The following paper was originally published in the
Proceedings of the USENIX Windows NT Workshop
Seattle, Washington, August 1997

Delivery of High Quality Uncompressed Video over ATM to Windows NT Desktop

Sherali Zeadally
Department of Electrical Engineering
University of Southern California, Los Angeles

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Delivery of High Quality Uncompressed Video over ATM to Windows NT Desktop

Sherali Zeadally
Department of Electrical Engineering
University of Southern California
University Park, DRB 116
Los Angeles, California 90089
Tel: 213-740-1450; Fax: 213-740-9280
zeadally@marco.usc.edu

Abstract

The emergence of high bandwidth applications such as medical visualization and virtual reality has exposed significant deficiencies in network, protocol, and end-system design. In this paper we discuss important end-system issues which arise when supporting applications demanding networked delivery and manipulation of *uncompressed* video to the desktop.

Our experimental network environment consists of DEC Alpha workstations using the Windows NT 4.0 operating system and connected via an ATM switch. We present the design and initial results of a network architecture that demonstrates the creation, manipulation, and distribution of high-quality uncompressed video using standard industry-based technologies. In addition, we discuss networking performance results and present a simple Windows Sockets 2.0 cost model for TCP/IP and UDP/IP over ATM.

An early potential market where this work is expected to have a direct impact is video editing in motion picture and television studios. In this context, we hope to provide cost-effective networked solutions aimed at replacing costly dedicated video editing hardware with the versatile capabilities of general purpose workstations and non-proprietary network solutions.

1. Introduction

Recent advances in network technologies and computer hardware have led to the development of powerful computers and high-speed networks. Along with these developments, a wide range of multimedia applications have also emerged, particularly those involving digital video and audio. Multimedia has become a critical technology for professional applications in hospitals, educational establishments, advertising agencies, and video studios. It is therefore crucial that emerging open-system solutions be able to support the creation, manipulation, distribution, storage, and retrieval of real-time multimedia data.

In the past, poor network bandwidth coupled with hardware limitations of workstation architectures prevented the support of uncompressed digital video to the desktop. However, the emergence of high-speed networks coupled with hardware improvements (e.g. CPUs, buses) is paving the way to supporting the high bandwidth requirements of those applications using digital video and high quality images. So far, approaches adopted by hardware/software designers have mainly been proprietary involving solutions such as special add-on cards or the use of special local buses to achieve high data transfer rates within the workstation. These approaches are expensive, inflexible, and very monolithic which makes it hard to modify or extend the underlying hardware/software design space. What is needed is an *open* architecture that exploits industry-based standards to provide *cost-effective scalable* solutions for supporting high bandwidth multimedia applications such as those using uncompressed video.

The structure of this paper is organized as follows. Section 2 describes our motivation for the need for uncompressed video for certain applications. Section 3 presents our project goals. In section 4, we discuss the architectural design challenges that must be addressed to support full motion, high quality uncompressed video to the desktop and our proposed solutions. In addition, we also present networking performance results and a user-level architecture for flexible multimedia processing. Finally, section 5 makes some concluding remarks.

2. Motivation

In the past, the only way to capture, store, and deliver uncompressed video was to use dedicated proprietary systems such as digital disk recorders. These devices are not only limited in the amount of storage, but are highly specialized focusing on specific functions and therefore cannot be used as general purpose devices. Furthermore, such equipment is costly and difficult to upgrade. However, general purpose workstations do not suffer these serious limitations. They can be easily upgraded to take advantage of advances in new technologies such as faster networks, hardware improvements, and new operating systems.

To cope with hardware limitations such as disk access speed, memory access time, bus transfer capability, limited network bandwidth, and limited storage space, various compression schemes have been used for the storage and transmission of digital video over the network and its delivery inside the workstation. However, the majority of compression methods are based on lossy algorithms such as Motion-JPEG and MPEG [27][11]. The resulting video stream after a compression/decompression operation with lossy algorithms is of lower quality due to data loss during compression and added unwanted artifacts during decompression. The degradation in quality is tolerable and not apparent to the viewer for those applications where the compression/decompression cycle takes place only once. The problem arises when the video has to go through a series of intermediate compression/decompression cycles as is often the case with video editing/compositing in a post production process. In this case, the accumulated loss of data and artifacts become more obvious thereby causing considerable degradation in the original quality of the source video. This is illustrated in Figure 1 where a typical video editing session involves decompressing an incoming video stream into main memory, performing editing or adding special effects, compressing the modified video data before transmitting to the next workstation over the network. When several edit sessions are required, successive decompression/compression cycles are performed which degrade the quality of the original video stream.

Lossless algorithms commonly used by motion picture studios for editing typically produce around 2:1 compression for color images with moderately complex scenes. For instance, lossless JPEG works well only with continuous-tone images but does not provide useful compression of palette-color images [21]. Other lossless image compression algorithms such as ERIC and RICE also achieve around 1.4-1.96:1 compression ratios [14]. In reference [16], a mathematically lossless algorithm for Motion-JPEG is proposed which results in a compression factor of 1.6:1. Thus, it is obvious that lossless compression algorithms do not really give substantial decrease in data throughput (only by a factor of two). Therefore, we argue that rather than incurring the cost of minimal compression, a better and more effective solution is to use no compression at for all phases of video production: capture, creation, editing, assembly, and playback. In this way, we ensure the fidelity of the final product during the editing and composition stages of content creation and avoid quality degradation caused by compression/decompression cycles.

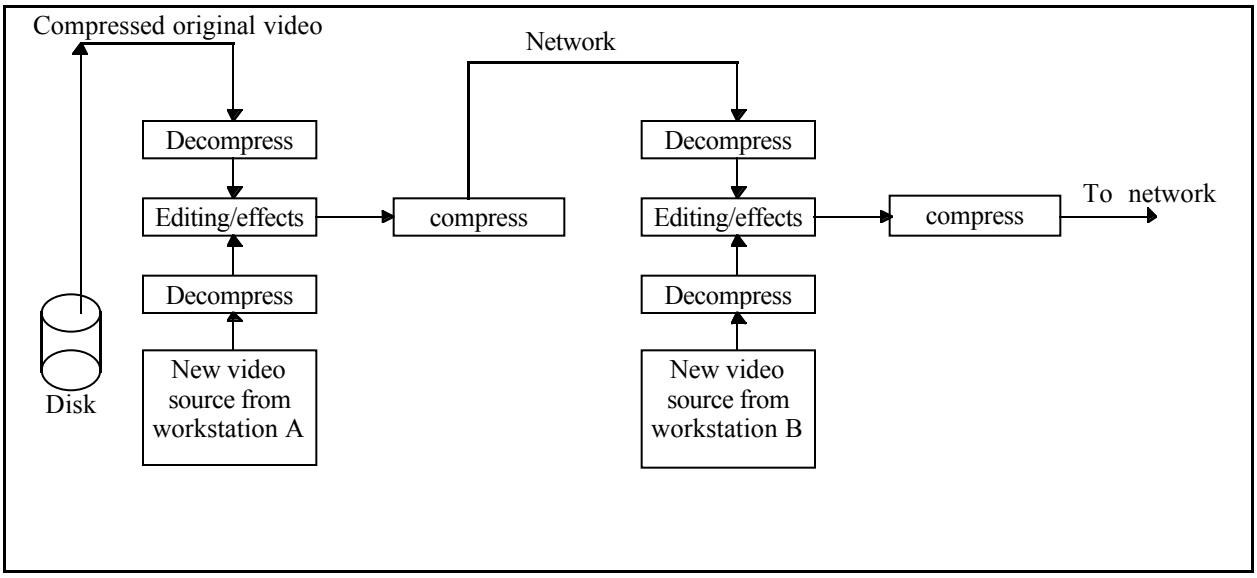

Figure 1 A typical workflow for editing/compositing/special effects requires multiple access to the original video data.

3. Project goals

One of the most important goals of the project is the design and implementation of a truly open architecture capable of delivering real-time, uncompressed, *high quality* digital video to the desktop. The use of the term high quality here means CCIR-601 studio quality video employed by most professional systems and movie studios. The digitized CCIR-601 [4] serial digital video signal contains 216 Megabits per second (Mbits/s) of video data (using 4:2:2 color encoding, 8 bits per sample at 27 MHz [28]). Ideally, we would like to perform all distribution and manipulation of high quality video in the digital domain. However, the lack of serial digital interface (e.g. serial digital interface cables for connecting a digital camera to frame grabbers) components means that we are forced to implement certain parts of the system using analog signals. For instance, in our prototype we are using an off-the-shelf camera for our live video source which delivers full-frame video to a frame grabber accepting an analog video input. Although we do not use CCIR-601 quality video directly, we are using a comparable data rate namely studio-quality digitized NTSC video which has a data transfer rate of 221 Mbits/s (30 frames, 640x480, 24 bit color pixel) compared to broadcast-quality NTSC which has a data rate of 120 Mbits/s. At present, the display is driven by analog RGB signals. In future, an all digital system might use Liquid Crystal Display (LCD) or digital light projector.

We plan to use Asynchronous Transfer Mode (ATM) (an international networking standard which transports network data as fixed-size 53-byte cells)[26] technology for the distribution and transmission of uncompressed digital video over the local area network. Another important goal is to provide a user-level architecture that allows flexible processing and manipulation on the video data stream. This is discussed in detail in section 4.4.

In order to achieve these goals, we are building an architecture based on industry standards including:

- General purpose workstations - DEC Alpha.
- Standard Microsoft Windows NT operating system.
- PCI bus architecture.
- Standard file format such as Audio/Video Interleave (AVI) and ActiveMovie [17].
- RAID hardware.
- ATM networking technology.

Operating system platform

One of the most crucial decisions we had to take early on in the course of this project was the choice of the operating system platform. We have chosen Windows NT due to a number of factors including: price, multiprocessor capability, its versatile networking capabilities such as the support for multiple protocols (TCP/IP, NFS, AppleTalk, DECnet, IPX/SPX, NetBEUI which allow connectivity to multiple platforms including UNIX-based systems, AppleTalk networks), true preemptive multitasking and multithreading, the availability of well-defined Application Programming Interfaces (APIs) including those supplied recently by Winsock 2.0 which allow applications to run natively and specify their Quality of Service (QoS) requirements on high-speed networks such as ATM, and access to numerous existing applications. Moreover, future porting of our software to other hardware platforms will be simpler since Windows NT already runs natively on other hardware platforms such as Intel.

4. Architectural design challenges

4.1 High performance video card

Content creation for broadcast-quality media involves a number of stages: pre-production (storyboards, script-writing, planning), production (animation, graphics, live video), and post-production (assembly and video editing). Each of the stages of content creation process requires distinct computer resources. We want to perform all of these steps in content creation using general-purpose high-performance workstations in a collaborative networked environment.

To perform all stages of content creation in the digital domain requires equipment such as cameras and video board (frame grabbers) to have the capability of delivering all media data digitally. Although digital cameras have now become available, there are no frame grabbers on the market that take an input digital video stream from such cameras and transfers the digitized video stream in *uncompressed* form. Consequently, we have chosen a frame grabber that accepts an analog video input such as that delivered by standard cameras. We have opted for a Coreco card [5] for our project since it is capable of delivering uncompressed digitized NTSC video stream at full-frame rate (640x480, 30 frames per second, 24 bit pixel) to the host. The Coreco board is a PCI-based adapter which has 2 MB on board video memory and an onboard VGA chip. We have not chosen other frame grabbers such as Matrox Meteor [15] (which can also deliver real-time full-motion video) because they need another display card such as MGA Millennium/Mystique [15] to work with since it has no on-board memory and also requires an additional PCI slot in the machine. At present the Coreco frame grabber works only on Intel-based architectures running Windows 3.1, Windows 95 or Windows NT 3.51/4.0, but will be ported to the DEC Alpha platform (with PCI slot) during the course of the project.

4.2 Bus bandwidth requirements

We are using DEC Alpha 600 Series high-performance uniprocessor workstations for our project. A simplified diagram of the DEC Alpha 600 Series architecture is illustrated in Figure 2. The Data switch implements the primary data path in the system and provides a 256-bit bus running at 33 MHz to main memory, 128-bit bus to the 21164 microprocessor and secondary cache, and a 64-bit bus running at 12 MHz to the CIA switch. The 21164 microprocessor has 3 on-chip caches: an 8 KiloByte (KB) primary cache (Dcache), an 8 KB primary instruction cache (Icache), and a 96 KB second level data and instruction cache (Scache). The CIA controls the Data switch, main memory, and interfaces to the 64-bit Peripheral Component Interconnect (PCI) local bus. In this Alpha model, there are three 64-bit and two 32-bit PCI slots [2].

The high data transfer rate associated with high quality video requires efficient transfers between the different components of the workstation. One of the crucial resources needed to deliver this high sustained data throughput is the bus bandwidth between the I/O devices and main memory. The DEC Alpha workstation architecture meets this requirement by a design which incorporates industry standard components such as the PCI bus which is capable of a maximum data transfer rate of over a gigabit per second for 32-bit PCI devices. However, raw peak bus bandwidth alone is not enough for high sustained data throughput between network and user application. We plan to apply careful optimizations (e.g. efficient video/image display, the use of shared memory to reduce data copying) in the network-application data path at *all* levels namely network, operating system, and user level in order to achieve high end-to-end application throughput.

All peripheral cards (e.g. network, frame grabbers) used in this project are 32-bit PCI-based. Other features that led us to the choice of the PCI bus include support for multiple bus masters and the ability to perform device-to-device transfers (with no intermediate stops in memory) resulting in much more overlap between I/O and CPU operations. This can be exploited in teleconferencing applications where live video from a frame grabber can be routed directly to the network without the intervention of the CPU. However, the initial prototype architecture is not going to implement this feature. Instead, live uncompressed NTSC video will first be streamed to host memory and then from main memory to the network adapter.

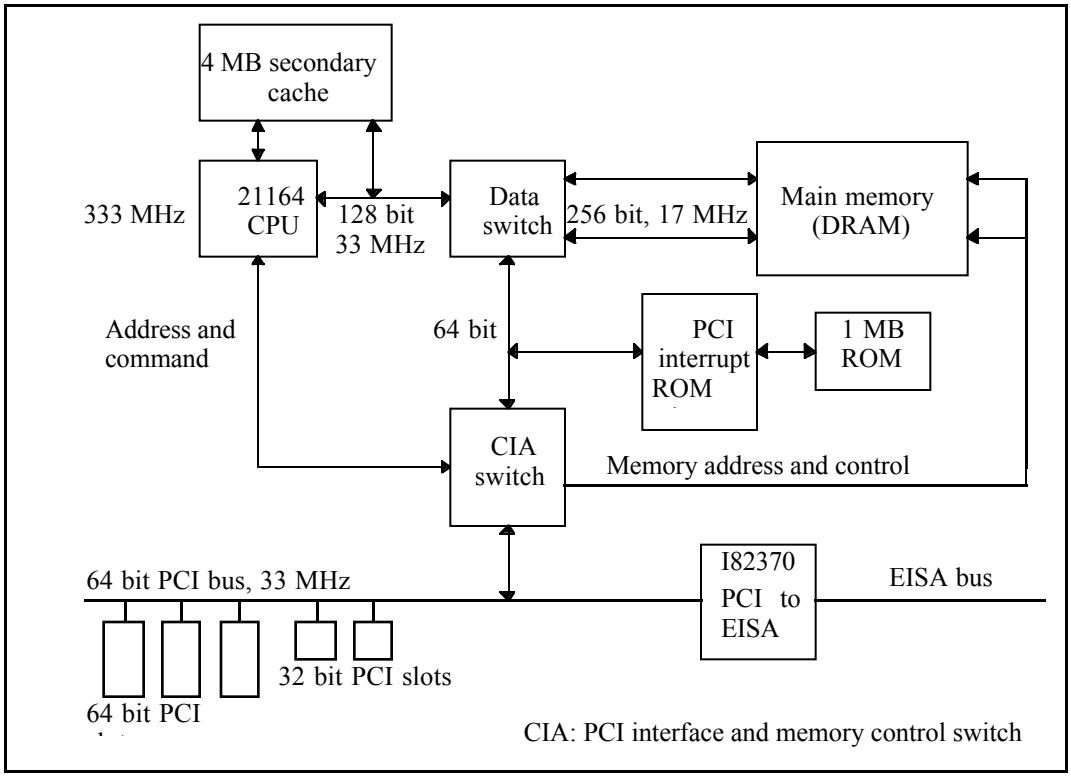

Figure 2 A simplified block diagram of the AlphaStation 600 Series [2].

4.3 Networking support

We have chosen ATM for our high-speed local area network testbed which consists of 4 DEC Alpha 600 workstations running Windows NT 4.0, each with a CPU speed of 333 MHz and 1 gigabyte of Random Access Memory (RAM). All workstations are connected to a DEC GIGAswitch/ATM [25] via multimode fiber. There are many reasons for the choice of ATM for our local area ATM network: although in this paper we focus only on one application area - editing involving uncompressed video, our long term goal however is to support different types of professional multimedia applications including interactive medical visualization, collaborative CAD, and interactive video collaboration. Each of these applications has its own traffic requirements (e.g. high throughput, low delay). ATM technology has the flexibility of providing the features necessary for different types of multimedia applications. For instance, ATM can provide QoS guarantees such as maximum cell rate, cell-transfer delay, or cell-loss ratio to user applications. Therefore, applications with different requirements will be able to negotiate their individual QoS requirements with the underlying network and be guaranteed a certain level of performance. This in contrast to other high speed local area networks such as Fast Ethernet which operates on a "best effort" basis and provides no QoS guarantees. Windows Sockets 2.0 [29][30] includes support for direct access to ATM and allows new multimedia applications to take advantage of QoS features via an ATM Service Provider for an ATM network as illustrated in Figure 3.

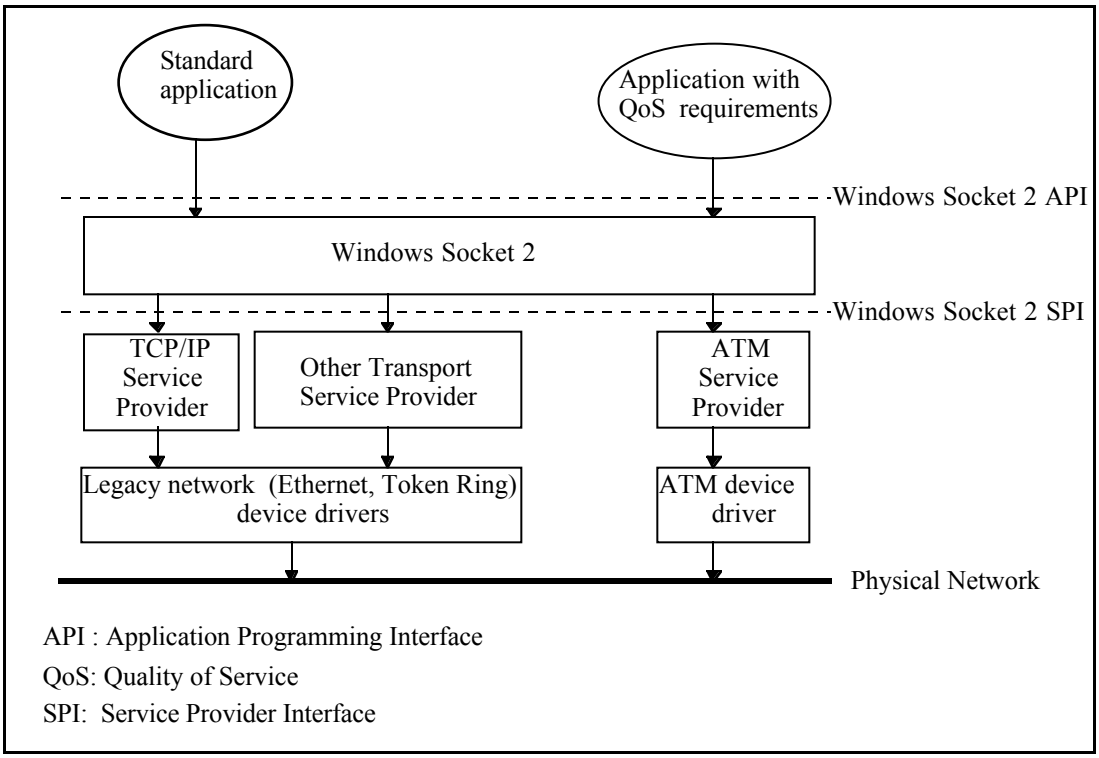

Figure 3 Windows Sockets 2 software support for multiple transport protocols and networks.

Windows Sockets 2 provides an API for user applications and a Service Provider Interface (SPI) for transport stacks. The Windows Sockets 2.0 API [29] includes mechanisms for applications to negotiate QoS with an underlying network such as ATM. Briefly, during connection establishment, a user application negotiates via Windows Sockets 2.0 the attributes of the connection. QoS attributes supported by Windows Sockets 2.0 API include: the source traffic description which describes (using parameters such as peak bandwidth) the way the application will send data over the network, upper limits on latency and latency variation acceptable, the level of service guarantee needed (the four levels defined are: absolute guarantee, controlled, predictive, or best effort), and cost for which is a metric is yet to be determined [29]. The QoS mechanism also allows applications to re-negotiate their QoS requirements after connection establishment using appropriate IOCTL calls.

Another important reason for the adoption of ATM is its ability to scale to higher speeds. The amount of bandwidth an application requires varies depending on the frame size, frame rate, and the quality of the image. Full-motion uncompressed studio quality NTSC video requires one-way sustained data transfer rate of 220 Mbits/s. We cannot satisfy this network bandwidth requirement with our current OC-3 ATM network which is only capable of transferring data at a signaling rate of 155.52 Mbits/s. Consequently, we have decided to choose a frame rate of 15 frames per second which requires a transfer rate of around 110 Mbits/s for our initial prototype architecture but as OC-12 (622.08 Mbits/s) becomes available, we intend to support full-frame rate of 30 frames per second. We do not believe that the cost of using OC-12 ATM will defeat one of our primary goals namely providing cost-effective solutions considering that motion picture studios incur significantly higher costs to effectively implement multiple editing sessions using expensive proprietary systems.

We have measured the throughput that can be delivered over ATM using conventional protocols such as TCP/IP and UDP/IP for a Classical IP [10] implementation - an ATM-aware layer below the traditional IP network layer which replaces the data link layer of the protocol stack (e.g. the media access control functions) with equivalent ATM functions.

For our experimental test configuration, we used two DEC Alpha workstations (each with a CPU speed of 333 MHz , 1 gigabyte of RAM, 4 gigabyte hard disk, and equipped with one ATM adapter (ATMworks 351 from DEC)) connected via the DEC switch using multimode fiber. All tests were conducted using Windows NT 4.0

and a socket buffer size of 64 KB for both TCP and UDP tests. We developed our own test programs for the throughput measurements. We measured the average application-application throughput between the two workstations connected via the DEC switch by timing bulk transfers from main memory over a sufficiently long period of time. All measurements were made using half-duplex connections. We also measured the CPU utilization using windows NT performance monitor [19] for the corresponding throughput achieved. The TCP/IP and UDP/IP results are shown in Figures 5 and 6 respectively. Note that the throughput result for TCP/IP is that obtained at the receiver - the transmitter throughput is the same (i.e. there were no dropped packets).

The results show that theoretical limit of 134 Mbits/s for OC-3 ATM is achieved on the DEC Alpha platform with Windows NT 4.0 for both TCP/IP (for message sizes above 40 KB) and UDP/IP. The dip in throughput for message size between 4 KB and 40 KB observed in Figure 5 is probably due to flow-control and acknowledgement algorithms used by TCP. There is a significant decrease in throughput at 8 KB. This is due to memory paging since 8 KB exceeds the amount of data that can be stored in a memory page. The page size used on the DEC Alpha 600 Series workstations is 8 KB but can hold less than 8 KB of data since it also keeps control information. Thus, with an 8 KB user message, the data has to be stored in two memory pages rather than only one memory page introducing extra memory overheads. CPU utilizations for TCP/IP and UDP/IP are around 55-65% and 50% respectively for reasonably large message sizes. In addition, we have also verified whether the CPU becomes the bottleneck at higher network speeds.

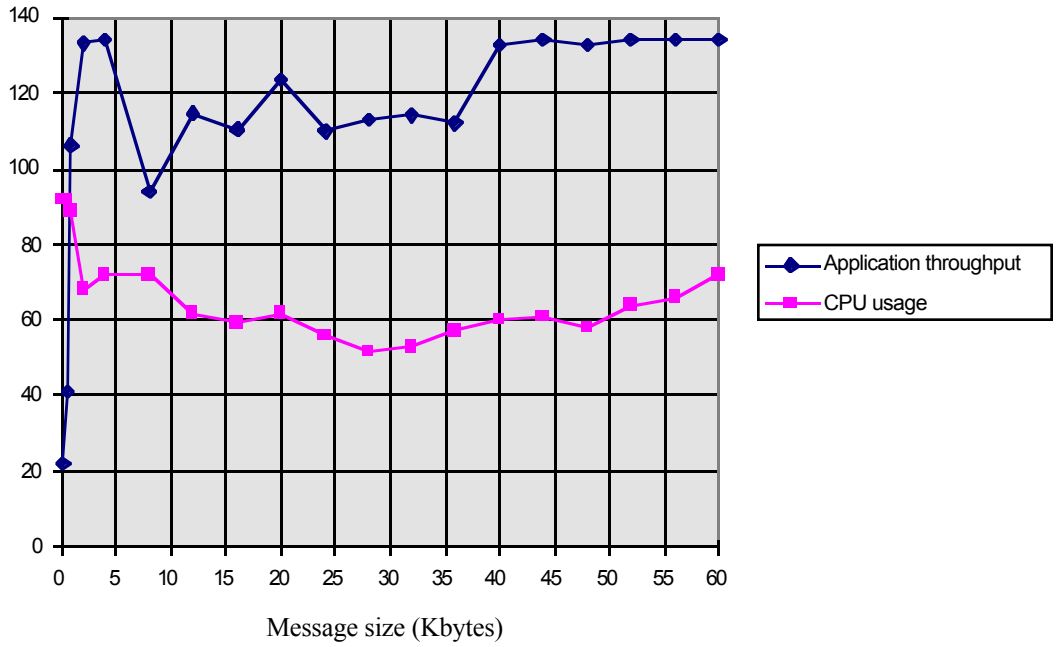

Figure 5 Variation of application throughput with message size for TCP/IP over ATM.

Using a full-duplex configuration (where each machine sends and receives data at the same time), we obtained an aggregate throughput of 240 Mbits/s for UDP/IP and the corresponding CPU utilization was around 90-95%. This clearly shows that at higher network speeds (e.g. OC-12 ATM), CPU becomes the bottleneck. The CPU availability factor becomes even more important since there are other operations that need to be performed in addition transferring data to user memory - the multimedia data needs to be displayed in addition to any intermediary processing that may be needed.

We are currently investigating how much CPU is required to display sustained full-frame (640x480, 24 bit color pixel) video. The software that allows grabbing of live video and transmission over the ATM network is still under development. This means that we have not yet been able to have a full data path from camera to the network and then to the display. However, the networking portion of the software has already been

implemented, and allows a series of images stored in main memory to be transmitted over the ATM network and displayed at the receiving workstation. In a preliminary experiment, a series of high quality uncompressed video images (BITMAP - 640x480, 24 bit color pixel) stored in main memory at the sender were transmitted using TCP/IP over ATM, and the images were continuously displayed at the receiving workstation - a DEC Alpha 333 MHz. Thus, at the receiver data was being transmitted from the network adapter to main memory, and then from main memory to the graphics adapter which is a Digital PowerStorm 4D20 [9]. We obtained a CPU usage of almost 100% at the receiver. The frame rate observed after displaying the images was around 11 frames per second. This observed frame rate translates to a throughput of 81 Mbits/s. This is explained as follows: the throughput achieved when displaying image data read from the network is really made up of two components - the time it takes to read data from the network added to the time it takes to display the image. When either of these time components is high, this lowers the final throughput perceived. In our experiment, we achieved 120 Mbits/s throughput for data transfer between network adapter and main memory. Therefore, the decrease in overall throughput is due to the time it takes to display the image data. One of the factors that contributes to slowing down the display throughput is that the CPU is shared between displaying the images and handling incoming network data. That is, CPU cycles left over after handling network data are not sufficient to achieve higher display rates. Thus, CPU becomes the bottleneck. In this context, we would like to point out that the code for displaying the images is still in its early phase and has not really been optimized yet. As a result, we believe that there is still plenty of scope for improving the display performance and it is quite likely that the non-optimized display code has introduced unnecessary overheads causing high CPU consumption. We are currently focusing on understanding how the PowerStorm device driver works in order to optimize the data transfer path from main memory to the graphics buffer.

Based on these preliminary results, it is clear that there is a strong need for optimizing the display rate with as little CPU intervention as possible. One solution that might be worth exploring is the use multiple PCI buses - and possibly multiple processors as well - so that network data transfer between network adapter and main memory takes place in parallel with data traffic for the display on a separate PCI bus. A faster CPU will be needed not only for sustaining studio quality video data rate over the network but also for fast image/video displays. In this context, we are presently investigating methods that will optimize the display rate for high bandwidth applications.

The network throughput results show that Windows NT is **not** the factor responsible for the poor performance of TCP/IP and UDP/IP over ATM previously reported by other researchers [1][3]. The low performance (around 70 Mbits/s) achieved in [1][3] is due to several factors such as processor performance, network adapter hardware, ATM device drivers, and so on. Our results show that with careful integration of well implemented software (e.g. network device driver) and good hardware design (e.g. network adapter, host bus), it is possible to deliver high performance with Windows NT operating system. A more detailed discussion of networking performance of Windows NT 4.0 over ATM is given in [31].

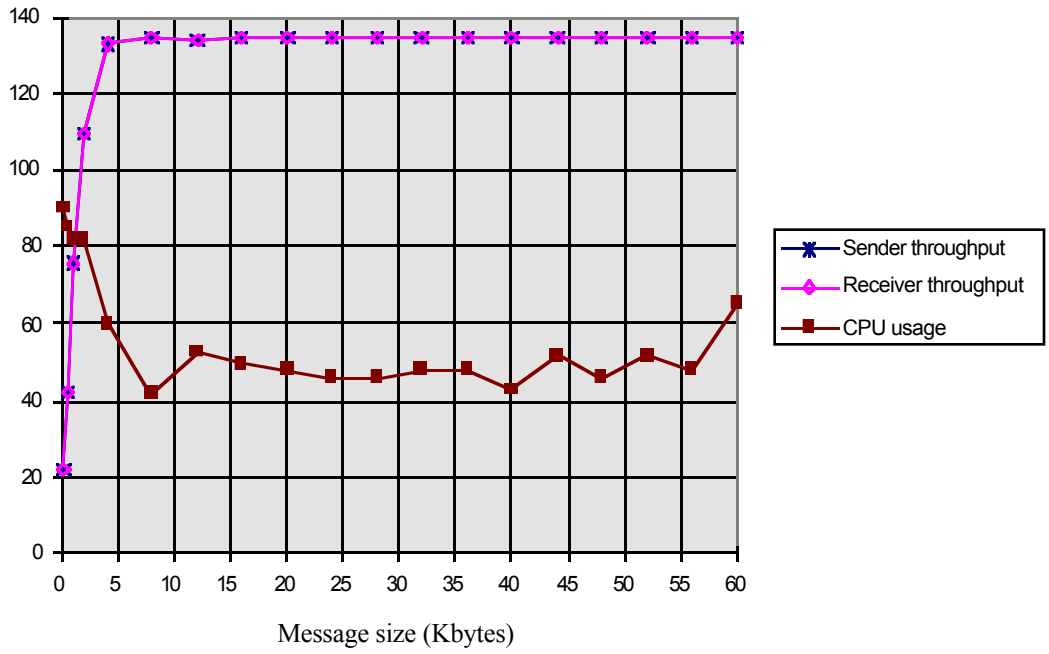

Figure 6 Variation of application throughput with message size for UDP/IP over ATM. The sender and receiver throughput graphs overlap.

4.3.1 A simple cost model for Windows Sockets 2.0

Based on the results obtained from Figures 5 and 6, a simple cost model can be derived for Windows Sockets 2.0 as follows:

The 21164 microprocessor on the DEC Alpha 600 Series can issue two integer/memory class instructions and two floating point instructions for every clock cycle [2][24]. However, experimental measurements performed with several applications in [7] have demonstrated that two clock cycles per instruction for the processor is more likely. Therefore, using the result in [7] for the number of clock cycles per instruction, a 333 MHz clock (used in our experiments) implies a processor performance of 167 MIPS.

In order to obtain the fixed cost that has to be incurred by the underlying socket implementation protocol, and other operating system overheads, we measured the latency for a 4 byte message which was transmitted over the ATM network. The actual measurement made was the round-trip delay and the latency was then calculated by halving the round-trip result. We obtained 178 microseconds for TCP and 84 microseconds for UDP. In these measurements, we ignored the fiber delay since the workstations used were separated by a small distance (only a few meters).

Cost for TCP/IP message over ATM:

Latency for TCP: 178 microseconds.
Fixed message cost = 178 * 167 = 29726 instructions.

From Figure 5, we observed an average CPU usage of 60% corresponding to a throughput of 134 Mbits/s (i.e 16.75 megabytes/second).

Number of instructions used to achieve 16.75 megabytes/second = 167 * 0.6 = 100.2 MIPS.
Number of instructions per byte = 100.2/16.75 = 5.98 ~ 6 .

Therefore, cost of message for TCP = 29726 + 6 instructions per byte.

Cost for UDP/IP message over ATM:

Latency for UDP: 84 microseconds.
Fixed message cost = 84 * 167 = 14028 instructions.

From Figure 6, we observed an average CPU usage of 50% corresponding to a throughput of 134 Mbits/s (i.e. 16.75 megabytes/second).

Number of instructions used to achieve 16.75 megabytes/second = 167 * 0.5 = 83.5 MIPS.
Number of instructions per byte = 83.5/16.75 = 4.98 ~ 5 .

Hence, cost of message for UDP = 14028 + 5 instructions per byte.

The above results show that UDP gives 17% better performance over ATM compared to TCP on a per byte basis. The fixed cost is twice for TCP compared to UDP. It has not been possible to get similar results for native ATM (i.e. without using protocols such as TCP/IP or UDP/IP) since the ATM device driver used does not support it for user applications. However, this will be done in future work.

4.4 Manipulation/processing of multimedia data

The first generation of multimedia applications have been mostly based on simply *presenting* multimedia data to the end-user with little or no processing (e.g. video display for a video conferencing application). However, the second generation of multimedia applications will not only involve mere presentation of multimedia data but also *manipulation/processing* on either all the media data stream or specific portions of it. Our goal is to have a flexible and extensible architecture capable of processing different media types in a modular fashion. In this context we are developing a User-level Multimedia Module (UMM) which allows full user control on the data path of a video stream originating either from the network (from another workstation) or from a live source such as a camera as shown in Figure 7.

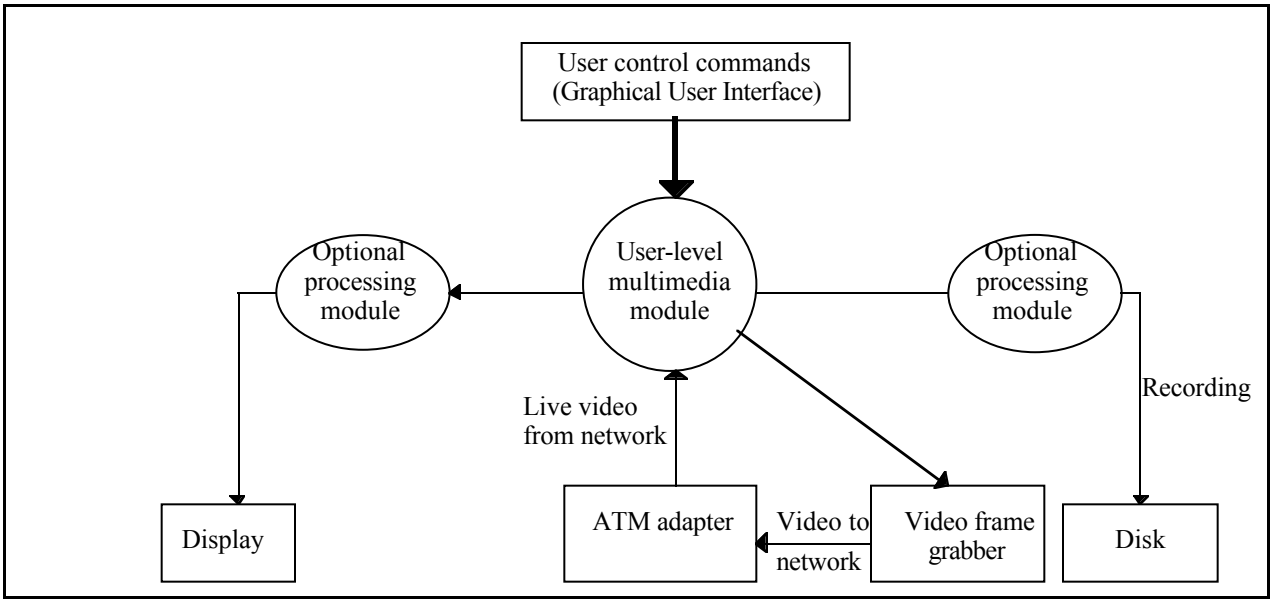

Figure 7 A functional description of the User-level Multimedia Module (UMM) and its interaction with display, network, and storage devices.

Two common scenarios are depicted for a receiver for video coming from the network: the user may choose to display/save all or part the video data (with or without intermediary media data manipulation using optional processing modules). An example of an application where such functionality is necessary is one which performs

tracking of features or objects in a video stream, and takes appropriate actions (such as saving to a file or displaying only a specific object) based on the actual media content. Figure 7 also shows the case for a transmitter whereby live video can be grabbed using the frame grabber and routed to the network adapter for transmission. We have already implemented this portion of the UMM. To simplify integration of the UMM with other components of the system, we have chosen ActiveMovie as our design space. Some of the major features that make ActiveMovie an attractive choice include: the support of multiple multimedia data types (e.g. digital video from various types of codecs, still images, digital audio, and so on), high data throughput can be achieved using large buffers (compared to Video for Windows which limits transfers to a maximum of 64 KB), major operating system overheads such as CPU data copying between applications are eliminated using shared memory, and low-level synchronization support (i.e. at field level rather than frame level) which ensures timely delivery of data to presentation devices. Another interesting feature of ActiveMovie is the use of processing elements known as *filters* which can be connected to each other (thereby giving what is called a *filter graph*) to provide a required processing sequence. Thus, in Figure 7, the processing elements will be implemented as filters, and the various functions of the UMM itself will really be implemented as a filter graph.

So far, we have successfully demonstrated the capability of the Coreco cards for grabbing full-frame live video in real-time host memory with the option of saving to disk. The capability of sustaining full-frame uncompressed video to main memory has shown that the PCI bus is capable of providing the bandwidth required. We have enhanced the Coreco software to include the capability to save live uncompressed video clips as AVI files. However, since high throughput is limited by the AVI format, we are presently developing new software that will exploit the use of OpenDML [12] format which should allow higher playback rates than AVI format.

4.5 Video transmission format

The actual format to be used for transporting uncompressed video over our ATM network has not yet been decided. The advent of ActiveMovie Streaming Format (ASF) [18] as an open and extensible data-independent format for storing and transmitting multimedia content over a wide range of networks appears promising and is one of the options we are presently considering. However, the lack of information on the specification of ASF has made it hard to come to a decision on this issue. Furthermore, it is also not clear whether ASF will be able to provide the high sustained data rate required for uncompressed video. Meanwhile, we are currently looking at other possible transport format for video.

4.6 Storage and video playback

Uncompressed video requires a large amount of storage. For instance, a 45 seconds uncompressed video stream uses one gigabyte of storage space. Uncompressed video playback from storage disks also needs high sustained throughput - typically 220 Mbits/s for full-frame, full-rate digital video stream. Initially, we are using five 4 gigabyte Fast Wide SCSI II disk drives for storage space. The current focus on the delivery and manipulation of live video rather than investigating issues such as video playback issues from video storage servers. This has been dealt by other researchers [13][20][23] and is not the subject of our research. Yet, if we store uncompressed video, we will need to achieve a reasonable data rate for video playback. Initial experiments on a Fast Wide SCSI II disk gave a throughput of about 50 Mbits/s. This is clearly insufficient to provide the necessary data rate (220 Mbits/s) required for uncompressed video. One option is to use faster SCSII disk controllers such as UltraSCSI which results in a peak data rate of 40 Megabytes per second and multiple disk drives configured to use software stripe sets supported by the NT File System (NTFS [6]). In order to achieve the required data rate for an uncompressed video stream, striping has to be performed across controllers. Another option that will be investigated is the use of a hardware RAID [22] system such as Digital StorageWorks [8] to provide the required throughput.

5. Conclusions

In this paper, we have described our motivations behind the need for the delivery of high-end *uncompressed* video to the desktop. We have outlined the major architectural design issues that need to be addressed to achieve this goal. We have presented solutions we are currently implementing to solve the architectural challenges. Our design approach is based on an open architecture which exploits existing industry-based standard technologies rather than the use of proprietary hardware or software. This will allow cost-effective,

scalable, networked solutions for multimedia applications that use high-end digital video (e.g. video editing in motion picture studios). In addition, we have also reported application-application throughput reaching theoretical limit for TCP-UDP/IP over ATM and presented a simple cost model for Windows Sockets 2. The results demonstrate that Windows NT is a suitable operating system choice for meeting the high performance requirements of applications running over high-speed networks such as ATM.

Acknowledgements

The work presented in this paper is the result of a joint effort between Digital Equipment Corporation (Massachusetts) and a group of investigators in the NSF-sponsored Integrated Media Systems Center at the University of Southern California which is investigating the manipulation and distribution of full-motion, high quality *uncompressed* video to the desktop (running Windows NT 4.0) over an ATM network.

The author expresses his gratitude to Jim Gray who provided valuable feedback and ideas on early drafts of this paper and whose patience and support throughout the revision process are greatly appreciated. The author gratefully acknowledges suggestions from Tony Levi on an earlier version of this paper. The author thanks Grig Gheorghiu for his comments on the paper and the many employees of Digital Equipment Corporation (Massachussetts) for their support, in particular Doug Washabaugh for his valuable discussions on many aspects of this work. The author also expresses his gratitude to the anonymous reviewers for their valuable ideas. The USC work is supported by the Integrated Media Systems Center NSF Grant EEC-9529-152.

References

[1] Andrikopoulos I. et al. "TCP/IP Throughput Performance Evaluation for ATM Local Area Networks." In Proc. of 4th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks, Ilkley, July 1996.

[2] Bhandarkar D. "Alpha Implementations and Architecture, Complete Reference and Guide." Digital Press, ISBN 1-55558-130-7, 1996.

[3] Carbone J. "Preliminary Network Performance Study of a Windows NT ATM Network." Internal Report, Odyssea Systems Corporation, June 1996.

[4] CCIR Recommendation 601-2. "Encoding Parameters of Digital Televison for Studios." Vol. XI - Part 1, International Telecommunications Union, Geneva, 1990.

[5] Coreco Inc. "The Oculus Driver Command Interpreter Manual." Edition 1.0, Revision 0, Coreco Inc, St. Laurent, Quebec, Canada, 1990.

[6] Custer H. "Inside Windows NT." Microsoft Press, ISBN 1-55615-481-X, 1992.

[7] Cvetanovic Z. and Donaldson D. "Alpha Server 4100 Performance Characterization." Digital Technical Journal, Vol. 8, No. 4, 1997. Http://www.europe.digital.com/info/DTJ001/DTJ001PF.PDF.

[8] Digital Equipment Corporation. "StorageWorks Solutions 7 device, 16-bit Deskside, Expansion Pedestals, BA356-K Series User's Guide." Digital Equipment Corporation, Massachusetts, December 1995.

[9] Digital Equipment Corporation. Http://www.alphastation.digital.com/products/powerstorm/pssld13.html.

[10] Laubach M. "Classical IP and ARP over ATM." RFC 1577, Network Working Group Internet Draft, January 1994.

[11] LeGall D. "MPEG: A Video Compression Standard for Multimedia Applications." CACM, Vol. 34, No. 4, April 1991.

[12]Legault A. and Matey J. "OpenDML AVI File Format Extensions, Version 1.02." http://www.matrox.com/videoweb/odml_826.html, February 1996.

[13] Lougher P. and Shepherd D. “The design of a storage server for continuous media.” Computing Journal, Vol. 36, page 32-42.

[14] Majani E. “Lossless compression.” http://www-ias.jpl.nasa.gov/HPCC/eric/node4.html.

[15] Matrox Inc. “Display Capabilities with Matrox Meteor and MGA Millennium/Mystique.” http://www.matrox.com/imgweb/met_mga.html.

[16] Matrox Inc. “Mathematically Lossless Motion JPEG - Better Than Uncompressed Video.” http://www.matrox.com/videoweb/hot_math.html.

[17] Microsoft Inc. “Microsoft ActiveMovie 1.0 Software Development Kit.” October 1996.

[18] Microsoft Inc. http://www.microsoft.com/corpinfo/press/1996/mar96/pronetpr.html, 1996.

[19] Microsoft Inc. “Microsoft Windows NT Resource Kit.” Microsoft Press, ISBN 1-57231-343-9.

[20] Mourad A. “Issues in the design of a storage server for video-on-demand.” Multimedia Systems, pages 70-86, Vol. 4, 1996.

[21] Oxford University. “JPEG image compression.” Oxford University Libraries Automation Service, http://www.lib.ox.ac.uk/internet/news/faq/archive/jpeg-faq.part1.html.

[22] Patterson D., Gibson G., and Katz R. “A case for redundant arrays of inexpensive disks (RAID).” In Proc. of the ACM SIGMOD 1988, Chicago, Illinois, pages 109-116.

[23] Reddy A. L. N. and Wyllie J. “Disk scheduling in a multimedia I/O system.” In Proc. of the ACM Multimedia , pages 225-233, Anaheim, CA, 1993.

[24] Sano B. Private Communications.

[25] Souza R. et al. “The GIGAswitch System: A High-Performance Packet Switching Platform.” Digital Technical Journal, Vol. 6, No. 1, 1994.

[26] Vetter R. “ATM concepts, architectures, and protocols.” CACM, Vol. 38, No. 2, pages 30-38, 1995.

[27] Wallace, G. “The JPEG still-picture compression standard.” CACM, Vol. 34, No. 4, April 1991.

[28] Watkinson J. “The Art of Digital Video.” Butterworth-Heinemann Ltd. ISBN 0-240-51369-X, pp. 399-411, 1994.

[29] Stardust Technologies Inc. “Windows Sockets 2, Application Programming Interface, Revision 2.2.0.” http://www.stardust.com/wsresource/winsock2/ws2sdk.html/wsapi22.doc, May 1996.

[30] Stardust Technologies Inc. “Windows Sockets 2, Service Provider Interface, Revision 2.2.0.” http://www.stardust.com/ wsresource/winsock2/ws2sdk.html/wsspi22.doc, May 1996.

[31] Zeadally S. “TCP-UDP/IP Performance over ATM on Windows NT.” In Proc. of 3rd IEEE ATM’97 Workshop, Lisbon, Portugal, May 1997.