# Transport Protocol Requirements for Distributed Multimedia Information Systems

L. H. NGOH AND T. P. HOPKINS

*Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL*

***Typical* Distributed Multimedia Information Systems *are made up of clusters of workstations connected by local area networks. Such systems allow users to exchange information in the form of 'documents' containing text, graphics and voice; some systems support both store-and-forward and real-time material. Here we consider various transport protocol requirements and conclude that some essential features are not available with existing protocols.***

***The main objective of the work described here is to investigate the use of transport protocols which support the notion of multicast logical connectivity, both within a single network and in an internet environment. In this paper we show that protocols with multicasting capabilities can yield a substantial performance improvement over protocols which support only unicast logical connectivity. We propose ways in which multicasting can be exploited to suit our application.***

***A prototype implementation of a multicast communication facility based on these requirements is presented, and an application model is used to measure the transport protocol performance under various possible system configurations and loads. Results of performance measurements are presented, based on experiments carried out on an Ethernet-based internet.***

*Received October 1988*

## 1. INTRODUCTION

Increased understanding of the design of distributed systems has resulted in very complex applications being developed, based on networked computing environments. One such application which is attracting increasing interest is the design of *Distributed Multimedia Information Systems* (DMIS).[23,17] Typically, a DMIS is made up of clusters of workstations connected by high-speed Local Area Networks (LANs). Such systems allow users to exchange information in the form of *documents* containing text, graphics and voice information. Recently, there have been proposals for such systems supporting information interchange in both a store-and-forward (e.g. mail, document processing) and a real-time (e.g. conferencing) manner.[25] A DMIS can provide highly effective interpersonal communications by supporting multiple information types. Effective information transfer may be provided in a DMIS by encompassing the store-and-forward mode currently provided by the Computer-Based Message Systems (CBMS)[16] and the real-time mode used for computer-based conferencing.[29,2]

## 2. DMIS ARCHITECTURES

A DMIS is capable of providing users with store-and-forward and real-time multimedia information interchange using a homogeneous interface. The information manipulated is in the form of a multimedia document, with the DMIS providing a smooth transition from one mode of communication to the other. Thus the DMIS supports effective information representation by allowing the creation and manipulation of multimedia documents. They also incorporate the functions of a mail system,[27] a document processing system[15] and a computer-based conferencing system. In this paper we concentrate on the information-interchange aspects of such systems.

### 2.1 DMIS system configuration

In general, DMIS are designed to operate in a distributed computing environment consisting of loosely coupled autonomous workstations, equipped with high-resolution raster displays and advanced I/O devices (e.g. voice I/O). They are connected by high-speed LANs, with gateways (see Fig. 1).

In a DMIS, many different document creation mechanisms may be used, possibly by different users. A document can be created as a mail message or as a conference workspace (shared object), or even for both purposes. For example, after a joint editing session using conference-mode communication, a copy of the final document is 'mailed' to several individual users.

A DMIS can support real-time conferencing by providing either centralised or decentralised control[29]. A centralised conferencing architecture (see Fig. 2) consists of a conference controller, which manages all the shared objects in a conference*. During conferencing, participants are allowed to communicate only with the conference controller. In turn, the controller can only talk to one of them at a time. This is often done by the controller sending a token to one of the participants. The participant who receives the token is said to have the conference floor and hence allowed to communicate with the controller. Any changes to the shared objects as a result of the communication are distributed to all the participants by the controller.

Alternatively, a decentralised conferencing architecture (see Fig. 3) may be used; this allows each participant to communicate directly with the rest of the participants. Such arrangements are in practice more difficult to achieve, as messages may arrive in a different order at different participants, creating a time-stamping problem. A decentralised conferencing arrangement also requires

* For performance reasons, some or all of these shared objects may be replicated at the rest of the workstations.

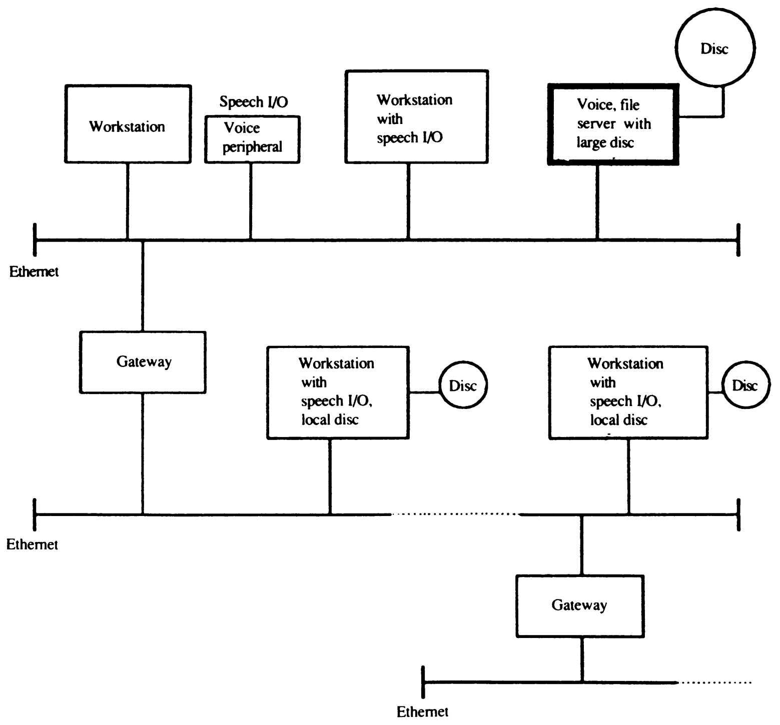
Disc
Workstation
Speech I/O
Voice peripheral
Workstation with speech I/O
Voice, file server with large disc
Ethernet
Gateway
Workstation with speech I/O, local disc
Disc
Workstation with speech I/O, local disc
Disc
Ethernet
Gateway
Ethernet

**Figure 1. A possible DMIS hardware configuration.**

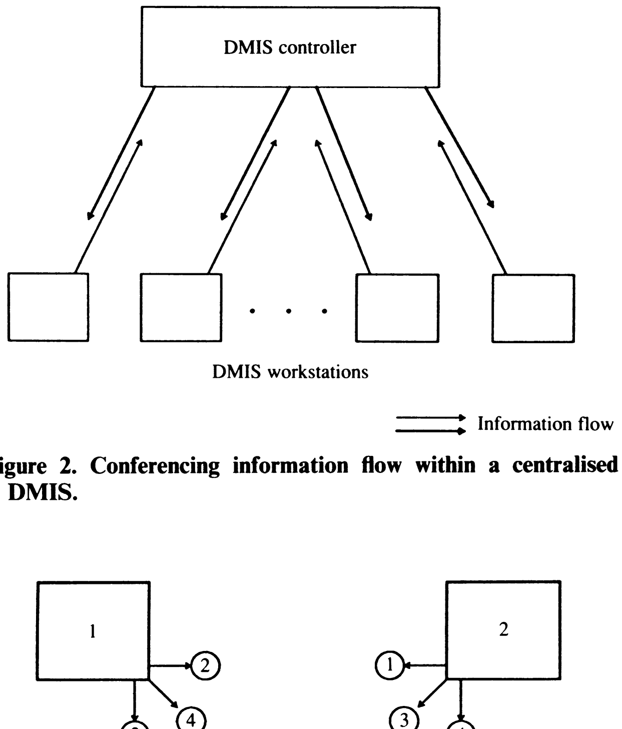

**Figure 2. Conferencing information flow within a centralised DMIS.**

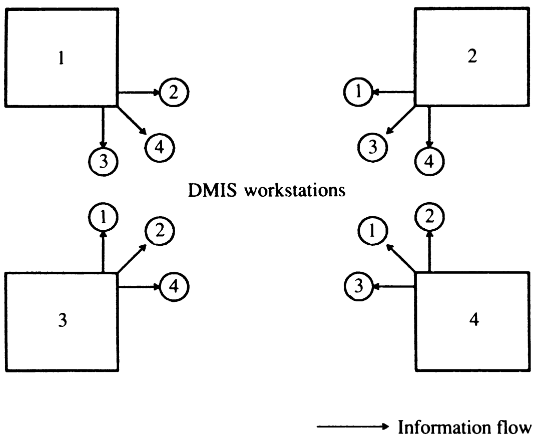

**Figure 3. Conferencing information flow within a decentralised DMIS.**

replication of all the shared objects at all the participants in order to function effectively.

The store-and-forward transmission model used by a DMIS is similar to that used by text-based mail systems.[20] This model consists essentially of two components (see Fig. 4). The first component is called the User Agent (UA); its function is to interact with the human user and provide the necessary mechanisms to describe the content of the final message. The second component is called the Mail Transfer Agent (MTA), which delivers the message prepared by the UA. The MTA achieves this by interpreting the mail address and delivering it to the appropriate UA or another MTA.

## 2.2 Application protocols

Various proprietary and international protocol standards are currently available for building a DMIS. These protocols can be divided into two groups: content protocols and transfer protocols. The content protocols include a set of standard communication formats defined for each of the data types contained in the document. For example, a graphics protocol can be used to describe all the graphical components in the document during conference data transmission. To permit communication among heterogeneous systems, some translation between the standard communication format and the local graphics system format may be required. There are a number of actual and proposed standards for graphics information exchange[5,1] and voice data fomat[28] suitable for conferencing use.

The content protocols group also includes a document-interchange protocol, which describes both the content and layout structures in a standard format. This ensures that both the content and layout of the document are preserved as the document is transferred from one machine to another. The document-interchange protocol

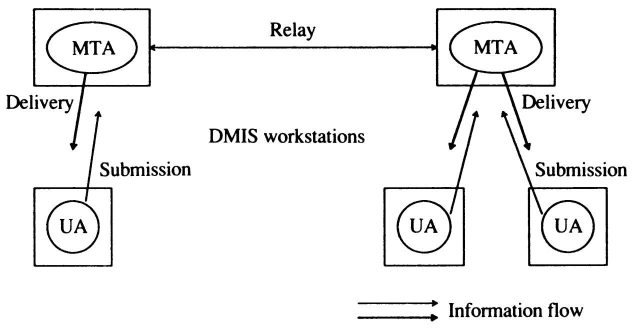

**Figure 4. Information flow in a DMIS with store-and-forward transmission.**

is important, as it ensures that a document can be processed by any system. The international standard multimedia document interchange protocol is called the Office Document Architecture* (ODA).[18] ODA is expected to be further developed to include more data types (e.g. voice, digitised image) in the near future.

The DMIS transfer protocols can be divided into two subgroups: protocols for store-and-forward transmission and for real-time conferencing. The transfer protocol for store-and-forward transmission contains primitives for delivering messages to the users and communicating with its peers on other machines. Since the exact content of the data is of no concern to the transfer protocol, it is advantageous that current-standard text-based mail-transfer protocol is used. One such possibility is the CCITT X. 400 series of message handling transfer protocols, which is the widely accepted international standard for computer message systems.

The real-time conferencing transfer protocol, on the other hand, is less well defined. This is partly because the design issues involved are more complex. Important primitives for real-time conference transfer protocol are those which provide the conference management functions, and functions for manipulating the shared objects. Furthermore, all these primitives are also delay-time critical. We expect that much more research is required in this area before an acceptable standard can emerge.

## 3. COMMUNICATION SUPPORT

The transfer protocols described in the last section rely on lower-level communication facilities for organising interaction among their peer entities. This gives rise to the question of the 'quality of service' required of these communication facilities to support a DMIS. To investigate this further, we must first identify categories of DMIS data traffic.

### 3.1 Data traffic in DMIS

*Control commands.* Various control commands are used for conference management; these include floor control, conference initialisation and termination, and parameter negotiation. There are also control commands being exchanged for the submission and delivery of documents in document processing and mail operations. This category has low-data traffic and is not particularly time-critical. However, highly reliable and reasonably fast transmission is required.

*Real-time interactions.* This category includes voice data and interactive graphical information, such as mouse and cursor movement. In particular, graphical update information which is associated with speech delivery comes into this category. The amount of traffic is expected to be relatively low, but fast propagation to all stations is critical. However, it can tolerate transmission errors and losses, provided they are corrected rapidly by subsequent transmissions.

*Non-real-time interactions.* Experience with conference systems[2] has shown that it is not necessary to have instantaneous replication of certain user actions, such as page scrolling and short graphical updates not associated with voice information. Hence, this category of low-data traffic might use a similar transmission mechanism to the control commands.

*Conference bulk data transfers.* Bulk data transfers typically occur when new users join in a conference session. This data traffic should be highly reliable, and should be completed within a reasonable time limit.

*Store-and-forward bulk data transfer.* This type of bulk data transfer occurs during the submission, delivery and relaying of multimedia documents in main and document processing systems. Beside the contents of the documents, the data to be transmitted include information on the structure of the mail documents, control information for processable documents, presentation-control information and possible timing relationships between different media. Reliable transmission is the main requirement for this category of data.

### 3.2 Communication model

A typical communication model for a DMIS which supports centralised real-time conferencing is illustrated in Fig. 5. This shows the relationships between the DMIS, the host operating system and the underlying data communication support. Fig. 5 assumes that the conference controller also acts as an MTA for data using

* The actual ODA definitions also cover each specific content architecture (e.g. character) and the interchange format.

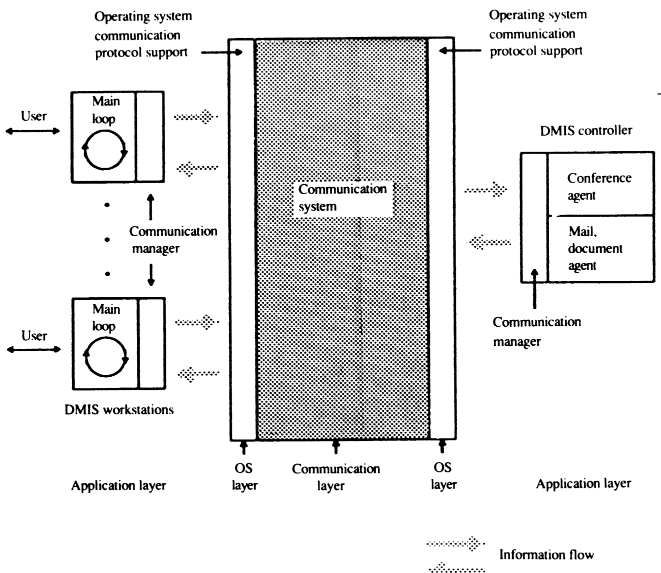

**Figure 5. Centralised DMIS communication model.**

the store-and-forward transmission mode. This configuration makes the controller a data distributor to all the workstations after receiving inputs from the same set of workstations. The communication model for decentralised real-time conferencing will be similar to Fig. 5, with the exception that communication will be direct between workstations and not routed via a controller.

From now on, we will concentrate mainly on the structure of the communication system. The highest layer of protocol we wish to consider here is the transport protocol, which provides a universal data transport service between peer processes.[33] One of the most notable transport protocols is the DoD Internet Transmission Control Protocol (TCP).[26] In general, transport protocols can be classified into two categories, depending on the way in which data is transported. The first of these provides a connection-oriented service, while the second category is the connectionless service. Currently, both these services only support unicast logical connectivity; this provides the ability of a host to send a packet to only one other host in the network.

With the prominence of broadcast networks (particularly for LANs), and the ability to provide some broadcast support even in point-to-point networks, some connectionless transport services also offer broadcast logical connectivity. This permits a host to send a packet to all the other hosts in the network, and can be extended to include hosts in an internetwork environment.[4] However, here we maintain that transport protocols supporting both unicast and broadcast communications are not adequate in supporting the functions of a DMIS.

### 3.3 Unicast communication

Unicast transport service can introduce an unacceptable transmission delay when used for information transmission among the workstations in a DMIS. In the case of a centralised DMIS, as shown in Fig. 5, whenever any input (such as new mouse coordinates) is given to the controller, it sends the result (e.g. a new cursor position) to each workstation. If a unicast transmission scheme is used by the controller, multiple copies of the same data will have to be sent, one to each workstation. Because of the finite throughput available at the sending device, the delay could reach an unacceptable level with a large number of workstations.

Furthermore, transmitting multiple copies of the same data will increase the traffic in the network, hence introducing greater end-to-end delays. Thus, it is expected that the performance of a DMIS will degrade as the number of workstations increases. The long end-to-end delay will also restrict the ability of the DMIS to support media, such as voice and video, which are very time-critical.

### 3.4 Broadcast communication

Most current transport protocols also support the notion of logical broadcast connectivity over a wide range of physical networks. Various internet broadcasting mechanisms have also been developed over the years and are supported by various transport protocols.[30]

Clearly, fewer copies of the same message will be generated in a DMIS which uses broadcast techniques for information interchange. This overcomes the high delay problem of the unicast case. However, a simple broadcast transmission will not be able to meet some of the DMIS requirements.

Broadcast mechanisms are rather inelegant ways of implementing distributed applications. A broadcast packet will be received and processed by all hosts on the network, thus imposing unnecessary work on all hosts. For these reasons, broadcast transmission should only be used relatively infrequently. Unfortunately, a DMIS will require rather frequent broadcast transmissions.

Unwanted broadcast packets processed by hosts also have the potential to cause security and other (e.g. oscillation) problems, especially when internet broadcast is involved.

## 4. TRANSPORT PROTOCOL REQUIREMENTS

There is yet another network addressing mode called multicast logical connectivity; this means that a host can send a packet to a group of hosts in the network. Hence, in our application the controller can distribute the data to the other workstations simply by using one single copy addressed to the group. This not only eliminates the problem of transmitting multiple copies of the same data, but also allows a subset of workstations to be grouped together under a single multicast identity. Multicast communication has been addressed at the data link level by networks such as the Ethernet,[11] and at the transport level by the Internet Datagram Protocol (IDP) in Xerox Network Systems (XNS).[32] In general, however, such capability remains largely unexploited. In the rest of this section we explain the relevance of multicast transport service to DMIS operation. Other important requirements of the DMIS transport protocol are also discussed.

Still other transport protocol design issues, such as error-reporting mechanisms and security,[31] should also be taken into consideration; however, they are not discussed here.

### 4.1 Message service interface

Multicast message transmissions among workstations in a DMIS are one-to-group unidirectional transactions. In the case of a centralised DMIS, the output of the controller is distributed to all the other workstations after receiving an input from one of the workstations. In the decentralised case, one-to-group multicast is performed between any one of the DMIS workstations and the remainder.

We envisage the provision of a process group-send primitive which provides multicast transmission. Also, a process group-receive primitive is required, which receives data from a particular multicast group. Both primitives should also provide efficient buffering of DMIS data from small single-packet transfers, such as voice data and small updates, to bulky multi-packet data transfers.

When using the conferencing mode, a conference group is normally initiated with a *create-group* group management operation* by the conference originator.

* Actual process group management is expected to be done at the session level. However, the multicast transport protocol should provide adequate end-to-end addressing to support these functions.

Other workstations can then issue a *join-group* operation to join in the conference. Any workstation can also leave the conference group by issuing a *leave-group* operation. A conference group can also be destroyed explicitly by issuing a *destroy-group* operation by the appropriate workstation, subject to higher-level control.

In the store-and-forward mode, when a document is to be transmitted to various workstations a list of address strings is normally supplied. The DMIS should use this to create a mail-group, which has the intended recipients as the group members. Each group member is then invited to receive the document (or informed of its existence) by the document distribution agent issuing a *join-group* operation. This type of group is temporary, and is destroyed once the data transfer has been completed.

Each host should be allowed to join in more than one communicating group. Such a situation arises when a user decides to join in more than one DMIS (perhaps by opening up separate windows on the workstation screen), or when more than one user is sharing the same host. However, we restrict the communication so that only hosts belonging to the same DMIS group are able to communicate with one another.

### 4.2 Data reliability, delay and throughput

As discussed in Section 3.1, reliable data transmission and low elapsed time are the two main requirements for several categories of DMIS data. However, not all categories need to be transferred reliably and fast. For example, mail data can tolerate fairly large delays, but requires highly reliable transmission. Conversely, voice data is time-critical but any lost, or 'old', data is meaningless and should not be retransmitted. Thus our group communication primitives should be able to specify the Quality of Service (QOS) required.

In general, DMIS data objects can vary from a few bytes (e.g. mouse coordinate data) to a few megabytes (e.g. entire multimedia documents). The transport protocol should be able to sustain a high data throughput for both small and large quantities of data. When transporting very large data objects, the provision of mechanisms to avoid data copying from process level to protocol level and vice versa is important.

The communication model of a DMIS (see Fig. 5) shows that the main function loop is interacting with the user, interpreting commands and supporting the user interface. Thus communication between the DMIS workstations is carried out asynchronously. It is the responsibility of the multicast transport- (or session-) level protocol to notify the associated user processes whenever input data is pending. Notification can be performed, for example, by sending software signals to the processes concerned.

Efficient support for conventional unicast communications is also an important requirement of a DMIS transport protocol. Unicast transmissions are used between individual workstations and the controller in a centralised DMIS. Unicast transmission is also used in a decentralised DMIS to allow work-stations to transmit specific data such as reporting error conditions. The transport protocol's support of unicast or 'one-member-only' group transmissions should incur only a minimum overhead due to the existence of the multicast facilities.

### 4.3 Network-level data delivery

In essence, the requirements of the network data-delivery mechanisms are simply to support the transport protocol specification. For example, the network protocol should be able to support host-level multicast facilities. It should also provide internet address-binding facilities to support dynamic group membership, allowing a host to become a member of different DMIS groups simultaneously. The network-level data-delivery service should also support multicast service implementation on a single network as well as over multiple networks interconnected by gateways.

## 5. RELATIONSHIP TO OTHER WORK

Besides applications in DMIS, multicast communication can also be useful in several other areas. This includes distributed replicated databases/file-servers,[12] parallel computation[10] and even distributed games[3]. In this section we consider some of these protocols and explain why none of them is suitable for a DMIS.

Cheriton[7] has proposed the Versatile Message Transaction Protocol (VMTP), VMTP is primarily designed to be the transport protocol of distributed operating system V.[8] VMTP supports the communications between many instantiations of the same V-system kernel code, one per participating machine, using message-based transaction-oriented communications. VMTP also supports multicast communication, allowing a process to communicate with a group of processes using a unique entity identifier, regardless of the exact location of these processes. VMTP has also been proposed to support other forms of message communication (e.g. unreliable multicast datagram) under non-distributed operating systems.[24]

However, we consider the structure of VMTP does not provide suitable support for DMIS. The main function of VMTP is to support transaction-oriented request-reply communication. However, our application requirements do not fit into the 'invoke and block', 'return and continue' sequence of the Remote Procedure Call (RPC) paradigm.

As a variant of the basic message-transaction operations, VMTP does support multicast datagram transmission with optional reliability control. However, no asynchronous communication facilities are provided. Current implementations of VMTP[24] also do not include the necessary mechanisms for group creation and deletion at the process level.

VMTP relies on the network-level multicast support for delivery of multicast packets.[9] As the transport protocol for the V distributed operating system, VMTP does not at present create temporary host groups, but relies on one or more permanent host groups. Creations of temporary host groups, however, are essential for protocols supporting the operation of a DMIS.

VMTP is designed to support a wide range of communication mechanisms including datagram, connection-oriented and transaction-oriented transmissions. This introduces far more complexity than is required for our application.

Crowcroft[12] proposed a multicast transport protocol which supports transaction-oriented, N-reliable multicast with arbitrary message size. The design is primarily intended to support replicated procedure calls usually found in a replicated, reliable file-server system. Much emphasis has been put on supporting a wide range of reliability semantics during transactions; this ranges from 1-reliable to all-reliable, with a user option to specify an explicit list of receivers from which replies must be received. However, Crowcroft's proposal does not address the support of a real-time datagram service required by a DMIS.

Deering[9] proposed and implemented a host-group model which supports network-level multicast facilities. This model is implemented as an extension to existing network and data-link level protocols.[13] However, it does not address the issues of transport-level multicast, but only provides a platform to support such an implementation.

In addition, there are a number of more specific experimental implementations of multicast extensions under Berkeley UNIX*.[19,21] These extensions are implemented in user-level processes (i.e. in user space), rather than being part of the operating system kernel. However, further work is needed to improve the performance and handling of bulk data in these implementations.

## 6. MULTICAST COMMUNICATION

As a 'proof of concept' prototype, a layer of software which supports the transport protocol requirements specified in Section 4.1 has been designed. In this section, we briefly describe the implementation of this software as user-level code under the SunOS† version of the UNIX operating system. In Section 7 we present the results of performance evaluations of this software together with evaluations of the existing unicast transport protocols. This prototype implementation can also be regarded as a 'simulation' of a full implementation, which is considered to be adequate for estimating the performance.

The multicast software is implemented on top of the DoD Internet User Datagram Protocol (UDP) available with the UNIX operating system. UDP provides a service for application programs to send messages using connectionless unreliable datagrams with minimum protocol overheads. Data I/O between the multicast software and UDP service is achieved through the creation of 'UNIX domain' sockets, using the socket abstraction supported by the operating system. Multicast data are delivered between the multicast software modules on different hosts using the UDP broadcast mechanism.

### 6.1 Process group management

The multicast software supports one or more processes on one host, which can belong to one or more multicast groups. If necessary, a process can also be the member of more than one group. Each multicast group is identified by a 64-bit unique group identifier generated by concatenating the process's host identifier, process identifier and the current system time.

To support management functions within the process group, such as allowing more than one process on a single host, a local membership table is kept. Each membership entry includes the process identifier and its corresponding multicast group identifier. Data are distributed to each group member via UNIX domain sockets created for each of the local members. Group management operations supported by the multicast software include multicast-group creation, joining a group, leaving a group and removal of a group association. For example, a process can create a group by using the `create` *o*peration, which takes the form:

```
NewGroupId=create();
```

`NewGroupId` is the unique multicast group identifier returned by the `create` operation. It is to be passed as an argument to `join` and `leave` operations by processes invoking the operations. Finally, a multicast group can be destroyed by the creator of the group using the `destroy` function.

### 6.2 Data transmission

The multicast software supports both unicast and multicast data transmissions. Data transmission is carried out with a multicast `send` operation which takes the form:

```
ReturnCode=send(id, datapointer);
```

Where `id` is a multicast identifier. If, however, `id` is a unicast identifier, the data pointed by `datapointer*` will be transmitted via the usual UDP mechanisms.

To support the asynchronous communication required by a DMIS, software interrupt signals are generated and sent to each group member whenever input data are available. These signals can then be *trapped* by the receiving process to exit temporarily its main control loop and process the input data. Interrupt signals are generated for both unicast and multicast data, and the data are received with the receive primitive which takes the form:

```
ReturnCode=receive(id, datapointer);
```

Fig. 6 shows the format of a multicast packet used by the multicast software. As far as the user processes are concerned, the multicast packet appears to have a very large data field. The implementation uses *scatter–gather* arrays to allow the software to gather output data from, and scatter input data to, a specified list of non-contiguous areas of memory. This avoids data copying overheads. Within the multicast software, the data are divided into UDP packets with 1024-byte† data fields for transmission over the network.

### 6.3 Data reliability

Reliable data transmission issues in our prototype implementation are tackled based upon the assumption that a very low error rate is provided by the underlying communication hardware. We further assume that packet loss is mainly due to the receiver being overrun by the transmitter, and that there are no packet duplication or sequencing problems. These assumptions are made with regard to the intended DMIS operating environmens;

* UNIX is a registered trademark of AT&T.

† SunOS is a trademark of Sun Microsystems, Inc.

* A pointer to a *scatter–gather* array.

† For buffering reasons, 1024 bytes is the optimum data size for Berkeley UNIX 4.2 BSD. The highest throughput is thus expected.[6]

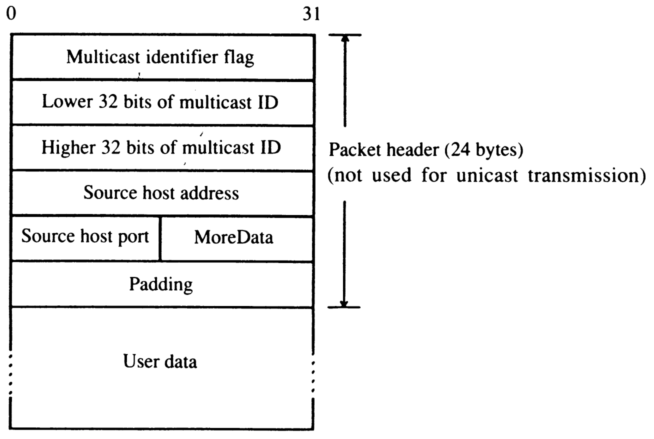
0
31
Multicast identifier flag
Lower 32 bits of multicast ID
Higher 32 bits of multicast ID
Source host address
Source host port
MoreData
Padding
User data
Packet header (24 bytes)
(not used for unicast transmission)

**Figure 6. A multicast transport protocol packet.**

we believe they are reasonable for a prototype implementation, although more stringent requirements might be appropriate for a full-scale system.

A simple 'stop and wait' flow control mechanism has been implemented in the multicast software. When transmitting multiple-packet data, an acknowledgement from the receiver must be received by the transmitter before the next packet is sent. Each receiver will generate the acknowledgement packet if the *MoreData* field (see Fig. 6) of the received packet header is set. The *MoreData* field is set by the transmitter whenever multipacket data are transmitted. Acknowledgements are sent by the receivers using the unicast mechanism, using the information in the 'Source Host' fields in the packet header. An acknowledgement packet contains a zero-length data field.

Usually, multiple acknowledgements are generated by the receivers of a multicast message. In order to ensure that the intended receivers are ready to accept the next data packet, the transmitter multicast module makes sure that all the acknowledgements have arrived before the next data packet is sent.*

## 7. THE EXPERIMENT

Formal analyses of multicast communications have been presented elsewhere.[12,22,14] In this section we present experimental results obtained with a DMIS application model. Our experiment involved the measurements of interprocess communication (IPC) delay as a result of frequent interchange of multimedia information within a DMIS. The IPC delay is defined as the total time elapsed from the moment a message is transmitted from a send process buffer to when that same message arrives at the receiving process buffer.

The aim of this experiment is to compare the IPC delay obtained using a unicast communication strategy and those obtained using the 'proof-of-concept' prototype multicast implementation described in the previous section.

### 7.1 Experimental model and methods

All experiments were carried out on UNIX-based Sun workstations connected by 10 Mbit/s Ethernet LANs,[11] which supported the TCP/IP protocol suite. A communication model consisting of a message exchange client/server pair was designed to simulate some of the possible communication functions required by a DMIS. The UNIX '`gettimeofday(2)`' system call was used to measure the elapsed time. It was assumed that the *precision* of this time source was adequate to allow measurements to the nearest millisecond to be made. Unfortunately, the clocks at the transmitting and receiving hosts had very poor absolute accuracy, and were not synchronised. To overcome this, the same message was 'echoed' by the receiving host to the sending host, using the same protocols, during the experimental measurements. With such an arrangement, the elapsed time was taken to be half the total time measured from when a sending process sent a message until the same message was received back again.

In order to achieve a controllable environment, the simulations involved only two selected machines at any one time and, for inter-network experiments, only a 'two-Ethernet' (i.e. one-gateway) system was used. For measurements involving two networks, a user process acting as a 'broadcast agent' was run at the host acting as a gateway. The task of this agent was to receive any data from the multicast software (transmitted using a low-level broadcast mechanism) from one network and broadcast the same data on to the other network. The code of the agent was deliberately kept as simple as possible to avoid any unnecessary overheads.

Experiments were carried out with different combinations of data sizes and transport protocols under different network configurations. Three 'single-packet' data sizes of 10, 200 and 1000 bytes were chosen to represent typical small data transfer such as might be used for voice, mouse coordinates and protocol control data. Bulk data of 100 kilobytes and 300 kilobytes were used as representative of large multimedia documents exchanged within a DMIS.

During our experiments, it was observed that our model is highly sensitive to the presence of other processes and to other network traffic. To ensure good repeatability, all measurements were taken with the load on the machines involved kept to the minimum and the Ethernet traffic practically free. As a result, all simulations were run overnight. Since, in practice, other network traffic will be present and the workstations will be performing other activities, this implies that the figures obtained represent the best possible case.

To improve further the repeatability of our final results, our measurements were repeated a large number of times. Furthermore, the entire experiment was repeated on different days and on different – but similar – machines to check the consistency of these results.

### 7.2 Results and discussion

The results of our experiments are presented here. First, Fig. 7 shows the measurements of the average time taken to transmit data items with various sizes using the multicast software. The results show a linear relationship between the amount of data transmitted and the total elapsed time, except when the data size is 1000 bytes. This is because this data size represents a total of 1024 bytes (the multicast header is 24 bytes) actually being transmitted by the underlying UDP software. The

* With a large group, this can increase the packet processing overheads significantly. Alternative schemes[14] are being examined.

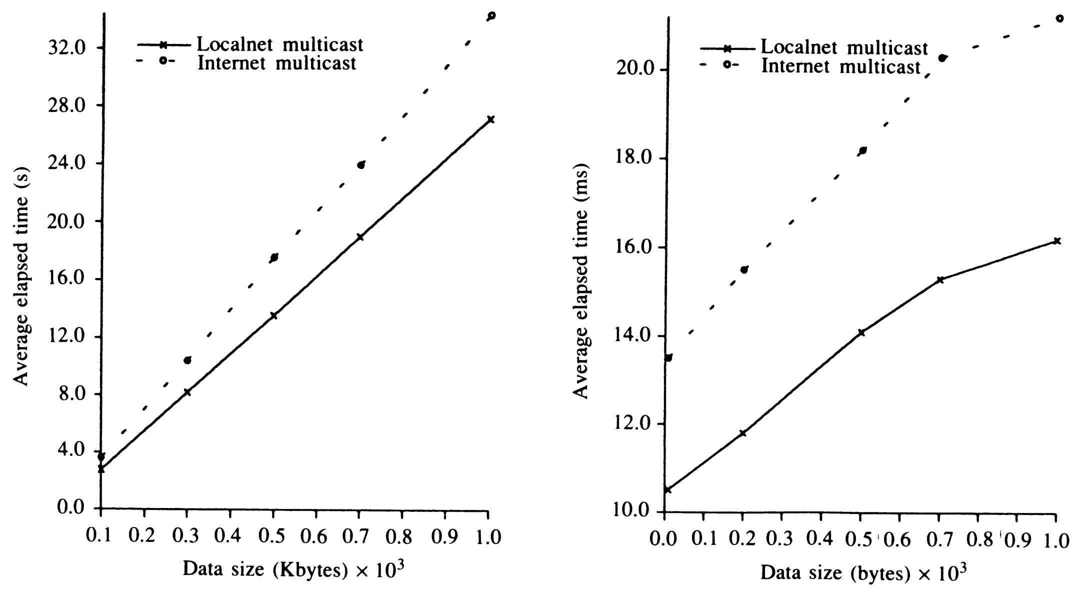

**Figure 7. Multicast communication performance with different data sizes.**

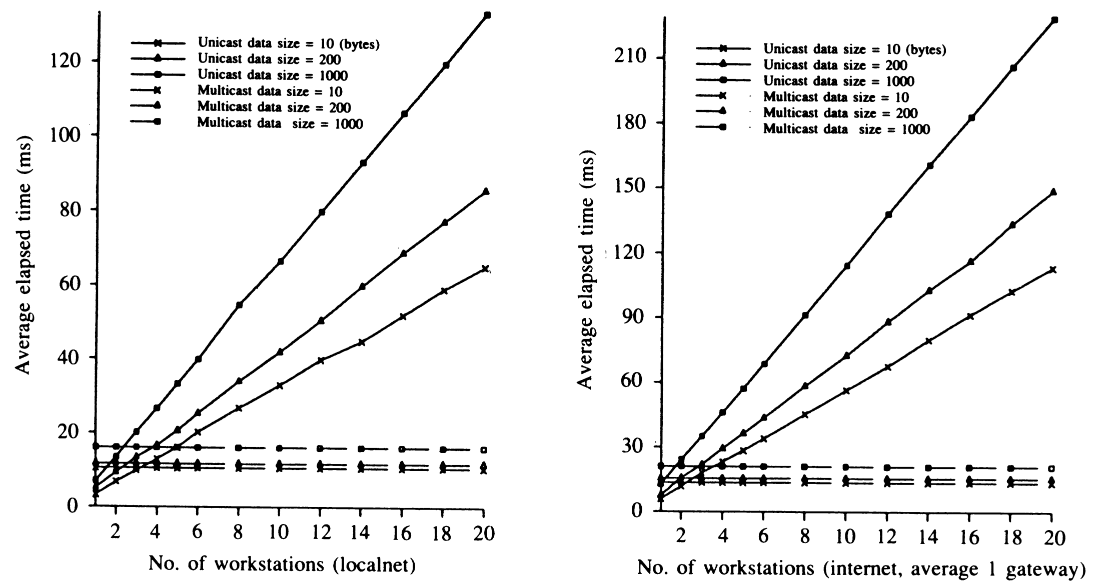

**Figure 8. Conference data distribution using unicast UDP.**

networking software of the operating system has been optimised for this size, hence maximum throughput was achieved. Similar observations have also been made elsewhere.[6]

In the remainder of this section, we present measurements of the elapsed time plotted against the number of workstations. To simulate the effect of two or more DMIS workstations involved in the information exchange, our simulation simply repeated the same experiment used for a single workstation the appropriate number of times. The results measured thus represent a situation where the loads on the network and hosts are identical for all DMIS workstations. These results are compared with the values obtained for the multicast case (plotted in long-dotted lines).

Our experimental results also include message transmission using the DoD TCP, as supported by the UNIX operating system. TCP is a connection-oriented transport protocol which requires connections to be set up prior to any data transmission. In our experiment we observed that the cost of establishing a connection makes TCP an unsuitable choice for message transmissions involving frequent and repeated connection set-up and shut-down.

Figs 8 and 9 show the results obtained using the DoD UDP, which is also supported by the UNIX operating system. The 'raw' UDP does not provide mechanisms for flow-control and necessary message re-transmissions. The lack of these facilities had made it difficult to carry out the unicast simulation, as the loss of a packet would cause the simulation to 'hang'. This was particularly serious when simulations involved the transmission of multi-packet (large) data objects via a gateway. To overcome this, a simple 'time-out' algorithm, which would determine the loss of a packet and re-transmit the data accordingly, was used during the experiment.

From the results of the above experiments we see that in most cases multicast transmission represents a reduction in the total elapsed time when the number of

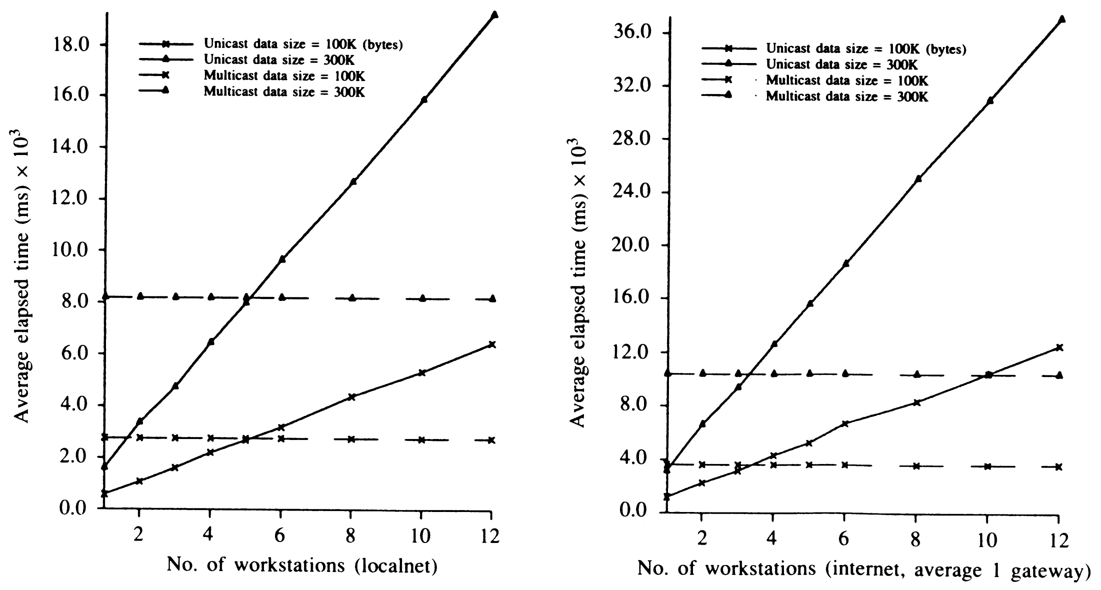

**Figure 9. Store-and-forward data distribution using unicast UDP.**

DMIS workstations is greater than two or three. Our results also show that high data throughput is achievable with the multicast software. Although we have not considered in detail the timing requirements for each type of multimedia data here, we can conclude that multicast communication is a necessary facility to support the operation of a DMIS.

## 8. CONCLUSION AND FUTURE WORK

This paper reports on-going work on the design of appropriate communication facilities to support the operation of a DMIS. It also presents some of our findings so far.

A DMIS is a complex distributed application, which is emerging as a result of advances made in a number of related areas of computer technology. We have presented an overview of possible DMIS architectures, with main focus on the information communication aspects. We have also examined components which allow the users of a DMIS to handle multimedia information, and exchange this information in both a real-time and a store-and-forward manner.

The way that information is interchanged between DMIS workstations depends greatly on the service primitives provided by the transport-level protocol. We have explained why current unicast transport services do not provide an adequate mechanism. Further, we have argued that data transmission using a simple broadcast technique is also inadequate. Instead, a transport protocol which supports multicast communication, together with other requirements, has been investigated.

To verify the feasibility of a transport protocol which supports the above requirements, a 'proof-of-concept' prototype implementation transport protocol was designed. The performance of this protocol was measured, and compared with that obtained using existing unicast transport protocols. This was achieved using a simple application model and some data representative of the different categories of DMIS traffic. The results of our simulations showed that the multicast transport protocol gave rise to significantly reduced IPC overheads compared to the unicast case. The amount of IPC overheads was also independent of the number of DMIS workstations.

Currently, a inter-network-level multicast similar to that proposed by Deering,[9] but with emphasis on real-time support, is under investigation. We also intend to consider the use of network hardware which supports physical multicast connectivity. In order to evaluate more closely the performance of our transport protocol, a more realistic application model is also required.

## REFERENCES

1. L. Aguilar, A format for a graphical communications protocol. *IEEE Computer Graphics and Applications*, **6** (3), 52–62 (1986).
2. L. Aguilar *et al.*, Architecture for a multimedia teleconferencing system. In *SIGCOMM '86 Symposium, Communications Architecture and Protocols*, pp. 126–136. *ACM SIGCOMMS* (1986).
3. E. J. Berglund and D. R. Cheriton, Amaze: a multiplayer computer game. *IEEE Software* **2** (3), 30–39 (1985).
4. D. R. Boggs, *Internet Broadcasting*. Technical Report CSL-83-3, Xerox Corporation (1983).
5. P. R. Bono, A survey of graphics standards and their role in information interchange. *IEEE Computer* **18** (10), 63–81 (1985).
6. L. F. Cabrera *et al.*, User-process communication performance in networks of computers. *IEEE Transactions on Software Engineering* **14** (1), 38–53 (1988).
7. D. Cheriton, VMTP: a transport protocol for the next generation of communication systems. In *SIGCOMM '86 Symposium, Communications Architecture and Protocols*, pp. 406–415. *ACM SIGCOMMS* (1986).
8. D. R. Cheriton, Distributed process groups in the V Kernel. *ACM Transactions on Computer Systems* **3** (2), 77–107 (1985).
9. D. R. Cheriton and S. E. Deering, Host groups: a multicast extension for datagram internetworks. *SIGCOMM Ninth Data Communication Symposium* **15** (4), 172–183 (1985).
10. D. R. Cheriton and M. Stumm, Multi-satellite star:

structuring parallel computations for a work-station cluster. *Proceedings of Distributed Computing 1986* (1986).

11. Digital Equipment Corporation, Intel Corporation and Xerox Corporation, *The Ethernet: A Local Area Network–Data Link Layer and Physical Layer Specification* Version 2.0 (1982).
12. J. Crowcroft and K. Paliwoda, A multicast transport protocol. In *SIGCOMM '88 Symposium on Communications Architectures and Protocols*, pp. 247–256. *ACM SIGCOMMS* (1988).
13. S. E. Deering, *Host Extensions for IP Multicasting*. Technical Report RFC 988, DARPA (1986).
14. A. Erramilli and R. P. Sigh, A reliable and efficient multicast protocol for broadband broadcast networks. In *SIGCOMM '87* pp. 343–352. *ACM SIGCOMMS* (1987). *SIGCOMMS* (1987).
15. H. Forsdick *et al.*, Initial experience with multimedia documents in Diamond. In *Computer-Based Message Service*, edited H. T. Smith, pp. 99–113. IFIP, Elsevier-North-Holland Amsterdam (1984).
16. J. J. Garcia-Luna-Aceves. A hierarchical architecture for computer-based message systems. *IEEE Transactions on Communications* **30** (1), 37–45 (1982).
17. J. J. Garcia-Luna-Aceves, Towards computer-based multimedia information systems. In *Computer Message System – 85*, edited R. Uhlig, pp. 61–77. IFIP, Elsevier-North-Holland, Amsterdam (1985).
18. W. Horak, Office document architecture and office document interchange formats: current status of international standardization. *IEEE Computer* **18** (10), 50–62 (1985).
19. L. Hughes, Multicast Communications in Distributed Systems. *PhD thesis*, University of Newcastle upon Tyne, England (1986).
20. F. F. Kuo and J. J. Garcia-Luna-Aceves, Issues in multimedia computer based message system design and standardisation. In *Information Technology and the Computer Network*, edited K. G. Beauchamp, pp. 43–52. NATO ASI Series, Springer, Heidelberg (1984).
21. Y. Lui *et al.*, A session layer design of a reliable IPC system in the Unix 4.2 environment. In *Proceedings of the Computer Networking Symposium*, pp. 120–9. IEEE Computer Society Press, New York (1986).
22. P. V. Mockapetris. Analysis of reliable multicast algorithms for local networks. In *Proceedings of Eighth Data Communications Symposium*, pp. 150–157. IEEE Computer Society Press, New York (1983)
23. L. H. Ngoh, An investigation of the development of computer-based distributed multimedia information systems. *Master's thesis*, Department of Computer Science, University of Manchester, England (1987).
24. E. Nordmark, *VMTP Implementation Notes*. Technical report, Stanford University (1987).
25. A. Poggio *et al.*, CCWS: a computer-based, multimedia information system. *IEEE Computer* **18** (10), 92–105 (1985). (1985).
26. J. Postel, *Transmission Control Protocol*. Technical report RFC 791, DARPA (1981).
27. J. K. Reynolds *et al.*, The DARPA experimental multimedia mail system. *IEEE Computer* **18** (10), 82–91 (1985).
28. J. K. Reynolds *et al.*, *Voice File Interchange Protocol* (*VFIP*). Technical report RFC 978, DARPA (1986).
29. S. K. Sarin, Interactive On-Line Conference. *PhD thesis*, M.I.T. Department of Electrical Engineering and Computer Science (1984).
30. D. W. Wall, *Mechanisms for Broadcast and Selective Broadcast*. Technical report DSL-TR-190, Stanford University (1980).
31. R. W. Watson, IPC interface and end-to-end (transport) protocol design issues. In Lecture Notes in Computer Science, Vol. 105, pp. 140–174. Springer, Berlin (1981).
32. Xerox, *Internet Transport Protocols* (*XSIS 028112*). Xerox System Integration Standard (1981).
33. H. Zimmermann, OSI reference model – the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications* **COM-28** (4), 425–432 (1980).

# Announcement

**Information Management Study Group**

The Institute of Administrative Management has recently taken responsibility for convening a small group of senior college tutors to discuss what was loosely referred to as 'The creative use of management information' or latterly 'Managing Information as a Resource'.

There was a large measure of agreement within the group that the greater, and hoped for, use of information in many organisations was inhibited by several important factors, including the following:

(1) Lack of understanding, at middle or senior levels of management, of the handling and accumulation of information through the normal administrative processes.

(2) The lack of information personnel with the necessary business and technical background to progress development within the organisation.

(3) The need to reorganise and reorientate the administrative structure within organisations to allow the growth and use of information as a resource.

As a first step in a programme of investigation and research, the group has decided to hold a one-day workshop in London in the autumn of 1989. It is hoped that some 50 employers and those involved in education and training will come together to make a preliminary survey of the situation as they see it. If, as a result of this gathering, certain problems can be identified, further activities will be arranged.

Further details of the workshop will be announced shortly.