IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 11, NO. 2, FEBRUARY 1993

A Network Interface Unit to Support Continuous Media

Gordon Blair, Andrew Campbell, Geoff Coulson, Francisco Garcia, David Hutchison, Andrew Scott, and Doug Shepherd

Abstract— **The combination of high-speed multiservice networks and multimedia workstations offers considerable potential for the development of distributed multimedia applications. However, many challenges remain before this potential can be realized. One key challenge is how to integrate continuous-media types such as audio and video into a distributed workstation environment. This paper describes an experimental system architecture based on a specialized multimedia network interface that attempts to provide this integration. The design and implementation of this system is discussed in depth in terms of workstation enhancement and distributed system support. A new approach to the problem of media synchronization is introduced, and the importance of quality of service in the architecture is highlighted. The paper also relates our experiences from this work, compares it with other approaches, and outlines our views on likely future developments in multimedia network interfacing.**

I. Introduction

MULTIMEDIA computing has emerged in the last few years as a major area of research. This work is motivated by the wide range of potential applications (e.g., music composition, computer-aided learning, and interactive video) made possible by such sources of information as voice and hi-fi quality audio and video. The combination of multimedia workstations and networking allows the communication and sharing of multimedia information. This combination offers far greater potential than stand-alone multimedia units, particularly now that high-speed multiservice networks are becoming available [33], and opens the way to novel applications such as desktop conferencing, distance learning, and multimedia groupware.

Research at Lancaster is focusing on the problem of handling *continuous media* such as voice, audio, video, and animation in a *network interface unit* situated between a host and a multiservice network. The term "continuous media" is used to refer to media types which have an implicit temporal dimension to their presentation. Manipulating these media types in distributed systems is challenging because of the requirement to maintain a guaranteed level of service over a significant period of time. This requires guaranteed resource allocation and continuous monitoring of quality of service.

Manuscript received February 15, 1992; revised August 31, 1992. This work was carried out partly within the Multimedia Network Interface (MNI) project, funded under the UK SERC Specially Promoted Program in Integrated Multiservice Communication Networks and co-sponsored by British Telecom Labs, and partly within the European Commission ESPRIT OSI 95 project.

The authors are with the Distributed Multimedia Research Group, Department of Computing, Lancaster University, Lancaster LA1 4YR, United Kingdom.

IEEE Log Number 9205974.

When considering the network interfacing requirements of continuous media, it can be seen that there are two distinct aspects to the problem. First there is the *high processing requirement* for protocol processing due to the extremely high data rates involved. For example, an uncompressed full-screen color video stream requires of the order of 140 Mb/s. Second, even assuming sufficient processing resources, there is the problem of *mismatch in bandwidth* between high-speed multiservice networks and the lower rates supported by typical host system buses. For example, a typical workstation bus such as VME can handle many fewer video connections than a 622 Mb/s high-speed network. Compression techniques such as MPEG can help by significantly reducing data rates, but this results in a correspondingly higher processing overhead.

To ease the host processing overheads of network interfacing, a number of techniques have traditionally been used. Early efforts used DMA hardware with incoming data notified to the processor by means of interrupts at the end of a block transfer. More recently, a number of researchers have experimented with network interfaces involving coprocessors which relieve the main host processor of protocol processing overheads [11]. Complementary developments have taken place in system software, with virtual memory remapping being used to eliminate the processing overhead of data copying from system to user space [1].

However, even with the application of these techniques, the system bus bottleneck remains. One solution to this problem is to build a network interface unit that directly supports the sourcing and sinking of continuous media data. This can be achieved by attaching media-specific processors and I/O units to the interface by means of dedicated high-speed data paths. Thus, high-volume audio and video data streams can be directly transferred without accessing the system bus, and the network interface becomes, in effect, a semi-autonomous unit which provides both network interfacing and specialized media processing.

The research at Lancaster is developing an experimental distributed multimedia system based on such a network interface unit. In addition to its basic network interfacing function, this unit provides considerable software support to distributed applications that run on the host. The software consists of data link, network and transport protocols, media-specific device drivers, and a distributed systems platform to provide applications with high-level access to the unit. This paper reports on the results and experiences of our research and is structured as follows. Section II discusses the requirements of handling continuous media in heterogeneous distributed systems. This is followed in Section III with an in-depth discussion of the Lancaster approach to supporting continuous media. This section describes both our architectural approach and implementation strategy. Section IV presents experiences from this work, and Section V describes related work. Finally, Section VI offers some concluding remarks and identifies likely developments in the field.

II. Requirements Study

At the outset of the research, it was clear that the design of our system would be influenced by two factors: the services offered by the new high-speed multiservice networks, such as HSLAN's, DQDB, and ATM-based WAN's, and the needs of the applications beginning to emerge in this new environment.

In order to assemble requirements for applications support, a detailed study was carried out. This study included a wide-ranging survey of both current and projected distributed multimedia applications, and also some experimental work in handling continuous media in various network environments. This section summarizes the results of this requirements study. Further details can be found in [42].

A. Supporting Continuous Media Communications

1) Simplex Connections: Continuous-media applications require unidirectional connections rather than the more conventional full-duplex connections. This follows from the inherently unidirectional nature of many continuous-media transmissions. For example, a remote camera-to-local video monitor connection is typical in a multimedia conferencing environment.

2) Quality of Service (QOS): A wide range of levels of QOS has emerged as the single most important requirement in the support of continuous-media applications. For example, to support video connections, high throughput is required and therefore high bandwidth guarantees must be available. Audio, on the other hand, will not require such a high bandwidth. Additionally, both video and audio data can tolerate some percentage loss of packets and bit errors. Error control with continuous-media information is problematic as retransmissions of lost or corrupted data are often inappropriate because the retransmitted data would arrive too late.

QOS guarantees must be made at all levels of the system so that a continuous level of performance is perceived by the user. These guarantees will take different forms at different levels. For example, in the operating system, the required QOS is expressed in terms of guaranteed processing resources for threads which handle streams of real-time continuous media. In contrast, at the user level, QOS is described in terms of quantities such as frames per second for video or sampling rate for audio.

A set of suitable QOS parameters [22] which are meaningful to the transport level and below, and which may be employed in characterizing individual continuous media, is comprised of throughput, end-to-end delay, delay variance or jitter, packet or bit error rates, and priority. At connection establishment time, it should be possible to quantify and express preferred, acceptable, and unacceptable tolerance levels for each of these parameters. The requested parameters should then undergo full end-to-end negotiation. The finally agreed upon tolerance levels should then be *guaranteed* for the duration of the connection.

An important requirement of continuous-media QOS management at the network level is a suitable reservation scheme [40], [5] which is able to set up and guarantee network resources, such as bandwidth and end-to-end delay, for high-performance continuous-media communications between network subscribers. While network resource reservation strategies lead to a lower utilization of network bandwidth, guaranteeing continuous-media QOS is more important than maximizing utilization of the link bandwidth.

Finally, it is important that QOS support is *dynamic.* For example, in the presence of best-effort QOS, levels may degrade and users should be given the choice of accepting the reduced level of service or renegotiating the QOS without disrupting service.

3) Synchronization: One of the biggest problems in distributed multimedia systems is maintaining synchronization relationships across multiple activities. Our survey identified two categories of synchronization in multimedia systems:

- *event-driven synchronization:* This is the act of notifying that a relevant event or set of events has taken place, and then causing an associated action or actions to take place. This must all be done in a *timely* manner due to the real-time nature of continuous-media communication. For eample, a user clicking on the stop button relating to a video playout should cause the playout to stop instantaneously.
- *continuous synchronization:* This is an on-going commitment to a *repetitive* pattern of event-driven synchronization relationshps such as the "lip sync" relationship between the individual frames of the audio and video components of a playout.

4) Group and Multicast Communications: The requirement for multicast arises from the nature of projected multimedia applications, many of which will be in the area of cooperative working. In such applications, a multicast facility is required for both transactional communication (RPC) and continuous-media connections. In a continuous-media-based multicast session, a simple $1:n$ topology is usually all that is required. Appropriate support for group addressing must be provided in the transport layer, although ideally multicast support will be the responsibility of the underlying communications subsystem.

B. Heterogeneous Distributed Systems

The introduction of multimedia exacerbates the ever-present problem of heterogeneity in distributed systems. The major contributory factor is the specialized nature of the hardware and systems support required by multimedia services. A second factor is that wide area (as opposed to local area) interworking is seen as an essential element in many prospective distributed multimedia applications, and this will inevitably mean that end systems are of different sorts. It is essential to provide support for heterogeneity in such systems but, at the same time, performance must not be compromised by excessive layering and inefficient interchange formats.

C. Impact on System Design

The need to support continuous media has a global impact on system design, which demands a high level of integration between system components. For example, the notion of quality of service is present at all levels from the user to the network. A user may specify a transfer of video of a given size, color, depth, and frame rate, and this will map onto particular QOS parameters at the transport level which will ultimately result in a particular level of resource allocation at network nodes. At each level, the quality of service requirements from the level above must be reinterpreted and supported as appropriate. This is quite unlike the situation in conventional data transfers, where the nature of the user data is of no importance to the lower layers.

Another area where the impact of continuous media on system integration is evident is the passage of continuous media data through the system. Such integration must extend from the user level to the level of I/O and network interface hardware. Because of the importance of real-time behavior, the scheduling of all processing threads involved must be closely coordinated and data copies between threads must be minimized. For example, if video is being taken from a storage server and shipped across the network, the disk scheduler must cooperate with the reading thread which must in turn cooperate with the communications software to ensure that the temporal integrity of the video is not violated.

D. Summary of Requirements

The following key requirements have been identified in the discussion above.

i) Continuous media imposes many *new challenges* on system technologies, which impact on *all levels of system design* including distributed systems platforms, communications infrastructures, and hardware architectures.

ii) The concept of *quality of service*-driven architecture is particularly important and must be supported by mechanisms to negotiate, renegotiate, and monitor the quality of service provided.

iii) Other key requirements include the requirement to support a range of *synchronization* services and *multicast* communications.

iv) It is important to take an *evolutionary,* rather than revolutionary, approach to continuous media and hence to work within existing standards wherever possible.

These requirements have influenced the design of the multimedia network interface and associated support software described in this paper. However, our views on the implications of these requirements, in particular quality of service, have evolved during the implementation of this system. A higher-performance redesign of the network interface unit, outlined in Section IV-D, is currently under way. This is taking fully into account the demands of an integrated quality of service architecture, including operating system support.

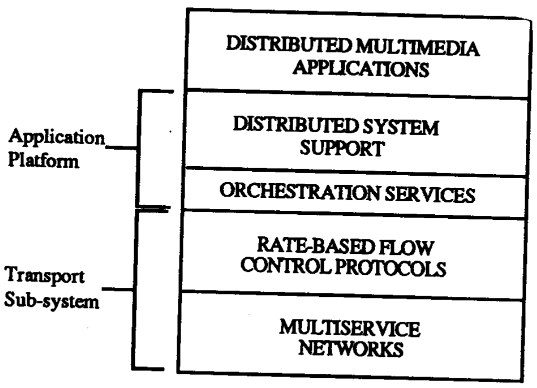

DISTRIBUTED MULTIMEDIA APPLICATIONS
DISTRIBUTED SYSTEM SUPPORT
ORCHESTRATION SERVICES
RATE-BASED FLOW CONTROL PROTOCOLS
MULTISERVICE NETWORKS
Application Platform
Transport Sub-system

Fig. 1. The Lancaster multimedia architecture.

III. The Lancaster Approach

The research at Lancaster has concentrated on the development of an experimental distributed multimedia system which allows experimentation with distributed multimedia applications in a heterogeneous environment. This system is supported by our transputer-based *multimedia network interface* (MNI) unit described below. This section describes both our architectural approach and the implementation strategy we have adopted for the MNI unit.

A. Architectural Approach

1) Overall Architecture: The architecture of the experimental distributed multimedia system is shown in Fig. 1. Distributed multimedia applications view an object-based distributed application platform, known as the *base services platform* [15]. The platform isolates applications from the complexities of multimedia devices and continuous-media communications by providing high-level run-time services to which applications can dynamically bind and access.

The platform runs over an experimental distributed infrastructure consisting of multimedia workstations and a real-time high-speed network emulator. The multimedia workstations are, in fact, standard Unix or PC machines augmented with MNI units which attach to these standard workstations and provide hardware support for the entire software infrastructure described in this section (see Section III-B-1 for a full description of the MNI hardware).

As shown in Fig. 1, the distributed system support layer is, in turn, supported by a transport subsystem. This includes an experimental transport protocol [37], [6] specifically designed for continuous media, and a set of *orchestration services.* The purpose of the orchestration services is to provide synchronization support for continuous-media applications. Support is provided for the two categories of synchronization identified in Section II, i.e., continuous synchronization and event-based synchronization. The orchestration services are visible in the application platform, but much of the mechanism of orchestration is carried out in close association with the rate-based transport protocol. The various components in this architecture, i.e., the base services platform, orchestration services, and transport protocol, are described in the following subsections.

2) Distributed System Support: The distributed system support is based on the ANSA architecture [3]. ANSA was chosen as the starting point for the systems support infrastructure because of its prominence in the ISO standardization activity in Open Distributed Processing (ODP) [26].

The Lancaster base services platform [15] implements a set of generic base objects which encapsulate continuous-media services provided by the lower layers. The platform consists principally of three types of object: *devices, streams,* and *synchronization managers.* These encapsulate the control and transmission of continuous media, but are seen by the higher layers as abstract data type (ADT) services. *Groups* also play an important role in our distributed systems model, and interfaces may be grouped either to allow the transmission of continuous-media data or to allow invocation of a number of interfaces at one time. We now describe the platform object types in more detail.

Devices are an abstraction of physical devices, pieces of stored continuous media [23], or software processes. Video windows are also modeled as device objects. Devices may be either sinks, sources, or *transformers* of continuous-media data. Examples of transformers are processes which perform compression or data format conversion. Devices present the following pair of interfaces:

- *A generic control or chain interface:* A piece of continuous media is visualized as a *chain* comprising a sequence of *links,* each of which represents an atomic unit particular to the media type in question (for example, a frame of video).[1]
- *A device-dependent interface:* The device-dependent interface contains operations specific to the device. For example, cameras have operations such as *pan* and *tilt.*

The chain interface is common to all continuous-media devices. It contains operations to selectively set up the device to produce or consume continuous-media data, and to switch the information flow on and off. Sequences of stored media do not have a device-dependent interface for control; they are only visible through the chain interface and, hence, are often referred to simply as *chains.* A virtual pointer moves through the chain as it is played or recorded, and this pointer may be located and moved using additional operations available in the chain interface.

Using the chain interface, clients of a device may create an *endpoint interface* on the device. This interface abstracts over all aspects of a device which are concerned with the transport of continuous-media data (essentially, it presents a pair of operations, *get_link* and *put_link,* through which links can be read or written).

Streams are services which are used to connect devices together via their endpoint interfaces. They are abstractions of continuous-media transmissions and map onto an underlying transport protocol. The stream interface contains operations

[1]Chains are an extension of the voice ropes abstraction developed for the Etherphone project and may be edited using similar operations (e.g., concatenate chains form a subchain etc.).

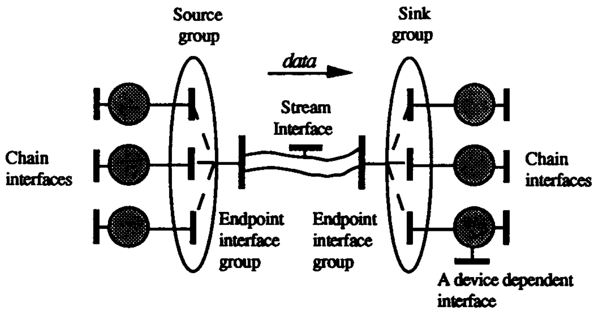


Fig. 2. Using streams to connect endpoint interface groups.

which allow the client of a stream service to connect and disconnect endpoint interfaces. Streams support arbitrary m : n connections, i.e., they allow m sources to be connected to n sinks. This is modeled by allowing endpoint interfaces to be grouped together by means of the interface group mechanism mentioned above (see Fig. 2). Endpoint interfaces may be dynamically added to or removed from these groups.

Synchronization Managers offer a service to which all application synchronization requirements may be delegated. Requirements for event-driven synchronization are expressed as an interpreted script which specifies actions to be taken on the basis of events generated by base service objects such as devices or streams. Continuous synchronization is specified according to parameters such as the packet ratio between the related streams, the tightness of synchronization required, and permissible actions to take in order to regain lost synchronization (such as packet discard, increased throughput, etc.). The synchronization manager makes use of *orchestrator* processes (described in Section III-A-4) to implement continuous synchronization.

3) Transport Subsystem: Traditional transport protocols such as TCP or OSI-TP4 are designed to work over an older generation of communication infrastructure which did not provide high-speed and multiservice capabilities [16]. Protocols of this generation tend to be complex in both design and implementation and, thus, limited in their throughput capabilities [14]. Although it has been demonstrated that it is possible to produce efficient implementations which are able to handle high throughputs [19], [43], such protocols still lack the support for many of the requirements of emerging distributed multimedia applications [22]. More recently, new protocols have been developed to handle high throughputs and exploit the resilience of the underlying high-speed networks [10], [11], [13] but, again, these do not directly meet the demands of distributed multimedia applications.

Due to these factors, we decided to design and implement a new transport service specifically for the support of continuous media [37]. We now outline some of the principal features of our transport service and protocol. The service offers multipeer-to-multipeer connection-oriented services. Connection-oriented services are necessary for continuous-media support as QOS guarantees must be upheld not only on an end-to-end basis but also hop-by-hop when traversing

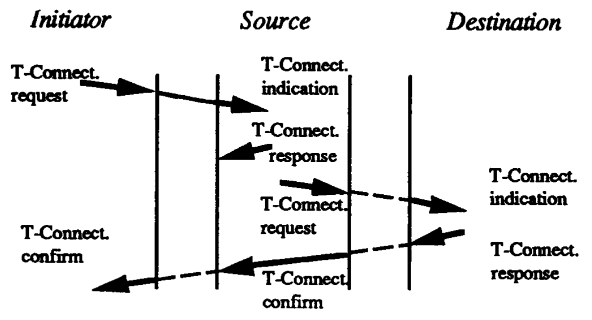


Fig. 3. Remote connection establishment.

an internet.

a) Connection establishment and release: Connection establishment is a fully confirmed three-way handshake. The QOS parameters, which undergo full-option negotiation at the transport level are: throughput (expressed in TSDU's/second, and maximum size of TSDU), end-to-end delay, delay jitter, and packet error rate. For each of these parameters, it is possible to express preferred, acceptable, and unacceptable tolerance levels. Once a connection has been accepted, the agreed tolerance levels are guaranteed for the duration of the connection. The transport service user can also define the *class of service* required. This permits the user to choose between error control, error reporting, or combined control/reporting options.

Three addresses are provided with connection primitives to cater for *remote connects.* This facility permits a management entity to set up a connection between two remote TSAP's. The initiator address represents the caller of the service and the destination and source addresses represent the two endpoints to be connected. All management indications which are issued during the lifetime of the connection are passed to the initiator as well as the source. Fig. 3 illustrates the implementation of the remote connection facility. Note that if a conventional connection-oriented service is required, the caller simply sets the initiator to be the same as the source address.

b) Notification of QOS degradation: If the class of service selected from the transport protocol for a particular connection includes error reporting, a primitive is employed to convey to the transport user any errors or QOS degradations which may have occurred. This primitive is generated by the transport entity which monitors the connection over a suitable sample period. The primitive indicates the negotiated QOS tolerance levels, the sample period, and the measured performance of the negotiated tolerance levels within that sample period.

c) QOS renegotiation: To deal with QOS renegotiation, a fully confirmed service is employed with similar options to those provided by the connection establishment service. The primitive takes parameters which include the new QOS parameters to be applied. The transport protocol may need to tear down a connection and open a new one in order to supply the required QOS. Thus, it must be capable of sustaining state so that it can resynchronize once the new connection is established. If, for some reason, a modified service cannot be provided, the service request is rejected but the existing connection is *not* torn down.

d) Protocol operation: The transfer of information is flow-controlled by sending timed bursts at regular rates on the basis of burst rates negotiated between the sender and receiver. This rate-based scheme has been used in a number of other transport protocols in recent years [10], [11], [13]. In brief, the protocol operates in the following manner.

- The transmitter and receiver agree on a *burst size* representing the maximum number of packets which may be transmitted without the transmitter and receiver having to synchronize state.
- The transmitter and receiver agree on a *burst interval* which is the time required by the receiver to consume and process a full burst. During data transfer across the network, the flow of information is controlled by this burst interval. The transmitter sends a single burst (which may be made up of a number of packets) during each burst interval.
- A large "state out of synchronization" is negotiated between source and sink to allow for continuous jitter constrained throughput. This means that for a particular connection, both sender and receiver are permitted to buffer a number of bursts.
- When a burst is consumed and processed by the receiver, it immediately informs the transmitter of its current buffer capabilities. The transmitter can then use this information to decide whether to send a burst in the next burst interval.
- Blocking will only take place if the receiver has informed the transmitter that it has no buffer capacity to consume or process another burst. It is, of course, implicit that blocking will also take place if the source has no data to transmit in its buffers. Thus, the producer may block data flow by not providing data, or the receiving client may block data flow by not consuming already received data. This is referred to as *client-level flow control* and is heavily exploited by the orchestration services discussed below.
- If selective retransmission of lost or corrupted data is not required then, as soon as a burst is transmitted, the allocated buffer for that burst becomes free. Also, as soon as a burst has been consumed and processed, even in the event of errors that buffer may be consumed by the transport user.

4) Orchestration: Orchestration is introduced into the architecture to provide support for the synchronization problem identified in Section II. Focusing on the *continuous synchronization* requirement, the following functionality must be provided by the infrastructure to ensure that media flowing in separate but related connections can be correctly coordinated in time [8].

i) The ability to start related continuous-media data flows *precisely* together. If the relationship is not correctly initiated, there is no possibility of maintaining a correct temporal relationship.

ii) The ability to dynamically start and stop the flow of continuous-media information on sets of related connections together in real time.

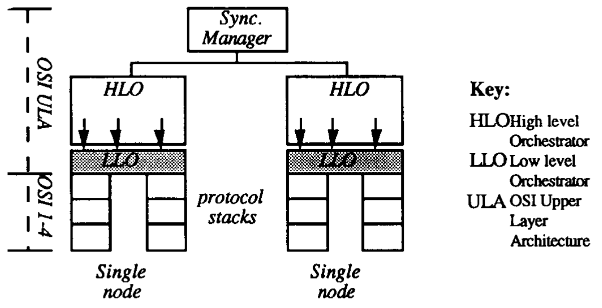

Sync. Manager
HLO
HLO
LLO
LLO
protocol stacks
Single node
Single node
OSI ULA
OSI 1-4
Key:
HLO High level Orchestrator
LLO Low level Orchestrator
ULA OSI Upper Layer Architecture

Fig. 4. Three-level orchestration architecture.

iii) The ability to create related connections with compatible QOS. This is so that the connections will *maintain* a compatible temporal transmission rate in the required ratio.

iv) The ability to *monitor* the ongoing temporal relationship between related connections, and *regulate* the connections to perform fine-grained corrections if synchronization is being lost. It is almost inevitable that related connections will eventually drift out of synchronization due to such factors as the potentially long duration of continuous-media connections in typical applications, the inevitable discrepancies between remote clock rates, and temporary "glitches" occurring in individual connections.

Our orchestration design is illustrated in Fig. 4. The *synchronization manager* provides the view of orchestration seen by users of the base services platform. The view presented is of a service interface defined as an ADT containing orchestration-related operations. Applications pass stream interfaces to these operations and the synchronization manager arranges to have the required continuous synchronization performed by the lower layers according to a *policy* which is specified by the application. Policies include constraints on how "strict" the continuous synchronization should be and actions to take on failure.

Below the platform level, the remaining two orchestration components are responsible for realizing the behavior and policy required by the synchronization manager. At this level, the orchestration process is realized as *high-level orchestrator* (HLO) objects, which monitor and regulate multiple transport connections via a *low-level orchestrator* (LLO) interface in a continuous feedback loop. For each orchestrated group of connections, there is a single HLO instance. Also, a separate LLO instance runs on all source and sink nodes of all the orchestrated connections. The HLO and multiple LLO instances interact with each other via orchestrator PDU's, on separate transport connections (actually, the HLO only interacts with a single LLO per connection). These connections must be provided with constrained latency QOS to support the necessary real-time communication of orchestration primitives.

Fig. 5 illustrates the interaction between the HLO and a LLO instance. The HLO supplies the LLO with *rate targets* for each orchestrated connection over specified *intervals.* These targets ensure that each orchestrated connection runs at the required rate, relative to a global reference clock, for the required synchronization relationship between the orchestrated

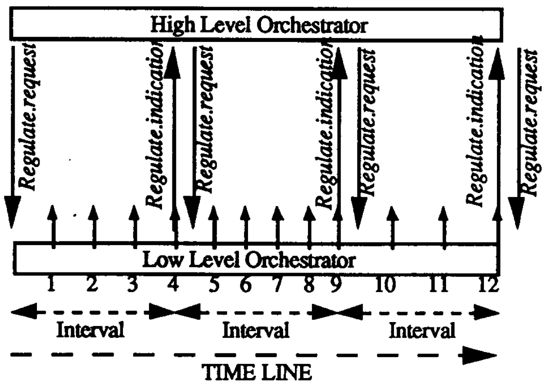


Fig. 5. Interaction between HLO and LLO.

connections to be maintained. The LLO attempts to meet the required rate target over each interval for each connection, and reports back at the end of the interval on its success or failure. Then, on the basis of these reports, the HLO sets new targets for the next interval which compensate for any relative variation in speed among the orchestrated connections. The LLO operates on a *best effort* principle; it is the responsibility of the HLO to take appropriate action (e.g., set new targets or renegotiate the connection QOS) if the LLO consistently fails to meet targets.

The LLO orchestration interface consists of two sets of primitives. The first set operates over a grouping of transport connections and provides the ability to *prime, start,* and *stop* the flow of data in these connections both atomically and instantaneously. The prime primitive is worthy of special mention: it is used to prefill the receiver's buffers so that a subsequent start can take effect immediately (with minimal latency). The second set of primitives operate on single-transport connections in an orchestrated grouping. They enable the controlling HLO to set and monitor the above-mentioned flow rate targets. Primitives are also provided to report back to the HLO agent on the actual performance achieved at the end of each interval.

Further details of the design and implementation of the orchestration system are provided in [8].

B. Implementation Strategy

1) Overall Approach: Lancaster's Multimedia Network Interface (MNI) unit [35] is designed to provide multimedia capability on a wide range of workstations (including machines such as IBM-PC compatibles) via a host-independent hardware and software extension. The MNI is an autonomous unit, and the only host interaction aside from normal data communication is the control information necessary to initiate and supervise continuous-media connections. All such control information is communicated to the MNI unit via the base services platform which runs on the MNI. Once a multimedia connection is established, continuous-media transfers are under the direct control of the MNI, with no data being streamed over the host bus subsystem. The implication is that a direct link must be established between the I/O devices and the network,

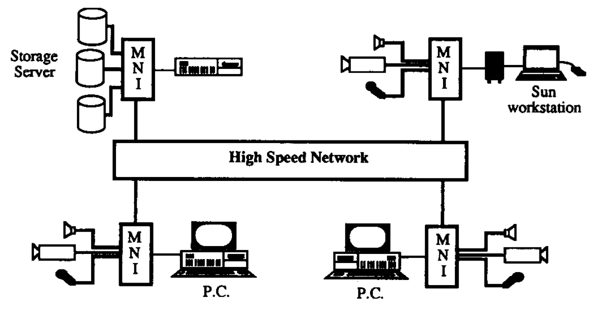


Fig. 6. The role of the MNI unit.

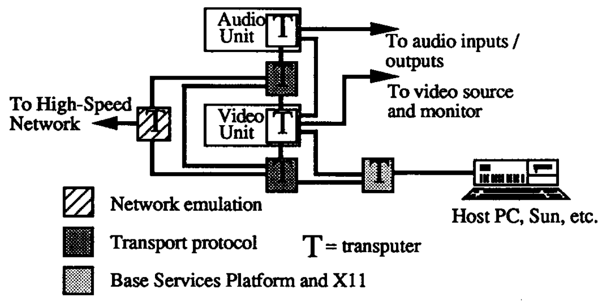


Fig. 7. Schematic of the multimedia network interface.

allowing data to be transferred to and from the network with no host intervention. The role of the MNI unit is illustrated in Fig. 6.

Each MNI unit is composed of a cluster of six transputers as illustrated in Fig. 7. A transputer-based approach was chosen as an intermediate step toward full hardware realization: transputers have the flexibility of a general-purpose processor-based solution while approaching the performance of a hardware-based solution. A further advantage of a transputer-based approach is the easy interfacing to commercial cards, offering a wide range of facilities including graphics and signal processing, via the standard transputer link interface.

The MNI includes host-independent audio and video I/O devices providing multimedia capabilities for the host machine (see Fig. 7). It also runs the Lancaster transport and orchestration services, the base services platform, and an X-windows server adapted to incorporate video windows. We now describe the various hardware and software aspects of the MNI unit in more detail.

2) Base Service Platform: The base service platform described in Section III-A-2) above is implemented on the transputer directly connected to the host workstation. The platform is written in parallel C and exploits the lightweight threads provided in the language run-time system to improve the throughput of RPC calls from applications. It is possible to transparently access services in the distributed environment whether they reside on the host workstation, on an MNI unit, or on any of the other nodes (primarily Unix workstations) in the distributed system. The platform is implemented as three major processes acting as "stubs" communicating directly with the occam device drivers and communications services which run on the other transputers. The three processes are as follows.

- A *device* process which supports instances of the platform's *device* abstraction with its chain and device-dependent interfaces.
- A *stream* process which supports instances of the stream abstraction and arranges and manages all continuous-media connections at its workstation.
- A *trader* process which acts as a database of all the ADT services at the node.

The first two processes also support *factory* interfaces which enable the creation of device and stream instances.

The distributed system support code is not identical on each MNI unit, but varies according to the role of each particular workstation. For example, the unit attached to the storage server (see Section III-B-7) runs a specialized device process which communicates with the low-level storage server code.

3) Transport Protocol Implementation: The transport services are implemented over two transputers. One of the two, designated as the master, supports the connection state tables and the low-level orchestrator functionality. The services provided by the slave transputer are initiated and managed by the master via the transputer link which directly connects the two. The transport protocol design is based on techniques which have been employed in the implementation of high-speed protocols on parallel processor architectures for exploitation of the high bandwidths of the new high-speed networks [43]. Thus, the protocol is subdivided horizontally into a send part and a receive part, which work concurrently and periodically communicate with each other for the exchange of synchronization information. Results employing these techniques for implementing conventional transport protocols such as TP4 and TCP have shown a noticeable increase in throughput performance [19].

On each transputer, we have implemented separate send and receive shared circular buffer pools which are structured to implement the queue disciplines appropriate to the various services supported. The protocol data units have fixed-size headers for processing efficiency.

4) Network Protocol Emulator: In order to investigate the effect of the various network protocols on multimedia traffic, we have designed and built a network emulator to interconnect the various MNI-enhanced workstations. The emulator is intended to achieve throughput rates equivalent to a real high-speed network, i.e., it is a real-time emulation, not a simulation. Our aim is to provide emulation of a number of multiservice networks, specifically B-ISDN [38], FDDI [27], and DQDB [25]. At the present time, we have a working emulation of FDDI.

The emulator is also implemented using transputer technology and, in hardware terms, is configured as a two-plane packet switch with a potential throughput of 40 Mb/s point-to-point. The network emulation software is layered on top of this basic switch and is implemented on the network transputer of each MNI unit. The emulator as currently configured permits four nodes to be connected.

5) Video Support: The video hardware is based around a T800 transputer and a Brooktree Bt473 RAMDAC digital/analog converter. The transputer has 4 Mbytes of memory divided into a program store and two frame buffers. The RAMDAC has full read/write access to dual-ported frame buffers allowing PAL color-encoded video to be digitized and stored at the full 25 frame/second rate used for video systems in the UK.

The X-window system was chosen to provide the graphical interface to the MNI unit because it is becoming the *de-facto* standard interface to workstations. The windowing code on the MNI is implemented in two parts. The first part corresponds to the standard X-windows "dix" and "ddx" modules and is written in C. The second part is a virtual frame buffer module written in occam which performs all screen updates for both X-window graphics and video pictures. The video and X-window servers run in tandem and exchange status information, which allows both systems to move windows around the desktop without interference.

The video server provides operations for moving, resizing, clipping, and scaling. It is configured to allow a number of processes to be extracting data from the transputer links and updating windows simultaneously and at the same time allow another process to be changing the position, size, and layout of the windows in response to commands coming from the base services device process. This is achieved by enforcing a transaction policy allowing atomic updates and reads of the window server's data structures.

In loopback mode, where images from the local camera are displayed on the local screen, users can become confused as they expect to see themselves as if in a mirror. This disorientation has been overcome by providing a facility to flip the image displayed in any window, thus making it into a mirror image rather than the "true" image transmitted by the camera. This transformation is performed in real time as data is received from the network and transferred to the window. The software allows mirror and true images to be displayed in different windows connected to the same source stream.

6) Audio Support: The audio unit comprises three digital/analog converters together with an Inmos T222 transputer for control. All the memory required for program space and audio sample buffering is provided by the T222's internal 2 Kbytes RAM. Three audio channels are provided which are usually configured as a single low-quality voice channel and two high-quality channels for CD quality stereo sound reproduction.

As with the video system, all samples are transferred via the transputer links but, unlike video, even minor errors or variation in transmission time (jitter) can cause an unacceptable loss in audio reproduction quality. This implies that sizes of buffers used must be large enough to contain enough samples to allow efficient use of the transputer links and also to allow the D/A converters to be continuously fed while the transmission of the next sample buffer is taking place.

7) Storage Server: One node in the network is configured as a digital storage server [30]. This uses the same basic MNI unit to provide a real-time continuous-media storage system and is currently configured with 1.2 Gbytes of on-line storage on which a number of video and CD quality audio streams can be stored and retrieved from disk simultaneously. The server is built from three winchester disks, one of which serves as a directory disc. The remaining two disks store video and audio data files using a parallel striping technique. Typical media storage operations such as fast forward, seek, slow motion, etc. are provided allowing flexible access to recorded music and television programs. The storage server appears on the network as a standard MNI unit capable to sourcing and sinking standard continuous-media streams. At the base services platform level, each stored file on the server is individually visible as a device with a standard chain interface.

IV. Experiences

We feel that the MNI approach of providing workstation-independent support for continuous media has been successful. The requirements imposed on the host workstation are minimal, which means that it is possible to add continuous-media functionality to any host. The hardware connection is simply by means of a standard serial connection, and the software interface only requires that the host operating system run the base services software.

The use of occam device driver code on dedicated processors is an effective way of ensuring the necessary real-time guarantees of continuous-media processing. The static nature of occam, together with the lack of operating system overhead, means that the timing of processes can be precisely predicted and managed.

An important aspect of our design is its open systems approach. Applications view the functionality offered by the MNI unit as a well-defined set of services which are integrated into a general-purpose open distributed system architecture. The base services platform infrastructure is closely related to emerging ISO Open Distributed Processing (ODP) standards and permits MNI functionality to be remotely accessed from any part of an ODP-like distributed system. No special software is required to make use of this functionality, as all the continuous-media-related functionality has been encapsulated behind standard ADT interfaces.

The unit also takes account of relevant standards in the lower layers. The transport protocol is an evolution from the current OSI standards and our work in this area is actively contributing to forthcoming OSI standardization through our involvement in the European OSI 95 project [32].

A. Sample Applications

The platform, and its underlying support, have been validated by building a number of sample applications. For example, we have built a videophone application with an X-windows user interface which runs over both the network emulator and an attached Ethernet. The workstations on the Ethernet have an identical base services interface but this is supported by the standard audio facilities of Sun SparcStations together with VideoPix frame-grabbing boards, and hence the performance is limited in comparison with the MNI implementation. We have also implemented a multiuser video jukebox and disc jockey console.

In addition, we have been involved in application studies with end-user organizations who are looking to distributed

TABLE I
PERFORMANCE FIGURES FOR THE EXPERIMENTAL CONFIGURATION

	Theoretical Maximum	With Transputer Link Protocol	With Link Extenders	With Network Emulator	End-to-End Throughput
Throughput (Mb/s)	40	29	15	11.5	4.5

multimedia systems for the solution of real-world problems [18]. After consultation with a major multinational chemical company, we are developing a microscope controller application based on the work of the company's microscopy research team. This application supports collaborative working over wide-area networks, where the object of collaboration is slow-scan digital video output from scientific microscopy. It is also possible to create multimedia documents which contain video clips, text, and video annotation.

B. Current Limitations

Although the current MNI unit and base services platforms have been successful, there remain a number of problems which we would like to address in a future version of the system. The major limitation is a lack of flexibility in terms of building services. Currently, the set of services available on the MNI unit is static and it is impossible to dynamically add new services (e.g., compression modules) to the system without reassigning processes to transputers and reconfiguring the software. We see the need for closer *integration* between the hardware, communications environment, and systems support layers so that continuous media becomes an information type which is as freely manipulable as text or numerical data.

The major drawback with the transputer approach is the relatively disappointing performance in comparison to a dedicated hardware solution. We are relying on transputer links for all communication and these have proved to be a bottleneck. Although the links have a theoretical bandwidth of 20 Mb/s, this is considerably reduced in practice due to an embedded protocol for reliable link transfer. In addition, we use physical link extenders between workstations and the network emulator and this further reduces the available link throughput. Another system bottleneck is the switching speed of the network emulator. However, as the two-plane transputer-based switch is a relatively isolated and self-contained component, it is an ideal candidate for replacement by more specialized hardware.

An indication of the performance achieved with the current configuration is given by the following figures. An end-to-end video stream rate of 12 frames per second with frames of 128×180 pixels is obtained for a point-to-point connection. This represents an end-to-end throughput (measured at the transport service interface) of about 4.5 Mb/s. The point-to-point figure attainable over the bare FDDI network emulator without the transport-layer overheads is 11.5 Mb/s, and the raw throughput of the switch itself (with the link extenders) is 15 Mb/s. These figures are summarized in Table I, which shows the performance degradation caused by the introduction of the various components.

The significant overhead of the transport protocol is largely explained by our implementation techniques, i.e., the communications protocols are implemented as packet trains passed along a pipeline of processes which each represent a protocol layer. The occam programming language encourages such modular process-based programs. However, the overhead in message passing is considerable. This is unavoidable when messages are passed across physical links but unfortunate where messages are being passed between processes running on the same transputer. The strength of occam is that it has allowed us to implement naive, rapid prototypes which can later be reimplemented using shared memory communication and/or dedicated hardware to improve performance.

V. PLANNED DEVELOPMENTS

The major aim of our future work with the base services and MNI unit is to improve the levels of performance, flexibility, and integration. Our approach to these goals will be based on an enhanced hardware configuration for the MNI unit, an early design of which is described in [36]. The new unit will offer significant performance improvements through the use of specially developed hardware and, in particular, a 32-bit minicell switch developed in-house, called the LANC (local area network on a chip). The whole system, from network to device, is based on small fixed-sized cells allowing relatively simple interfacing to cell-based networks such as ATM systems.

With this new configuration, it will be possible to enhance the flexibility of the system. Our intention is to run a microkernel operating system, specifically Chorus [7] on both the host workstation and the processors attached to the LANC. In this environment, a programmer could write a filter process for, say, real-time video and transparently run it on a processor attached to the LANC where it would receive real-time processing resouces. Such filters will appear in the base services layer as transformer devices with standard base-service interfaces and, thus, be made available in the open distributed environment. Because such devices are not bound to a particular processor, they can be dynamically loaded on demand when their service interfaces are imported from the system trader. Not all the processors need run a microkernel; it will be perfectly possible to include specialized LANC nodes whose software runs on the bare processor as with our current configuration.

The introduction of an operating system kernel to improve flexibility also raises new requirements. In particular, suitable scheduling mechanisms are required so that continuous media may be isochronously processed in a dynamic environment where such processes may be continually created and destroyed. Current microkernels, although they make claims of real-time support, usually provide priority-based scheduling together with features such as page locking and system call timeouts. For continuous-media support, however, deadline-based priority policies [20] are more appropriate. Another requirement is that the scheduling mechanism is properly integrated with the transport services so that threads can be efficiently scheduled on the basis of incoming messages via upcalls and a carefully designed buffer management scheme.

VI. Related Work

Commercial workstations now available, such as the Sun SparcStation and DECstation 5000 series, are capable of basic multimedia support. Typically, they have conventional bus-based architectures, and many of them have limited audio capabilities and sufficient internal bandwidth to allow small video windows to be produced using processor-driven bitmap copying. To enhance video handling capabilities, systems in this class are often supplemented by video capture cards, for example Sun Microsystems' VideoPix card, which can capture slow-scan video images.

These systems also benefit from recent advances in compression technology as employed in the Intel DVI and Philips CD-I technologies. For example, the latest version of CD-I enables full-screen video to be spooled from a hard disk and viewed on a standard PC screen. However, despite these advances, bus-based systems still suffer from their dependence on standard workstation architectures and are essentially stand-alone systems with limited multimedia processing capabilities and no high-speed digital network support.

An extension of the standard bus-based approach, which has been adopted by some researchers, is to provide a dedicated high-speed bus [28], [34] to interconnect multimedia devices. This provides a clear separation of the multimedia and conventional data, and thus allows better performance. The use of a separate high-performance bus targeted at multimedia support also allows designers to produce faster interfaces to high-speed networks. Our MNI approach, using interconnected transputers, extends this idea and provides independent optimized paths from each of the multimedia devices to the network interface.

An example of nonmultimedia-specific high-speed network interfacing for existing workstations is provided by work in the Aurora project [17], [41]. The Aurora interfaces offer considerable performance improvements over conventional interfaces by off loading segmentation and reassembly tasks from the host. The Nectar system [4] extends this approach to include sophisticated protocol processors, which also aim to off load the transport protocol handling from the main workstation processor onto the network interface itself.

The system bearing closest comparison with the MNI unit is Pandora's Box, which was developed by Olivetti Research [24]. Pandora is an add-on unit for Unix workstations and offers comparable functionality to the MNI unit. The Pandora and MNI systems are more than simply protocol processors; they offer full end-to-end multimedia support.

Whereas the first-generation MNI unit relies mainly on off-the-shelf components, the Pandora is constructed from specially-developed hardware subsystems. The unit consists of five transputer-based modules interconnected by FIFO queues rather than transputer links. The five modules consist of a video capture card, a video mixer card, a host interface, a main controller/server unit, and a network unit for the Cambridge Fast Ring.

Compared to Pandora, the MNI places greater emphasis on high-level software support. Pandora provides only a simple, proprietary packet protocol for control by applications whereas the MNI includes an integrated X-server, continuous media transport service, synchronization services, and a high-level distributed platform interface. The MNI also offers greater support for heterogeneity as it provides all the multimedia support internally and places fewer demands on the host machine. Finally, there are differences between the two systems in their approach to video mixing: the Pandora unit uses analog mixing whereas the MNI uses fully integrated digital mixing. This improves the degree of integration at the application level by allowing application programs to manipulate snapshots of live video through standard cut and paste operations.

VII. Concluding Remarks

This paper has described an approach to the problem of handling continuous media in heterogeneous distributed systems. The approach is centered on a transputer-based multimedia network interface unit which manages multimedia services in a distributed environment. To cater for heterogeneity, the unit provides an ODP-compatible interface. In addition, the available services adhere as closely as possible to the framework of standards such as OSI and X-Windows. The approach has been validated by a number of pilot applications, including a collaborative work scenario derived from a collaborative project with a major chemical company.

Our experiences with the experimental systems have been largely positive. The use of the interface has enabled a variety of workstations to be integrated successfully into a distributed multimedia system. However, although the network interface guarantees the required real-time behavior to support continuous media, the following problems and issues remain. First, the current implementation does not allow dynamic instantiation of multimedia services by applications. Second, the performance of transputers, while sufficient for experimentation, is not sufficient to deal with a realistic level of continuous media traffic. Based on these observations, various extensions are now being designed to enhance the level of performance, flexibility, and integration of the multimedia network interface. Finally, we have become convinced of the usefulness of QOS as a unifying theme, and have started to develop a QOS architecture which offers integrated QOS handling at each system layer [9].

Taking a broader view, our design can be viewed as one step in an evolution of network interface designs. Through the successive stages of programmed I/O, DMA transfers, dedicated protocol processors, and multimedia network interfaces, there has been an increasing trend towards autonomy and additional functionality. However, it has become clear from our research that there must be a high degree of software integration at the base services level. Our design for the next-generation MNI, outlined in Section IV-D, aims to provide better support for such integration: all the low-level complexity is contained within a single integrated circuit, providing a simple interface to multimedia devices. All the system components have equal status and are connected by means of a fixed-size minicell switch. Integration is also improved at the software level through the use of a distributed microkernel.

If this approach is pushed to its logical conclusion, a totally integrated design may be envisaged where the intraworkstation interconnection is identical to the interworkstation networking technology. In fact, proposals of this kind have been made in the literature. For example, [21] suggests an ATM-based design where the universal unit of interchange is the ATM cell. In this design, all the processors within a host are interconnected by a *desk area network*, which is part of a local ATM network that may be attached to a wide-area ATM network. I/O devices, such as cameras and video displays, are treated as producers/consumers of ATM cells.

Although this is an appealing viewpoint, and we agree that it is a plausible scenario for future systems, we consider that there are disadvantages in taking the idea to the extreme of total ATM integration. The major drawback is that a design in which every processor is a network host means that each processor must do its own protocol processing through all protocol layers. Also, pure ATM within the workstation precludes the advantages that special-purpose interconnects may bring, for example, the use of highly parallel buses and efficient caches. Finally, we contend that the 53-byte cell adopted for ATM standards is too large a granularity for intraworkstation interconnection. In the new MNI design, we have opted to use a word (of 32 bits) as a smaller unit of transfer targeted towards simple hardware devices.

As a consequence of these arguments, we project that the architecture of future multimedia workstations will be based around a switch interconnect, but that there will still be a place for a network interface processor which mediates between the cells used internally and externally. Such a design allows local nonstandard interconnect optimizations to be made and enjoys the additional advantages of scalability and network independence.

Acknowledgment

The authors would like to thank the research assistants and students at Lancaster who contributed to the developments described in this paper. Special thanks are due to F. Ball, N. Davies, P. Lougher, and N. Williams.

References

[1] M. Accetta, R. Baron, D. Golub, R. Rashid, A. Tevanian, and M. Young, "Mach: A new kernel foundation for UNIX development," Tech. Rep., Dept. Comp. Sci., Carnegie Mellon Univ., Aug. 1986.

[2] D. P. Anderson, S. Y. Tzou, R. Wahbe, R. Govindan, and M. Andrews, "Support for continuous media in the DASH system," in *Proc. 10th Int. Conf. Distrib. Comput. Syst.*, Paris, France, May 1990.

[3] Advanc. Netw. Syst. Arch., *ANSA Reference Manual Release 01.00.* Cambridge, UK: Architecture Projects Management Ltd., Mar. 1989.

[4] E. A. Arnould, F. J. Bitz, E. C. Cooper, H. T. Kung, R. D. Sansom, and P. A. Steenkiste, "The design of Nectar: A network backplane for heterogeneous multicomputers," *Comput. Arch. News*, vol. 17, no. 2, pp. 205–216, Apr. 1989.

[5] A. Banerjea and B. A. Mah., "The real-time channel administration protocol," in *Proc. 2nd Int. Workshop Netw. and Operat. Syst. Support for Digit. Audio and Video*, Heidelberg, Germany, 1991.

[6] G. Blair, G. S. Coulson, F. García. D. Hutchison, and D. Shepherd, "Towards new transport services to support distributed multimedia applications," presented at the *4th IEEE COMSOC Int. Workshop Multimedia Commun.*, Monterey, CA, 1992.

[7] A. Bricker, M. Gien, M. Guillemont, J. Lipkis, D. Orr, and M. Rozier, "Architectural issues in microkernel-based operating systems: The CHORUS experience," *Comput. Commun.*, vol. 14, no. 6, pp. 347–357, July 1991.

[8] A. Campbell, G. Coulson, F. García. and D. Hutchison, "A continuous media transport and orchestration service," presented at *ACM SIGCOMM '92*, Baltimore, MD, Aug. 1992.

[9] A. Campbell, G. Coulson, F. García. D. Hutchison, and H. Leopold, "Integrated quality of service for multimedia communications," to be presented at INFOCOM'93.

[10] D. R. Cheriton, "VMTP: A transport protocol for the next generation of communication systems," in *Proc. SIGCOMM '86*, pp. 406–415.

[11] G. Chesson, "XTP/PE overview," in *Proc. 13th Conf. Local Comput. Netw.*, Minneapolis, MN, 1988, pp. 292–296.

[12] C.-H. Chow, "Achieving multimedia communication on heterogeneous networks," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, Apr. 1990.

[13] D. D. Clark, M. L. Lambert, and L. Zhang, "NETBLT: A high throughput transport protocol," *Comput. Commun. Rev.*, vol. 17, no. 5, pp. 353–359, 1987.

[14] D. D. Clark, and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," *Comput. Commun. Rev.*, vol. 20, no. 4, pp. 200-208, 1990.

[15] G. Coulson, G. S. Blair, N. Davies, and A. Macartney, "Extensions to ANSA for multimedia computing," *Comput. Netw. and ISDN Syst.*, vol. 25, pp. 305–323, 1992.

[16] A. S. Danthine, "Communication support for distributed systems OSI versus special protocols," *Protocols for High-Speed Networks.* Amsterdam, The Netherlands: Elsevier Science, 1989, pp. 181-190.

[17] B. S. Davies, "A host-network interface for ATM," in *Proc. SIGCOMM '91*, Zurich, Switzerland, Sept. 1991, pp. 307–315.

[18] N. Davies, G. Coulson, N. Williams, and G. S. Blair, "Experiences of handling multimedia in distributed open systems," in *Proc. SEDMS '92: Usenix Symp. Distrib. and Multiproces. Syst.*, Newport Beach, CA.

[19] D. Giarrizzo, M. Kaiserwerth, T. Wicki, and R. C. Williamson, "High-speed parallel protocol implementation," in *Protocols for High-Speed Networks.* Amsterdam, The Netherlands: Elsevier Science, 1989, pp. 165-180.

[20] R. Govindan and D. P. Anderson, "Scheduling and IPC mechanisms for continuous media," in *Proc. 13th ACM Symp. Operating Syst. Principles*, Pacific Grove, CA.

[21] M. Hayter and D. McAuley, "The desk area network," *ACM Operat. Syst. Rev.*, vol. 25, no. 4, pp. 14–21, Oct. 1991.

[22] D. B. Hehmann, M. G. Salmony, and H. J. Stuttgen, "Transport services for multimedia applications on broadband networks," *Comput. Commun.*, vol. 13, no. 4, pp. 197–203, 1990.

[23] R. G. Herrtwich, "Time capsules: An abstraction for access to continuous media data," in *Proc. IEEE Real-Time Syst. Symp.*, 1990.

[24] A. Hopper, "Pandora—An experimental system for multimedia applications," *Operat. Syst. Rev.*, vol. 24, no. 2, Apr. 1990.

[25] "IEEE 802.6 Distributed Queue Dual Bus—Metropolitan Area Network," IEEE Draft Stand. Vers. D15, Oct. 1990.

[26] Int. Standards Org., "Basic reference model of open distributed processing," ISO/IEC JTC1/SC21 Working Doc N4022, Nov. 10, 1989.

[27] M. J. Johnson, "Reliability mechanisms of the FDDI high bandwidth token ring protocol," *Comput. Netw. and ISDN Syst.*, vol. 11, pp. 121-131, 1986.

[28] A. T. Kundig, "Future computer and communication supported working environments," in *Research into Networks and Distributed Applications: EUTECO '88.* North Holland: Apr. 1988, pp. 27–37.

[29] T. D. C. Little and A. Ghafoor, "Synchronization properties and storage models for multimedia objects," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 413–427, Apr. 1990.

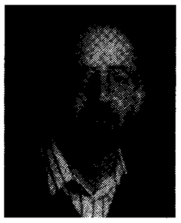
[30] P. K. Lougher and D. Shepherd, "The design of a storage server for continuous media," *Comput. J.*, vol. 36, no. 1, Feb. 1993.

[31] C. Nicolaou, "An architecture for real-time multimedia communication systems," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 391–400, Apr. 1990.

[32] Esprit Proj. 5341, "High performance OSI protocols with multimedia support on HSLAN's and B-ISDN (OSI 95)," *Bull. SA*, Route de Versailles, Louveciennes, Paris, France.

[33] G. Schurmann and U. Holzmann-Kaiser, "Distributed multimedia infor-

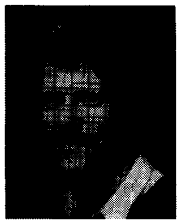
mation handling and processing," *IEEE Network Mag.*, pp. 23–31, Nov. 1990.
[34] A. Sciarappa, "SOMIW—A multimedia workstation with real time capabilities in a public network (ISDN)," in *Proc. IEEE GLOBECOM*, Houston, TX, 1986, pp. 479–483.
[35] A. C. Scott, F. Ball, D. Hutchison, and P. Lougher, "Communications support for multimedia workstations," in *Proc. 3rd IEE Conf. Telecommun., (ICT '91)*, 1991.
[36] A. C. Scott, W. D. Shepherd, and A. Lunn, "The LANC—Bringing local ATM to the workstation," to be presented at *4th IEE Telecommun. Conf.* (ICT '93), Manchester, UK, Apr. 1993.
[37] W. D. Shepherd, G. Coulson, F. García, and D. Hutchison, "Protocol support for distributed multimedia applications," *Proc. 2nd Int. Workshop Netw. and Operat. Syst. Supp. for Digital Audio and Video*, Heidelberg, Germany, 1991.
[38] D. R. Spears, "Broadband ISDN service visions and technological realities," *Int. J. Digit. and Analog Cabled Syst.*, vol. 1, pp. 3–18, 1988.
[39] R. Steinmetz, "Synchronization properties in multimedia systems," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 401–412, Apr. 1990.
[40] C. Topolcic, "Experimental internet stream protocol, Version 2 (ST-II)," Internet Req. for Comments No. 1190, Oct. 1990.
[41] C. B. S. Traw and J. M. Smith, "A high performance host interface for ATM networks," in *Proc. SIGCOMM '91*, Zurich, Switzerland, Sept. 1991, pp. 317–326.
[42] N. Williams, G. S. Blair, and R. A. Head, "Multimedia computing: An assessment of the state of the art," *Inform. Services and Use*, to appear.
[43] M. Zitterbart, "High-speed transport components," *IEEE Network Mag.*, pp. 54–63, 1991.



Gordon Blair received the Ph.D. degree from Strathclyde University, Glasgow, UK, in 1983.

Since 1983, he has continued research into distributed operating systems at the University of Lancaster, initially as a Research Associate and then as a SERC IT Research Fellow. He is currently a Senior Lecturer in the Department of Computing at Lancaster. He has published extensively in his field, and is the coauthor of a book on the UNIX operating system and coeditor of a book on object-oriented systems.

Andrew Campbell has over ten years experience of developing real-time embedded microprocessor applications. Before joining the Distributed Multimedia Research Group at Lancaster, he spent five years working as a consultant with Raytheon and GTE in the U.S., developing solutions for local area networks, packet switching, and tactical communications protocols. His current research interests include the design of new high-performance protocols which can exploit the next generation of integrated services networks.



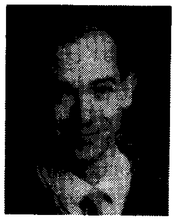
Geoff Coulson received the first-class honors degree in computer science from the University of Lancaster.

Having worked for a number of years in the medical technology field, both overseas and within the UK's National Health Service, he joined the Distributed Multimedia Research Group at Lancaster as a Research Associate in 1989. His current research interests lie in the areas of distributed systems architectures and operating systems support for multimedia communications.

Francisco Garcia received the B.Sc. (honors) degree in computer science from Lancaster University.

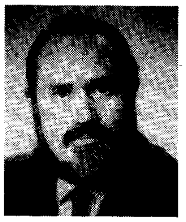
He is currently in the third year of Ph.D. work, and is now completing research in the area of communication protocol support for distributed multimedia applications. During the early part of his work, he was awarded a fellowship with British Telecom Research Laboratories on the investigation of design issues for lightweight communications protocols to support distributed applications over high-speed networks.

David Hutchison is a Professor in Computing at Lancaster University and has been involved in research in local area network architectures and distributed systems applications for the past ten years. Most of this research experience has been in collaboration with industry. He is currently an investigator in the European Commission funded ESPRIT OSI 95 and DELTA JITOL projects, and in UK SERC/industry funded projects in distributed multimedia systems and quality of service architecture. He is the author of many research publications and of a book on local area network architectures.



Andrew Scott received the first-class degree in computer science from Lancaster University.

Since graduation, he has been involved in several research projects with the Distributed Computing Group at Lancaster. The first of these projects examined the application of object-oriented computing to distributed process control applications. He has also worked on several contracts to develop industrial applications, especially in robotic and process control systems. For the past three years, he has been working on the Lancaster MNI project.



Doug Shepherd is the Director of IT and a Professor in Computing at Lancaster University. He has been working in the field of computer communications and distributed computing for over ten years and has published extensively. Recently, he was a Visiting Scientist at IBM's European Networking Center, where he was a member of a small group carrying out a feasibility study into the use of high-speed networks to support multimedia applications. He is also the Chairman of the UK SERC/DTI Communications and Distributed Systems Club.