Lip Synchronisation for use over the Internet: Analysis and Implementation

Isidor Kouvelas, Vicky Hardman and Anna Watson

Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK

Abstract

This paper presents the first implementation of multicast inter-stream synchronisation over the Mbone / Internet. Variable bit-rate video, packet audio with silence supression and the unpredictable Mbone traffic characteristics provide a real test for the design. The paper also describes an efficient novel video tool architecture to provide intra-stream synchronisation, and implementation of inter-stream synchronisation using a local conference bus. Subjective performance results indicate that the efficient implementation is good enough to provide lip synchronisation for multimedia conferencing applications.

1 Introduction

The transmission of multimedia over packet networks and the use of independent systems for audio and video, means that listeners often experience a time lag between a remote user's audible words, and the associated lip movements. The more quality demanding applications of multimedia conferencing, such as remote language teaching, require both lip synchronisation and good quality audio in order to be effective. Lip synchronisation can be provided by artificially delaying either the audio or the video output, so that words and sounds are matched in time.

The Mbone is a multicast overlay over the Internet which provides multiway communication. Existing Mbone audio tools - such as vat [1], nevot [2] - and existing Mbone video tools - such as vic [3], ivs [4] - do not provide support for lip synchronisation. A new audio tool, RAT [5, 6], developed at UCL, together with a modified version of vic [3] provides the first implementation of lip synchronisation over the Mbone. RAT and vic use the proposed IETF standard real-time protocol, RTP [7], to provide the necessary functionality required for multicast voice and video communication. The lip synchronisation mechanism developed for RAT uses the time-stamp information from RTP to identify the required artificial time-lag.

This paper describes a lip synchronisation technique, which includes inter-system communication. Implementation efficiency considerations heavily influenced the design chosen, since the obvious method consumes far too much processing power to make the system viable. The subjective analysis provides results which show that the mechanism is good enough for all multimedia applications requiring lip synchronisation.

This work has been accomplished as part of project ReLaTe (Remote Language Teaching over SuperJANET, a BT/JISC funded project), project MERCI (Multimedia European Research Conferencing Integration, European Union Telematics Applications Programme #1007), and project RAT (Robust-Audio Tool, an EPSRC funded project #GR/K72780).

2 Background

Good quality packet audio systems must provide packet loss protection and low end-to-end delay [8]. The requirement of low delay means that a packet audio system must launch small packets frequently, rather than big packets less often [8]. Typical packet sizes used for audio over the Internet vary from 20 to 80ms in length.

Mbone video tools, such as vic [3], typically provide slow-scan video (2-10 frames/second), rather than the broadcast rate (25 frames/second), because of the potentially large amounts of bandwidth consumed by video, especially during multiway communication. The processing time associated with video encoding is also often great, which results in a large delay between video grabbing and rendering. Video frames are usually split into a number of packets for transmission.

The Mbone is a shared packet network [9]. Routers operate on a first-in first-out basis and statistically multiplex traffic from different sources. The impact of this behavior on real-time traffic is to introduce jitter to the inter-packet timing relationship, a second order effect. Jitter must be removed from audio packet streams, since it renders the speech unintelligible. A voice reconstruction buffer is used to artificially add delay to the audio stream to smooth out the jitter (figure 1). The mechanism is adaptive, since jitter on the Mbone can vary significantly [5].

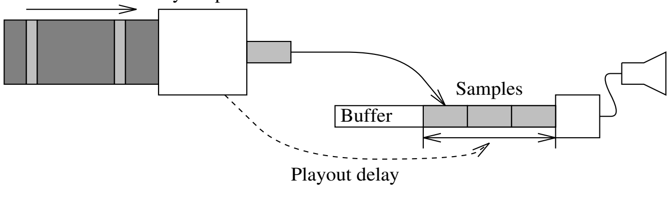

Figure 1: Playout delay adaptation to remove network jitter

The artificially added delay in the reconstruction buffer is calculated from the time that packets take to traverse the network [10, 11, 12]. A tradeoff exists between the amount of delay that is introduced in the buffer and the percentage of late packets that are received in time [10]. Consequently, audio tools aim to receive 99.9% of the packets, since the delay characteristic shows a long tail for packets that are abnormally delayed [2, 7].

The effect of jitter on video is to result in jerky rendering of frames, a degradation which can be tolerated by users. Current video tools, such as vic [3], nv [13], and ivs [4], display frames as soon as they are received and decoded.

3 Synchronisation of Audio and Video in a Packet Network

The difference between the end-to-end delay of audio and that of video often varies substantially, because of different processing and hardware manipulation times. Consequently, in order to provide lip synchronisation, one or other of the media streams must be delayed at the receiver. Subjective studies have shown that the two streams do not have to be exactly matched, but that a skew of 80-100ms is below the limit of human perception [14, 15, 16].

The provision of synchronisation delay for the audio stream can be achieved simply by increasing that provided for voice reconstruction (see above). In contrast, video tools do not usually have a reconstruction buffer. Lip synchronisation, however, requires that the time when a video frame is rendered must be known or be accurately estimable. This requirement means that a reconstruction buffer must be added to a video system.

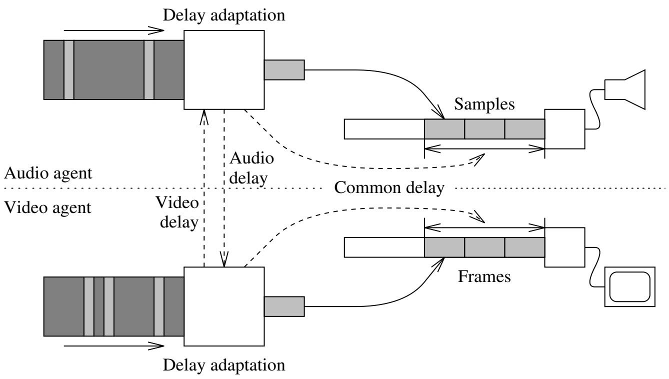

Figure 2: Playout co-ordination to achieve synchronisation

The end-to-end delay of video will be frequently longer than that incurred for audio, but this is not always the case. The end-to-end delay of audio must be tightly bounded - if interactivity is to be preserved - which means that negotiation between the systems must take place. Consequently, a means of inter-system communication must be devised, so that the relevant stream can be delayed to match the other (figure2).

Inter-media communication is needed in order to transfer the desired playout delays between the audio and video applications. Negotiation can be facilitated by using a local conference bus such as the Conference Control Channel Protocol developed at UCL [17] or the LBL conference bus [3]. Both these architectures employ multicast loopback facilities to provide communication. The mechanism restricts the traffic to the local host by setting the time-to-live parameter of the multicast packets to zero.

4 System Design

Lip synchronisation requires the following functionality to be implemented in addition to that normally provided by existing video and audio tools:

- Audio end-to-end delay measurement
- Video end-to-end delay measurement
- Video reconstruction buffering
- Negotiation facilities
- Desired playout point calculation

4.1 Audio End-to-End Delay Measurement

The end-to-end delay of audio on a per source basis can be easily measured, because most of the functionality is required for the playout point calculation. The audio device buffer (in the device driver) adds an extra delay component to this figure, but this is also known [18].

4.2 Video End-to-end Delay Measurement

The end-to-end delay in the video stream is the delay from the instant the camera scans a frame, to the time it appears on the remote screen. This delay is made up from the following components (figure 3):

- Time for the video hardware to grab a frame and it becoming available to the user application
- Time for the transmitter to process and transmit the frame (This includes the encoding and packetisation delays)
- Network propagation time
- Processing delay in receiving application (This includes decoding and frame rendering)
- Time between the receiver outputting the frame and it actually appearing on the screen

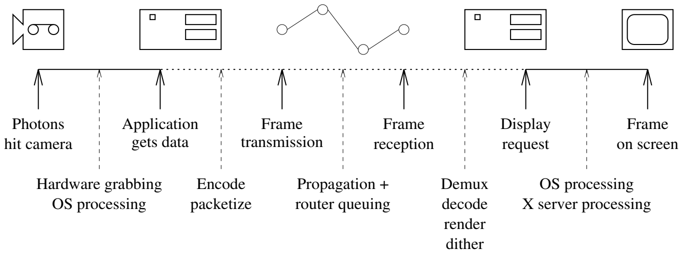

Figure 3: Video stream delays

Video frames are tagged using media specific time-stamps as they are received by the user application from the video encoding hardware. Since the video system is a user process in a multi-tasking operating system, it is not possible to get an accurate measure for delays occurring outside the process (see section 4.5.2). The unknown delays are those incurred before the frame data is available to the transmitter, and those incurred after the frame is rendered.

The delay incurred before frames are available in the transmitter can be dealt with by realising that frames are scanned by the camera at regular intervals. A low pass filter can be used to smooth out the effect of any workstation scheduling jitter on the time stamping process. The delay from the time a frame is output from the user application to the time when it is displayed on the screen is more of a problem. This delay is assumed to be constant, which is true for light workstation loading, but less correct for heavier loading.

The delay associated with the transmission of the frame across the network and the extra delay needed to cope with network jitter is known if a video reconstruction buffer is used in a similar way to that recommended for audio, RTP [7]. The time to decode and render a frame is likely to be variable, but can easily be allowed for if the buffering to smooth network jitter is accomplished after decoding. The artificially added reconstruction delay in the buffer can be extended to include the time taken by decoding and rendering.

4.3 Video Reconstruction Buffering

A video reconstruction buffer is needed to smooth out the effects of network jitter, and to ensure that video frames are displayed at regular predictable intervals. The video buffer can be designed in a similar way to that usually implemented in audio tools.

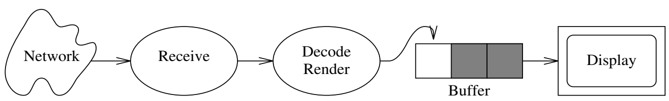

Figure 4: Video tool receiver with playout buffer before display stage.

After decoding and rendering, frames can be placed in a playout buffer and displayed at a pre-calculated playout point. Figure 4 shows the sequence of operations in this arrangement. Unfortunately due to the way in which H.261 [19] video frames are decoded, the design involves making two extra copies of a rendered video frame, which is a very significant overhead in terms of processing power.

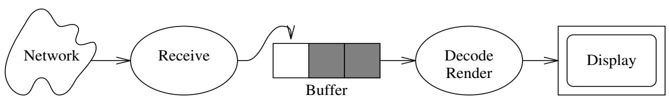

Figure 5: Video tool receiver with playout buffer before decoding stage.

In order to reduce the amount of processing time required, a novel architecture was used, similar to that developed for RAT [5]. The design positions the reconstruction buffer before video decoding has taken place, which eliminates the need for the two extra copies of each video frame. The configuration can be seen in figure 5. The design assumes that the decoding and rendering of frames varies only slowly with respect to time. Most video encoding algorithms encode the differences between video frames [20, 21], which means that frames from a period containing significant change in the image will take longer to encode and decode. The time estimated for decoding and rendering is added to the playout point.

4.4 Media Negotiation Facilities

RTP headers are used for the transmission of audio and video packets across the Mbone (figure 6). The header contains a time-stamp that is used to smooth out the effects of network jitter. The time-stamp is media specific and different reference clocks are used for audio and video. Audio uses its own device interface as a clock, since it is available [5]. The format of the time-stamps used for the video stream depends on the type of compression used. The time-stamp used with H.261 encoding [20] is based on a 90kHz clock [19].

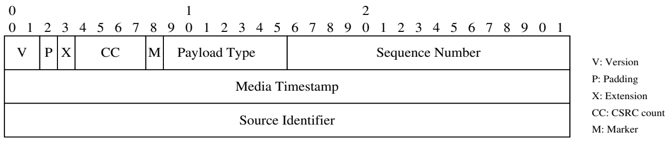

Figure 6: Real-time transport protocol header

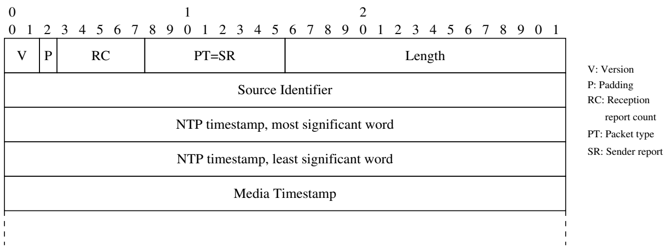

Figure 7: RTP control protocol (RTCP) packet

Playout delay negotiation between the audio and video processes needs to be in a common format, and with a common reference clock. RTCP control messages (figure 7 are transmitted alongside each data stream, and provide individual mapping between the media and Network Time Protocol (NTP) time-stamps [7, 22]. The NTP time-stamps therefore give a common reference to the transmitter workstation clock, which can be used to provide inter-stream synchronisation. Figure 8 shows the exchange of RTP and RTCP packets across the network and synchronisation information over a local conference bus, between multiple conference participants.

As explained earlier, the audio and video tools announce their required playout delay over the conference bus. Delay coordination is achieved by arranging for both tools to enforce the larger of the two delay requests. An advantage of this model is that it can be easily extended to include other media. For example, a shared

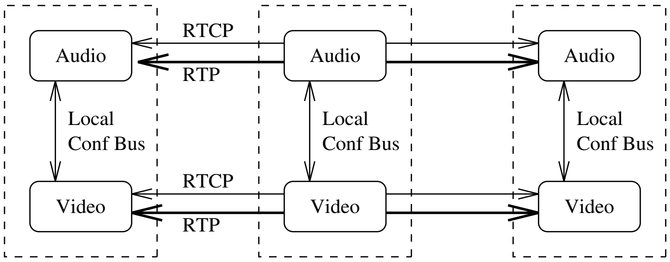

Figure 8: Media Agent Communication

white-board tool could participate in the protocol by announcing its own required playout delay without existing audio and video tools having to be modified.

The selection of the largest value of delay between the tools ensures lip synchronisation, but this may compromise other quality of service factors such as the low delay bounds required for interactivity. An enhanced scheme can be envisaged, where each application announces an acceptable range of delays to a coordinating agent, which then selects the delay to be enforced (figure 9).

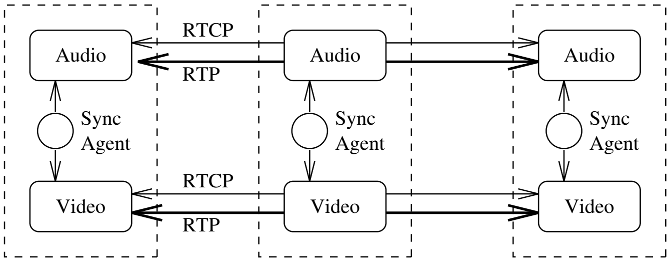

Figure 9: Media Agent Communication with Negotiation Agent

4.5 Further issues

There are a number of other considerations that must be taken into account when implementing a lip synchronisation mechanism: clock drift, and lack of real-time support on general purpose operating systems.

4.5.1 Clock Drift

The playout adaption algorithms allow for clock drift by adjusting the length of silence periods for audio, and interframe times for video. The internal timing facilities in audio and video are provided by different reference points; audio is clocked from the audio device interface, and video from the workstation clock. Inter-system negotiation is accomplished using RTP time-stamps. Clock drift occurs for both reference points, but can be allowed for, since both audio and video media timers are mapped via RTCP messages to the transmitter workstation clock.

4.5.2 Lack of Real-Time Support

The heterogeneous range of platforms that are connected to the Internet and used for multimedia conferencing are usually general purpose computing facilities, such as UNIX workstations or Windows 95 PCs. Such traditional time-sharing operating systems do not provide adequate support for real-time applications, such as audio and video [23]. Several attempts have been made to modify the operating system schedulers by increasing the number of preemption points in order to provide bounded dispatch latency for applications [24, 25, 26, 27]. However, as dynamic real-time scheduling is NP hard [28], and these improvements are not available on most Internet hosts, the UNIX round-robin scheduling approach with limited preemption is likely to persist.

Without real-time support, event timings on the receiving workstation become less accurate as the load on the receiving workstation increases [29]. The lack of real-time support in a host platform therefore can be expected to have a profound impact on the success of a lip synchronisation mechanism.

5 Implementation Assessment Results

The main results in this paper assess the subjective performance of the lip synchronisation implementation. However, the implementation relies on the assumption that decoding and rendering times of video frames vary only slowly, therefore results supporting this assumption are presented first.

5.1 Video Decode and Render Times

The following results show that it is reasonable to assume that video decode and render times vary only slowly. The first graph (figure 10) shows the combined decode and render times for a video sequence that contained movement throughout. This example was chosen as the worst-case scenario, since current video coding algorithms encode the changes between one frame and the next.

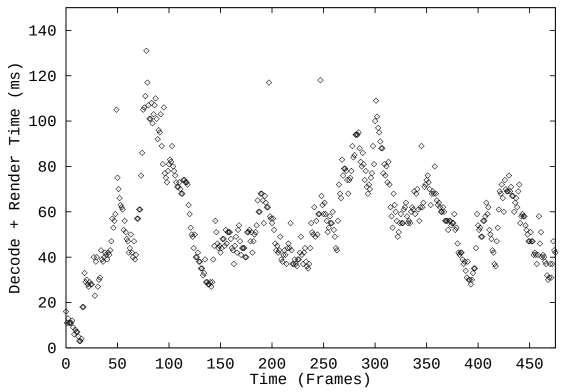

Figure 10: Decode and render times for H261 video frames.

The decode and render times over 500 frames were collected using vic [3] on a lightly loaded SUN Sparc 10 workstation, where H.261 was used as the video encoding algorithm, and CIF sized frames were transmitted. The frames extend over a period of 50 seconds, and contain movement over the whole period. It can be seen that overall the decode and render times vary over a range of about 100 milliseconds, but between frames by a much smaller amount.

Figure 11 shows the absolute values of the differences between the decode and render times for consecutive frames. It can be seen that the differences extend over 20 milliseconds for the vast majority of frames. This result enables the system to predict how long it will take to decode and render a frame to within a resolution of 20ms. The information can be used to correctly schedule the presentation time of each frame. It is expected that the 20ms jitter associated with the frame presentation time will produce skew that is still well within the limits of human perception [14, 15, 16].

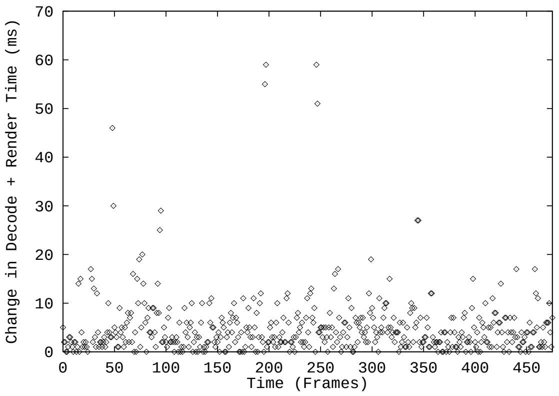

Figure 11: Change in decode and dither times between frames.

5.2 Subjective Performance Results

The subjective performance results show perceived level of synchronisation for different frame rates, and for both synchronised and unsynchronised audio and video. The material consisted of audio and video transmission of a speaker counting. Eight subjects assessed this material at rates of 2, 5, 6, 8 and 12 frames per second. The performance was measured using a 5 point rating scale, where 5 is rated as synchronised, and 1 as unsynchronised. The experiments were performed on a pair of Sun SPARC 10 workstations, using H.261 video coding, and CIF sized frames.

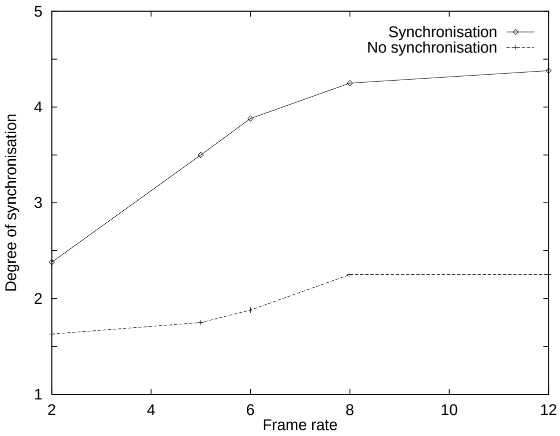

Figure 12: Subjective assessment of synchronisation

The results can be seen in figure 12, and indicate that audio and video is not perceived as being synchronised for frame rates of less than 5 frames per second. The results also support the earlier conclusion that the jitter associated with video frame presentation times is below the limit of human perception.

6 Conclusion

This paper has presented results from the first implementation of lip synchronisation between audio and video over the Mbone. The technique uses RTP time-stamps to provide intra-stream synchronisation for video, and inter-stream synchronisation between media. A novel efficient architecture for video reconstruction was also implemented to remove network jitter in frame presentation. Media negotiation facilities were provided by the use of a local conference bus.

Results were collected to measure the subjective performance of the lip synchronisation implementation, and to test the validity of an assumption that was made in the design of the system. The results show that the efficient implementation is good enough to provide lip synchronisation for use over the Mbone.

7 Further Work

We intend to collect further subjective performance results using forced comparison measurements, in order to identify the limit of human perception in lip synchronisation, estimated as being in the region of 80-100ms. We also intend to assess the impact of workstation loading on the performance of this system, which initial results indicate will be significant. The synchronisation agent mentioned in section 4.4 will be added to this implementation of lip synchronisation.

8 Acknowledgements

We would like to thank Jon Crowcroft, Angela Sasse and Mark Handley at UCL for their valuable input. We would also like to thank Steve McCanne (University of California, Berkeley) for providing us with the source code for vic and for fruitful discussions via e-mail. Lastly a big thank you to all our colleagues at UCL for being subjected to a number of perception experiments!

References

[1] Van Jacobson and Steve McCanne. The LBL audio tool vat. Available from ftp://ftp.ee.lbl.gov/conferencing/vat/, July 1992.

[2] Henning Schulzrinne. *Guide to NeVoT*. GMD Fokus, Berlin, Germany, 3.32 edition, September 1995. The software is available from ftp://gaia.cs.umass.edu/pub/hgschulz/nevot.

[3] Steve McCanne and Van Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, November 1995.

[4] Thierry Turletti. The INRIA Videoconferencing System (IVS). *ConneXions - The Interoperability Report*, 8(10):20–24, October 1994.

[5] Vicky Hardman, Isidor Kouvelas, Martina Angela Sasse, and Anna Watson. Robust-audio over the internet: Analysis and implementation. Research Note RN/96/8, Dept. of Computer Science, University College London, February 1996.

[6] Vicky Hardman and Isidor Kouvelas. RAT general architecture. Internal Note IN/96/2, Dept. of Computer Science, University College London, February 1996.

[7] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. A transport protocol for real-time applications. Request for comments RFC 1889, Internet Engineering Task Force, January 1996.

[8] Vicky Hardman, Martina Angela Sasse, Mark Handley, and Anna Watson. Reliable audio for use over the internet. In *International Networking Conference (INET)*, 1995.

[9] Stephen Casner. Are you on the MBone? *IEEE MultiMedia, Summer 94*, pages 76–79, 94.

[10] Warren A. Montgomery. Techniques for packet voice synchronization. *IEEE Journal on Selected Areas in Communications*, SAC-1(6):1022–1028, December 1983.

[11] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994.

[12] Van Jacobson. Multimedia conferencing on the internet, August 1994. SIGCOMM '94 Tutorial.

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

[14] Paul W. Jardetzky, Cormac J. Sreenan, and Roger M. Needham. Storage and synchronization for distributed continuous media. *Mulimedia Systems*, 3(3):151–161, September 1995.

[15] J. Escobar, D. Deutsch, and C. Partridge. A multi-service flow synchronisation protocol. *BBN STC Tech Report*, March 1991.

[16] L. Lamont, Lian Li, and N.D. Georganas. Synchronization of multimedia data for a multimedia news-on-demand application. *IEEE Journal on Selected Areas in Communications*, January 1996.

[17] Mark Handley, Ian Wakeman, and Jon Crowcroft. The conference control protocol (CCCP): a scalable base for building conference control applications. In *SIGCOMM*, pages 275–287, Cambridge, Massachusetts, September 1995.

[18] Isidor Kouvelas and Vicky Hardman. Overcoming workstation scheduling problems in a real-time audio tool. In *Usenix Annual Technical Conference*, Anaheim, California, January 1997.

[19] Thierry Turletti and Christian Huitema. RTP payload format for H.261 video streams. Internet draft (work-in-progress) *draft-ietf-avt-h261-01.txt*, July 1995.

[20] Video codec for audiovisual services at p x 64 kbit/s. ITU-T Recomendation H.261, International Telecommunication Union, 1993.

[21] ISO/IEC JTC1/SC2/WG11. MPEG. *ISO*, September 1990.

[22] David L. Mills. Network time protocol (version 3) specification, implementation and analysis. Request for comments RFC 1305, Network Working Group, March 1992.

[23] Newton Faller. Measuring the latency time of real-time Unix-like operating systems. Technical Report TR-92-037, International Computer Science Institute, Berkeley, California, June 1992.

[24] Olof Hagsand and Peter Sjodin. Workstation support for real-time multimedia communication. In *Usenix Winter Technical Conference*, San Francisco, California, January 1994.

[25] Sandeep Khanna, Michael Sebrée, and John Zolnowsky. Realtime scheduling in sunos 5.0. In *Usenix Winter Technical Conference*, 1992.

[26] Sape J. Mullender, Ian M. Leslie, and Derek McAuley. Operating-system support for distributed multimedia. In *Usenix Summer Technical Conference*, Boston, Massachusetts, June 1994.

[27] Tom Fisher. Real-time scheduling support in Ultrix-4.2 for multimedia communications. In *Third International Workshop on network and operating system support for digital audio and video*, pages 282–288, San Diego, California, November 1992. IEEE Communications Society.

[28] John A. Stankovic, Marco Spurl, Marco Di Natale, and Giorgio C. Butiazzo. Implications of classical scheduling results for real-time systems. *IEEE Computer*, pages 16–25, June 1995.

[29] K. Fall, J. Pasquale, and S. McCanne. Workstation video playback performance with competitive process load. In *International Workshop on Network and Operating System Support for for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, pages 179–182, Durham, New Hampshire, April 1995. Springer.