A Distributed Approach to the Interconnection of Heterogeneous Computer Networks

R. Braden*, R. Cole, P. Higginson, P. Lloyd

Department of Computer Science, University College London,
Gower Street, London. WC1E 6BT, UK

ABSTRACT

This paper describes a distributed architecture for the flexible interconnection of heterogeneous networks with a number of mini- and micro-computers, in a research environment. The interconnected networks include the DARPA Internet, which uses the DoD protocols, and the X25-based networks PSS and SERCNET.

The approach described here distributes the network access into a set of microcomputers acting as network front-ends ("network access machines"), with a local area network (Cambridge Ring) as a common bus. Communication between the hosts and the network access machines uses an interprocessor communication mechanism with a standard transport-level virtual-call interface, which is described. This arrangement provides local hosts with flexible access to any of the networks, and supports a relay system which allows users on one network to access hosts and facilities on any of the other networks.

1. A Distributed Network Access Facility

The INDRA research group of the Department of Computer Science, University College London (UCL), is involved in research with computer communication networks of various types. As part of this work we require inter-computer communication with hosts on a variety of computer networks. The communication is needed both for research and for service utilities such as file transfer, mail, and terminal access.

UCL currently has one or more connections to each of the following wide-area networks:

1. PSS, the X25 public data network provided by British Telecom [Medcra80].

2. SERCNET, a private X25 network of the UK Science and Engineering Research Council.

3. DARPA Internet, a concatenation of networks using the DoD communication protocols [Postel82].

The UCL research group's major computing resource is a set UNIX systems running on DEC PDP/11 computers. Other UNIX systems in the Computer Science Department are used for teaching. Local terminal access is provided to all these machines via terminal multiplexor microprocessors across a Cambridge Ring [Wilkes75].

The UNIX operating system is ideal for the development of system and application software. However, like most general-purpose operating systems, standard UNIX does not provide a suitable environment for the implementation of computer network protocols [Clark82]. There are serious problems of reliability, flexibility, and kernel address space. To operate the two (or more) network protocols needed at UCL within UNIX would have required an unacceptable share of the UNIX system resources, even if it were feasible.

A common solution to our problem of adding network support to a general-purpose system is to place the network software in a small "front-end" machine, linked to the mainframe by a high speed hardware interface. Our problem was, however, somewhat more complex than simply providing access to just a few UNIX

---

*Present address: R. Braden, Office of Academic Computing, UCLA, Los Angeles 90024.

systems from multiple networks. The range of service and research activities of the group currently require network access from all of the networks listed above to:

1. Two or more different UNIX systems, providing not only time-sharing services but also file transfer and electronic mail.

2. A "Terminal Gateway" which provides terminal protocol conversion between the X25 networks and the DARPA Internet.

3. A real-time network monitoring facility.

4. A variety of experiments in network interconnection and in higher-level protocols for messages, facsimile, and multi-media mail.

This range of problems led us to generalise the front-end concept to create a distributed network access facility. Instead of a single high-speed interface between a mainframe and front-end, a high-speed local network is used to link multiple front-end machines and hosts. The local area network, which is a Cambridge Ring, is used as a common hardware "bus" linking the following machines:

i. a number of PDP-11 machines (currently 5) running UNIX;

ii. a large number of dedicated LSI-11 computers (currently twelve) running network software; and

iii. a set of local terminal multiplexors based on Z80s.

Figure 1 shows the components of this distributed architecture discussed here.

Particular LSI-11s are used as front-end machines, dedicated to interfacing to individual networks; we call these the "network access machines". The rest of the machines are users of the network services provided by the network access machines. We will describe the two classes of machines later.

The network access machines use the special purpose real-time operating system MOS [Cole81]. In principle, each network access machine handles the network and link-level protocols appropriate to its network. The higher level protocol implementations reside on the user hosts, which are running either UNIX or MOS.

2. The Network Access Interface

An important simplification in the implementation of this distributed system was achieved with the design of a standardised interface between user processes and the network software. The interface design was named (somewhat coyly) 'Clean and Simple' [Bennet80].

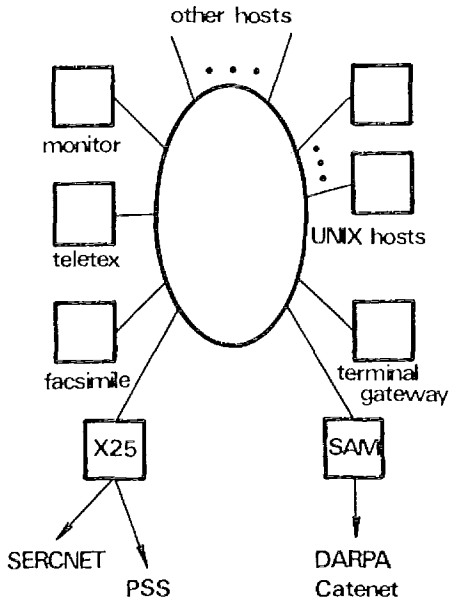

Figure 1. **Interconnection Components**

2.1 Clean and Simple

The Clean and Simple specification defines a simple transport-level connection-oriented interface between a user process and a network. The use of a standardised interface means that user processes can be implemented to use any available network protocol rather than a particular protocol. In this sense, the interface creates an idealised network service. For example, Clean and Simple defines data transfer as a simple byte stream with records optionally marked with "push" bits; as a result, the sender and receiver need have no knowledge of packet sizes or boundaries.

Figure 2 lists the request types which the Clean and Simple interface provides to a user process.

In practice, of course, there are unavoidable minor differences between networks, and hence between the Clean and Simple interfaces to them. A user process that uses special features of a particular network must take account of these differences; however, the fact that all networks are nearly the same introduces

GETCID	Get Call Identifier (id used in all subsequent requests for this call).
OPEN	Open a call to a destination
LISTEN	Wait for an incoming call
READ	Get data
WRITE	Send data
CONTROL READ	Accept asynchronous signals
CLOSE	Close call (and release identifier)
ABORT	Abort call (and release identifier)

Figure 2. **Clean and Simple Request Types**

great simplifications into the development of user processes.

For example, the CLOSE function in the Clean and Simple interface is defined as a full-duplex function (that is, a connection can be "half-closed"). The user processes at both ends must issue CLOSE requests before the network process will release resources. This definition is required to handle the semantics of the DARPA Internet protocol TCP. The half-duplex CLOSE of X25 has to be mapped into the CLOSE of the interface, and the fact that X25 may lose data during the close is considered a network-specific feature. The Clean and Simple interface has a second form of close primitive (called ABORT, see Figure 2) which is requested to complete as soon as possible, losing data if necessary. An ABORT maps into a RST in TCP, but is identical to a CLOSE for X25.

The Clean and Simple interface provides a request/response interface for the user process. Under UNIX, this is a set of subroutine calls. Under MOS, it is implemented by interprocess signals using structures very similar to those for device I/O. To date, we have implemented Clean and Simple interfaces for X25, for TCP, and for the Cambridge Ring (using the BB/BSP protocols [Needha82])

We can easily extend the idealised network concept of the Clean and Simple interface to encompass a broader class of connection-oriented services. At one end of the range, a full transport service interface such as the Network Independent Transport Service (NITS) [PSS8Ø] is possible. At the other end, a Clean and Simple subset (only the OPEN, CLOSE, READ, and WRITE primitives) has been implemented as an interface to a MOS file system.

2.2 Remote Clean and Simple

The Clean and Simple specification defines the interface between a user process and a network process. In most cases, these processes will in fact be in different machines - the network process in a network access machine, and the user process in another MOS system or a UNIX system. This led to the creation of remote network service interfaces, with a set of routines known as "Interprocessor Clean and Simple" or IPCS. A user process that wishes to access a network uses a Clean and Simple interface to a local IPCS module, which passes the requests across the Cambridge Ring to its corresponding IPCS module in the network access machine; see Figure 3. The responses are passed back over the same path to the user.

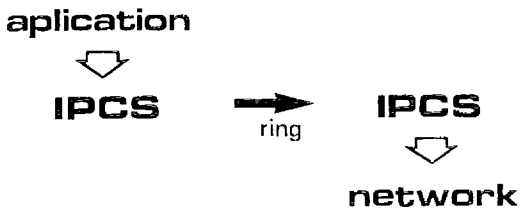

Figure 3. **IPCS Model**

The IPCS modules thus provide an interprocess communication mechanism which extends the Clean and Simple network interface transparently from the network access machines into all the user machines. With the use of IPCS, a user program can be moved to any machine (even into a network access machine) and function without change (except variation in delay). IPCS avoids the insertion of an additional network (or transport) level protocol, with the attendant gateways and addressing problems.

The IPCS implementation uses a private protocol which in effect envelopes the user process's requests and data, and transmits them over the Cambridge Ring. The enveloping is necessary in order to correctly reflect all of the network facilities to the user. For example, interrupt messages and X25 Q-bits can be used by user processes without any such facilities in the Cambridge Ring protocols.

The distributed architecture makes it easier to share scarce network interfaces between service uses and experimental work, since problems with experimental software are kept at arms length from the network access machines. Finally, IPCS can also be used over other transmission methods (or ring protocols), and it

minimises the amount of code required in the over-committed LSI-11 MOS systems.

The main disadvantage of the IPCS approach is that it does not decouple the user process from the buffering requirements and flow control of the network process. For example, the user must supply sufficient buffers for the anticipated total network delay. While X25 connections are local to the UK and experience only modest delays, the TCP connections may be half-way round the world and can show delays of several seconds.

3. Network Access Machines

A typical network access machine contains the MOS processes to implement the network access protocols as well as an IPCS process. It also contains the MOS real-time operating system with hardware drivers for the network and a ring interface. Finally, each machine contains status display and control processes.

Much of the basic network software in the network access machines was derived from previous research projects at UCL or the DARPA research community. In each case, it was necessary to add a Clean and Simple interface "on top" of the existing user interface of the network code. In most cases this has proven to be a small programming task, and resulted in an efficient implementation.

3.1 X25 networks

Each of the physical links to an X25 network may have only one network (DTE) address, and X25 is strictly a virtual-call protocol. This means that all multiplexing has to be carried out using higher level addresses, and that there can be at most one network access machine for each X25 network. IPCS is essential as a mechanism for obtaining X25 access from the application machines; the only alternative would be a network-level gateway.

On the other hand, a single LSI-11 network access machine can support multiple X25 lines. PSS and SERCNET are almost exactly compatible, so one copy of the network code can be shared.

A wide range of user-level protocols is employed over X25; these include: X29 and an older SERCNET terminal protocol called ITP; experiments with Teletex protocols which use the S.70 [CCITT80] transport service protocol; and the NIFTP and Mail services which use the NITS transport protocol.

3.2 DARPA Internet Access

The DARPA Internet uses an end-to-end virtual call protocol (Transmission Control Protocol or TCP) above a datagram protocol (Internet Protocol or IP) [Postel81]. The datagram basis and addressing structure of IP allow a great deal of flexibility in the use of multiple network access machines, even though there is only a single physical connection to the Internet.

To exploit the flexibility of IP, UCL has implemented a local IP datagram switch called the SATNET Access Machine (SAM) [Lloyd82]. The SAM connects on one side to the Cambridge Ring and on the other to the UCL/SATNET Gateway, a full Internet Gateway within the DARPA Internet structure. In effect, the SAM does the logical packet switching, using the Ring as the physical bus for packets. IP datagrams are sent across the Ring using the datagram-level of the Ring Protocol.

Thus, any host at UCL that wishes to interchange IP datagrams with the Internet will send and receive IP datagrams across the Ring from the SAM. For example, there is an LSI-11 TCP access machine used for some of the virtual calls to the Internet. Application machines use IPCS (across the Ring) to open virtual TCP calls through this machine, which in turn exchanges IP packets with the SAM (again, across the Ring).

In addition to performing its basic function as datagram switch, the SAM can operate as a complete network access machine. Thus, it contains an IPCS module and TCP/IP implementations, so it can accept IPCS requests and act as the endpoint for the DARPA communication protocols. The choice of an integral or a separate network access machine depends upon considerations such as delay and available buffering.

4. Applications

We will now describe three different applications of the distributed architecture, to illustrate the flexibility and diversity of the system.

4.1 Terminal Gateway

UCL operates a terminal protocol converter, or "Terminal Gateway" (TG), which allows terminal access between the DARPA Internet and the X25 networks PSS and SERCNET in the UK [Braden82].

The TG is a good example of the application of the distributed architecture. As shown in Figure 4, the TG

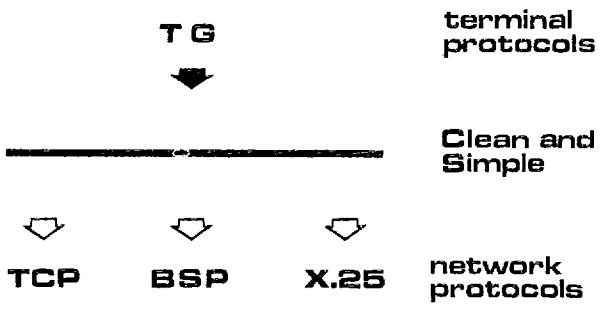

Figure 4. **Terminal Gateway Operation**

uses IPCS to communicate with the network access machines for the DARPA Internet, PSS, and SERCNET. The TG also provides access to server ports on the various UNIX systems, over the Cambridge Ring. This only required implementing in the TG the terminal access protocol used between these UNIX systems and the multiplexors, and using the Clean and Simple interface to the ring protocol BSP. As a result, a user on any of the networks supported by the TG can obtain terminal access to UNIX.

The TG is implemented in an LSI-11 under MOS. It makes much use of the uniformity and simplicity allowed by the Clean and Simple interface to all these networks. In addition to the basic network interface driver, there are terminal-protocol-specific modules to implement:

1. Telnet, the terminal access protocol used in the Internet;

2. X29, the Public Data Network terminal protocol, and its UK variant TS29;

3. ITP, a private terminal protocol used in SERCNET;

4. The UNIX terminal access protocol mentioned above.

Each of these modules translates between its terminal protocol and a canonical internal format; this is to avoid the "n squared" problem of all possible inter-protocol translations.

The TG also contains a user command interpreter. A user must log into the TG, to implement access control to the various networks. The user may then enter a TG "open" command to select a target network and host. There are also commands for closing calls, for displaying status, and for initiating control signals appropriate to the host's terminal protocol.

4.2 NIFTP and Computer Mail

The UCL file transfer and electronic mail applications are handled using the UK file transfer protocol NIFTP [PSS80a]. As a complex, file-oriented, application program, the service NIFTP [Higgin82] is implemented under UNIX. It executes using background jobs, on several of the UNIX systems. Each NIFTP process accesses a network by calling Clean and Simple subroutines in the UNIX IPCS package. These routines communicate across the Ring with the IPCS routine in the MOS network access machine, as described earlier.

Mail is exchanged with other systems using the NIFTP, so the main task of the mail programs is the sorting and queueing of mail. In addition, mail can be relayed and redistributed to other sites, and of course there are separate programs to examine and prepare mail.

4.3 Network Monitor

This application does not use transport level protocols, but does require the use of a real time system communicating with a UNIX system. Code in a dedicated LSI-11 MOS system periodically sends IP datagrams to the SAM; these datagrams contain echo request packets to probe various components of the DARPA Internet. The same probe system collects the replies (if any) to determine the status of the destination and if possible the round-trip time. When the status of a destination changes or the round-trip time varies, the probe system reports its statistics (across the Ring) to a server program running on a PDP/11 under UNIX.

The information thus collected on the UNIX system is used to create a real time status display, giving the latest information on those components of the Internet which have been probed. This information is available on any UNIX terminal throughout the building and also to any network user accessing a UNIX. In addition, the status reports are logged into a file which is later processed to produced a history of Internet availability.

5. Discussion and Conclusion

The distributed architecture described in this paper arose from the need to communicate using a number of heterogeneous computer networks, with maximum flexibility for future research. The two most important aspects of this architecture are:

1. The use of a local area network as a switch, both between the networks and between the end-to-end applications.

2. The use of a standard interface between the network protocol implementations and the end-to-end applications.

The distributed approach has separated the network-dependent protocols from the end-to-end protocols. This has relieved our bigger, timesharing, hosts from the burden of running several real time protocol implementations. In addition, where an end-to-end protocol can be used over several networks, such as the NIFTP, we are able to use the same implementation for all those networks.

Nearly all of the research projects in the UCL Computer Science Department now depend upon this distributed architecture; applications include: Teletex, facsimile transmission, terminal access, file transfer, mail transfer, and research into network interconnection. It will be very easy to add further types of networks and protocols into this system, and to extend the end-to-end applications.

6. Acknowledgements

The UCL network interconnection design was the result of a series of discussions involving many members of the research group. We would particularly like to acknowledge the contribution of some former members of the group: Steve Treadwell, Chris Bennett, David Frost and Colin Bradbury. The design and implementation of the software has involved almost the entire research group over the past two years; we are pleased to acknowledge their contributions. Finally, we are grateful to SRI International for the original MOS and TCP/IP software, and to the Royal Signals and Radar Establishment (RSRE) for the basic X25 software and interface hardware.

The work described in this paper was supported by the Ministry of Defence under grant 2047/84, and the Science and Engineering Research Council under grants A/75695 and N2BIR0188.

References

[Bennet80] Bennett C., "Clean and Simple", INDRA Note 980, University College London, September 1980.

[Braden82] Braden, R., Cole, R., "Some problems in the Interconnection of Computer Networks", Proc ICCC '82, 969-974, North Holland, September 1982.

[CCITT80] CCITT, "Network-Independent Basic Transport Service for Teletex", Draft Recommendation S.70, Geneva, 1980.

[Clark82] Clark, D., "Modularity and Efficiency in Protocol Implementation", RFC 817, in Internet Protocol Implementation Guide, SRI International, Menlo Park, July 1982.

[Cole81] Cole, R., Treadwell, S., "MOS User Guide", University College London, INDRA Note 1042, January 1981.

[Higgin82] Higginson, P., Moulton, R., "Experiences with the use of the UK Network Independent File Transfer Protocol", Proc ICCC '82, 913-918, North Holland, September 1982.

[Lloyd82] Lloyd, P., "The SATNET Access Machine: Current Design and Operation", INDRA Note 1326, University College London, September 1982.

[Medcra80] Medcraft, D., "Development of the UK Packet Switched Services", Proc. Conf. on Data Networks, ONLINE, 173, 1980.

[Needha82] Needham, R. M., Herbert, A. G., "The Cambridge Distributed Computing System", Addison Wesley, 1982.

[PSS80] PSS User Forum, SG3, "A Network Independent Transport Service", SG3/CP(80)2, February 1980.

[PSS80a] PSS High Level Protocol Group, "A Network Independent File Transfer Protocol", DCPU, February 1980.

[Postel81] Postel, J., Sunshine, C. A., Cohen, D., "The ARPA Internet Protocol", Computer Networks, 261-271, July 1981.

[Postel82] Postel, J. B., Sunshine, C. A., Cohen, D., "Recent developments in the DARPA Internet Program", Proc ICCC '82, 975-980, North Holland, September 1982.

[Wilkes75] Wilkes, M., "Communication using a Digital Ring", Proc. PACNET Conf. Sendai, Japan, 47-55, August 1975.