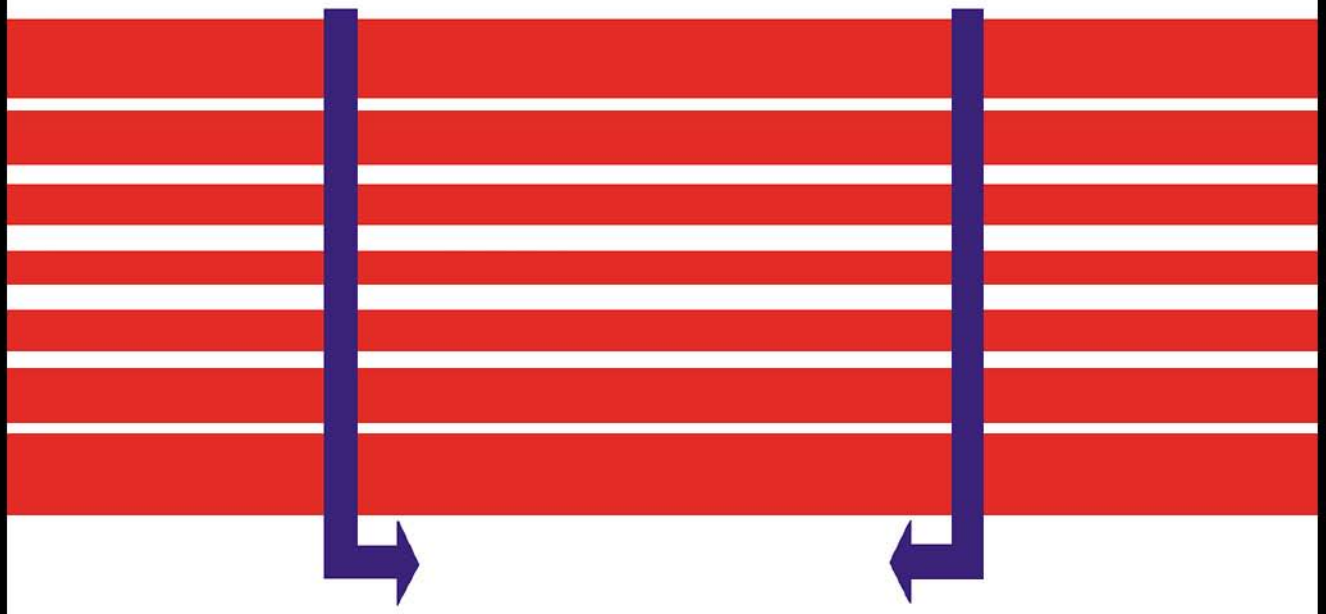
Packet Switching and X.25 Networks

Simon Poulton



Pitman

Packet Switching and X.25 Networks

Simon Poulton

Pitman

PITMAN PUBLISHING
128 Long Acre, London WC2E 9AN

A Division of Longman Group UK Limited

First published in Great Britain 1989

This edition published in the Taylor & Francis e-Library, 2003.

British Library Cataloguing in Publication Data

Poulton, Simon
Packet Switching and x.25 networks
1. Computer systems. Networks. Data
transmission. Packet switching systems
I. Title
004,6′6

ISBN 0-203-16884-4 Master e-book ISBN

ISBN 0-203-26412-6 (Adobe eReader Format)
ISBN 0-273-02986-X (Print Edition)

Acknowledgements

There are a number of people who have provided invaluable assistance in the preparation of this book, and to whom I wish to give my thanks.

Chief amongst these is Neil Matthew. Neil spent a great deal of time and effort on a detailed reading of the manuscript, and found many places where my brain had gone offline without telling the pen. His sheer hard work is much appreciated. Neil also made numerous suggestions, both stylistic and technical, which have made the book much better than it otherwise would have been.

I am very grateful to my wife Judith, who spent many hours on the word processor deciphering my handwriting and turning it into a presentable document. She also suffered my preoccupation in this project without complaint, and offered much needed support and encouragement.

Thanks are also due to Anselm Waterfield one of my colleagues, and to Ian Campbell of Exeter University, who both read through the finished work and gave me valuable reassurance.

Finally to Racal-Milgo Ltd., my employers, who loaned the equipment for the photograph, and Camtec Electronics Ltd. who gave permission for me to describe their Network Management System.

Contents

1 The packet switching network

1.1 People and computers

When computers started to become business tools and to assume an essential role in all companies, they were very large both physically and in terms of capital and revenue cost. Indeed, when erecting a new building the company would have to design features especially for the computer. A big room to house it, separate power supplies, large air-conditioning plants, and hoisting gear to lift the bits of computer off a lorry into the computer room.

Once the computer was in place then it was treated very much like the directorial suite. It had its own retinue of staff who were somehow a different breed of people to the rest of the organization, and its own set of security procedures not only to prevent access but to deter enquiries.

The computer users—the accountants and engineers—would approach the building clutching a pile of punched cards or papertape and hand it over to an operator who would take them to the inner sanctum to be fed into the machine to be processed. Some hours later the results of this labour would by placed into a pigeonhole to be collected by the user.

In the early seventies Time Sharing started to become common. We now know this as accessing the computer via a terminal, though in those days it was more typically a mechanical teletype running at 50 or 110 baud, and even quite large installations would only be able to run half a dozen of them. Nevertheless this sent a shiver of excitement throughout the user community. It was possible for a department to have its own interface to this vast resource, and this confirmed a great degree of status. For the first time people felt computing power to be in their grasp and, although perhaps not driven by need, people wanted access and fought to get it.

By the mid-to-late seventies minicomputers were available and computing power started to become decentralized. The actual use of these machines, certainly at the start, was far less significant than their potential effect. For the best part of ten years there were bloody boardroom battles with Data Processing Managers (DPMs) fighting to save their traditional status and power, and everyone else was striving to make use of the machines and facilities that were becoming available. This is not to say that the DPMs were driven by anything other thanreasonable motives. A company with departmental minis and no

centralized buying policy could easily find itself with a collection of different machines none of which were being run efficiently.

By the mid-eighties the users had won, mainly because of the technology revolution that had been going on largely in the background. The departmental mini then had as much power and access potential as the mainframe that the DPM was fighting to preserve.

As this battle raged a new issue came to the fore, that of networking. With the mainframe, as its number of terminals grew, there was an obvious and consistent network architecture, something like that in Fig. 1.1. The mainframe always has a star configuration. If terminals are added they are always a new ray from the centre whether they are connected directly, multiplexed, or via modems. The strategy is simple and easy to budget.

In the late seventies the same strategy applied to the minis in the organization. They were all treated as little mainframes and the network diagram changed to that shown in Fig. 1.2.

By the early eighties this was starting to creak a little. By this time users had expectations of technology and were starting to think for themselves about what they wanted to do. Users on their departmental minis could see advantages in being able to access the machine in another department as well. Perhaps because they had a software package to

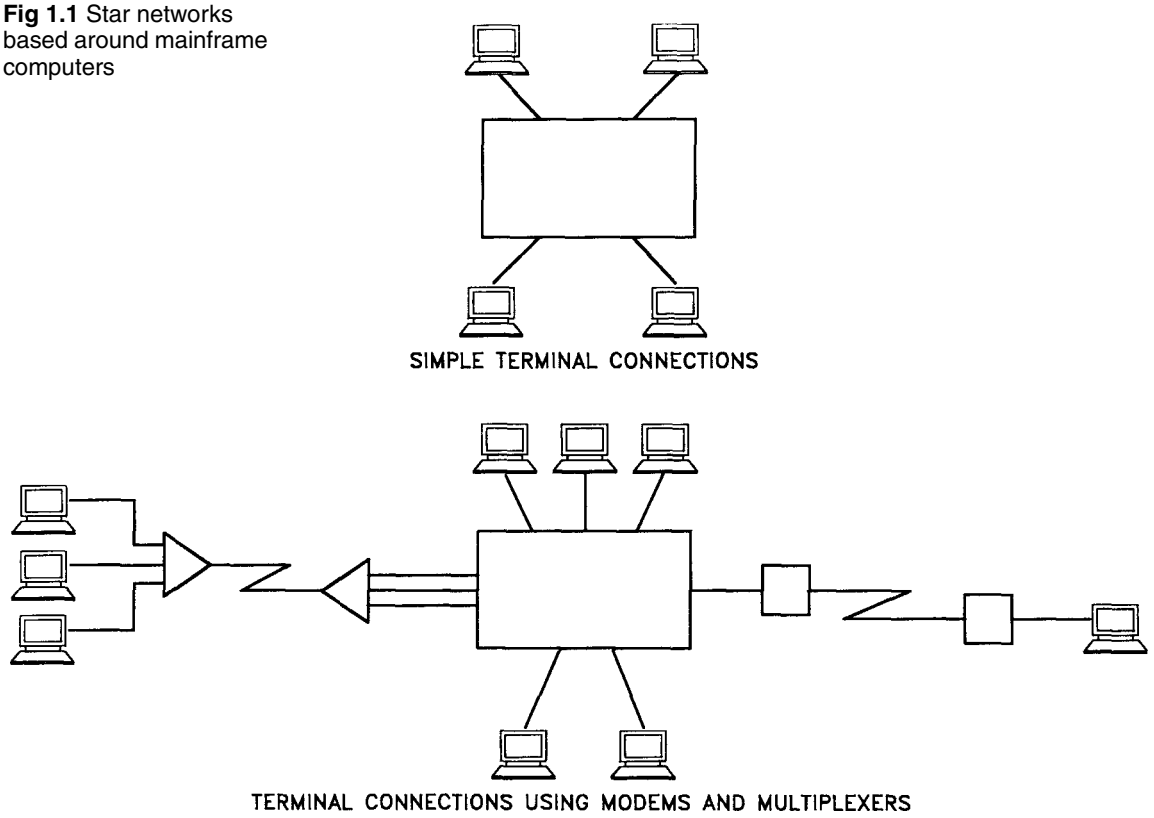


Fig 1.1 Star networks based around mainframe computers

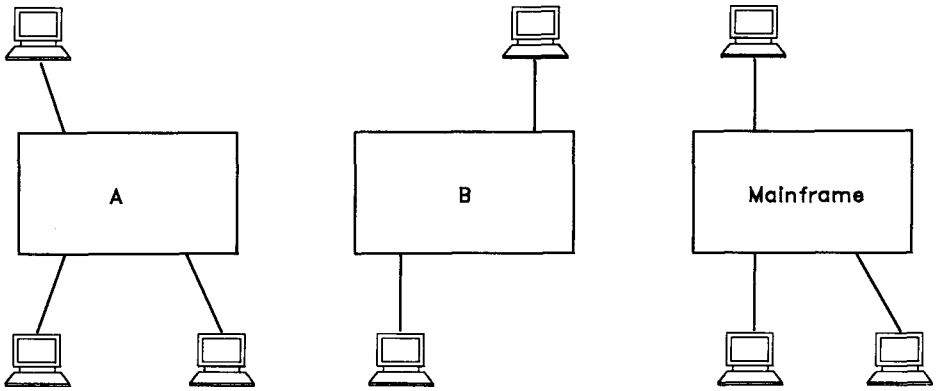


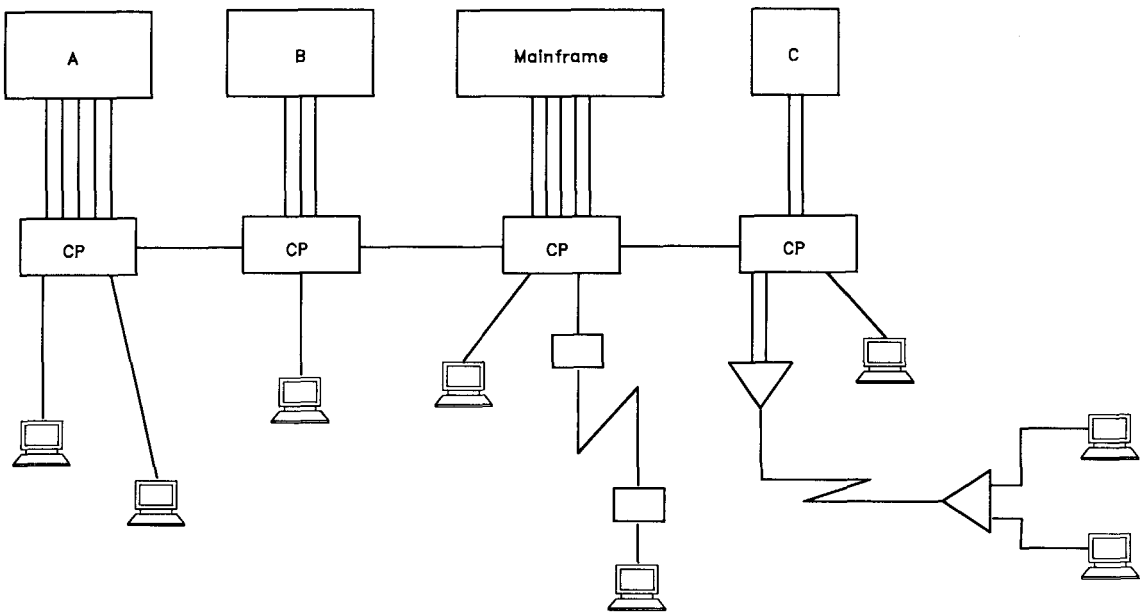
Fig. 1.2 Discrete networks of mainframe and minicomputers

which access was occasionally required, or perhaps the user sometimes had a very large task which would swamp the mini and ought to run in the mainframe.

Suppliers of modems and multiplexers in star configurations were quick to spot this need and developed equipment to address it. The network then changed to that shown in Fig. 1.3. What this equipment did was to make the users connect to a Communications Processor rather than to a computer.

When beginning a session the user now got a message from the communications processor, and had to tell the processor which machine to connect to and what precise facilities were required. The processor had knowledge of all machines and passed a message along

Fig. 1.3 Connecting the discrete networks together



CP = Communications processor

the trunk line, perhaps through intervening processors to the destination processor. This then found a line to the required computer that conformed to the facilities profile that the user requested, and made a connection. Nowadays this concept is seen as natural, but it should be appreciated that this was all happening as the mini/mainframe battle was being fought, and represented a fairly major change in thinking.

Although this is somewhat less sophisticated than the systems around today—as will be shown—it has all the hallmarks of a Network. These are:

- All services (computers) and users (terminals) connect physically to the network, not to each other.
- The user has to tell the network which service is required to be accessed, and which facilities such as speed of access, amount of memory, and peripherals are required.
- The network determines whether the request can be met and, if so, allocates the service to the user.
- When the user has finished, the network has to reset everything that has been done back to its initial state.
- Both user and service are users of the network.

By the early eighties attention was being focussed on the network, as it was an area with high potential growth. It would also require international standardization to ensure that equipment from all manufacturers could be made part of a network.

Many types of architecture were developed to implement a network and many schemes of carrying data along wires were developed. One such architecture and scheme is Packet Switching which is the subject of this book. The book will explain and illustrate the subject with particular reference to X.25 networks.

1.2 Networking solutions

Before describing packet switching and networks that conform to international standards, it is instructive to take a brief look at other—older—technologies that have been used to connect users to the services that they want to access.

The use of multiplexers has already been mentioned as a tool that was in use right from the start of remote access to computers. The multiplexer is a straightforward data funnel that has several access points on one side, and an aggregate or trunk port on the other. The multiplexer then shares the transmission capability of the trunk between the terminals or host ports connected to it. The advantage of this scheme, as shown in Fig. 1.1, is that there is only a single communications link between the two multiplexers, and therefore there is only a single line rental to pay and a single pair of modems to buy. It is thuscost-effective to link say a

branch office to a corporate headquarters, because several co-located users can all access the computer using a single link.

Multiplexers work on three basic principles: Frequency Division Multiplexing (FDM); Time Division Multiplexing (TDM); and Statistical Multiplexing (Stat. Mux.).

A frequency division multiplexer works by splitting the bandwidth of the cable into several narrow frequency ranges, and allocating a range to each communications channel. It is therefore very like the splitting of the airwaves into several radio channels. The technique is costly and is not suited to the limited bandwidth of telephone circuits, and is therefore very rarely used in data communications.

In time division multiplexing the device scans each of the channels in turn, and passes the selected data onto the aggregate. It is like a railway turntable which constantly takes a wagon from each of the branch rails and passes it to the main line. This can only work if the aggregate can handle the combined data of the channels; otherwise the multiplexer would have to stop the flow of data on the channels.

Most TDMs allow the aggregate bandwidth to be split unequally, so a 9600 bps link could support say two 2400 bps terminals and four 1200 bps printers. The two TDMs attached to a link must be configured so that each has the same understanding of how the time on the aggregate is allocated. This allows the individual channels to be derived correctly at the far end.

Statistical multiplexers work by having a buffer for each of the channels, and they allocate the aggregate according to how full the buffers are. Thus each terminal or host port "talks" to its own buffer, and does not know that the data is not being transmitted at that instant. The buffers are emptied into the aggregate at a speed relative to how full they are. It is like the postal service emptying letter boxes; each box is emptied regularly, but those that have a lot of letters are emptied more frequently.

The statistical multiplexer allows the sum of the individual channel speeds to exceed the aggregate link speed, because, when the multiplexer is not servicing a particular channel, the channel data is inserted into its buffer. To avoid the buffer filling, and the consequent necessity of the multiplexer stopping the channel, the average data rate of the channels should not exceed the data rate of the aggregate. This means that if a terminal runs at a speed of 9600 bps, but is only sending data one minute out of ten, then its effective data rate is 960 bps. Five such terminals could be statistically multiplexed down a 4800 bps aggregate.

Since there is no way of knowing which channel may be using the aggregate at any instant, it is necessary for the sending stat. mux. to precede each transmission with an indication of which channel the data is intended for. This allows the receiving stat. mux. to direct the data to the appropriate destination port. The indication forms an address which informs the destination how to route the data.

So far, all the devices mentioned connect two sites together, and data input to a given port on one site is always output from a known port at the other site. In the case of statistical multiplexers which must exchange addressing information anyway, these limitations can be removed.

If we consider Fig. 1.3, then the communications processors can be statistical multiplexers. The addressing information preceding data on the aggregates simply has to be extended to show the destination stat. mux. as well as the port. The intervening stat. muxes. can examine the destination field and route the data down the appropriate aggregate if it is not for them.

To allow the user to choose the destination, rather than to have it pre-configured by the Network Manager, it is only necessary for there to be a dialogue between the user and the stat. mux. The user simply indicates that all data should be preceded by the address for port X on stat. mux. Y, and the network of processors then routes all data accordingly.

Once this expansion had been made, then stat. mux. manufacturers quickly enhanced their machines to include a range of useful features:

- Destination Names so that the user can connect to FINANCE DATABASE and the processor looks up the actual network address in its configuration data.
- Hunt Groups so that, rather than trying to connect to a particular host port, the destination processor will simply allocate the next free port on the host.
- Mesh Networks so that there may be multiple aggregate links allowing the data load to be balanced, and offering resilience to faults.

These are in addition to Port Contention which is automatically provided by the statistical multiplexing concept. This provides the ability for there to be twenty users of ten host ports, although only ten can be working at once.

The major drawback of the statistical multiplexer is that different manufacturers have different techniques of performing it, so users are locked into a particular system. This also means that there can be no aggregate link into the host, only a multiplicity of low-speed connections.

1.3 Packet switching

In a Packet Switching Network, as the name implies, data is communicated between the user and the service in the form of packets. Before we can look at this type of network we therefore need to know that:

> **A packet** is a structure containing some data to be communicated, an indication of where the data is going to (destination address), and an indication of where the data has come from (source address).

Having said that, it is clear that the function of the components in

the network is to receive the packet, look at the destination address, and from its knowledge of the network send the packet to the next appropriate component. See Fig. 1.4.

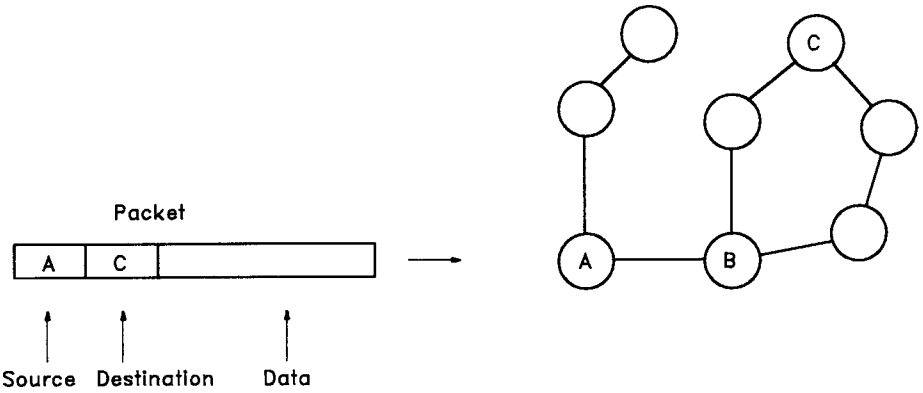


Fig. 1.4 A packet switching network

The packet has destination address C. Component A therefore must route the packet to component B. Component B has a choice; either of the onward routes is acceptable. The upper route looks preferable because there are fewer components; therefore the packet will suffer less delay. However, if the upper route is already being heavily used, or if it has a noisy, error-prone component, then the lower route is preferable. Whichever is chosen and for whatever reason the packet will arrive at component C as requested. When it does arrive, the receiver can extract the data and can determine the sender.

It is important to note that component C is a part of the network. It may not be a user or a service. For example if the packet is being routed to a screen, then only the data should appear, not the addresses. The network must deliver information appropriate to the end point, which is why both the service and the user of it must be considered users of the network.

When a user obtains access via the network to an accounting package in a computer, then neither the user nor the accounting package is concerned with addresses or the routing of packets. All of these are network functions, and both the user and the computer running the package are users of the network.

1.4 The layered network model

The above explanation of packet switching is simplistic because it does not address important issues such as the following:

- What is the format of the packet? How long are the fields in the packet and how are they encoded—Binary, ASCII, EBCDIC, etc?

- What is the style of addresses-numeric, alphabetic and so on. How are addresses assigned in the network, and how do you ensure there are no duplicates?
- What happens if the packet is corrupted in transit?
- What happens if the packet is completely lost in transit?
- What happens if there is more than one conversation going on?

Even if these issues are resolved in a particular network then how are the issues resolved in equipment from different manufacturers and how is connection between dissimilar networks achieved?

Fortunately, international standards bodies, such as the International Consultative Committee for Telephony and Telegraphy (CCITT), the Institute of Electrical and Electronic Engineers (IEEE), and the International Organization for Standardization (ISO) have addressed these issues and produced recommendations and standards. ISO have developed a model of a communication network which divides the problems into seven groups, and this is referred to as the Reference Model for Open Systems Interconnection. This is defined in the ISO 7498 standard. More concisely it is referred to as the ISO OSI Seven Layer Model.

The model is discussed in more detail in Chapter 6; however, for the moment we need only be concerned with the bottom three layers:

- *Layer 1*
  Covers the problems of getting a data bit from one component, via a transmission medium, to another component.
- *Layer 2*
  Covers the problems of ensuring that the bits are transferred with error notification.
- *Layer 3*
  Covers the problems of synchronizing the two users of the network.

The model only relates the functions of the layers and does not attempt to solve the problems. A network must therefore choose from a range of standards which are practical implementations of the narrow layers. Since the model defines what tasks are performed in each layer, any choice of standards is satisfactory as long as each standard conforms to the service required from the layer of the model.

In order that equipment from various manufacturers can interwork, and so that different networks can successfully be connected together, it is necessary that everyone chooses the same standards. There are therefore a limited number of network technologies, the most popular being X.25 and Ethernet. The international standards for these networks that conform to the OSI model will be discussed later.

In order to illustrate the functions of the layers we will examine X.25. Before doing so, a simple analogy. Consider a normal postal letter being carried in a postal van along the road between two postal depots. The

letter is part of an individual conversation between the sender and the recipient, and we can imagine it as part of an ongoing series.

As part of the delivery process the letter has to be carried between various sorting offices. Each pair of offices will have individual control of the traffic flowing between them but will only be interested in the information written on the envelope, not the contents. The van on the road is only interested in the fact that it is going from A to B; it is not interested in the individual letters or their envelopes. In fact any carrier could be used equally effectively though the delivery times may be different.

This hierarchical arrangement—where each layer offers a service to the next—is similar to the way in which X.25 works.

1.4.1 Layer one

Layer one simply states how data bits are transferred. This could be implemented by emitting a squeak to indicate 0 and a squawk to indicate 1. Such a scheme would be less than satisfactory in most environments and the normal method is to send an electrical signal. Chapter 7 covers details of what happens, but essentially there are two signal paths between each two components. One signal path is for data in one direction, and the other path is for data in the other direction. The voltage level determines whether a 0 or 1 is indicated. Having two signal paths allows data to be sent in both directions simultaneously.

X.21 is the electrical system used and is a synchronous method of transmission. This is examined in Chapter 7, but for the moment this only needs to have one consequence to us which is that there is a continual "beat" called the *clock* and it is compulsory to send a bit at every beat.

1.4.2 Layer two

X.25 layer two defines methods for detecting and correcting errors in the transmission of bits from one component to another. In fact this is not precisely true as will be shown in Chapter 6, but it is nearly always the case.

At this point we have to look at a network with more than one conversation taking place (see Fig. 1.5). If there is a conversation between A and P, and one between B and Q, then it is clear that on the wire between M and N the two conversations have to be multiplexed. The multiplexing is nothing to do with layer two. Layer two is only concerned with the correct transmission of data along a wire; it is only concerned with the carrying of data between M and N. The combination of layer one and layer two provides a link between the nodes of the network, and layer two is often referred to as the *data link layer*.

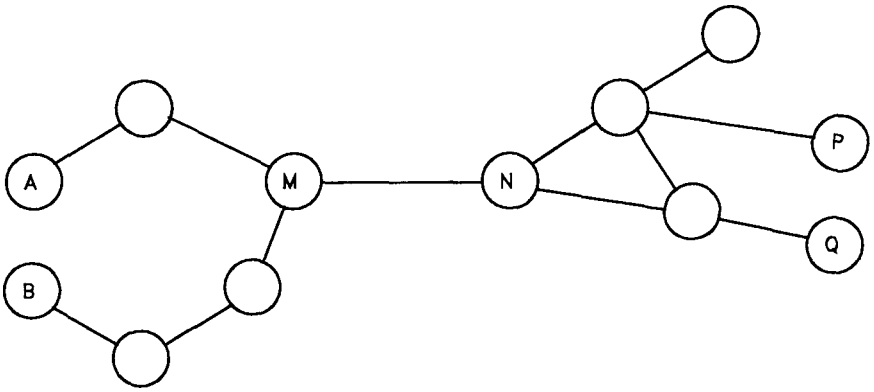


Fig. 1.5 Multiple conversations in a single network; conversations between A and P, and B and Q, share the M to N link

Layer two sends the data in a defined structure called the *frame*.

- The frame is the unit of transmission for layer two.
- The frame is contiguous—that is, once the sender starts to send a frame it will carry on to the end or give up completely and try again.

There is a further feature imposed by most hardware implementations that is not actually imposed by the standard:

- The frame is organised in bytes and always contains a whole number of bytes.

Since layer one requires that a bit is sent on every beat of the clock, layer two requires something to send between frames. This is the *flag byte* which is a set sequence of bits which the receiver must essentially ignore. Remembering that layer one is full duplex we can look at our example in more detail. See Fig. 1.6.

Notice that there must always be a flag byte to delimit one frame from

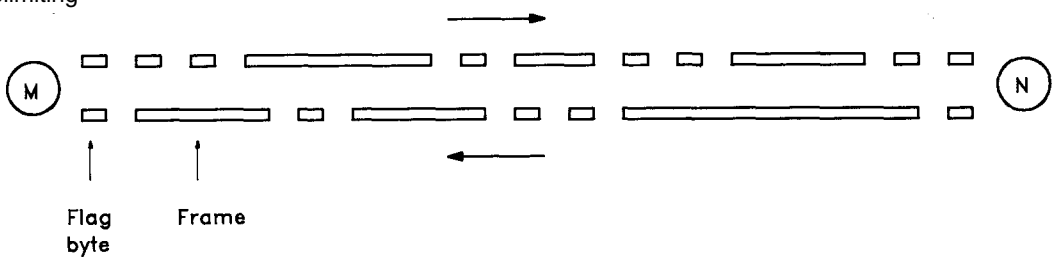


Fig. 1.6 Flag bytes and frame delimiting

the next. It acts as a delimiter as well as an idle condition. Notice also that the frames are of arbitrary length. The layout of the frame is fixed and is as shown in Fig. 1.7.

The first byte of the frame is the *Address* byte, and simply indicates whether the frame is a command or response. This is explained in more detail in Chapter 4.

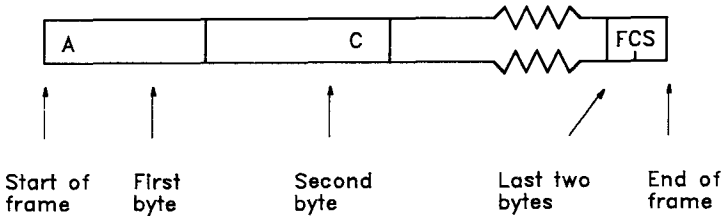


Fig. 1.7 Format of the layer two frame

The final two bytes are called the *Frame Check Sequence* (FCS) and are a value related to the contents of the frame. The field is like a checksum, but is actually a Cyclic Redundancy Check. Using the FCS the receiver of the frame can determine whether or not the frame has been corrupted during transmission.

Since the frame is of variable length with no length indication, the receiver must capture all of the data up to the terminating flag byte, before it can determine the length of the frame and where the FCS is located. It then calculates the FCS itself, and checks whether this agrees with the FCS in the frame to determine whether corruption has occurred.

The FCS does not catch all errors. Being 16 bits long it has 64000 possible values. Thus if corruption occurs during transmission there is some chance that it will not be detected if the FCS happens to be correct.

The second byte of the frame is the *Control* byte and indicates what type of frame this is. The control byte therefore indicates the format of the frame, different frames having different formats. One of the frame types is the *Info* frame and is used to carry data from one node to another. The layout of the Info frame is shown in Fig. 1.8.

Fig. 1.8 Conceptual layout of the Info frame

The third and fourth fields of the Info frame are the Send and Receive sequence numbers which are explained below. They are followed by the data. Note that there is no length indication. The length is determined by the position of the terminating flag byte. The sequence numbers are actually included in the Control byte, but in the early part of this book they will be shown separately for clarity.

The *Send Sequence Number,* depicted N(S), indicates which number this is in a series of Info frames. The first Info frame sent will have an N (S) of 0, the next will have an N(S) of 1, and so on. This mechanism allows the receiver to verify that it has received all of the frames and that none have been lost in transit. The N(S) can rise to a value of seven and will then cycle back to zero. This limits the length of the N(S) field to three bits in the frame. There is a problem here in that if the N(S) can cycle, and if eight frames are lost in transit and the next frame is sent correctly, then the receiver cannot detect the loss. To prevent this, the

receiver has to acknowledge receipt of the Info frames, and the sender cannot send any new frames if seven are outstanding and awaiting acknowledgement.

The acknowledgement is carried in the N(R) field of returning Info frames, and indicates the next frame that is expected.

Suppose a frame is sent with N(R) of 3 and N(S) of 7:

- The Receiver of the frame must check whether frame seven is the next one that it was expecting.
- The Receiver may now use an N(S) of 3 again.

It is not necessary for every frame to be individually acknowledged. Fig. 1.9 shows an example conversation where only some of the frames are acknowledged, but the acknowledgement includes earlier frames as well.

Suppose that a user has requested that a file be listed out on screen and that the conversation is over a packet switched network. During the

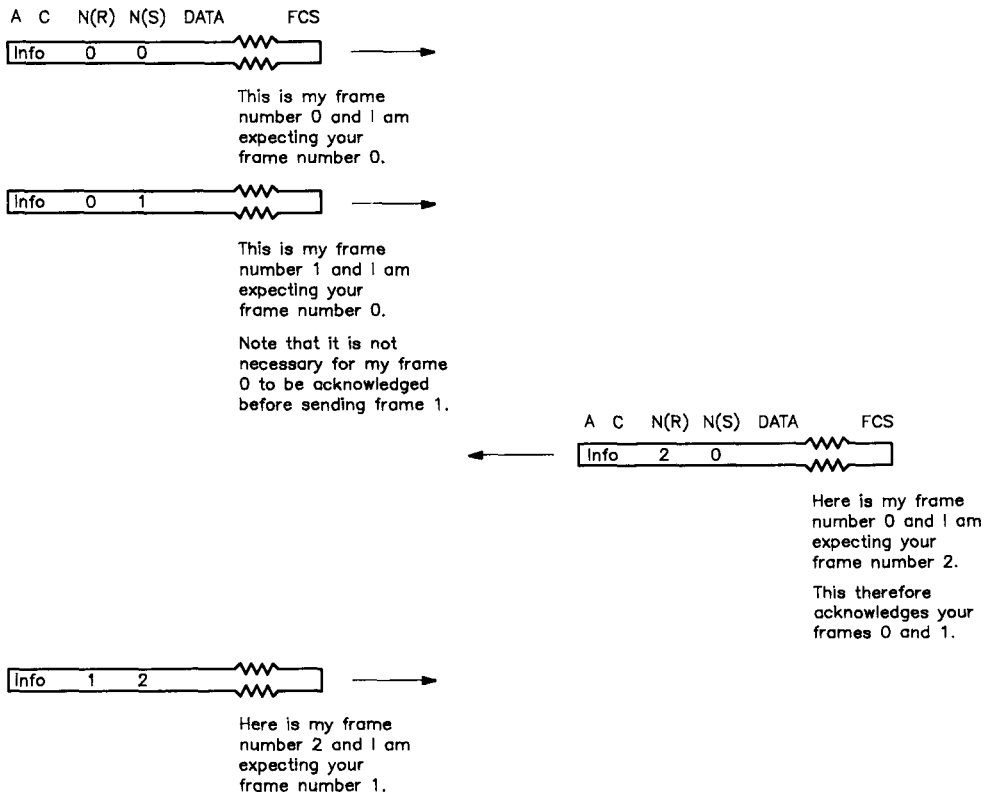


Fig. 1.9 A layer two conversation; note that it is not necessary for every frame to be individually acknowledged

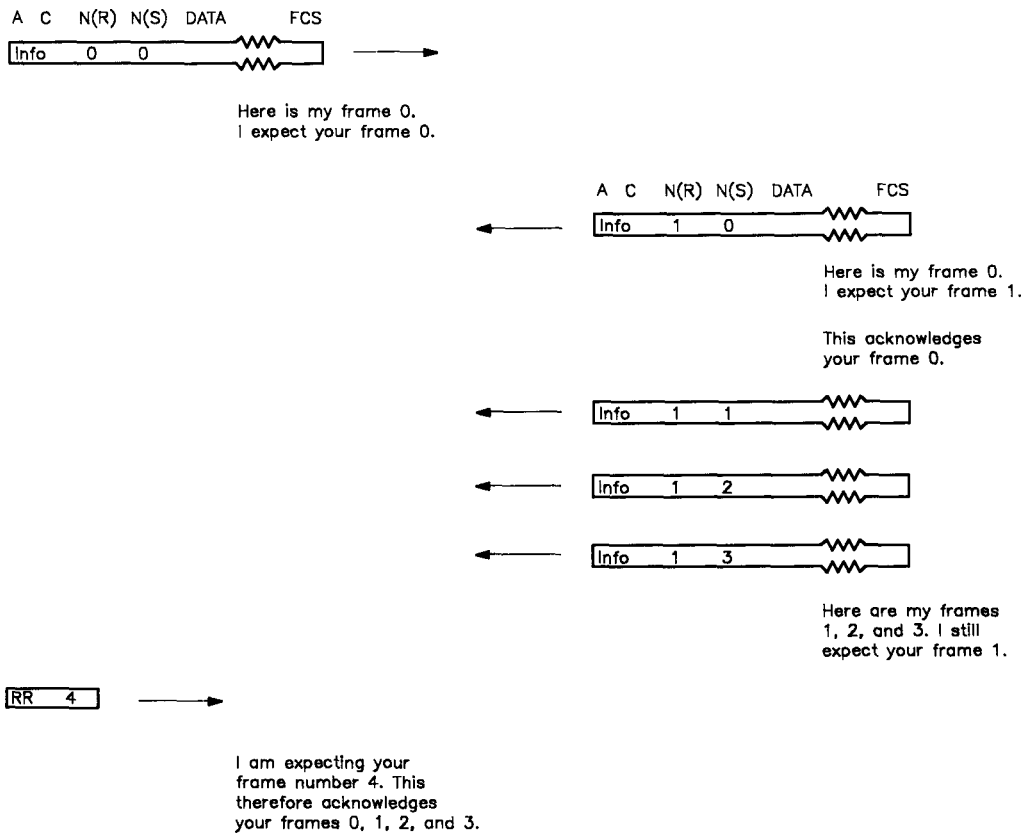
listing the data will be essentially unidirectional and the receiver in the network needs to be able to acknowledge the receipt of Info frames without sending Info frames back. This is achieved with the *Receiver Ready* (RR) frame which just sends the N(R) value to the sender. Fig. 1.10 shows an example of this.

Unless the data is unidirectional, or unidirectional for a period, then there is no need for the use of the RR frame. However, most X.25 implementations require that frames are acknowledged within a short time of being sent, and the RR frame is very common.

The frame numbering scheme gets a little more complex when the numbers wrap around. The same logic still applies though. See Fig. 1.11.

We now need to look at what happens if there is a loss of data on the link. Consider the conversation shown in Fig. 1.12. Clearly something has gone awry. The most likely cause is that Info frame 4 from the lefthand side has been corrupted in transit, thus when it arrives at the

Fig. 1.10 Use of the RR frame to acknowledge receipt of data without sending more data in return



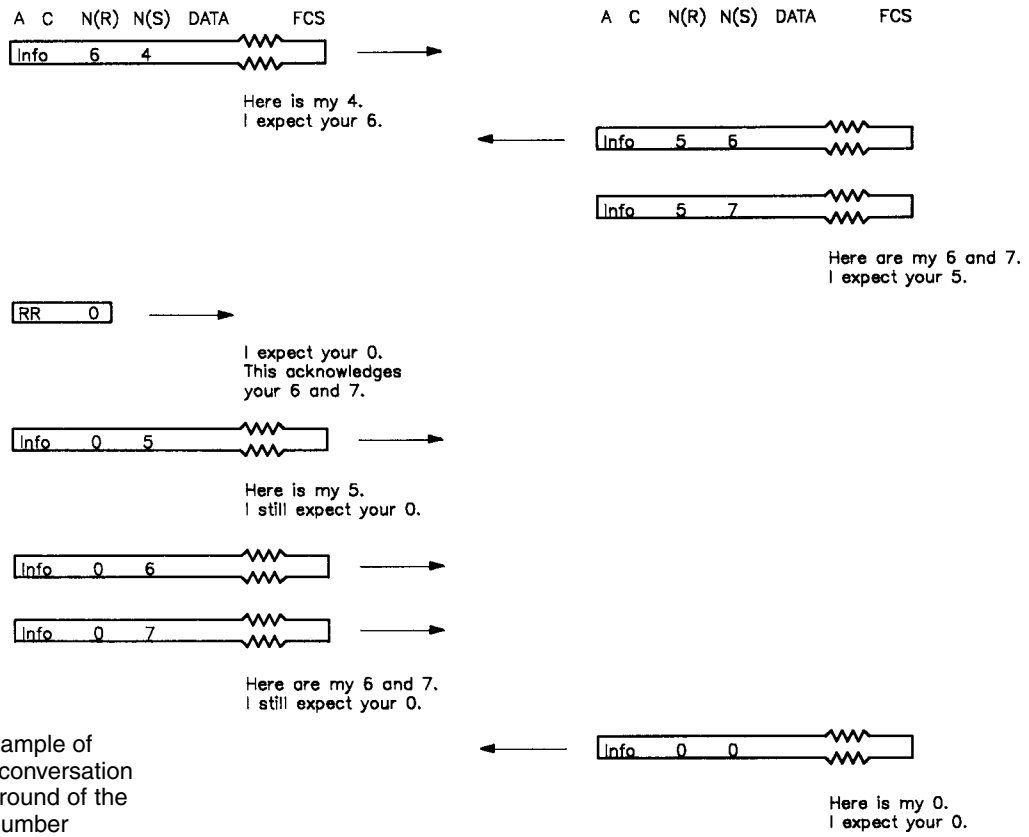
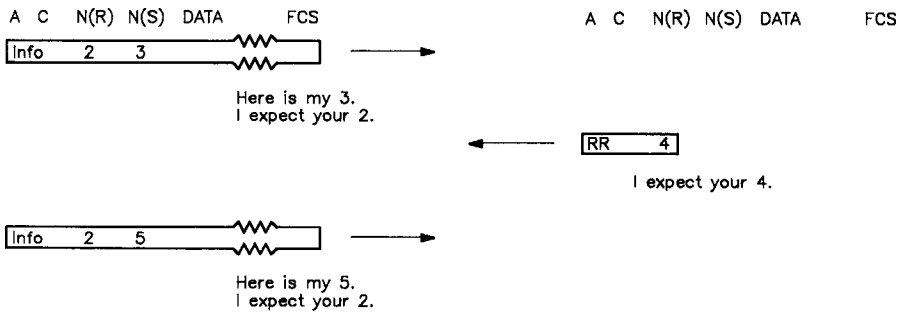


Fig 1.11 Example of a layer two conversation with wrap-around of the sequence number

receiver the FCS is incorrect and the whole frame is discarded without further attention. This is normally done by the hardware

What happens in these circumstances is that the righthand side will issue a *Reject* frame. This is done on receipt of an out-of-sequence frame—in this case number five—and will indicate where retransmission is to start. In this case we require frame 4 to be retransmitted. However, all subsequent frames will also be retransmitted as the receiver will ignore frames following the one that was out-of-sequence. The conversation would therefore continue as shown in Fig. 1.13.

Fig. 1.12 Loss of frames in transit; note that it is unusual to lose the whole frame; normally it is corrupted, thus has a bad FCS and is therefore ignored by the hardware and not passed to layer two



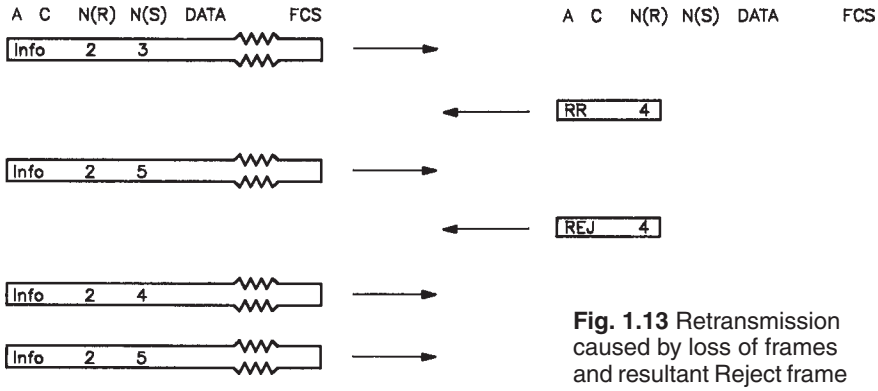


Fig. 1.13 Retransmission caused by loss of frames and resultant Reject frame

Consider the conversation in Fig. 1.14. If we were now to wait for the time required by the network, then the righthand side would acknowledge receipt of all the data by sending an RR with N (R) of 4. Suppose that the timeout has not expired and the lefthand side has more data to send, what will happen? The lefthand side cannot send frame 4 because its frame 5 has not been acknowledged. If it were to send frame 4 and receive an RR with N(R) of 5 then either:

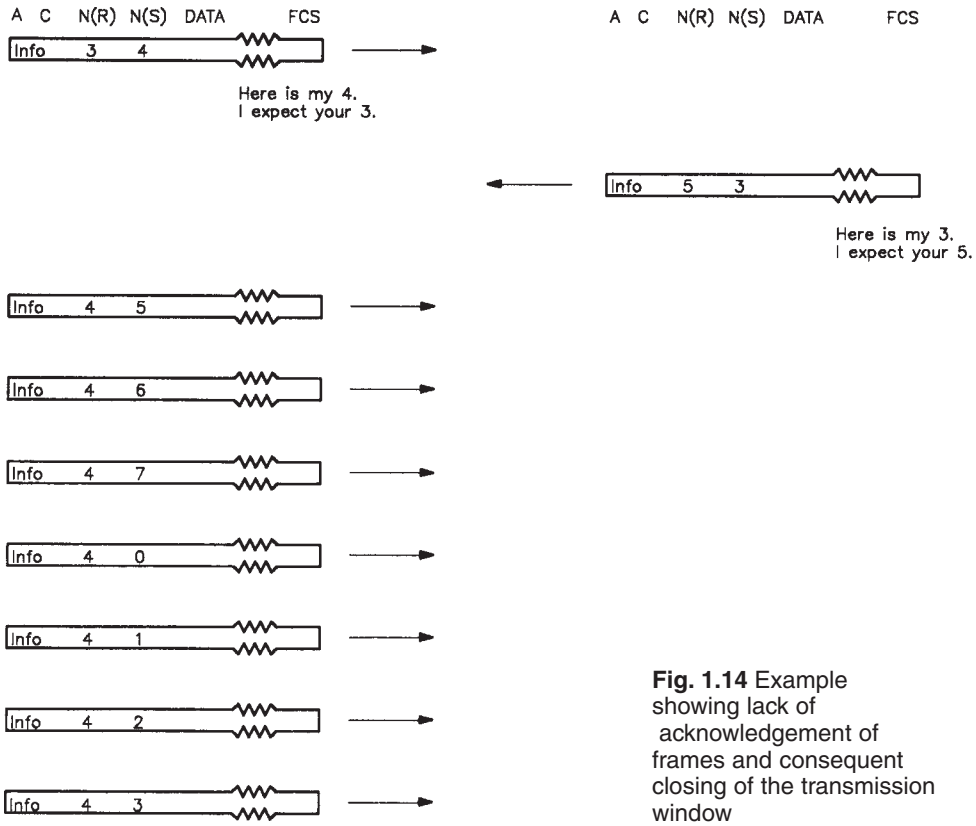


Fig. 1.14 Example showing lack of acknowledgement of frames and consequent closing of the transmission window