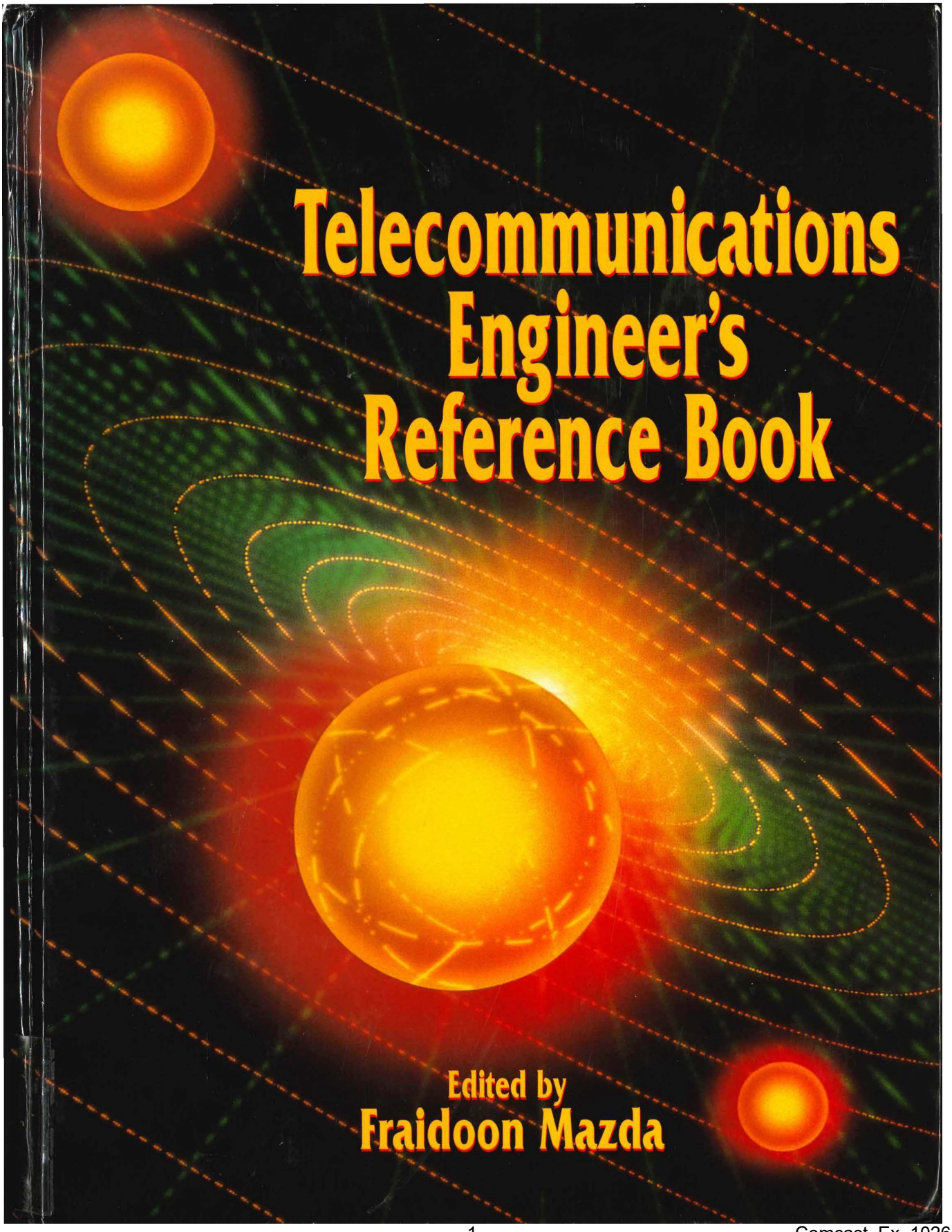

Telecommunications Engineer's Reference Book

Edited by
Fraidoon Mazda
MPhil DFH DMS MIMgt CEng FIEE

With specialist contributions

BUTTERWORTH HEINEMANN

Butterworth-Heinemann Ltd
Linacre House, Jordan Hill, Oxford OX2 8DP

PART OF REED INTERNATIONAL BOOKS

OXFORD LONDON BOSTON
MUNICH NEW DELHI SINGAPORE SYDNEY
TOKYO TORONTO WELLINGTON

First published 1993

British Library Cataloguing in Publication Data
Mazda, Fraidoon F.
Telecommunications engineer's reference book
I. Title
621.382

ISBN 0 7506 1037 9

Library of Congress Cataloguing in Publication
Mazda, Fraidoon F.
Telecommunications engineer's reference book/Fraidoon Mazda
p. cm.
Includes bibliographical references and index.
ISBN 0 7506 1037 9
1. Telecommunications. I. Title
TK5101.M37 1993
621.382–dc20 92–27846
CIP

Printed and bound in Great Britain

14

Coding

Dr M D Macleod
MA PhD MIEEE
University of Cambridge

Contents

14.1 The need for error control coding

In a digital (discrete) communication system, information is sent as a sequence of digits, which are first converted to an analogue (continuous) form by modulation at the transmitter, and then converted back into digits by de-modulation at the receiver. An ideal communication channel would transmit information without any form of corruption or distortion. Any real channel introduces noise and distortion, however, and these cause the corruption or loss of some digits at the receiver. The system designer can try to reduce the probability of digit errors by appropriate design of the analogue parts of the communication system, but it may be either impossible or too costly to achieve a sufficiently small probability of error in this way. A better solution will often be to use error control coding (ECC).

14.2 Principles of ECC

Error control coding is the controlled addition of redundancy to the transmitted digit stream in such a way that errors introduced in the channel can be detected, and in certain circumstances corrected, in the receiver. It is therefore a form of channel coding, so called because it compensates for imperfections in the channel; the other form of channel coding is transmission (or line coding), which has different objectives such as spectrum shaping of the transmitted signal. ECC is used to lower the probability of error from the input to the output of the communication system. The added redundancy means, however, that extra digits have to be transmitted over the channel, so that either the channel transmission rate must be increased, or the rate of transmission of digits from input to output must be reduced.

The controlled addition of redundancy in ECC contrasts with source coding (data compression) in which redundancy is removed from the source signal. For a signal (such as speech) with a high degree of intrinsic redundancy it would in principle be possible to perform some error detection and correction at the receiver without adding further redundancy. However this is generally too complex and too dependent on the uncontrolled redundancy of the source signal to be attractive. The functions of error control coding and source coding are therefore usually separated.

The functional blocks used for error control coding are a coder preceding the modulator in the transmitter, and a decoder following the demodulator in the receiver. The decoder may be designed to detect digit errors, or it may be designed to correct them. These functions are known as error detection and error correction respectively.

14.2.1 Types of ECC

There are two main types of ECC: block coding and convolutional coding. In block coding, the input is divided into blocks of k digits. The coder then produces a block of n digits for transmission, and the code is described as 'an (n,k) code'. Each block is coded and decoded entirely separately from all other blocks. In convolutional coding, the coder input and output are continuous streams of digits. The coder outputs n output digits for every k digits input, and the code is described as 'a rate k/n code'.

If the input digits are included unmodified in the coder output the code is described as systematic. The additional digits introduced by the coder are then known as parity or check digits. As well as the conceptual attractiveness of systematic codes, they have the advantage that a range of decoder complexities is made possible. The simplest decoder can simply extract the unmodified input digits from the coded digit stream, ignoring the parity digits. A more sophisticated decoder may use the parity digits for error detection, and a full decoder for error correction. Unsystematic codes also exist, but are less commonly used.

14.2.2 Feedforward and feedback error correction

In feedforward error correction (FEC) the decoder applies error correction to the received codeword, and it may also detect some uncorrectable errors. However, no return path from the receiver to the sender is assumed. Either block or convolutional codes may be used for feedforward error correction.

In feedback error correction, for which only block codes can be used, the receiver only attempts to detect errors, and sends return messages to the sender which cause repeat transmission if any errors are detected in a received block. In the OSI model for packet data networks, this function is carried out within the data link layer, by the return of a positive or negative acknowledgement (ACK or NAK) to the sender on receipt of a data block, a system known as stop-and-wait ARQ. In go-back-N ARQ, receipt of a NAK by the transmitter makes it retransmit the erroneous codeword and the N–1 following ones, where N is chosen so that the time taken to send N codewords is less than the round trip delay from transmitter to receiver and back again. This obviates the need for a buffer at the receiver. In selective repeat ARQ, only the codewords for which NAKs have been returned are re-transmitted. Performance analysis (Lin, 1983) shows that this is the most efficient system, although it requires an adequate buffer in the receiver.

14.2.3 Arithmetic for ECC

In a digital communication system, each transmitted digit is selected from a finite set of M values and is described as an M-ary digit. For example, binary digits (bits) have one of two values, which may be represented as 0 and 1. (The actual values of the physical signal used to transmit the digits, for example +12V and –12V, are irrelevant here). We shall assume that the input message digits use the same value of M as the transmitted digits; if not, the message digits can simply be converted to M-ary before coding.

The analytical design of coders and decoders for M-ary digits requires the use of Galois field arithmetic, denoted GF(M). If M is a prime number (including the important case of binary digits, where M = 2), Galois field arithmetic is equivalent to arithmetic modulo-M. GF(M) arithmetic also exists when M is equal to a power of a prime, so coders using the principles described below can be designed for quaternary (4-valued) and octal (8-valued) digit systems, for example, but in these cases GF(M) arithmetic is not equivalent to modulo-M arithmetic.

In the case of binary systems (M = 2), the addition and multiplication operations in GF(2) (i.e. modulo-2) arithmetic are as follows:

$$(0 + 0) = (1 + 1) = 0$$
$$(0 + 1) = (1 + 0) = 1$$
$$(0 \times 0) = (0 \times 1) = (1 \times 0) = 0$$
$$(1 \times 1) = 1.$$

Clearly, addition in this system is equivalent to the logical exclusive OR (XOR) function, and multiplication is equivalent to the logical AND function. For non-binary systems, multiplication and addition operations can be implemented using either dedicated logic or lookup tables.

Note that in modulo-2 arithmetic, addition and subtraction are equivalent. Some textbooks only discuss binary coders, and therefore treat all subtractions as additions. However for values of M other than 2 addition and subtraction are not equivalent, and it is essential to implement subtractions correctly.

14.2.4 Types of error

If the physical cause of digit errors is such that any digit is as likely to be affected as any other, the errors are described as random. A typical cause of random errors is thermal noise in the received signal. Other types of interference, however, may make it likely that when an error occurs, several symbols in succession will be corrupted; this is known as a burst error. A typical cause of burst errors is interference. Although the true behaviour of the channel may be more complex than either of these simple models, the random error and burst error models are simple, effective, and universally used for describing channel characteristics and error control code performance.

A channel used for transmitting binary digits is known as a binary channel, and if the probability of error is the same for 0s and 1s, the channel is called a Binary Symmetric Channel. The probability of error in binary digits is known as the bit error rate (BER).

14.2.5 Coding gain

Coding gain is a parameter commonly used for evaluating the effectiveness of an error correcting code, and hence for comparing codes. It is defined as the saving in energy per source bit of information for the coded system, relative to an uncoded system delivering the same BER. The effects of both error correction and the increase in transmission rate by a factor of n/k must be included when calculating the coding gain.

14.2.6 Criteria for choosing a code

The primary objective of error control coding will be to achieve a desired end-to-end probability of either uncorrected or undetected digit errors. The choice of code will depend on the error characteristics of the channel (particularly the random and burst error probabilities). The other important factors are likely to be the value of n/k (the increase in transmission rate over the channel), and the implementation complexity and cost of the coder and decoder.

14.3 Block coding

14.3.1 Single parity checks

The simplest block coder appends a single parity digit to each block of k message digits. This produces a systematic (k + 1, k) code known as a single parity code. For binary digits, the parity bit may be either the modulo-2 sum of the message bits or 1 minus that sum. The former case is known as even parity, because the sum of the k + 1 bits of the codeword (including the parity bit) is 0, and the latter is known as odd parity.

The decoder forms the sum of the bits of the received codeword. For even parity a sum of 1 means that there has been an error, and a sum of 0 is assumed to mean that the received codeword is correct. (For odd parity 0 indicates an error, and the correct sum is 1). Note that when an error is detected there is no way to deduce which bit(s) are in error; also, if more than one error occurs, and the number of errors is even, the single parity code will fail to detect them. This code is therefore a single error detecting (SED) code.

14.3.2 Linear block codes

The even parity code is the simplest example of a powerful class of codes called linear block codes. For such codes, the block of k message digits is represented as the k-element row vector d and the n digit codeword produced by the coder is represented by the n-element vector c. The function of the linear block coder is described by the Equation 14.1 where **G** is the (k x n) generator matrix.

$$\mathbf{c} = \mathbf{d}\,\mathbf{G} \tag{14.1}$$

The multiplications and additions in this equation are carried out in GF(M) arithmetic (i.e. modulo-2 for binary digits). The codewords generated by the equation are called valid codewords. Since there are 2^k possible datawords, only 2^k of the 2^n possible n-digit words are valid codewords.

Systematic linear block codes are produced by a generator matrix of the form shown in Equation 14.2, where $\mathbf{I}_k$ is the (k x k) unit matrix.

$$\mathbf{G} = [\,\mathbf{I}_k \,|\, \mathbf{P}\,] \tag{14.2}$$

When **G** has this form, the codeword c has the form of Equation 14.3, in other words the first k digits of the codeword equal the dataword, and the last n – k digits are parity digits.

$$\mathbf{c} = [\,\mathbf{d} \,|\, \mathbf{d}\,\mathbf{P}\,] \tag{14.3}$$

Let r be the received codeword; in the absence of errors $\mathbf{r} = \mathbf{c}$. The decoder performs the operation of Equation 14.4, where **H** is the ((n – k) x n) parity check matrix, to produce the (n – k) element syndrome s.

$$\mathbf{s} = \mathbf{r}\,\mathbf{H}^{\mathrm{T}} \tag{14.4}$$

H is chosen so that all valid codewords produce a zero syndrome; the syndrome then plays a crucial role in error correction. For the systematic code given above, the optimum form of parity check matrix is simply as in Equation 14.5.

$$\mathbf{H} = [\,\mathbf{P}^{\mathrm{T}} \,|\, \mathbf{I}_{n-k}\,] \tag{14.5}$$

For unsystematic codes, construction of the parity check matrix is more difficult.

14.3.3 Distance and code performance

The Hamming distance between two codewords is simply the number of bit positions in which they differ. If the Hamming distance between two codewords $\mathbf{c}_1$ and $\mathbf{c}_2$ is d, and $\mathbf{c}_1$ is transmitted, then errors would have to occur for codeword $\mathbf{c}_2$ to be received. More generally, if the minimum Hamming distance between codewords and any other valid codeword is d_M and $\mathbf{c}_1$ is transmitted, then the received codeword will not be a valid codeword if between 1 and $d_M - 1$ errors occur. The decoder could therefore detect up to $d_M - 1$ errors.

Assume that an invalid codeword is received. The distance (number of discrepancies) between it and all the valid codewords can

40 Multiplexers

J Hoolan
Dowty Communications Ltd

Contents

40.1 Introduction

The multiplexer is one of the most important components in communications networking. Its central function, from the network managers viewpoint, is to concentrate many users (or information channels) on to a single transmission channel in order to maximise the efficiency of that channel: it is used in almost every aspect of networking digital data, voice and video. This section will describe the advantages and disadvantages of different data multiplexing techniques, why these different techniques evolved to solve particular network engineering problems and how they fit in to modern networks.

Figure 40.1 shows the basic theoretical model for a multiplexer with the composite line speed exactly equal to the aggregate speed of the inputs.

Given a transmission channel, there are two ways the available bandwidth can be used: firstly by dividing the available bandwidth frequency spectrum into a subset of frequencies, each of which can then simultaneously use the transmission channel and allocate each frequency band to an input channel that needs to be multiplexed; or secondly, allocate all the available bandwidth to each channel for a fixed discrete time period. The first of these methods would be Frequency Division Multiplexing (FDM) and the latter Time Division Multiplexing (TDM). FDM is used primarily as an analogue solution to multiplexing and, for example, has been used extensively in telephony; indeed many of the FDM standards and techniques such as the multiplexing ratios dictated by the early designs of telephone exchange multiplexers (such as 24:1) are still in evidence in some of the latter digital exchanges.

This chapter is concerned however with the use and applications of latter type of multiplexing technique, Time Division Multiplexing which can be used in one of two modes, deterministic or non-deterministic. Deterministic TDM (which is the more commonly known just as TDM) allocates the available transmission channel bandwidth of a fixed regular basis to the input channels, whether they have data to send or not. Non-deterministic TDM or statistical TDM (which is more commonly just statistical multiplexing) allocates the available bandwidth to input channels only on demand when data is present.

40.2 Time Division Multiplexers

40.2.1 Principles

Time division multiplexing is the earliest and simplest form of digital multiplexing. It was developed to solve the communications problem created by the growth in remote computer processing during the 1960s. Computer terminals (or consoles) were locally attached in early computers but as computers became multi-user, so the need grew for more users gaining remote access over a single transmission line. Single users could gain access by simply attaching their terminal to a modem that converted the digital information by modulation into a voice frequency that could then be transmitted down a telephone to a modem at the receiving end that demodulated the signals back into digital form and hence onto the computer. However the need for groups of co-located users to gain access to computing resource meant that multiplexing techniques had to be applied to maximise the utilisation of rented PTT lease line circuits or costly dial up calls.

The early forms of multiplexers, which evolved to meet this growth in computing power, were designed for use with asynchronous data as this was the most commonly available type of terminal and printer. Typically running at speeds of up to 1200bit/s and with code formats of 5 and 7 bits (Baudot and ASCII respectively). As computer manufacturers moved to synchronous data, as a more efficient form of communications, so the design of multiplexers was adapted for handling this type of traffic. It was in this period that many of the design techniques to overcome specific engineering and networking problems evolved.

In common with all multiplexing techniques the primary requirement was for the two multiplexers to have some form of synchronisation procedure whereby they could calculate the location of each channel. This was done by defining a fixed length composite frame format, as in Figure 40.2, that had a unique synchronisation code at the start. When the link was first started the multiplexers would enter into a 'training' period, with only empty frames plus the synchronisation 'header' code being sent to each other to determine the frame boundaries. The composite overhead or bandwidth needed could be taken by reducing the available aggregate from the input channels.

Starting with the first commercially available asynchronous multiplexers the initial multiplexing technique was bit interleaved. In this method, the incoming data on each channel is sampled bit by bit and stored in a transitory central buffer before being assembled on to a composite. This is the simplest technique that can be employed as it only requires that the multiplexer recognises the bit boundaries of the incoming data. Providing that the sequence of data bits is then forwarded to the remote end and demultiplexed without losing integrity, then any type of data format can be handled. However, as this meant sending all data bits for an asynchronous character including the stop, start and parity bits, the multiplexing technique adopted quickly moved to byte interleaved. In this method the ancillary bits could be stripped off and only the data sent. At the receiving end they could be reconstituted. The byte was

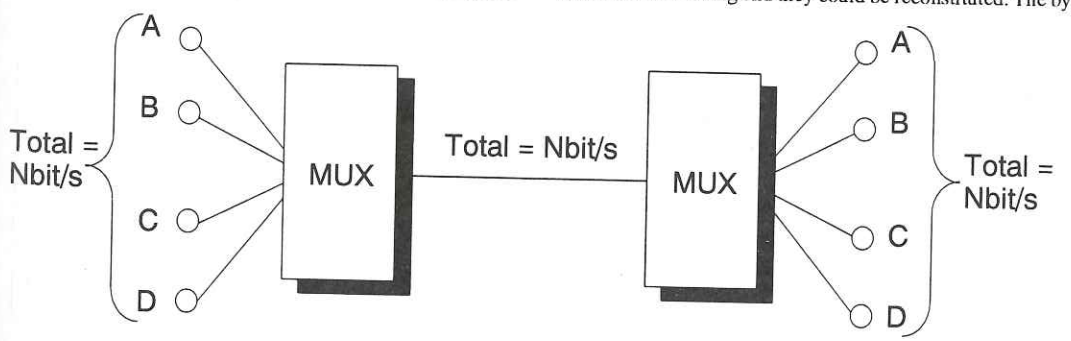

Figure 40.1 Basic model for a multiplexer

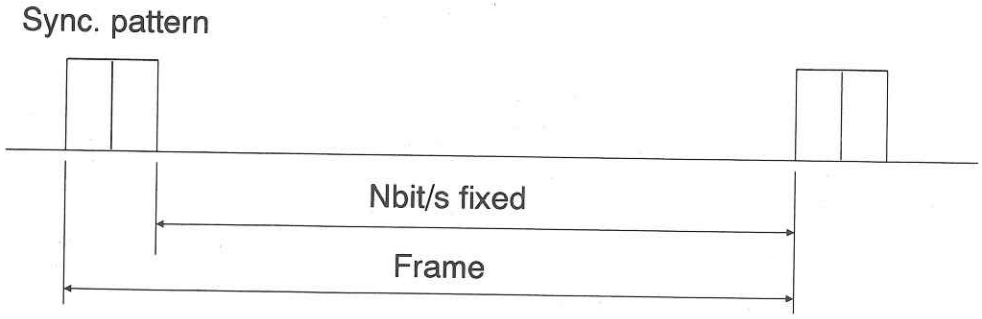

Figure 40.2 The basic operation of framing

8 bits with data requiring 7 bits and the 8th bit in each byte indicating whether data or controls were being sent. Although it meant that 5 bit data such as Baudot had 2 bits per byte wasted and that 8 bit codes could not be sent, as the majority of traffic was based on 7 bit code (ASCII), the overall gains made it worthwhile.

As stated above, the original design was for handling asynchronous traffic and the only rules for configuration would involve working out the total aggregate input and formulating the best 'scanning' method to cater for all the different speeds, so that they could mapped on to the available composite bandwidth. According to the design of the multiplexer a small percentage of the bandwidth had to be allocated for the synchronisation pattern, or sequence, so the general design rule was to limit the overhead to around 1% or 2 bits in a 200 bit frame. In addition, if the status of the V.24 (RS232) control lines such as Data Terminal Ready (circuit 108/1 or 2) or Request to Send (circuit 105), then this had to be allocated bandwidth as if it were data. As an example, if a half duplex circuit is being employed, which means that the data and control signals have to be kept contiguous, then the control information could require as much bandwidth as the data. The result is that the bandwidth actually available for data could be less than the theoretical maximum by up to 50%. The technique developed to overcome this problem is to use multi-frames with two types of controls, high and low priority. High priority is dealt with as described above. Low priority control information is sent every N frames, where N could be from 4 to 64 depending on the frame size and speed of the composite.

Figure 40.3 shows the ideal model of time division multiplexing operation, where all eight channels are being efficiently multiplexed onto a composite line. At any one instant in time there is always one of the channels using the available bandwidth. Figure 40.4 shows the realistic utilisation of the line. Note that only channels 1, 2 and 6 have any data to send. However, time slots in the form of bandwidth still have to be allocated since the TDM cannot recognise the absence of real data; this is the trade-off against data transparency and speed.

To summarise, the technique of sending asynchronous data by TDM is to use a frame synchronised, byte interleaved method with facilities for control information using a redundant bits.

The advantage is transparency to data format; any format of asynchronous data can be sent, at any speed.

The disadvantages are:

1. Fixed speeds; it was usually difficult to change any speeds without completely re-programming the multiplexers.
2. Poor efficiency; input channels always have a fixed allocation of composite bandwidth, whether they have data to send or not. As asynchronous data is by definition 'bursty' interactive type traffic, there will almost certainly be long periods of inactivity which still has bandwidth being allocated to it.
3. Prone to errors; in the process of being transmitted, any errors generated on the composite circuit are passed to the receiving end without any indication to the end device (in fact the parity recovery method makes this worse as it hides dual bit errors).

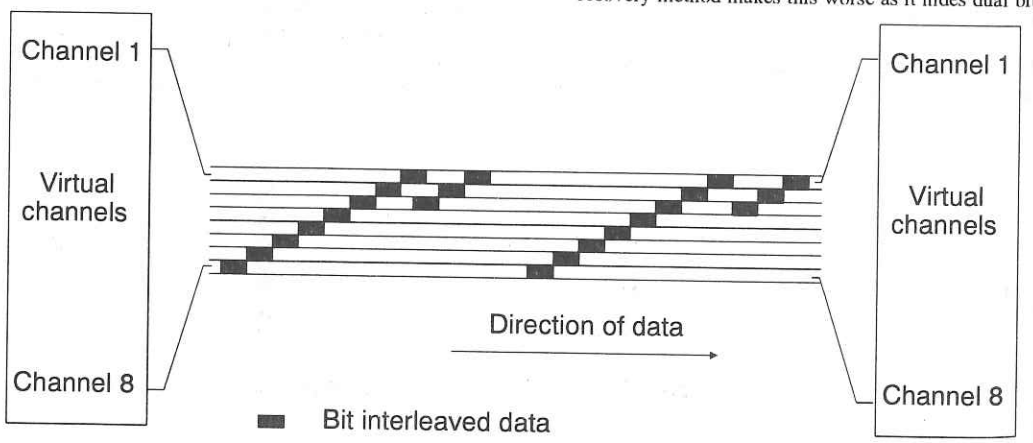

Figure 40.3 Ideal model of a TDM

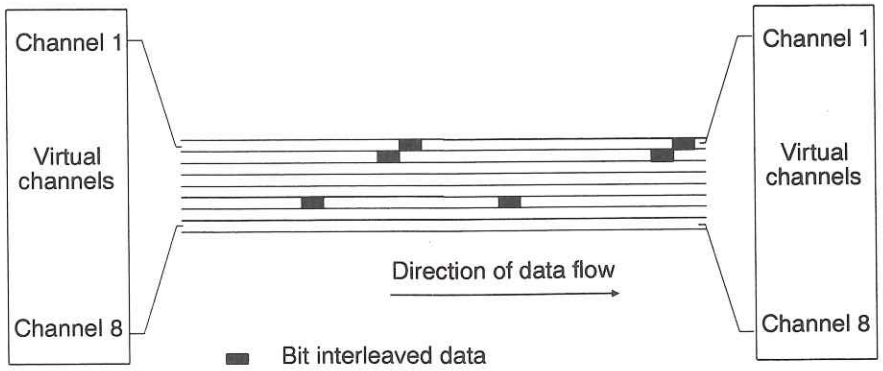

Figure 40.4 Line utilisation in a typical TDM

Unless the link synchronisation actually fails the remote end has no indication that there is a problem on the line.

40.2.2 Synchronous TDM

As synchronous protocols evolved the need to multiplex them grew; networks with remote sites that had a mix of asynchronous and synchronous terminals needed to connect them to the central site computing resource.

For synchronous TDMs, the bit-interleaved method was adopted as the most superior technique. The main difference from asynchronous multiplexing being that as well as keeping data integrity, the multiplexers had to keep the associated clock timing synchronisation so that the demultiplexed recovered data could have a regenerated clock associated with it. This point will be referred to later as it imposes major restrictions on the design and operation of networks and techniques have been developed to solve the problem. Apart from timing considerations, the synchronisation header in the composite frame has to be included in the overall bandwidth calculation.

In asynchronous data the actual speed of transmission can vary, according to the manufacturer's specification, by up to ±15%, therefore devices running overspeed were a particular problem. Synchronous data has a much tighter specification and may only vary by 0.1%, as recommended by CCITT V.28. This is particularly important for synchronous modems which cannot tolerate any deviation from the specified speed.

The technique of 'robbing' the low speed input channels and allocating it to the composite was normally only applied within these constraints. Two alternatives were:

1. To designate one of the input channels as a 'vari-speed' input which may be attached to non-critical devices. This approach, for obvious reasons, has to be handled with caution.
2. By the preferred technique used by latter designs to permanently allocate a synchronisation overhead.

A mixture of bit and byte interleaving with a fixed synchronisation header was also adopted by some high order multiplexers, where multiple voice channels needed to be multiplexed after being digitally encoded; the US Bell D1 system for example. The basic technique employed is for an analogue sample of the voice signal to be converted into an 8 bit code word, after being manipulated through a compander circuit operating on a logarithmic law to reduce the signal to noise level. The Bell D2 system used bit robbing for signalling purposes. The standards which has been adopted for this type of high order multiplexing are discussed later in this Chapter e.g. the CCITT standard G.711 for pulse code modulation.

From the viewpoint of line efficiency between two point to point multiplexers, the advantages and disadvantages of synchronous TDM compared to asynchronous TDM are almost identical. The main exception is that error detection and correction is not an issue with a synchronous data, as it will be enveloped in its own protocol independent from the multiplexer and therefore will have its own error recovery mechanisms via retransmission. However, from a networking viewpoint, the network timing recovery mechanisms and the methods by which the nodes in a network can be synchronised together, is a major issue.

Figure 40.5 shows a more realistic model of a multiplexer, with the overheads taken into account, that results in a composite that

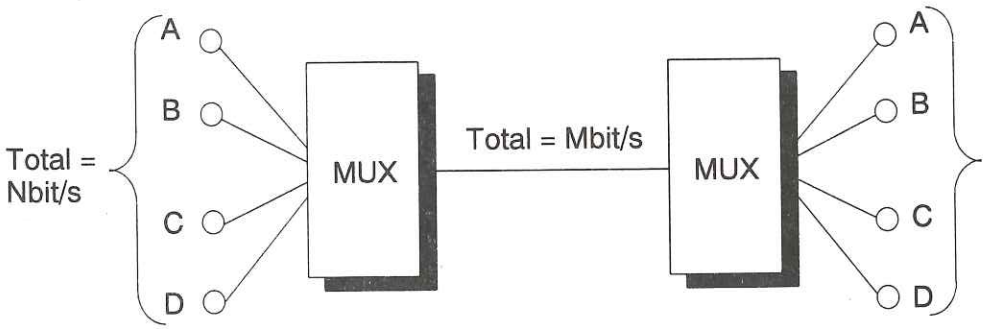

Figure 40.5 Model of a multiplexer with overheads considered

must be faster than the input aggregate. The value of M is given by Equation 40.5, therefore M > N.

$$M = A + B + C + C + \textit{synchronisation header} + \textit{control information} \quad (40.1)$$

In summary it can be stated that time division multiplexing has established itself as the most efficient technique for handling synchronous protocols and data at different speeds, even though it suffered from the fact that bandwidth is always wasted. It is not however very effective at handing asynchronous data due to the inflexibility and lack of error detection and correction. It is from this major disadvantage that the use of statistical multiplexers evolved.

40.3 Statistical time division multiplexing

40.3.1 Principle of Operation

Caught between the exponential growth of computing power and the use of asynchronous terminals as a cheap and convenient way of accessing computers, the statistical TDM was developed. It was first formulated in the early 1970s and in particular on the Arpanet network in the USA from which the X.25 packet switch standard evolved (Davis, 1973).

As stated above, the main problems with using TDM techniques on asynchronous data were lack of error detection and correction, and poor efficiency on the composite line, especially since the average asynchronous device is only active for 5 to 10% of the time.

The solution for asynchronous multiplexers was to use a formal protocol between the multiplexer nodes which could concentrate the data and provide the means of detecting and recovering from errors on the line. The protocol invariably chosen was based on a High Level Data Link Control (HDLC) which in turn had been derived from the IBM's SDLC protocol. This protocol had several design advantages over previous synchronous protocols such as IBM's 3270, as follows:

1. It was solely concerned with the maintenance of the data link and did not have to get involved with device control.
2. It had one common frame format for data, control and supervision with a unique flag indentifier for synchronisation.
3. It was bit oriented and therefore inherently transparent to data structures. However, it used byte aligned boundaries to permit byte interleaving, if required.
4. It had a Frame Check Sequence (FCS), in the form of a Cyclic Redundancy Check sum (CRC), that allowed error checking of the complete contents of all frames and thereby providing a mechanism for error recovery.
5. It was being considered as an international standard.

There were still drawbacks however, in that early versions of HDLC only allowed one virtual circuit (or user) per frame, and therefore modifications were made by many manufacturers to allow several users to share the same frame. In addition the multiplexers needed to pass not just data but also various other types of information such as:

1. The status conditions of the V.24 (RS232) interface controls.
2. Test information, such as setting a specific remote channel into loopback for testing purposes.
3. Information for validation of a channel connection, to check that it is working end to end.
4. A SPACE condition for so many milliseconds.
5. Information to set remote buffers into an XOFF state so that no more data should be sent until further notification.

It also had to allow the input channels to locally emulate some asynchronous polled protocols e.g. HP3000, so that polling could be locally acknowledged rather than end-to-end. The link need only carry a poll count and report polling exceptions.

The standard HDLC format has no inherent mechanism for allowing these facilities as it was designed purely to control the link level functions. It is up to the designer to implement this in the information (data) fields in order to make the product as flexible as possible.

Figure 40.6 shows a basic HDLC Information frame with the unique flag identifier 7E (hexadecimal), the address field and the control field, which in this case is indicating that the frame contains data.

Figure 40.7 shows a modified HDLC information frame format, used by a proprietary STDM manufactured by Dowty Case, other manufacturers using different encoding techniques. Here the information frame does not require an address field. It does, however, require a mapping field for each block of 4 bytes for a given channel, which indicates whether the following bytes are data or control and timing codes e.g. DTR dropped, set channel loopbacks, set SPACE condition for 100mS.

Given the mechanisms of a link protocol it was a small step to design a multiplexer that only transmitted data when there was data to sent. By scanning the input channels for activity and placing any data present into a common buffer pool, a frame could be assembled with data from different channels, plus any control information, and transmitted to the multiplexer at the other end of the link for demultiplexing.

Since HDLC uses a go-back-N frame sequencing count, any errors on the line due to a bad CRC or out of sequence frame would be detected and a request for the frame to be retransmitted issued from the remote end (also known as Automatic Repeat Request or ARQ). At that time, no standards existed for describing how the

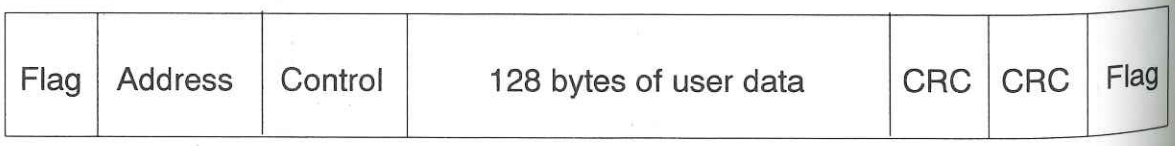

Figure 40.6 Basic HDLC information frame

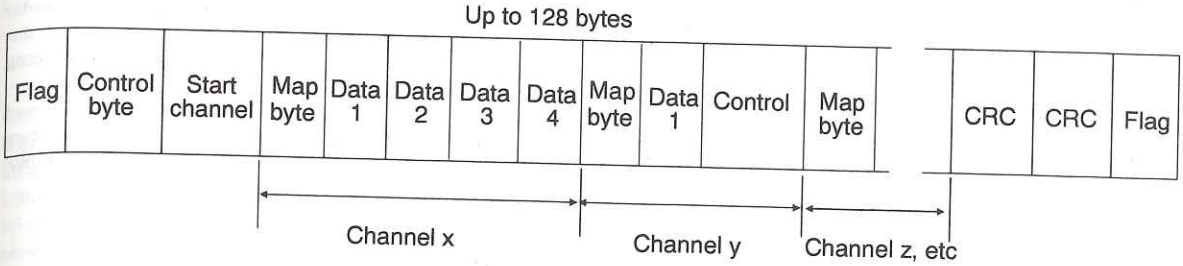

Figure 40.7 Proprietary HDLC information frame format (Dowty Case Communications Ltd.)

frames should be constructed for multiple users and interface control information, so every manufacturer developed their own proprietary standard, compared to the present development of LAPD which overcomes these restrictions. The outcome of this was to see a generation of proprietary statistical multiplexers before the establishment of X.25.

Figure 40.8 shows the simplified operation of a statistical multiplexer which operates by scanning the input buffers for data presence (indicated by the flag). Data is transferred into a buffer queue waiting to be assembled and transmitted as a frame.

40.3.2 Features

By definition, statistical multiplexers have to concentrate many input channels onto a composite link, where the aggregate input speeds may temporarily exceed the available composite bandwidth. In this situation it is essential that the multiplexer has the facility to flow control the data input, either by use of XON/XOFF or V.24 control signals such as DTR or CTS. The decisions by which the network designer decided how to size a network with the mix of input speeds and output speeds were assisted by queuing theory.

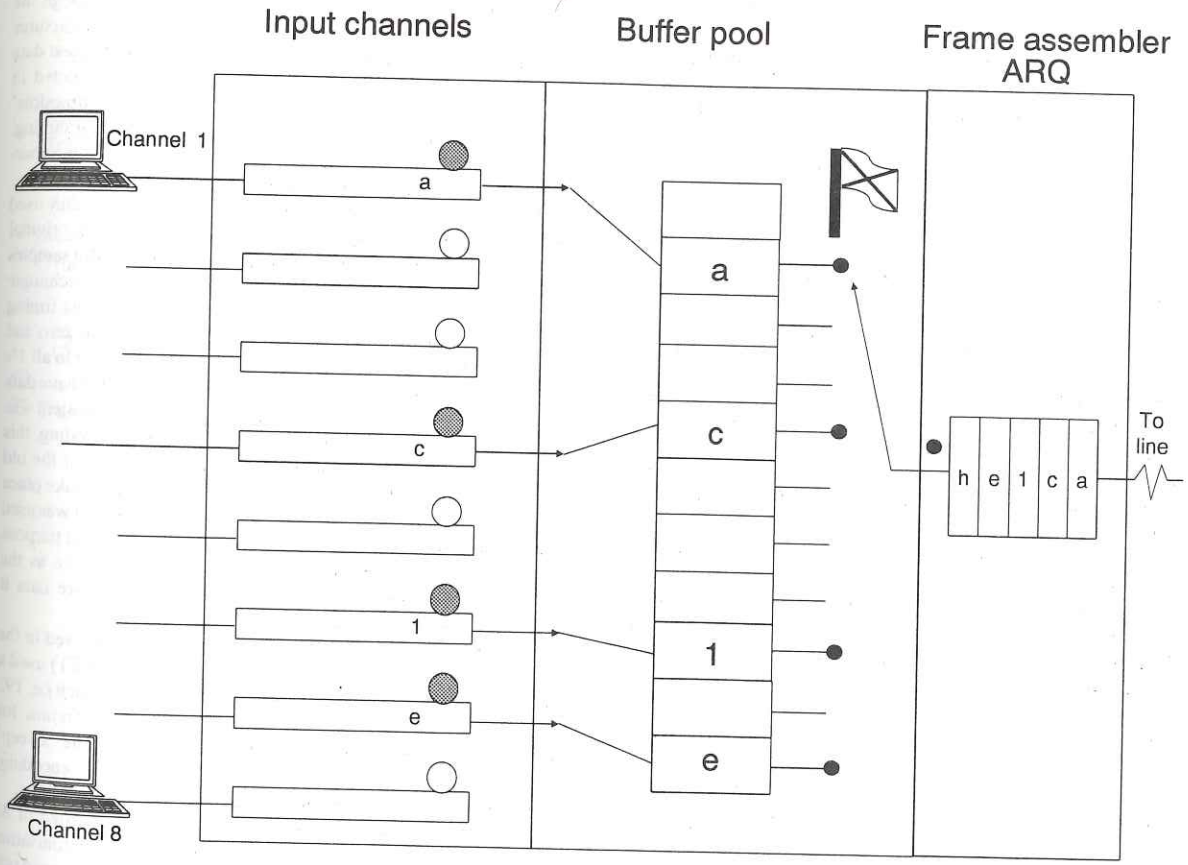

Figure 40.8 Simplified operation of a statistical multiplexer

In summary, statistical time division multiplexers offer asynchronous users the advantages of:

1. Better use of the composite line by only allocating bandwidth when sending data.
2. Techniques for encoding additional control and timing information to make the system more flexible.
3. Speed translation end to end.
4. Intelligent interaction to allow some asynchronous protocols to be emulated locally rather than over the whole link.

However some of the disadvantages are:

1. Only asynchronous data was practical. Synchronous protocols could be handled in the same manner as some asynchronous protocols in that local emulation of the host end and terminal end would result in only the actual data being transmitted over the link and the majority of the control information could be handled locally.

40.3.3 Networking

Only point to point multiplexers have been considered so far, as they illustrate the basic principles of multiplexing. However, a major development in the mid 1970s was to give a user on a terminal the facility to select more than one destination. The requirement grew out of the development in computing power, which meant that many companies were expanding on the number of computers they used and were perhaps dropping the mainframes in favour of distributed minis. To allow these to be accessible the users required the multiplexers to become networked. In effect rather than let the computer switch the user to an application it was left to the network.

40.4 High order multiplexing

40.4.1 Standards

The term, high order multiplexing, is used to describe the technique for multiplexing multiplexers. By successively taking input channels of fixed speed and creating a hierarchy of 'groups' of multiplexers, a final high speed output composite is produced, that is logically structured to contain all the input channels in a defined sequence.

High order multiplexers were originally developed as described above to allow voice circuits to be multiplexed. This simple statement is the underlying design principle for all the decisions that have been made on PTT digital speeds and frame structures throughout the world in the past 20 years. Although today they can be used for any type of digitally encoded data, including fax and video, these are the result of data communications manufacturers, and lately PTTs, 'highjacking' the circuits developed for digitised voice and using them for pure data.

The lowest input speed for high order multiplexing contained within CCITT is 64kbit/s. This is the speed arrived at from encoding voice, with a maximum frequency of 3.4kHz, at 8000 samples a second (which is above the Nyquist sampling rate) and then converting each sample into an 8 bit word or byte. The original technique used by the PTTs was to use frequency division multiplexing with 12 or 24 voice channels being frequency multiplied. For example, in the CCITT 960 channel system, channel 1 would be frequency translated to 62kHz in the first group and then to 420kHz in the second or supergroup.

At present the three principal hierarchical systems in existence, are from Europe, US and Japan.

From Table 40.1 (based on CCITT G.702) it can be seen that each system, from a base speed of 64kbit/s, multiplexes in fixed, discrete bands, up to the Gigahertz bands, suitable for transmission over microwave and satellite carriers.

Table 40.1 High order systems

Multiplex level	*Europe (kbit/s)*	*USA (kbit/s)*	*Japan (kbit/s)*
0	64	64	64
1	2048	1544	1544
2	8448	6312	7876
3	34368	44736	32064
4	139264	274176	97728
5	565148		397200

The underlying system employed is bit interleaved, although the framing structure is byte oriented. Analysis of the framing structures gives more detail on how they were used as general purpose data multiplexers. In general the input channels were constructed in frames, each consisting of 12, 24 or 30 voice channels or 'timeslots' depending on the system. This gave the first hierarchical grouping. Within this group the frames could be assembled together in superframes to contain signalling and control information.

The early US systems need to be examined separately as they used a technique that was completely alien to pure data use. The original Bell D1 system had an input rate of 56kbit/s (7 times 8000 samples per second) because the timeslots used a 'bit robbing' technique whereby the 8th bit in every slot was used for maintaining timing information on the line. By always setting the 8th bit to zero and inverting the PCM data sample so that it was normally set to all 1's i.e. no speech signal, the line signal could be guaranteed to have data transitions which kept the timing content. When the D2 system was introduced using HDB3 (high density bipolar 3) line encoding, this allowed 8 bits per slot. However, the system used part of the old standard in that it again allowed the same 'bit robbing' to take place in the same way but only every 6th frame and in this case it was used for passing control and signalling information. For speech purpose this did not impose an reduction in the quality of service as the human ear will not detect this impairment, but for pure data it introduced an unacceptable error rate.

The speeds which became standard offerings were derived in the following ways. US Bell system D2 (usually known as T1) used a frame made of 24 timeslots (or input channels) at 8 bits each i.e. 192 bits: an additional bit was added on to each of the frames for synchronisation purposes giving 193 bits per frame. The 'sweep' rate or number of frames per second was 8000 (the voice encoding, or codec, sampling rate) giving an overall speed of 1.544Mbit/s.

In the version adopted by the CCITT the frame consisted of 32 timeslots, of which 30 were for voice and two for synchronisation and signalling information, giving a total of 256 bits. At a sweep rate of 8000 frames per second this gives a speed of 2.048Mbit/s.

All digital exchanges and the space/time switches used these speeds at the core of their design.