A Prototype Video Dissemination Application over ATM

George C. Polyzos and *Kirk Taylor**

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, U.S.A.

Abstract – We have developed a prototype video distribution application in an ATM environment to investigate the effectiveness of our open-loop approach to continuous media dissemination. We advocate using a time-driven rather than data-driven presentation approach and layered coding for increasing the statistical multiplexing gain of the network and for efficiently supporting heterogeneous receivers. We present measurements demonstrating that independently-received signal components do not cause significant communication overhead, while improving tolerance to rising host system load and network delay and jitter. We also discuss the impact of these techniques on application programming.

1 Introduction

We are investigating the communications aspects of layered (also referred to as hierarchical or subband) coding techniques for video dissemination over packet-switching networks. Layered coding (LC) techniques split signals into components of varying importance [4, 3]. The aggregation of these components reconstructs the original signal, but subsets of the data can also provide various degrees of approximation.

Layered coding is important for the efficient use of high-speed packet-switching networks. The main issue for such network architectures is the significant congestion control problems that can arise due to the statistical multiplexing of many very high burstiness signals [1]. A key property of any congestion control approach not based on resource reservation at the peak rate of the sources is the ability to shed load quickly and without causing an avalanche of retransmissions of dropped traffic. With LC of continuous media, when network congestion arises it is possible to drop the less important signal components without causing service interruption, and without the need for retransmissions. LC can also play an important role at the receiver because it provides the system software with the capability of allocating resources based on local specifications and priorities.

2 Experimental Environment

We experimented with a Fore Systems ASX-100 ForeRunner ATM switch and Fore Systems TCA-100 Turbochannel adapters. The ASX-100 is a 16 port non-blocking switch using a TDM bus running at 2.5 Gb/s with output queueing [2]. The host adapters were based on the 100 Mb/s TAXI chip-set using multimode fiber and are programmed I/O devices [2]. The device driver and application programming library support AAL3/4, AAL5, and a null AAL, as well as the standard Berkeley sockets interface. The driver performs all protocol processing in software, including segmentation and reassembly, with the exception of the CRC computation for AAL3/4.

Signaling is performed with Fore's proprietary Simple Protocol for ATM Network Signaling (SPANS) [2]. Virtual-circuits (VCs) are uni-directional and can have multiple destinations. Due to the bus-based architecture of the switch, support of multicasting is straight-forward and efficient. A bit mask is stored with each VCI indicating the destination port(s) of a VC. The data for the VC is "broadcast" on the bus and all appropriate ports read the data.

SPANS allows peak and mean bandwidth to be specified when VCs are set up. However, the switch implementation only supported *priorities*, with two separate output queues. If a VC is opened with non-zero requested bandwidth, then the VC is set up in the switch as high priority and cells for this VC use the high priority output queue in the switch. VCs requesting zero bandwidth use the low priority output queue. A VC exceeding its requested bandwidth for some period of time is reassigned to low priority. The service discipline is threshold-based for both output queues, but those thresholds were not user-settable in the switch used. The host adapters did not support priorities.

We used two DECStation 5000/200 workstations, running Ultrix version 4.2A, communicating over the ATM LAN. The switch was generally lightly loaded. In informal testing we achieved approximately 15 Mb/s of application-layer throughput using the native ATM interface and observed about 130 ms delay (application to application) for VC set up. The limiting factor for throughput appears to be the software-based segmentation and reassembly.

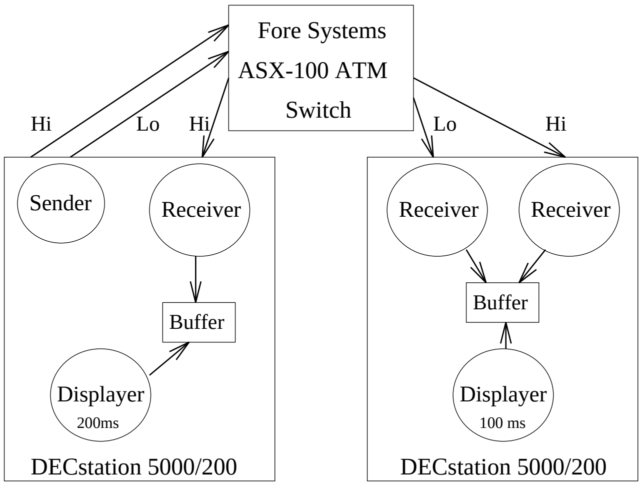


Figure 1: System Configuration and Software Design

3 The Application

Traditional image and video communications systems are designed to be data-driven, i.e., the system displays a frame at the moment it has received all data pertaining to this frame. Images and video frames are typically segmented over multiple network packets, even in non-ATM environments, because of packet-size constraints. Delay and loss in the

Research supported in part by Digital Equipment Corporation and the UC MICRO program.
* Now with QUALCOMM, Inc., San Diego, CA.

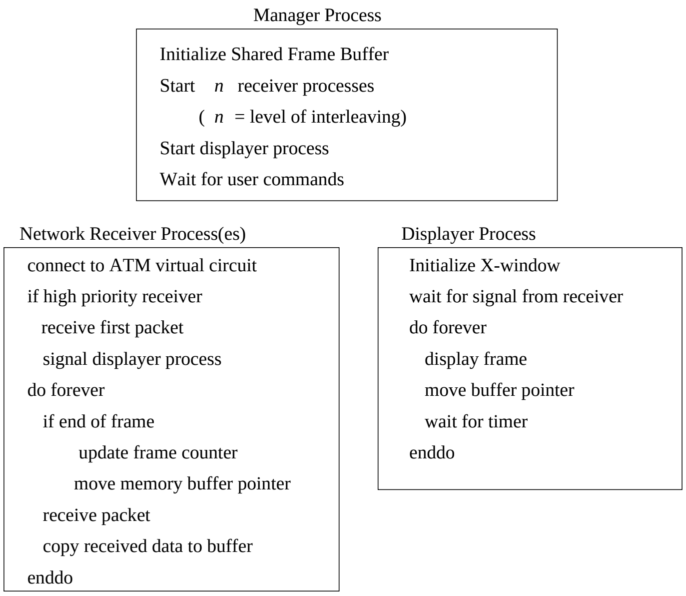


Figure 2: Processes on a Receiving System

network may cause frame jitter, i.e., result in irregular display rates.

We have developed a prototype interactive video dissemination system which uses elementary LC techniques. The system developed is time-driven, i.e., it displays a frame at the appropriate, predetermined point in time, irrespective of whether all related data have been received. Our assumption is that LC and system design and tuning would assure that all data are present most of the time, and the most important data are present all of the time.

3.1 Design Description

The application was implemented with a time-driven display process and a simple LC technique based on interleaving frames to form two independent layers (or streams). The application transmits each layer of the video on a different VC. One layer is transmitted as high priority and the other as low priority. The hardware configuration and basic software architecture is demonstrated in Figure 1. Both low and high priority streams use multicast AAL3/4 VCs.

We have experimented with a "talking heads" type video sequence from the TV news program *The McNeil-Lehrer News Hour*. The application used 256$\times$256, 8-bit color frames which are transmitted uncompressed. The source video frames are pre-packetized and sent from memory. The typical frame rate is 10 frames/s (f/s). Each video frame was divided into 32 packets of 2048 B each. 24 B of application layer header was prepended to each packet, for a total of 2072 B PDUs delivered to the ATM driver. The ATM driver itself handles the segmentation (and reassembly) of the packet into cells. This packet size was chosen as a compromise between larger packets, which provide better throughput (due to fewer calls to the ATM driver, and therefore, less overhead), and smaller packets, which provide better interleaving granularity and reduce the effects of packet loss.

The receiving application is comprised of several cooperating processes. The general structure of these processes is shown in Figure 2. A convenient and useful separation of tasks is implemented between network receiver process(es) and the displayer process. The receiver processes are purely data-driven and simply wait and receive data. The displayer process on the other hand is purely time-driven. Thus, the displayer process can run at a constant rate and is independent of the variability in the received streams. All the receiver and displayer processes use a single shared memory buffer. Receiver processes write frames to the buffer, and the displayer reads from the buffer and displays frames.

The time-driven display is implemented by a periodic software interrupt based on the real-time clock (UNIX interval timer). When the interrupt occurs, standard X-windows functions are used to display the next eligible frame. Frames are displayed at a fixed rate, which is given at start up.

Transmission is performed by a single process. This process sets up the ATM connections, reads in video frame data into memory, and then interleaves the sending of two frames at a time on two different VCs. The operation of the sender is straightforward and only the interleaving technique will be discussed further.

The LC scheme used in our application is based on interleaving 2 consecutive frames to form 2 independent components, each sent on a different VC. This technique was chosen for its simplicity, since this prototype is not intended to promote a specific technique, but rather to investigate the communications aspects of LC techniques in general. In particular, some other LC techniques we have used in other prototypes (e.g., quad-tree coding and bit-plane separation) demonstrated severe performance problems because of inadequate system support [8, 7].

The frame interleaving technique basically achieves a reduction in the rate required for the "high" priority component to be transmitted and also smoothes out the streams, particularly from the perspective of a receiver receiving just one stream. For the 2-way interleave case, the "high" priority frames are the even frames and the "low" priority frames are the odd frames. If an odd frame is missed or incomplete, error concealment can be performed by redisplaying the previous even frame (actually, by just doing nothing and skipping the odd frame). This can produce reasonable results for many video sequences of the "talking heads" variety.

The application is extended to include multiple receivers by simply using the multicasting support offered by SPANS. Each component stream, normally mapped to a VC, is mapped to a multicast VC. This allows each receiver to attach to the multicast VC of the layer(s) which it desires (only).

In our example, which uses 2-way interleave, one receiver attaches to both VCs and runs at the full rate, which is 10 f/s. The second receiver attaches to only the "high" priority VC (see Figure 1), and runs at half the full rate (5 f/s). This demonstrates the advantages of LC for a receiver with a limited bandwidth network link and/or limited system architecture (CPU, Memory, I/O Bus etc.) and incapable to receive, decode, or display the video at the full offered rate. A single VC running at half the rate allows the "low-speed" receiver to receive only the part it can handle, without having to receive and examine the entire stream, and presumably, discard the unwanted portion.

3.2 Performance and Limitations

The performance of the prototype application in terms of frame rate was limited to approximately 10 f/s. The bandwidth used on the ATM network was about 6 Mb/s (uncompressed). However, the performance was limited by the time to display the 256$\times$256, 8-bit color frames, which was approximately 60 ms, using standard X-windows functions. A frame size of 128$\times$128 was also tested. Display time was found to be approximately 25 ms. Using the smaller frame would allow frame rates close to the target rate of 30 f/s. However, this frame is only the size of a large postage stamp. The larger frame size was selected because of the importance of observing the visual effects of the techniques as well. Changing the mechanism for displaying frames from standard X-windows functions to X-windows functions with shared memory extensions may decrease the above reported frame display times.

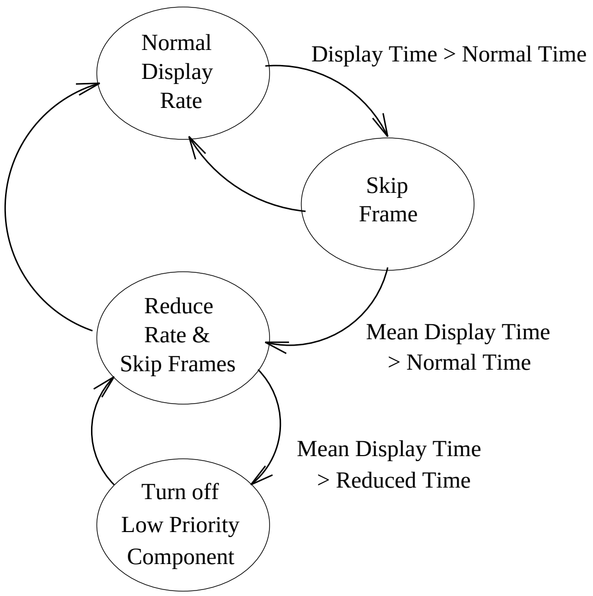
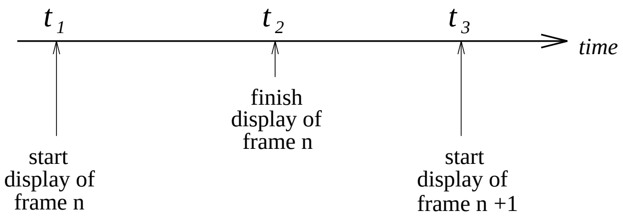


Figure 3: Frame Skipping with Independent Components



$$t_{frame} = t_2 - t_1$$

$$t_{interframe} = t_3 - t_1$$

Figure 4: Timing Calculations

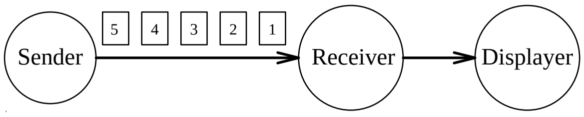


Figure 5: Configuration for Experiment 1

4 Experiments

We designed and performed a set of experiments to investigate performance issues of LC video transmitted (and received) on two different VCs. These experiments were intended to measure two effects. The first was the host overhead of using two interleaved components versus using a single component. The second was the evaluation of independent control of the components and to what degree they help a receiving system adapt to system load.

4.1 Overview of Experiments

The experiments were based upon the software designed and implemented for the previously described prototype application. All experiments consisted of a sending system and a receiving system. The sending system runs one sender process. The receiving system runs one or two receiver processes (receiving data from the network), and a displayer process. Sending is sequential or interleaved (2-way only), and may be on one or two VCs. A receiver process receives on only one VC, but multiple receiver processes can be active. Receiver processes insert frames into a buffer for the displayer process to access. Receiver processes can discard frames to reduce system load. When a receiver discards frames to reduce system load, the displayer skips alternate frames (the discarded frames). This algorithm is illustrated by the top three states in Figure 3. When multiple streams (VCs) are in use, this can be extended by including the bottom state shown in Figure 3. The criteria for state transitions shown in the figure are examples of ways to monitor the system load based on display timing.

Four metrics were collected in each experiment: Frame Display Time, Interframe Display Time, Frame Receive Time, and Interframe Receive Time (see Figure 4 for definitions). In addition, packet loss per frame was collected for reference. Data were collected for sequences of 500 frames for each experiment.

Each experiment was performed with an increasing amount of competitive CPU load. The CPU load was created by running a number of synthetic, CPU-intensive processes at different priorities. Seven different load scenarios are used, from no load up to the heaviest load. The results of the experiments are primarily the comparison of mean Frame Display times and Frame Display time variance, mean Interframe Display times and Interframe Display time variance. These times are plotted versus receiver CPU load in Figures 10-13.

Experiments 1 and 2 do not use the frame interleaving technique while experiments 3, 4 and 5 do. Experiments 1 and 3 are the normal 10 f/s case. Experiments 2 and 4 run at 5 f/s with receivers discarding frames and the displayer using frame skipping to handle system load. Experiment 5 also runs at 5 f/s and skips frames, but takes advantage of control of the independent components.

Comparing experiment 1 with 3, and also 2 with 4 gives an indication of the overhead involved in using this layered-coding technique. Comparing experiments 2 and 4 with 5 shows the benefits of independently controlling components. Comparing experiments 1 and 3 with 2, 4 and 5 shows the advantages of frame skipping to handle system load.

4.2 Observations and Interpretation of the Results

Experiment 1 and Experiment 3 performed basically the same when subjected to load; their mean display (see Figure 10) and mean interframe display (see Figure 11) times are very close. This indicates that there is no measurable overhead due to using two VCs and two processes to receive the interleaved frame transmission. These results also suggest that the use of multiple components and VCs to allow for heterogeneous receivers does not have a high communication cost for the receivers that receive multiple components. The only overhead on the receiving system is having an additional process running on the system. The overhead of context switching is nearly the same whether one or two receivers are used— the target of the context switch simply alternates for the two receiver case.

Experiments 2, 4 and 5 perform much better when subjected to load. This can be most clearly seen from the consistency of the mean interframe display times (see Figure 11). Time continuity is one of the most important

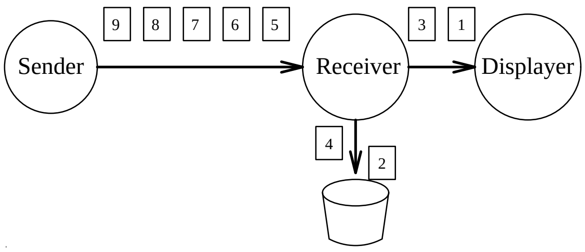


Figure 6: Configuration for Experiment 2

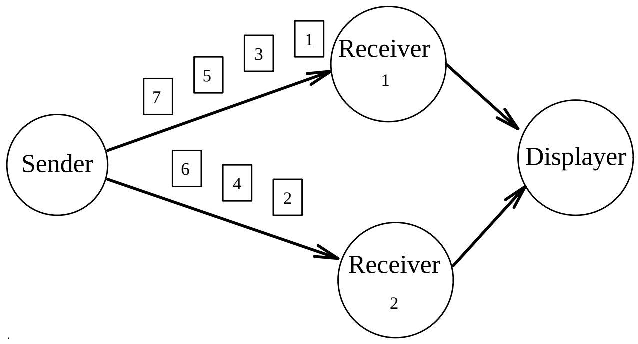


Figure 7: Configuration for Experiment 3

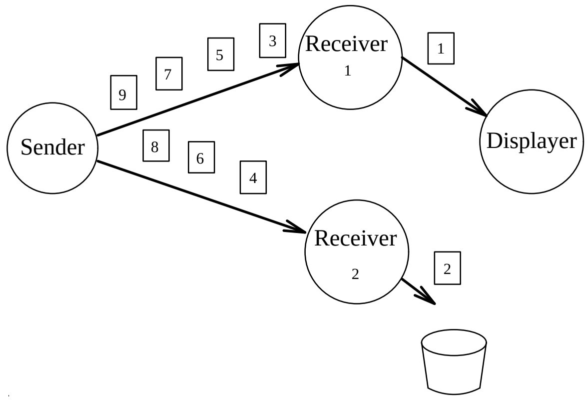


Figure 8: Configuration for Experiment 4

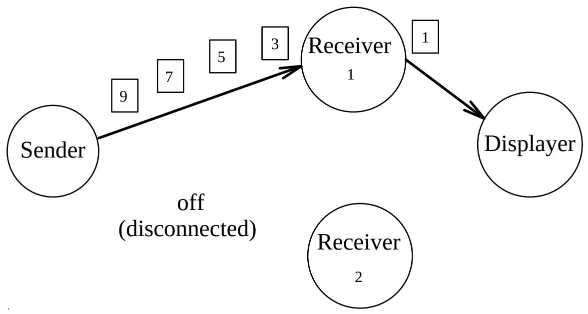


Figure 9: Configuration for Experiment 5

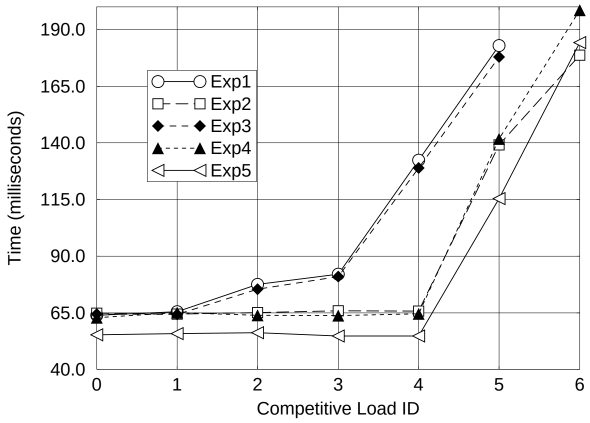


Figure 10: Mean Frame Display Time

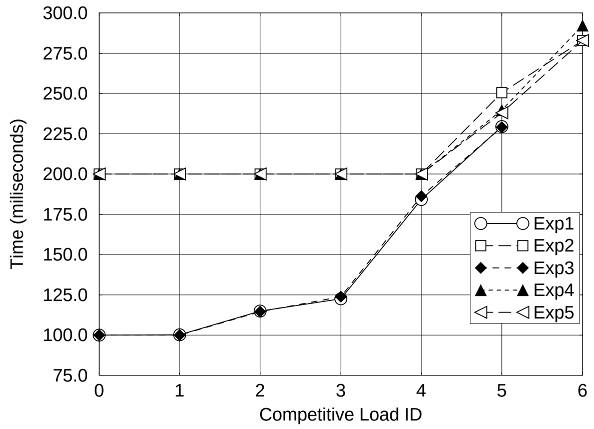


Figure 11: Mean Interframe Display Time

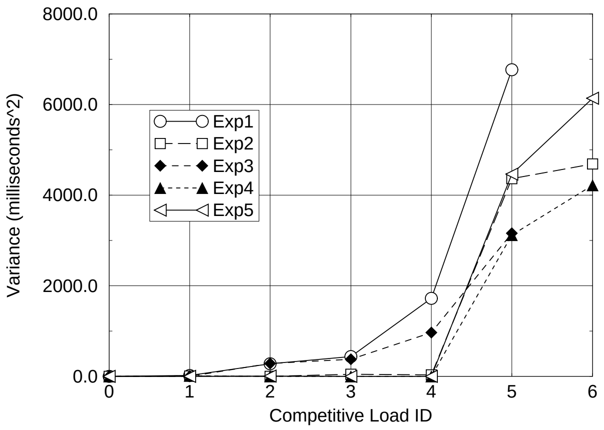


Figure 12: Frame Display Time Variance

factors for video applications. The continuity (or regularity) of the frame and interframe display times can be seen in more detail in the graphs of the variance of these times (see Figures 12-13). The mean frame display times show that Experiment 5 does actually perform a little better due to its ability to “turn off” one component; thus, it only has to handle half of the data. Experiment 5 consistently has frame display times about 10 ms faster than 2 and 4. This represents about a 15% improvement over 2 and 4 before the load begins to affect the performance.

The biggest gain for handling system load was realized by reducing the frame rate. We reduced the frame rate by half to 5 frames per second (and simultaneously skipped alternate frames) which also reduced the CPU utilization by about half. This is a good technique in this environment for two reasons. First, the display time dominates the total application time, and second, the types of video sequences that we are using are “talking head” type sequences which are typical of video conferencing. If the application was not dominated by display time, then reducing the display frequency would not be as beneficial. If network or decoding time were the bottlenecks, then actually “turning off” the receiving and/or decoding of one component would be much more beneficial. Second, “talking head” video typically has little movement and can tolerate frame skip.

5 Discussion

Our experience suggests that writing a time-driven application actually adds the time-driven paradigm on top of the data-driven paradigm, rather than replacing it. Such a time and data-driven application is necessarily more difficult. But, timeliness is a critical requirement for real-time,

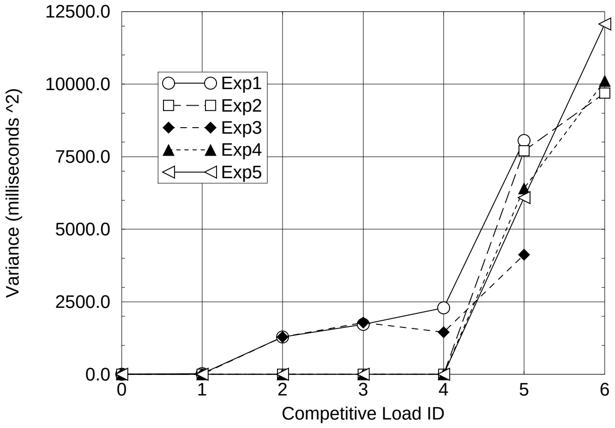


Figure 13: Interframe Display Time Variance

continuous media. In our initial implementation, we have a single process for the displayer, and different processes to receive each component of the layered video signal. The receiver processes are data-driven, and the displayer process is time-driven. This separation proved quite useful and made the implementation rather straightforward. However, while this is a convenient and clean conceptual separation, additional decision-making information appears useful to both types of processes to handle clock skew, network delay and other sources of variability.

The displayer could use a mechanism to fall back or to pause (possibly wait one frame time, which is equivalent to redisplaying the previous frame) or, as a last resort, resynchronize to a new display frequency based on the speed of incoming data. For the receivers, information on whether the data for the current frame is late or not, will allow them to perform better by simply discarding late data. In the environment of this application, the bottleneck was display time. Therefore, the displayer can monitor its display time and begin skipping frames if necessary. However, if delay was due to the network, a receiver would have to communicate that to the displayer in some way. A method for an application to measure its own bottlenecks would make it easier to adapt itself to system load based on where the bottleneck occurred (display, decode, or receive). Such monitoring and control could be put into the manager process which spawns all the other processes. This process could determine where system bottlenecks are occurring and take appropriate action to handle the situation.

For the applications programmer, support for multicast VCs in the ATM network is very useful. The transmission of the data is identical to the unicast case, as it should be. In our application, switching from unicast to multicast connections was done simply by changing the type of the VC and changing the sending application to accept connections on one VC from more than one receiver. Although, the sending host accepts the connection, the ATM switch actually handles the multicasting. Therefore, the actual data transmission from the sending system is identical (except now the VC type has changed). It should be simple to separate the connection handling function from the transmission function, to add multicasting to an existing unicast application.

6 Conclusions

Layered coding (LC) of continuous media can help reconcile the great economic advantages of statistical multiplexing in high-speed packet-switching networks with the complexity of controlling the congestion phenomena that it produces. It promises to provide the necessary freedom to design networks that provide the needed quality of service for applications, without reverting into tight control of all network resources.

We argue that interactive continuous media applications should be written using a time-driven rather than a data-driven model. This will be a departure from the traditional approach and will require dealing with error concealment to address potential cell loss or excessive cell delays. This is exactly where layered coding techniques, and more generally decomposition of information flows into hierarchically related prioritized components, can help.

The prototype application developed demonstrates the capability to incorporate heterogeneous receivers and provides a method for adapting to system load. The application used a very simple LC technique, frame interleaving, which even though by itself may not be a realistic solution, it demonstrates many of the advantages that make LC attractive from a communications perspective. For example, it uses non-redundant components and the components are independent. The drawback of frame interleave is its limited error concealment properties.

Multiple receivers with different capabilities can easily be supported through the use of independent components, each sent through individual VCs. A less powerful system can receive only a subset of the components, while other receivers might use them all. If only one component is sent with high priority, then the *guaranteed* bandwidth required is reduced. This can significantly increase the degree of statistical multiplexing in the network.

We also demonstrated the applicability of the time-driven programming paradigm (mainly through the visual effects of frame skipping for the low bandwidth receiver). Finally, this application illustrates the notion of open-loop control in a heterogeneous, multicast environment [5, 6], i.e., allowing receiving systems to control signal quality by selecting the components they receive, without explicit communication with the source.

References

[1] Special Issue on Congestion Control in ATM Networks, *IEEE Network*, Vol. 6, No. 5, September 1992.

[2] E. Biagioni, E. Cooper and R. Sansom, “Designing a Practical ATM LAN,” *IEEE Network*, Vol. 7, No. 2, pp. 32-39, March 1993.

[3] M. Ghanbari, “Two-Layer Coding of Video Signals for VBR Networks,” *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 5, pp. 771-781, June 1989.

[4] G. Karlsson and M. Vetterli, “Packet Video and Its Integration into the Network Architecture,” *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 5, pp. 739-751, June 1989.

[5] J.C. Pasquale, G.C. Polyzos, E.W. Anderson, and V.P. Kompella, “The Multimedia Multicast Channel,” *Network and Operating System Support for Digital Audio and Video*, Lecture Notes in Computer Science #712, Springer-Verlag, pp. 197-208, 1992.

[6] J.C. Pasquale, G.C. Polyzos, and V.P. Kompella, “Real-time Dissemination of Continuous Media in Packet-Switched Networks,” Proc. *IEEE Compcon Spring ’93,* San Francisco, CA, February 1993.

[7] G.C. Polyzos, “Image Browsing and Animation over a Local ATM Testbed,” Proc. *5th IEEE COMSOC International Workshop on Multimedia Communications,* Kyoto, Japan, May 1994.

[8] K. Taylor and G.C. Polyzos, “Performance Measurements of a Simple Hierarchically Coded Image Animation over Various Network Testbeds,” Technical Report CS93-346, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1993.