Technical Report

UCAM-CL-TR-43
ISSN 1476-2986

Number 43

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Extending the local area network

Ian Malcom Leslie

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted February 1983 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Darwin College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

SUMMARY

This dissertation is concerned with the development of a large computer network which has many properties associated with local area computer networks, including high bandwidth and low error rates. The network is made up of component local area networks, specifically Cambridge rings, which are connected either through local ring-ring bridges or through a high capacity satellite link. In order to take advantage of the characteristics of the resulting network, the protocols used are the same simple protocols as those used on a single Cambridge ring. This in turn allows many applications, which might have been thought of as local area network applications, to run on the larger network.

Much of this work is concerned with an interconnection strategy which allows hosts on different component networks to communicate in a flexible manner without building an extra internetwork layer into the protocol hierarchy. The strategy arrived at is neither a datagram approach nor a system of concatenated error and flow controlled virtual circuits. Rather, it is a lightweight virtual circuit approach which preserves the order of blocks sent on a circuit, but which makes no other guarantees about the delivery of these blocks. An extra internetwork protocol layer is avoided by modifying the system used on a single Cambridge ring which binds service names to addresses so that it now binds service names to routes across the network.

PREFACE

I wish to thank my supervisors, Professor R.M. Needham and Professor M.V. Wilkes, for their advice, guidance and encouragement throughout the course of this research. J. Dion, N.J. Ody, and P.F. Linington have also contributed to this work through constructive criticisms and discussions.

The work described here was funded by the Royal Commission for the Exhibition of 1851, by the Science and Engineering Research Council, and by the Department of Industry.

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of any work done in collaboration. Futhermore, this dissertation is not substantially the same as that I have submitted for a degree or other qualification at any other university. No part of this dissertation has already been or is being concurrently submitted for any other degree, diploma or other qualification.

CONTENTS

CHAPTER 1

INTRODUCTION

In the past fifteen years the development of packet-switched computer communication networks has allowed computer users to access remote machines. More recently, local area computer networks have made the introduction of distributed computing systems possible. These systems allow the sharing of services provided by processors in the system. Such services include, for example, authentication [Girling 82], compiling [Schutt 81], and archiving of rarely accessed data.

There are two types of computer communication networks. For over a decade, wide area networks have been connecting components separated by distances of up to 10000 Km. Local area networks have been designed to take advantage of limited transmission distances, which are never more than a few kilometres [Clark 78]. Simple protocols have been developed for local area networks which are suited to the characteristics of high bandwidth, low delay and low error rates. The motivation for developing simple protocols has been to ensure that a high degree of interaction between processors on a network is not hampered by protocol overheads. The protocols used on wide area networks tend to be more complex in order to deal with the larger transmission times, lower bandwidths, and the lower signal-to-noise ratios associated with these networks.

Wide area networks are, in general, used for remote terminal access, file transfer (including mail), and remote job entry. Local area networks, while supporting these applications, can be used to allow the distribution of computing systems. Distributed computer systems permit a flexible approach in building a computing facility. Potentially a decision to upgrade a large mainframe may be transformed into a decision to add a more modest machine to an existing group of processors. Such systems have some inherent fault tolerance, in that failure of one element, if detected, may cause a degradation in service rather than a complete system collapse.

Continuing developments in integrated circuit technology have meant that processors and memory are becoming less expensive. The prices of mechanical devices necessary for personal computing, such as printers and discs, remain relatively high. Local networks can offer a solution to the problem of expensive peripherals by allowing them to be accessed, and thus shared, over the network.

1.1 Goals

This thesis is concerned with the application of local network protocol techniques to larger networks. A network is developed which possesses many properties associated with local area networks, but which can span large distances and can tolerate failures of parts of the network. This network is built by interconnecting local area networks, specifically Cambridge rings [Wilkes 79a]. The component Cambridge rings may be remote from one another or physically adjacent. The interconnection of physically adjacent rings results in a larger local network; the interconnection of rings which are remote from one another results in a wide area network. Protocols comparable to those used in practice on a single Cambridge ring are used on the resulting network, both in the context of local and of remote interconnection. A particular aim of this thesis is to determine whether the distributed systems built around local networks can be built successfully around the wide area network described here.

1.2 Overview of the Thesis

The next chapter is an introduction to computer networks, focusing on the accepted differences between wide area and local area networks. Since the network being developed here results from interconnecting component networks, an examination of previous work and issues involved in network interconnection is made in chapter three. Familiarity with the Cambridge ring and the protocols used on the ring is important in understanding the adopted interconnection strategy; chapter four gives some background information on this local

network technology. The interconnection strategy itself is presented in chapters five and six. Chapter five discusses some alternatives and presents a solution for local interconnection, while chapter six describes the Universe network, in which Cambridge rings at various sites in the United Kingdom are joined by means of a satellite link. Chapter seven reports some measurements and early experience with the network. Finally, some conclusions are drawn in chapter eight.

CHAPTER 2

LOCAL AREA AND WIDE AREA NETWORKS

Local area and wide area networks can be compared in terms of transmission characteristics, protocols, and applications which they can support. Before such a comparison is begun, some terms and ideas which are common to both types of network are discussed.

2.1 Terminology

The definitions below follow those of Cerf and Kirstein [Cerf 78].

A **packet** is a "sequence of bits, divided into a control header part and a data part. The header will contain enough information for the packet to be routed to its destination." The term **block** will be used interchangeably with the term **packet**.

A **datagram** is a "packet of data with destination host address information (and, usually, source address) which can be exchanged in its entirety between hosts independent of all other datagrams sent through a packet switched network."

A **gateway** is a "collection of hardware and software required to effect the interconnection of two or more data networks, enabling the passage of user data from one to another".

A **host** is a "collection of hardware and software which utilizes the basic packet-switching service to support end-to-end interprocess communication and user services".

Cerf and Kirstein define a **packet switch** as a "collection of hardware and

software resources which implements all intranetwork procedures such as routing, resource allocation, and error control and provides access to network packet-switching services through a host/network interface." Here, the provision of access to hosts will be considered as an optional feature of packet switches.

A **protocol** is a "set of communication conventions, including formats and procedures which allow two or more end points to communicate. The end points may be packet switches, hosts, terminals, people, file systems, etc."

A **virtual circuit** is a "logical channel between source and destination packet switches in a packet-switched network. A virtual circuit requires some form of "setup" which may or may not be visible to the subscriber. Packets sent on a virtual circuit are delivered in the order sent, but with varying delay".

A **PTT** (Post Telephone and Telegraph) is "the authority (or authorities) licensed in a country to offer public data transmission services".

It is useful to distinguish among the concepts of naming, addressing and routing. Shoch suggests that a "**name** of a resource is what one seeks, an **address** indicates where it is, and a **route** is how to get there" [Shoch 78]. The mapping of names into addresses and addresses into routes represents a binding of information. Saltzer [Saltzer 82] has produced a more general model of names and the binding of names in computer networks in which, for example, services, hosts, attachment points to networks, and routes are simply types of entities which have names. Shoch's names, addresses, and routes correspond respectively to Saltzer's services, network attachment points, and routes.

2.2 Protocols and Protocol Layering

Protocols describe a variety of types of behaviour. They may be merely conventions which any two hosts can agree upon, independent of the network. They may be rules, which if not adhered to, prevent hosts from using the network. At a lower level they may specify the inner workings of the network

which are, to a certain extent, irrelevent to hosts. For example, physical protocols describe how bits are represented by signals in transmission media, while application protocols might describe how two computers interact to transfer funds from one account to another. A hierarchy of seven distinct protocols layers has been defined [ISO 82] by the International Standards Organization (ISO) in a model for computer communications. An overview of the ISO model is presented in [Zimmerman 80]. The philosophy behind the approach of a hierarchy of protocols is that each protocol layer provides a service to the the layer above it, and also isolates the lower layers from the higher layers. For example, a protocol layer might provide a service which delivers datagrams to a destination host with a low reliability, such that only 90% of the datagrams are delivered. A protocol designed to achieve a higher reliability may use the first protocol both to deliver data and to deliver acknowledgements of the receipt of data by the destination. The second protocol would retransmit any datagram which went unacknowledged by the destination.

2.3 A Comparison of Wide and Local Area Networks

2.3.1 Topologies

Wide area networks span large distances; some span several countries. A local network might span a building or a collection of buildings such as would be found on a university campus or an industrial site. One effect of this difference in scale of area covered is network topology. Wide area networks must have general topologies[1] which allow links to be placed where needed, permit alternative routing, and tolerate failures on network links. It would be absurd if a break in a link between, say, New York and Boston prevented hosts in San Francisco and Los Angeles from communicating.

---

1 Clark, Pogran and Reed use the term "uncontrolled topologies" [Clark 78]

Local networks adopt a more simple strategy. It is not so unreasonable for failure of a single link to cause total failure in a local network. Also, there is no great performance loss for many applications if the traffic between two machines in the same room happens to travel around a collection of buildings, since added delay will be comparable to the transmission delay in the medium in question. The total length of transmission media in a local network can be less than the length of a single link in a wide area network. Indeed, in the ETHERNET [Metcalfe 76], it is difficult to distinguish between a physical network link and the physical network itself. Faults on some local network networks can be located easily [Wheeler 79] and either by-passed or repaired. Local network topologies therefore tend to be less general. Common topologies are buses, e.g. the ETHERNET, rings, e.g. the Cambridge ring and stars.

The difference in topological complexity has made a large difference in the nodes of the two types of network. The general topologies of wide area networks require complex nodes, often minicomputers, to act as packet switches. These switches perform a store and forward function, and make routing decisions since they will have many incoming and outgoing links. The topologies typical of local area networks require neither routing decisions to be made, nor store and forward functions. Packet switches in the local network are often indistinguishable from host access logic. The one exception to this is star, where the hub or centre is performing a packet switching function, although in many cases this is simply broadcasting an incoming signal; again neither routing nor store and forward functions need be provided.

An immediate consequence of the existence of alternative routes and the store and forward process, is that datagrams may arrive at a destination in an order different to that in which they were originally sent. In local networks, this is usually not the case, so that there is no need for a receiving host to re-sequence datagrams.

2.3.2 Delay

Delay is another characteristic which is affected by the difference in network size. Not all of this delay, however, results from propagation delay in transmission media. For example, the distance between New York and Los Angeles is on the order of 4000 Km. The propagation rate of a signal in coaxial wire is approximately $2 * 10^8$ m/s., so that the raw propagation delay between the two cities could be as low as 20 ms. Wide area networks such as the ARPANET [Roberts 73] do not achieve this because of the delay added by the store and forward nodes of the network and the limited capacity of the links.[1] Since local networks do not, in general, have store and forward nodes, they can come close to realizing the delay imposed by distance.

The delay, Td, in transferring a datagram between two hosts across a local network can be approximated[2] by

$$Td = l/p + k/r,$$

where l is the length of the medium between the hosts,
p is the signal propagation rate in the medium,
k is the length of the datagram, and
r is the data rate.[3]

Within the range of current local networks, either delay term can be dominant. Figure 2.1 shows a table of points of data rate-distance equivalence, that is, where the two terms are equal, for $p = 2.0 * 10^8$ m/s. and k = 800 bits. The trend towards higher data rates in local area networks, for example 50 Mb/s. in HUBNET or 50-100 Mb/s. planned for the next ring currently under development at the Cambridge Computer Laboratory, suggests that delays on the order of tens of microseconds will be common in the near future.

---

1 [Kleinrock 74] reports some delay measurements for the ARPANET.

2 This approximation ignores node delays, which are found in some local area networks.

3 The data rate, r, may be a function of l in some networks, as in the Cambridge ring for example.

Data Rate (Mbits/s)	Length (km)	Time (microseconds)
1.0	160	800
10	16	80
100	1.6	8

Figure 2.1 Data Rate - Distance Equivalents for k = 800 bits and p = $2 * 10^8$ m/s.

2.3.3 Bandwidth and Error Rates

Local networks also tend to have higher bandwidths than wide area networks. This is for two reasons. The first is a direct result of the distances involved. Signals travelling shorter distances are subject to less noise, less attenuation and less jitter than signals travelling large distances. Thus high frequency transmitters and receivers are much less expensive for local area networks than for wide area networks. The second reason is more a result of the different scales, in the organizational rather than the geographical sense, of the two types of network. The opportunities offered by new technology, such as glass fibre, are more easily exploited in the local network where links can be replaced in a matter of days (and without involvement of the PTT). The replacement of all the links of say, the ARPANET, with glass fibre is a task of enormous proportions.

Error rates on wide area networks tend to be higher than local area networks for precisely the same reasons that their bandwidths tend to be lower. For example, the ARPANET in 1971[1] had an error rate slightly in excess of one in 10^6 bits, whereas the Cambridge ring has an error rate on the order of one in 10^{11} bits.[2]

---

1 based on the figures reported in [Kleinrock 74]

2 as reported in [Spratt 80]

2.3.4 Protocols

The bandwidth, delay and error characteristics of local networks provide an opportunity to use protocols which are simpler than those found on wide area networks. It has already been noted that it is often unnecessary to re-sequence blocks. Simplification is also evident in both error and flow control.

Flow control is the means by which a source is prevented from sending data at a higher rate than it can be received by the destination. A scheme which works well on a local area network is for the destination to send a control block every time it is willing to receive a data block. In a wide area network this can drastically reduce data throughput because of the end-to-end delays of the network.

On wide area networks it is thus common to allow a **window** of a number of blocks to be in transit at any given time. If after receiving an acknowledgement for block **k**, a transmitter may send block **k+n**, the protocol is said to be using a window size of **n** blocks. Protocols with a window size of one block do not perform badly on local area networks simply because the end-to-end delay is so low. It is even possible that protocols using large windows, as well as requiring more buffer space and more code to implement them, will have lower throughputs on local networks because of increased software overheads.

Transmission errors and network failures, such as the breakdown or congestion of a packet switch, occur less frequently on local networks than on wide area networks. This affects error recovery strategy. When errors are expected to occur, a communication protocol should provide automatic retransmission to hide the errors from application programs, except in the case of total network collapse. This event must be dealt with by the application program. When errors are not expected, one can allow an entire transaction to fail if an error occurs. The penalty paid when an error occurs is high, but it is not paid very often. The benefit is the elimination of a layer of protocol. In the extreme, one might allow an error to cause the collapse of an entire system or remain undetected. This is the approach taken in most computers, where memory errors are either never discovered or simply cause an

unrecoverable fault.

An example in which an error causes a large transaction to fail can be seen in the Cambridge fileserver protocol [Dion 81]. Reading a megabyte of data from the fileserver can be done by sending a block requesting a megabyte from a particular file, starting at a specified location. The fileserver will send a megabyte of data in a sequence of blocks followed by a control block. The probability of error is sufficiently low that the need to retry the whole operation will not arise often.

This example also shows the use of a protocol which has no flow control. The flow control is performed around, rather than within, the transaction; if the client did not have enough room for a megabyte, it should not have requested it. A common example of protocols which define neither error nor flow control are the exchange [Clark 78], or single shot, protocols in which a client makes a single block request to a service and is provided with a single block result. The flow control here is on a transaction, and thus single block, basis and so is similar to the flow control provided by a window-of-one protocol.

The results of using simple protocols are significant. Firstly hosts which are attached to local networks can be smaller than those attached to wide area networks. These hosts can include microprocessors [Gibbons 80] which have limited space both for code to implement complex protocols and for large buffers, as well as limited programming support.

Besides allowing smaller machines to be attached to local networks, simple protocols reduce protocol overheads which can limit performance. The bandwidth and delay restrictions on wide area networks mean that protocol overheads, by comparison, are small. The situation on local area networks is quite different; the time an operating system takes to schedule a task may be significant in comparison to the time it takes to transfer a block of data across the network.

A third effect of simpler protocols arises from this increase in performance. It is simply that new applications, such as virtual storage swapping over a local network [Dellar 80], become practical propositions.

2.3.5 Applications

Both local and wide area networks are used for terminal traffic, file transfer, remote job transfer and manipulation, and moderate response time transactions. It is important to distinguish both remote job control and moderate response time transactions from the interactions which can occur between hosts on a local area network.

In remote job transfer and manipulation, jobs to be run on a remote machine are presented to client's local machine, and then transferred across a network to the remote machine. The job is executed on the remote machine and some output documents may be produced. Job transfer and manipulation protocols, such as the proposal in [DCPU 81], allow the client to retrieve or dispose of these documents, and to enquire about the status of a job. The client need not be a user but may be a task running in a machine, such as the job scheduler, which might wish to offload a backlog of work. The amount of computation to be performed by a job might be on the order of seconds or minutes, and the response time from job submission to job completion and retrieval on the order on minutes or hours.

Transaction oriented wide area networks such as SITA [Hirsch 74] have been used since 1966. Many wide area networks follow CCITT recommendation X.25 [CCITT X.25]. This recommendation was modified in 1980 to allow the optional features of datagrams and "fast select" to facilitate transaction-oriented traffic. Both of these features are discussed in [Folts 80]. The fast select feature allows the call establishment features of X.25 to be used as an exchange protocol. Data is sent in the calling block and the destination may include data in a response block which also clears the call. While the datagram option may disappear from X.25, only one PTT authority has so far failed to implement the fast select option [Linington 83]. SITA (Société International de Télécommunications Aéronautiques) is a special purpose network organized by the airlines to handle reservations, baggage inquiries, and flight service information. The higher priority traffic, between a reservation clerk and an airline's central computer, is a good example of transactions which occur on wide area networks. The typical request is 18 characters long, while the average response is 100 characters which is returned in an average response

time of approximately two seconds.

In a local network the response times obtainable are much shorter. The low delay in transferring a datagram across a local network is on the order of a few tens or hundreds of instructions on a mini or microcomputer. This delay is also comparable to the task switching times in operating systems used on these machines. For example, simple tests have shown the task switching time of the TRIPOS [Richards 79] operating system on a variety of machines (Computer Automation LSI4's, Motorola M68000's and Intel 8086's) to range from 250 to 800 microseconds.

Thus the grain of computation which can be reasonably transferred from one machine to another is comparable to a small subroutine call or task switch or access to a peripheral device. This has led to systems which are distributed to provide advantageous features, but whose overall performance would be inadequate without the small overheads in data transfer. Two such features are the sharing of peripherals such as printers, plotters, terminals and discs, and the isolation of services in separate machines.

The sharing of peripherals has an obvious economic benefit. The isolation of programs in separate machines provides a strict protection domain for each program. This benefit could not be realized were it not practical to connect small machines to local networks.

2.4 Local Area Network Applications on a Wide Area Network

The term **distributed computing** is used in describing various types of systems. These include: the remote terminal access to computers, mail systems in which a number of machines co-operate in the relatively slow propagation of data from a sender to a receiver's mailbox, and systems which rely on the high interaction rates possible on local networks. In this thesis, the use of term is biased towards this last type of system, where operating system functions can be spread among a number of servers, or where the grain of division of function among a group of processors can be as small as a single procedure. The mechanisms required for remote procedure call (RPC), that is the invoking

of a procedure on a machine different from the one on which the the calling program is running, is the subject of Nelson's thesis [Nelson 81].

It is a primary aim for the network described in subsequent chapters to support this type of distributed system. More specifically, distributed systems whose components are situated in proximity to each other should not be penalized simply because they are attached to a network which spans large distances. If components of such distributed systems are separated by large distances, performance will be limited by propagation delays, and the grain of division may have to be larger. However, the performance loss these systems encounter should be attributable mainly to propagation delays, rather than interference by the network.

CHAPTER 3

NETWORK INTERCONNECTION ISSUES

In order to extend the local area network, some of the problems of network interconnection must be solved or, where possible, avoided. In this chapter the issues of addressing, routing and level of network interconnection are examined. These issues affect the complexity of gateways used to interconnect networks as well as the functions that they must perform. The last section presents these issues in relation to some existing network connection schemes.

3.1 Addressing

In a system made up by component networks, there must be a way for hosts on one network to send data to hosts on another network. This will not be trivial if, as is the case in many networks, the local addressing scheme does not allow hosts outside the network to be addressed. A common solution to this problem is to introduce a new layer in the protocol structure, the internet layer, which handles an internetwork addressing scheme. This could mean that every block of data travelling between networks would contain an internet address in, what is to the lower levels, the data part of the block. In order to use the same mechanisms for both local and internetwork communication, the internet address might be placed even in blocks which are to be sent to local hosts.

It has been argued [Clark 78] that the insertion of an internet or global address is not a large penalty to pay, especially on local area networks where bandwidth is readily available. There are, however, other considerations beside network bandwidth as will be illustrated below.

3.2 Routing Decisions, Interconnection Levels, and Gateway Complexity

The way in which blocks are routed by gateways is very much related to the level in the protocol hierarchy at which the gateways connect networks. Both of these issues in turn dictate the state information required by a gateway.

The simplest approach a gateway can take is to make no routing decisions and retain no state. This is the case in a source routing scheme [Saltzer 80] where the source of a block specifies the route by which a block is to travel. Each datagram contains a variable length routing field. Gateways simply manipulate the routing field in a block and send the block onto the next hop in the route. The route field of a datagram is of variable length with a number of elements or hops along the route. As a datagram passes through a gateway, the element at the beginning of the route field is read and deleted, while a return hop element is added to the end of the field. Thus each gateway need only read the first element and know where the end of the route field is; it does not have to know the format of other hops in the route field. When the datagram reaches the destination, the field contains a reverse route field which the destination can use to contact the source. The responsibility for binding a service to a route resides with the source, possibly through the use of another service.

The next level of complexity is to hold routing tables in gateways. Datagrams containing global addresses are then delivered to the next hop in the route by examining the routing tables each time a block arrives. If these routing tables can be altered dynamically, blocks travelling between the same source and destination may travel by different routes. This allows recovery from gateway failure if alternative routes can be found quickly. It has the disadvantage that blocks may arrive at the destination in an order different to that in which they were sent. In this scheme a service is bound to an address by the source. An address is bound to a route by the gateways and the source, which must pick the first hop on the route. This binding is made every time a block passes through the networks.

If state information about connections is kept in a gateway, then routing decisions need only be made while a connection is established. Thereafter, blocks arriving on the connection are sent down the same route. Hence the binding of a service to a route is made when connection to that service is established. The overhead involved in setting up a virtual circuit may be significantly higher than that involved in routing a datagram. Thus for performing small transactions, datagrams may be more suitable.

Even more state information is required if gateways join networks at a protocol level which includes error and flow control, such as the in transport level gateway at the University of Kent joining a Cambridge ring to the Science and Engineering Research Council (SERC) Network [Dallas 81][1]. Such gateways implement the two standard error and flow control protocols used on the two networks in question. If the services offered by these protocols are dissimilar, some translation between services must be provided or a common subset of services must be found. A translation could be performed at an even higher level in the protocol hierarchy, for example between two terminal protocols. Unfortunately the flow control needed for various types of traffic, for instance real time voice and terminal traffic, is not always the same [Cohen 80]. Thus it may be necessary to implement more than one standard error and flow control protocol on each side of the gateway if the needs of all traffic are to be met.

The primary benefit which a transport level approach gives is to help in matching dissimilar networks by using protocols which are appropriate to each. For example on a local area network a protocol using a window of one block is quite acceptable. On a satellite network where the delay between sending a block and receiving an acknowledgement is on the order of seconds, a multiple block window protocol is more appropriate. A transport level gateway would allow both these protocols to be used, by translating from one to the other.

---

1 although some will argue that by current ISO definitions this is a network level gateway

3.3 Two Interconnection Strategies

In this section the PUP Internet and the interconnection of X.25 networks are examined. The PUP scheme is an example of datagram level connection, while the interconnection of X.25 networks is performed at a virtual circuit level.

3.3.1 The PUP Internet

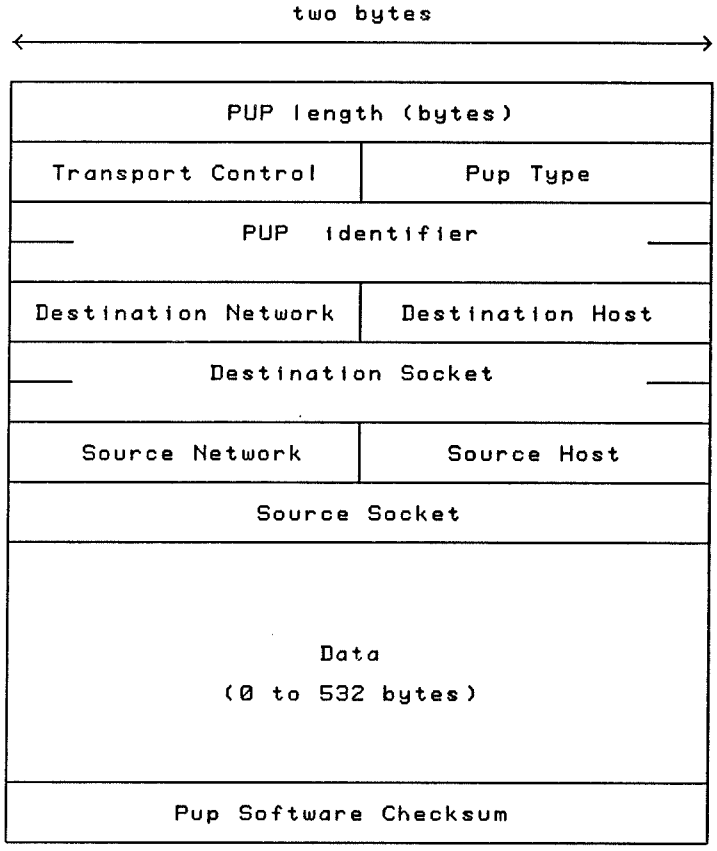

Figure 3.1 The PUP Internet Datagram

The PUP (PARC Universal Packet) Internet [Boggs 80] joins several networks including ETHERNETS, the ARPANET, a packet radio network, and leased lines. The internet is datagram-oriented in that gateways have no knowledge of connections. Virtual circuits are operated on an end-to-end basis. One of the reasons given for not implementing virtual circuits is that datagrams are useful for transaction-oriented protocols.

The task of a PUP gateway is to receive PUP's from one network and send them on to the next hop on another network. Gateways contain routing tables in order to determine from the internet address in each PUP where the next hop on the route is. The internet address, consisting of the network, host and socket, and the packet formats are shown in figure 3.1. In sending a PUP on the packet radio network, several packets must be used. The encapsulation of a PUP in packet radio packets must therefore include some fragmentation control information in each packet. The encapsulation of a PUP within an ETHERNET packet in shown in figure 3.2.

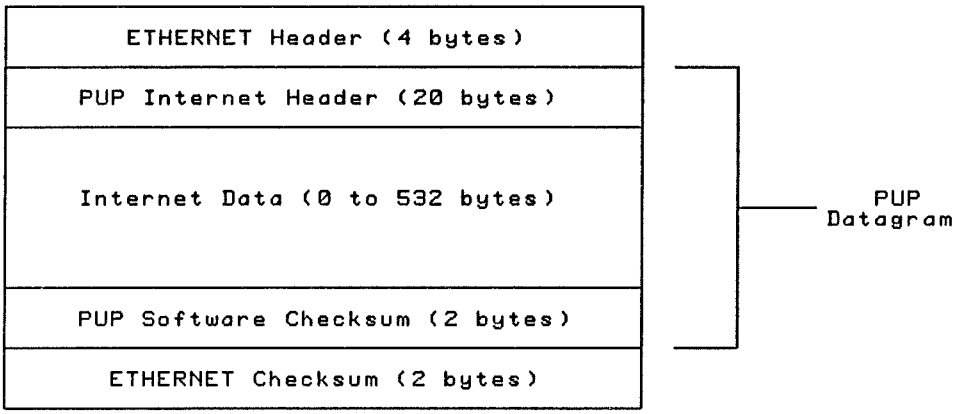

Figure 3.2 PUP Encapsulation in an ETHERNET Packet

Some experimentation in protocol overheads incurred in remote procedure call have been performed by Nelson [Nelson 81] using an ETHERNET. Nelson reports that the extra protocol layer required for the internet functions does result in a significant overhead. Almost all traffic local to a single ETHERNET in the PUP Internet consists of encapsulated PUP's [Needham 82b], so that for some applications, local communication is incurring unnecessary overheads.

3.3.2 The Interconnection of X.25 Networks

The public data networks in many countries follow CCITT Recommendation X.25 [CCITT X.25]. These include PSS in the U.K., Telenet in the U.S.A., and Datapac in Canada. Many private networks such as the SERCNET in the U.K. also follow this recommendation.

Although datagrams capabilities are currently an optional feature in X.25 networks, these networks are virtual circuit oriented. An X.25 network node is called data ciruit-terminating equipment (DCE). A host attached to a DCE is referred to as data terminal equipment (DTE). An X.25 virtual circuit can actually be thought of as three concatenated virtual circuits. An acknowledgement from a DCE to an attached DTE indicates only that the block being acknowledged has been received by the DCE, not that it has been received at the other end of the X.25 circuit. Thus there are virtual circuits between each DTE-DCE pair and between the two DCE's as shown in figure 3.3.

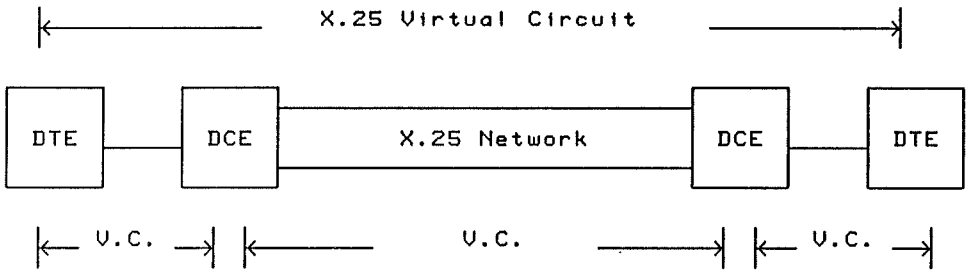

Figure 3.3 X.25 Virtual Circuit

CCITT Recommendation X.75 [CCITT X.75] for the interconnection of X.25 networks specifies a new object, a signalling terminal (STE). An STE, like a DCE is a node on an X.25 network, but instead of attaching to a DTE, attaches to an STE on another network, the two thus forming a gateway. The STE-STE interface is similar to the DTE-DCE interface. The overall result is thus a concatenation of virtual circuits as shown in figure 3.4.

DTE's must only use full DTE addresses in the blocks setting up a call. Full addresses conform to CCITT recommendation X.121 [CCITT X.121]. The two possible formats of X.121 addresses are shown in figure 3.5. The difference between the two formats is that the address space within a country may be

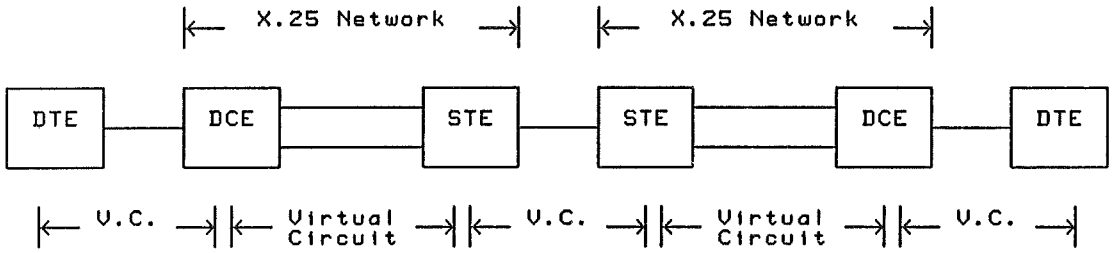

Figure 3.4 Interconnection of X.25 Networks

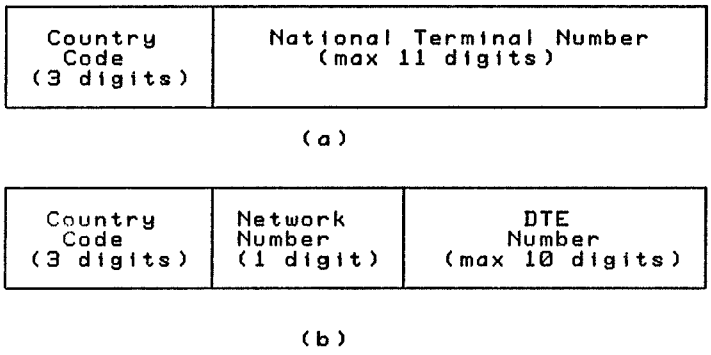

Figure 3.5 X.121 Address Formats

flat as in figure 3.5a or may be divided into the network and the DTE address within that network, as in figure 3.5b. After a call is set up, DTE's use a logical channel number which indentifies the virtual circuit.

CHAPTER 4

THE CAMBRIDGE RING

The Cambridge Ring is a local area network developed at the University of Cambridge Computer Laboratory between 1975 and 1978. It has been used for work in distributed computing at the Laboratory as well as other universities in the U.K.

4.1 Physical Level Protocol

Hosts are connected to the ring through a station and a repeater. The repeater regenerates the ring signals and allows the station to alter the data flowing around the ring. The station's function is to control access to the ring.

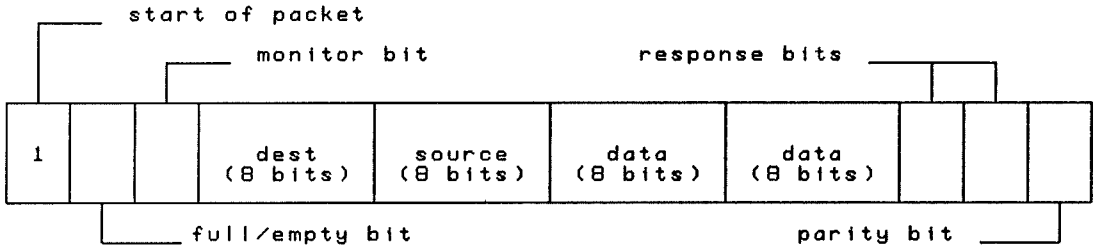

Figure 4.1 Minipacket Format

The Cambridge Ring has a raw bandwidth of 10 Mbits/s. It works on the empty slot principle. A slot, or minipacket, is a collection of 38 bits with the format shown in figure 4.1. There is a constant number of slots circulating continuously around the ring. Stations wishing to transmit wait for an empty slot to arrive, mark it full and fill in the address and data fields. The minipacket continues around the ring to its destination which marks the response bits and normally accepts the data. The minipacket then returns to

the sender where the slot is marked empty and the response bits are read.

Ring stations have a selection register to screen out unwanted traffic. Stations select a particular source to listen to by writing the address of that station into the select register. Alternatively, a station may listen to all stations, by writing the number 255 into the select register, or to no stations, a process known as "going unselected", by writing zero into the select register. When a minipacket is not accepted because the destination chose not to listen to its source, this is indicated to the sender in the response bits. There are four possible responses:

ignored	- no station with the destination address was active,
busy	- the host attached to the destination station had not yet read the last received minipacket from the station logic,
unselected	- the destination station did not wish to receive from the source, and
accepted	- the packet was received by the destination station.

The monitor and parity bits are used to detect error conditions. The monitor bit allows a special station, the monitor, to detect when a transmitter fails to empty a returning slot. The parity bit is used to determine the links of the ring on which transmission errors are occuring [Wheeler 79]. A more detailed discussion of the ring hardware may be found in [Wilkes 79a].

The time delay around the ring is a combination of propagation delay and delays in the repeaters. This time determines the number of bits which fit into the ring. This number may not make up an integral number of slots, so gap digits are used to pad out the train of bits in circulation. In fact, ring stations require a minimum of three gap digits in order to synchronize to the slot structure. However, in order to simplify some comparisons, it will be useful to speak of rings which are exactly filled by an integral number of slots.

The system bandwidth of a ring, that is the useful data bandwidth of the ring, is independent of ring size assuming that there are no gap digits. It is a constant determined by the number of data bits in a slot, the total number

of bits in a slot, and the raw bandwidth. This constant is thus 16 / 38 * 10 MHz or 4.2 MHz. The point-to-point bandwidth, that is the highest data rate from one station to another, does depend on ring size. A station must wait for each minipacket to return before sending another packet. Furthermore, an anti-hogging policy prevents a transmitting station from immediately re-using a slot on its return. Thus the point to point bandwidth is bounded by 1 / (n + 1) times the system bandwidth, where n is the number of slots in the ring.

The actual bandwidth at which a station can transmit may, of course, be lower than this maximum. When transmitting a basic block, a host will take some time in discovering that a packet has returned, reading the response bits and sending another minipacket if it was accepted. If this time is expressed in slot times (1 slot = 3.8 microseconds), then a host is capable of transmitting at 1 / (n + m + 1) times the system bandwidth where m is the transmitter response in slot times. This time must always be rounded up, so that m >= 1.

4.2 Higher Level Protocols

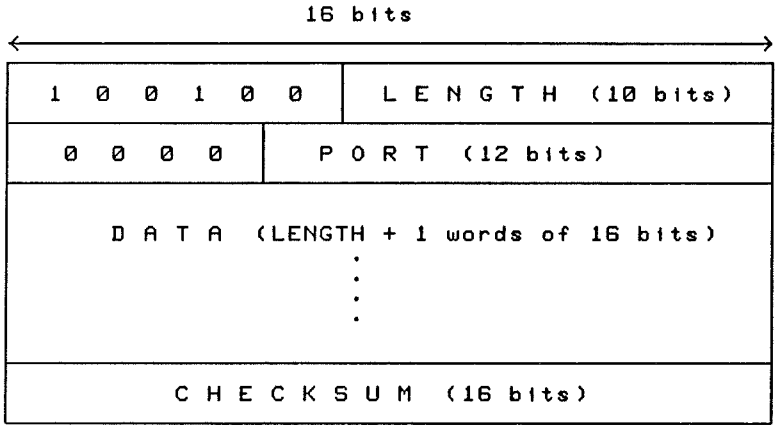

Figure 4.2 Basic Block Format

Almost all communication over the ring is done is basic blocks which are made up of the data part of a sequence of minipackets. The format of a basic block is shown in figure 4.2. A complete description of the basic block protocol can

be found in an annex of [Johnson 80]. While receiving a basic block, a host will select the source of the block, so that other stations are locked out. The minipacket busy response is used to prevent a transmitter from overrunning a receiver.

The port number contained in a basic block is used as an address within the destination host. It would, for example, be the means by which a machine would determine the process for which a basic block was destined. The handling of ring traffic by a host might be organized as follows. A ring handler task would be responsible for dealing with reception and transmission requests from other tasks. A reception request would specify the source of the transmission and the port number on which the block must arrive. When a block was received on that port from the source specified, the ring handler would return the satisfied request to the originating task.

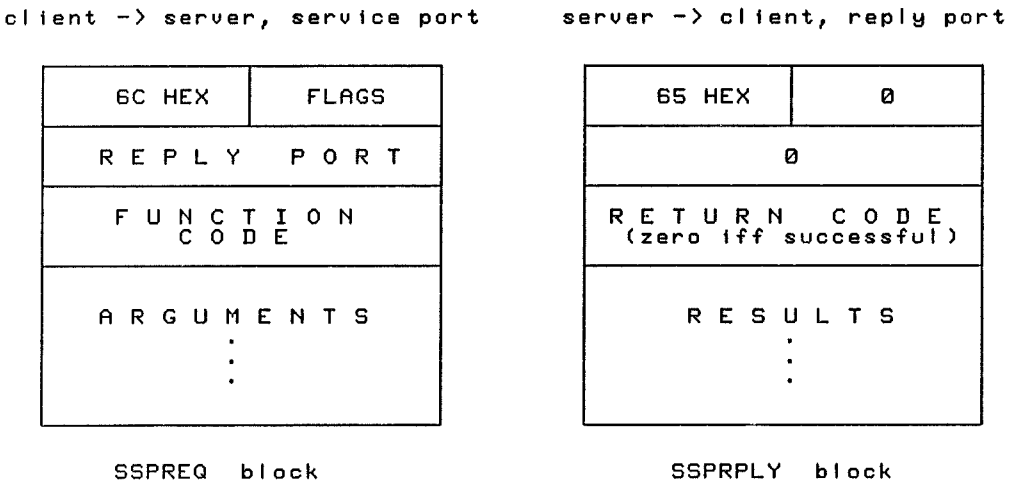

Figure 4.3 Single Shot Protocol Block Formats

Immediately above the basic block protocol are two network protocols, the Single Shot Protocol (SSP) and the Byte Stream Protocol (BSP) [Johnson 80].

The Single Shot Protocol is used when the amounts of data to be supplied and returned can each be contained in a single basic block. Thus no special flow control need be implemented and the protocol defines no error control; in the absence of a reply to a request, the client must try again. Since client retries may cause an operation to be perfromed more than once, the use of the

single shot protocol is best confined to idempotent operations. An operation is idempotent if repeating the operation an arbitrary number of times has the same effect as performing the operation only once. Increasing a file size by 10 bytes is not an idempotent operation whereas changing a file size to 1010 bytes is idempotent. A client sends out an SSP request (SSPREQ) block and waits for an SSP reply (SSPRPLY) block to return. The formats of these blocks are shown in figure 4.3. (The formats shown would be encapsulated inside a basic block; the length fields, route packet and checksum are not shown.) The reply port given in the SSPREQ block is the port to which the SPPRPLY block should be sent. The function code is used to allow one task in a host to offer many services while only making one reception request, since receptions requests are usually organized on the basis of port number. Thus the Cambridge fileserver [Dion 80] has one service port and uses several function codes representing operations such as read, write, open and close.

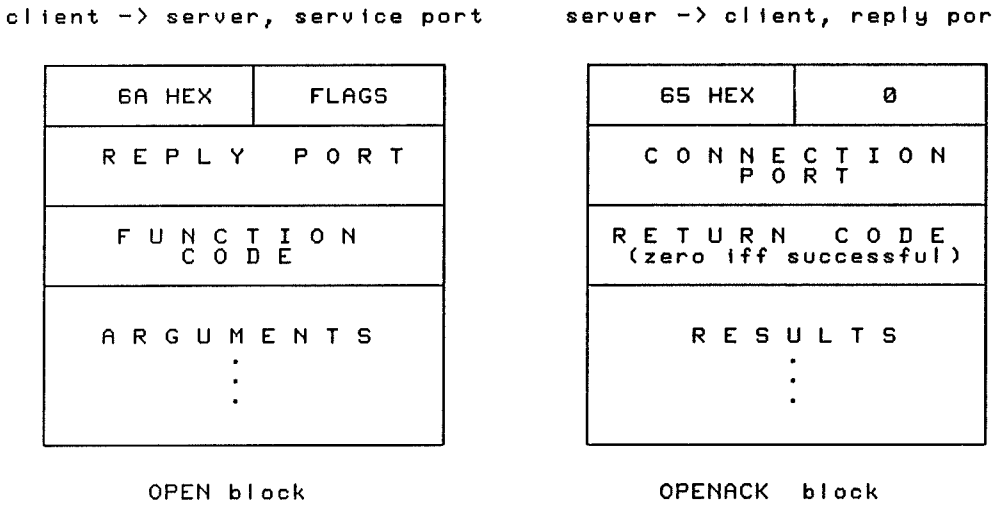

Figure 4.4 OPEN and OPENACK Block Formats

The Byte Stream Protocol is a lightweight error correcting and flow control protocol which is used either when large amounts of data are to be transferred, or when the risk of losing or repeating information cannot be accepted. For the purpose of this thesis, the primary concern is the way in which a byte stream is set up. This so-called "initial connection" is described in annex B of [Johnson 80]. Briefly, an OPEN-OPENACK exchange, very much like an SSPREQ-SSPRPLY exchange, takes place. The difference is primarily the

inclusion of a connection port number returned in the OPENACK block, as shown in figure 4.4. After the blocks are exchanged, the reply and connection ports remain in use until the stream is closed.

The station and port number to which an SSPREQ or OPEN block is sent is obtained from a name lookup service. This service, often referred to simply as the nameserver, binds service names to addresses. The address of the nameserver is the only one that is written into programs. In order to find any other service, the name of the service is given to the nameserver in a name lookup request. The nameserver will reply with a station number, port number and function code. The address of this public port may then be cached by a machine. The name lookup service is in fact an example of the use of the single shot protocol.

4.3 Applications

The uses of the ring at the Cambridge Computer Laboratory are described in detail elsewhere [Needham 82a]. It is worth noting, however, that the distribution of computer systems as touched upon in Chapter 2 is the object of much research at the laboratory. Resources such as printers, discs and terminals are shared by different computers. Functions which might be performed by separate modules in a standard operating system, such as authentication, resource management and file system management, reside in separate processors on the ring. The simple protocols used, especially the single shot protocol, allow these machines to interact with the low overheads necessary for reasonable performance from any interprocess communication system.

CHAPTER 5

THE LOCAL INTERCONNECTION OF CAMBRIDGE RINGS

The task of joining homogeneous networks together is potentially simpler than that of joining different types of networks, since there is no need to perform protocol conversion. Also, by keeping to a minimum the propagation delay through a gateway joining two local area networks, there is an opportunity to create a larger network while preserving the properties of a local area network.

This chapter discusses the problem of joining together Cambridge rings which are on the same site. The collections of software and hardware used to join rings are referred to here as **bridges** rather than gateways since they are not joining networks together, but rather are part of a single network. A question which must be addressed first is why one would choose to make a local network out of many rings joined by bridges, rather than by simply having one large, single ring.

5.1 A Large Ring versus a Multiple Ring Network

The advantages of a multiple ring network over a large single ring can be divided into two areas: performance and reliability.

The most obvious enhancement to reliability offered by a multiple ring network is that a catastrophic failure in one ring, for example a break in the transmission media, does not have a catastrophic effect on the rest of the network. A less obvious benefit is that physical reorganization of the network can be carried out one ring at a time, so that the whole network need not be out of operation for a long period. While this is particularly important in a research environment where there is constant upheaval, it will be of some consequence in any large application.

Performance issues are rather easier to quantify. Two slotted rings, each with exactly one slot, have twice the system bandwidth of a single ring of exactly two slots. Furthermore, the maximum point-to-point bandwidth on each of the single slotted rings is one and one third times that of the larger ring, assuming an anti-hogging policy. Although the minipacket delay around the smaller rings is one half that of the larger ring, basic block delay is related almost directly to point-to-point bandwidth.

Thus if the system bandwidth being used by hosts on the ring causes a significant degradation in traffic throughput,[1] or if some hosts are being limited by the point-to-point bandwidth of the ring, dividing the ring can result in improved performance. One cannot state categorically that performance will improve. There is a great opportunity to degrade system performance by arranging for a significant amount of traffic to travel around both rings and through the bridge joining them. In the case of a bridge using basic block protocols, this degradation would be a result of congestion in the bridge. Two hosts on single slotted rings communicating through an otherwise idle bridge with a delay of a few slot revolution times would be able to transfer a basic block faster than if they were on the same two slot ring. This is simply because the bandwidth limit would be the point-to-point limit of a single slotted ring rather than a two slot ring.

Thus if a ring is divided into subnetworks, some thought must be given to traffic patterns so that, for example, a large processor and its backing store service are not placed on separate rings.

5.2 Some Design Alternatives

The main design goal for the network resulting from the interconnection, apart from the improvement in performance and reliability outlined above, is to maintain local network properties. Avoiding an extra layer of protocol to deal with addressing of nonlocal hosts is also desirable, since this layer adds

---

1 A local area network becomes useless for many applications long before it reaches 100% utilization!

overhead even to local transactions. A related goal is to avoid the need for hosts to perform new functions, particularly in the binding of services to addresses and addresses to routes. Although less important than avoiding new functionality in network hosts, bridge complexity should be kept to a minimum.

Some means for hosts on one ring to communicate with hosts on another ring must be provided. A number of schemes for achieving this are outlined below. For each of these, indications of the complexity of bridges required to implement the scheme and of the effect on host software, are given.

5.2.1 Minipacket Bridges

Bridges could be made to accept minipackets for a number of destinations. This could be done by adding a ring number to both the destination and source addresses and having bridges accept minipackets for a set of rings (a hierarchical addressing scheme), or even with an eight bit address, having bridges accept minipackets for a set of destinations by the use of a simple table lookup (a flat addressing scheme). The major problem in using minipacket bridges is in preserving the response bits described in chapter 4. These response bits are used as a flow control mechanism within the transmission of a basic block. Preserving response bits end-to-end would involve major changes to the ring, in that a station would have to wait for a response for an arbitrary time after it transmitted a minipacket.

A solution to this problem is to implement the basic block protocol in the bridges. Although the responses would only be preserved hop-to-hop, they would be fulfilling their major function, avoiding overrun of a receiver by a transmitter within a basic block. The only reason that these bridges could be described as minipacket bridges is that addressing is performed at the minipacket level; the actual protocol level of interconnection is the basic block. Bridges could be made to receive and transmit a number of basic blocks concurrently, or could operate on one at a time. In the latter case, bridge hardware would involve little more than two modified stations, two microprocessor based intelligent ring interfaces such as the Type II system [Gibbons 80], and some memory for buffer space.

In order to work with this system, host software would only have to be changed to reflect changes, if any, to the address size. Unfortunately, any such changes would require modification to all ring stations and host interface logic.

Preventing blocks from arriving out of order would require controls on the bridges, so that when a route disappeared, no replacement route would appear for some time. The existence of alternative routes would have catastrophic effects since it might allow minipackets within a basic block to arrive in the wrong order at the destination. Alternative routes would thus have to be prohibited.

5.2.2 Globally Addressed Datagrams

A different approach which would have similar bridge hardware, would be to change the basic block to include the global addresses of both the source and destination hosts. A global address could consist of a ring number as well as a station number. Bridges would contain routing tables which would enable them to forward basic blocks either to the destination or the next bridge on the route. To some extent this is the approach taken by the PUP internetwork, although the extra layer of encapsulation in the PUP scheme would not be necessary here, since there is only one type of underlying network. As in the PUP internet, blocks could arrive in an order different to that in which they were sent, unless alternative routes were prohibited.

Host software would thus have to re-sequence blocks, deal with the change in address size, and perform part of the binding of an address to a route, since they would have to determine the first bridge on the route. In practice this would be performed by the nameserver, but the host would still have to cope with the concept of a first hop.

There is an additional problem in that the guarantees which the ring hardware makes about sources addresses would be lost. It would be possible for a host to forge a source address within a basic block. Although bridges could obviously make checks on addresses, this is not sufficient, since a forger could be on the same ring as the destination host. In order to have

the same degree of security as that of a single ring, a host would have to know where all the bridges on its ring were. It could then trust all traffic coming from bridges and perform its own check on traffic from local stations.

5.2.3 Lightweight Virtual Circuits

An alternative, which requires bridges to retain some state information about connections, is to use lightweight virtual circuits. These circuits are termed lightweight as bridges would perform no error control or flow control.

The benefits provided by virtual circuits are that blocks arrive at the destination in the same order as they were sent, and that extra addressing information need only be sent in the block establishing the connection, in this case OPEN and SSPREQ blocks. Blocks within the same circuit are guaranteed only to arrive in order relative to each other; bridges need not preserve sequentiality among different circuits. The scheme still allows alternative routes on a virtual circuit basis; each circuit between the same source and destination may involve a different set of bridges.

After a connection is set up, blocks will follow the same path as the block which set up the path, either in the forward or reverse direction. Thus if authentication is sent down the stream, or in the OPEN block, the connection remains secure. The only additional trust involved is trust of the bridges.

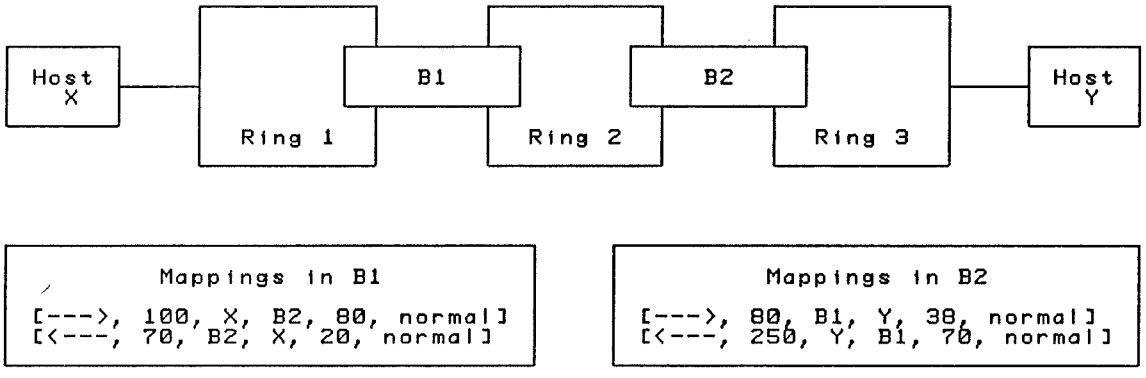

Figure 5.1 Virtual Circuit Mappings

A virtual circuit in this scheme would consist only of a pair of port mappings

in each bridge on the route. These mappings indicate, for a given port on the bridge, from whom traffic is to be accepted and to which station and port it should be forwarded. As an example, figure 5.1 shows two hosts communicating through two bridges. Mappings in the bridges are indicated by an ordered sextet, the components of which are direction, reception port on the bridge, the source station from which traffic can be received, the next station and port to which traffic should be sent, and the state of the mapping. Thus the sextet [--->, 100, X, B2, 80, normal] represents a port mapping from port 100 on the left side of B1 to B2 port 80. Blocks arriving on this port must originate from X, and the state is normal meaning that there is no special processing to do on received traffic. (The use of the state component will be explained below.) Host X is sending to port 100 on the first bridge and is receiving from the bridge on port 20. Host Y is sending to port 250 on the second bridge and is receiving from it on port 38. The mappings stored in the two bridges enable the two hosts to communicate with each other using the same procedures as they would on the same ring. If bridges ensure that no two ports map onto the same destination station and port, then host X can be assured that any traffic it receives from the bridge on port 20 originated from the other end of the circuit.

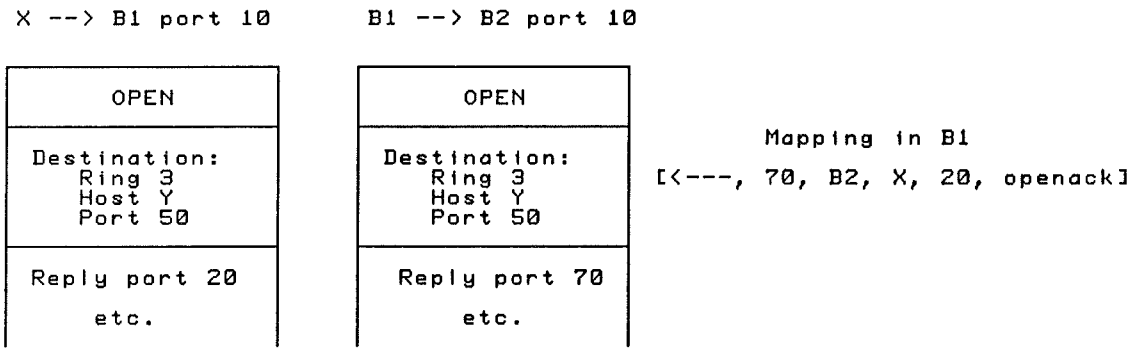

Figure 5.2(a) Virtual Circuit Setup

The way in which a circuit might be set up is shown in figure 5.2. (The example shown is for an OPEN block. The SSPREQ-SPPRPLY exchange can be thought of as a degenerate case.) First, X looks up the name of the service it wishes to contact. The nameserver will respond with the global address, in this case ring 3, station Y, port 50, and the first hop on the route, B1 port 10. X

then sends an OPEN block with this global address to port 10 on B1. In the OPEN block, X places the reply port 20.

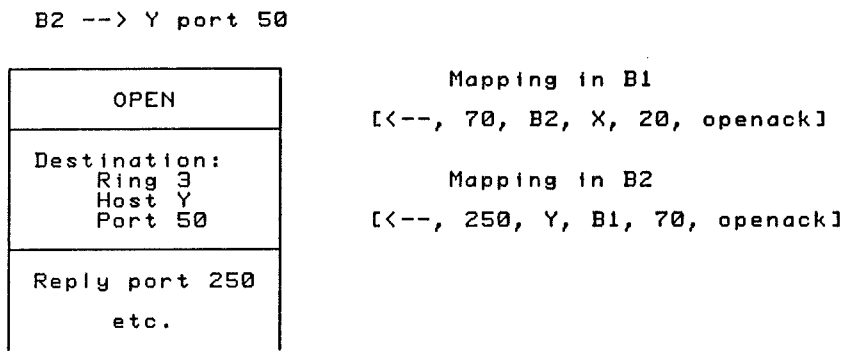

Figure 5.2(b) Virtual Circuit Setup

As shown in figure 5.2a, B1 sets up a port, 70, for the returning OPENACK block and establishes a mapping from this port to port 20 on X. The bridge then replaces the reply port in the OPEN block and sends the block on to the port on B2 which handles OPEN blocks, again port 10. B2 carries on a process identical to that performed by B1, except that its routing tables indicate that the block can be forwarded to the destination, Y (figure 5.2b).

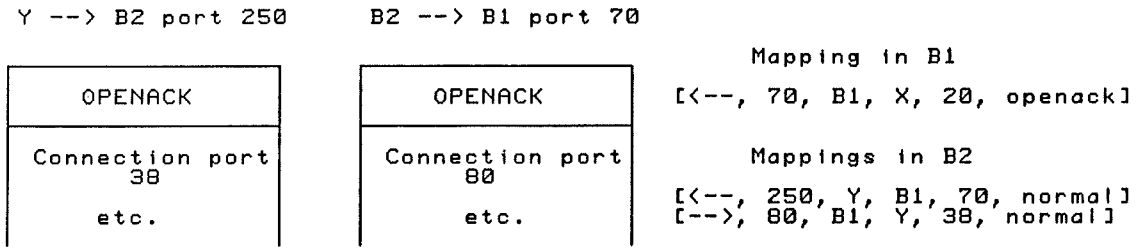

Figure 5.2(c) Virtual Circuit Setup

Y then replies to port 250 on B2 (the reply port in the block it received) with the connection port 38. B2, which realizes that an OPENACK is expected on port 250, allocates the other port of the pair, 80, to map onto Y's connection port. The connection port in the OPENACK block is overwritten by this new port and the block is forwarded to port 70 on B1 as indicated by the mapping for port 250 (figure 5.2c). Again an identical process takes place on B1 so that an OPENACK arrives at X on port 20, and the circuit is complete

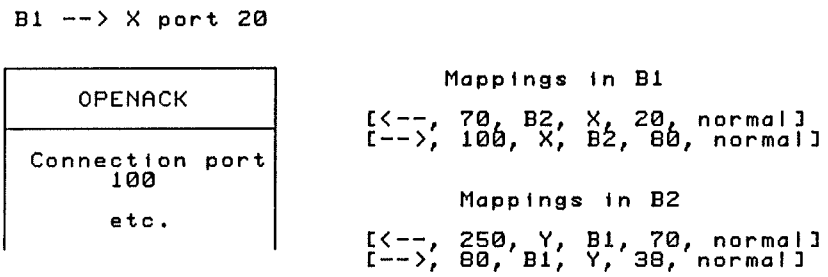

Figure 5.2(d) Virtual Circuit Setup

(figure 5.2d).

There must be a mechanism to close down virtual circuits, that is, to delete the mappings from the bridges. This might be done by an explicit close block being sent down the stream, by a separate interaction of host with the first bridge on the route, by timeout by the bridges through inactivity on a port, or by bridges recycling ports on a least recently used basis.

The changes to host software required are again the handling of a larger address space and dealing with the first hop. However, re-sequencing of blocks would be avoided.

5.2.4 Lightweight Virtual Circuits with Nameserver Path Setup

An alternative to sending a global address in the OPEN block is for a path to be set up by the nameserver as a side effect of the name lookup procedure. Thus in the above example, when host X looks up the address of the service on Y, the nameserver would carry on a transaction with B1, which would in turn carry on a transaction with B2 to produce the mappings shown in figure 5.3. The nameserver would then respond to X telling it that the address of the service was B1, port 32. X would then send an OPEN block to B1 port 50 and a process similar to that shown in figure 5.2 would take place.

As with the closing down of virtual circuits, there must be a way of deleting the mapping in the bridge set up by the nameserver. This could be done as soon as an OPEN block was received on the port, or by timeout through

```
Mapping in B1
[-->, 32, X, B2, 22, open]

Mapping in B2
[-->, 22, B1, Y, 50, open]
```

Figure 5.3 Mappings to a Public Port

inactivity. The timeout approach allows hosts to cache the address of services, since the validity of ports on the bridge will persist for some time. If the mapping was deleted through timeout it would not be necessary to restrict the source in the port mappings for OPEN blocks, since these are mappings to public ports.

This scheme removes the need for hosts to cope with the new address space or to bind an address to a route, since the nameserver is binding a service to a route. On the single ring this was considered to be a binding from a service to an address. What were once service addresses, ie. station and port numbers, are now routes; network addresses are now invisible to host software. This has some repercussions as will be seen in chapter seven.

5.2.5 Discussion

A comparison of the schemes in terms of the binding of the address space format is interesting. The minipacket bridge approach binds this decision into the hardware. Globally addressed datagrams and virtual circuits without nameserver path setup bind this decision into host software. Virtual circuits with nameserver path setup bind the address space format into nameservers and bridges only. To illustrate the importance of this, when the choice between these schemes was being made, a global address consisted of a ring, station and port number. However, in Project Universe, described in the next chapter, global addresses also include an eight bit site number.

The scheme which was chosen is the virtual circuit with nameserver setup. The most important factor in choosing this scheme over the others was that it involved the smallest changes to host software. In the light of the change in

global address structure, this has proved to be a fortunate choice.

The adopted scheme allows hosts to operate exactly as they do an a single ring, provided they use the standard protocols outlined in chapter 4. This contrasts sharply with the PUP approach, where communication on a single ETHERNET is carried on in PUP's, adding extra overhead which can prove to be significant.

As with transport level gateways, virtual circuit bridges hide the world beyond from the hosts. There is in fact no need for the protocols used between bridges to be identical to those between hosts. An example of a different protocol being used between bridges is the use of BRIDGEOPEN and BRIDGESSP blocks as described in the next chapter.

Bridges do not perform error or flow control in any of the schemes outlined above. This is for several reasons. One of the commonly used protocols, the single shot, defines no error or flow control. Also, as mentioned in chapter 3, provision of error and flow control functions in bridges would limit the kinds of traffic which could pass through the bridge, unless different mechanisms were provided for each type of traffic. Placing error control in the middle of a network also has the unfortunate effect of requiring the end hosts to engage in two types of error control. Not only must they ensure that data is correctly received at the next bridge on a route; they must also ensure that the other end has received the data correctly. While error control on a hop-by-hop basis is optional, it is necessary on an end-to-end basis.

The only advantages offered by hop-to-hop error and flow control are the isolation of hosts from retransmissions over error prone links and possibly the conversion between units of flow control and/or between windowing mechanisms. Neither of these is relevent to the low delays and low error rates of a multiple ring system.

5.3 Performance Issues and Implementation

As was mentioned above, if the delay through a bridge can be kept on the order of a few ring delays, basic block throughput between two rings can approach the throughput of the larger, and therefore slower, ring. Thus a desirable property of a ring-ring bridge is that it should be able to transmit a basic block as it is receiving it. This has some immediate consequences for the hardware required for a ring-ring bridge. The hardware responsible for reception on one ring must be tightly coupled to the hardware responsible for transmission on the other, in that reception of a minipacket on one ring must be detected within tens of microseconds by the transmission process on the other ring.

It is normal for hosts to select one transmitter while they are receiving a basic block. If a bridge were to select one transmitter during reception of a basic block, then other hosts would be locked out. This happens in communication on a single ring, but this is an end-to-end effect. The selection of one source by a bridge causes a bottleneck in the middle of the network. In particular, a slow transmitter can lock out a fast transmitter for a considerable length of time. There is a completely analogous effect in transmission of basic blocks. Thus it is desirable for bridges to multiplex both the reception and transmission of basic blocks.

In order to implement this multiplexing, some modifications to the ring stations used by the bridge are necessary. For reception, a table of response bits with an entry for each source station would be necessary. (Simply to accept all packets is a possible strategy, but some low level information would be lost.) A normal ring station can receive faster than it can transmit since it can receive all minipackets addressed to it, while in an n-slotted ring, it can only transmit once in every n + 1 slots. This could be overcome by altering the ring hardware to allow a station to have more than one packet circulating. However, the primary goal of multiplexing the reception and transmission of basic blocks is not to increase bridge throughput, but to prevent a slow transmitter or receiver from tying up the bridge. Thus it would quite reasonable for a bridge to give a busy response to all source stations when it found itself unable to transmit at the rate at which it was receiving.

The ring-ring bridge designed by the author consists of a Motorola MC68000 system with two ring interfaces based on the Motorola 6809 microprocessors. These ring interfaces communicate with the central MC68000 system through direct memory access (DMA) channels.

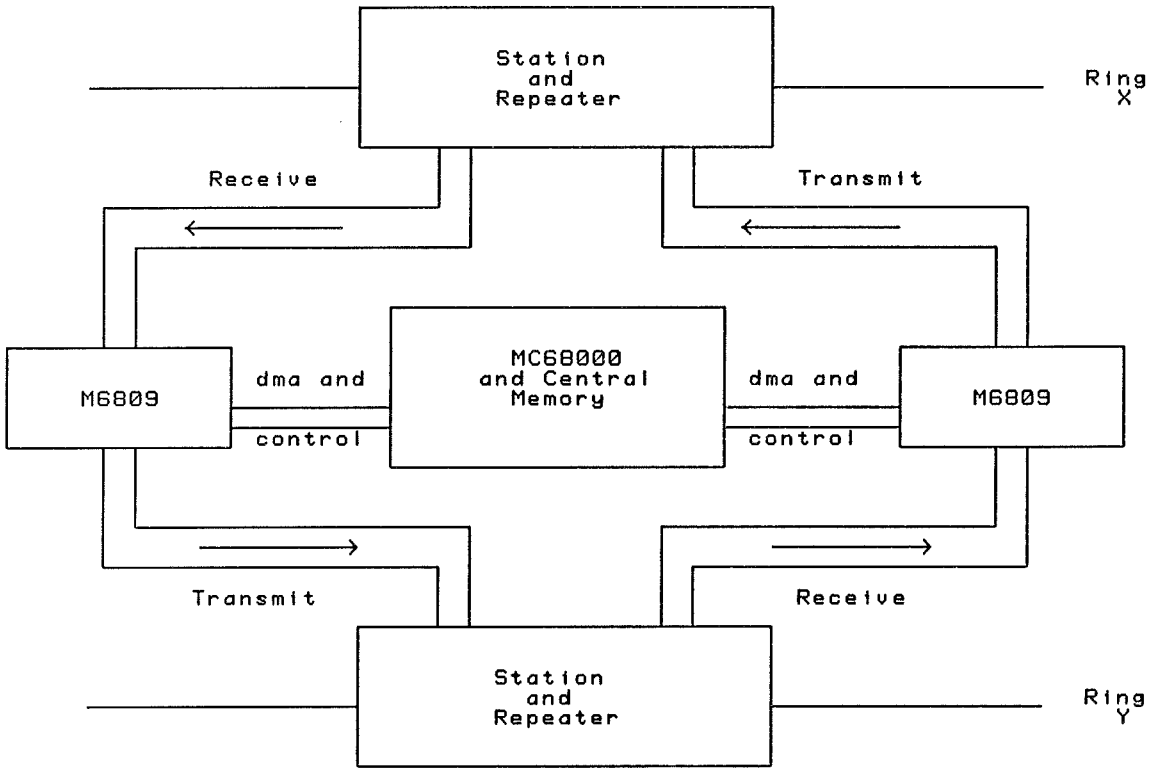

Figure 5.4 Ring-Ring Bridge Configuration

Transmission and reception operate completely independently in a ring station. The ring interfaces, although designed for hosts with only one ring station, cross-couple the rings to which the bridge is attached so that each receives on one ring and transmits on the other as shown in figure 5.4. Thus, as illustrated below, transmitting a block while it is being received is a relatively simple matter. The central MC68000 runs a TRIPOS [Richards 79] operating system. There is a task for each of the ring interfaces, the bridge halves. Port allocations and mappings are handled by these tasks as is the routing of blocks. The interfaces use the memory of the central machine as buffer space, and all traffic goes in and out of central memory. The interfaces do not multiplex blocks on reception or transmission, and queueing

policy for blocks is first in, first out.

Some blocks, such as OPEN's, OPENACK's and SSPREQ's, cause ports to be allocated and mappings to be set up. As well, port numbers inside these blocks must be altered. Other blocks require no such processing and can be dealt with by the central machine simply on the basis of their port number. The state of a mapping indicates which type of block is expected on any port. The interfaces use this information to decide when to notify the central machine of the arrival of a block. This can either be when the port number of the block is received or when the entire block has been received, as is the case for OPEN, OPENACK, and SSPREQ blocks.

In order to illustrate the interaction of the central machine with the ring interfaces, the steps involved in receiving and transmitting a block are described below. Because of the arrangement of the hardware, this will only involve one of the ring interfaces. The block in the example is travelling down a path which has already been set up. In order to show the task switches involved, it is assumed that the bridge is otherwise idle, and that the queue in question is empty.

The ring interface will receive a header minipacket followed by one containing the port number. The interface reads from the central memory to determine the status of the port. If the port was not allocated, it would reject the block by going unselected ie. writing zero into its source selection register. In this example the port is allocated, and the status indicates that the central processor may be interrupted after the port minipacket is received.

The interface writes the header and port into the queue in central memory and interrupts the processor. The interrupt is fielded by a TRIPOS device handler which sends a TRIPOS packet to the appropriate bridge half (one task switch). The interface continues to receive the block while the bridge half determines the next hop for the block, and resets the inactivity timer for the port. The bridge half then sends a TRIPOS packet to the device handler indicating that there is now a block for the interface to transmit (second task switch). The device handler interrupts the ring interface, passing this message on. The ring interface now begins to transmit the block on the next ring, possibly while it is still being received.

Some measurements of the performance of these ring-to-ring bridges are given in chapter seven.

CHAPTER 6

THE UNIVERSE NETWORK

Project Universe [Kirstein 82] is a research project which uses a network made of several Cambridge rings linked together through a high speed satellite link. The Cambridge rings are located at several institutions in the U.K.: British Telecom Research Laboratories, Martlesham Heath; Cambridge University; GEC Marconi Research Laboratory, Chelmsford; Logica Ltd, London; Loughborough University of Technology; the Rutherford Appleton Laboratory, Chilton, and University College, London. The single ring at Logica is connected to University College through a 1.2 Km. high speed terrestrial link. The remaining sites all have earth stations which receive from and transmit to the satellite.

In this chapter a discussion of the underlying network will be presented. This includes a description of the characteristics of the satellite link as well as the network protocols.

The ring-ring bridges described in the previous chapter are part of the Universe system. Some of the decisions made in Universe have thus affected their functional behaviour. The term **bridge** in this chapter refers to the satellite bridges, ring-ring bridges and the bridges used in the Logica-UCL link.

Universe is a co-operative venture; the work described in this chapter is the result of the efforts of a group of people. The main contributions to the design of the network were the work described in the previous chapter as presented in [Leslie 82a], and an independent but similar scheme outlined by Adams, Burren, Cooper, and Girard [Adams 82a] of the Rutherford Appleton Laboratory. Refinements to the scheme were made by the Protocols Working Group of the project. The membership of this group has changed throughout the course of the project, but it is worth noting the more permanent members who have contributed to the current design. Beside the author, these include C. Adams, G. Adams and G. Waters of the Rutherford Appleton Laboratory,

S. Wilbur and C. Kennington of University College London, P. Kirk of Marconi Research Laboratories, W. Lees of Logica Limited, and G. Morrow of British Telecom Research Laboratories. Nick Ody at the Computer Laboratory in Cambridge also contributed to the design.

Although it is difficult to accurately indicate one's own contribution to a co-operative project, ideas which one had little part in are easier to identify. The mechanism for call set up, described below, in which bridges use global addresses is based on an idea put forward by G. Adams.

6.1 Satellite Link

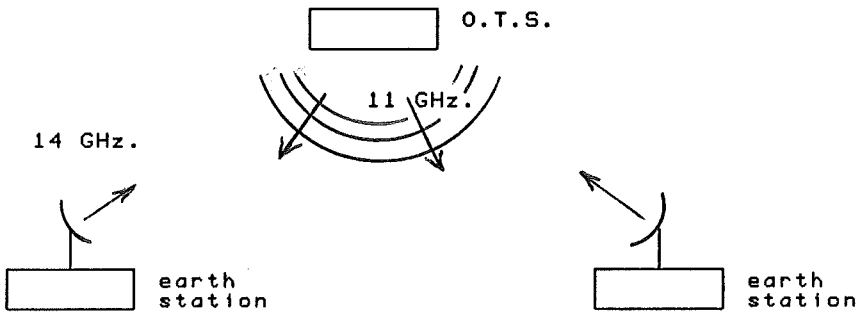

Figure 6.1 Satellite Channel Configuration

The satellite used in the Universe Project is the Orbital Test Satellite (OTS), a geostationary satellite situated over the Gabon. The link consists of two 2 Mbit/s half channels centred at 11 and 14 GHz. The earth stations transmit at 14 GHz. and receive at 11 GHz. Conversely, the module on OTS receives at 14 GHz. and transmits at 11 GHz. as shown in figure 6.1.

The error rate on the total channel is one in 10^6 bits [MCSL 76]. In order to reduce this, a forward error correcting technique is used. This technique employs half rate encoding resulting in a 1 Mbit/s channel with an error rate measured at less than one in 10^9 bits [Adams 82b].

Access to the channel is controlled by a master station as described in [Waters 82]. Put simply, the channel time is divided by the master into 135 ms. frames. The remaining earth stations make requests to the master using small, permanently reserved slots within each frame. The remainder of the frame is allocated dynamically by the master station in response to requests from the slave stations. Stations are only allowed one dynamically allocated slot within a frame. Thus traffic arriving at an earth station which is to be transmitted over the satellite link will encounter an average delay of approximately one half the frame size before being transmitted, assuming the channel is idle. The propagation delay over the link is .25 seconds, so that the delay on an idle channel, excluding processing time, will be approximately uniformly distributed from .25 seconds to .385 seconds.

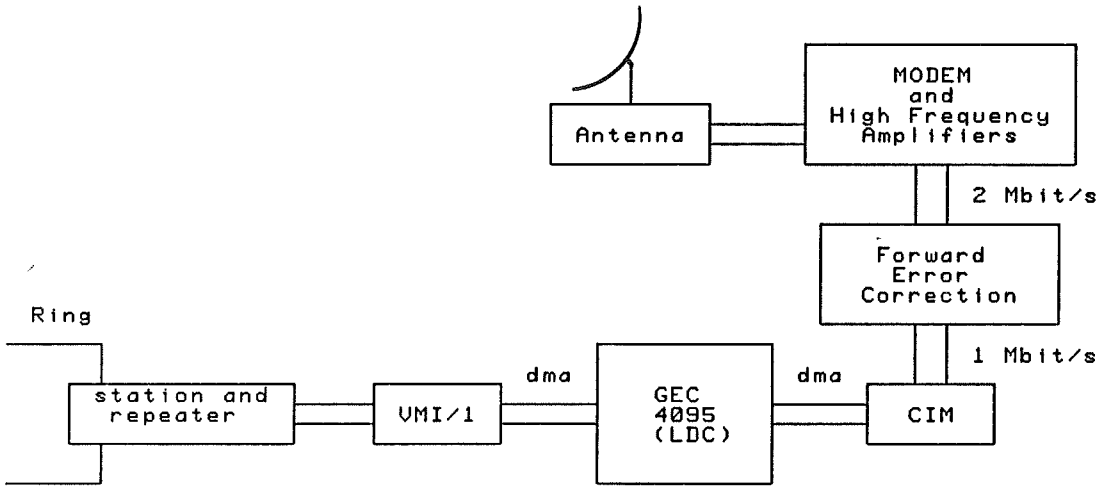

Figure 6.2 Satellite Bridge Configuration

The hardware used in a Universe earth station is shown in figure 6.2. The central control is provided by a GEC 4065 computer which is referred to as the link driving computer (LDC). An LDC is attached to a ring through a Logica VMI [Logica 81] ring interface, and to the satellite channel through an interface based on the Motorola 6800 called a Computer Interface Module (CIM) [Olofsson 80]. The CIM is responsible for frame and slot synchronization, and the actual transmission and reception from or into the LDC's memory. Similarly, the VMI interface receives and transmits basic blocks in and out of the LDC's memory. The entire system is referred to as a satellite bridge.

6.2 Network Overview

The bandwidth and error rate of the satellite link are comparable with those of a single Cambridge ring. The delays across these systems are, however, orders of magnitude apart. This has immediate consequences for applications to be run, and protocols to be used, on a network combining the two types of system.

It is possible to find protocols which are satisfactory on a single ring, but which will perform badly over the satellite channel. Others, however, will run well over the satellite link. The byte stream protocol, with a window size of one block, will clearly perform badly. On the other hand, the fileserver read protocol [Dion 81] in which a client sends a request to the fileserver which in turn responds with a number of blocks (possibly containing several megabytes of data), works as well as any data transfer over the satellite link possibly could.

The performance of protocols must be put into perspective in terms of the applications which they support. For example, terminal traffic which is line-oriented, that is to say a block is sent when a line is input or output, will not be degraded significantly by using the byte stream protocol over the satellite link. This is because a full second is a tolerable response time in most interactive systems. However, in character oriented terminal traffic, where each block contains only a single character, the degradation will be significant. A file transfer using the byte stream protocol will be limited to a throughput of 2 Kbytes of data every .75 seconds ie. the average delay across the satellite link for a block and its returning acknowledgement, or approximately 20 Kbits/s.

Protocol conversion could solve this problem of mismatch between some of the single ring protocols and the delay on the satellite link. The conversion would be between window-of-one protocols used on a ring, or collection of local rings, to a multiple block window protocol used on the satellite link. This could be achieved by using the satellite bridges to perform a flow control conversion, or by providing other machines at each site to perform this conversion. These two alternatives are shown in figure 6.3.

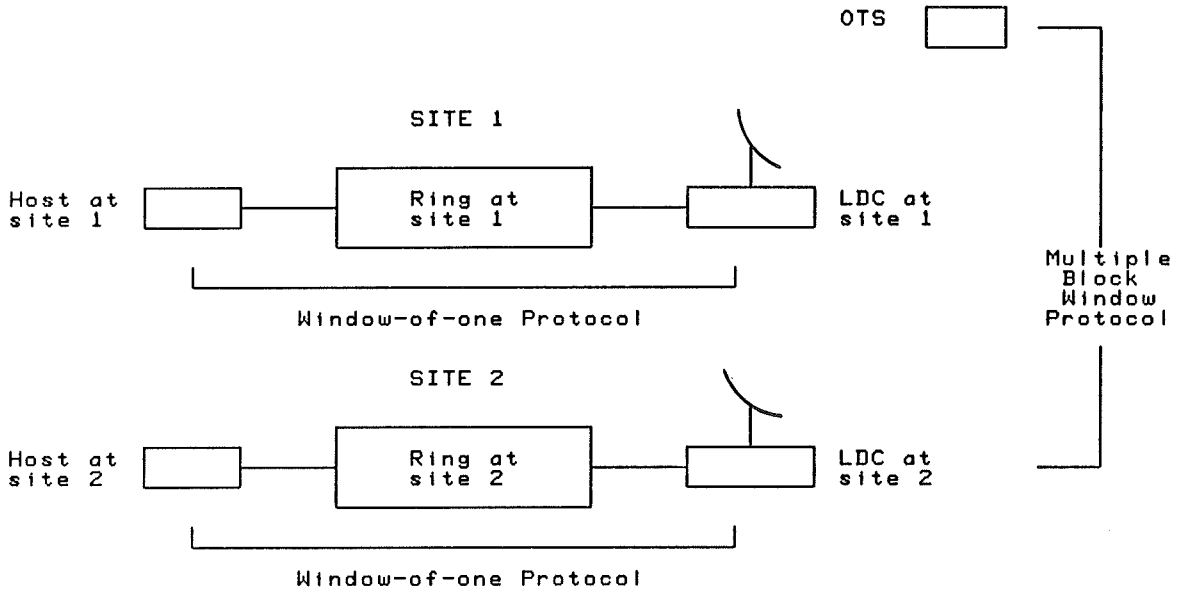

Figure 6.3a Flow Control Conversion in LDC

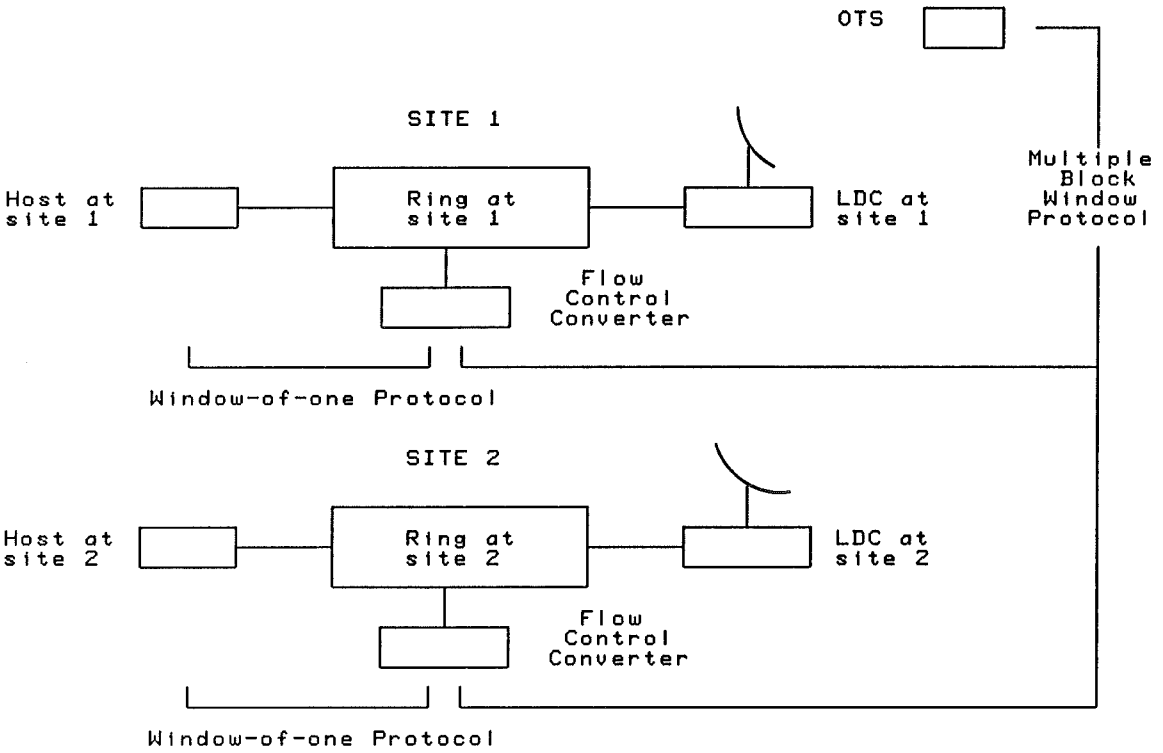

Figure 6.3b Separate Transport Level Gateways

Another approach to using a flow control conversion is to modify some of the protocols which were used on a single ring, such as a proposed modification to BSP which will add a multiple block window mechanism. The modified protocol allows a several blocks to be in transit simultaneously. The effect of window size on throughput over the satellite link depends upon the satellite channel allocation algorithm. Throughput is also directly related to the data block size used. If a window size of n blocks is used and n blocks can be transmitted in the link delay time, in this case two frames, then the throughput limit will be n times higher than that of normal BSP, assuming the same data block size is used. Increasing n beyond the number of blocks which will be sent in two slots on the satellite channel will not increase the throughput limit further.

Connecting rings through a satellite using a scheme similar to that outlined in the previous chapter allows protocols which are used on a single ring to be used over the satellite link, although some may perform badly. It also allows the protocol conversion approach of figure 6.3b to be used, as well as the modification of end-to-end protocols, such as the extension to BSP. This is in fact the initial approach taken in the Universe network.

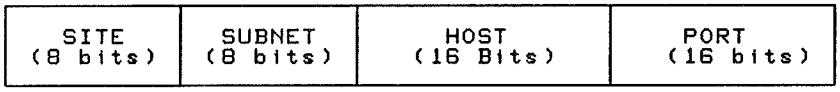

Figure 6.4 Universe Address Format

The format of a Universe address is shown in figure 6.4. A site corresponds to a satellite bridge, so that Logica and University College are both in the London site. A subnet may be thought of as a Cambridge ring, although other local network technologies are not ruled out. The host field is sixteen rather than eight bits to allow for local networks which have host addresses of sixteen bits.

The address format need only be known by nameservers and bridges on the network. However, a new type of block, the datagram, which contains full network addresses is supported. The format of a Universe datagram is shown

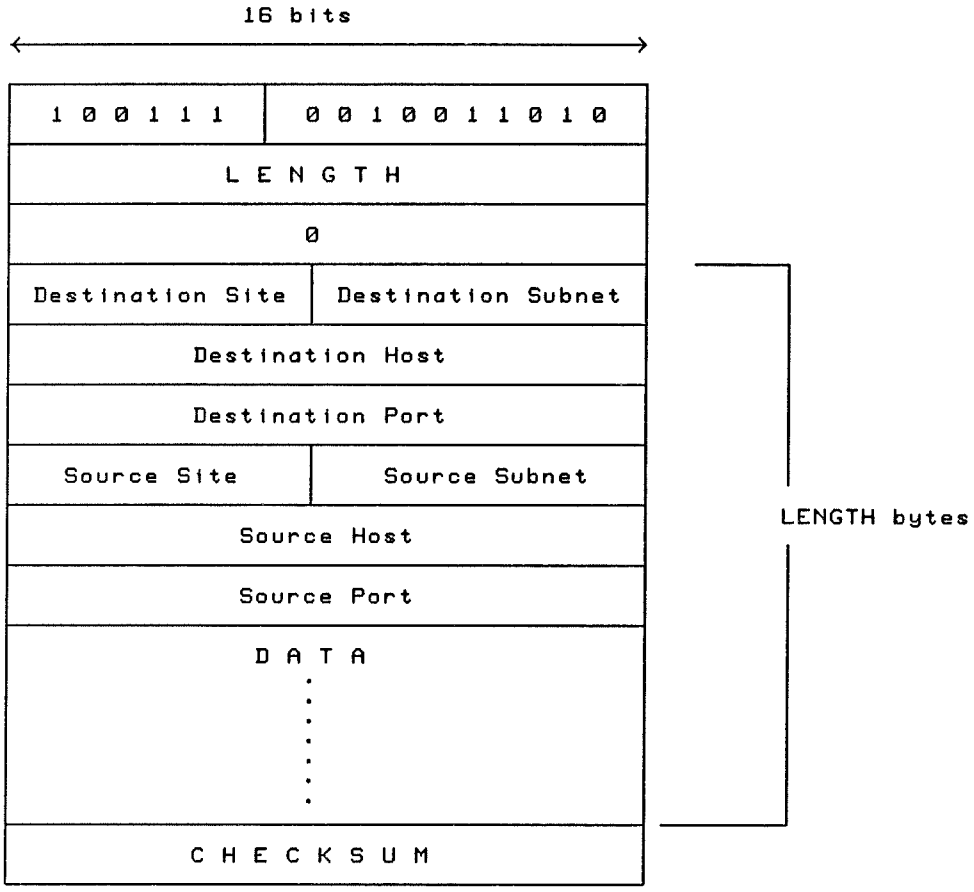

Figure 6.5 Universe Datagram Format

in figure 6.5.

Another type of block, the so called Joint Network Team (JNT) block [JNT 82] is also used on the network. This block is functionally equivalent to the basic block outlined in chapter 4, and need not be discussed separately.

6.3 Network Paths

The previous chapter described virtual circuits in terms of port mappings stored in bridges. In the Universe network, a new procedure has been adopted for setting up paths to public ports, and the timeout of paths has been defined explicitly [Adams 82c].

6.3.1 Path Set Up

Path set up is still performed by an interaction between a Universe nameserver [Leslie 82b] and a bridge. However, no further interaction with other bridges takes place, mainly to avoid carrying on a set up transaction over the satellite link. Instead, the mapping stored in the first bridge on the route is a mapping to a full network address. When an OPEN block is received on a port corresponding to such a mapping, the block is transformed into a BRIDGEOPEN block whose format is shown in figure 6.6a.

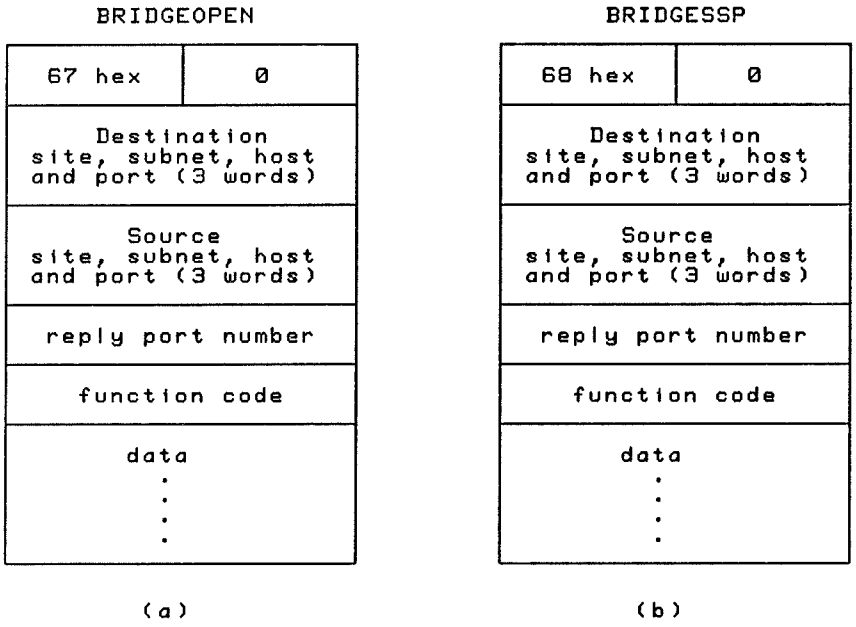

Figure 6.6 BRIDGEOPEN and BRIDGESSP Formats

BRIDGEOPEN blocks are passed among bridges which use the network address in the block to route it towards its destination. A reply path is set up as described in chapter 5, and the reply port in a BRIDGEOPEN block is altered as it passes from bridge to bridge. The last bridge on the route transforms a BRIDGEOPEN block back to an OPEN block which is then delivered to the destination. Thus only bridges are aware of the existence of BRIDGEOPEN's. There is a similar arrangement for delivering SSPREQ blocks, using BRIDGESSP blocks, to public ports. The format of a BRIDGESSP is shown in figure 6.6b.

6.3.2 Path Deletion

The policy for deleting paths is based on port inactivity; ports are not closed down explicitly by the end hosts. This means that bridges do not have to look at the contents of blocks once a connection is established. Using a policy of timing out ports means a large number of ports may be allocated unnecessarily in the bridge. Since the information stored per port is small, a few words in the case of the ring-ring bridges, this presents little problem to the bridges. It is possible for a host to erroneously create ports continuously, at a faster rate than they are being timed out. This is a problem no matter which policy is used for port deletion. Bridges may protect themselves from such a situation by limiting the number of connections which they will accept from any given host at a time, not including bridges.

The policy for path deletion is as follows. A bridge keeps an inactivity timer on each path. When the timer for a path expires the bridge is entitled to delete that path. The timeout to which a port is subject depends upon the status of the port. These timeouts are described below and the values given by the initial Universe specification [Adams 82c] are shown in figure 6.7.

Tn	public path	120 seconds
Ts	initial reply	20 seconds
Tr	multiple SSP	2 seconds
To	OPENed path	120 seconds

Figure 6.7 Path Deletion Timeouts

Paths to public ports set up by the nameserver may be deleted if no valid traffic is received for a time Tn. These paths are open to traffic from any source since they lead to public ports, and a bridge may respond to a nameserver request with a port number which is already allocated, provided of course, that it leads to the appropriate destination port. This means that many clients on one ring can use the same public route to a public port, so that in the case of a heavily used service, such as a fileserver, the port will tend to remain active. Any request from a nameserver to set up a path will also reset the timer on the allocated port to Tn.

SSPREQ, OPEN, BRIDGESSP and BRIDGEOPEN blocks cause reverse paths to be set up. A reverse path may be deleted in a time Ts if no block returns down the path. Once a block returns, the timeout on the reverse path changes as follows. If the path was set up by an SSPREQ or BRIDGESSP block, the timeout is set to a time Tr, and is reset to Tr when any traffic appears. This is to allow a group of blocks to be returned in quick succession, as in a fileserver read operation. Tr is thus much smaller than Ts.

If the reverse path was set up by an OPEN or BRIDGEOPEN block, the first returning block must be an OPENACK, otherwise it is ignored. The return code in the OPENACK is inspected, and if nonzero causes the path to be deleted. If the return code is zero, then a forward path is created, as described in chapter 5. The timeouts on both the forward and reverse paths are set to a time To, which is much larger than Ts, as is shown in figure 6.7. Traffic received on one path only resets the timer for that path; there is no coupling of paths within a bridge.

6.3.3 Path Tracing

An additional function performed by Universe bridges is the tracing of paths. On a single ring, the recipient of an OPEN or SSPREQ block can determine the source of the block simply from the ring source address. On a multiple ring system, this is no longer the case. Knowing the source of a request can be useful for logging purposes and, for example, in determining where a user's terminal is located.

On the single ring system, a service called **reverse lookup** was offered by the nameserver. The only argument to this SSP transaction was a station number. The reply to the request was a string giving the name of the corresponding host. Universe nameservers offer a similar service, called **reverse trace** which takes two arguments: the address of the host on the local ring which sent the OPEN or SSP, and the reply port from the block. If the host is not a bridge, the operation is identical to reverse lookup. Otherwise, the nameserver will ask the bridge to trace the given path and return the network address of the host to which the path leads. This may require the bridge to make a similar request of another bridge on the path. The mechanism for setting up a

reverse path for the reply to the trace is identical to that for setting up a reverse path for an SSPRPLY.

6.4 Names and Naming Domains

The simple scheme where each local ring has a nameserver in which all network service names are stored is strained by a network as large as Universe. The large network size introduces a problem both in storing all the names in all nameservers and in updating the contents of all nameservers. This is an example of the general distributed data base problem. The mechanisms for dealing with the problem, for example as presented in [Gifford 79] are too complex for the simplicity required by vital network components such as the nameservers.

This problem is solved to a certain extent in Universe by the introduction of naming domains. The Universe name space is broken into several subspaces, each called a naming domain. A fully qualified Universe name consists of a domain name such as 'CAMB', a delimiter (the character '*') and the name within the domain such as 'PRINT'. Neither the domain name nor the name within the domain may contain the character '*'. Each subnet belongs to one domain, its home domain, although a domain may contain several subnets, possibly at several sites.

A nameserver, which stores only the names in its home domain, will treat unqualified names, such as 'PRINT', as name from its home domain. Thus in the domain 'CAMB', the names 'CAMB*PRINT' and 'PRINT' are equivalent. If presented with a name from another domain, a nameserver will perform a remote lookup by sending a request to an appropriate nameserver. Nameservers may cache addresses of names in other domains, but must always perform a remote lookup after responding to a client request for a cached name. This means that if a cached address is incorrect, the next request by the client will be answered with the correct response.

Site	Institution	Naming Domain
Baddow	Marconi Research Centre	MRC
Cambridge	Cambridge University Computer Laboratory	CAMB
Chilton	Rutherford Appleton Laboratory	RAL
London	Logica Limited	LOGICA
London	University College	UCLCS
Loughborough	Loughborugh University of Technology	LUT
Martlesham Heath	British Telecom Research Laboratories	BTRL

Figure 6.8 Universe Naming Domains

Naming domains reduce the number of names each nameserver must store, but perhaps more importantly, they allow each domain to be controlled by separate naming authorities which can specify their own policy for the addition and deletion of names. It is not surprising therefore, that the domains in Universe have so far been associated with institutions as shown in figure 6.8. Each naming authority controls a master copy of its name table which nameservers in the domain read at startup, or when told to reload.

CHAPTER 7

RESULTS AND MEASUREMENTS

This chapter presents some observations about the introduction of a ring-ring bridge into the Cambridge Computer Laboratory, about ring-ring bridge performance, and about early Universe Network experience. The splitting of the ring at Cambridge has resulted in some changes to host software. These changes will be outlined first.

7.1 Protocols and Software Changes

The connection strategy described in the previous two chapters relies on clients to use standard SSP or OPEN connections. Any hosts which used other sorts of protocols have had to be modified, or have agents on each ring to act for them.

An example of this is the loading of stand-alone processors on the ring. Each type of processor has associated with it an **ancilla** service which is responsible for dealing with the loading properties of the machine in question. The loading protocol is neither SSP nor OPEN based, so that an ancilla service must be on the same ring as a machine it is loading. Similarly, for the debugging of small servers, a private protocol was developed. An agent called a **gyp** has been developed which will use this private protocol to communicate with small servers on its local ring, and use standard protocols to communicate with any client wishing to debug a small server. These are not major upheavals, both the ancilla and gyp are themselves microprocessor based small servers.

Another change involved the use of a feature in the Byte Stream Protocol known as REPLUG. REPLUG allowed a host with two byte streams open to plug one stream into the other simply by passing the appropriate station addresses to

the parties at the other end of each byte stream. This does not work through bridges. REPLUG was used in connecting a terminal session to a processor allocated to it by a resource management service. The terminal session would connect to a session manager through a byte stream. The session manager would then be placed in contact with an allocated machine, again through a byte stream. The session manager would then REPLUG the two streams together, connecting terminal to processor. This is now accomplished at a higher level in the protocol structure. A service name, rather than an address, is now passed to the end which re-establishes the connection.

Another protocol which would not work across bridges was the fileserver protocol [Dion 81]. This is because it made use of multiple ports in one connection. These ports were used to distinguish data from control information. This protocol has been modified to use **subports** of one port. Subports are distinguished from one another by using the top bits of the word containing the port number, which occupies only the lower twelve bits, as shown in figure 7.1. Bridges ignore these top bits.

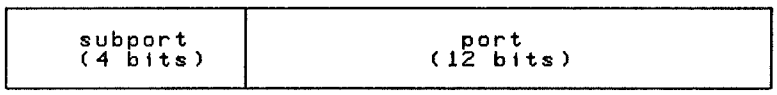

Figure 7.1 Subport and Port Fields

A more widespread change has been the move from reverse lookup to reverse trace. Reverse lookup was used extensively on the single ring to map addresses into host names. In the multiple ring network, it is necessary to map paths into host names. This is the function performed by reverse trace. Although this is a relatively small change, it had to be performed in nearly all hosts on the ring.

A related change is the invisibility of network addresses to hosts. In the CAP [Wilkes 79b] operating system, terminal streams were represented by an address of the station to which the terminal was connected and a terminal number. However, reverse trace does not bind a route to an address; it binds a route to a host. Thus it was necessary for the operating system to be

modified to represent terminal sessions as strings rather than numbers.

These protocol modifications can be thought of either as inconveniences caused by the appearance of bridges, or as a standardization procedure. The fact that modifications were necessary could indicate that originally these protocols were implemented incorrectly. An example which supports this view is in the debugging of small servers. The CAP operating system prevents users from performing arbitrary ring operations necessary in an ad-hoc protocol such as small server debugging. The introduction of the gyp service means that small server debugging will now be possible on CAP.

7.2 Ring-Ring Bridge Performance

Experiment	Hosts	Single Ring	Double Ring
1. Name Lookup (path set up)	MC68000, Z80	10 ms	17 ms
2. SSP Transaction (Request 20 bytes)	MC68000, Z80	10 ms	17 ms
3. SSP Transaction (Request 200 bytes)	MC68000, Z80	14 ms	24 ms
4. Byte Stream (200 byte blocks)	MC68000, MC68000	95 Kbits/s	70 Kbits/s
5. Byte Stream (2000 byte blocks)	MC68000, MC68000	295 Kbits/s	235 Kbits/s
6. Fileserver Read (2000 bytes)	MC68000, LSI4/30	110 ms	130 ms
7. Fileserver Read (20000 bytes)	MC68000, LSI4/30	550 ms	710 ms

Figure 7.2 Ring - Ring Bridge Performance

Figure 7.2 shows some performance measurements for ring-ring bridges. For each activity, a comparison is made between performance between two hosts on a single ring and performance between two hosts on two rings joined by a bridge. The exception to this is the first entry which compares nameserver

lookup of a local name with nameserver lookup of a name which causes a path setup. The difference of 7 ms. is an indication of the nameserver-bridge interaction time. The hosts in the experiments are either 6 Mz. Z80 systems (the nameservers), MC68000 systems similar to the ring-ring bridge with the same ring interface hardware but a different interface program, or the Cambridge fileserver which runs on a Computer Automation LSI4/30 with a high speed ring interface, described in [Gibbons 80], based on the Signetics 8X300 microinterpreter. The MC68000 systems and the fileserver all run the TRIPOS operating system.

The second and third entries show the difference between performing an SSP to a local and remote host, for request block sizes of 20 and 200 words respectively. In both cases the reply blocks are 6 bytes, since the SSPREQ's are all in fact to a trivial service on the nameservers on each ring.

The fourth and fifth entries show the performance of byte streams. The data in these byte streams are generated artificially; there is no application layer interaction, so the streams have higher data rates than one would usually expect. In the fourth entry, the bytes streams use block sizes of 200 bytes, for the fifth entry, blocks of 2000 bytes are used.

The sixth and seventh entries compare delays for reading from the fileserver. The response times here are measured from the application level process which would normally generate such requests. The sixth entry is a read of 2000 bytes, the seventh is a read of 20000 bytes. The figures for these times are only accurate to the nearest 20ms.

The performance measurements show that the overheads incurred by transactions through an idle bridge are significant, but not so high as to require a different approach in host protocols. Byte streams and bulk data movement from the fileserver are the least affected. Indeed, in lower data rate byte streams, such as terminal traffic, there is no perceptible difference to the user. This is true even of single character interactions which occur in screen editors.

As is the case in any performance measurement exercise, certain deficiencies in the implementation were highlighted. Some of these were corrected; others remain and may be corrected in the future. The bridge itself provides a very useful tool for recording traffic patterns. These patterns will be used in deciding which improvements are worthwhile.

Tests have shown that the standard M6809 ring interface is limited to a transmission data rate of 400 Kbits/s. As well as the raw data rate limitation, there is some overhead incurred for each block which passes through the bridge. Observing bus request and interrupt signals has shown that most of this overhead is processing in the interface and interface-host interactions. In particular, for the path setup time of 7 ms., less than 2 ms. is spent in the MC68000 outside device drivers. Some of this remaining 5 ms. is expected to be reclaimed by modifying the interface program.

The MC68000 seems quite adequate for the role it plays. The M6809 ring interfaces are, however, far from optimal. Beside the performance problems outlined above, they are not suitable for the multiplexing of basic blocks, i.e. the concurrent reception or transmission of a number of basic blocks. Any purpose-built bridge interface should be designed with this function in mind. The slotted ring shares bandwidth out in a fair manner; a bridge which does not multiplex basic blocks introduces contention into the network.

7.3 Universe Experience

At the time of writing, the Universe network has only been operational for a few months, with only three sites, Cambridge, Rutherford and University College, active throughout this period. The only application experiment which has been performed is an extension of a system in use at the Cambridge Computer Laboratory into the Rutherford Appleton Laboratory.

The system consists of the Cambridge fileserver, a front end to the fileserver known as the TRIPOS Filing Machine[1] (TFM), and a number of single

---

1 described in more detail in [Richardson 83]

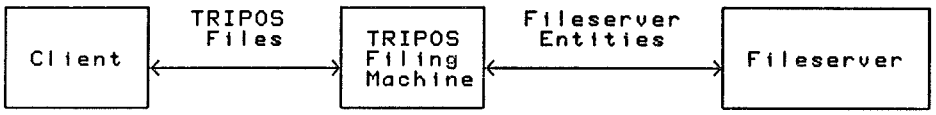

Figure 7.3a Tripos Filing Machine System

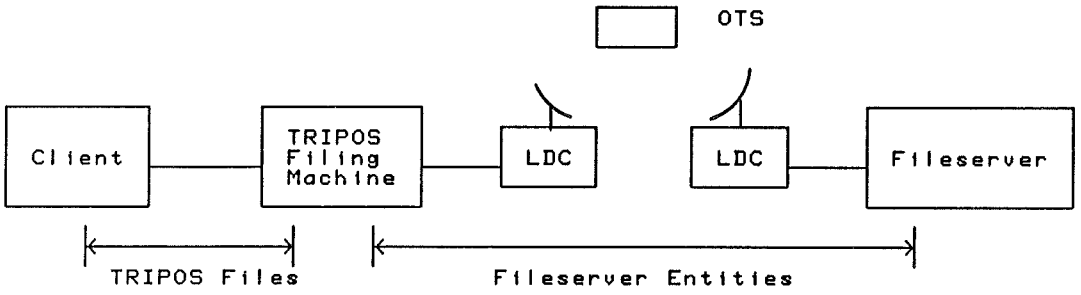

Figure 7.3b Tripos Filing Machine System Split Over Satellite Link

user processors running the TRIPOS operating system. The fileserver does not provide filing systems as such, but the primitives which allow clients to build filing systems. The TRIPOS filing system is one of these. Originally, each TRIPOS client contained the code which implemented the TRIPOS filing system on the fileserver. Now this implementation is done by the TFM, so that TRIPOS clients make requests to the TFM in terms of TRIPOS files, while the TFM deals with the fileserver in terms of fileserver entities, as shown in figure 7.3a. The TFM serves two main purposes. It reduces the load on the fileserver and improves response times to client requests, by caching files, performing read ahead, and using the efficient large block transfers from the fileserver. The other benefit is that the filing system is provided with a protection domain where access controls can be implemented and accounting may be performed.

The performance features make the splitting of the system over a satellite link, with the TFM remote from the fileserver, an interesting proposition. This scheme is shown in figure 7.3b. The filing machine hides, to some extent, the delay of interacting with the fileserver. The policies implemented in the filing machine regarding caching and read ahead were not changed for this experiment.

The early results have been encouraging from a user point of view. Detailed measurements have yet to be performed, but the indications are that it is realistic to extend the distributed system at Cambridge to other sites. The experiment also involved the booting of a number of machines at the Rutherford Appleton Laboratory over the satellite link. The protocols used in this experiment are extremely lightweight; all transactions use either single shots or the fileserver protocol [Dion 81] which is an extended single shot protocol. This has played no small part in allowing the experiment to be performed so quickly, as well as making the experiment successful.

CHAPTER 8

CONCLUSIONS AND FURTHER WORK

Two sets of conclusions can be drawn from the work described in the previous chapters. These are conclusions about what is now the architecture of the Universe network, and conclusions about the distinctions between local area and wide area networks. After a presentation of these conclusions, some further areas for investigation are mentioned, again both with reference to Universe and in a more general context.

8.1 The Universe Architecture

The interconnection scheme described in chapters five and six has many interesting properties besides allowing hosts to use protocols comparable, and in many cases identical, to those used on a local network. The service which Universe bridges provide is almost trivial, beyond allowing hosts on one ring to communicate with hosts on another ring. This service is simply that blocks within a call arrive in the same order as they were sent. The cost of this service is that failure of a bridge in Universe destroys all calls which were routed through that bridge.

This is not the case in datagrams networks where routing is performed on a block by block basis. While in Universe recovery from errors of this sort must be performed on a call basis, datagram networks can recover internally by dynamically rerouting blocks. It is by no means clear that this has a significant effect on network reliability; indeed it is rather dubious to recover from catastrophic errors at lower protocol levels where information, such as the importance of a call, may be absent.

The Universe network binds routes to services. Thus two different services on the same host can be reached by different routes. The difference in the requirements between, say, real time voice interaction and file transfer can be easily catered for by having different preferred routes to each service.

The functions which are notably absent from Universe bridges are error and flow control. This reflects the local network philosophy of Universe, that is the assumption that the medium between two hosts has good error characteristics and low delay, so that flow control and error control can be done on an end-to-end basis, at whatever level in the protocol hierarchy that is appropriate. Error controls must be performed on an end-to-end basis anyway. Adding error and flow controls in the network itself simply adds overhead to communications. The bridge design also does not limit the uses of the network to the applications as perceived when the network was designed. The delay on the satellite does not fit into the local network pattern, but the problems it poses can be solved external to the satellite bridge if desired. If terrestrial links were used rather than a satellite link, these problems would be dramatically reduced.

8.2 Local Area and Wide Area Networks

In the Universe project some of the distinctions between local area and wide area networks have become blurred. Using simple protocols, machines have been loaded and fileservers accessed over the satellite link. The distinctions would become almost unnecessary if high speed terrestrial links were to be used. The technology available when local networks were developed was superior to the technology available when wide area networks were first developed. Local area networks have been able to be take advantage of new transmission technology as it has become available. Similarly, some applications for which local networks are used were not envisaged when wide area networks were first developed. Local networks have been able to adapt to new applications because protocol experiments are much easier to perform than they would be on wide area networks.

Thus the usual distinctions between local area and wide area networks are largely spurious. The differences are historical and administrative. While there is every reason to believe that some of the lessons learned on local networks can be applied to wide area networks of the future, there are reasons to doubt that this will actually happen. Standards in computer networks are extremely important. Computers can only communicate if they use the same set of protocols or if mechanisms exist to convert between the protocols which they are using. Thus there is a drive towards standards, even in local networks. It is quite possible that future standards for both wide area and local area networks will be based on experiences with the technology used in the first wide area networks, rather than on experiences with the current or expected future technology. In the light of even the early results of the work described here, this is unfortunate.

8.3 Further Work

Some possible improvements to ring-ring bridges were mentioned in the previous chapter. These are to improve the performance of the bridges and to allow the bridges to multiplex both the reception and transmission of a number of blocks. The latter of these improvements has a special significance for real time voice which will be impaired by the present variable delays in bridges.

Perhaps a more important area for further work is the development of a long haul terrestrial part of the network using a service such as MEGASTREAM [BT 82] offerred by British Telecom. MEGASTREAM provides a 2 Mbit/s. service with a delay which is made up mainly of the propagation delay between the two ends. The error rates at present differ from line to line; some may require forward error correction techniques, others will not. An important effect of such a development is that alternative routes to services would be available. Although it is relatively simple to give different routes to different services on the same host, mechanisms will have to be added to both bridges and nameservers to provide alternative routes when bridges fail. Routing would still only be performed on a call basis.

In a more general context, the development of new wide area networks which use media such as fibre optics, presents an opportunity to extend local network philosophy to a very large scale. The topologies will have to be more general than those common to local area networks. This will introduce new problems. In particular, the store-and-forward delays of current wide area networks must be avoided when possible.

REFERENCES

[Adams 82a] Adams, G.C, Burren, J.W., Cooper, C.S., and Girard, P.M.
"The Interconnection of Local Area Networks via a Satellite Network".
New Advances in Distributed Computer Systems, pp 201-210,
K.G. Beauchamp (ed), Reidel, 1982.

[Adams 82b] Adams, C.J.
Private communication, December 1982.

[Adams 82c] Adams, G.C., and Leslie, I.M.
"Client Protocols Specification".
Universe Paper 94.2, Rutherford Appleton Laboratory, June 1982.

[Boggs 80] Boggs, D.R., Shoch, J.F., Taft, E.A., and Metcalfe, R.M.
"Pup: An Internetwork Architecture"
IEEE Transactions on Communications, vol COM-28, no 4, April 1980,
pp 612-624.

[BT 82] British Telecom
"Design Requirements fo the Connection of Privately Owned and Maintained Equipment to British Telecom MegaStream 2".
British Telecom, 1982.

[CCITT X.25] CCITT Recommendation
"Interface Between Data Terminal Equipment (DTE) and Data Control Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks".
CCITT Yellow Book, Volume VIII - Fascicle VIII.2, Rec X.25.

[CCITT X.75] CCITT Recommendation
"Terminal and Transit Call Control Procedures and Data Transfer System on International Circuits Between Packet-Switched Data Networks".
CCITT Yellow Book, Volume VIII - Fascicle VIII.3, Rec X.75.

[CCITT X.121] CCITT Recommendation

"International Numbering Plan for Public Data Networks".

CCITT Yellow Book, Volume VIII - Fascicle VIII.3, Rec X.121.

[Cerf 78] Cerf, V.G., and Kirstein, P.T.

"Issues in Packet-Network Interconnection".

Proceedings of the IEEE, vol 66, no 11, Nov 1978, pp 1386-1408.

[Clark 78] Clark, D.D., Pogran, K.T., and Reed, D.P.

"An Introduction to Local Area Networks".

Proceedings of the IEEE, vol 66, no 11, Nov 1978, pp 1497-1517.

[Cohen 80] Cohen, D.

"Flow Control for Real Time Communication".

Computer Communication Review, vol 10, nos 1&2, Jan/April 1980, pp 41-47.

[Dallas 81] Dallas, I.N.

"An X25 Gateway for the Cambridge Ring".

Proceedings of Networkshop 9, 15-17th Sept 1981, Heriot-Watt University,

pp 29-38.

[DCPU 81] The JTP Working Party of the Data Communication Protocols Unit

"A Network Independent Job Transfer and Manipulation Protocol".

Data Communication Protocols Unit, Department of Industry, Sept 1981.

[Dellar 80] Dellar, C.N.R.

"Removing Backing Store Administration from the CAP Operating System".

Operating System Review, vol 14, no 4, Octerber 1980, pp 41-49.

[Dion 80] Dion, J.

"The Cambridge File Server".

Operating Systems Review, vol 14, no 4, October 1980, pp 26-35.

[Dion 81] Dion, J.

"Reliable Storage on a Local Network".

Ph.D. dissertation, Cambridge University, Feb 1981.

[Folts 80] Folts, H.C.

"X.25 Transaction-Oriented Features - Datagram and Fast Select".

IEEE Transactions on Communications, vol COM-28, no 4, April 1980,

pp 496-500.

[Gibbons 80] Gibbons, J.J.

"The Design of Interfaces for the Cambridge Ring".

Ph.D. dissertation, University of Cambridge, September 1980.

[Gifford 79] Gifford, D.K.

"Weighted Voting for Replicated Data".

Proceedings of the Seventh Symposium on Operating Systems Principles,

Dec 1979, pp 150-162.

[Girling 82] Girling, C.G.

"Object Representation on a Heterogeneous Network".

Operating Systems Review, vol 16, no 4, October 1982, pp 49-59.

[Hirsch 74] Hirsch, P.

"SITA: Rating a Packet-Switched Network".

Datamation, March 1974, pp 60-63.

[ISO 82] International Standards Organization

"Information Processing Systems - Open Systems Interconnection - Basic

Reference Model".

ISO DIS 7498, International Standards Organization, 1982.

[JNT 82] Joint Network Team

"Cambridge Ring 82 Protocol Specifications".

Computer Board and Research Councils, Department of Education and Science,

Nov 1982.

[Johnson 80] Johnson, M.A.

"Byte Stream Protocol"

Computer System Research Group Note, Sept 1980.

Reproduced as an appendix in [Needham 82a].

[Kirstein 82] Kirstein, P.T., Burren, J.W., Daniels, R., Griffiths, J.W.R., King, D., McDowell, C., and Needham, R.M.

"The UNIVERSE Project"

Proceedings of the Sixth International Conference on Computer Communication, Sept 1982, pp 442-447.

[Kleinrock 74] Kleinrock, L., and Naylor, W.

"On Measured Behavior of the ARPA Network".

AFIPS Conference Proceedings, vol 43, NCC, Chicago, May 1974, pp 767-780.

[Lee 82] Lee, E.S., Boulton, P.I.P., Dmitstrevsky, S., and Ikeman, H.

"HUBNET: A Glass Fiber 50 Mb/s Local Area Network".

Proceedings of the 11th Biennial Symposium on Communications. Queens University, Kingston, Ontario, Canada, June 1982.

[Leslie 82a] Leslie, I.M.

"A High Performance Gateway for the Local Connection of Cambridge Rings".

Local Computer Networks, pp 267-277, North Holland 1982, P.C. Ravasio, G. Hopkins, and N. Naffah (eds.)

[Leslie 82b] Leslie, I.M., and Ody, N.J.

"Universe Nameservers".

Universe Paper 126.1, Rutherford Appleton Laboratory, June 1982.

[Linington 83] Linington, P.F.

Private communication, January 1983.

[Logica 81] Logica VTS Ltd.

"Interface Unit Manual Multibus Intelligent Interface Unit".

Logica VTS Ltd., 1981.

[MCSL 76] Marconi Communication Systems Limited

"Technical Proposal for the Supply of Data Transmission Terminals".

ESR.3507, MCSL, March 1976.

[Metcalfe 76] Metcalfe, R.M., and Boggs, D.R.

"ETHERNET: Distributed Packet Switching for Local Computer Networks".

Communications of the ACM, vol 19, no 7, July 1976, pp 395-404.

[Needham 82a] Needham, R.M., and Herbert, A.J.

The Cambridge Distributed Computing System.

Addison Wesley, London 1982.

[Needham 82b] Needham, R.M.

Private communication, Decemeber 1982.

[Nelson 81] Nelson, B.J.

"Remote Procedure Call".

Ph.D. dissertation, Carnegie Mellon University, May 1981.

[Olofsson 80] Olofsson, K.S., and Segal, B.

"STELLA Project: Communications Interface Module".

CERN Report DD/80/2, June 1980.

[Richards 79] Richards, M., Aylward, A.R., Bond, P., Evans, R.D., and Knight, B.J.

"TRIPOS - A Portable Operating System for Mini-Computers"

Software Practice and Experience, vol 9, no 7, July 1979, pp 513-526.

[Richardson 83] Richardson, M.F., and Needham, R.M.

"The TRIPOS Filing Machine, a Front End to a Fileserver"

to appear in Proceedings of the A.C.M. Ninth Symposiom on Operating System Principles, October 1983.

[Roberts 73] Roberts, L.G., and Wessler, B.D.

"The ARPA Network".

Computer Communications Networks, N. Abramson and F. Kuo (eds.),

Prentice Hall 1973, pp 485-500.

[Saltzer 80] Saltzer, J.H., Reed, D.P., and Clark, D.D.

"Source Routing for Campus-Wide Internet Transport".

Local Networks for Computer Communications, pp 1-23, North Holland 1980,

A. West and P. Janson (eds.)

[Saltzer 82] Saltzer, J.H.

"On the Naming and Binding of Network Destinations".

Local Computer Networks, pp 311-317, North Holland 1982, P.C. Ravasio,

G. Hopkins, and N. Naffah (eds.)

[Schutt 81] Schutt T.E., and Welch P.H.

"Applying Micro-Computers In the Local Area Network".

Local Networks and Distributed Office Systems, Online 1981, pp 491-501.

[Shoch 78] Shoch, J.F.

"Internetwork Naming, Routing and Addressing".

Proc. 17th IEEE. Computer Society International Conference, Sept 1978,

pp 280-287.

[Spratt 80] Spratt, E.B.

"Operational Experience with a Cambridge Ring Local Network in a University

Environment".

Local Networks for Computer Communications, pp 81-106, North Holland 1980,

A. West and P. Janson (eds.)

[Waters 82] Waters, A.G., and Adams, C.J.

"Satellite Transmission Protocol, Implementation Specification".

Universe Paper 125.2, Rutherford Appleton Laboratory, March 1982.

[Wheeler 79] Wheeler, D.J., and Hopper, A.

"Maintenance of Ring Communication Systems".

IEEE Transactions on Communications, vol COM-27, no 4,April 1979, pp 760-761.

[Wilkes 79a] Wilkes M.V., and Wheeler, D.J.

"The Cambridge Digital Communication Ring".

Local Area Communication Network Symposium, Mitre Corp. and National Bureau of Standards, Boston, May 1979.

[Wilkes 79b] Wilkes M.V., and Needham, R.M.

"The Cambridge CAP Computer and Its Operating System".

Elsevier North Holland, New York, 1979.

[Zimmerman 80] Zimmerman, H.

"OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection".

IEEE Transactions on Communications, vol COM-28, no 4, April 1980, pp 425-432.