EDINBURGH LOCAL AREA NETWORK

W.P. Enos, I.B. Hansen, R.W. Thonnes

Abstract

This document describes the design of the Edinburgh University Computer Science Department's local area network. The network is based upon a carrier sense, multi-access broadcast discipline with collision detection and was first developed within the Xerox PARC research establishment.

The document first surveys the range of factors which were important in determining the nature and detail of the eventual design. The basic principles of local networking and ethernets are introduced. Major differences between the Department's ethernet and the Xerox model are summarised. Because EU ethernet controllers have their own processing and memory resources they can offer a range of network services and absorb the overheads of protocol handling that would otherwise fall upon the network's hosts. It follows that the scope of protocol design is more extensive, incorporating higher layers of protocol than the existing Xerox proposals. The second part of the document summarises the design decisions that were taken, presented in the form of a guide to network services and the means whereby a host may access them. This constitutes a complete set of the functional requirements which directed the successive stages of implementation.

The network controller hardware, departmental link and parallel Interdata minicomputer interfaces were designed and built by I. Hansen. The Unibus interface to Departmental VAX was designed and built by R.W. Thonnes. He is also the designer of an earlier version of the network.
The survey on the users view of the network was done by W.P. Enos. He is also the author of one of the first versions of the protocols and station's controller software. His MSc thesis was the first iteration of this document.
The final version of controller's software was written by R.W. Thonnes.
H. Dewar provided most of the external software support (filestore and ISYS software) and together with P. McLellan provided extremely useful and constructive criticism.

Contents

page

port - specific characters: CHP.n

1n	RDY	Ready The station is ready to accept a data frame following OPN or DTX
2n	STX	Start of data frame {+ n data bytes + ETX}
3n	DTX	Transmit Data Packet Evokes RDY if port is open, ERR if closed
4n	SOT	Start of protocol-free Transfer Echoes SOT if all is well
5n	<NAK>	Negative Acknowledge (not accepted, only generated)
6n	---	Unassigned
7n	---	Unassigned
8n	OPN	Open port Evokes RDY. (Has side-effect of closing port first, not allowed on port 0)
9n	CLS	Close port (If used on port 0, port is then re-opened)
An	<ERR>	Protocol Rule Violation Error (only generated)
Bn	---	Unassigned
Cn	ACK	Acknowledge Accepted in place of RDY to discard a packet Generated when packet was received by destination station or when a broadcast packet has been sent.
Dn	---	Unassigned
En	---	Unassigned
Fn	---	Unassigned

Introduction

This document describes the local area network, designed and built at the Department of Computer Science of the University of Edinburgh.
The network is based on the principle of Carrier Sense Multiple Access with Collision Detection (CSMA/CD), known under the more popular name of ethernet.

Section 1 of this introduction briefly presents some key terms and classifications underlying the subject of local area networks.
In section 2 the concept of ethernets, and the leading work done in this area at Xerox's PARC is summarised.
Section 3 contrasts Edinburgh's network model to Xerox's proposed standard and argues that it is significantly different: because Edinburgh's controllers are 'intelligent' their capability is greater and the task of protocol design is, in turn, more extensive, incorporating higher-level protocols than the Xerox proposals treat.

Chapter I describes the parameters of the network like frequency, data encoding and packet format.

Chapter II presents the architecture of the station and provides functional description of its separate components.

Chapter III presents a guide to services on the ethernet and how users can access them. It also presents the results of a survey about the department's views of the future network conducted when the design was being developed. Most of the design decisions were based on the results of the above survey. Taken together with Xerox's low-level protocol for driving the ethernet channel, chapter III is a complete functional description of the Edinburgh controller.

1. Local Area Networks

Local area networks (LANs) lie midway along a spectrum of implementations ranging from closely-coupled multiprocessor systems at one end to long-haul packet-switching networks (e.g. the ARPANET and Britain's EPSS) at the other.

The distinguishing features of LANs are:
- locality: they span distances from tens of meters to kilometers. Thus signals can be transmitted with very low propagation delay, and, for example, carrier sense becomes feasible as a means of controlling access to the transmission medium.
- high data-rate capability: rates up to at least of 10 Mbaud can be supported. Thus bandwidth is a relatively cheap resource. As with long-haul networks, the data flow is typically "bursty" (high ratio of peak-to-average data rates) and packet switching is a more appropriate technology than circuit switching.
- inexpensive transmission media and hardware components: the costs of setting up an LAN are modest, and the site is usually owned and run by a single establishment without intervention or regulation by a common carrier.

The main benefits that derive from LANs are:
- low-cost and flexible means of connecting and supporting communications between heterogeneous machines, or hosts, within the establishment.
- simpler protocols and lower control overheads on data transfers, due to the significantly higher bandwidth, lower signal delay, lower error

rates and less complex functions compared to long-haul networks.
- increased scope for resource sharing around the site and distribution of function among loosely-coupled hosts. They can be configured into more sophisticated and reliable systems, supporting an enhanced set of applications.

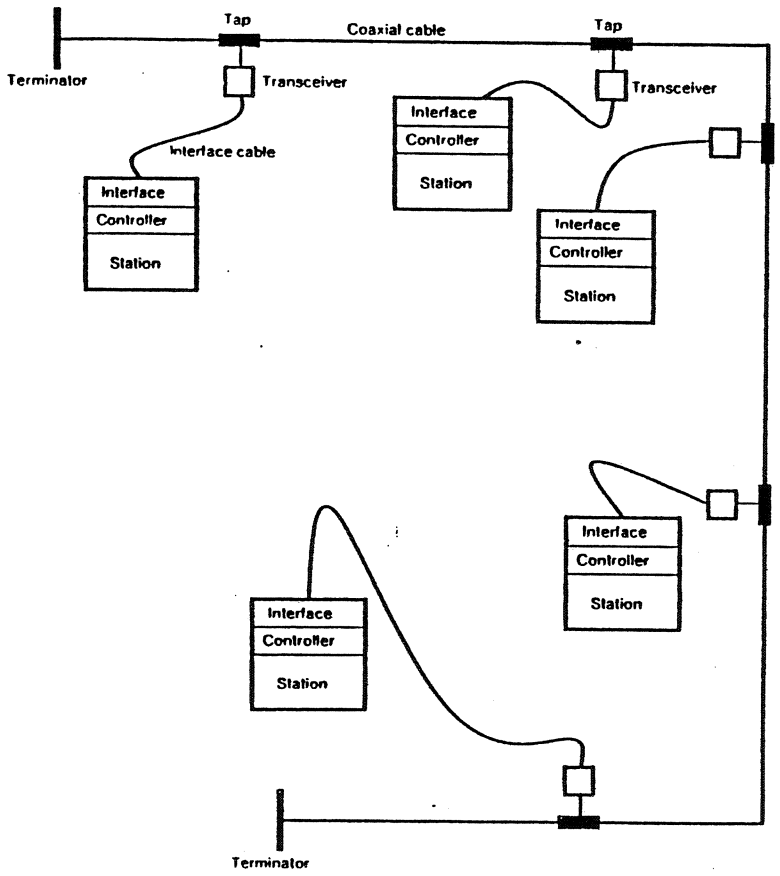
The emergence of LANs has coincided with and facilitated the growth of a new model of functionally distributed computing systems [Wilkes & Needham, 1980]. It was noted in one of the earliest survey papers [Clark, et al., 1978] that LANs have brought an important shift in the balance of comparative costs between the network-related and host-related parts of the total costs of attachment of a new host to a network. Communications costs are dominant in a long-haul network: e.g. the costs of maintaining a public satellite circuit or a private microwave link. This justifies the use of fairly expensive minicomputers as packet switches to handle the elaborate protocols and the complex algorithms for flow control and routing decisions- all of which make heavy demands on host resources in order to economise on the comparatively scarce resource, bandwidth. LANs, however, have reversed these traditional cost relationships. With wide bandwidth and simple protocols there is no longer a case for interposing hefty front-end processors or terminal-interface processors (TIPs on the ARPANET) between the network and hosts. It becomes economic to attach lightweight, microprocessor-based hosts. With their flexibility and their better price-performance ratio than mainframes, it is attractive to dedicate them to supporting particular user services and use a LAN to unify them into a functionally distributed system.

There are three major dimensions by which local network architectures have been classified and compared [Shoch, 1979] [Clark, et al., 1978]:
- the communications medium: e.g. radio, twisted pairs, fibre optics
- the connection topology: e.g. star, bus, ring, loop, or partially-connected store-and forward systems
- the control procedures governing channel access: e.g. token passing, centralised polling, contention, register insertion

One of the increasingly popular classes of LANs is the packet-switched, multi-access bus structure with decentralized control. The ethernet developed at Xerox's PARC (Palo Alto Research Center) is the best known implementation of this class [Metcalfe & Boggs, 1976] [Shoch, 1979].

2. The Ethernet

In terms of the dimensions listed above, the ethernet is based on a broadcast medium (a co-axial cable) of limited length (up to 2.5 km at 10 MHz) which guarantees a fully-connected topology (every station will be heard by every other). The ethernet's control discipline comes under the label CSMA/CD: Carrier Sense Multiple Access with Collision Detection. CSMA is a channel access discipline used in other implementations of the multi-access bus architecture. The ethernet, though, has the added refinements of collision detection, collision enforcement (the jam) and contention resolution (deference), as well as automatic adjustment to network loading (the back-off algorithm) by deferring re-transmission according to increasing multiples of the collision interval (slot time). Because these procedures are fully distributed among the stations, network control is not vulnerable to the failure of any one of them. And because the topology is simple and the shared components are kept to a minimum and are passive, the susceptibility to hardware failure is low, as is signal



An overview of an Ethernet communications network (after [Metcalfe & Boggs, 1976]).

delay and the cost of adding new stations. The network can be easily "grown", tapping the cable (the ether) where and when ... red without interrupting normal activity.

Among the main design goals that have motivated the development of the ethernet at Xerox [Crane & Taft, 1980], [Digital, et al., 1980] are the following:

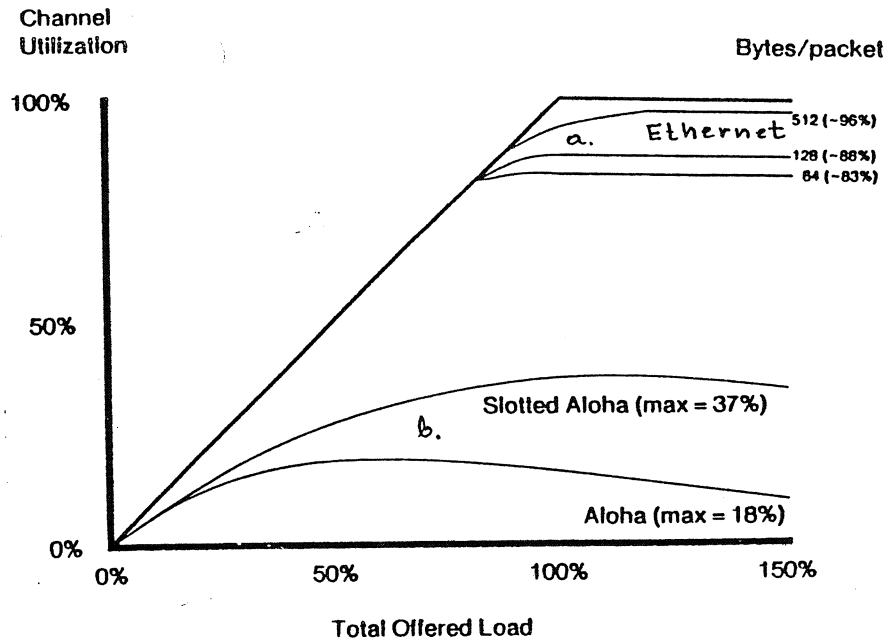
- stability: the network should remain stable under all load conditions, i.e. delivered traffic should be a monotonically non-decreasing function of total offered traffic
- maintainability: installation and maintenance of ethernets should require little effort
- availability: the network's resistance to catastrophic, system-wide failures should be high
- fairness: all stations should have equal access to the network when averaged over time

There is impressive evidence that the ethernet performs very well when measured against these goals. Xerox's interconnected set of ethernet installations probably represents the largest collection of LANs currently in use. The experience of one typical installation, comprising over 120 hosts (mainly single-user, stand-alone Altos, with a few time-shared mainframes, dedicated servers, and gateway machines), has been studied in detail. Statistics were collected passively by a monitoring station when the system was functioning under normal conditions and, later, when conditions of heavy loading were artificially generated [Shoch & Hupp, 1980].

The measurements indicate the following performance characteristics under normal loading:

- very good reliability: system-wide failures have only occured a few times over several years, and were then due to either carelessness or to direct hits by lightning bolts
- low utilisation of available bandwidth (averaging 0.9% over a 24-hour period) with a relatively high variance over shorter intervals (as high as 17% utilisation over the busiest minute and 37% over the busiest second). Total traffic volumes over a 24-hour period averaged 2.2 million packets (300 million bytes) which amounts to about one-half the Arpanet's daily average. The high variance in utilisation rates accords well with the expectation that the pattern of computer communications is, indeed, "bursty".
- very low error rates: rates of 10^-^6 erroneous packets were recorded. Most of these were traceable to faults in individual stations, e.g. receiver's too long turnaround time causing packets that were well-formed on the ether to be subsequently corrupted.
- low incidence of collisions and delays: under normal load conditions only 0.79% of packets were delayed due to deference, and less than 0.03% were in collisions. Thus over 99% of packets were moved through the network with no delay.

Although utilisation is typically only a fraction of the available bandwidth, early modelling of the ethernet [Metcalfe & Boggs, 1976] led to predictions that it should continue to perform well even during periods of very heavy loading. Thus the same researchers who had measured what constitutes a normal loading on the ethernet went on to induce abnormally heavy loads in order to verify this prediction. Their results are summarised in the following figure (curves a):



As total offered load increases, so does throughput (utilisation) and delay, until a maximum throughput is reached. In unstable systems like Aloha (curve b) the throughput falls off sharply beyond that point and delays rise towards infinity. However in the ethernet studied throughput continued to rise up to and beyond 90% of the channel capacity. Thereafter delays worsened but throughput remained stable, i.e. the network did not become congested. At the same time there was fairness in allocation of capacity: with 100% offered load (10 hosts at 10% per host) the 94% total throughput was made up of individual throughputs ranging between 9.3% and 9.6%.

Modelling also predicts that channel utilisation will be a function of packet size. For small packets, the time lost in collisions and collision resolution is a relatively greater fraction of packet size, and thus the total utilisation is relatively less, than for large packets. With very small packets (or a very wide-area network) the packet's transmission time approaches the slot time, and utilisation falls to 1/e, which was the upper limit on the efficiency of the slotted Aloha model, a packet radio precursor to the ethernet. Experiment confirmed that the highest utilisation rates are with full-size data packets. In practice, however, packet lengths exhibit a bimodal distribution. Although the mean packet length was recorded as 122 bytes, over 65% of the packets in this study were less than 32 bytes (which was the median) and therefore the impressive results of the figure above must be interpreted in this light. Still, the evidence is strong that ethernets do have favourable performance traits and they do meet the design aims mentioned above.

These conclusions were re-affirmed by a second study from within Xerox [Crane & Taft] which went on to identify the real bottleneck to better end-to-end throughput as the limited capability of hosts themselves, e.g.

in handling the software overheads and per packet processing necessary for using the network. Whereas previously the effective constraint has been the low performance of the (long-haul) network, now the high-bandwidth, low-delay properties of LANs have begun to highlight the constraints operating at the network's interface.

It is clear that efficient communications depends upon more than a reliable medium and a theoretically-sound method for regulating channel access and collision recovery. With LANs especially, it depends on the efficiency of higher-level protocols and the interaction between adjacent protocol layers [Donnelly & Yeh, 1979]. It depends also upon hosts' characteristics, e.g. buffering resources and I/O capabilities, and on the extent to which certain communications processing tasks can be offloaded onto their network controllers. Overall system performance cannot be predicted from measurement of the low-level contention mechanism alone. Significantly, in the Shoch and Hupp study it was only this mechanism that was being measured: the artificially high load of packets were being addressed to a fictitious destination which, of course, "accepted" them all. A recent study of a similar multi-access contention bus model indicates that under reasonable assumptions about host buffering and host-network data rate mismatch, the maximum rate of channel utilisation (approximately 50%) was far lower than the rate when its contention-mechanism was measured alone [Watson, 1980].

3. Edinburgh CSMA/CD Network

Edinburgh network incorporates all the essential control structures and behavioural traits of ethernets as discussed in the section above. Nonetheless there are important differences between the Edinburgh and Xerox networks which can be brought under three headings: network parameters, target environment, controller architecture.

Network Parameters. The Edinburgh network adheres closely to the general principles of the Xerox ethernet and the overall packet structure. At the time it was designed the proposed industry standard 10 mbit/sec ethernet had not been defined and the constraints of using currently available components dictated the slower baud rate (2.1 mbit/sec) in any case. There are also differences on the following counts: maximum and minimum allowable packet lengths, algorithms for determining the back-off and checksum, interpacket spacing, length of preamble appended to the front of packets, and data rate. It is expected to install a compatible network once the specification is fully defined and accepted as a wide standard and there are available appropriate integrated components that would simplify the design of the station. To the user, all the resulting differences will be invisible except perhaps, under conditions of extreme loading. Present lower data rate combined with the present lower maximum packet length would manifest themselves in a lower throughput of the network. All other differences should be screened from the user by the protocol.

Target Environment. Our network is aimed at one particular teaching and research environment, where resources are shared between sophisticated systems for data management, graphics, and distributed processing. Xerox's ethernet, on the other hand, is a product aimed at world markets for information processing and office automation systems. The intention is to promote a clear standard which is widely acceptable and easily interconnected. Compatibility is perhaps the main objective. As stated in the specification document [Digital, et al. 1980] "every implementation of the ethernet should be able to exchange data with every other implementation". The choice of wide address spaces and their protocol philosophy bears this out. The network's interface is designed as a common denominator to which the existing universe of hosts can be attached with minimal effort. Starting with any common denominator, the result must be fairly low, i.e. the interface that Xerox has proposed is one of link-level service only. Functions such as acknowledgements, error checking (excluding line transmission errors caught by the CRC), security, and flow control are relegated to higher layers (collectively termed "the Client Layer") and excluded from the scope of the specifications. The issue of enforcing compatibility at these higher levels (network, transport, and above) is not treated in the specification published so far. The sole concern to date has been with the lowest two layers of any eventual network architecture.

Controller Architecture. In Xerox's 3 Mbit prototype the controller was implemented as a mixture of hardware, microcode, and software running on the Alto and Dorado host processors [Thacker, et al. 1979]. In the 10 Mbit proposed standard the potential of LSI for communications hardware is to be exploited: the controller's functions are to be partitioned between several chips produced by the Xerox-Intel-DEC consortium. The controller will offer clients a single, uniform, low-level interface, integrated in silicon and satisfying the physical layer and data link layer specifications as published [Digital, et al., 1980]. In contrast, Edinburgh's controller is implemented as a single-board device built around a Z-80 processor with 4K bytes of PROM holding the firmware, 4K bytes of buffering RAM, and a 4-channel DMA controller handling the station's four logical channels: ToHost, FromHost, ToEther, FromEther. There is also custom hardware to handle checksum computation, transmit deference, address recognition, and serialisation/deserialisation of data.
Thus, subject to certain real-time constraints, the station's processor is relieved from most of the tasks of data link management and is able absorb the overheads of processing higher layer protocols from its host. The 4K bytes of writable memory is sufficient for maintaining a set of dynamically-assignable data buffers; it can also hold the contexts corresponding to multiple network connections which a station can maintain concurrently.
In brief, the controller is able to provide the host with a powerful, programmable, multi-channel interface to the network.

I. Network Parameters and Protocol

1. Signal on the cable

Data is transferred over the network cable in Manchester-encoded form at the rate of 2.1 Mbits/sec. Binary 0's are encoded by negative transitions: 0V -> -5V and binary 1's by transitions: -5V -> 0V. When the network is idle the signal on the cable is at 0V.

2. Packet format

A packet on the network must be at least 16 and not more than 548 bytes long.
It is composed of a 14-byte fixed-length header, up to 532 bytes of a variable-length data field, and a 2-byte Cyclic Redundancy Check (CRC). A packet's data field may contain further sub-divisions of control fields supporting higher-level host-to-host protocols, but for the network's purposes the whole of a packet's data field is opaque.

The header is a sequence of bytes conveying the packet's type as well as its source and destination addresses on the network. Together with 2 bytes of CRC at the end of the packet, it contains all the control information necessary for the station to perform its functions of packet transmission, acknowledgements, packet sequencing, setting up or severing connections, and screening the flow of packets through them.

The Digital-Intel-Xerox proposals on the lengths and respective functions of the header's fields are observed, thus ensuring an important degree of compatibility between the network and the proposed standard. The header is formatted as follows:

6 bytes: destination address
6 bytes: source address
2 bytes: type

The bytes of a field are transmitted over the eventual local interface in the same order as they are transmitted over the ether.

An address (source or destination) field is defined as follows:

byte 0: Most significant bit (bit 7) is reserved and equal 0.
The remaining 7-bits (6:0) when in the range 1:127 denote a unique station address within the network, or else, if zero, a broadcast that will be received by all stations on the net.

byte 1: an 8-bit value used to identify the context within a station to which the packet refers. At present, for non-broadcast packets (byte 0 <> 0), a context identifier is in the range 0-15, and denotes one of 16 available logical ports through which the station should route the packet to and from the ether. Two such ports (if appropriately set up) in two different stations are used to maintain a logical channel between hosts of these stations.
For a broadcast packet (byte 0 = 0) byte 1 identifies one of 256 broadcast channels, where each station can be set up to receive packets only from a chosen subset of these channels.

bytes 2-5: these bytes will not be used to distinguish between stations locally, i.e. they will be set uniformly for all packets moving within the bounds of a network. If, later, a license were to be obtained from Xerox, these bytes would hold the 32-bit universal address that distinguishes the network from all other sites. At present all these bytes are set to zeroes.

The first byte in type field identifies the contents of a packet, with the 7 least significant bits indicating the packet type. Bit 7 is used to differentiate between the transmitted packets (=0) and network acknowledgements to those packets (=1).
In other words: if bit 7 is set the packet is a station-to-station acknowledgement, and the low-order bits encode the type of packet that it is an acknowledgement to. If bit 7 is clear, then the packet is "live" (i.e. not an acknowledgement although perhaps a re-transmission) and must be acknowledged.

The second byte of the type field holds a sequence number starting at 0 and then cycling in the range 1-255. Any port supporting a virtual circuit will maintain two such numbers, reflecting the full duplex potential of these connections. Value 0 is used to distinguish the very first packet transmitted through a given port of the station.

Differences remain between Edinburgh network and the proposed standard over two formatting conventions. Xerox et al. propose that total packet lengths must be at least 64 octets (bytes). This way, the transmission time of any packet is guaranteed to be far greater than the slot time (i.e. the round-trip propagation delay)on even the most wide-scale (2.5 km) site, and thus collisions are guaranteed to be detected. Edinburgh network, being of a smaller scale, has a low slot time (3 usec) so that the header length alone ensures a transmission time that will exceed it. Since the Xerox checksum is 4 bytes, their data fields must be at least 46 bytes (64 - 14 byte header - 4 byte CRC) and can be as long as 1500 bytes. This large upper bound on packet size facilitates file transfers and, as noted in section 2 of the introduction, can yield a higher rate of channel utilisation under heavily-loaded conditions. Data fields on the Edinburgh network may be of null length, and there is an upper bound of 532 bytes which was also set by the station's fixed buffering capacity.

A second point of difference is the function of the high-order bit of byte 0 of a network address. Xerox et al. name this the "physical/multi-cast bit". If clear, it denotes a physical station address that is unique over the universe of all ethernet sites; if set, it denotes a multi-cast address associated with a set of logically-related hosts. A broadcast is a limiting case of a multi-cast address denoting the set of all hosts on a given ethernet. On the Edinburgh network this bit serves no addressing function. It is always set to zero to synchronise the receiving hardware of all stations listening to a packet on the ether.

```
+==========+  Packet destination address                          6 bytes
¦ 0aaaaaaa ¦   station address within the network (0 => broadcast)
¦ bbbbcccc ¦   non-broadcast: 0000cccc is a port number
¦          ¦   broadcast     : bbbbcccc is a broadcast channel number
+----------+
¦ 00000000 ¦   4 bytes of network address
¦ 00000000 ¦
¦ 00000000 ¦
¦ 00000000 ¦
+==========+  Packet source address (format as above)              6 bytes
¦ 0eeeeeee ¦
¦ ffffgggg ¦
+----------+
¦ 00000000 ¦
¦ 00000000 ¦
¦ 00000000 ¦
¦ 00000000 ¦
+==========+  Packet type field                                    2 bytes
¦ hiiiiiii ¦   h: acknowledge flag , iiiiiii: packet type
¦ jjjjjjjj ¦   sequence number: start at 0 and cycle in 1-255
+==========+  Packet data field                            0 - 532 bytes
¦ ........ ¦
¦ ........ ¦
¦ ........ ¦
+==========+  Cyclic Redundancy Check                              2 bytes
¦ kkkkkkkk ¦
¦ kkkkkkkk ¦
+==========+

                                                  ----------------------
                                          total     16 - 548 bytes
```

Packet format on Edinburgh network

3. Network protocol

Station acknowledges all non-broadcast packets that were received by any of its opened ports. For each port it buffers only those, which sequence number is different from the recently buffered packet or is equal to 0. The uniqueness of sequence number 0 and the fact that its packet is always buffered should help in situation of multiple opening of the same port to the same remote node (e.g. repetition of bootstrap request after the first unsuccessful attempt).
Broadcast packets are not acknowledged by the receiving station(s).

II. Station Architecture and Construction

The station can be divided into the following functional units:

- the tap responsible for the physical handling of the ether
- ether receiver and transmitter, which implement the ethernet link-level protocol
- host interface responsible for customizing the station to the interfacing requirements of a particular host
- station controller, whose task is the buffering of data and implementation of higher level protocols thus relieving the host from the details of handling the ether.

The complete station consists of 3 printed circuit boards: one containing the tap which can be placed directly on the cable, the second (the main board) with ether transceiver and station control and the third one containing a customized host interface.

1. Ether cable and tap

Ether cable is a 750hm coaxial cable with terminators at both ends. One point of the cable shield must be grounded. When the network is passive, the core of the cable is at the potential of the shield. The ether is driven by tap driving transistors through shorting the core to -5V.
In the following descriptions we will refer to the active ether as being "high" and passive as being "low".

The tap is responsible for the actual driving of the ether and reception of signal from it as well as for detecting the collisions and jamming the ether.

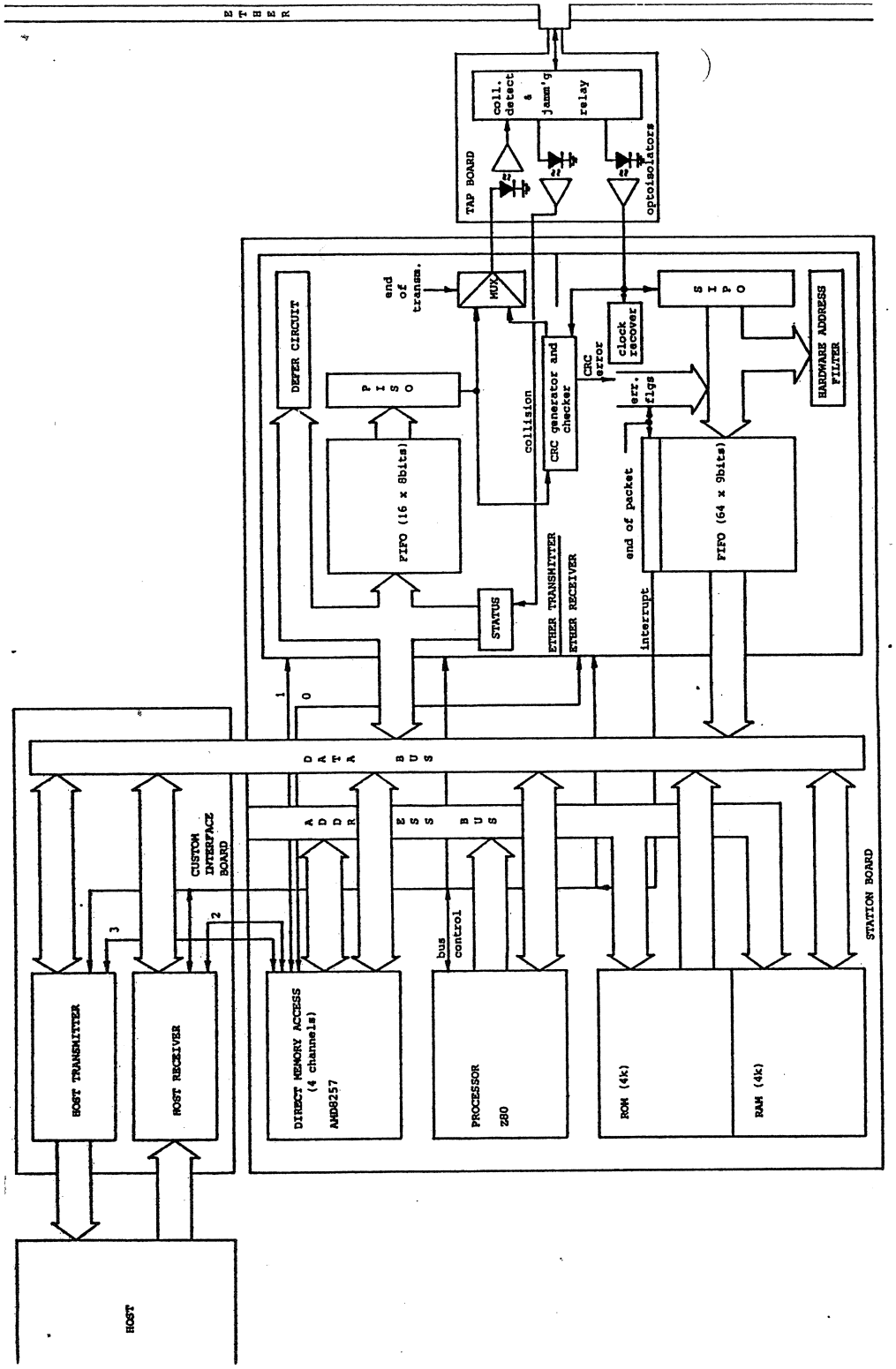
1.1. Collision detection and jamming

Collisions are detected in a D-type flip-flop by clocking into it the signal received from the ether with the positive edge of a signal currently being transmitted. Thus, a high signal on the ether while the transmitter is attempting to invoke a low-to-high transition means that some other station is also driving the ether at the same time.

The rising edge from the collision detecting circuit triggers off a 3 microseconds monostable whose output is OR'ed with the transmitted signal. This enforces a slot-time jam on the ether, halting any other transmissions (those involved in collision) by forcing them to collide and also causes a loss of synchronization in all the receiving stations. A collision signal is sent to the station transmitter setting its collision latch, which halts the transmitter untill it is explicitly released by the controller.

1.2. Safety relay

The tap is connected to the actual wire of the network through a fast relay. The serial signal produced by the transmitter is integrated and averaged in time by an analogue circuit, which upon detection of the transmitter activity rising above the permitted level will disconnect the tap from the ether by switching the relay off. The connection will be reestablished automatically after a certain time of transmitter being inactive. The circuit is used as a safety measure for hardware failures of the tap (e.g. failure of the driving transistor) as well as not to affect the ether when powering on the station.



1.3. Electrical isolation

Because of the differences in ground reference voltages in different locations of the network site, the tap circuitry is electrically isolated from the rest of the station. The tap has a separate, isolated power supply and the signals are coupled through optoisolators with over 3kV threshold voltage and 10 Mhz bandwidth.

The 3 signals: "data out", "data in" and "collision" are sent between the tap and the controller boards using line transceivers, thus allowing the tap to be up to 30 metres from the station, which is usually located in the frame of the host.

2. Ether receiver

2.1. Silence detection

The ether receiver listens to the incoming signal from the ether and declares it "silent" if the signal remains low for 3 microseconds. Detection of silence has the effect of enabling the remainder of the receiver circuitry and also allows the defer circuitry of the transmitter to start its countdown before starting any pending transmission.

2.2. Clock recovery, demodulation and serial-to-parallel conversion

Clock and data are recovered from the serial Manchester-encoded signal arriving from the tap by a phase-locked loop type digital circuit with resolution grain of about 60 ns. The circuit is activated upon detecting the silence on the ether and then locks to the first transition in the received signal. It then resynchronizes to every transition appearing in a 120 ns wide "window" around a point 500 ns away from the previous transition. Loss of synchronization (lack of transition in a window) is treated by the receiver as the end of the current packet.
The recovered clock is used to shift the demodulated data through a serial-in parallel-out receive register and a CRC-checking circuit.

2.3. Hardware address filter

The first byte of the received packet, which holds the address of the destination station is compared with the contents of the station address register and if these values match, the packet will be accepted by the receiver. Otherwise the packet is ignored. In addition to the above, the address recognition circuit also accepts the all 0's address used for broadcasts on the network.
Such address filtering relieves the station controller from having to process all the packets that appear on the ether (as is necessary with the Xerox prototype systems) and by limiting them to those relevant to the particular station gives the controller more time for other activities.
Obviously, the station used for monitoring the traffic on the network has to have the above filtering switched off.

2.5. First In First Out buffer

During the reception of the packet, the receiver must provide a guaranteed buffering capability of 1 byte in every 4 microseconds. Thus, there is a need for accommodating the difference between the fixed rate of transfer from the ether and the throughput of the station data bus which though potentially twice as high as the ether rate is bursty because of processor contention.

There is also a need for providing sufficient buffering capability to give the controller enough time for setting up new buffer for packets arriving closely one after another (the spacing could be as low as 3 microseconds).
This is achieved by placing a 64 byte First In First Out memory between the receiving circuitry and the station data bus. The memory is designed to work with 1-2 bytes of data in it during the transfers leaving over 60 bytes of spare buffering capacity for any temporary halts in transfers on the bus.
The memory is 9 bits wide, 8 bits being used for data and the 9-th to flag the last byte appended by the receiver at the end of every received packet. This terminating status byte contains flags of errors detected by the receiver while receiving the packet (see next section).

2.5. CRC evaluation and error logging

Receiver and transmitter share (depending upon which one is active) a CRC generator and checker (station cannot be receiving and transmitting at the same time). Serial data received from the ether is shifted through the checker (together with the last 16 CRC bits) and at the end of reception its (zero if the CRC remainder=0) output is inserted as bit 7 of the byte appended by the receiver to the end of the received packet.
This last byte is also used for logging other reception errors like:
- ether-high at the end of the packet, which is usually due to the jam occuring in collisions (bit 6)
- buffering overflow which occurs when there is no FIFO buffering space upon the reception of a consecutive byte from the ether (bit 5).

The last error condition completes also the reception and results in ignoring of the rest of the packet.

3. Ether transmitter

3.1. *Transmitter FIFO

A 16 byte FIFO is used by the transmitter to cater for any mismatch between the bursty character of transmissions on the station bus and the steady (4 microseconds per byte) transfer rate required by the ether transmitter. The buffer is controlled to operate almost full during the transmission in order to provide a large safety margin of about 15 bytes for any irregularities in transfers on the bus.

3.2. Defer circuit

The defer circuit consists of a down-counter that can be loaded with a transmission defer value by the controller. It counts down with 3 microseconds cycle once the Ether Receiver declares the ether to be silent. Upon reaching zero the circuit allows the transmitter to go on with any pending transmission, provided the ether remains silent. The counter remains at 0 unless reloaded by the controller, thus defining the default intertransmission spacing to be equal to the minimum "silence" time on the ether.

Loading the defer counter also clears the collision latch that can halt the transmitter as the effect of a collision detected in the previous transmission attempt.

3.3. CRC generation

During the transmission, serial data shifted out from the parallel-in serial-out transmit register is also shifted through the CRC generator (assigned for time of transmission to the transmitter). At the end of data transmission (detected by the FIFO empty signal) the 16 bits of CRC remainder are shifted out from the generator and appended to the transmitted packet.

Data is sent from the transmitter logic to the tap board already in Manchester-encoded form.

4. Controller

4.1. Random access memory

There are 4k bytes of writeable memory which accommodate six 548-byte buffers for data passing between the ether and the host. The memory is also used to store contexts for 16 ports maintained by the station on behalf of its host as well as parameters for processes supporting different activities of the station.

4.2. Direct memory access

A 2Mhz Intel8257 Direct Memory Access controller is used to implement fast data transfers to/from the ether transceivers and the host. It is capable of tranfering data at the rate of 0.5 Mbyte/sec and its channels: 0, 1, 2, 3 are assigned to Ether Receiver, Ether Transmitter, FromHost and ToHost channels respectively.

4.3. Processor

A 4Mhz Z80 microprocessor is used to supervise the activity of the channels to implement the ethernet protocol as well as to present a variety of higher level service to the station host.
The processor can read data and control from all four channels as well as status information of those channels (e.g. collision latch, host interface busy bit etc.). It can write data and control into the ToHost channel, set up control registers of ether transceivers (transmission defer and address filter) and also reset them (e.g. clear collision latch and empty Ether Transmitter FIFO).
The processor is responsible for setting up the DMA channel and control registers as well as for handling the completion of the transfers on individual channels.
It can be interrupted from the FromHost channel by a control character received on its interface or from the Ether Receiver by bit 8 in the receiver FIFO. Byte appending the received packet appearing in the Ether Receiver channel reperesents the completion of buffering the packet in memory by DMA. The processor operates in what is called by its manufacturer the interrupt mode 2, in which upon interrupt acknowledge, the interrupting devices place data on processor bus. The data is then interpreted as the 8-bit extension to an 8-bit interrupt register, thus forming the jump address to the appropriate interrupt serving routine. Interrupts from the ether are given priority over those from host.
On top of these functions the processor has the task of implementing the protocol on the station-host interface.

4.4. Read only memory

4k bytes of ROM are used to store the processor firmware as well as the assigned individual characteristics of the station. These include station network address and implied by it, mode of operation over the interface to host.

5. Host interface

5.1. Byte interface

At present station and host are physically connected through a 9 bits wide parallel bidirectional interface, which from the user point of view can be seen as two separate, unidirectional 9 bits wide logical channels.

Versions of this interface have been constructed to connect to:
- DEC Unibus (for PDP11 and VAX)
- Interdata multiplexor bus
- EUCSD general-purpose links (this allows for additional separation of the station from its host)
- EUCSD "Meccano Kit Computer"

5.2. Memory-mapped interface

In the future it is likely that hosts and stations will share common memory, where the station would deposit the received data in places indicated to it by the host and would collect data to be transmitted from appropriate "mailboxes" in that memory. This is the most probable interface for the target station in the Meccano Kit Computer currently under construction in this Department.

III. Station Organisation, Services and User Interface

A network controller (or station: the terms are used interchangeably here) is the network's interface to a host. All traffic across this local interface must be subject to exact control conventions, or protocols, which govern access to the ether as well as communications across it. The obvious purpose of any protocol is to facilitate reliable end-to-end interaction. Precisely, protocols regulate exchanges between corresponding functional levels of any two host-station pairs on the network. Thus the protocols themselves are commonly distinguished by function, arranged into layers, and named accordingly, as in the definitive I.S.O. reference model for open-systems interconnection [Zimmerman, 1980]. The model presents a network architecture spanning seven layers:

7) application layer
6) presentation layer
5) session layer
4) transport layer
3) network layer
2) data-link layer
1) physical layer

The layering framework reflects three main design principles:

- that each layer should serve a well-defined function: that is, it should give a different means of abstracting from the details of any host-to-host exchange

- that the design should be modular: each layer should be independent from the others, and boundaries drawn so as to minimise the control flow between layers. Layers are thus decoupled, and for any one layer the workings of all higher layers (client layers) are transparent to it

- that the overall framework should be comprehensive, but not unwieldy

The protocol project was concerned with the design of protocols at the transport and network levels. Specifications of physical layer and data-link layer protocols for the ethernet already exist. The design of higher layer protocols (session layer and above) were beyond the project's scope. There must, of course, be some protocols operative at these higher layers in order to sustain the typical kinds of applications for which LANs are intended- e.g. resource sharing, load levelling, message transfer, terminal access- but standards have not yet been proposed and, within the Department, responsibility must fall to the designers of the particular end-systems.

A station that handles or absorbs a protocol layer relieves its host of that processing task: thus it offers its host a level of service. As a station handles successive layers of protocol it offers successively higher-level service to its host. Because of the processing capability built into Edinburgh's stations, it was possible to assign to it the processing functions associated with the data-link, network, and transport layers.

There was a survey carried out in the Department to find out what were the eventual future users' views about the network and the services it should provide.

Not everyone consulted had views on every question, nor did their views

always coincide; but it was possible to discern these areas of agreement:

- the Department's motivation was to rationalise an increasingly unmanageable mass of point-to-point links and to allow for easily slotting in new hosts where and when desired, as suits an experimental environment. Also, the network would join up fairly hefty hosts with high bandwidth requirements: the data curator, the filestore, Vax, Charles II, the M68000-based machines. To this extent it will be a configuration that differs from Xerox own sites. There, the ethernet is primarily a thread between lightweight desk-top processors with their own disc storage.

- among these Departmental hosts, many will want to sustain multiple concurrent connections over the network. If their station offers only one port onto the network through which all this traffic must be directed, then overall performance will be constrained by the speed of the slowest of these connections- e.g. all outgoing packets must be delayed while an acknowledgement is awaited from the other end, or successive re-transmissions are attempted. Thus it is desirable for stations to have a multi-port architecture, letting hosts sustain their multiple connections, with each progressing at a rate independent of any other

- rather than each host having its own customised station, running firmware tailored to its own needs, it would be preferable to develop a uniform implementation that could be to some extent parameterised- i.e. it should be flexible and its functions "negotiable". This generality in implementation was felt to be worth any slight efficiency loss compared with firmware tailored ad hoc to specific hosts

- stations should ideally absorb all the processing overheads of network communications. Too often the combined burden of network and higher-level protocols can overwhelm hosts which have only modest memory and I/O resources- e.g. the Department's Interdata 70's. It is clear, though, that different hosts will exhibit very different patterns of use and that their ideal network interfaces will vary accordingly. In the case of the Interdatas, they will primarily use the network to converse with the filestore, and ideally the network should appear as a dedicated point-to-point link to it. They should be able to pass and receive data streams over the local interface with a minimum of network control information. A system like Vax, on the other hand, will want to exploit the full multi-point potential of the network. Since it has the resources for efficient higher-level protocol processing it is better suited with a low-level network interface. Thus the uniform station implementation should support a range of services that takes account of these varying needs.

1. Resources

1.1. Ports

The station maintains on behalf of its host 16 bidirectional ports, each providing individual context for data transfers directed through it. For each port its associated context can be modified by the host with the freedom of modifications varying according to station "trust" in host software.

Each host is given the capablitity of opening and closing ports 1:15 in its station. Opening of a port sets it up to perform data transfers with a chosen context for the remote address.

Port 0 always remains open and as it holds no addressing context, it can perform transfer of data with multiple remote addresses. The remote address of the transfer must accompany every data transfer transaction directed through it.

1.2. Buffers

There are 5 general data buffers each 548 bytes long. These are shared by the opened ports for the purpose of data transfers.

2. Services

There are three types of service: two associated with point-to-point transmissions and the third one being the broadcasts.

- The first type of of the first two services is offered solely by port 0 and allows transfers to/from different nodes (station & port) in the network.
- The second type offered by ports 1:15 supports only single node to single node transfers.

transmitting
For both above services the station attempts to deliver a submitted packet through transmitting it on the Ether, with automatic retransmission, where both the upper limit on the number of retransmissions and their spacing are determined by the port context.

Upon receiving a station-to-station acknowledge for the transmitted packet the port releases the appropriate data buffer and passes an acknowledge message to the host.

Failure to receive a station-to-station acknowledge after the maximum number of retransmissions results in the data buffer being released and a negative acknowledge message passed to the host.

receiving
Packets accepted from the network by the port are immediately acknowledged on station-to-station level and the resulting data frames are appropriately delivered to the host.

2.1. Single node to many nodes service

This service is offered by and only by the permanently open port 0.

It holds no context concerning the destination of data submitted by the host and thus, every submitted data frame must contain a 6 byte destination address for the resulting packet.
For incoming packets, port 0 performs no checks on the source of the received packet and passes to host within the resulting data frame 6 bytes of the packet source address.

2.2. Single node to single node service

Port 1:15 holds as part of its context the remote address of the packets transferred through it.

A port can be open or closed and in the second case it will accept no packets from the network. An attempt to transmit data through a closed port is treated as error in the host-station protocol.

Access to this service is obtained through opening a c' en port by passing to it the remote address for all future data f s directed through it to/from the host.
An attempt to open an already opened port is treated as an implicit closing prior to reopening.

Closing of a port releases all data buffers associated with it.

Data frames submitted to an open port contain only up to 532 bytes of user data and the resulting packets are addressed by the port with the remote address held in its context.
A port accepts from the network only those data packets whose source address matches the remote address held in its context. Only the user data of these accepted packets is passed over to the host.

2.3. Broadcasts

A broadcast packet is distinguished solely on the basis of the first destination address byte being 0. It is transmitted onto the network only once and is not acknowledged at the station-to-station level.

Broadcasts can be sent through any of the station's 16 ports, accordingly to the specific procedure of the port (see above).

They are always received through port 0 and delivered to the host in the form specific to that port.

The second destination byte of a broadcast packet indicates one of 256 possible broadcast channels. Port 0 holds within its context a host-modifiable 256-bit masking register which enables the station to receive broadcasts only on the chosen subset of the channels.
Upon power-up stations have the masking register set to all 0's.

3. Accessing the services through the host-station byte interface

3.1. Interconnection

From the user point of view the interface can be seen as two separate, unidirectional 9 bits wide logical channels.
The 8 least significant bits (7:0) of each of the channels are used to transfer streams of data and control bytes, with control bytes flagged by bit 8 (=1).

3.2. Control characters

In the following description, classes of control characters are denoted by upper case letters, while names of individual characters are written in lower case. Occurrence of a class symbol in the position of an individual character stands for any individual representative of the class.

There are two basic classes of control characters:

- general control characters
  There are 15 general control characters: CHG = 1,2,...,15
  The value of the character is encoded in the 4 least significant bits of the control byte with all 4 more significant bits set to 0. Thus, a control byte containing a general control character has the form: 0.CHG
- port-specific ntrol characters
  15 port-speci)control characters: CHP = 1,2,...15 are encoded in the 4 most s.gnificant bits of the control byte with the 4 least significant bits carrying the number of the port addressed by this character. A control byte containing a port-specific control character addressing port n has the form: CHP.n

A complete list of control characters and their values is given in the appendix of this document.

3.3. Transactions

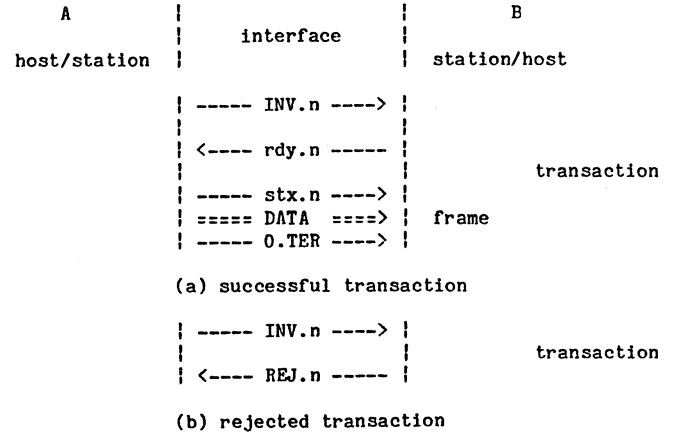
The traffic between the station and its host consists of:
- control exchanges of fixed size and format (general or port-specific)
- port-specific transactions providing a mechanism for unidirectional transfers of single data frames between the host and the associated port in the station.
- port-specific protocol-less data streams

Let (A,B) {(station,host),(host,station)}

A will attempt to invoke a transaction from A to B through port n , by sending to B a port-specific invoking character from the class: INV = {dtx,opn} CHP : INV.n

This attempt can be rejected by B due to the current state of the context held by the addressed port.
A character invoking the transaction will also be rejected when it violates the interleaving rules stated in 3.4.
B rejects the attempt to invoke the transaction by sending a rejecting character from REJ = {err,ack} CHP : REJ.n

A transaction is accepted by B by returning a rdy.n to A. This allows A to send a data frame starting with an stx.n and ending with a terminating character from TER = {etx,eot} CHG : 0.TER



Data contained in a single frame can be network control data, user data or a concatenation of both. It cannot exceed the maximum length of a

transfer assigned to the particular transaction.

3.4. Interleaving

The following rules limit the interleaving of activities over the interface:

a. For each port there may be no more than one open transaction in each of the directions.
A -- INV.n -> B occurring between A -- INV.n -> B and the terminating A -- O.TER -> B will result in ignoring the last A -- INV.n -> B and an error message: A <- err.n -- B

b. No control characters can be nested inside data frames in either of the two directions. This also forbids the nesting of data frames.
A -- CHR.n -> B occurring between A -- stx.m -> B and the terminating A -- O.TER -> B will result in erroneous closing of the frame and an error message: A <- err.n -- B All data transmitted within the frame before and after the erroneous character will be discarded.
Thus, while transmitting a data frame A should queue all its outgoing control characters in a first-in-first-out mailbox and deliver them to B once the frame is finished.

c. The error action specified in (b) has priority over (a).

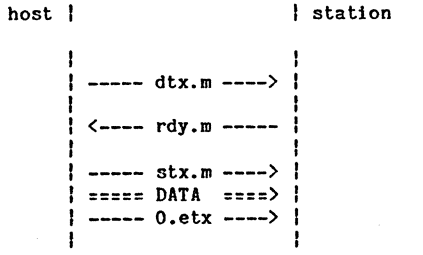
d. Error messages are returned as soon as possible but within the rules a.-c.

3.5. Accessing the services

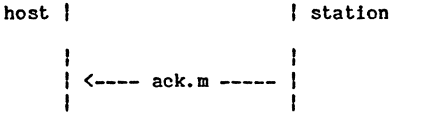
The protocol of transactions for the data transfer is the same for all services and this is described first, before the functional differences.

transmitting

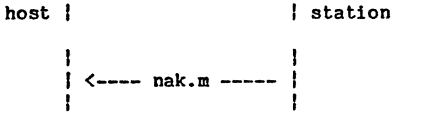
The host can transmit data through any of 16 ports (here m) by invoking the transaction:



Upon receiving station-to-station acknowledge for the transmitted packet the port will release the data buffer at the head of its outgoing queue and will inform the host by sending a single acknowledge character:



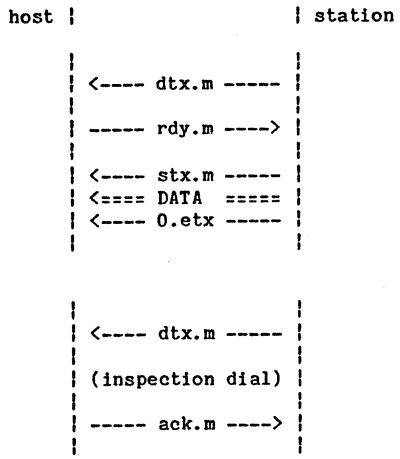
Failure to receive a station-to-station acknowledge after the maximum number of retransmissions results in releasing the data buffer and a negative acknowledge character passed to the host:



receiving

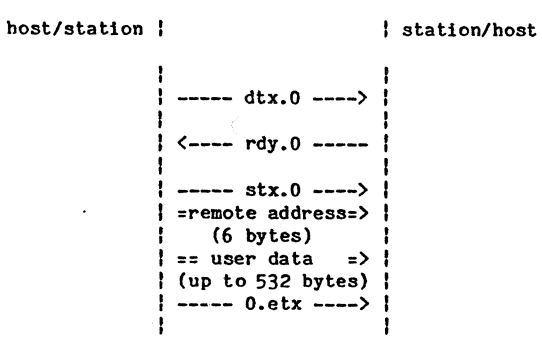
Packets accepted from the network by the port are immediately acknowledged on the station-to-station level and the resulting data frames are put at the end of the port incoming queue.

A port with a nonempty incoming queue tries to deliver the first frame in the queue through invoking an appropriate data transfer transaction. The buffer at the beginning of the queue is freed upon the completion of a successful transaction resulting in delivery of the frame to host or upon rejection of the transaction through positive acknowledge character:



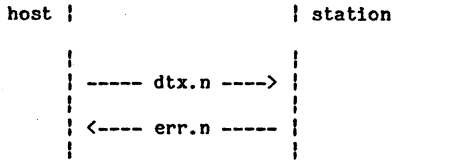
3.5.1. single port to many ports service

Data frames passed over the interface to/from permanently open port 0 must contain in their first 6 bytes the remote (destination/source) address of the transfer.

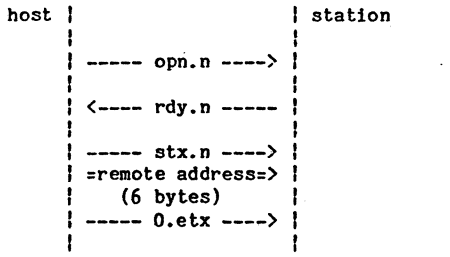


3.5.2. single port to single port service

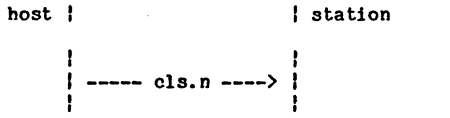
An attempt to transmit data through a closed port (1:15) results in the error message to host:



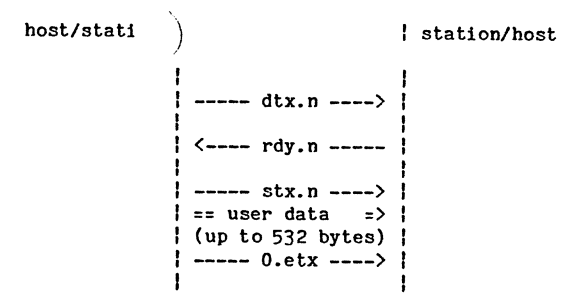
Port n can be opened with the following opening transaction which passes to the port the remote address for all future data frames passed to it by the host:



A port can be closed with a single closing character:



Data frames transferred over the interface contain only up to 532 bytes of user data.



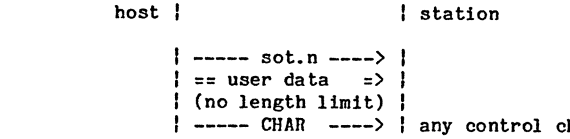
3.5.2.a Protocol - less data transfers in the service

Data can be also transferred between the host and a chosen port (1:15) of the station in the protocol-less mode.

The host invokes this service in an open port n by sending a starting character sot.n The submitted data is then packetized by the controller on the basis of:
- filling up the available buffering space of 532 bytes within the packet (new buffer is connected to the interface once it is available)
- timeout between transfers of consecutive data bytes.

Provided there are no errors in the control exchange the station prompted for a protocol-less service replies by sending an sot.n to the host, thus opening an unlimited data frame in the opposite direction.

Protocol-less mode is terminated by any control character passed by the host to its station.



It should be pointed out that while driving the interface with no protocol the host is denied any control exchanges initiated by the station that otherwise would take place with the remaining ports. All control characters directed by those ports to the host will be queued (with an upper limit on the length of the queue) and delivered once the protocol is reenabled.

It is intended that all client nodes requiring a remote bootstrap source will power up with their stations set up to perform protocol-less data transfers with a source of bootstrap files.
At present, all the Interdata single-user machines power up with a protol-less mode on port 15, being set up to perform data transfers with the Departmental filestore.

3.5.3. Broadcasts

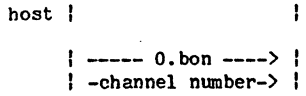
Broadcasts can be sent through any of the ports of the station and are recognised solely by the first destination address byte being set to 0. The procedure for sending broadcast data frames is the same as for any other data transfers on the appropriate port.

When the host passes a broadcast packet through a port, the station broadcasts it once on the network, releases the buffer and returns locally appropriate acknowledge character to host (there is no station-to-station acknowledge of broadcasts).

All broadcast packets are received through port 0 of the station and if accepted by one of the broadcast receive channels of that station, they are delivered to host through that port.

The host can disable all 256 possible broadcast receive channels through sending a single control character: bof

It can enable individual channels by sending a control character: bon , followed immediately by a single byte indicating the number of the channel to be enabled.



All stations power up with all broadcast channels disabled.

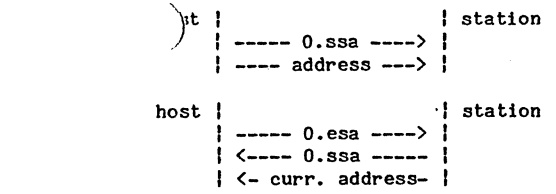
The broadcast channel is specified by its number in the second byte of the destination address of the broadcasted packet.

3.6. Special features

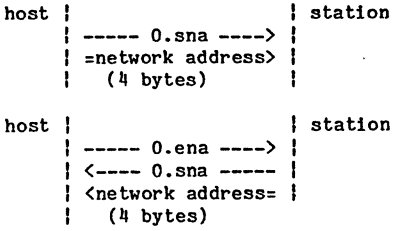
The host can both sample and modify different data fields maintained by each of the ports or by the station in general.
Sampling is available to all hosts.
Modifications of all fields is allowed only to a small number of trusted hosts. Only setting up of the remote address while opening a port is available to all.
Control characters allowed only to the trusted hosts are marked by (*) in their list in the appendix.

3.6.1. Station and network addresses

A host can set the station address by passing to it a control character: ssa followed immediately by one byte of the required address. It can enquire about the current address with control character: esa. This results in the station returning the character: ssa followed by the current station address.



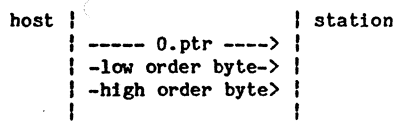
In a similar fashion the host can set and also enquire about the network address in the station. The data transfered contains 4 network address bytes in the order they appear in the bytes 3:6 of the packet address field.



3.6.2. Peek and poke

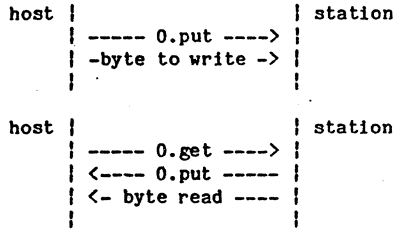
The host can read or write from or to chosen locations in controller memory. The address for those operations is provided by a 2 byte pointer field, maintained by the station. The pointer can be set by the host and is incremented after every memory reference from host.

The pointer is set by sending a control character: ptr , followed immediately by two bytes of the required new value. Low-order byte is transmitted first.



Data can be written to the location indicated by the pointer by sending a character: put , followed by one byte of data to be written.

The host can read from the location indicated by the pointer by sending a character: get This invokes a put character from the station followed by one byte of data.



IV. References

[Boggs, et al., 1979]
David R. Boggs, John F. Shoch, Edward A. Taft, and Robert M.Metcalfe, "Pup: An Internetwork Architecture", Xerox PARC, CSL-79-10, July 1979.

[Brenner, 1980]
John Brenner, "Using Open System Interconnection Standards", ICL Technical Journal, 2:1, May 1980, pp. 106-116.

[Cerf, 1981]
Vinton Cerf, "Packet Communication Technology", in Franklin Kuo (ed.) Protocols and Techniques for Data Communications Networks, Prentice-Hall, 1981, pp. 1-34.

[Clark, et al., 1978]
David D. Clark, Kenneth T. Pogran, and David P. Reed, "An Introduction to Local Area Networks", Proceedings of the IEEE, 66:11, November 1978, pp.1497-1517.

[Crane & Taft, 1980]
R. C. Crane and E. A. Taft, "Practical Considerations in Ethernet Local Network Design", Xerox PARC, CSL-80-2, February 1980.

[Digital, et al., 1980]
Digital Equipment Corp., Intel Corp., Xerox Corp., The Ethernet: A Local Network, Data Link Layer and Physical Layer Specifications, version 1.0, September 30, 1980.

[Donnelly & Yeh, 1979]
James E. Donnelley and Jeffrey W. Yeh, "Interaction between Protocol Levels in a Prioritized CSMA Broadcast Network", Computer Networks, 3, 1979, pp. 9-23.

[Freeman & Thurber, 1980]
H. A. Freeman and K. J. Thurber, "An Updated Bibliography on Local Computer Networks", Computer Communication Review (SIGCOM), 10:3, July 1980, pp.10-18.

[Hopper, 1978]
Andrew Hopper, Local Area Computer Communication Networks (Cambridge Ph.D. thesis), University of Cambridge Computer Laboratory, Technical Report No. 7, April 1978.

[Metcalfe & Boggs, 1976]
Robert M. Metcalfe and David R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks", Communications of the ACM, 19:7, July 1976, pp. 395-404.

[Post Office, 1979]
Post Office Packet Switching Study Group 3, "A Transport Service: Draft for Comment" (The Yellow Book), BIG/CP(79)7, 1979-4-4.

[Pouzin, 1976]
Louis Pouzin, "Virtual Circuits vs. Datagrams- Technical and Political Problems", AFIPS Conference Proceedings, 1976, pp.483-493.

[Pouzin & Zimmerman, 1978]
Louis Pouzin and Hubert Zimmerman, "A Tutorial On Protocol" Proceedings of the IEEE, 66:11, November 1978, pp. 1346-1370.

[Shoch, 1979]
John F. Shoch, Design and Performance of Local Computer Networks, (Stanford Ph.D. thesis), University Microfilms, August 1979.

[Shoch, 1980]
John F. Shoch, "An Annotated Bibliography on Local Computer Networks", Xerox PARC, third edition, April 1980.

[Shoch & Hupp, 1980]
John F. Shoch and Jon A. Hupp, "Measured Performance of an Ethernet Local Network", Xerox PARC, CSL-80-2, February 1980.

[Sunshine, 1981]
Carl Sunshine, "Transport Protocols for Computer Networks", in Franklin Kuo (ed.) Protocols and Techniques for Data Communications Networks, Prentice-Hall, 1981, pp. 35-77.

[Tanenbaum, 1981]
Andrew S. Tanenbaum, Computer Networks, Prentice-Hall, 1981, pp. 3-21, 174-197, 288-320.

[Thacker, et al., 1979]
C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull and D. R. Boggs, "Alto: A Personal Computer", Xerox PARC, CSL-79-11, 1979.

[Watson, 1980]
W. B. Watson, "Performance In Contention Bus Local Network Interconnection", Computer Communication Review, 10:3, July 1980, pp. 19-25.

[Wilkes & Needham, 1980]
Maurice V. Wilkes and Roger M. Needham, "The Cambridge Model Distributed System", Operating Systems Review (SIGOP), 14:1, January 1980.

[Zimmerman, 1980]
Hubert Zimmerman, "O.S.I. Reference Model: The I.S.O. Model of Architecture for Open Systems Interconnection", IEEE Transactions on Communications, 28:4, April 1980.

V. Appendix

1. Control characters on station/host byte interface

* denotes that a character is accepted only by the stations of priviledged hosts.

Code(hex) Mnemonic Meaning, parameters, and description

Port-independent characters: 0.CHG

00		---	Unassigned
01		---	Unassigned
02		---	Unassigned
03		---	Unassigned
04		ENA	Enquire Network Address Evokes response {SNA + 4 bytes}
05		ESA	Enquire Station Address Evokes response {SSA + 1 byte}
06	*	SNA	Set Network Address {+ 4 bytes} Changes the Station's Network address
07	*	SSA	Set Station Address {+ 1 byte} Changes the Station's Station address
08		PTR	Set Peek/Poke pointer {+ 2 bytes, low-order first} Sets pointer for subsequent GET/PUT operations
09		BON	Broadcast On {+ 1 byte} Allows station to accept broadcasts for the specified context number
0A		BOF	Broadcasts Off Makes station deaf to ALL broadcasts
0B		ETX	End of Data Frame Follows last byte of a data frame started with STX
0C		(EOT)	End of File Transfer (not implemented)
0D		GET	Peek Evokes {PUT + 1 byte} and increments pointer
0E	*	PUT	Poke {+1 byte} Alters memory and increments pointer
0F		RESET	Reset Station The station is brought back to its power-up state