The Network Video Terminal*

Dorgham Sisalem
Henning Schulzrinne
GMD FOKUS
Hardenbergplatz 2
10623 Berlin, Germany
{sisalem, schulzrinne}@fokus.gmd.de

Christian Sieckmeyer
Institute for Telecommunication
Dept. of Electrical Engineering
Technical University of Berlin
sieckmeyer@fokus.gmd.de

Abstract

Currently, a variety of the MBONE video tools provide video conferencing capabilities on different platforms and with a variety of compression algorithms. However, most of these tools lack the ability to interact with other media agents that might be used during a conferencing session. Such interaction is required, for example, for achieving lip synchronisation between audio and video streams or for quality of service control.

In this paper, we present a new video tool, NEVIT. *This tool provides the basic capabilities needed for video conferencing services such as video capturing, compression and decompression engines and multicasting and ATM network interfaces. To ease the interaction with other media agents,* NEVIT *incorporates a message handling facility to interact over a local conference bus with other media agents, a floor controller or the conference controller. Currently, we are working on adding lip synchronisation and quality-of-service control using this conference bus.*

1 Introduction

When looking at the currently available multimedia conferencing tools, we can distinguish roughly two approaches. First, there are large, monolithic tools that support different tasks such as video, audio and application sharing. Such tools consist either of a single program or a tightly integrated set of applications that can only interoperate within the set. Adding new features to such tools or upgrading the media agents by, say, replacing a video agent by a faster one is rather difficult. Secondly, there are loosely coupled tools such as the Internet multicast (MBONE) tools [2, 4]. In this approach, each media is handled by a distinct media agent. Conference control issues such as floor control, joining and leaving a conference and starting the appropriate agents is delegated to an external conference controller. With such an approach, the media agents can be easily replaced, updated and reused. The main drawback of this second approach is that once the conference controller has started a media agent, this media agent is on its own and the controller is no longer in control of it. This means that the user has to employ a different user interface for each application with all of the interfaces doing in part common tasks such as initiating or terminating a service or displaying the members of the session. A more severe drawback of using loosely coupled tools is that these tools do not interact with each other.

In Section 2 of this paper, we describe a local conference control architecture and a communication protocol that allow to tie together different media agents into a single conferencing application. In Section 3, we present a new video tool, the network video terminal (NEVIT), that uses this scheme to interact with other tools. While this tool is still in its early stage of development, it already offers features that are not provided for in other MBONE tools such as direct ATM connectivity besides IP multicast, different software codecs and the ability to display video streams of a variable number of conference participants. To avoid the need of using different user interfaces for different media agents, we use a common conference controller, the integrated session controller (ISC). ISC supports any number of media agents, using a unified communication protocol for controlling them.

As a first step in utilising the video tool as well as the conferencing protocol we investigate different aspects of guaranteeing a certain level of quality of service to the video streams through reservation and adaptation to the network conditions in Section 4. In Section 5, we describe some experiments and implementation ideas for synchronising audio and video streams at the receiver.

Finally, some experimental work and measurements concerning the performance of video transmission over ATM compared to traditional protocol stacks such as

*This work was funded in part by DeTeBerkom.

UDP/IP/Ethernet are presented in Section 6. From those measurements we conclude that the additional processing overhead required for the protocol stack UDP/IP/AAL5 compared to native ATM mode using only AAL5 is only of significance for applications consuming large amounts of bandwidth and sending data in small packets.

2 Local Conference Control Architecture

As already mentioned, the loosely coupled tools approach employed with the MBONE tools allows for flexible and easy upgrading and reusing of the same tools within different conferencing styles. For example, the same video application might be used within seminar-style conferences initiated from session directories or telephone-style, invitation-based conferences. However, this flexibility is achieved at the cost of complicating the interaction between the different applications used within the conference.

Such interaction is needed, for example, to achieve intermedia synchronisation, as discussed in Section 5. Other examples include quality of service control, automatically displaying only the conference participants that are actively talking or using the display coordinates of video windows to control artificial spatial placement of the audio from individual speakers (artificial stereo, holophony) [3, 6, 7].

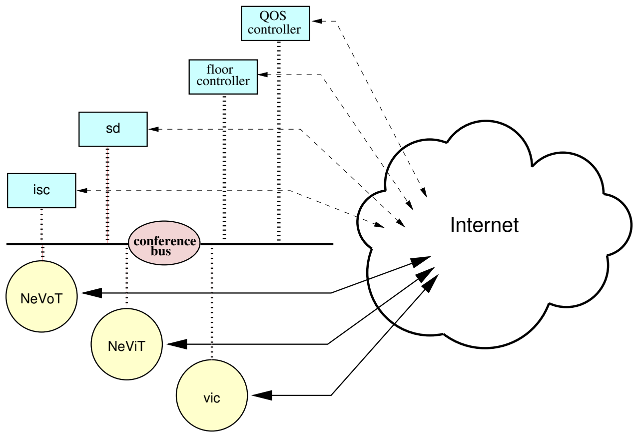

Figure 1. Example multimedia conferencing control architecture

In an earlier paper [11], one of the authors introduced a communication protocol that can be used for the initiation and termination of media agents, for establishing the parameters of the session (such as a unique session number and the email addresses of the participants) as well as the dynamic control of the parameters of the media agents, such as bandwidth and frame rate of a video source or the compression algorithm. The messages used in this protocol are sent as ASCII text and are formatted to be directly interpretable by a standard Tcl interpreter [9].

There are different approaches for implementing a local conference bus. We have chosen the configuration depicted in Fig. 1, in which messages are exchanged using local IP multicast. That is, packets are sent with the time-to-live (ttl) value set to zero. In another approach, a replicator process listens to a well known TCP port for messages and sends them to all media agents and controllers that have expressed interest in that particular type of message. This latter method has the advantage that only the processes that have expressed interest in a certain message type are woken up instead of all processes subscribed to the multicast address. However, this adds another process, which also needs to keep track of which client processes are still alive.

3 The Network Video Tool (NeViT)

The first MBONE video tools were the Xerox PARC Network Video tool (nv) [5], and the INRIA video Conferencing System (ivs) [16]. While both of these tools were intended for supporting low bit-rate multicast over the Internet, they choose different compression algorithms and video representations. As extending any of the two tools to support the other's compression style was a non-trivial task, both tools were non-interoperable. More recent tools such as VIC[8] support both hardware and software compression engines and offer better performance.

VIC supports a limited form of interaction with the VAT audio tool: VAT can signal speaker activity through a mechanism similar to the conference bus. The speaker activity indication then selects which site is displayed in full size.

However, in MBONE conferences, much of the participant information and control is duplicated across several media agents. Rather than a participant-centric view (which user is using which media), MBONE tools encourage a media-centric view, with each media agent displaying a separate list of participants, often using different ways of identification. It is also difficult to integrate media agents into new, domain-specific applications with their own user interface. Based on these observations, we designed NeViT with no graphical user interface at all. Instead, all functions of the tool, including establishing and terminating sessions, are controlled by messages sent on the conference bus. These messages constitute a well defined communication protocol as was defined in [11]. Through the messages NeViT can either communicate with the conference controller or with any other media agent listening to the local conference bus and capable of interpreting the messages.

In a first approach, we tried to implement the message handling interface into one of the existing video tools, namely VIC. However, extracting the graphical user interface of the tool and replacing it with the message handling one turned out to be more complicated than we expected. The user interface was tightly interwoven with the whole program. So, with nearly no documentation describing the tools architecture, implementing a new tool seemed to be an easier solution. With our own simple tool experimenting with new ideas and adaptation to our conferencing protocol was straight forward.

3.1 The Integrated Session Controller (ISC)

As NEVIT has no user interface of its own we use a session controller to provide for the necessary actions of initiating and terminating a session, displaying or ignoring the data received from a participant, choosing the appropriate compression algorithm and adjusting the bandwidth. Each interaction with ISC results in the sending of a message describing the desired action. Fig. 2 shows a simple example for such an interaction. Each message contains a hierarchal session identification and the message body. The identifier "s/video/1" used in Fig. 2 denotes the conference (s), the media type (video) and the instance of the media. The first message "session" creates a blank media session and starts an appropriate media application, in this case a video tool. The Application responds with a "created" message. With the "statistics" message the application is asked to respond with some of the measured values such as the data rate of outgoing or incoming video streams or the loss rate. Finally, with the "close" message a media session can be closed.

As the controller is not explicitly integrated with NEVIT it might actually be used to control any other media agent that uses the conference bus and the here used communication protocol. ISC offers, for example, a user interface for calling and terminating an audio session as well as controlling the codings and different other characteristics of the audio agent.

3.2 Implementation Issues

While our video tool supports different compression algorithms, multicasting and the ability of handling a variable number of video streams, our main goal was to produce a flexible tool that can easily be extended to achieve intermedia synchronisation, automatic quality of service control and interaction with other media agents without necessarily being dependent on those agents. The tool can be roughly divided into three parts: routines that process messages arriving on the local conference control bus, the routines dealing with network protocols including RTP and the video compression/decompression routines.

In implementing the conference control bus, some care is necessary to avoid locking out commands from the conference control bus during high-rate video processing. This lock-out is possible due to the use of event-based programs

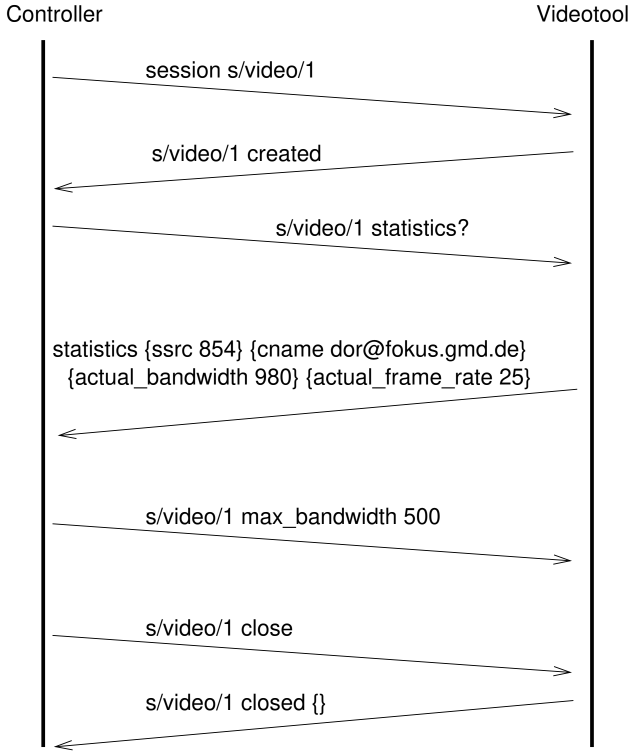

Figure 2. Example for using the conferencing protocol

in Unix and X11. More precisely, Unix applications using X11 or processing data from several sources typically use the `select()` event multiplexer as their main loop. When one or more sockets have data waiting to be read, `select()` returns a bit mask with the indices of sockets with data waiting. If the event handler always checks first for the socket on which video data arrives and the CPU can barely keep up with the compression or decoding of the data, the conference control bus socket may never be read as there is always video data waiting. Thus, the conference controller looses control over the application. Unfortunately, with most event handling packages such as the Tcl or Xlib routines, the programmer cannot predict the order of event processing, so that experimentation is required to ensure that the socket handling the conference control bus is always checked first for messages. Alternatively, a threads-based approach can be used, but requires greater care in managing concurrent access to data structures.

For transmitting video data, we use the real time protocol (RTP) [12] designed within the Internet Engineering Task Force (IETF). RTP is an end-to-end protocol that is often used together with other transport protocols, in particular UDP. RTP has no notion of a connection; it may operate over either connection-oriented (say, ATM AAL5) or connectionless lower-layer protocols (typically, UDP/IP). It does not depend on particular address formats and only requires that framing and segmentation is taken care of by lower layers. RTP offers no reliability mechanisms. It is typically implemented as part of the application or as a library rather than integrated into the operating system kernel.

RTP sessions consist of two lower-layer data streams, namely a data stream for audio and video, say, and a stream of control packets (using the sub-protocol called RTCP). In UDP, data and control streams use separate ports, however, they may be packed into a single lower-layer stream as long as RTCP packets precede the data packet within the lower-layer frame. A single stream may be advantageous for systems where connections may be costly to manage, e.g., ATM PVCs. Since the control streams consume only a small fraction, typically around 5%, of the data bandwidth, resource reservations are not seriously affected. (NEVIT currently uses two separate VCs for data and control packets.)

RTP data headers allow to distinguish dynamically between different audio and video encodings and has a marker to delineate video frames and audio talk spurts.

The control protocol (RTCP) allows monitoring of the received and transmitted data rates, delay jitter and packet losses. Each session member periodically multicasts control packets to all other session members. RTCP packets also contain identifying information that allows to connect different media streams with the same participant.

3.2.1 Video Handling

NEVIT currently only supports the SunVideo card for capturing and compressing video images. The card supports JPEG, MPEG, CellB and YUV video [15]. NEVIT on the other hand, provides the appropriate algorithms for decompressing and displaying JPEG and YUV video images.

The JPEG decoder was ported from the VIC video tool. It is based upon the JPEG decompression code provided by the Independent JPEG Group and enhanced with conditional replenishment. With conditional replenishment, only those parts of a frame that have changed are actually decoded, resulting in a faster decoder and lower processing overhead. On a Sun SPARC 20/712, we manage to receive, decompress and display JPEG frames at the full rate of 30 frames/s. The user-level handling consumes around 70% of the available processor capacity. The network data rate is about 1.3 Mb/s.

Currently, we are working on adding MPEG-1 compression and YUV receive capability to NEVIT. YUV directly encodes luminance and the subsampled-by-two chrominance values (4:2:2), with four bytes for two pixels. While the rendering of YUV data is simple, a YUV-coded video with 30 240x320 frames/s generates about 30 Mb/s, making it suitable mainly as a test load for ATM networks. (Compressing this video stream into JPEG frames with a quality factor of 50 yields a data rate of about 1-1.5 Mb/s.)

4 Quality of Service Enhancement Through Reservations and Adaptation

Video conferencing applications have two characteristics in terms of their quality-of-service requirements. On the one hand, audio and video transmissions requires a minimal guaranteed bandwidth and, for interactive sessions, an upper bound on the end-to-end delay. On the other hand, video conferencing applications can adapt to a wide range of available bandwidths to provide increased perceptual quality. This section discusses how reservation and adaptation were implemented in NEVIT.

4.1 Resource Reservations using RSVP

Bandwidth reservation is important for videoconferencing to guarantee a minimal level of uninterrupted service throughout the session. For resouce reservations, NEVIT makes use of the receiver-based RSVP [17], building on the ISI RSVP implementation (version 3.2). The data flow is illustrated in Figure 3. The session controller sends a 're-serve' message containing flow specifications on the local conference bus, which is picked up by NEVIT. Through a library interface, NEVIT communicates with the local RSVP daemon, which then forwards RSVP messages into the Internet.

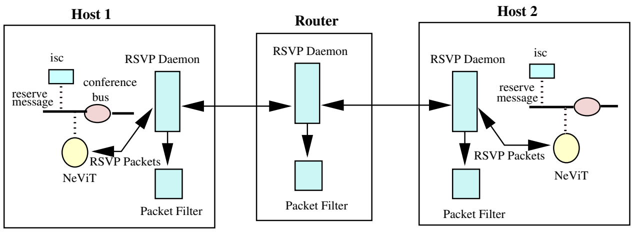

Figure 3. Reservation scheme using RSVP with NEVIT

The user specifies both a desirable and a lower threshold below which he would rather not use video at all. NEVIT first tries to reserve the desired level, but falls back in stages to the lower threshold if that reservation fails due to lack of resources. Note that a user could also decide to just rely on "best effort" transmission. (Usually, only a fraction of the total bandwidth is available for reservations. Thus, even if the reservable bandwidth has been exhausted, video communication may still be possible.)

Instead of integrating the RSVP interface with NEVIT, we are also considering building a separate reservation agent that would listen to reservation and connection requests on the local conference bus and communicate with the RSVP daemon on behalf of any number of media agents. This would have the advantage of being able to retrofit existing media agents with RSVP capability. Due to the conference bus design, this can be done without changing ISC.

4.2 Adaptive Applications

For a number of years, reservation will not be supported in large parts of the Internet, in particular since bandwidth reservation requires new charging and settlement mechanisms. If reservation is not available, the conference participants or at least the conference organizer currently have to manually set the appropriate bandwidths for the different media, currently with little guidance from the media agents. Instead of guessing at the appropriate bandwidth setting and then suffering or causing unnecessary packet loss, applications should adapt themselves to the available network bandwidth. Also, even if resource reservation is available, it may be desirable to attain the best possible quality of service given current network load and fair sharing of bottleneck bandwidth.

For these reasons, NEVIT implements a bandwidth adaptation algorithm that tunes the frame rate to achieve different transmission data rates. Adapting bandwidth to current network conditions requires the exchange of state information between the source and the network nodes or the receivers. The first approach is used for the ATM available bit rate service [13], while the RTP control protocol provides periodic loss feedback from the receivers. Reporting intervals for each receiver range from five seconds to a minute or more. Thus, load spikes can lead to sustained packet losses or unfairness in bandwidth usage[1]. On the other hand, the bandwidth of audio and video sources should only be adjusted over longer time periods to avoid perceptually annoying rapid quality changes, e.g., rapidly changing frame rates.

Using the loss information sent within the RTCP packets, a sender can estimate the congestion state in the network. Whenever the loss rate reported in the RTCP packets is above a predefined threshold, the source reduces its rate by a multiplicative reduction factor. If the loss rate drops below a setable threshold, the source increases its rate additively. A detailed description and some results for different network topologies can be found in [1].

[1] Those senders with receivers who happen to discover the packet loss first reduce their bandwidth disproportionally.

4.3 Combining Resource Reservation and Adaptation

Adaptation and bandwidth reservation can also cooperate. A participant would reserve the minimal useful bandwidth, probably at a price premium above standard "best effort" service, but send data in excess of this reserved bandwidth. The fraction of the stream exceeding the reservation will be carried as "best effort" packets. If the network is congested, they will be dropped and, through RTP-based source adaptation, cause a reduction in sending bandwidth. If the network has spare capacity, the "best effort" packets will experience little loss and the sender can probe for additional bandwidth by increasing its rate.

5 Audio-Video Intermedia Synchronisation

5.1 Components of End-to-End Delay

Delay variations cause the timing characteristics of audio and video streams to be distorted at the receiver, e.g., packets that were generated at constant intervals reach the sender with varying interarrival times. The receiver must use a playout buffer to ensure periodic delivery to the audio or video output device. This process of re-establishing the original timing at the receiver is referred to as intramedia synchronisation. The depth of the buffer is adjusted dynamically [10, 14], based on on-line estimates of the delay jitter. Increased buffer depth causes fewer packets to miss their playout time, but also increases end-to-end delay.

If lip sync between audio and video is desired, the end-to-end delay of the audio and video streams has to be the same. Lip sync is an important example of intermedia synchronisation. The end-to-end delay consists of the time to acquire and compress an audio block or a video frame[2], the network propagation delay, any software decompression delays, the playout delay and the delay from writing the audio or video data to the output device until the speech or video can be heard or seen.

None of these delay components are guaranteed to be the same for two streams, even when they originate at the same host. For example, it is unlikely that the playout delay for two different media streams will be the same, even if they traverse exactly the same route, due to statistical fluctuations in the delay jitter and different playout delay adjustment algorithms. As will be discussed later, audio and video capture and playout delays also differ significantly from each other.

Intermedia synchronisation requires that the receiver has information about the absolute time a particular media segment was generated. RTP provides this information by having senders periodically transmit, as part of the RTCP sender

[2] We will refer to these collectively as media segments.

report packets, mappings between a media time stamp and an absolute (wallclock) time. Each RTP packet contains a media time stamp. However, this is not sufficient. At the receiver, different media agents have to agree on at what time $t + \delta$ to play back a media segment that was generated at some time t. We are working on using the local conference control bus to have media agent i announce its preferred values of δ_i. Each media agent then actually uses the maximum of the δ_i's seen on the local conference bus.

As mentioned earlier, media agents have to know their capture and playback delays. These are defined as follows. The capture delay is the time it takes the media sender to acquire and compress a media segment, while the playback delay is the time from the moment that the receiver calls the routine that display a video frame or plays back a block of audio until the video frame is seen or the audio block is heard. The capture delay enters the computation of the mapping between media timestamps and absolute time, while the playback delay is added to the playout delay to compute δ_i.

Typically, video hardware and compression takes significantly longer than audio processing. In two experiments, we measured the video capture and playback delays. While the values measured are specific to the hardware used, the methods are generic. We are currently investigating how to measure audio delays.

5.2 Measuring Capture Delays

The measurement setup is shown in Fig. 4. A process running on the workstation increments a counter every millisecond and displays the current counter value and the local system time t_d on a standard terminal window. A camera is focused on the workstation screen. The analog video signal from the camera is digitised and possibly compressed by the video capture card whose capture delay is to be measured. A second process reads and saves the compressed video frames and notes the system time t_a when the frame was read by this process. Afterwards, the difference between this time value t_a and the time shown in the captured image, t_d yields the capture delay.

This measurement suffers from some inaccuracies, including the unknown delay of writing values to a terminal window, which, at the average is the display refresh period of about 20 ms.

Results for a SunVideo card performing JPEG compression are shown in Fig. 5. The mean capture delay is about 79 ms, but the delay measured varies due to measurement inaccuracies, different process scheduling delays and other factors between about 75 ms and 85 ms.

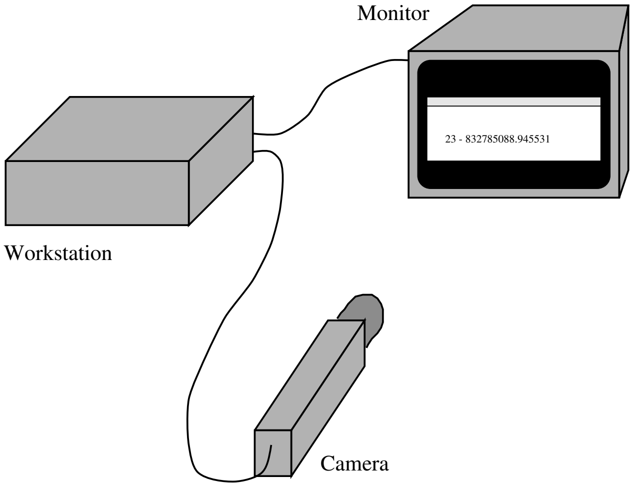

Figure 4. Measurement setup for estimating the video capture delay

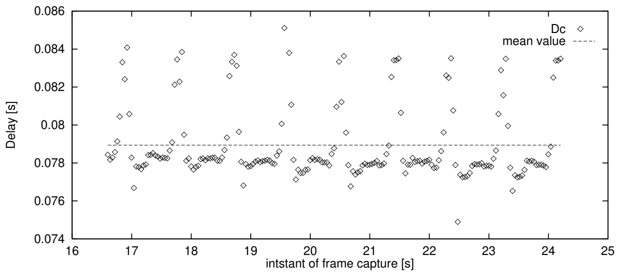

Figure 5. Delay between a visual event and its capture by the frame grabber

5.3 Measuring Playback Delays

The measurement setup for estimating the video playback delay is shown in Fig. 6. First, the display of a frequency counter, which counts pulses from a standard laboratory frequency generator running at a rate of 1 kHz was captured as a sequence of JPEG video images. The JPEG images are stored together the value of the counter displayed on each image.

The workstation then displays the images as quickly as possible using the software video decompression routines. Each time a video frame is rendered, the current system time and frame identifier is also written to a terminal window and saved in a log. The workstation monitor output is scan converted to PAL analog video and recorded on a studio video tape deck with single-frame display capability. The video frame shown on the monitor will be one written to video output routine some display cycles ago. The recorded video tape is then analysed to determine which image identifier is visible together with which displayed frequency counter image. From the video tape, we can see what time it was when, say, counter image "373" was visible on the screen. From the log, we know when that image was handed to the output routine and can thus compute the delay as the difference between those two timestamps.

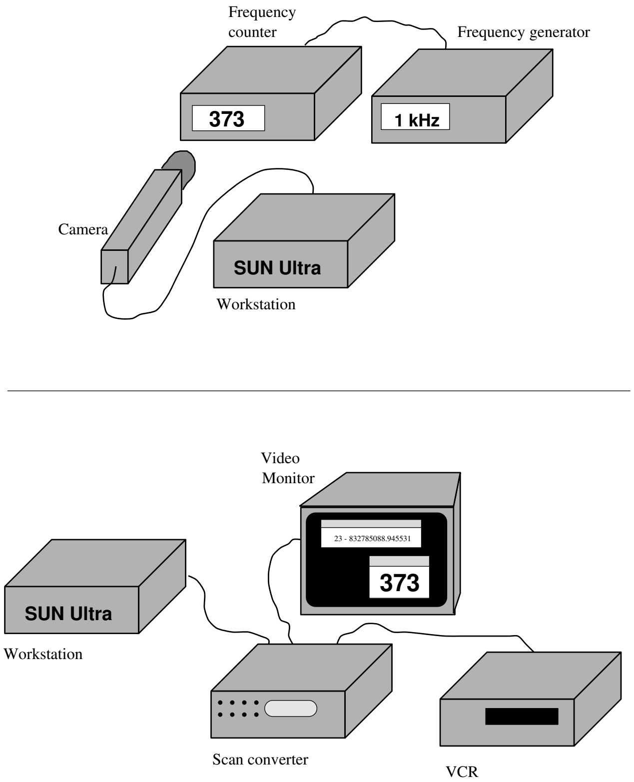

Figure 6. Measurement setup for estimating the video playback delay

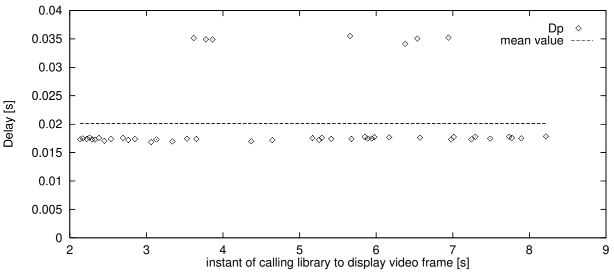

Figure 7. Delay between calling video driver to show a frame and actual display on the monitor

From Fig. 7, we can see that image display is much faster than capture, with delays of around 20 ms on average, varying between 17 ms and 35 ms. The timestamp is only displayed for every image shown, limiting the delay resolution. (Since all decompression is done in software, the delay equals the time between writing two images, namely 17 ms.)

6 Video Transmission over ATM and IP

NeViT supports sending RTP-encapsulated video frames directly over ATM AAL5 point-to-point connections, without IP. This allowed us to compare the performance of NeViT as a video sender using different protocol stacks. We evaluated the cases UDP/IP/Ethernet, UDP/IP/AAL5 and AAL5.

The measurement setup, consisting of two Sun SPARC 20 workstations with Fore SBA 200 ATM cards, is shown in Fig. 8. The two workstations were connected to each other either through a switched Ethernet or an ATM connection. For the ATM connection, the workstations were connected directly, back-to-back through PVCs, without a switch. We used a Sun NTSC video camera attached to a SunVideo board to acquire 320-by-240 video frames with a maximum frame rate of 30 Hz.

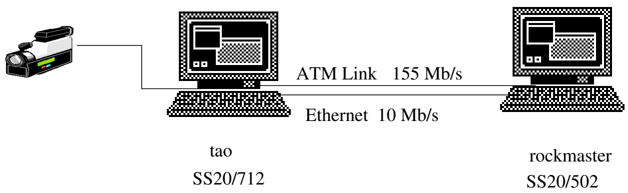

Figure 8. Throughput test setup

We measured the data rate at the sender by capturing and transmitting 2000 video frames as quickly as possible using the three different protocol stacks. Video frames were distributed over several AAL5 frames or UDP packets for segment sizes of 1, 2 or 8 kbytes. YUV video frames were 153.6 kbytes long, while JPEG frames had an average size of 6 kbytes. For YUV and 1 kbyte frames, the sender generates about 4600 packets per second. The results for YUV are shown in Fig. 1; frame rates can be readily computed from that table.

Table 1 shows that the achievable data rate[3] depends strongly on the protocol stack and that per-packet processing costs are significant since the cost of data copies is the same for all YUV cases. Since the AAL5-only results differ little across segment sizes, the UDP/IP stack seems to by the major contributor to the overall protocol processing overhead. For Ethernet, the sender rate is limited by the rate supported by the Ethernet, i.e., the sender process blocks (rather than

[3]The frame rate can be readily computed from the data rate since each YUV frame has the same number of bytes.

Protocol stack	YUV		
	1 kbyte	2 kbytes	8 kbytes
UDP/IP/Ethernet	8.6	8.6	8.6
UDP/IP/AAL5	14	28	35
AAL5	29	36	36

Table 1. A comparison of the achievable data rate in Mb/s with UDP/IP/Ethernet, UDP/IP/AAL5 and AAL5 as a function of segment size for YUV

simply dropping excess packets) when the Ethernet output buffer is full.

For JPEG, the data rates (about 1.3 Mb/s) and packet rates (between 180 and 30 per second) were low enough so that all three protocol stacks supported full-rate video transmission.

7 Summary and Future Work

NEVIT is currently hardware dependent and supports only the Sun workstations platform equipped with the SunVideo card. We are working on adding support for a number of video cards and displays. To increase its flexibility we consider adding additional compression engines to the tool such as CellB, H.261 and H.263. However, our main goal is to investigate the possibility of building a set of loosely coupled conferencing tools that offer integrated video and audio services.

NEVIT and NEVOT, the audio tool, will shortly support intermedia synchronisation between audio and video. With NEVIT, resource reservation and adaptation can ensure acceptable video quality, full bandwidth utilisation while avoiding congesting networks that do not support reservations. The possibilities of integrating reservations, particularly reservation mechanisms that exploit the soft-state nature of RSVP to probe for available bandwidth, and adaptation are now under investigation.

Acknowledgements

Bernd Deffner contributed to the writing of NEVIT as well the application control implementation and its testing.

References

[1] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, Jan. 1996.

[2] S. Casner and S. Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.

[3] M. Cohen and N. Koizumi. Exocentric control of audio imaging in binaural telecommunication. *IEICE Transactions on Fundamentals*, E75-A(2):164–170, Feb. 1992.

[4] H. Eriksson. MBone – the multicast backbone. In *Proceedings of the International Networking Conference (INET)*, pages CCC–1 – CCC–5, San Francisco, California, Aug. 1993. Internet Society.

[5] R. Frederick. Experiences with real-time software video compression. In *Sixth International Workshop on Packet Video*, Portland, Oregon, Sept. 1994.

[6] N. Kanemaki, F. Kishino, and K. Manabe. A multi-media teleconference terminal controlling quality of flow in packet transmission. In W. A. Pearlman, editor, *Visual Communications and Image Processing IV*, volume 1199, pages 259–266, Philadelphia, Pennsylvania, Nov. 1989. Society of Photo-Optical Instrumentation Engineers.

[7] S. Masaki, T. Arikawa, H. Ichihara, M. Tanbara, and K. Shimamura. A promising groupware system for broadband ISDN: PMTC. *ACM Computer Communication Review*, 22(3):55–56, Mar. 1992.

[8] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, Nov. 1995.

[9] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, Reading, Massachusetts, 1994.

[10] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pages 680–688, Toronto, Canada, June 1994. IEEE Computer Society Press, Los Alamitos, California.

[11] H. Schulzrinne. Dynamic configuration of conferencing applications using pattern-matching multicast. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science (LNCS), pages 231–242, Durham, New Hampshire, Apr. 1995. Springer.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, Jan. 1996.

[13] S. S. Shirish. ATM Forum traffic management specification version 4.0. Technical Report 94-0013R6, ATM Forum, June 1995.

[14] C. Sieckmeyer. Bewertung von adaptiven Ausspielalgorithmen fr paketvermittelte Audiodaten (Evaluation of adaptive playout algorithms for packet audio). Studienarbeit, Dept. of Electrical Engineering, TU Berlin, Berlin, Germany, Oct. 1995.

[15] Sun Microsystems. *SunVideo User's Guide*. Sun Microsystems, Mountain View, California, Aug. 1994.

[16] T. Turletti. H.261 software codec for videoconferencing over the Internet. Rapports de Recherche 1834, Institut National de Recherche en Informatique et en Automatique (INRIA), Sophia-Antipolis, France, Jan. 1993.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource reservation protocol. In *Proceedings of the International Networking Conference (INET)*, pages BCB–1, San Francisco, California, Aug. 1993. Internet Society.