ViewStation Applications: Intelligent Video Processing Over a Broadband Local Area Network *

Christopher J. Lindblad, David J. Wetherall, William F. Stasior, Joel F. Adam, Henry H. Houh, Mike Ismert, David R. Bacher, Brent M. Phillips, and David L. Tennenhouse †

Telemedia Networks and Systems Group
Laboratory for Computer Science
Massachusetts Institute of Techology

Abstract

This paper describes applications built on the ViewStation, a distributed multimedia system based on Unix workstations and a gigabit per second local area network. A key tenet of the ViewStation project is the delivery of media data not just to the desktop but all the way to the application program. We have written applications that directly process live video to provide more responsive human-computer interaction. We have also developed applications to explore the potential of media processing to support content-based retrieval of pre-recorded television broadcasts. These applications perform intelligent processing on video, as well as straightforward presentation. They demonstrate the utility of network-based multimedia systems that deliver audio and video data all the way to the application. The network requirements of the applications are a combination of bursty transfers and periodic packet-trains.

1 Introduction

The ViewStation project [3] integrates the technologies of broadband networking and distributed computing with those of digital video to produce systems for video-intensive computing. The ViewStation platform is composed of a set of programmable digital video processing devices connected together in a personal local area network.

The project focuses on getting real-time data such as voice and video from the network all the way to the application. ViewStation hardware is centered around the VuNet [5], a prototype desk area network to link the display, camera, central processor, and wide-area network components. Since the ViewStation takes a software-intensive approach to multimedia, the VuNet and custom multimedia hardware were designed to provide efficient support for software-driven handling of multimedia streams.

This paper describes applications built with the VuSystem [1], the programming system that provides application support for the ViewStation project. The system provides simple scheduling and resource management functions to allow intelligent media-processing applications to run on workstations not specifically designed for multimedia. Examples of such applications include The Room Monitor, The Whiteboard Recorder, The Video Rover, The News Browser, The Joke Browser, and The Sports Highlight Browser.

*This research was supported by the Advanced Research Projects Agency of the Department of Defense, monitored by the United States Air Force (AFSC, Rome Laboratory) under contract No. F30602-92-C-0019, and by a grant from Nynex.

†The authors can be reached at: MIT Laboratory for Computer Science, Room 504, 545 Technology Square, Cambridge, MA 02139; Tel: +1 617 253 6042; Email: cjl@lcs.mit.edu.

2 Principles

The ViewStation architecture embodies a software-oriented approach to the support of interactive media-based applications. Starting from the premise that the raw media data, e.g., the video pixels themselves, must eventually be made accessible to the application, we have derived a set of architectural guidelines for the design of media processing environments.

Video information must be accessible to and manipulatable by the application. We want to enable a new wave of media applications in which the computer is an active participant. These applications analyze their audio and video data input and take actions based upon the analysis.

A software approach, preserving scalability and graceful degradation, is called for. Software-oriented applications adapt to the resources of the execution platform and to the dynamic load applied by the mix of concurrently active applications.

Perceptual time is the domain of interest for interactive applications. Real-time software environments are discrete systems that approximate real world time to the finest level of temporal granularity that can be achieved by the available technology. However, we need only approximate real world time at a granularity that satisfies human perceptions, a granularity we refer to as perceptual time.

The resultant software and media substrate must ride the technology curve. To be of long-term relevance, media architecture and software strategies must leverage rapid change in the underlying technologies, especially processors and memories. Each generation of technology affords increased sophistication in the media processing performed by the applications.

3 Other Work

In general, the design philosophy behind the ViewStation differs from other network-based multimedia approaches that are more hardware based, such as Pandora [15], a special purpose hardware sub-system which can be controlled by a workstation. Pandora's box performs video and audio capture as well as mixing in dedicated hardware attached to the workstation and under workstation control. The output of the box goes directly to the workstation display.

The Pandora approach fits into the dedicated hardware style of media delivery. Very specialized hardware is built which allows decent performance, but prevents taking advantage of gains in performance elsewhere in the system.

The Medusa project [4] has improved on the design of Pandora in the areas of portability, security, programmibility, and the support of multiple streams. Its design is similar to that of the ViewStation. It connects the network to the workstation, allows video to pass all the way to applications, and provides for the development of software *agents* to assist in collaboration. It is built upon simple, lightweight connections, so that it is easy to build large, understandable groups of modules.

Other approaches have attacked multimedia systems from the network aspect. The Atomic approach [13] uses a networking chip designed for multiprocessing. The network chips are used to interconnect the various subsystems such as a display, camera, DSP, and monitor. The Atomic approach uses specialized network hardware as a multimedia workstation's internal network.

The approach of the University of Cambridge [14] replaces the traditional bus structure of the workstation with an ATM network. This network interconnects a variety of multimedia devices on a fast communications substrate internal to the workstation, and resides between the processor and its memory hierarchy. Our VuNet approach, while similar, chooses not to intercede into this memory hierarchy, and thus places the ATM network boundary at the edge of the memory subsystem.

4 The VuNet

The VuNet is a gigabit-per-second network using Asynchronous Transfer Mode technology (ATM) [9, 11] that interconnects general-purpose workstations and multimedia devices. The VuNet is aimed at the desk-area and local-area environments.

A key goal motivating the design of the VuNet hardware has been simplicity. Our goal is to design network hardware that is easy to build with off-the-shelf VLSI components. More sophisticated

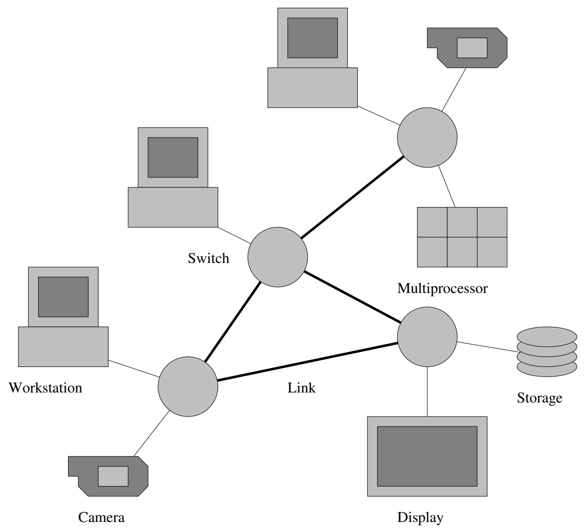

Figure 1: The VuNet.

network functions, including multicast, backpressure, support for ATM adaptation layers, and service classes, were pushed to the edge of the network where they become the responsibility of the clients. We believe that a local environment such as the ViewStation does not need sophisticated congestion-control functions. It can be effectively served by a switch fabric having a limited number of access ports and an internal speed that is greater than that of the clients.

The VuNet is based on the interconnection of two types of components: switches and links. The switches provide ports through which clients connect to the network and execute cell switching. The links interconnect the switches and implement cell relaying. With these two basic components, different network topologies are possible. Within a desk-area context, an office is equipped with a network switch which interconnects workstations and multimedia devices. Offices are then networked by connecting individual switches with links.

The VuNet switch consists of bidirectional ports with first-in first-out (FIFO) buffers which feed a crossbar matrix. The current version of the switch has four ports and has been operating reliably at a speed of 700 Mb/s. The switch has tested reliably at an internal data rate of 1.5 Gb/s.

High speed optical links provide inter-connection between VuNet switches. Cells are routed hop-by-hop as they pass through the links. Links contain header lookup tables that map VCIs to VCIs and switch ports. Connection management is performed through special control cells on a particular VCI. Control cells, received by a link on this special VCI, program the link's header lookup tables.

Each workstation in the VuNet is responsible for opening, maintaining, and closing its own connections. This can be done in a "wormhole" fashion by way of ATM control cells embedded in the data stream. The allocation algorithms in the connection daemon prevent nodes from stealing other nodes' connections. Processes are also run in the background which verify link tables and refresh connections if necessary, such as in the case where a switch has been restarted.

4.1 VuNet Clients

Three types of clients have been integrated into the VuNet: workstations, specialized multimedia devices, and inter-network interfaces. General-purpose workstations are connected through a host interface. Specialized multimedia devices, including video capture boards and an image processing system, are connected directly to switch ports. Inter-network bridges connect the VuNet to local-area and wide-area ATM networks.

The VuNet interface between client and switch was designed to be simple, allowing clients to be easily built. This follows the software intensive philosophy, which emphasizes a simple, flexible hardware substrate, and pushes complex functionality into workstation-based software.

The VuNet host interface is a simple DMA interface. Incoming cells are packed along with other relevant information into processor memory, where a kernel device driver reassembles packets and delivers them to the proper application. Outbound cells are packed by the driver in main memory with routing information and timing information (outgoing cells can be paced on a per-cell basis), and are transferred to the interface using block DMA transfers.

From the application perspective, media data is presented at the Unix socket interface. This provides a uniform software interface for all media data. Each media stream is mapped to its own ATM based virtual circuit. The VuNet kernel device driver implements the one-to-one mapping between application sockets and network virtual circuits. For incoming traffic, the device driver maps ATM Virtual Circuit Identifiers (VCIs) to their respective application socket interfaces. Throughput to the application has been measured at 40 Mb/s.

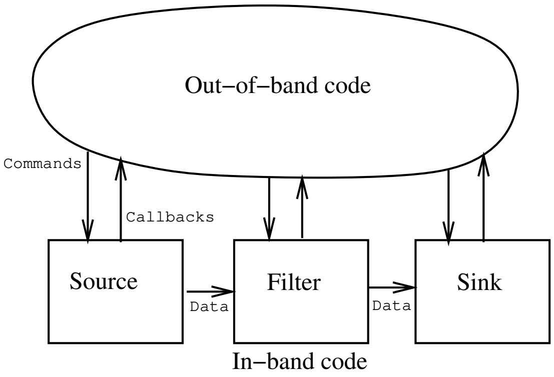

Figure 2: The structure of VuSystem applications.

5 The VuSystem

VuSystem applications are split into two partitions: one which does traditional *out-of-band* processing and one which does *in-band* processing (Figure 2). Out-of-band processing is the processing that performs event handling and other higher-order functions of a program. In-band processing is the processing performed on every video frame and audio fragment. In-band code is more elaborate in the VuSystem than in traditional multimedia systems [8, 10] because VuSystem applications perform more analysis of their input media data.

In the VuSystem, the in-band processing partition is arranged into processing *modules* that logically pass dynamically-typed data *payloads* though input and output *ports*. These in-band modules can be classified by the number of input and output ports they possess. The most common module classifications are sources, with no input ports and one output port; sinks, with one input port and no output ports; and filters with one or more input ports and one or more output ports.

The out-of-band partition is programmed in the Tool Command Language, or Tcl [12], an interpreted scripting language. Application code written in Tcl is responsible for creating and controlling the network of in-band media-processing modules, and controlling the graphical user-interface of the application. In-band modules are manipulated with *object commands*, and in-band events are handled with asynchronous *callbacks*.

The VuSystem is implemented on Unix workstations as a shell program that interprets an extended version of Tcl. All out-of-band code, including all user-interface code, is written as Tcl scripts. In-band modules are implemented as C++ classes and are linked into the shell. Simple applications that use the default set of in-band modules are written as Tcl scripts. More complicated applications add additional in-band modules to the default set.

VuSystem programs have a *media-flow* architecture: code that directly processes temporally sensitive data is divided into processing *modules* arranged in data processing *pipelines*. This architecture is similar to that of some visualization systems [16, 17], but is unique in that all data is held in dynamically-typed time-stamped *payloads*, and programs can be reconfigured while they run. Timestamps allow for media synchronization. Dynamic typing and reconfiguration allows programs to change their behavior based on the data being fed to them.

6 Applications That Process Live Video

We have found that the ViewStation provides a good platform for the investigation of concrete ways that computers may become more responsive to their human users. We are developing a prototype "Computerized Office Multimedia Assistant" (COMMA), that assists its user by performing various tasks that require the analysis of live video.

We have developed a library of vision service modules as a foundation for COMMA applications. Example modules include a *change detector*, which accepts two input images and outputs a binary image which shows which pixels differ on the two input images; a *motion detector*, which detects and localizes motion in a video stream by outputting a true value for those pixels which correspond to moving objects in the scene; and a *stationary filter*, which is effectively the converse of the motion detector.

The vision service modules communicate with the higher level scripting language by signaling events or callbacks to the Tcl layer. For example, a face recognition service would be implemented as a filter that calls a Tcl subroutine whenever a model face appears in the video stream. The Tcl program would then determine how to use this information.

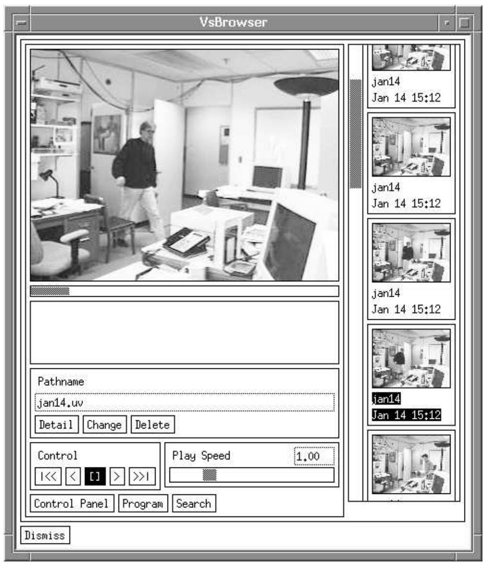

Figure 3: Browsing the output of the Room Monitor.

6.1 The Room Monitor

The Room Monitor processes periodic video data[1] from a stationary camera in a room. It processes

[1] In cases where the average data rate equals the peak channel capacity, applications pass data continuously. However, on the VuNet the data rate is less than the channel capacity, so typically connections pass data in periodicly recurring packet-trains.

the live video to determine if the room is occupied or empty, and records video frames only when activity is detected above some threshold. It produces a series of video clips that summarize the activity in the room. A video browser (Figure 3) is used to view the segments. The video clips allow the user to check who was in the room and when.

With additional processing, it is possible to extract people from the clips, using motion segmentation techniques. By processing these extractions through a recognition module, it might be possible to automatically figure out who the people are. The program might also be able to determine how many people are in the room, and what changes were made to the room during each visit.

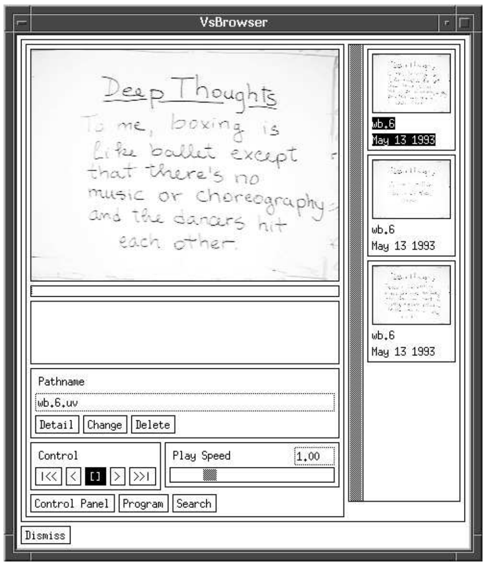

Figure 4: Browsing the output of the Whiteboard Recorder.

6.2 The Whiteboard Recorder

We have also written *The Whiteboard Recorder*, an application that keeps a history of changes to an office whiteboard. It works by taking video from a stationary camera aimed at the whiteboard and filtering it. By following a simple set of rules, the filtering distills the video into a minimum set of images. A browser (Figure 4) can be used to view the saved images.

The whiteboard recorder uses motion analysis to distinguish between the person writing on the board and the writing itself. Live video captured from a fixed camera is processed so that transient image features are filtered out, and only relatively stationary features are retained. This filters out people passing between the camera and the whiteboard. The filtered images are reconstructions derived from many partial images of the whiteboard — it is possible to capture an unobstructed view of the board from a sequence of input images where the board is partially occluded at all times.

From these relatively stationary images of the whiteboard, the program next distinguishes changes to the whiteboard due to writing from changes due to erasing. The system saves away images that represent "peaks" in the information written on the board.

6.3 The Video Rover

We have built *The Video Rover* to explore additional uses, such as remote sensing and telepresence, for live video networking. The Video Rover is an untethered vehicle that communicates with the VuNet using wireless transmission. Built from off-the-shelf electronics, the rover can be driven from a computer console on the VuNet.

The Video Rover consists of a video camera mounted on a small remote-controlled car and a wireless video link. We have integrated the radio controller with the VuSystem so that the car can be controlled from a VuSystem application, with the rover returning video feedback to driver of the vehicle.

7 Content-Based Processing Of Television Programs

We expect a significant portion of data carried on wide-area broadband networks to be pre-produced media similar to broadcast television. We have used the ViewStation to explore the potential of media processing applications to support content-based retrieval of pre-recorded television broadcasts.

We have developed content-based media browsers that use textual annotations that represent recognizable events in the video stream. These annotations are analyzed and processed to create higher level representations that may be meaningful to a human user. Finally, these representations are matched against user queries to generate an interactive presentation in the form of a browsable set of relevant video clips.

Annotations are generated through the recognition of audio or video cues from the media stream, or by the extraction of ancilliary information included in the stream, such as closed captions. The News Browser, The Joke Browser, and The Sports Browser are built on increased levels of processing of these annotations.

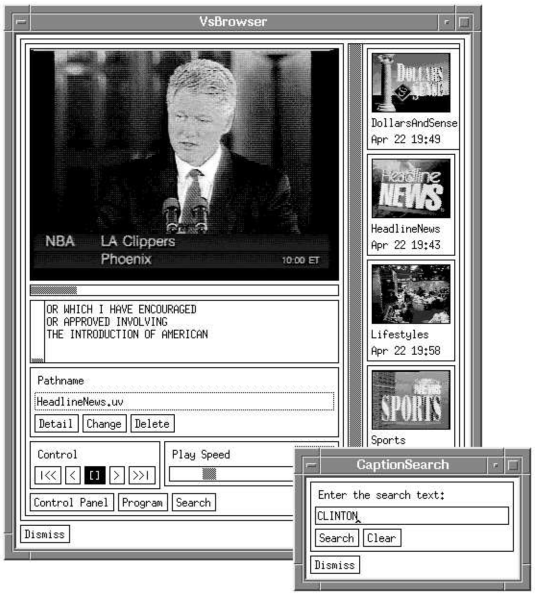

Figure 5: The News Browser.

7.1 The News Browser

The News Browser provides interactive access to a simple database of broadcast television news articles. Live television news programs such as CNN Headline News are automatically captured to disk at regular intervals. The stories are viewed with a video browser program (Figure 5).

News stories that are closed-captioned can be retrieved based on their content. Many broadcast television programs are closed-captioned for the hearing-imparied. Closed-captions provide a text translation of the audio component of the program — a significant amount of information. We wrote closed-caption capturing code for the Vidboard [6]. The caption information is extracted from the digitized video signal and converted into a common format so that modules capable of processing it can be constructed.

The news browser makes direct use of the closed-captioned annotations. A text search specification supplied by the user causes the browser to jump to stories with captions that match.

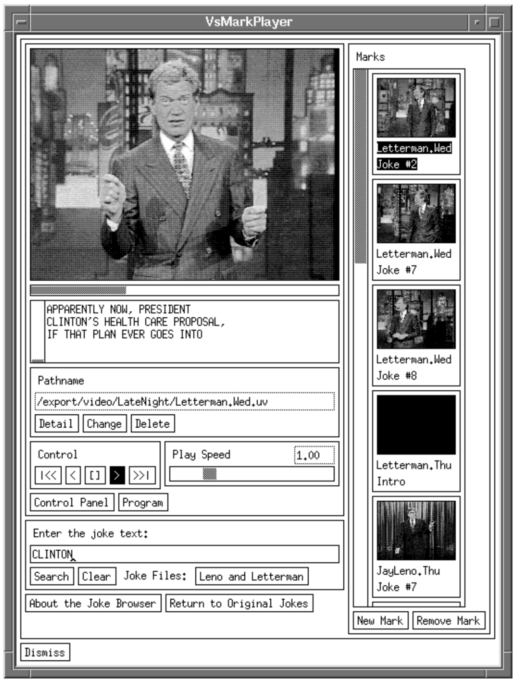

Figure 6: The Joke Browser.

7.2 The Joke Browser

We have developed *The Joke Browser*, which further demonstrates the potential of content-based media processing using closed-captions [2]. It records late-night talk show monologues, and segments them into jokes by processing the closed-captioned text. A special browser program (Figure 6) is queried to select all the jokes on a certain topic that have been made in the last week.

The Joke Browser extracts information from a recorded monologue through the analysis of the closed-caption data. In addition to the text of the jokes, the closed-captions contain hints to the presence of audience laughter and applause. A joke parsing module groups captions into jokes. This module is program specific, as it uses knowledge of the format of a particular program to make its grouping decisions.

7.3 The Sports Highlight Browser

We have developed *The Sports Highlight Browser*, which segments a recorded sporting news telecast into a set of video clips, each of which represents highlights of a particular sporting event. Video highlights of a particular game can be requested with a browser as shown in Figure 7.

The sports highlight browser demonstrates the feasibility of content-based media processing using graphical cues. Instead of closed-captioned text, this application generates its annotations through the examination of the video imagery. In particular, it marks frames in the video sequence that match graphical templates.

The annotation analyzer is built with assumptions about the format of a sports telecast. In particular, this analyzer depends on the news cliche of first an anchor person, then a set of narrated

Figure 7: The Sports Highlight Browser.

video clips, and finally a scoreboard graphic. The analyzer groups into a highlight the video sequence that falls between two scoreboard graphics. The analyzer labels each highlight with the names of the teams that competed.

8 The Media Server

We developed *The Media Server* to extend the reach of our applications to wide-area networks. It integrates the *World Wide Web* with the VuSystem and VuNet. By leveraging off of the network and operating system portability of the Web, and its straightforward browsing clients, The Media Server provides a publicly accessible interface to selected ViewStation applications.

Our server appears to Web users as a series of pages culminating in a form that leads to video display. The Web pages act as a navigational interface to applications, using *forms* to select program options. Figure 8 shows the form used to launch a live video processing program. When a form is submitted, an application appears as though an external viewer were spawned, but without a downloading delay.

The Media Server is implemented as an HTTP server and an associated set of scripts. The scripts customize Web pages to reflect available resources and characteristics of the client. To manage network and computational load, they distribute the video processing applications across a cluster of a dozen workstations. Video files can be viewed by many clients simultaneously, but live video sources are restricted to one client at a time. The video itself is distributed using the X Window System, and audio is distributed with AudioFile [7]. This approach provides wide-area accessibility at the cost of reduced performance.

Our media server has been operational since January 1994. It serves over 10,000 HTTP requests per day, distributing video to viewers in many countries. It is accessible through the URL `http://tns-www.lcs.mit.edu/vs/demos.html`.

9 Implications for Network Traffic

To understand what these media-processing applications imply for network traffic, we have developed three reference models for our media processing application components. The first model is that of

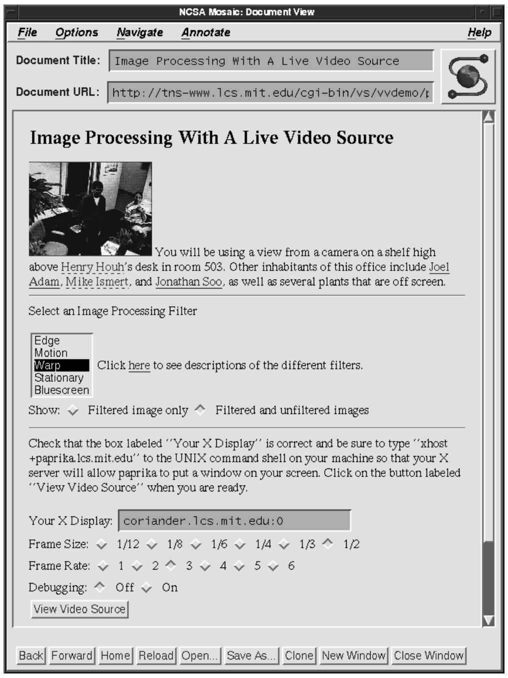

Figure 8: Launching an Image Processing program from the World Wide Web.

an *intelligent video capture* application component, the second is that of a *video browser* application component, and the third is of a *direct-viewing intelligent video processing* application component. These models demonstrate that the shape of network media traffic is not necessarily that of long continuous streams of media data requiring tight bandwidth guarantees.

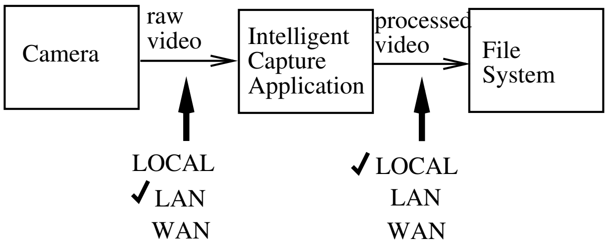

Figure 9: The reference model for an intelligent video capture application component.

9.1 Intelligent Video Capture

The reference model for an intelligent video capture application component is shown in Figure 9. This model comprises a camera or other video capture device, an intelligent capture program, and a file system. In this model, both the video capture device and file system can be connected to the intelligent video capture program through a network.

The data traffic from the camera or other video capture device is generally that of a periodic trains of video data packets. The connection between the camera and the intelligent media capture program can be through a local computer bus, a LAN, or a WAN. In the ViewStation, the connection is usually through the VuNet, our ATM LAN. This connection passes raw uncompressed video.

The connection between the intelligent video capture application and the file system can be through a local computer bus, a LAN, or a WAN. In the ViewStation, the connection is usually local to the computer. The data traffic from the intelligent media capture program to a video display or file system is not necessarily a periodic sequence of video data packets. In the extreme case of the Whiteboard Recorder application, only single video frames are saved on disk.

Processed video traffic most closely resembles traditional file system traffic, such as that observed with NFS. For many applications, the network between an intelligent media capture and the file system need not provide *guaranteed* bandwidth. With sufficient buffering at the output of the intelligent media capture program, a bursty network with acceptable *average* bandwidth is enough. For appplications like the Whiteboard Recorder and the Office Monitor, relatively short fragments of video are recorded. Here, local buffering at the output of the capture program is quite cheap.

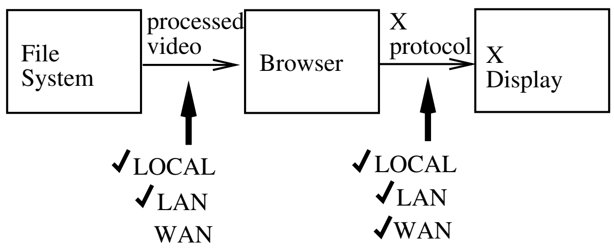

Figure 10: The reference model for a video browsing application component.

9.2 Video Browsing

The reference model for a video browsing application component is shown in Figure 10. This model comprises a file system, a video browsing program, and a video display. In the model, both the file system and the video display can be connected to the browsing program through a network.

The connection between the file system and the video browsing program can be through a local computer bus, a LAN, or a WAN. In the ViewStation, the connection is generally local or through the VuNet, our ATM LAN. The data traffic from the file system can have a wide variety of characteristics, depending on the application. The News Browser is used to view television news segments that can be several minutes long. If the browsing program had no local storage, some guarantee of bandwidth of the network would have been required. In the Whiteboard Recorder application, the browser views only one video frame at a time. A burst-tolerant network with good average performance is called for.

The connection between the video browsing program and the display can be through a local computer bus, a LAN, or a WAN. In the ViewStation, all three types of connections are used. Our video browser uses the X Window System for display. The display traffic from the video browsing application to the X display passes as much data as would come from the file system, but has a stronger requirement for the support of periodic transfers.

9.3 Direct-Viewing Intelligent Video Processing

The reference model for a direct-viewing intelligent video processing application component is shown in Figure 11. This model comprises a camera or other video capture device, an intelligent video processing program, and a video display. In this model, both the video capture device and the video display can be connected to the intelligent video capture program through a network.

As with the intelligent video capture model, the data traffic from the camera or other video capture device is generally that of a periodic sequence of video data packets. The connection between the

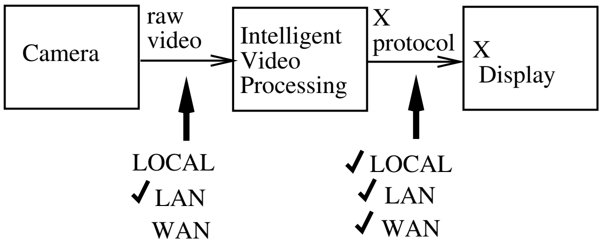

Figure 11: The reference model for a direct-viewing intelligent video processing component.

camera and the intelligent media capture program can be through a local computer bus, a LAN, or a WAN. In the ViewStation, the connection is usually through the VuNet, our ATM LAN. This connection passes raw uncompressed video.

As with the video browsing model, the connection between the intelligent video processing program and the display can be through a local computer bus, a LAN, or a WAN. In the ViewStation, all three types of connections are used. Our video browser uses the X Window System for display. The display traffic from the video browsing application to the X display passes as much data as came from the file system, but has a stronger requirement for the support of periodic transfers.

9.4 Usage Classes

Table 1 shows the network usage classes for the capture and browsing components of the ViewStation applications we have developed. The table indicates the video data format, network traffic type, bits transferred, subsession duration, and subsession frequency for example application components. For application components that pass data in periodic packet-trains for long periods of time, the number of bits transferred is indicated per second. Otherwise, the number of total bits transferred is indicated. We define a subsession as a single transfer of a video segment. One program session could include many subsessions. For example, a Joke Browser session consists of a subsession for each joke replayed.

For most application components, we use a video frame rate of 10 frames per second, a resolution of 320x240 8-bit pixels, and no compression. A video connection with these parameters averages approximately 6 Mb/s. To the network, this traffic appears as periodic packet-trains of 600 Kb per train, 10 trains per second.

Application components that capture video use a raw video data format, with periodic packet-train network traffic, transferring from 600 Kb/s to 6 Mb/s. Components that store and retrieve video use a processed video data format with file-like network traffic, transferring from 100 KB to 1.3 GB of video data. Application components that display video use the X Window System network protocol, with periodic packet-train network traffic, transferring from 600 Kb/s to 6 Mb/s.

9.5 Discussion

A wide variety of subsession durations and frequencies is indicated by the table. The duration column of the table indicates the duration of video traffic transmitted over a network for a subsession. This varies from a single video frame for Whiteboard Recorder storage and retrieval, to periodic packet-trains for capture components. The frequency column indicates the subsession frequency for each application component. This varies from once every few seconds to once per day.

Clearly, the transfer of video data over the net appears on the table in many forms. Raw video is only a fraction of the total traffic. We handle processed video as bursty traffic, because that is how it is generated by our applications, and also that is how our applications prefer to receive it. ViewStation applications generate bursty video traffic because they generate video segments based

Component	Video Data Format	Network Traffic Type	Bits Transferred	Duration of Subsession	Time Between Subsessions
Room Monitor					
capture	raw	periodic	6 Mb/s	days	-
storage	processed	file	100 Mb	seconds	minutes
retrieval	processed	file	100 Mb	seconds	seconds
display	X protocol	periodic	6 Mb/s	seconds	seconds
Whiteboard Recorder					
capture	raw	periodic	600 Kb/s	days	-
storage	1 frame	file	600 Kb	1 frame	minutes
retrieval	1 frame	file	600 Kb	1 frame	seconds
display	X protocol	interactive	600 Kb	1 frame	seconds
Video Rover					
capture	raw	periodic	6 Mb/s	minutes	minutes
display	X protocol	periodic	6 Mb/s	minutes	minutes
News Browser					
capture	raw	periodic	6 Mb/s	1/2 hour	daily
storage	processed	file	10 Gb	1/2 hour	daily
retrieval	processed	file	500 Mb	minutes	minutes
display	X protocol	periodic	6 Mb/s	minutes	minutes
Sports Highlights					
capture	raw	periodic	6 Mb/s	1/2 hour	daily
storage	processed	file	10 Gb	1/2 hour	daily
retrieval	processed	file	100 Mb	seconds	seconds
display	X protocol	periodic	6 Mb/s	seconds	seconds
Joke Browser					
capture	raw	periodic	6 Mb/s	10 minutes	daily
storage	processed	file	3 Gb	10 minutes	daily
retrieval	processed	file	100 Mb	seconds	seconds
display	X protocol	periodic	6 Mb/s	seconds	seconds

Table 1: The network usage classes for some ViewStation application components.

on events. They also prefer to receive bursty video traffic in order to process it in processor time slices.

Although much consideration has been made for the support for long-lived streams of so-called continuous traffic on broadband networks, support for short bursty traffic is an important issue. The presence of bursty traffic may have significant implications with respect to the distribution of memory buffers across the network, and the presence of short subsessions may affect the design of network signalling systems.

10 Conclusion

We have described applications built on the ViewStation, a distributed multimedia system based on Unix workstations and a gigabit per second local area network. These applications are built with the VuSystem, a programming toolkit that combines the programming techniques of visualization systems with the temporal sensitivity of traditional multimedia systems.

The Room Monitor, The Whiteboard Recorder, and The Video Rover are applications that directly manipulate live video. They provide more responsive human-computer interaction through the intelligent processing of live video.

We have also developed The News Browser, The Joke Browser, and The Sports Highlight Browser applications that operate on pre-recorded video. They demonstrate the practicality of automatic extraction to support content-based retrieval of produced video.

These applications may be called *computer-participative* applications, as opposed to traditional *computer-mediated* applications. We believe they represent a significant class of future applications for broadband networks. The VuSystem provides a unique foundation for their development.

Experience with the ViewStation has shown that a software approach preserves network and computational scalability and graceful degradation. Network-based multimedia systems that deliver audio and video data all the way to the application provide substantially more capability than systems that only allow data to be manipulated in a small set of pre-defined ways, away from the application, in the operating system kernel or server process, or by separate peripherals.

Many ViewStation application components manipulate relatively short video sequences. These components could use local buffering of video data and bursty network transfers of data, instead of periodic sequences of media data with bandwidth guarantees. In such a scenario, video traffic on a broadband network would be a *combination* of bursty transfers and periodic packet-trains.

References

[1] C. J. Lindblad, D. J. Wetherall, D. L. Tennenhouse, "The VuSystem: A Programming System for Visual Processing of Digital Video," *Proceedings of ACM Multimedia 94*, October 1994.

[2] D. R. Bacher, "Content-Based Indexing of Captioned Video," SB Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, May 1994.

[3] D. L. Tennenhouse, J. Adam, D. Carver, H. Houh, M. Ismert, C. Lindblad, W. Stasior, D. Wetherall, D. Bacher, and T. Chang, "A Software-Oriented Approach to the Design of Media Processing Environments," *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994.

[4] S. Wray, T. Glauert, and A. Hopper, "The Medusa Applications Environment," *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994.

[5] J. F. Adam, H. H. Houh, M. Ismert, and D. L. Tennenhouse, "A Network Architecture for Distributed Multimedia Systems," *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994.

[6] J. F. Adam, "The Vidboard: A Video Capture and Processing Peripheral for a Distributed Multimedia System," *Proceedings of the ACM Multimedia Conference*, August 1993.

[7] T. M. Levergood, A. C. Payne, J. Gettys, G. W. Treese, and L. C. Stewart, "AudioFile: A Network-Transparent System for Distributed Audio Applications," *Proceedings of the USENIX Summer Conference*, June 1993.

[8] Apple Computer Inc., "Inside Macintosh: Quicktime, Inside Macintosh: Quicktime Components," Addison Wesley, 1993.

[9] J. Le Boudec, "The Asynchronous Transfer Mode: A Tutorial," *Computer Networks and ISDN Systems*, pp. 279-309, May 1992.

[10] Microsoft Corporation, "Microsoft Video For Windows Users Guide," 1992.

[11] M. de Prycker, "Asynchronous Transfer Mode: Solution for Broadband ISDN," Ellis Horwood, 1991.

[12] J. K. Ousterhout, "Tcl: An Embedded Command Language," Computer Science Division (EECS), University of California, Berkeley, CA, January 1990.

[13] G. Finn, "An Integration of Network Communication with Workstation Architecture," *ACM SIGCOMM*, pp. 18-29, October 1991.

[14] M. Hayter and D. McCauley, "The Desk Area Network," *ACM Operating Systems Review*, pp. 14-21, October 1991.

[15] A. Hopper, "Pandora - an Experimental System for Multimedia Applications," *ACM Operating Systems Review*, 24(2):19-34, April 1990.

[16] C. Williams and J. Rasure, "A Visual Language For Image Processing," in *IEEE Computer Society Workshop on Visual Languages*, Skokie, Illinois, 1990.

[17] C. Upson, T. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam, "The Application Visualization System: A Computational Environment for Scientific Visualization," in *IEEE Computer Graphics and Applications*, pp. 30-42, July 1989