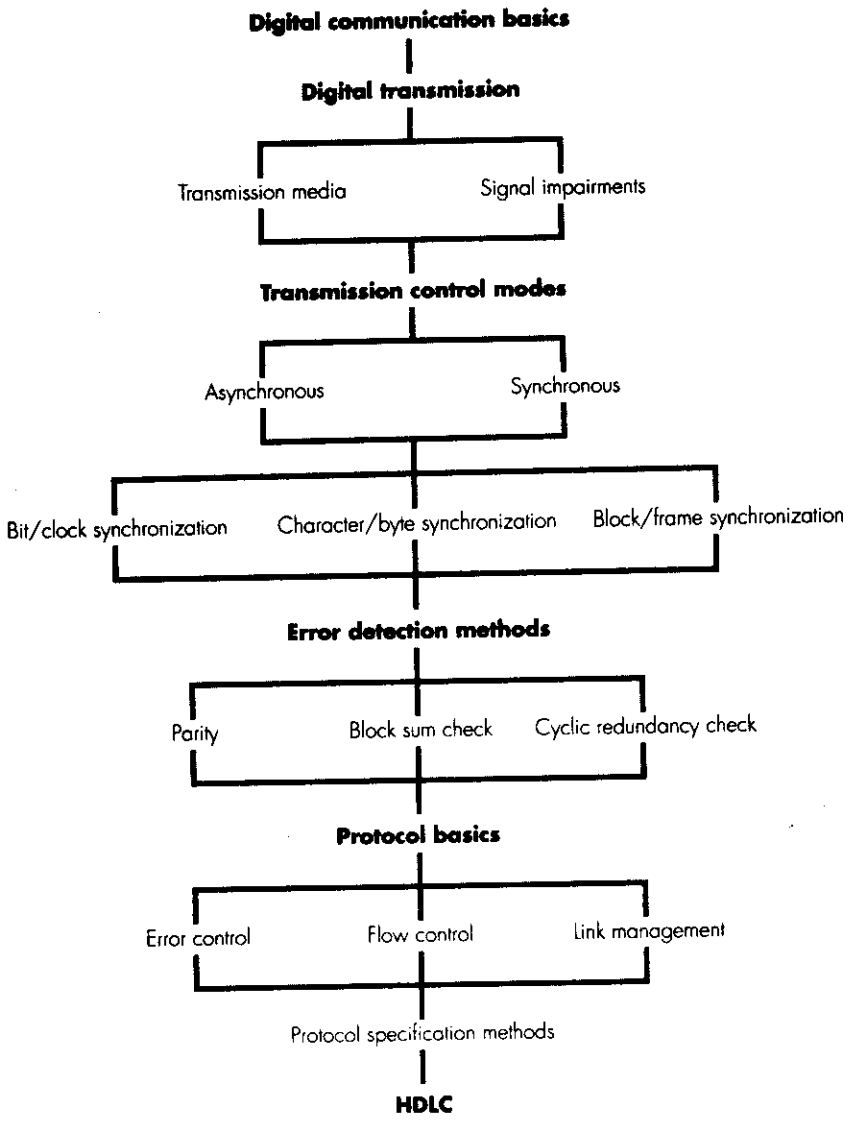


Figure 6.41 Summary of topics discussed relating to digital communications.

Exercises

Section 6.1

6.1 With the aid of the diagrams shown in Figure 6.1(a) and (b), explain the basic principles of the following transmission modes:
(i) baseband,
(ii) modulated.

6.2 With the aid of the waveform set shown in Figure 6.2, explain why the receiver samples the incoming signal as near to the center of each bit cell period as possible and, in some instances, bit errors occur.

Section 6.2

6.3 Explain why twisted-pair cable is preferred to non-twisted pair cable. What are the added benefits of using shielded twisted-pair cable?

6.4 With the aid of diagrams, explain the differences between the following transmission modes used with optical fiber:
(i) multimode stepped index,
(ii) multimode graded index,
(iii) monomode,
(iv) wave-division multiplexing.

6.5 State the meaning of the following relating to satellite systems:
(i) microwave beam,
(ii) transponder,
(iii) geostationary.

6.6 With the aid of diagrams, explain the operation of a satellite system that is used in
(i) TV broadcast applications and
(ii) data communication applications.

6.7 The maximum distance between two terrestrial microwave dishes, d, is given by the expression:

$$d = 7.14\sqrt{Kh}$$

where h is the height of the dishes above ground and K is a factor that allows for the curvature of the earth. Assuming $K = 4/3$, determine d for selected values of h.

6.8 With reference to Figure 6.6(b), determine the frequency assignments for a cellular system assuming a 7-cell repeat pattern. Explain the advantages of this over the 3-cell repeat pattern shown in the figure.

6.9 Explain the terms "signal propagation delay" and "transmission delay". Assuming the velocity of propagation of an electrical signal is equal to the speed of light, determine the ratio of the signal propagation delay to the transmission delay, a, for the following types of data link and 1000 bits of data:
(i) 100 m of UTP wire and a transmission rate of 1 Mbps,
(ii) 2.5 km of coaxial cable and a transmission rate of 10 Mbps,
(iii) a satellite link and a transmission rate of 512 kbps.

Section 6.3

6.10 With the aid of sketches, explain the effect on a transmitted binary signal of the following:
(i) attenuation,
(ii) limited bandwidth,
(iii) delay distortion,
(iv) line and system noise.

6.11 Assuming 8 signal levels are used to transmit a data bitstream of 10 kbps, derive
(i) the number of bits per original element, and
(ii) the signaling rate in baud.

6.12 Explain why the binary sequence 101010 ... is referred to as the worst-case sequence when deriving the minimum bandwidth requirements of a transmission line/channel.

6.13 Derive the minimum channel bandwidth required to transmit at the following bit rates assuming (a) the fundamental frequency only and (b) the fundamental and third harmonic of the worst-case signal are to be received:
(i) 500 bps,
(ii) 2000 bps,
(iii) 1 Mbps.

6.14 A modem to be used with a PSTN uses four levels per signaling element. Assuming a noiseless channel and a bandwidth of 3000 Hz, derive the Nyquist maximum information transfer rate in bps.

6.15 With the aid of a diagram and associated waveform set, explain the meaning of the term "adaptive NEXT canceler" and how such circuits can improve the data transmission rate of a line.

Section 6.4

6.16 Explain the difference between asynchronous and synchronous transmission.

Assuming asynchronous transmission, one start bit, two stop bits, one parity bit, and two bits per signaling element, derive the useful information transfer rate in bps for each of the following signaling (baud) rates:

(i) 300,
(ii) 600,
(iii) 1200,
(iv) 4800.

6.17 With the aid of a diagram, explain the clock (bit) and character synchronization methods used with an asynchronous transmission control scheme. Use for example purposes a receiver clock rate ratio of ×1 and ×4 of the transmitter clock.

6.18 With the aid of the diagrams shown in Figure 6.13, explain how frame synchronization is achieved with an asynchronous transmission control scheme assuming the data being transmitted is

(i) printable characters,
(ii) binary bytes.

Section 6.5

6.19 With the aid of the diagrams shown in Figure 6.14 relating to bit/clock synchronization with synchronous transmission, explain how synchronization is achieved using

(i) clock encoding,
(ii) DPLL.

6.20 With the aid of the waveform sets shown in Figure 6.15, explain

(i) the Manchester and
(ii) the differential Manchester clock encoding methods.

Why do both methods yield balanced codes?

6.21 (i) Explain under what circumstances data encoding and a DPLL circuit may be used to achieve clock synchronization. Also, with the aid of a diagram, explain the operation of the DPLL circuit.

(ii) Assuming the receiver is initially out of synchronism, derive the minimum number of bit transitions required for a DPLL circuit to converge to the nominal bit center of a transmitted waveform. How may this be achieved in practice?

6.22 Assuming a synchronous transmission control scheme, explain how character and frame synchronization are achieved:

(i) with character-oriented transmission,
(ii) with bit-oriented transmission.

6.23 Explain what is meant by the term "data transparency" and how it may be achieved using:

(i) character stuffing,
(ii) zero bit insertion.

Section 6.6

6.24 Explain the operation of the parity bit method of error detection and how it can be extended to cover blocks of characters.

Draw a circuit to compute the parity bit for a character and explain the difference between the terms "odd" and "even" parity.

6.25 With the aid of examples, define the following terms:

(i) single-bit error,
(ii) double-bit error,
(iii) error burst.

Produce a sketch showing the contents of a frame to illustrate the type of transmission errors that are not detected by a block sum check.

6.26 (a) Explain the principle of operation of a CRC error detection method. By means of an example, show how:
(i) the error detection bits are generated,
(ii) the received frame is checked for transmission errors. Use the generator polynomial.

$x^4 + x^3 + 1$

(b) Use an example to show how an error pattern equal to the generator polynomial used in (a) will not be detected. List other error patterns that would not be detected with this polynomial.

6.27 In an engine management system, the sixteen binary messages 0000 through 1111 are to be transmitted over a data link. Each message is to be protected by a 3-bit CRC generated using the polynomial:

$x^3 + x^2 + 1$

(i) Derive the 3 check bits for each of the three messages:

0000 0001 0010

(ii) State the meaning of the term "Hamming distance" and derive this for your code
(iii) Show how a single-bit error and a double-bit error in a transmitted codeword are detected at the receiver assuming the transmitted message is 1111
(iv) Give an example of a invalid received codeword that will not be detected.

Section 6.7

6.28 With the aid of frame sequence diagrams and assuming an idle RQ error control procedure with explicit retransmission, describe the following:
(i) the factors influencing the minimum time delay between the transmission of two consecutive information frames
(ii) how the loss of a corrupted information frame is overcome
(iii) how the loss of a corrupted acknowledgment frame is overcome.

6.29 A series of information frames with a mean length of 100 bits is to be transmitted across the following data links using an idle RQ protocol. If the velocity of propagation of the links is 2×10^8 ms^{-1}, determine the link efficiency (utilization) for each type of link.
(i) a 10 km link with a BER of 10^{-4} and a data transmission rate of 9600 bps
(ii) a 500 m link with a BER of 10^{-6} and a data transmission rate of 10 Mbps.

6.30 With the aid of frame sequence diagrams, describe the difference between an idle RQ and a continuous RQ error control procedure. For clarity, assume that no frames are corrupted during transmission.

6.31 With the aid of frame sequence diagrams, describe how the following are overcome with a selective repeat error control scheme:
(i) a corrupted information frame,
(ii) a corrupted ACK frame.

6.32 Repeat Exercise 6.31 for a go-back-N scheme.

6.33 Discriminate between the send window and receive window for a link and how they are related with:
(i) a selective repeat retransmission scheme,
(ii) a go-back-N control scheme.

6.34 With the aid of frame sequence diagrams, illustrate the effect of a send window flow control limit being reached. Assume a send window of 2 and a go-back-N error control procedure.

6.35 Assuming a send window of K, deduce the minimum range of sequence numbers (frame identifiers) required with each of the following error control schemes:
(i) idle RQ,
(ii) selective repeat,
(iii) go-back-N.

Clearly identify the condition when the maximum number of identifiers is in use.

6.36 With the aid of Figure 6.31, explain the meaning of the following terms:
(i) layered architecture,
(ii) interlayer queues,

(iii) local event,
(iv) user services,
(v) used services.

6.37 Using the abbreviated names listed in Figure 6.32, show how the idle RQ primary can be specified in the form of:
(i) a state transition diagram,
(ii) an extended event–state table,
(iii) a high-level pseudocode program.

6.38 With the aid of a time sequence diagram, show a typical set of link layer service primitives assuming the link layer provides
(i) a reliable service and
(ii) a best-effort service.

Section 6.8

6.39 In relation to the HDLC frame format shown in Figure 6.36, explain the meaning and use of the following terms:
(i) supervisory frames,
(ii) unnumbered frames,
(iii) poll/final bit,
(iv) command and response frames,
(v) extended control field bit definitions,
(vi) piggyback acknowledgment,
(vii) unnumbered information frame.

6.40 With the aid of the frame sequence diagram shown in Figure 6.37, explain how a corrupted I-frame is overcome in the normal response mode. Include in your explanation the use of the P/F bit.

6.41 With the aid of the frame sequence diagram shown in Figure 6.38, explain how the piggyback acknowledgment procedure works in the asynchronous balanced mode.

6.42 With the aid of the frame sequence diagram shown in Figure 6.39, describe the window flow control procedure used with the UDLC protocol.

Circuit-switched networks

7.1 Introduction

Although in Chapter 1 we discussed the operation and applications of a PSTN and an ISDN separately, in practice, as we shall expand upon later, both network types utilize the same core transmission and switching network to provide the related services and the only difference between the two networks is the way subscribers gain access to the core network. In this chapter, therefore, we shall discuss the operation of both a PSTN and an ISDN under the general heading of public circuit-switched networks.

All public circuit-switched networks consist of three hierarchical subnetworks:

- a relatively large number of **local access and switching networks**: these connect subscribers within a localized area to their nearest local exchange (LE) or end office (EO) and are concerned with the transmission and switching of calls within their own area;
- one or more **interexchange trunk/carrier networks**: these are national networks concerned with the transmission and switching of calls between different regional and national exchanges/offices;