Session VI: Projects I

Chair: Andy Hopper, Olivetti Research Center and University of Cambridge

The papers presented in Session VI dealt with four very different multimedia projects which are summarized in chronological order.

The first paper was "Design Considerations for a Multimedia Network Distribution Center" by Riccardo Gusella of Hewlett-Packard Laboratories and Massimo Maresca from the University of Genoa. In his talk, Massimo examined the issues involved in the design of a central facility called the Multimedia Network Distribution Center, which manages a number of different multimedia applications. The design revolved around issues of asymmetric communication, support for heterogeneous networks, adaptability to changes in networks or host loads as well as system integration.

Experiments were done on an Ethernet network with SUN hosts running UNIX, including the TCP/IP protocol set. Video frames were in the order of 100 Kbytes. These frames were stored in compressed form on a mass storage system or can be taken from an uncompressed source and then compressed by special hardware before being sent out onto the network. A compression factor of 15 was used. This produced a required throughput of 200 Kbytes/s for video conferencing applications. Performance results were presented for memory copying, TCP processing, UDP processing, decompression, visualization directly to the frame buffer and visualization through the X Server.

"Next Generation Network and Operating System Requirements for Continuous Time Media" was presented by Scott Stevens from Carnegie-Mellon University. Scott highlighted the requirements for building complex multimedia applications in which different elements of the application can be combined in various ways. The proposed technique involves using fine-grain elements and a rule-based system with a facility to specify how these elements should be combined and displayed.

Other mechanisms are used to deal with the scaling of images, synchronization, etc. It was pointed out that in order to build such systems a more abstract method of defining the multimedia elements is needed which allows different elements to be readily connected together. Synchronization at the frame level was also proposed to allow users to specify additional responses based on a given action.

The paper "Dynamicity Issues in Broadband Network Computing," joint work with Jose Diaz-Gonzalez, Russell Sasnett and Vincent Phuah, was presented by Steven Gutfreund from GTE Laboratories. It looks at dynamicity issues for multimedia applications in a distributed environment. These issues include the movement of applications from one network to another, the migration of active applications, dynamic changes in network topology and the ability to assemble applications from a number of different sources.

The SHOWME application is a multimedia environment which deals with these issues by defining a uniform homogeneous term called an Element which can be used to describe different multimedia entities. The binding of Elements is handled by a Resource Dispatcher which provides a tuple space, similar to Linda. Tuples are used to control the multimedia facilities specified by different elements. The system supports pattern-directed binding in which the binding between elements occurs at runtime.

The last paper of the session came from Ralf Cordes of Telenorma ("Managing Multimedia Sessions on a Private Broadband Communication System," Ralf Cordes, Dieter Wybranietz, Rolf Vautz). This paper addresses the problems of multimedia applications in which connections to different servers are dynamically changing depending on the interactions of one or more clients. To provide these facilities, software systems must be geared to support a number of new features, including in-call bandwidth modification, quality-of-service negotiation, integration of servers, transaction-oriented protocols and object-oriented structuring of generic applications of service elements.

Object classes are used to define composite objects called Pages and monomedial objects called Particles as well as ways for linking and anchoring objects and specifying telecommunication service elements. Transaction support is provided via the use of an ATM network, support for multipoint communications and the use of fine-grained set-up and roll-back facilities. An advanced distributed communication system for the customer premises market was also proposed. This network will also facilitate the use of several network interfaces and is based on FDDI-II.

DESIGN CONSIDERATIONS FOR A MULTIMEDIA NETWORK DISTRIBUTION CENTER

Riccardo Gusella
Hewlett-Packard Laboratories
1501 Page Mill Rd.
Palo Alto, CA 94303 - USA

Massimo Maresca
DIST - University of Genova
Via Opera Pia 11A
16145 Genova - Italy

ABSTRACT

In this paper we consider a distributed system in which a central facility, called a Multimedia Network Distribution Center, serves a number of clients and handles simultaneously different multimedia applications. These applications include digital TV distribution, interactive TV (consisting of hypermedia techniques applied to TV), as well as collaborative applications such as virtual distributed classrooms. Clients are allowed to request services independently of each other, or groups of clients can make joint service requests.

In order to satisfy a variety of applications and to support the largest possible number of clients, a server must be able to deal with a heterogeneous environment, in which clients range from small PCs to powerful workstations with or without special hardware for compression/decompression and with different visualization throughput. Moreover, the server must be able to handle shifting network load conditions and to offer concurrent synchronous and asynchronous access.

The server is connected to a set of devices (either live or playback) that generate digital video and audio streams under its control and to a set of networks and links used to transport the streams to the clients. We define the functionality and the performance requirements for an MNDC. To verify the validity of the MNDC functionality, we have built and measured a prototype system and taken performance measurements using applications that moved video and audio data across a local-area network.

1. INTRODUCTION

A Multimedia Network Distribution Center (MNDC) is a facility in a distributed system that can be used to deliver multimedia information to a number of clients simultaneously. Because communication of multimedia information is much more demanding than data communication–not only are there real-time and synchronization issues, but there are also

new error modes, and more demanding quality-of-service requirements–the systems at the server and client ends must be closely coordinated. We can think of two classes of applications that can be categorized as MNDCs.

The first class is characterized by the *one-to-one* communication paradigm. Several users of this class of applications (clients) can be active simultaneously, but they are independent of each other and each one of them opens a point-to-point, bidirectional connection with the MNDC. An example of an application in this class is interactive video, in which client viewers of stored documentaries can choose to follow one of several different paths of the documentary at a number of predetermined points during playback. The means of making the selection–a typed command, a button click, or a voice command–is a user interface question, which we will not address here; we will simply assume that the user's choice is signaled, using the return path of the connection, to the server, which will then begin to deliver the new documentary segment to the user. A second example of an application that requires a one-to-one communication paradigm is an MNDC that provides multimedia database access. In this type of database, queries can return text, images, sounds, video clips, and other types of timed data [3]. It is different from the previous application in that the focus is on queries, making the interaction between client and server much more frequent.

The second class of applications that can be classified as MNDC is characterized by the *one-to-many* communication paradigm. In this case, several clients are concurrently active and the server, possibly on the basis of input provided by the clients, decides autonomously on the sequence of the information delivered to the clients. A simple example of this class of applications is video distribution of the type available today over CATV networks; in this application the return communication channels from the clients to the server are never used: to change video channel clients listen to different port numbers. A second example is represented by a teaching system in which the instructor is in control of the server, which in turn sends clients (students) lecture information. Occasionally, one of the clients may interrupt the lecture flow by asking a question. Rather than providing full physical connectivity and connections to all other clients, it is the server that relays the question to the other clients. The slight additional delay is a small price to pay in return for the great simplification in system design. (This is similar to what happens in large conferences when a question from the audience is repeated by the speaker before he or she provides the answer.) A third example of an application that fits this communication structure is a medical application in which the server handles the instrumentation in an operating room. This instrumentation may comprise a set of cameras directed to an ongoing surgical operation as well as a number of sensors monitoring blood pressure, heart and brain activity and other vital signs. The server directs this multimedia information to a number of medical students and a few specialists, whose task is to supervise the operation and help the surgeon during its most critical phases. The specialists may volunteer a comment, or they may be asked questions from the operating room. In either case, the server will relay the data to the various output lines as we have discussed in the previous example.

An MNDC may transmit live data, recorded data or a combination of the two within the same application. The video distribution and database applications are examples of systems that transmit recorded data. The medical application is an example of live data transmission;

the teaching system may transmit a combination of both types of data. This distinction is important as we will see that live and recorded data will be handled differently by the network.

2. General Design Principles

Before discussing specific design requirements for an MNDC, it is important to consider several fundamental characteristics of multimedia data transmission that make the design of an MNDC different from that of traditional data communication systems. First of all, the MNDC transports mainly continuous media data, which has two key properties: it consists of a sequence of messages sent at a fixed rate, and the quality of its presentation can be adapted to the varying load conditions of the network and of the hosts. In addition, the MNDC is based on an asymmetric connection between one server (or in some circumstances a limited number of servers) and a set of clients; in contrast, in other multimedia distributed systems connections among the hosts may be more uniformly distributed.

The asymmetric structure of the MNDC creates certain requirements for hosts and network. The server host must have enough power to handle a large number of continuous media channels simultaneously. One means of obtaining this power is to use special hardware devices and file systems for storing/retrieving continuous media information in real time and special hardware devices for compression/decompression. Client hosts, instead, need not be high-performance, as they will be handling a smaller number of channels: clients may be simple PCs or workstations with no special hardware. The network should be structured in a hierarchical fashion so that links closer to the server have greater bandwidth than links closer to the clients.

One common issue in the design of distributed multimedia applications, which arises in the MNDC context as well, is how the network can provide real-time communication. One approach is based on the reservation of the network and computing resources during the connection establishment in order to guarantee the performance [2], while another approach is based on the development of flexible communication systems, in which the emphasis is on the adaptability of the quality of service to the varying load conditions of the network and of the hosts [4]. The MNDC follows this latter philosophy; the only assumption about communication management is that the network is able to separate data belonging to different media. This ability to distinguish and separate various media is critical because each medium has different performance parameters and different quality-of-service demands (e.g., audio requires, in general, more stringent error handling than video).

The asymmetric structure allows identifying some specific design goals, which have to be met by distributed systems that support an MNDC. Such design goals are listed below.

1) *Heterogeneity*: the MNDC must support networks and hosts having different characteristics, performances and costs. For example, clients of different performance classes must be able to receive and present the same video stream simultaneously. The quality of the presentation in each client will depend on its capabilities.

2) *Adaptability*: the MNDC must be able to adapt the quality of the presentation to load variations in the network and the hosts. For example, when the load in a client host

grows to the point at which the client no longer can present the frames received at the rate they are produced, the client must reduce the quality of the presentation, in a manner that will be the least disruptive or perceptible to the viewer.

3) *Integrated approach*: the MNDC must handle multimedia data no differently from the way it handles regular computer data. Neither special devices for continuous media presentation nor special communication channels for continuous media transmission should be used.

3. DESIGN REQUIREMENTS

The system design requirements of an MNDC involve two sets of issues: first network architecture and communication protocols, and second, computer organization and operating system architecture to support the multimedia data traffic. Much work is in progress in these areas [7], but in this paper we will only address the issues that are relevant to the design of an MNDC, and we will start with networking issues.

While current bus bandwidths are on the order of 100 Mbytes/s and newer, higher-parallelism busses promise to be much faster, the networking community is working hard to develop large scale networks in the Gbit/s range (equivalent to 128 Mbytes/s) within the next several years. Despite the increased capacity possible with such large scale networks, however, network bandwidth will still be insufficient for the aggregate traffic that can easily be produced by several workstations generating multimedia data such as high-resolution motion video. Thus, image compression is necessary to support these applications.

The most prominent image compression standards, JPEG, MPEG, and H.261, involve transform coding techniques [12]. But these algorithms are computationally very intensive, precluding their implementation by software means if real-time performance is required. However, in the case of an MNDC, since it is conceivable that the server will be considerably more powerful than client workstations, one could exploit asymmetric compression schemes that require a considerable amount of work during compression but could use quick table lookup methods for decompression. Alternatively, especially in the case of one-to-many communication serving heterogeneous clients, hierarchical or pyramidal compression schemes appear very promising. Displaying a rough picture would be quite cheap, and more powerful receivers could obtain better images by decoding more and more subbands.

In our view, multimedia traffic will be transported by general purpose, integrated networks. Because current internet packet switching nodes do not distinguish between various types of traffic, temporary congestion in a network segment affects all traffic across the segment. Provided that network access of multimedia users is controlled so that the total amount of multimedia traffic is always below a certain bound, we claim that it is possible to satisfy the real-time properties of multimedia traffic without changing the network in any drastic way. We suggest that all that is needed is an appropriate queuing algorithm in the internetwork gateways and buffer management in the receiving host. Although a detailed description of our network architecture is outside the scope of this paper, one issue–how to map the quality-of-service requirements to the services provided by the transport protocols–is of major importance to the design of an MNDC.

Each gateway will have separate queues for high-priority traffic and low-priority traffic. The difference between the two types of queues is primarily in the jitter they introduce in the packet delivery process, which could be quite high for low-priority queues. We also assume that the network bandwidth will be much higher than the bandwidth required by a single conversation, so that transmission can proceed at speeds faster than real time even on lower priority queues.

We have classified the traffic produced by the various applications listed in the Introduction as live and playback. Since playback traffic can be transmitted ahead of the time is it required by a client, we assign playback traffic to lower-priority queues and use large buffers in the receiving clients to correct the jitter introduced by the communication system. This arrangement will allow us to reserve the high-priority queues exclusively for live traffic, and, assuming that the proportion of the total live traffic is small, we can design a queue service discipline that in most cases will produce small delay and small jitter. We believe that this kind of network architecture and buffering scheme would produce the performance required by multimedia traffic under appropriate traffic conditions.

The second set of requirements for the design of an MNDC is concerned with computer organization and operating system architecture. Since cache memories do not help much when the flow of data is from the network interface to the frame buffer memory, a first fundamental problem is to try and avoid data paths that include main memory–by far the slowest component in today's workstation architecture: the DECStation 5000/200, the HP 9000/720, and the Sun Sparc 2 machines have respectively main memory chips with clock cycles of 100 ns, 80 ns, and 70 ns, respectively. A related computer organization problem is the position of the decompression engine with respect to the frame buffer. We claim that, in order to reduce main memory traffic and achieve the highest performance, the two should be next to each other, connected through a separate bus.

Another important issue is how to deal with the skews between the clocks of the server and those of the clients. The problem arises because the server and a client produce and consume data at the same rate, but the respective rates are determined by their own clocks, which, running at even slightly different speeds, may in the long run cause queue under- or overflows. To quantify the amount of skew, let us assume that clocks diverge no more than four seconds over 24 hours and that we transmit 30 frames per second. Then, in 60 minutes we may be off, in either direction, of up to five frames.

One simple way to deal with this problem is to have the operating system synchronize the rate of a client's clock with the rate of the clock of the server using algorithms analogous to those presented in [6]. The alternative method of letting an application do the resynchronization is not optimal because a client may have several application programs running simultaneously, all of which would have to apply the clock transformation on their own. However, since a machine may be part of an administrative domain whose clocks are synchronized independently of the clock of the MNDC, if the rate of a client's clock is changed, then the client must have two clock sources, one for supporting multimedia timed operations, the other for the regular OS time services that needs to be synchronized locally. Notice that popular time synchronization programs such as `ntp` [9] and `timed` [5] affect an operating system software variable and not the hardware device that UNIX uses to produce interval

timer interrupts for user processes requesting them.

In terms of operating system support other issues are include media synchronization, real-time scheduling support, performance. We believe that for these issues the solutions that have been proposed for general multimedia systems [1] apply as well to MNDCs.

4. EXPERIMENTS

To verify the validity of the MNDC design principles stated in Section 2 and to evaluate the feasibility and the cost of design solutions meeting the specific MNDC goals and requirements outlined in Sections 2 and 3, we ran certain video and audio distribution experiments on an MNDC testbed. In this section we concentrate on describing the video experiment, which is the most demanding one in terms of computing and communication throughput.

The testbed was an Ethernet network connected to a number of different hosts; for our experiments, we confined our analysis to Sun hosts. These hosts all run the Unix operating system including the TCP/IP protocol set. The experiment with full motion video consisted of the distribution of sequences of frames of CIF size [8] (352×288 8-bit pixels, about 100 Kbytes). In the reminder of the paper we use the term "frame" to refer to a CIF video frame.

The MNDC server reads the frames of a video stream from its mass storage or from an input device, compresses them and sends them to one client (one-to-one communication) or to more clients simultaneously (one-to-many communication) at a speed of 30 f/s (frames per second). Considering that a compression factor of 15 produces little degradation (in terms of user perception) in video-conference type video sequences, the resulting required throughput is about 200 Kbytes/s. The clients receive the frames, absorb the jitter introduced by the network by synchronizing the stream with a local timer, decompress the frames, and display them, either writing them directly to the frame buffer or using the local X window server as a virtual display.

In order to meet the design goals introduced in Section 2, namely heterogeneity and adaptability, each client must have control over the quality of the presentation at its site, while the server must structure the transmission in order to support a presentation of the best possible quality. Because of the need to be flexible and adaptive, the structure of the client subsystem is a critical part of an MNDC, and was the main focus of our study, the remainder of this section is devoted to the analysis of this issue.

4.1 BASIC ASSUMPTIONS ON THE EXISTENCE OF A TRANSPORT SERVICE

We began with the general assumption that there was "good-quality transport service". With good-quality transport service we mean that a lightweight connection [11] can be established between the server and the clients. We define lightweight connection to mean that routes are chosen at establishment time and kept fixed during the session. Once a lightweight connection is established, no error checking need be done on the packet's data segment (we accepted some corruption), only rate-based flow control techniques are used (i.e. the receiver cannot delay the sender), and no message buffering and reordering is performed. Frames or fragments of frames are transmitted in datagrams following the same routes, but neither their delivery nor their correctness is guaranteed.

As mentioned before, in a multimedia system the network should be able to separate different media, and schedule the messages of each of them independently. In the experiment, however, we waived this requirement and relied on the FCFS-based scheduling techniques of the TCP/IP protocol suite, in the current Internet. In particular, we chose to use the UDP/IP protocol, which provides the service closest to the one we desired, offering unreliable datagram delivery, as we needed, and in practice also offering ordered datagram delivery (for the datagrams that are delivered) in local or low-complexity environments, in which the routing is trivial and fixed.

4.2 MULTI-PROCESS ORGANIZATION

An MNDC client program could be logically split into three processes, running concurrently and asynchronously. The first process, called the *receiver process*, queues the received messages in a FIFO buffer called the receiver queue. The second process, called the *transformation process*, extracts the messages from the receiver queue, does the necessary transformations (e.g., decompression, recombination and possibly error correction) and copies them to another FIFO buffer, called the presentation queue. The third process, called the *presentation process*, extracts the processed messages from the presentation queue and either displays (if video frames) or plays (if audio) the data contained in such messages using the proper output drivers.

Because of the dynamically varying load in the client host, it may happen that at the time a new frame is to be presented some of the frames received must be dropped, because it is discovered that they are late. Since these frames have been already decompressed, the computing power expended on their processing is wasted.

A two-process structure reduces the chances of such wasted processing; a *receiver process* receives the incoming frames from the network interface in the receiving buffer, and a *presentation process* checks that the frame is on time and, if it is, does the decompression. Some jitter, in the form of delay variation, may be introduced locally at each client, because the decompression may not require exactly the same time for each frame and because spending a large amount of time in processing the data (the decompression of a frame using the scheme described in this section takes about 30 ms in a Sun Sparc 2) increases the probability that other Unix processes will be served during such period of time. However, our experiments show that, in terms of user perception, the delay jitter introduced by decompression at display time is negligible.

4.2.1 PROCESS ACTIVATION MECHANISM

Assuming that the MNDC client subsystem is composed of the concurrent processes described above, we must decide how these processes are to be activated. The primary alternatives are the `signal` mechanism, the lightweight process library, and different Unix processes.

We chose to use the `signal` mechanism, which makes it possible to manage asynchronous events (at a certain minimum granularity). As soon as the client program starts, it sets up two different `signals`, one to be delivered by the I/O handler at each new frame received (`SIGIO`) and another to be delivered by a local timer at fixed intervals (`SIGALRM`).

The receiver process is activated upon receiving a message, while the presentation process is activated upon receiving an interrupt from the timer.

The choice of having the presentation process activated at regular time intervals rather than by an I/O signal originated by the presentation device whenever the driver is ready to accept new data offers the advantages of uniformity in the treatment of video and audio and explicit control, at the user level, over the length of the output buffer.

4.2.2 SYNCHRONIZATION

Video frames and/or audio messages must be synchronized in order to be presented at the right time. Each message leaving the server (we use the term message to refer to both video frames and audio packets) is assigned an increasing *sequence number*, which determines univocally the relative time (with respect to the beginning of the sequence) at which the message is supposed to be presented, according to the expression:

$$presentation\_time = sequence\_number \; / \; frame\_rate + start\_time$$

5. BASIC OPERATIONS

The experimental client subsystem presented in the previous section, as well as the analysis of the type of processing that needs to be performed in the acquisition, decompression and presentation of integrated continuous media data, shows that there are a number of basic operations that must be performed on each frame by each client, and that depending on the performance of these operations in each client host, a different quality of presentation is achieved.

These basic operations are *memory copy*, *protocol processing*, *decompression* and *visualization*. We have analyzed the performance of a specific system, a Sun Sparc 2, as a representative of the class of the current-generation high-performance workstations, to understand to which extent such a class of machines is suitable for use as MNDC clients without additional hardware. Table 1 shows the results obtained; the performance results are given in bytes per second, frames per second (f/s) and compressed frames per second (cf/s) assuming a compression factor equal to 15.

TABLE 1. – PERFORMANCE OF BASIC OPERATIONS IN A MNDC CLIENT

OPERATION	MBYTES/S	FRAMES/S	COMPRESSED FRAMES/S
Memory Copy	9.7	100	-
TCP Processing	4.4	-	675
UDP Processing	6.7	-	1035
Decompression	2.8	29	-
Visualization (directly to the Frame Buffer)	8.2	85	-
Visualization (through the X server)	3.2	33	-

The performance of the *Memory Copy* basic operation was studied by trying a number of different techniques (e.g., execution of integer and double precision assignments and use of the `memcpy()` library routine) to copy a memory buffer from one area to another. The experiments were done with a buffer of very large size (4 Mbytes) in order to eliminate the effect of the cache memory. In fact, continuous-media data processing does not exhibit data locality, as each message is processed at most one time (for decompression) and then consumed by its visualization. Many researchers have recognized this fact and, as a consequence, there is a wide consensus that the architecture of workstations oriented to multimedia data processing must be improved to permit workstation to bypass the cache memory when it is not needed. The timing results are given in terms of uncompressed frames per second, because obviously this type of data movement accounts for much of the processing time.

The performance of the *Protocol Processing* basic operation was studied both for connection oriented (TCP) and connectionless (UDP) communication. In order to evaluate the throughput of TCP/IP and UDP/IP, a communication session was established between two processes in the same machine through the loopback interface, so to avoid generating any data-link layer traffic. The performance figures are only given in terms of compressed CIF frames per second, considering that it is expected that only compressed images travel in the network.

The performance of the *Decompression* basic operation was studied by adopting a space-domain intraframe compression algorithm. up analysis of each frame (only intra-frame compression). The reason of this choice is that in a heterogeneous system there may be client hosts which are not equipped with special hardware for decompression; these hosts are likely to use space domain techniques to decompress in real-time. Our algorithm processes the image sequentially in blocks of size 16x16 pixels, according to a quadtree coding scheme [10]. For each block, the variance and the mean are computed. If the variance is smaller than a prespecified threshold, which controls the compression factor, the entire block is encoded using its mean value. Otherwise, the original block is subdivided into four square subblocks and the procedure is repeated. The smallest block size of 2x2 pixels is not further subdivided. This method does not require floating point computation and can be implemented using a recursive program.

The performance of the *Visualization* basic operation was studied both considering the case in which the frames are copied directly from the main memory to the frame buffer, using the library routines made available by SunOS (`pixrect()`), and the case in which the frames are copied to the frame buffer through the X server. In this second case, the X server and the client program exchange data using shared memory, in such a way to avoid the overhead of interprocess communication.

6. DISCUSSION

Using the figures presented in the previous section, it is now possible to discuss the performance requirements of a distributed system to support the implementation of the MNDC. We consider the case of Sun Sparc 2 workstations and the case of Ethernet and FDDI networks.

In Table 2, we indicate the physical-layer throughput of Ethernet and of FDDI. It is evident that the transmission of uncompressed frames, which would require 24 Mbit/s, is not supported by Ethernet. Compressed video, instead, assuming a compression ratio of 15 to 1, can be transmitted in real time and only takes 16% of the network bandwidth. Supposing that 50% of the network bandwidth is available for multimedia traffic, up to 94 cf/s can be transmitted, corresponding to three video channels of 30 cf/s each. In contrast, again assuming that 50% of the bandwidth is available, FDDI supports the simultaneous transmission of up to two uncompressed video channels or as many as 31 compressed ones.

TABLE 2. – THROUGHPUT OF ETHERNET AND FDDI

Ethernet	1.25 Mbytes/s	12 f/s	180 cf/s	6 video channels
FDDI	12.5 Mbytes/s	120 f/s	1800 cf/s	60 video channels

Let us now examine the minimal number of operations that each video frame must undergo from the network interface to the visualization device. The first step is *reception*, which involve one memory copy of a compressed frame from the network interface to the workstation main memory; the second step is *protocol processing*, which may or may not involve the calculation of a checksum (in our experiments it does because we have used UDP); the third step is *decompression*, which may be performed either by specialized hardware or by software (in our experiments we have used software decompression); the fourth step is *visualization*, which includes the communication between the client program and the X server.

TABLE 3. –REQUIRED OPERATIONS INSIDE AN MNDC CLIENT

OPERATION	TIME IN MS
Memory Copy	0.66
UDP Processing	0.96
Decompression	34.48
Visualization through X	30.22

Table 3 shows the times required to carry out each of the steps above, as measured in our experiments. As expected, the most time-consuming operations inside the client workstation are those requiring the generation and/or processing of uncompressed frames, namely decompression and visualization. The decompression time can be reduced by using special hardware for decompression, while the visualization time is bounded by the speed of the frame buffer (30 ms per frame; see Table 1) and can be reduced by optimizing the X server path from the client program to visualization (notice from Table 1 that visualization directly to the frame buffer is almost three times as fast as visualization through the X server) and/or by adopting a faster frame buffer.

7. SUMMARY

We have presented the architecture of a distributed system in which a central facility, called a Media Network Distribution Center, serves a number of clients and handles simultaneously different multimedia applications. We have described a set of possible applications of such a system, such as interactive video, database access, video distribution and distributed classroom. We have outlined the design principles upon which a MNDC is based and we have introduced a set of specific requirements to be met.

We have then focused on continuous media data and in particular on video, taking the case of CIF video (352×288 pixels) as a case study and performing a set of experiments on video sequences of CIF frames to verify the validity of the design principles and the feasibility of proposed design solutions meeting the specific system requirements. Our experiments concerned the implementation of an MNDC in a local environment based on Ethernet networks. We focused on the client program and partitioned the client part of the MNDC system into a set of basic processing steps that must be carried out in sequence on each video frame as it moves from the network to the frame buffer. We have measured the performance of each of these processing steps, by running a set of specific experiments, and have reported the results. Although building a successful MNDC certainly requires additional basic research, on the basis of our experiments, we conclude that the MNDC architecture that we have outlined in Sections 1, 2 and 3 is a valid one.

REFERENCES

1. Anderson, D. P. and G. Homsy, A Continuous Media I/O Server and Its Synchronization Mechanism, *IEEE Computer 24*, 10 (1991), 51-57.
2. Ferrari, D. and D. Verma, A Scheme for Real-Time Channel Establishment in Wide-Area Networks, *IEEE Journal of Selected Areas in Communications 8*, 3 (1990), 368-379.
3. Fox, I. A., Advances in Interactive Digital Multimedia Systems, *IEEE Computer 24*, 10 (1991), 9-21.
4. Gilge, M. and R. Gusella, Motion Video Coding for Packet-Switching Networks – an Integrated Approach, *SPIE Conference on Visual Communications and Image Processing*, Boston, 10-13 November, 1991.
5. Gusella, R. and S. Zatti, The Berkeley UNIX 4.3BSD Time Synchronization Protocol, Computer Science Technical Report, UCB/Comp. Sci. Dept. 85/250, University of California, Berkeley, June 1985.
6. Gusella, R. and S. Zatti, The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD, *IEEE Transactions on Software Engineering 15*, 7 (July 1989.), 847-853.
7. Multimedia Information Systems, *Special Issue of IEEE Computer Magazine*, October 1991.
8. Liou, M., Overview of the px64 Kbit/s Video Coding Standard, *Communications of the ACM 34*, 4 (1991), 59-63.

9. Mills, D. L., Internet Time Synchronization: The Network Time Protocol, *RFC 1129*, October 1989.

10. Tanimoto, S. and T. Pavlidis, A Hierarchical Data Structure for Picture Processing, *Computer Graphics and Image Processing*, April 1989, 104-119.

11. Zhang, L., A New Architecture for Packet Switching Network Protocols, Ph.D. Thesis, Dept. of EECS Massachusetts Institute of Technology, July 1989.

12. Digital Multimedia Systems, *Special Issue of Communications of the ACM 34*, 4 (April 1991).