A Combined Network, System and User Based Approach to Improving the Quality of Multicast Audio

Isidor Kouvelas

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London.

Department of Computer Science
University College London

May 1998

Abstract

In real-time interactive Internet multimedia conferencing, audio quality can be impaired by packet loss resulting from network congestion, lack of real-time process support in end-host operating systems, and acoustic problems. Furthermore, when audio is combined with other media, such as video, the lack of coordination in the presentation between the individual media streams is problematic.

The thesis is that the quality of Internet multicast audio conferencing can be improved by a range of application techniques that focus on a combination of the system, the network and the user.

A new architecture for a multicast conferencing audio-tool is presented and implemented in the Robust-Audio Tool (RAT). This software is used as a platform for most of the research in this thesis.

Within RAT, a novel forward error correction technique is developed which successfully addresses the network packet loss problem with minimal network load and delay overheads.

To address the operating system scheduling problems, a new adaptive scheme that analyses the real-time performance of the workstation and works around anomalies is developed and evaluated.

The feasibility of synchronised presentation of audio and video in a multimedia conference to solve the lip-synchronisation problem is investigated. To this end, the Mbone video conferencing program vic is modified to work together with RAT in stream presentation. The success of the approach is shown through a series of human subjective perception experiments.

Finally, the problem of audio delivery to participants of multicast conferences with heterogeneous network connectivity is addressed. The proposed protocol is run between the audio applications in end hosts that cooperate to work around congested network links and requires no modification to the network infrastructure. The effectiveness of the approach is shown through simulation.

Acknowledgements

I would first like to thank my parents and Maria for their support throughout my studies at UCL. Without their constant encouragement this dissertation might have never been completed.

My supervisors Jon Crowcroft and Vicky Hardman for providing excellent input and feedback with the development of the presented ideas and for a lot of constructive criticism on early versions of this thesis.

Peter Kirstein and Angela Sasse for providing a great environment to work in and for funding work related to my PhD as well as endless conference travel.

My Bayswater friends Demetrios Clerides, Nicolas Carabateas, Angela Tsouma, Konstantine Moutoussis and all the 'Georges' for making life in London fun.

My flatmates Anna Moutoussi, Nikos Doukas, Sotiris Doukas and of course my brother Antony for endless arguments! :-)

Finally a big thank you to all the people I worked with at UCL for contributing through many different ways, from moral support to technical discussions on countless pub nights. Most especially to Natasha Vassila, Julia Schnabel, Anna Watson, Lorenzo Vicisano, Mark Handley, Orion Hodson, Colin Perkins, Panos Gevros, Yannis Koufakis and Thanassis Tiropanis for being good friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent past, the Internet used to be perceived as a computer network used by researchers to transfer files and send text messages. It was believed that the best effort service model and lack of state in network nodes were unable to meet the real-time delivery constraints of multimedia conferencing traffic.

The origins of speech transmission over long-haul packet networks stem from the ARPANET and SATNET experiments, which took place in the mid 70s and spawned packet-based audio conferencing research on both sides of the Atlantic. In the last few years, there has been newfound interest in Internet based conferencing. Applications like Internet telephony have seen increasing popularity and now form a significant portion of Internet backbone traffic. This has mainly been the result of the following developments:

- There has been a significant increase in available bandwidth over Internet backbone links which can now support high volume multimedia traffic.
- The provision of multimedia peripherals as standard on commonly used networked workstations and PCs allows a low-cost approach to desk-top multimedia conferencing and has enabled a critical mass of users to be established quickly.
- The deployment of Internet Protocol (IP) multicast on the Internet over the Internet Multicast Backbone (Mbone) has provided an efficient multi-way communication platform for conferencing applications.

The Mbone is an overlay network over some high-bandwidth areas of the Internet which supports the delivery of IP multicast traffic. IP multicast offers optimal bandwidth utilisation in multi-way data distribution by forcing the network to do data replication only when necessary. This makes dissemination of high-bandwidth continuous media to a large number of recipients feasible thus enabling multi-way multimedia conferencing applications. In countries like the US and the UK, the Mbone has moved from pilot to service, and is used by a number of research and teaching collaborations.

In contrast with early packet audio work in the 70s, which focused on the provision of two-way communication, IP multicast over the Mbone has enabled new forms of Internet group communication. These range from lectures, where a multimedia presentation originating at a single site is disseminated to an audience of more than one receivers, to multi-way conferencing, where each participant in the session acts as a source and sink of information. Research at LBL, Xerox PARC, MIT, ISI and UCL in multi-way multimedia conferencing over the Mbone, has produced the lightweight session model [Jacobson, 1994; Handley, 1997c]. This model does not suffer from the scaling, reliability and complex session set-up and maintenance problems of connection-oriented conferencing approaches. It uses the network for multicast data distribution, accepting the best-effort delivery model, and relies on adaptive receiving applications to recover from adverse network effects. No explicit conference control is provided, and instead periodic session messages are used to report on membership and other session parameters. Additional mechanisms are thus required to support tightly controlled conferencing for small groups[1]. The lightweight nature of the model enables it to scale to very large sessions, where exact knowledge of session membership is not important.

Communication of individual media, such as audio and video, is provided by separate media specific applications run at each participating end-host. The dedicated media tools are implemented with the principles of application level framing (ALF) [Clark and Tennenhouse, 1990], thus providing efficient processing of the high-bandwidth multimedia data. Co-ordination between the different tools in each end-host takes place over a local conference bus. Conference policy is enforced by a controlling application that communicates commands to each media tool over the local bus. Different conferencing scenarios, from small tightly controlled conferences to large scale lectures, can thus be implemented by replacing the co-ordinating application while using the same specialised media tools.

[1] By tightly controlled we mean provision of membership control and floor control mechanisms.

Today, more users are becoming aware of the potential of the Internet as a general communication network. Its ability to support multimedia conferencing is being continuously demonstrated by a series of multicast events. One of the first such events was the Internet Engineering Task Force meeting on March 1992 [Casner and Deering, 1992]. Of the media used in multimedia conferences, audio is the most important, as speech is the natural mechanism of human communication, supplemented by video and shared workspace. Evidence from formal experiments, as well as observations of many sessions under real conditions, suggest that audio of sufficient quality is essential for successful multimedia conferencing. Multicast conferencing over the Mbone has the potential to offer low-cost real-time multimedia solutions to a wide range of user groups, provided that good audio quality can be sustained.

1.1 Contributions and Overview of this Dissertation

The use of early Internet audio tools in piloting activities at UCL, helped to identify a number of key problems which affect the perception of Mbone audio. The manifestations of problems to users can be categorised as follows:

Shared network effects. The Internet's service model offers best-effort transmission, and is unable to provide the quality of service (QoS) guarantees needed for real-time traffic. Increased queueing delay at network routers and packet loss occur when network links are in a congested state[2]. These produce speech clipping effects (gaps in audio) which result in serious quality degradation.

Poor end-system support. The lack of support for real-time multimedia applications in general purpose workstation operating systems, results in variable delays in audio presentation and losses of speech segments which are comparable to the effects of network congestion.

Acoustic problems. Audio transmitted over the Mbone uses one channel at toll quality to limit the bandwidth requirements on the network. This restricts the intelligibility of speech and makes speaker identification more difficult. In addition, poor

[2] The network and end-host environment shortcomings and their effects on packet audio conferencing are discussed in more detail in Chapter 2.

silence suppression, echoes and no distance cues, resulting from inadequate processing of the audio signal from the audio application and presentation hardware, contribute to making audio quality less natural.

Although the acoustic problems listed above are significant, the effects of clipping and variable round-trip delays on audio, due to the lack of support from the network and end-system environments, are perceptually the most important problems [Watson and Sasse, 1996]. Another significant shortcoming, which is also the result of network delay and host operating system deficiencies, is the lack of synchronisation in the presentation of different media.

The thesis is that the quality of Internet multicast audio conferencing can be improved by a range of application techniques that focus on a combination of the system, the network and the user. The real-time media applications, running in the end-hosts, can be designed to adapt around environment problems and smooth out adverse effects. Studying network characteristics, exploiting knowledge about human perception, and applying computational techniques to media tool design, can lead to considerable improvement to the perceived quality of Mbone audio.

An alternative solution to these problems is to modify the network and end-host architectures to provide the required quality of service. However, such an approach would sacrifice the flexibility of the current infrastructure that is critical to its scalability and robustness.

This dissertation provides several original contributions to Internet multimedia conferencing that address the challenges presented from the general purpose communication and end-host environment. The proposed techniques, which are introduced below, can be implemented at the application level. The solutions maximise the user perceived quality of the presented multimedia streams, for a given network and end-host performance, and attempt to fairly share available resources.

Packet loss robust audio tool. The effect of network packet loss on audio quality is currently the biggest obstacle to the realisation of the full potential of multimedia conferencing over the Mbone. In Chapter 3 we introduce a new robust-audio tool (RAT), which incorporates a redundancy mechanism to counteract the effects of packet loss. An introduction to the structure of the audio tool and loss repair mechanisms is made and the novel redundancy mechanism is presented and evaluated. Results from human perception experiments and real network measurements are used to show the potential and limitations of the audio transmission quality improvements possible with RAT. Parts of these results have been published in [Hardman *et al.*, 1998; Kouvelas *et al.*, 1997; Hardman *et al.*, 1996].

Workstation scheduling adaptation. Audio quality is affected when the processing capacity of a transmitting or receiving workstation is exceeded, resulting in speech quality degradation similar to that caused by packet loss. This is caused by the lack of support for real-time applications in today's general purpose workstation operating systems. In Chapter 4 we present an architecture for a real-time audio media agent, that copes with these effects at the application level. The mechanism produces a continuous audio signal, despite the variable allocation of processing time a real-time application is given, by trading off signal quality for buffering delay in a controlled fashion. The proposed solution, also described in [Kouvelas and Hardman, 1997], has been implemented in RAT and evaluated through simulation with success.

Lip synchronisation. Packet audio with silence suppression, variable bit-rate video and the unpredictable Mbone traffic characteristics produce different end-to-end delays in the transmission and presentation of different media streams. As a result, receivers often experience a time lag between hearing a remote user's words, and seeing the associated lip movements. Although in low frame rate applications, where the video is used only for presence information, this is not a problem, more demanding applications of multimedia conferencing, like language teaching, are adversely affected. In Chapter 5 we present an analysis of the requirements and implementation of the first multicast inter-stream synchronisation over the Mbone between RAT and the vic [McCanne and Jacobson, 1995] Mbone video tool. Implementation efficiency considerations heavily influenced the design chosen, since the obvious method consumes far too much processing power to make the system viable. Subjective performance results indicate that the efficient implementation is good enough to provide lip synchronisation for demanding multimedia conferencing applications. This work has also been published in

[Kouvelas *et al.*, 1996].

Scalable congestion control for multicast continuous media. The scale of the Mbone and heterogeneity in available bandwidth complicate the design of network adaptive multicast applications. The lack of transmission rate adaptation in such applications drives traditional TCP based traffic off the network and can result in network collapse due to congestion. In Chapter 6 we present a new scalable architecture for congestion controlled multicast real-time communication. The proposed scheme uses self-organisation to form groups out of co-located receivers with bad reception, and provides local repair through the use of transcoders. The receiver driven nature of the protocol ensures high scalability and applicability to large Mbone sessions. The viability of the proposed protocol is demonstrated through simulation. This work has also been published in [Kouvelas *et al.*, 1998].

Part of the work in this thesis was motivated by the requirements of networked multimedia research projects that were active at UCL during the period of study. In particular the Multimedia European Research Conferencing Integration (MERCI), Remote Language Teaching over SuperJANET (ReLaTe) and RAT projects provided significant support and feedback in the development of the RAT audio tool and promoted its success and widespread use in the Internet community. Other PhD work not presented in this dissertation that was supported by the above projects includes the implementation of the Conference Control Channel Protocol (CCCP) local conference-bus architecture [Handley *et al.*, 1995], the design and implementation of a unified language teaching multimedia conferencing user interface used in ReLaTe [Buckett *et al.*, 1995] and the porting of the vic video conferencing software [McCanne and Jacobson, 1995] under Microsoft Windows operating systems.

Chapter 2

Background

This chapter introduces some aspects of Mbone multimedia conferencing that are of direct relevance to the work presented in the following chapters. In particular the network and end-host environment used for communication is discussed.

2.1 Network Environment

The Internet Protocol (IP) used on the Internet, provides a clear separation of functionality between end-nodes and network routers [Carpenter, 1996]. End-nodes are responsible for managing end-to-end connections and controlling the amount of traffic they inject into the network, without any knowledge of the network topology. Network routers are responsible for managing the routing topology and forwarding packets according to the destination address, with little or no knowledge of end-to-end connections. This separation provides robustness and increases scalability, since the network can dynamically change the delivery path without the need of coordination between all affected nodes.

To optimise link utilisation, traffic from different sources is statistically multiplexed over shared network links. At network nodes buffering is used to accommodate bursts of incoming packets that need to be forwarded down the same link. The queueing of packets in network nodes introduces a variable delay in their end-to-end transit time. Figure 2.1 shows how this distorts the original timing relationship with which the packets were transmitted. In addition to the delay jitter resulting from the variable queueing time, when the buffer limits of a node are exceeded, incoming packets are dropped. This is the primary cause of packet loss on the Internet as bit-error rates on links are in the range of 10^{-15} to 10^{-3}.

Real-time multimedia conferencing communication has requirements for mini-

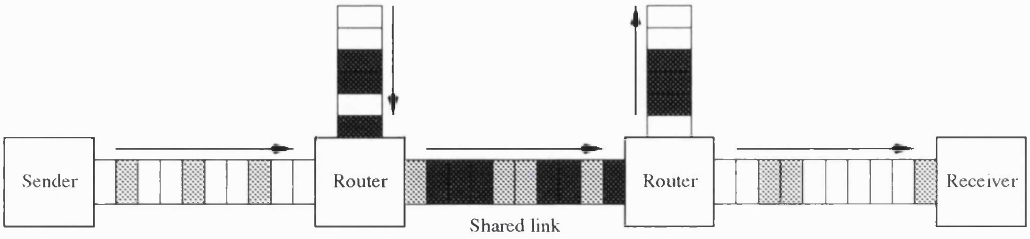


Figure 2.1: Effect of traffic multiplexing on real-time media stream.

mum throughput and constant network delay. Media applications transmit the encoded digitised audio or video signals in a series of packets across the network. In order to reproduce the original signal at the receivers, all the packets need to be received, and the timing relationship with which they were created needs to be maintained in presentation. The best-effort service that is currently available on the Internet can satisfy these requirements under unloaded network conditions, when the queueing of packets at network nodes is minimal and the end-to-end transit time consists of the link propagation delay. When the network is in a loaded state, no guarantees can be given for when or if a packet will reach its destination. As a result, media applications are designed to be adaptive and attempt to smooth out the effects of the network.

Looking at the problem from the network point of view, the newly introduced real-time multimedia flows do not obey the laws of existing traffic on the Internet. Traditional applications use the Transmission Control Protocol (TCP) for reliable delivery of data, which uses congestion control algorithms, in order to fairly share link bandwidth between multiple connections and to maintain the load on the network at useful levels [Jacobson, 1988; Stevens, 1997]. Real-time application packet streams do not respond to signals of congestion, as TCP does, and can lead to network collapse. Proposals exist for protecting the best-effort service, by deploying mechanisms in network nodes, to restrict the bandwidth of flows using a disproportionate share of resources [Floyd and Fall, 1998]. It is hoped that such policing will act as an incentive for application designers, to produce network-friendly applications that back-off in the face of congestion.

There is ongoing work within the IETF, looking at alternative solutions, to provide support for other service classes in an integrated services Internet. Specifications exist for a controlled-load service, which provides a quality of service closely approximating that of an unloaded network [Wroclawski, 1997], and a guaranteed delay and bandwidth service [Shenker *et al.*, 1997]. For applications to communicate to the network the required quality of service, the resource reservation protocol RSVP [Braden *et al.*, 1997] has been designed, and admission control policies are being considered. The mechanisms for implementing the new service classes are still under development. Although these services will eventually be available on the Internet, because of the pricing models that will apply it is expected that the best-effort class will still be used by a wide variety of applications including multimedia conferencing.

2.2 Internet Multicast

IP multicast provides scalable efficient multi-way data distribution over the Internet. Deployment of multicast capabilities on the Internet started in 1991 over the DARTnet testbed network. This test network was extended to distribute audio of the proceedings of the March 1992 IETF meeting to 20 sites worldwide [Casner and Deering, 1992]. An overlay network, called the Mbone, was set up using multicast routing daemons running in end-hosts. The daemons provided native multicast capabilities to hosts connected through local shared links. Multicast communication to remote sites was achieved through tunnels, using the Distance Vector Multicast Routing Protocol (DVMRP) implemented by Steve Deering. The Mbone has since expanded, to reach more than 4300 connected networks in July 1997, and is progressively becoming part of the operational infrastructure of the Internet. Examples include UKERNA, who has offered Mbone connectivity as part of its operational services over the JANET UK academic network since February 1996, and UUNET, one of the worlds largest Internet service providers who is currently providing the UUCast commercial multicast service.

A major factor in the success of the Mbone has been the simple service model offered by IP multicast to applications:

- Senders just send to the group using a multicast address.
- Receivers join the group by expressing an interest to the same multicast address.
- The network routers conspire to deliver the data to all interested receivers.

This model hides all the problems of multicast communication from applications, which use the abstraction of an IP class-D multicast address to identify the group. The group membership, distribution topology and scaling issues are left up to the network to tackle. As with unicast communication, by hiding the network routing from end-systems, topology changes affect only nodes local to a problem increasing scalability. Replication of data takes place in the network, instead of end-systems, and bandwidth efficiency is thus achieved by making data traverse each network link at most once.

The model is analogous to that of CB radio, where a sender transmits at a given frequency without knowledge of the receiver group membership, and receivers tune in to the same frequency to access the transmission. The difference with multicast is that every member of the group can transmit data, which is demultiplexed at receivers using the IP address of the sender.

Scalable multicast routing is still an active research topic with several existing proposals. These can be classified in two main categories: source-based tree and shared tree schemes.

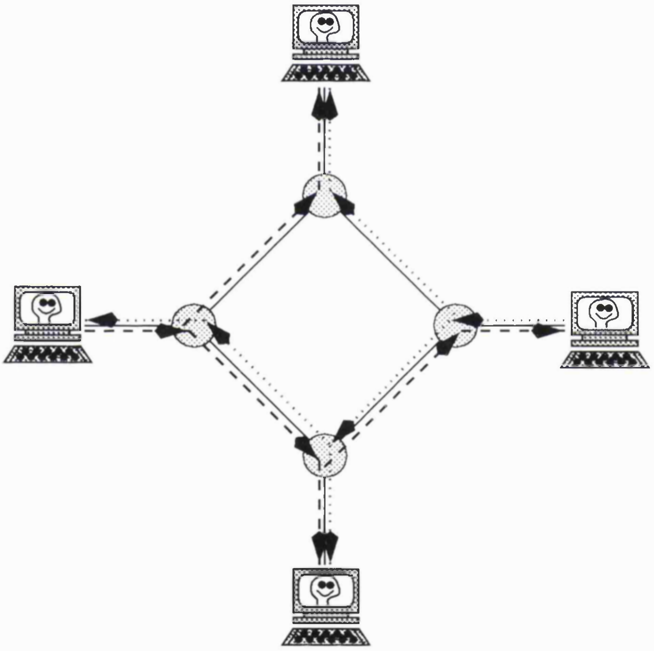


Figure 2.2: Source routed multicast trees for different senders.

Source-based tree techniques use a separate multicast delivery tree rooted at each sender of a multicast group (Figure 2.2). The tree for each sender is built using the shortest paths to receivers thus optimising delivery efficiency. Good network utilisation is achieved, as packets from different sources may use different links. A drawback is that every participating router must maintain information for each multicast group and source pair, providing poor scalability. Further, in order to build a delivery tree, multicast traffic is periodically flooded to the entire network. One such protocol is DVMRP [Deering *et al.*, 1988], based on the reverse path multicasting (RPM) algorithm, and is the main protocol in use currently on the Mbone, mainly because of its implementation simplicity. Other candidates include dense-mode PIM and MOSFP [Moy, 1994]. Source-based tree protocols are designed to work well in environments with high bandwidth and densely distributed receivers.

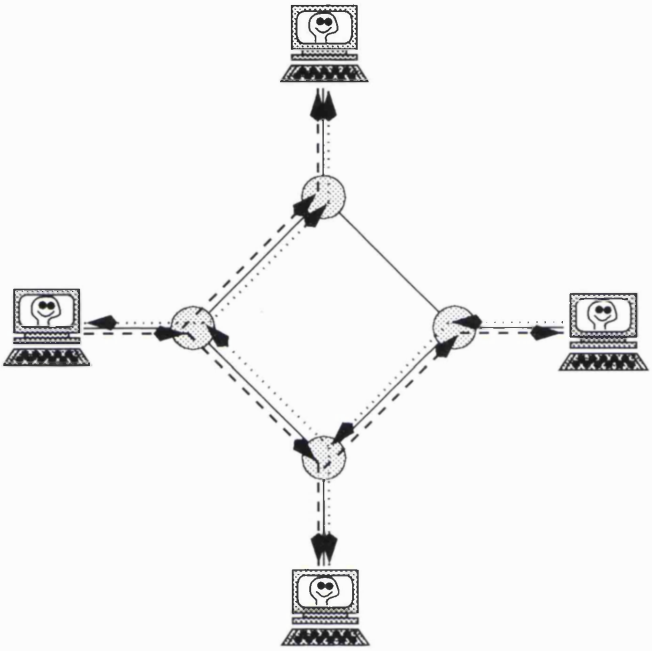


Figure 2.3: Shared tree multicast routing for different senders.

Shared tree techniques use the same delivery tree for multicast traffic from all the sources in a group (Figure 2.3). This results in more efficient use on router resources, since only routers involved in the distribution tree need to maintain group state information, and specific source information is not required. The disadvantage is that the trees built are sub-optimal, as they do not represent the shortest paths from the senders to receivers. Further, as the traffic from all senders shares the same paths, traffic congestion is likely to occur on bottlenecks near core routers. CBT [Ballardie, 1997] is the main example of a shared tree protocol. Sparse-mode PIM [Deering, 1997] is a hybrid scheme, that initially builds a shared tree, and if there is sufficient traffic from a sender, it switches to a shortest path source-based tree. Shared tree protocols are designed to work well in wide-area environments with sparsely distributed receivers.

The Internet group management protocol (IGMP) [Deering, 1991; Deering, 1989] is used to report end-host group membership to neighbouring multicast routers. A host joins a group by periodically announcing membership over a LAN to the local router. Leaving the group is achieved by the host not refreshing the membership and the router timing out the state. Since its first implementation, IGMP has been revised to Version 2 in order to reduce the group leaving delay. Version 3 is currently under development, which will allow hosts to express interest in specific senders for each multicast address joined, provided the multicast routing protocol supports it.

IP multicast provides a flexible platform on which multi-point communication applications can be built. The most common application on the Mbone currently is video-conferencing, using audio, video and shared-workspace tools. The typical session scenarios are dissemination of lectures and small scale interactive conferencing. In a lecture setup there is a single sender and multiple passive receivers and thus transmission bandwidth is low and near constant. In this case, the use of a shared tree routing protocol with the tree routed near the sender can be advantageous. The inefficiency when a listener needs to transmit to ask a question is acceptable as it will be a rare event.

In interactive conferencing a large subset of participants need to transmit data to participate. If all members were to continuously transmit audio and video, the total amount of bandwidth required on core network links would be great. In practice, the bandwidth used is limited by making sure that only active participants transmit. Silence suppression techniques are employed by the audio media applications to detect activity and audio activated video switching to control video transmission. This typically limits the sources to one or two at any time. In such a setup, a source routed protocol would create a different shortest-path tree for each participant, providing the lowest possible transmission delay and thus improving interactivity.

Clearly, the best-effort service model provided by the Internet, and as a result by the Mbone, is not ideal for the transmission of real-time traffic, but the advantages offered by the multicast communication facilities and link bandwidth sharing outweigh the problems. The alternative to using a packet switched network like the Mbone for group communication, is the use of virtual circuits which provide guaranteed bandwidth and fixed end-to-end delay and thus, the real-time requirements of audio and video traffic are satisfied. Virtual circuits have to be individually set-up, and in order for any member in a multimedia conference to be able to contribute, a circuit has to exist between every pair of participants. The management of the number of circuits needed to support full connectivity between all conferees rapidly becomes an impossible task as the size of the conference increases. Further, the system is inflexible, as if a new user wants to join a conference, complete knowledge of the conference membership is required, and circuits to all existing participants have to be established.

Despite the receiver join model of Internet multicast, membership control in Mbone conferencing can be achieved by encrypting media packets using keys that are distributed out-of-band to the intended participants. This way, although eavesdroppers can receive the packet stream, they cannot use it. In addition to membership control, authentication of contributions to the conference can be performed by the use of asymmetric public-key cryptography.

2.3 Real-Time Transport Protocol

The real-time transport protocol (RTP) provides end-to-end delivery services for real-time media streams in multi-participant multimedia conferences. RTP was designed within the IETF to support the lightweight session conferencing model. Following the principles of application level framing and integrated layer processing [Clark and Tennenhouse, 1990], RTP is implemented as part of the media application and not as a separate layer in the operating system. A full description of RTP can be found in [Schulzrinne *et al.*, 1996].

In multicast multimedia conferences, a separate RTP session is used for each of the separate media streams like audio and video. The functionality of the protocol is provided by a header that is introduced to media data packets. Header fields include a session-wide unique sender identifier for de-multiplexing between multiple senders. A payload identifier is used to describe the data in the body of the packet. Assigned values exist for specific audio and video payload types and are described in [Schulzrinne, 1996].

RTP does not provide any quality of service guarantees and relies on lower layers to do so. It is typically used over the IP User Datagram Protocol (UDP). The UDP header provides a port identifier for demultiplexing multiple streams originating at the same host. With the current Internet model, UDP over IP provides no mechanisms to ensure timely delivery of data. The task of reconstructing the timing relationship in the transmitted media stream is left up to the receivers. To assist in the intra-media synchronisation, RTP provides sequence numbers and media specific timestamps as well as a synchronisation marker.

RTP incorporates a simple control protocol (RTCP), which operates on the same multicast address and next UDP port to RTP. RTCP control messages are sent periodically by session members and provide participant identification and reception quality information. In order for RTP to be able to scale to sessions with large number of participants, the rate of transmission of RTCP messages is controlled, so that the total reporting bandwidth used by all receivers is maintained at a constant fraction of the data bandwidth.

2.4 Multimedia Conferencing Support in End-Hosts

Modern workstations and PCs are equipped with audio recording and playback and video capture devices.

A typical audio device provides microphone and line-in inputs and output to a headset, line-out or loudspeakers. During an audio conference it is important to prevent the output audio of the remote participants from feeding back into the microphone thus creating an echo. The problem is trivially solved by the use of headphones. In case more than one participants want to share a workstation, audio has to be output through loudspeakers and echo-cancelling hardware has to be employed. Analogue to digital conversion of signals is performed on the input and the reverse process for output. The digital formats offered by audio devices vary in the sampling rate of the signal, the resolution of the samples and the number of channels (mono or stereo). A range of qualities is offered, from 8 kHz single channel μ-law sampling for conferencing applications, to 44.1 kHz stereo linear 16 sampling for CD quality music applications.

Workstation video capture devices convert the video signal from conventional video cameras, to a series of bitmaps corresponding to video frames. The European and US PAL, SECAM and NTSC video formats are usually supported. The European formats provide 25 fps (frames per second) with a resolution of 768 by 576 pixels, whereas the NTSC format provides 30 fps at a resolution of 640 by 480 pixels. The amount of data produced by the fully sampled signal is significantly larger than that for audio. Current videoconferencing applications use a subset of this information due to network bandwidth limitations and workstation processing power capabilities. Typically, a quarter of the resolution of each frame is used, and not all frames are processed.

The application programming interface (API) to access multimedia devices is often incapable of supporting conferencing. The most common reason for this shortcoming is that the API was designed to support a different class of applications. Prime examples are the WAV audio and VIDCAP video capture APIs in Microsoft operating systems. The available device drivers for these systems evolved out of capture applications that recorded multimedia data to disk. As a result, it is almost impossible to capture video data without storing it to disk, and the interactive recording and playback of audio requires the complicated management of a large number of buffers, as buffering capabilities are not offered by the device[1]. Microsoft has responded to the demands of the new real-time applications, and new audio and video capture interfaces are being designed and made available. The available UNIX operating system variants offer significantly more flexible programming interfaces with similar characteristics and can better support conferencing applications. Extensions however are still required to accommodate some of the real-time performance requirements, especially those of audio [Rizzo, 1997].

The general purpose multitasking operating systems found in most networked workstations provide none or very little support for applications with real-time constraints. Available schedulers provide no quality of service (QoS) guarantees for the CPU time and dispatch latency received by an application. Attempts to modify popular operating systems to provide better service to applications requiring predictable performance have been made with promising results [Hagsand and Sjodin, 1994; Khanna *et al.*, 1992; Mullender *et al.*, 1994; Fisher, 1992]. These however still do not provide absolute guarantees, and more importantly are slow in being deployed and consequently, not currently available on the average workstation. The Pegasus project in their Nemesis operating system [Leslie *et al.*, 1997] adopted a different approach, by accepting that currently available systems were not designed with real-time applications in mind, and

[1] An analysis of general shortcomings of the APIs in Microsoft operating systems can be found in [Spinellis, 1998].

started from scratch with a new system architecture. Nemesis provides fine-grained control to the application of all system resources, including the CPU. While absolute deadlines cannot be satisfied, a low bound on dispatch latency and a share of CPU time are guaranteed.

Fortunately, the timing requirements of multimedia applications, although much stricter than those of traditional applications, are still quite soft. Although there is a deadline associated with events beyond which the quality of the presented media deteriorates, some room for inaccuracy exists, which is below the limits of human perception. The impact of the lack of real-time scheduling support on audio conferencing and a proposed solution to remove the negative effects is presented in Chapter 4.

2.5 Mbone Conferencing Tools

The composable toolkit model adopted by Mbone multimedia conferencing, has resulted in a small set of applications that cooperate to establish and maintain communication. The tools required to participate in an Mbone session include the following [Handley *et al.*, 1998]:

- Session directory;
- Audio media tool;
- Video media tool;
- Shared workspace tool;
- Conference controller.

Available Mbone sessions are advertised through the SDR [Handley, 1997c] session directory. Creators of sessions advertise a description on a well-known multicast address, using the Session Announcement Protocol (SAP) [Handley, 1996]. Session descriptions use the Session Description Protocol (SDP) [Handley and Jacobson, 1998] and consist of a list of media and encodings that will be used, as well as start and end time information. SDR listens on the session address collecting announcements and displays them for browsing. A user can then select a session and SDR will start and configure all the appropriate media tools needed to participate.

Audio and video media tools are used to communicate speech and video images of the participants of a conference. The first publicly available Mbone audio tools were vat [Jacobson and McCanne, 1992], VT, and later NeVoT [Schulzrinne, 1995], with vat still being one of the most popular. These tools provide toll-quality speech communication using transmission speeds up to 64 Kb/s. The audio quality provided is degraded by network packet loss, which is a result of the best-effort Internet service. Chapter 3 presents the Robust-Audio Tool (RAT), of which the author is a co-developer, which was designed to address this problem. Available video tools include IVṠ [Turletti, 1994], nv [Frederick, 1994] and vic [McCanne and Jacobson, 1995] which provide slow-scan video at a quarter of TV resolution. The frame rate achieved by these tools is limited by the processing power of the transmitting workstation and the available bandwidth in the network. Vic, which is the most popular of the three, can achieve full frame rate on modern workstations, producing around 300 Kb/s out of an image of a moving person, using a simplified H261 [H261, 1993] software codec.

Available shared workspace tools are wb [Floyd *et al.*, 1997], which provides shared whiteboard facilities, and nte [Handley, 1997b], which is a shared text editor. These applications present a working document that can be edited by any participant in the conference. The user making modifications is identified at remote sites. Facilities to import existing documents and save or print the current document state are provided. To protect against overwriting by other participants, locking mechanisms exist for objects within the document.

Each of the media tools in an Mbone session operates its own RTP session for data transmission and uses RTCP for membership and reception quality reporting. Any control functionality, like volume controls for the audio tool, as well as the information collected through RTCP reports, is displayed in the tool user interface. This arrangement contains a lot of redundancy, as a lot of information, like the participant list, is replicated for each tool. Summary information about each user is not available at a single location. To solve these issues and to provide co-ordinated control of the conferencing environment, a local conference bus is used within each end-host. The individual tools can exchange information and control messages over the bus. A co-ordinating application can provide centralised administration functionality for the media tools and enforce conference policy.

An example of such an arrangement is the unified conferencing interface built by the author for the ReLaTe [Buckett *et al.*, 1995] project (Figure 2.4). The project piloted remote language teaching over the SuperJANET UK academic network. Language students and teachers, who were novice users of multimedia conferencing technology, required a single monolithic user interface with a reduced number of options to what the individual media tools offered. The interface was constructed from modified versions of the RAT, vic and wb tools. The local conference bus was used to communicate information like audio power meter levels for individual users that were received by the video tool and displayed next to each users image.

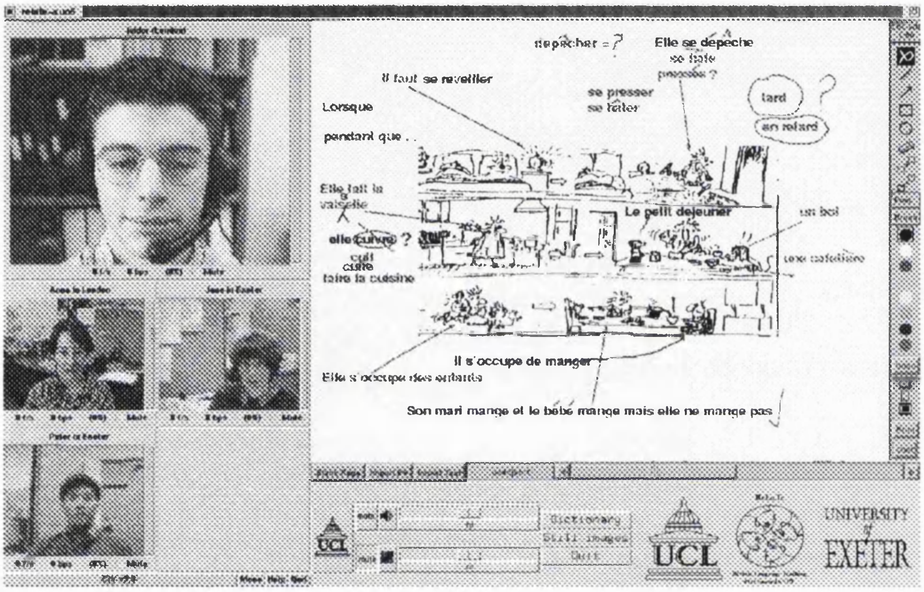


Figure 2.4: ReLaTe unified user interface for language teaching.

Commercial interest in Internet multimedia communication has focused on the provision of Internet telephony services through PC based networked hosts[2]. Voice and slow-motion video communication is typically provided between two end parties over modem speed connections. Many of the newer products use the ITU H323 [H323, 1996] standard for communication, which is based on RTP, and can thus interoperate. Gateway services are offered on the net which can relay a call between an Internet workstation and a PSTN telephone. Multi-way conferencing facilities, when available, operate

[2] A list of available Internet telephony applications is available through the voice on the net web page at http://www.von.com

through centralised servers that act as rendezvous points and thus do not scale. So far, commercial products that make use of multicast are restricted to one-way broadcast type dissemination of TV quality signals. Companies like Precept and Icast provide broadcast servers, that operate on high-end PCs, and viewing clients that can be run on an average desktop PC. The IETF RTP standard is used in the transmission and thus freely available Mbone tools can also be used to receive the broadcasts.

Chapter 3

Robust-Audio Tool

Development of the Robust-Audio Tool (RAT) started in October 1994. The aim of the project was to develop a robust and flexible audio conferencing component for use on general purpose workstations over the Mbone. Existing audio tools at the time, like vat [Jacobson and McCanne, 1992] and Nevot [Schulzrinne, 1995] had demonstrated the feasibility of audio conferencing over the Internet / Mbone, but suffered from a number of problems.

The voice reconstruction algorithm implementations in these tools used silence substitution mechanisms to recover from packet loss. The mechanism was chosen due to its implementation simplicity. Experience through use showed that this reconstruction method was inadequate and produced very poor results for the network loss rates experienced over the Mbone. The need for research into better voice reconstruction mechanisms was the main factor in the decision to implement a new audio tool.

Multimedia conferencing research projects running in UCL at the time, like MICE (Multimedia Integrated Conferencing for Europe) [Kirstein *et al.*, 1995; Handley *et al.*, 1993] and ReLaTe (Remote Language Teaching over SuperJANET) [Buckett *et al.*, 1995], had additional requirements. Investigation in conference control policies for the light-weight session model [Handley *et al.*, 1995], unified user interfaces for use with naive users, and synchronised presentation of multimedia streams, needed a configurable audio component that could be integrated in different architectures.

RAT was demonstrated publicly for the first time as part of the multicast of the Fourth World Wide Web conference in December 1995. At the time, the most popular Mbone audio tool was vat, which was used in all conference multicast transmissions. During the web conference, two parallel audio streams were transmitted with vat and

RAT, enabling receivers to compare the improvement in audio quality that could be obtained with the use of the new voice reconstruction techniques employed by RAT.

Following the initial success, RAT was adopted as the audio tool used in the network meetings of the Multimedia European Research Conferencing Integration (MERCI) project. It has since been used for the multicast of a number of Mbone events and integrated into various multimedia research platforms. The popularity of the tool is growing and it is even being used to receive sessions that are multicast using other audio tools. In such sessions, the improvement in audio quality offered by RAT is less significant, as the transmitted audio stream is not specially encoded to protect against packet loss. As an indication of the scale of use in such conferences, approximately 30 % of participants in NASA sessions[1] use RAT to receive the audio.

The development during initial releases of RAT was undertaken jointly by Vicky Hardman and the author[2]. The success of the tool in the Mbone community has fuelled research surrounding it, and now the RAT audio research group at UCL is continuously growing in size. The current focus of research has expanded to include new areas of multicast audio conferencing like complex receiver audio repair, higher quality audio transmission and sound spatialisation[3].

Attempts to make RAT available to wider audiences have included ports to the many different hardware architectures used in Mbone hosts. Platforms currently supported include SunOS, Solaris, IRIX, HP-UX, FreeBSD, Linux, Windows 95 and Windows NT and a port is under way for Digital Unix[4].

[1] NASA provides live coverage on the Mbone of many space missions. It is always one of the most populated sessions with receiver numbers in the low hundreds.

[2] A complete audio tool was first built by the author and then voice reconstruction functionality was designed and implemented in cooperation with Vicky Hardman.

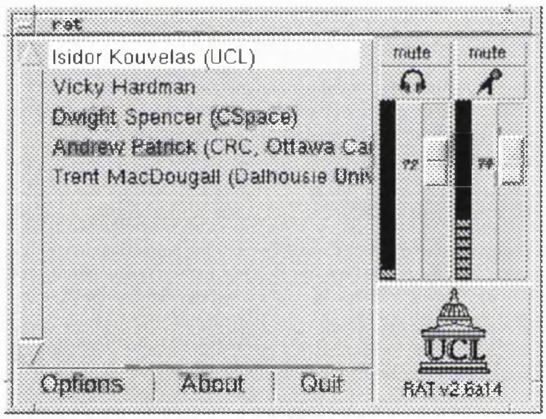
[3] In all releases up till now, the author has been responsible for several revisions of the main audio processing modules of RAT. These have provided operating system adaptation, inter-stream synchronisation, support for multiple sampling rate audio and multiple audio channels (stereo). Further research in voice reconstruction including complex receiver repair techniques is carried out by Orion Hodson who is also responsible for the sample rate conversion functionality and finishing off the stereo support. Colin Perkins has implemented remote control functionality through a conference bus, the RAT transcoder and added encryption modules. Markus Iken has provided the wide-band speech codec implementation and is currently working on provision of sound spatialisation.

[4] The author is responsible for the SunOS, Solaris, IRIX, Windows 95 and Windows NT ports. HP-UX

This chapter continues by outlining the basic operation of a multicast audio conferencing application and concludes by presenting the solutions and improvements introduced in RAT. In particular, the novel voice reconstruction technique used in RAT to counter network packet loss is described and evaluated.

3.1 Structure of an Audio Conferencing Tool

Mbone audio tools provide real-time multi-way audio communication. Single channel telephone quality speech is used to restrict the bandwidth requirements of the audio stream. The audio tools provide a user interface (Figure 3.1), which includes power meters to give the user an indication of send and receive volumes. Since it is difficult to distinguish between multiple speakers using monaural sound (one channel copied to both ears), a list of participants names appears in the interface window, with the current speaker being highlighted while they are speaking.



Isidor Kouvelas (UCL)
Vicky Hardman
Dwight Spencer (CSpace)
Andrew Patrick (CRC, Ottawa Ca
Trent MacDougall (Dalhousie Univ

Figure 3.1: RAT user interface.

The structure of the RAT audio tool can be seen in Figure 3.2. Speech samples are collected from the audio device and after processing, transmitted in packets across the network. At the receiver, the speech samples are retrieved, the reverse processing is applied to convert them back into the original sample stream, and they are fed out to the audio device for the user to hear. Modules in solid boxes in the figure represent basic functionality available in most Mbone audio tools and are described in the rest of this section. Implemented RAT solutions and current research are indicated with dashed and

and Linux was done by Colin Perkins and FreeBSD by Orion Hodson.

dotted boxes respectively.

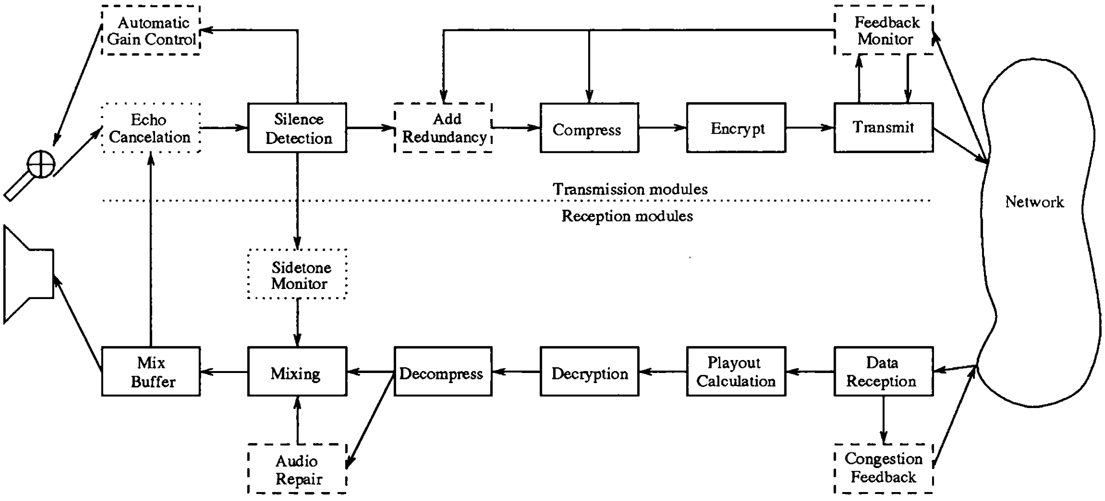


Figure 3.2: Audio processing modules of a conferencing tool.

3.1.1 Silence Detection Algorithms

Silence detection and suppression is commonly used in packet speech systems, since it can provide bandwidth savings of up to 50 % [Brady, 1968]. Input speech samples are categorised as either speech (a talkspurt) or silence, and during periods of silence packets are not transmitted. Silence detection algorithms in currently available audio tools use a simple average energy measure, that works well in acoustically quiet environments. A post-hang period of 200 ms is normally added to the end of the talkspurt to make sure that the silence detection does not cut in during a sentence. The push-to-talk facility is often used in conjunction with the silence detection mechanism to restrict the number of channels open at any one instant to approximately half of one PSTN voice channel per multi-way conference.

3.1.2 Speech Compression

Packet speech systems use standard speech coding algorithms to provide bandwidth reduction. Speech coding schemes have been developed for telephone networks, where a variety of algorithms have been standardised for the telephone or toll quality of service. The algorithms operate at a few discrete bit-rates. Speech encoding improvement in compression is obtained by increasing the complexity of the algorithm. The quality of the decoded speech at the receiver typically is worse for lower bit-rates. The relationship between bandwidth, complexity and quality for some of the codecs used in Mbone audio tools is given in Table 3.1.

Coder	Bandwidth (Kb/s)	Complexity (10^x ips)	Quality (MOS)
μ-law	64	4	4.2
ADPCM	32	5	4.1
GSM	13.2	7	3.6
G.723.1	5.3/6.3	7.3	3.5/4
LPC	2.4-5.6	6	2.4-3

Table 3.1: A comparison of the speech codecs (after Cox and Kroon [Cox and Kroon, 1996]).

Apart from the encoded stream that is produced by each codec, there is additional state that is maintained in synchrony at the encoder and decoder. When part of the encoded stream is not received by the decoder, due to packet loss, synchronisation in the maintained state breaks. To prevent decoder miss-tracking, packet speech systems transmit some additional transmitter state in the packet stream to enable resynchronisation. This is feasible as the Internet allows variable packet sizes. It is thus a requirement from the codecs used in Mbone tools that this extra state is available and small in size, so that it introduces only a small overhead in stream bandwidth. This is true for the ADPCM codec, where the extra state needed to resynchronise is only 3 bytes.

3.1.3 Audio Stream Encryption

In IP based networks like the Internet, it is almost impossible to guarantee that a connection is secure against eavesdropping, especially when multicast is used for the dissemination of the data [Ballardie and Crowcroft, 1995]. Multicast conferencing tools use end-to-end encryption to prevent uninvited third parties from joining a session. Asymmetric public-key cryptographic algorithms offer significant advantages as they provide authentication in addition to privacy. The private key of the participant is used to encrypt the data and the receiver can authenticate and decrypt the stream using the corresponding public key. Unfortunately, as a global public-key management infrastructure is not available on the Internet, symmetric encryption schemes like DES are used instead and the keys distributed through external mechanisms to the intended participant list.

3.1.4 Speech Transmission

The coded speech samples are collected into packets and transmitted over the network. At the receiver an artificial voice reconstruction delay is introduced, in order to smooth out the effects of network delay jitter (Chapter 2, Section 2.1). This is achieved by buffering received packets for enough time, so that most of the late packets have a chance of being received before their presentation time. In order for receivers to be able to reconstruct the timing of the received packets and calculate scheduled playout times, sequence numbers and timestamp information is included in RTP [Schulzrinne *et al.*, 1996] headers. The voice reconstruction delay needs to be adaptive to cope with changing network conditions. Receivers maintain a running estimate of network delay and jitter and readjust the amount of buffering between talkspurts [Ramjee *et al.*, 1994; Montgomery, 1983]. The amount of buffering is enough to receive most of the packets, but some will always arrive too late to be played back and hence, can be considered lost. This process is illustrated in Figure 3.3 provided by Dr Mark Handley.

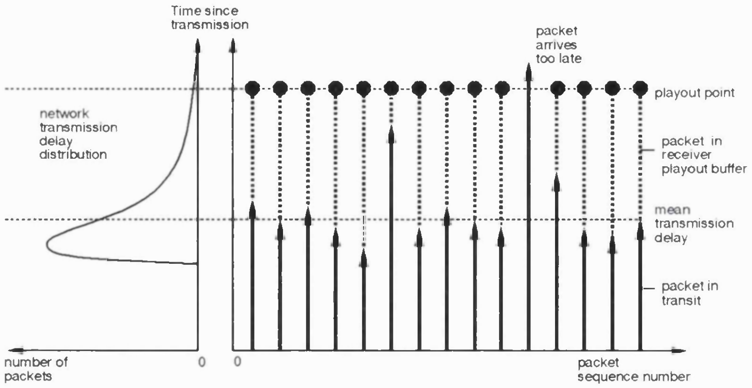


Figure 3.3: Voice reconstruction buffering at receivers to remove network delay jitter.

In packet speech systems, minimising the end-to-end delay is always an important criterion if user interactivity is to be maintained. The value of the end-to-end delay should be kept below 600 ms in the absence of echoes, but is recommended to be restricted to less than 400 ms if conversation patterns are not to break down [Brady, 1971]. End-to-end delay is directly impacted by the size of packets transmitted over the network. A delay equal to the size of one packet is incurred at the transmitter, since samples in the packet have to be collected before a packet can be sent. A rough estimate of the reconstruction delay required to smooth out packet arrival at the receiver is two packets worth in ms, although the true value may be substantially in excess of this rule of thumb, particularly if packet reordering is possible. To this delay, the network propagation delay has to be added, to derive the effective round-trip time for the connection.

Minimising the packet size used in transmission, to keep the end-to-end delay low, has the additional advantage of reducing the impact of packet loss to audio quality. The loss of a smaller packet results in a less perceptible gap in the reconstructed audio at the receiver. However, the use of a larger number of smaller packets increases the bandwidth overhead of packet headers and any processing incurred at network nodes. Consequently, a trade-off exists between the requirements of the network and the requirements of real-time voice connections.

3.1.5 Mixing

In a multicast conference more than one person can be speaking at any one instant. Presenting the multiple incoming audio streams using a single output device at receivers requires mixing. This is done by adding the scaled decompressed audio from each source.

3.2 Robust-Audio Tool Solutions

There are currently a number of problems which affect the quality of Mbone audio, of which packet loss is the most significant. Repair methods for packet loss are known as voice reconstruction mechanisms. Employing such mechanisms in real-time packet audio systems is necessary to maintain the correct timing relationship between the transmitter and receiver during packet loss. The goal of a voice reconstruction algorithm is to construct a suitable replacement dummy packet at the receiver, so that the loss is as imperceptible as possible. The presence of the dummy packet is usually discernible to the listener and the perceptibility of loss increases with increasing packet size and loss rate.

Voice reconstruction techniques can be split into two categories: receiver only and combined source and channel techniques. Receiver only techniques are those that try to reconstruct the missing segment of speech solely from information at the receiver. Combined source and channel techniques try to make the system robust to loss, either by arranging for the transmitter to code the speech in such a way as to be robust to packet loss, or by transmitting extra information to help with reconstruction.

The rest of this section describes voice reconstruction solutions and other developments in RAT. In addition to a number of receiver repair mechanisms, a novel combined source and channel technique using audio redundancy has been developed and evaluated. Solutions for workstation performance adaptation, lip-synchronisation, high quality audio and sound spatialisation are also outlined.

3.2.1 Receiver Repair

RAT uses a range of receiver repair techniques to recover from packet loss. The choice of techniques is based on restricted algorithm complexity, so that they can be executed on the multimedia workstations currently available on the Mbone.

- Silence substitution inserts silence in the reconstructed audio stream for the duration of the missing packet. It is used in many packet speech systems because it is very simple to implement. The technique gives adequate speech quality for small packet sizes (less than 16 ms), and up to 2 % loss [Jayant and Christenssen, 1981]. Since audio over the Mbone uses comparatively large packets (20, 40 or 80 ms) for speech transmission, the use of silence to repair speech has a serious impact on the intelligibility and quality. The degradation depends on the size of packet used and the loss rate experienced. Experience from multimedia conferences over the Mbone has shown that users will tolerate higher loss rates. Approximately 10 % is agreed upon as being the limit at which speech becomes difficult to understand. However, a loss rate of 10 % is frustrating to the listener.

- The use of white-noise instead of silence as the fill-in for the missing audio segments has been shown to be subjectively better and improve intelligibility [Miller and Licklider, 1950; Warren, 1982]. The packet size and loss rate limits of this approach are however similar to those of silence substitution.

- Packet repetition techniques assume that speech characteristics do not change very rapidly, and a preceding segment of speech is used to reconstruct the missing packet. The mechanism fails when the packet sizes are large and the loss rate is high, since the speech characteristics are more likely to have significantly changed [Warren, 1982].

Current receiver repair research in RAT is investigating the use of more complex techniques. These include interpolation based schemes, that attempt to interpolate from packets surrounding a loss, and regeneration based schemes, that use knowledge of the audio compression algorithm to derive a synthesised packet from codec parameters. The improvement in perceived quality from these repair schemes is small compared to the increase in computational requirements.

3.2.2 Audio Redundancy

Combined source and channel techniques are able to provide significant improvements over receiver only techniques. This section presents a novel scheme developed with RAT that uses audio redundancy. A survey of other possible source and channel solutions is first presented.

- Interleaved positioning of speech intervals in transmitted packets at the sender can considerably reduce the size of missing audio segments when loss occurs. The scheme operates by re-sequencing units of speech before transmission, so that originally adjacent units are transmitted in separate packets. Packet loss causes smaller segments to be lost, that are spread in the audio stream. These smaller segments are easier to repair with receiver techniques, than one large gap, equal in length to the sum of the small segments in a packet, would have been. The disadvantage of interleaving is that the transmitter has to wait for all the segments in a packet to be collected before the packet can be transmitted. This significantly increases the end-to-end delay of the connection making it unsuitable for interactive communication.

- Standard forward error correction techniques [Lin and Costello, 1983] can be applied to the audio stream making it possible to completely recover from a certain level of packet loss. The repair data can be in the form of set of parity packets that are calculated for each set of transmitted audio packets. Receivers experiencing loss can calculate the information in the missing packets, provided that the number of losses is less than the number of parity packets used. This technique suffers from similar delay problems as interleaving, since the transmitter has to collect a set of audio packets before the parity packets can be calculated and transmitted.

- Embedded speech coding techniques split a signal into a number of layers that can be combined at the decoder to achieve increasingly refined versions of the original. Base quality layers are transmitted in packets which can be treated with higher priority by the network and stand lower chances of being lost. This ensures that low quality information will always be available at the receiver. Higher layer packets can be sent with a lower priority and dropped by the network when congestion occurs. The technique can provide very good quality under congested conditions, provided that there is enough available bandwidth for the lower quality layers. The mechanism relies on network support which is not currently available on the Internet.

  An alternative for the layered transmission of information in multicast conferences is proposed in [McCanne *et al.*, 1996], where the task of reacting to congestion is left up to the receivers. Different quality layers are transmitted on separate multicast addresses, and receivers join only as many groups as their connection can accommodate. This removes the requirement for different quality of service support from the network. The two approaches differ, as a priority transmission scheme aims at reducing the impact of losses during congestion, whereas the layered transmission and receiver adaptation aims at driving the network in an uncongested state and operates on different timescales.

Within RAT we have developed a new voice reconstruction scheme that uses audio redundancy. The redundancy is a low bandwidth speech encoding, which, while not capable of providing adequate speech quality as the primary compression scheme, is very suitable for use as a fill-in segment in a voice reconstruction mechanism. The redundant encoding for an audio segment corresponding to a packet is piggybacked to the following packet (Figure 3.4). The output of a synthetic quality speech coding algorithm can be used to provide the redundant information. LPC is regarded as containing approximately 60 % of the information content of the speech signal and provides high compression of the input stream, thus, being ideal for the task. When a packet has been lost, the receiver decodes the redundant information available in the next packet. Consequently, the output speech waveform consists of periods of toll quality speech, interspersed with periods of synthetic quality speech.

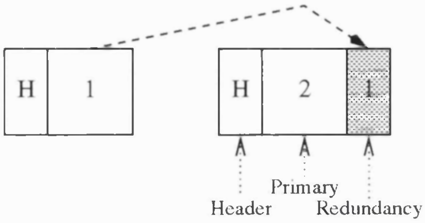


Figure 3.4: Piggybacking of redundant encoding on audio packets.

The use of redundancy requires an increase in the voice reconstruction delay, by the time equivalent to the distance of the redundancy component to the packet with the primary encoding. For redundancy placed in the packet behind that containing the primary speech codewords, an extra delay of one packet should be added to the voice reconstruction delay. The additional delay increases the chance of the redundant information being available at the receiver by its scheduled playout point.

3.2.2.1 RAT Subjective Quality Results

In order to assess the improvement in audio quality that results from the packet loss robustness technique, listening tests using RAT have been carried out. The material consisted of recordings of passages from magazines, spoken by a male speaker with no accent. RAT was used in the experiments, with input speech taken from the file. The packetised audio was transmitted across the network to a packet reflector that randomly loses packets for different loss rate settings. The packet speech was then received using RAT and stored in files.

The files were played back in a randomised order to 10 subjects, each of which listened to 12 different conditions, plus an experimental control condition. The speech quality was rated by the subjects using Mean Opinion Scores (MOS)[5]. The graph in Fig-

[5] Although MOS have traditionally been used in the telecommunications world to rate toll quality speech, there are a number of problems when they are applied to packet audio systems [Watson and Sasse,

ure 3.5 shows the perceived speech quality improvement gained from using redundancy (one level), against speech received at same loss rates without redundancy.

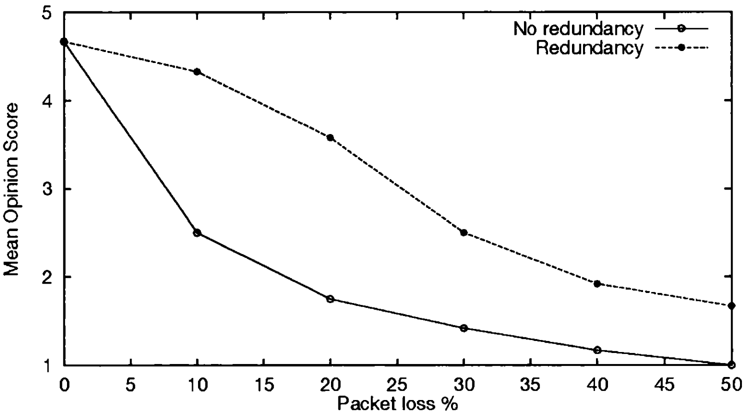


Figure 3.5: Perceived speech quality at different loss rates with and without redundancy.

It can be seen from the graph, that for all loss rates measured, redundancy offers a significant improvement in the subjective quality. At lower loss rates, the improvement in MOS rating is greater than for higher loss rates. This can be explained by the packet repetition scheme that was used to increase the repair interval from 1 to 2 packets, which was not used on consecutive losses for which it is known to fail.

3.2.2.2 Use of Multiple Redundancy Levels

The addition of a piggybacked redundant encoding for the previous packet can provide fill-in audio for one lost packet. If two or more consecutive packets are dropped by the network, then both the primary and redundant information for some period of time will be lost. Theoretically, by increasing the number of redundant blocks that we add to each packet, we can cover for a loss of any length. In practice, there are limitations on the packet size we can use and hence, the maximum bandwidth it is desirable for our audio stream to take up. Receivers that are suffering from loss, must ensure their playout buffers are large enough to cater for the amount of redundancy, and so incur extra decoding delay.

The following analysis attempts to determine the number of levels of redundancy

1996].

needed to cover for loss under different network conditions. As a first approximation, we assume that the losses occur independently and can be modelled by a Bernoulli random variable. Although this assumption is not entirely accurate, as is discussed in § 3.2.2.3, the results still give a fairly good indication of the usefulness limits of multiple levels of redundancy.

Assume a packet loss rate p over the network and n levels of audio redundancy. In order for a unit not to arrive at the receiver, all of the n packets carrying it have to be lost. Hence the perceived unit loss rate l at the receiver will be p^n. This can also be shown as follows. The probability $P(k)$ of the network loosing k consecutive packets and correctly delivering packet $k+1$ is given by:

$$P(k) = p^k(1-p) \tag{3.1}$$

So the probability of the network loosing n or more packets is:

$$l = \sum_{k=n}^{\infty} p^k(1-p) = p^n \tag{3.2}$$

By solving for n, we can calculate the number of levels of redundancy that are required to achieve a desired unit loss rate l at the receiver, given the network packet loss rate p:

$$n = \left\lceil \frac{\log(l)}{\log(p)} \right\rceil \tag{3.3}$$

Figure 3.6 shows the number of levels of redundancy required to achieve different stream loss rates at the receiver against network loss and was plotted using Equation 3.3.

Figure 3.7 contains three cross-sections of Figure 3.3, and shows the number of levels of redundancy required to achieve 5 %, 10 % and 20 % stream loss rates at the receiver against network loss rate. A 5 % loss rate in the resulting audio stream represents a reasonable loss rate that can be patched using receiver techniques like waveform substitution. Even with the highest available compression used for the redundancy (at the moment LPC), it is not reasonable to expect more than 5 or 6 levels, since it increases the end-to-end delay. For instance, 5 levels of redundancy and 20 ms packets requires an extra delay of 100 ms, meaning that the end-to-end delay must be significantly less than 150 ms to accommodate jitter and remain interactive. With lower compression schemes, such as ADPCM, maximum packet size becomes a limiting factor. The curves show that above a network loss rate of about 50 %, it becomes very hard to consistently cover for the packet loss, since the number of required redundancy levels becomes very high.

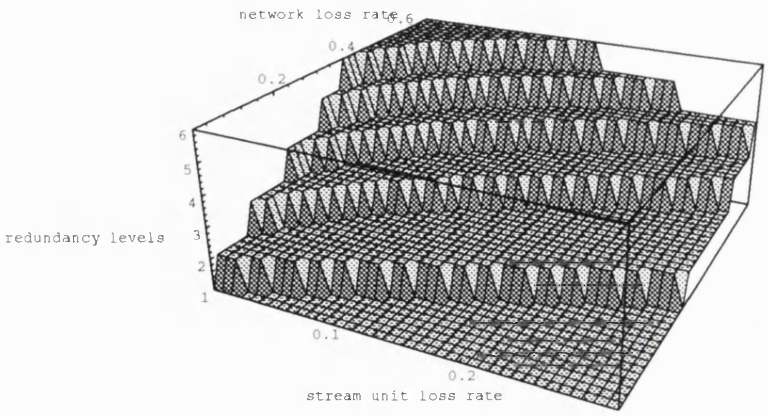


Figure 3.6: Number of redundancy levels required against loss rate and perceived quality.

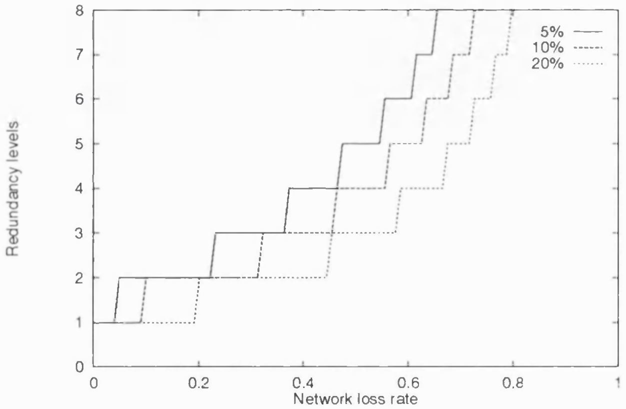


Figure 3.7: Number of redundancy levels required against network loss rate.

There should be a cut-off point for the network loss rate where the audio application decides, or indicates, that interactive communication is not possible. The limits indicated by the results in this section are optimistic, as, in reality, loss is not totally random and some correlation between losses exists.

3.2.2.3 Redundancy Spacing

There have been a number of surveys of the packet loss patterns on the Internet / Mbone investigating the correlation in loss. Handley [Handley, 1997a] analysed loss logs from popular Mbone sessions and concluded that although the deviation of the observed loss patterns from random loss is statistically significant, it is not sufficient to influence the design of a redundancy control algorithm. Similar results are shown in [Bolot, 1993] and [Yajnik *et al.*, 1996]. Here, we present a set of measurements that we collected which confirm the above findings and discuss their impact on audio redundancy.

Loss statistics were collected using a variety of both unicast and multicast links between UCL, France, Greece and the US. The results are not an exhaustive set of measurements, but rather are indicative of the conditions that might be expected. A number of data sets were collected for different loss rates, by looking for links that showed the approximate desired loss rate. Each data set has been constructed using 336 byte packets, generated at 80 ms intervals, which is typical of voice traffic generated by current audio tools[6]. Approximately 10,000 packets per data set were collected.

Each of the graphs in Figure 3.8 shows the deviation of the measurements from the expected random loss for a different packet loss length. We observe that because of an existing degree of correlation in packet loss patterns, the measured values deviate from the theoretical ones. The single packet loss graph shows lower actual losses, whereas the longer gap length graphs show that the actual values are larger.

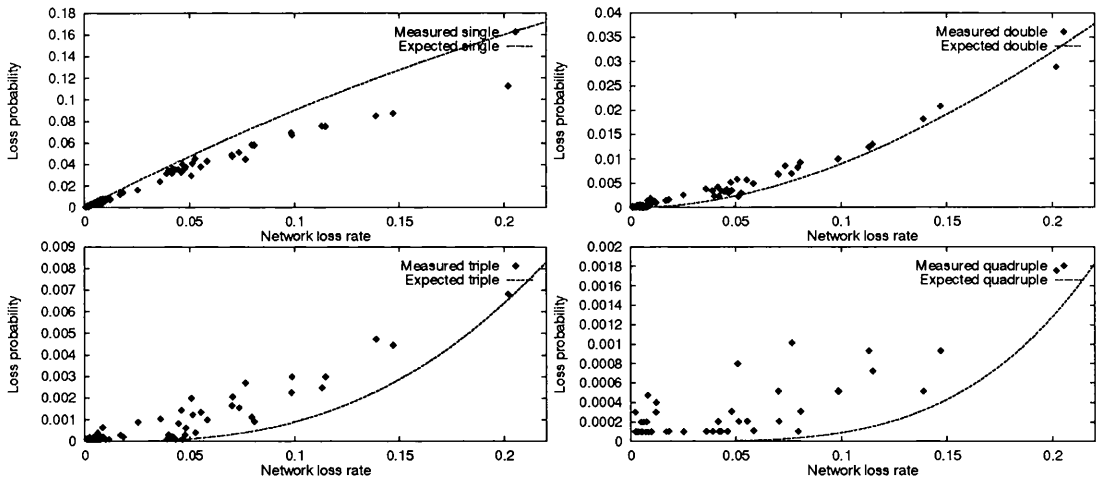


Figure 3.8: Expected and observed loss length probabilities against network loss rate. (Top row: single packet loss events, double packet loss events. Bottom row: triple packet loss events, quadruple packet loss events).

Because the measured loss patterns display some degree of correlation, if packet n is lost, then there is a higher than average probability that packet $n + 1$ will also be lost. This indicates that an advantage in perceived audio quality can be achieved in the

[6] The audio packet size is coded for 80 ms of 8000 kHz 4-bit mono-aural ADPCM.

redundancy mechanism by offsetting the redundancy in the packets. Figure 3.9 shows how this can be achieved. The change is that packet n carries the redundancy for packet $n - 2$ instead of $n - 1$. Increasing the distance of the redundancy from the primary encoding, forces the receiver to wait longer before playing out audio in received packets, to give a chance for the redundant block to arrive. The further the redundancy is offset, the longer the playout delay at the receiver has to be. As the measured probabilities of longer then double packet losses for moderate network loss are still very low, most of the improvement in quality can be gained by offsetting the redundancy by a single packet.

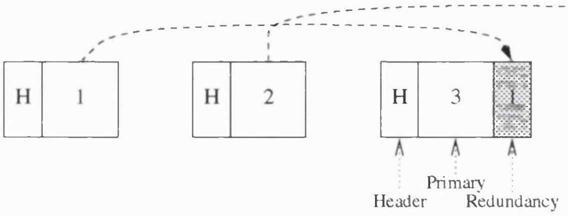


Figure 3.9: Offsetting redundancy.

Figure 3.10 shows the change in perceived loss rate at the receiver when the redundancy is offset by one packet (as shown in Figure 3.9). Positive values indicate an improvement, whereas negative values indicate a degradation. It is clear that for the majority of measurements the perceived loss rate is reduced by offsetting the redundancy. There is also a visible trend indicating that for higher loss rates, the improvement is more significant.

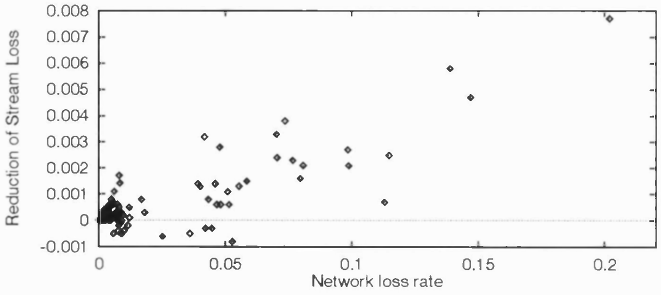


Figure 3.10: Perceived loss rate difference between normal and offset redundancy against network loss rate.

3.2.2.4 Congestion Control

Currently available audio conferencing applications transmit data at a constant rate. There is no control of the transmission rate to adapt to network conditions and no back-off in the face of congestion. The aim of any adaptation scheme should be to achieve optimal audio quality and fairly share available bandwidth in the network.

For a single network flow, an improvement in quality during congestion can be obtained by increasing the level of redundancy. This, however, results in an increase in the amount of traffic introduced into the network, which adds to the congestion. If all real-time flows react in the same way, then no improvement will be obtained, as the packet loss rate will increase proportionally to the added redundancy. In fact, quality may drop, as the burstiness of losses may increase.

A first step in achieving fairness, is to avoid increasing the amount of data being transmitted when packet loss is observed. A control algorithm can be tuned, to maximise perceived audio quality at the receivers for a given desired network bandwidth usage. This can be done by increasing the compression of the primary encoding and trading it for an increase in redundancy (Figure 3.11).

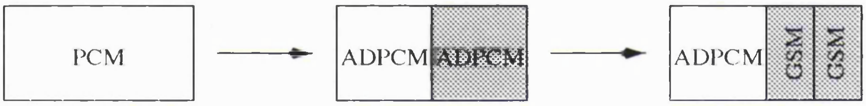


Figure 3.11: Trading off primary encoding bandwidth for redundancy.

However, maintaining a constant transmission rate and just adapting the level of redundancy to adjust perceived quality is not adequate. In addition to the limitations described above, such a scheme forces all adaptive traffic (like TCP) to back off completely, making communication through traditional non-real-time applications impossible [Jacobson, 1988; Floyd and Fall, 1998]. The aim of an application transmitting a real-time stream should be to detect congestion, through feedback information from receivers, and react by reducing the transmission rate.

There have recently been some attempts at providing network-friendly real-time audio applications. In [Bolot and Garcia, 1996], an adaptation scheme for multicast conferencing is described, where the transmission parameters are modified, based on averaged group statistics, collected through RTCP receiver reports. This is achievable with unicast communication (two-way conferencing). With multi-way conferencing the scaling requirements of multicast communication complicate the adaptation process. It is impossible to find a single optimal transmission rate and level of redundancy that satisfies the requirements of a diverse receiver group. Implosion of feedback information from the multiple receivers at the sender is an additional problem.

Chapter 6 presents and evaluates an architecture for congestion controlled multicast real-time communication. The proposed solution uses self organisation to form groups out of receivers suffering from a common bottleneck link, and arranges locally for a reduced bandwidth stream to be forwarded to the problem group. The bandwidth of the customised stream is continuously adapted in a TCP friendly manner, based on feedback information from a group representative, thus solving the implosion problem.

3.2.3 Operating System Adaptation

In addition to network loss, audio quality is affected when the processing capacity of the sending or receiving workstation is exceeded, resulting in speech quality degradation similar to that caused by packet loss. Although the processing power of current multimedia workstations is far greater than that required by an audio conferencing application, CPU time is a shared resource controlled by the operating system scheduler. Because of the lack of real-time process support in today's general purpose operating systems, no guarantees can be given that the audio process will receive the required share.

A mechanism has been developed within RAT, which produces a continuous audio signal, despite the variable allocation of processing time a real-time application is given. Continuity of audio is ensured during scheduling hiccups by using the buffering capabilities of workstation audio devices. The details of the design are presented in Chapter 4.

3.2.4 Inter-Media Synchronisation

The transmission of multimedia over packet networks and the use of independent systems for audio and video, mean that listeners often experience a time lag between hearing a remote user's audible words, and seeing the associated lip movements. RAT and the LBL video conferencing tool vic [McCanne and Jacobson, 1995] were modified to achieve the first implementation of lip synchronisation between audio and video over the Mbone. The details of the solution to provide the inter-media synchronisation are presented in Chapter 5.

3.2.5 High Quality Audio

The acoustic environment provided by current audio tools is somewhat unnatural and may deter first-time users. Increased speech intelligibility can only be provided by coding and transmitting more of the input speech bandwidth. A reduced complexity version of a wide-band speech coding algorithm [BT, 1984; Hardman, 1993] is currently being developed for RAT. The wide-band speech coding algorithm transmits 7 kHz audio, and is well-known to preserve more of an individuals speech characteristics, as well as increasing speech intelligibility. The bandwidth requirements of the wide-band speech codec are the same as PCM (64 Kb/s), making it ideal for use over the Mbone.

3.2.6 Sound Localisation

Sound localisation artificially manipulates a single input channel to produce 3D spatial sound. The sound can be presented over headphones or via loudspeakers, but the effect is better than stereo, since the sound appears to emanate from a particular location (outside the user's head with headphones). Sound localisation is being developed for RAT and will be used to separate conference participants out in space [Hardman and Iken, 1996]. This substantially eases speaker identification and improves speech intelligibility.

3.3 The RAT Transcoder

Releases of RAT since the beginning of 1997 include transcoding capabilities. When operating in such a mode, RAT acts as a gateway between two separate sessions, relaying RTP audio and RTCP participant information. There are several conferencing scenarios that require such functionality:

- Users wishing to participate in a multicast conference connected through a low capacity link, require some mechanism to restrict the session bandwidth. The RAT transcoder can provide bandwidth savings, by forwarding a single mixed stream with higher compression, from the set of participants on each side of the low speed link to the other. A common example of such a setup is a user that connects to the Internet, using a single basic rate ISDN connection at 64 Kb/s, and wishes to participate in a session with multiple participants sending audio and video. A transcoder situated at the site of the Internet service provider can customise the session stream before forwarding down the ISDN link. This configuration is shown in Figure 3.12.

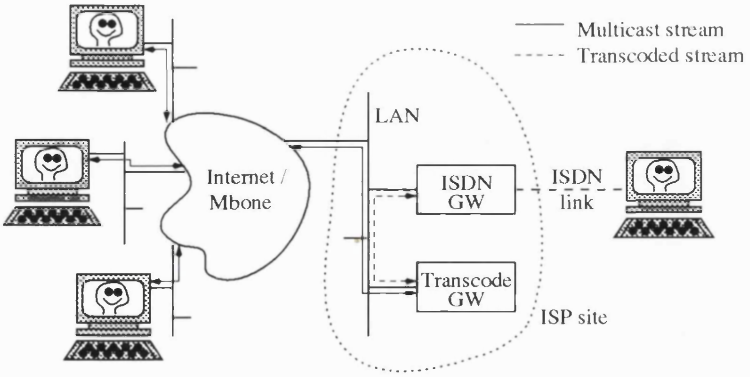


Figure 3.12: Connecting to a multicast session through a transcoder.

- Although deployment of the Mbone on the Internet is progressing rapidly, multicast routing is not yet ubiquitous, and so there are sites that wish to participate in Mbone sessions but cannot do so. An interim solution for users in such areas is to situate a transcoder at the closest site reached by the Mbone and forward a unicast stream to their site.

- Users connected through lossy links (either due to network congestion or other connectivity problem) may wish to transcode the audio stream to include additional redundant information before forwarding through the problem link. An improvement in quality at the receivers can thus be achieved.

The current implementation of the transcoder in RAT can convert between all the encoding schemes supported by the audio tool and add or remove a single or multiple levels of redundancy. In addition to audio data, session membership and other items of information carried in RTCP packets are forwarded according to the definition of a mixer in RTP [Schulzrinne *et al.*, 1996].

Chapter 6 presents a scalable congestion control scheme for real-time streams of which media transcoding is an integral part.

3.4 Conclusions

This chapter has presented a new Robust-Audio Tool (RAT) for use in multicast multimedia conferencing, which introduces a novel voice reconstruction technique for packet networks. Since its first release, the improved quality that RAT offers has made it popular in the Mbone community. RAT is currently being used for research project meetings, remote language teaching sessions and conference multicasts across Europe and the US.

The audio redundancy voice reconstruction mechanism used in RAT was evaluated and its potential and limitations were identified. Additional achievements of RAT were outlined, with some being presented in more detail in the following chapters.

Chapter 4

Overcoming Workstation Scheduling Problems in a Real-Time Audio Tool

Traditional time-sharing operating systems for general purpose workstations do not provide adequate support for real time applications [Faller, 1992]; they operate in an asynchronous fashion, and handle data in blocks [Hagsand and Sjodin, 1994]. Real-time audio applications are 'soft', in that they have to keep a continuously draining operating system audio device driver fed with blocks of audio samples, but there is no specified instant when audio must be transferred, just a dead-line. However, the lack of a delay bound for notification on timers and external events makes it impossible to guarantee a regular supply of audio samples. Current real-time audio applications ignore the problem, which results in frequently disrupted audio, and increased end-to-end delay.

A solution analogous to that of resource reservations to provide QoS guarantees over the network, is to modify the operating system scheduler to provide bounded dispatch latency for applications [Hagsand and Sjodin, 1994; Khanna *et al.*, 1992; Mullender *et al.*, 1994; Fisher, 1992]. Despite experimental proof that this approach can significantly improve the perceptual quality of multimedia applications, it has not been implemented in the majority of desktop workstation operating systems.

There is a very large number of general purpose workstations connected to the Internet, used for audio conferencing applications with the current scheduler. We present a solution to the problem at the application level, that smoothes out scheduling jitter in audio applications using the workstation device driver buffering capabilities. The mechanism produces a continuous audio signal, despite the variable allocation of processing time a real-time application is given. The solution is analogous to playout adaptation for smoothing network jitter, in that it does not require any modifications to workstation hardware or operating system.

In this chapter, we analyse the audio scheduling problem and its implications. The solution and implementation in RAT is described, together with evaluation results that show the success of our method.

4.1 Application and Operating System Interaction

The audio device of the workstation can be visualised as two buffers:

- The input buffer continuously inputs samples from the analogue audio input (microphone).
- The output buffer is fed with samples from the application and these are synchronously output to the loudspeakers.

The rate of input sample accumulation and sample output is the same (sampling rate). The operating system buffer size in the device driver has a fixed upper limit, which is adjustable under most OS platforms.

The audio application interacts with audio input and output through the two device driver buffers. Samples are read from the input buffer for processing and written out to the output buffer for playback. In contrast with actual audio playback, the transfer of blocks of samples between the audio application and the device driver is *ad-hoc*. Blocks are read from and written to the device driver buffers, and all audio processing takes place on this block unit size. The minimum block size is enforced by the workstation processing power and hardware capabilities. The larger the block size, the less frequently the application has to read and write audio blocks.

Figure 4.1 illustrates the basic functionality of our audio tool. The operation is similar to that of other existing conferencing audio tools like VAT [Jacobson and McCanne, 1992], NeVoT [Schulzrinne, 1992] and IVS [Turletti, 1994].

For the purposes of this chapter the audio tool can be considered as a process that acts as an interface between the audio device and the network. The process provides a two way communication facility. Samples input from the audio device are processed and transmitted across the network to remote conference participants. Packets are received from the network, processed, and samples played out to the audio device.

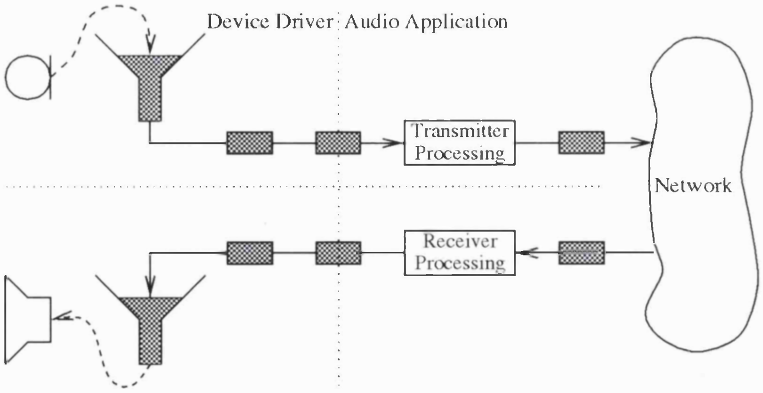


Figure 4.1: RAT operation.

A detailed description of the functions involved in processing the audio samples and more information on the structure of the RAT audio tool have been given in Chapter 3.

4.2 Audio playback problem

The effect of sample loss on audio may produce significant degradation in audio quality, depending on the length of individual gaps and the frequency of occurrence. Audio gaps indicate a pause in speech to the human brain causing confusion and reducing intelligibility. The minimisation of sample loss is a primary design goal in audio applications.

In real-time conferencing audio systems interactivity is known to be substantially reduced for round trip delays larger than 600 ms. Large round trip delays in conversational situations increase the frequency of confusions and amount of both double talking and mutual silence [Brady, 1971]. The components of this delay in a real-time Internet audio tool are:

- Processing and packetisation delay at transmitter.
- Variable delay experienced by audio packets traversing the network.
- Audio reconstruction buffering at the receiver necessary for smoothing out network delay jitter.
- Delay caused by queued audio samples in the operating system device buffers.

There are two main goals when considering operating system effects for achieving good quality real-time audio communication:

- Continuous audio playback with no unnecessary breakups.
- Low delay in device driver buffers to minimise the end-to-end delay, so that interactivity and normal communication patterns are preserved.

To play-back audio, buffers of samples have to be transferred to the audio device driver. If all the samples in the device driver are played out before more are supplied, then the buffers run dry, and audio playback stops. To avoid the resulting gap of silence in the output audio, the transfer of samples must take place regularly.

Modern workstation audio device drivers provide a selection of audio sampling frequencies. Conferencing applications usually aim to transmit telephone quality speech, and consequently, would like to use a sampling frequency of 8 kHz. However, audio sampling frequency crystals usually do not have a nominal frequency of exactly 8 kHz and can vary considerably from one workstation to the next [Jacobson, 1994; Rizzo, 1997]. Timing events using the workstation clock without compensating for the drift in the audio crystal will lead to one conference participant's audio buffers becoming full, while a remote workstation buffers may run dry. To simplify operations the sampling clock of the audio device is used instead. This is achieved by using the number of samples read from the audio device as an indication of the amount of time that has elapsed. Since one crystal is used for audio input and playback[1], the number of samples read gives a very accurate count of the number of samples played out. With the accurate knowledge of the number of samples that have been consumed by the audio device, a receiver can calculate the drift from a transmitter and compensate for by adjusting the duration of silence periods.

[1] This is true for most workstation audio hardware. A notable exception is the SoundBlaster card family available for PCs. In order to provide full-duplex operation in their latest hardware, Creative has incorporated an older model of their soundcard on new cards. Apart from giving different input and output audio qualities this model breaks the timing synchronisation between recording and playback.

4.3 Implications for a Real-Time Audio Tool

Networked audio tools have a strict loop of operation in order to meet real-time timing restrictions. In each cycle of this loop the following basic operations take place:

- A block of samples is read in from the device driver input buffer.
- An equivalent number of received and processed samples are written out to the audio device.
- Other processing takes place which may involve packets being transmitted onto or received from the network.

Using this order of operations attempts to ensure that as many samples as needed are fed to the output buffers of the audio device. Feeding more would create a backlog of samples in the device driver to play out and would increase the delay. Providing less would result in gaps of silence in playback because of the buffers running dry.

On a non-multi-tasking machine, keeping up with this loop is not hard to achieve. Under a time-sharing operating system — like UNIX — this may not always be possible. The UNIX scheduler decides when a process gets control of the CPU. Processes are serviced according to their priority and there is no useful upper bound on the amount of time a process may be deprived execution [Hagsand and Sjodin, 1994; Khanna *et al.*, 1992].

As a result of other processes being serviced, a relatively large amount of time may elapse in between two instances of the audio tool being scheduled. If the time between schedules is larger than the length of audio data that was written last time, then, an audible gap of silence will result in the output audio signal. The persistent effect of the gap in the audio output is to restrict the timing of the output, with respect to the input. Extra delay is accumulated in the device driver, since each block of samples is played out later than it should have been. Measurements have shown that the accumulated interruptions caused by an intensive external event on a loaded workstation, can create a delay of several hundreds of milliseconds over the period of a cycle of operation of our program. Since the audio system is timed from the read operations of the audio device, the time that elapses when the audio tool process does not have control of the CPU will be evident. There will be a lot of audio in the input buffer of the audio device waiting to be read. In response to the waiting input audio, the process will execute the loop several times in succession, until it reads the blocks in. For each of the blocks read, a block of equivalent size will be written out to the output buffers. Figure 4.2 shows the delay increase diagrammatically.

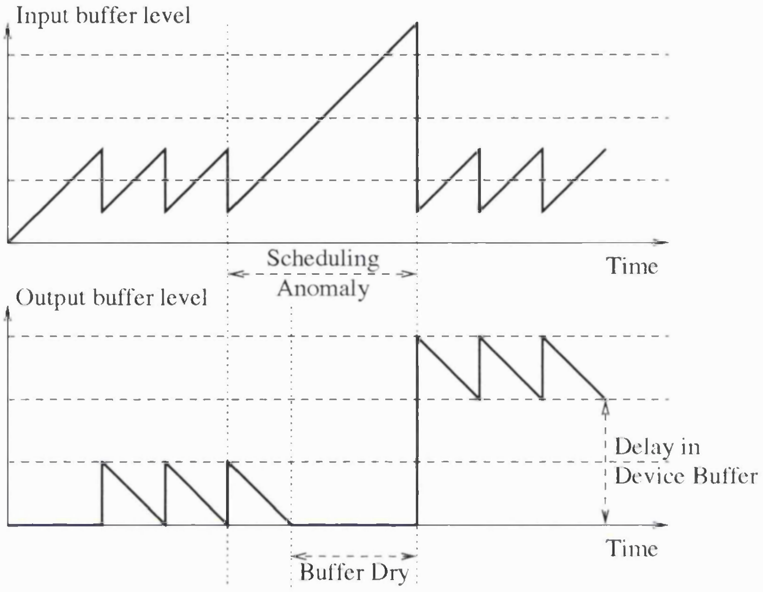


Figure 4.2: Accumulation of delay in output buffer.

The extra buffered audio samples in the device driver prevent additional gaps, as there is now extra audio to play out while the audio tool process is not executing. Current Internet conferencing audio tools like VAT [Jacobson and McCanne, 1992], NeVoT [Schulzrinne, 1992] and IVS [Turletti, 1994] rely on this effect. During a talkspurt, the first CPU starvation causes a gap in the audio. Successive starvations will only cause a gap, if they are longer in duration than the longest starvation so far. Consequently, the delay caused by the buffered samples in the audio device increases up to the length of the maximum interruption. At the end of the talkspurt the accumulated delay is zeroed, as transfer of samples to the audio device stops, and the audio device output buffer drains. If silence suppression is not enabled in the transmitting tool, then the increase in delay will persist throughout the duration of the session.

4.4 Adaptive buffer solution

In order to eliminate gaps in the audio signal and minimise delay in the device driver buffer, adaptive control of the buffers is sought. The adaptation algorithm will trade off the two variables. The adaptation algorithm controls the amount of audio in the buffers, and ensures that there is always a minimum amount of audio to reduce the gaps caused by scheduling anomalies. Monitoring a history of the size of scheduling anomalies means that excessive audio is not buffered.

Intuitively, a situation where the amount of samples in the device driver buffer (called the cushion) can cover for small frequent interruptions is ideal. This principle will not introduce excessive delay, but large infrequent interruptions will still cause a gap in the audio signal. The size of the cushion must be determined by the current performance of the workstation, which can be estimated by analysing a history of scheduling anomalies. As new processes start, or external events happen, the overall perceived load and behaviour of the scheduler will change. In order to maintain the desired sound quality and minimise the delay, the cushion size must be adaptive, and should attempt to reflect the state of the workstation (Figure 4.3).

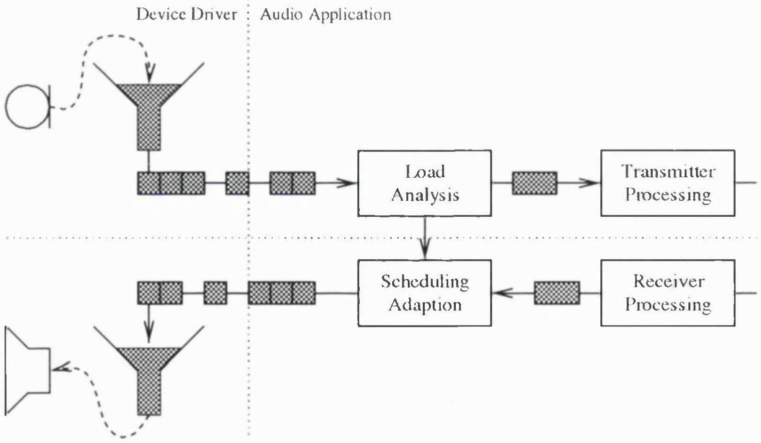


Figure 4.3: Adaptive cushion algorithm.

4.4.1 Measuring workstation state

An estimate of the state of the workstation can be achieved by a simple modification to the structure of the audio application. In the audio tool application loop, instead of reading a block of fixed size from the audio device driver, a non-blocking read is made, and all the stored audio is retrieved. The amount of audio gives an exact measure of the amount of time that has elapsed since the last time the call was made.

A history of the time between successive calls reflects the loading of the workstation. Figure 4.4 shows measurements from a lightly-loaded SUN Sparc 10 workstation over a period of 2 mins. There are readings as often as every 10 ms, which gives enough load samples to be able to monitor load changes as they happen.

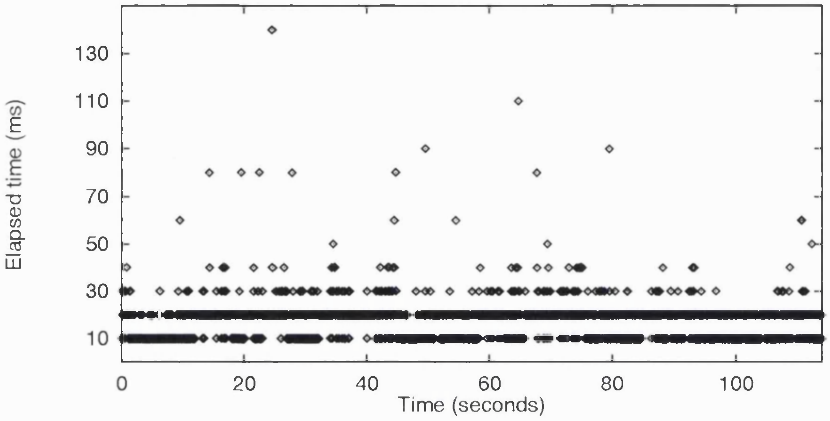


Figure 4.4: Read length variation.

There is an exception event to cater for in the implementation. When the amount of audio that is returned by the read call is equal to the total size of the device driver buffer. It is likely that an overflow has occurred and some input samples have been lost, and therefore the track of time. The audio tool process resynchronises using external mechanisms like the workstation clock. This situation does not happen very often, since the total length of the device driver buffer is configurable, and is usually set large enough to cater for a few seconds of continuous audio input.

4.4.2 Cushion size estimation

Based on the workstation load information, the target fill level for the device driver buffer can be estimated.

Figure 4.5 shows the distribution of the measurements presented in Figure 4.4. The X-axis represents the elapsed time in milliseconds. Y-axis is logarithmic and shows the number of times each different value occurred. It can be seen that the vast majority of measurements are smaller or equal to 40 ms (320 samples). However, the largest measurements are 130 ms long.

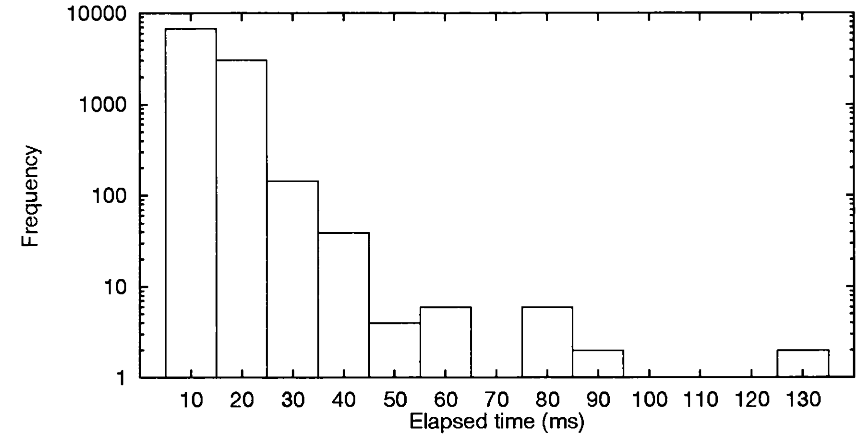


Figure 4.5: Load measurements distribution.

The estimation algorithm should maintain a cushion level that will cater for the majority of scheduling anomalies, without resulting in gaps in the playback. This might be trivially achieved by maintaining output buffer levels at the maximum measured read length, but this will result in excessive delay being introduced. The adaptation algorithm introduced here allows the user to specify how much audio breakup should be traded for a reduction in delay.

The algorithm operates by maintaining a history of past load measurements. After a new measurement, the desired cushion size is estimated based on the recent history. The load measurements are stored in a circular buffer, and to avoid the processing overhead of examining all the logged data each time a decision needs to be taken, a histogram of load measurements is maintained. The histogram is built incrementally. Each time a new load measurement is made, the oldest one in the circular buffer is removed from both the buffer and the histogram, and replaced with the new one. An estimate is made by examining the histogram, and calculating a cushion value that will cover a given percentage of measurements.

Let b_i be the number of load samples of length i and h the length of the history. Then, we have:

$$h = \sum_{i=1}^{\infty} b_i \tag{4.1}$$

If t is the threshold of read lengths which is being catered for, then the cushion c is given by:

$$c = \min x : \sum_{i=1}^{x} b_i > t \tag{4.2}$$

The cushion adaptation algorithm is configurable by adjusting the history length h and by varying the threshold t of load measurements that are going to be catered for.

The desirable behaviour would be to follow the trend in scheduling load and avoid rapid jumps in cushion size due to very short-lived bursts. This can be achieved by increasing the history length, which in effect low pass filters the load information. The increase in the history length, however, is at the expense of fast adaptation since the cushion estimate now depends on older data.

Figure 4.6 shows the estimated cushion for the data presented in Figure 4.4. A relatively small averaging period was used (200 measurements) and therefore, the cushion varies continuously in an attempt to match the changing workstation performance. Figure 4.7 shows the estimated cushion using a larger averaging period (2000 measurements). The estimate is a lot more stable and represents the trend in workstation load. By increasing the averaging period, a reduction in average delay and a small increase in audio gap have resulted.

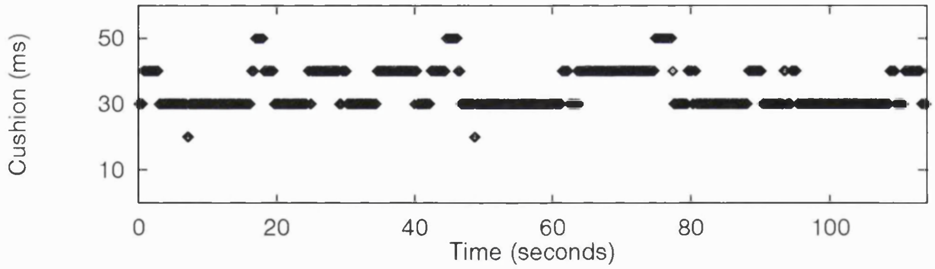


Figure 4.6: Fast adapting cushion.

It is important to note that the algorithm design aims to smooth out short-lived scheduling anomalies, and cannot provide a solution if the workstation cannot (on average) keep up with the requirements of the real-time process. This can be detected if there is a constant trend in cushion increase. In this case, the audio tool should reduce the amount of processing performed (load adaptation).

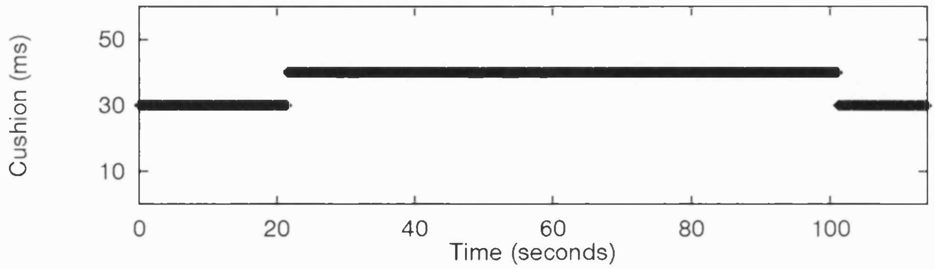


Figure 4.7: Stable cushion.

4.4.3 Buffer adjustment

Maintaining a given amount of audio in the device driver buffer can easily be accomplished. The output buffer is initialised with a given number of samples (cushion size). On each cycle of the program, we calculate how many samples remain in the device driver buffer using the elapsed time given by the read length:

$$\text{samples left in cushion} = \text{cushion size} - \text{read length} \tag{4.3}$$

The cushion is refilled by writing out the required number of samples. An outline of the code that achieves this is given in Figure 4.8.

```
/* Initialise cushion */
write(fd, mixed_audio_buffer, cushion_size);
for (;;)
{
    /* Read all there is (non blocking) */
    elapsed_time = read(fd, input_buffer, MAX_SIZE);
    /* Refill cushion */
    write(fd, mixed_audio_buffer, elapsed_time);

    /* Do all other processing */
}
```

Figure 4.8: Maintaining the cushion size.

When the elapsed time exceeds the cushion because of a scheduling anomaly, then we should not write back as much as we read, because this will effectively increase the cushion size. The cushion should be re-initialised by writing out its full size in this case. To maintain synchronisation between the input and playback operations the extra samples that were not written out are discarded. As these samples correspond to the silence period that resulted from the cushion overrun, no additional disturbance results from us discarding them and the timing relationship in the played out signal is maintained.

4.4.3.1 Varying the cushion size

The methods described here vary the cushion size in line with the desired value. The cushion size reflects the current state of the workstation, which may change in response to other processing on the workstation.

Changing the cushion size during silence periods has minimal effect on the perception of audio [Gruber and Strawczynski, 1985]. It is thus preferable to make any adjustments necessary during such periods.

However, it is possible to alter the cushion size during a talkspurt. A decrease in size will result in a reduction of the end-to-end delay that will be perceived at the end of the talkspurt. This can be achieved by simply writing out fewer samples than those needed to refill the current cushion. At this stage we have to decide what to do with the remaining samples that we did not transfer to the audio device. The simplest solution is to discard them. If the change in cushion size is small enough, as is usually the case, then the missing samples will not be noticed [Gruber and Strawczynski, 1985].

Increasing the cushion size is not as easy as decreasing it, since extra samples are needed. However, there may be extra audio available if the decision to increase the cushion follows a scheduling anomaly (see § 4.4.3). If there is no extra audio available to fill the gap, then it has to be artificially created. Techniques for artificially creating extra audio required during periods of sample loss have been presented in Chapter 3 under § 3.2.1 to solve the problem of network packet loss.

4.5 Discussion of results

The adaptive cushion algorithm has been implemented and tested in RAT. It has produced a perceivable reduction in the end-to-end delay of the system.

A simulator was built to evaluate the performance of the adaptive cushion algorithm. The simulator inputs load measurements and talkspurt information, and uses these to calculate the resulting delay and audio gaps for different adaptation strategies.

In our experiments, we used real load measurements collected from RAT, and modelled talkspurt and pause durations after statistical data given in [Brady, 1968]. The data presented below uses load information collected on a medium loaded Sun Sparc 10 workstation running Solaris 2.4 over a period of 20 mins. Results were collected during a multicast multimedia conference, while transmitting and receiving audio with RAT and moving video with vic [McCanne and Jacobson, 1995].

Table 4.1 shows resulting end of talkspurt delay values for different adaptation conditions. The first row represents operation of the audio tool without the use of the cushion mechanism as is used in all other existing audio tools. The remaining rows represent results from the use of different parameters in the adaptive cushion algorithm. The two values in the column describing the adaptation parameters show the number of load measurements that are required to be covered by the cushion and the length of the history that is kept. The column labelled "Read %" gives the ratio of these two values indicating the percentage of measurements that are expected to be covered by the cushion. For each adaptation method the average, standard deviation and maximum of the delay are given in milliseconds. Table 4.2 shows resulting audio gaps in talkspurts for the same adaptation conditions.

Adaptation	Read %	Avg del	Del σ	Max del
No cushion		43	36	240
195/200	97.5	31	4	70
970/1000	97	30	2	40
1800/2000	90	30	0	30
1970/2000	98.5	32	4	40

Table 4.1: Audio delay results (in ms).

It can be seen that the adaptive cushion algorithm results in improvement to both gap size and delay compared to the standard mechanism used in other audio tools.

4.6 Conclusion

Audio quality in multimedia conferencing is impaired by packet loss and variable delay in the network, and by lack of support for real-time applications in today's general

Adaptation	Read %	Avg gap	Gap σ	Total gap
No cushion		34	43	14340
195/200	97.5	21	47	8760
970/1000	97	21	48	9010
1800/2000	90	22	48	9090
1970/2000	98.5	20	46	8520

Table 4.2: Audio gap results (in ms).

purpose workstations.

This chapter has presented an adaptive cushion algorithm that copes with the problems presented to real-time audio conferencing applications by a general purpose operating system. The continuity of the audio signal is ensured during scheduling anomalies by using the buffering capabilities of the audio device driver. The algorithm also restricts the amount of audio in the output audio device buffer to minimise the end-to-end delay of the conversation. Negligible overhead in processing power is incurred with the adaptive cushion algorithm since a history of workstation load is built up incrementally over time.

The results presented in this chapter show that there is a significant improvement in both minimisation of delay and audio gap size to be obtained from using the adaptive cushion algorithm.

Chapter 5

Lip Synchronisation for use over the Internet: Analysis and Implementation

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, are likely to require a greater degree of both lip synchronisation and good quality audio in order to be effective [Watson and Sasse, 1996]. Lip synchronisation can be provided by artificially delaying either the audio or the video output, so that words and lip movements are matched in time.

Causes for the different delays between capture and presentation experienced by the audio and video streams are the variable network propagation times (Chapter 2, Section 2.1), and the different requirements in processing at the transmitting and receiving workstations. The network component of the difference could be removed by combining at the transmitter temporally related audio and video information in the same packets. However, the transmission of each media type as a separate stream adds significant flexibility to the Mbone conferencing architecture. Different paths and resources can be used within the network for each media, allowing preferential treatment for the more critical information[1]. Demultiplexing of information at the receiver at the lowest level possible by the operating system (UDP), improves processing efficiency, and

[1] The mrouted daemonm used for multicast routing, preferentially treats a certain UDP port range that is usually used for audio streams in conferences.

allows each media to be handled by a specialised processing component.

Existing Mbone audio tools, such as vat [Jacobson and McCanne, 1992] and nevot [Schulzrinne, 1995], and existing Mbone video tools, such as vic [McCanne and Jacobson, 1995] and ivs [Turletti, 1994], do not provide support for lip synchronisation. The Robust-Audio Tool, described in Chapter 3, together with a modified version of vic [McCanne and Jacobson, 1995] provides the first implementation of lip synchronisation over the Mbone. RAT and vic use the proposed IETF standard real-time protocol, RTP [Schulzrinne *et al.*, 1996], to provide the necessary functionality required for multicast voice and video communication. The lip synchronisation mechanism developed for RAT uses the time-stamp information provided by RTP to identify the required artificial time-lag.

This chapter 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.

5.1 Background

Good quality packet audio systems must provide packet loss protection and low end-to-end delay [Hardman *et al.*, 1995]. The requirement of low delay means that a packet audio system must launch small packets frequently, rather than big packets less often [Hardman *et al.*, 1995]. Typical packet sizes used for audio over the Internet vary from 20 to 80 ms in length.

Mbone video tools, such as vic [McCanne and Jacobson, 1995], typically provide slow-scan video (2–10 fps), rather than the broadcast rate (25 fps), 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, leading to 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 [Casner, 1994]. Routers operate on a first-in first-out basis and statistically multiplex traffic from different sources. The impact of this behaviour 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 5.1). The mechanism is adaptive, since jitter on the Mbone can vary significantly (see Chapter 2, Section 2.1).

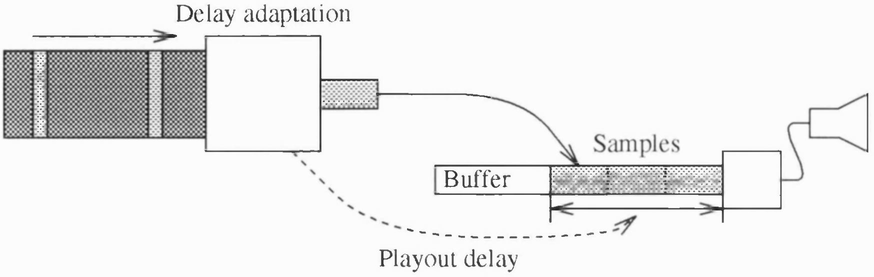


Figure 5.1: Playout delay adaptation to remove network jitter.

The artificially added delay in the reconstruction buffer is calculated from the time that packets take to traverse the network [Montgomery, 1983; Ramjee *et al.*, 1994; Jacobson, 1994]. 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 [Montgomery, 1983]. Consequently, audio tools aim to receive 99.9 % of the packets, since the delay characteristic show a long tail for packets that are abnormally delayed [Schulzrinne, 1995; Schulzrinne *et al.*, 1996].

The effect of network delay jitter on video streams results in jerky rendering of frames, a degradation which can be tolerated by users. Current video tools, such as vic [McCanne and Jacobson, 1995], nv [Frederick, 1994], and ivs [Turletti, 1994], display frames as soon as they are received and decoded without any attempt to remove the network induced artifacts.

5.2 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 80–100 ms is below the limit of human perception [Jardetzky *et al.*, 1995; Escobar *et al.*, 1991; Lamont *et al.*, 1996].

The provision of synchronisation delay for the audio stream can be achieved simply by increasing that provided for voice reconstruction (Chapter 3 Section 3.1.4). 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.

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 co-ordination 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 (Figure 5.2).

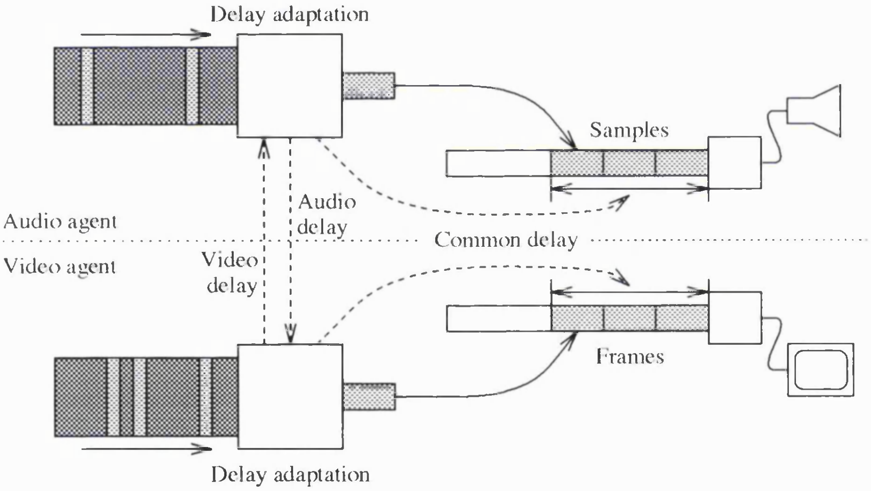


Figure 5.2: Playout co-ordination to achieve synchronisation.

Inter-media communication is needed in order to transfer the desired playout delays between the audio and video applications. Co-ordination can be facilitated by using a local conference bus such as the Conference Control Channel Protocol developed at UCL [Handley *et al.*, 1995] or the LBL conference bus [McCanne and Jacobson, 1995]. 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.

5.3 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;
- Co-ordination facilities;
- Desired playout point calculation.

5.3.1 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 by employing the buffer control techniques described in Chapter 4.

5.3.2 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 5.3):

- 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.

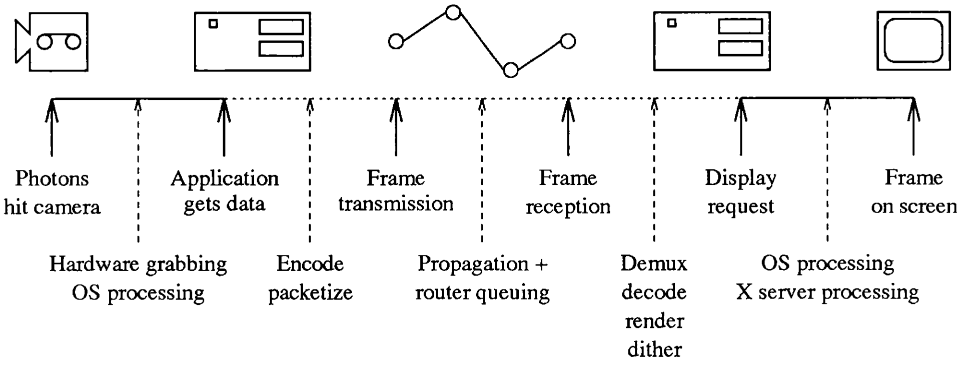

Photons hit camera
Application gets data
Frame transmission
Frame reception
Display request
Frame on screen
Hardware grabbing OS processing
Encode packetize
Propagation + router queuing
Demux decode render dither
OS processing X server processing

Figure 5.3: 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 § 5.3.5.2). 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 [Schulzrinne *et al.*, 1996]. 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.

5.3.3 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.

After decoding and rendering, frames can be placed in a playout buffer and displayed at a pre-calculated playout point. Figure 5.4 shows the sequence of operations in this arrangement. Unfortunately due to the way in which 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[2].

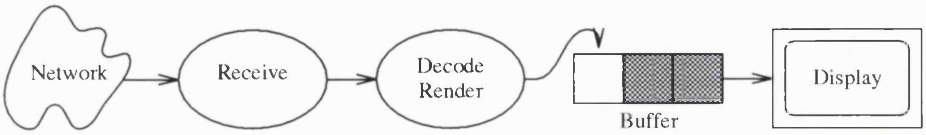


Figure 5.4: Video tool receiver with playout buffer before display stage.

In order to reduce the amount of processing time required, a novel architecture was used, similar to that developed for RAT [Hardman *et al.*, 1996]. 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 5.5. The design assumes that the decoding and rendering of frames is known in advance and can thus be incorporated in the playout delay calculation. Most video encoding algorithms encode the differences between video frames[3] [H261, 1993; JTC1/SC2/WG11, 1990], which means that frames from a period containing significant change in the image will take longer to encode and decode. Results from measurements presented in Section 5.4.1 demonstrate that in a videoconferencing scenario, the decode

[2] Because only the difference between video frames is encoded in transmitted packets, the incremental changes after decoding have to be applied to a shared buffer. If the decoded frame is to be stored in a playout buffer, then the frame has to be copied once. The same buffer used in decoding is also shared with the X server of the workstation and after a frame is decoded a request for display is sent to the X server without any further need for data transfer. If a playout buffer is used then the stored frames have to be copied back to the X server shared memory before display thus requiring a total of two copies of the full uncompressed frame.

[3] This is certainly true for all the encodings available in Mbone video tools.

and render times vary only slowly with respect to time and can thus be predicted based on past history. The time estimated for decoding and rendering is added to the playout point.

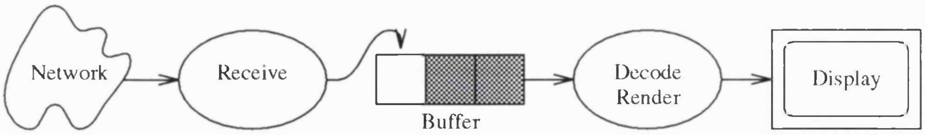


Figure 5.5: Video tool receiver with playout buffer before decoding stage.

5.3.4 Media Co-ordination Facilities

RTP headers are used for the transmission of audio and video packets across the Mbone (Figure 5.6). 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, and time-stamps are measured in audio samples [Schulzrinne, 1996]. 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 [H261, 1993] is based on a 90 kHz clock [Turletti and Huitema, 1995].

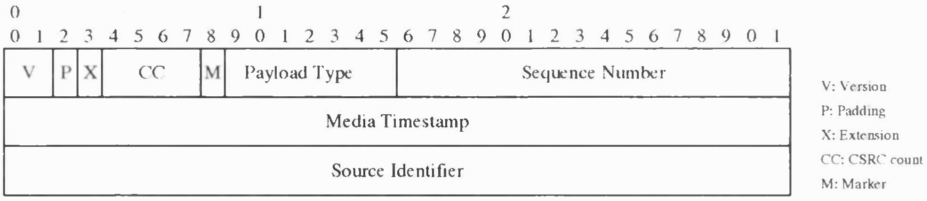


Figure 5.6: Real-time transport protocol header.

Playout delay co-ordination between the audio and video processes needs to be in a common format, and with a common reference clock. RTCP control messages (Figure 5.7 are transmitted alongside each data stream, and provide individual mapping between the media and Network Time Protocol (NTP) time-stamps [Schulzrinne *et al.*, 1996; Mills, 1992]. The NTP time-stamps therefore give a common reference to the transmitter workstation clock, which can be used to provide inter-stream synchronisation. Figure 5.8 shows the exchange of RTP and RTCP packets across the network and synchronisation information over a local conference bus, between multiple conference participants.

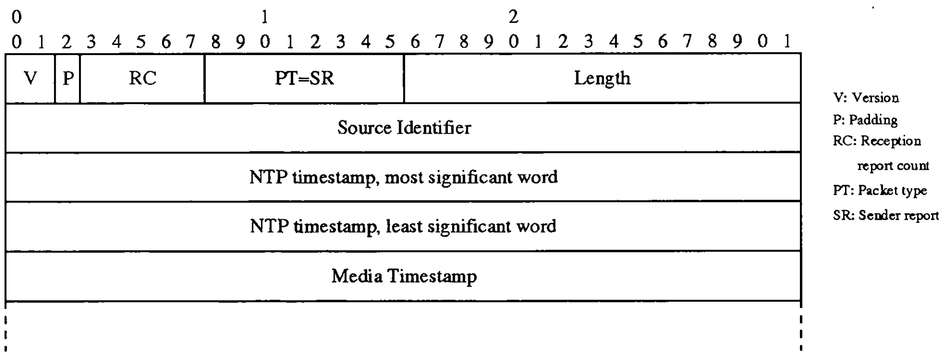


Figure 5.7: RTP control protocol (RTCP) packet.

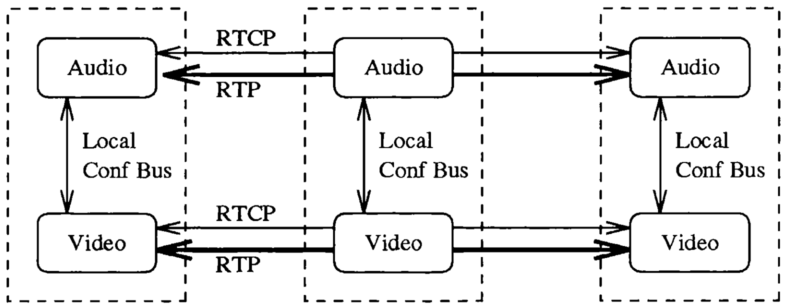


Figure 5.8: Media agent communication.

As explained earlier, the audio and video tools announce their required playout delay over the conference bus. Delay co-ordination 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 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 audio conferencing interactivity. An enhanced scheme can be envisaged, where each application announces an acceptable range of delays to a co-ordinating agent, which then selects the delay to be enforced (Figure 5.9).

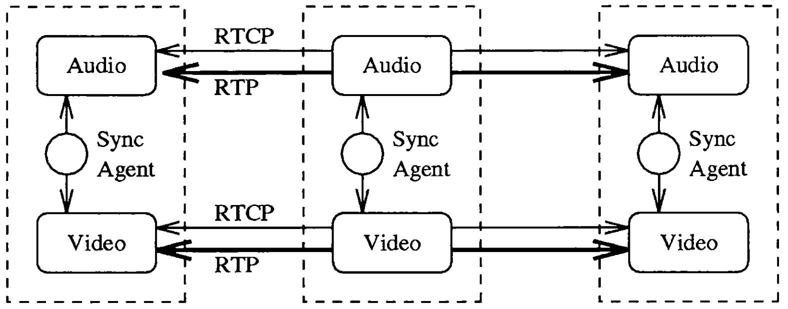


Figure 5.9: Media agent communication with co-ordination agent.

5.3.5 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.

5.3.5.1 Clock Drift

The stability of a clock is how well it can maintain a constant frequency and the accuracy how well its time compares with UTC. Drift of some degree occurs for all clocks and is measured as the first derivative of frequency [Mills, 1995]. The playout adaption algorithms in media tools allow for differences in clock drift between sender and receiver clocks, 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 co-ordination is accomplished using media specific time-stamps from these clocks transmitted in RTP media packets. Clock drift occurs for both reference points, but can be allowed for, since the mapping of both audio and video media timers to the transmitter workstation clock is periodically updated through RTCP session messages.

5.3.5.2 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 [Faller, 1992]. 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 [Hagsand and Sjodin, 1994; Khanna *et al.*, 1992; Mullender *et al.*, 1994; Fisher, 1992]. However, as dynamic real-time scheduling is NP hard [Stankovic *et al.*, 1995], 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 timing on the receiving workstation becomes less accurate as the load on the receiving workstation increases [Fall *et al.*, 1995] making the scheduled display of video frames less accurate. 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.

5.4 Implementation Assessment Results

The efficient design of the synchronisation mechanism described above relies on a predictable decode and render time for frame updates. Results supporting this are presented first. The main results which follow assess the subjective performance of the lip synchronisation implementation through a human perception experiment.

5.4.1 Video Decode and Render Times

Current video coding algorithms in Mbone tools encode the changes between one frame and the next to provide high degrees of compression. In videoconferencing the video image consists of a person moving in front of a fixed background. Thus the amount of data that needs to be encoded between one frame and the next is the area of the person that has moved. The following results show that it is reasonable to assume that video decode and render times vary only slowly. The first graph (Figure 5.10) 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 the differences between frames are significant.

The decode and render times over 500 frames were collected using vic [McCanne and Jacobson, 1995] 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

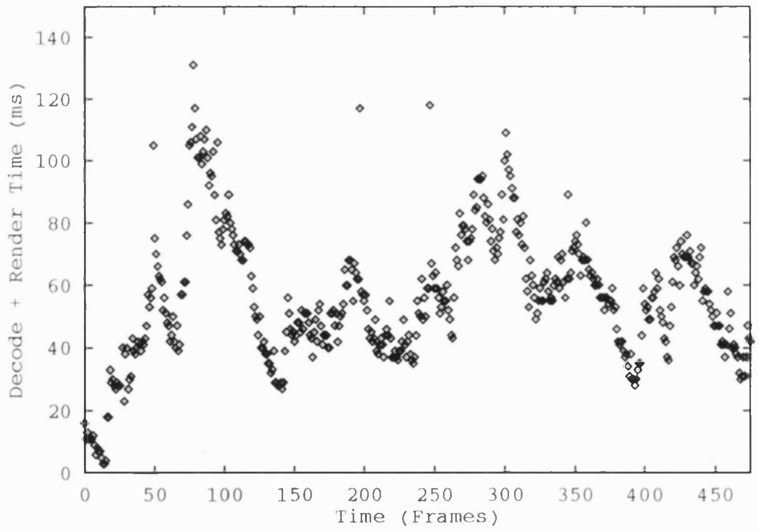


Figure 5.10: Decode and render times for H261 video frames.

frames extend over a period of 50 s 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 ms, but between frames by a much smaller amount.

Figure 5.11 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 ms 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 20 ms based on the time achieved for the previous frames. The information can be used to correctly schedule the presentation time of each frame. It is expected that the 20 ms jitter, associated with the frame presentation time, will produce skew that is still beyond the limits of human perception [Jardetzky *et al.*, 1995; Escobar *et al.*, 1991; Lamont *et al.*, 1996].

5.4.2 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 fps. The performance was measured using a 5 point rating scale, where 5 is rated as synchronised, and 1 as unsynchronised.

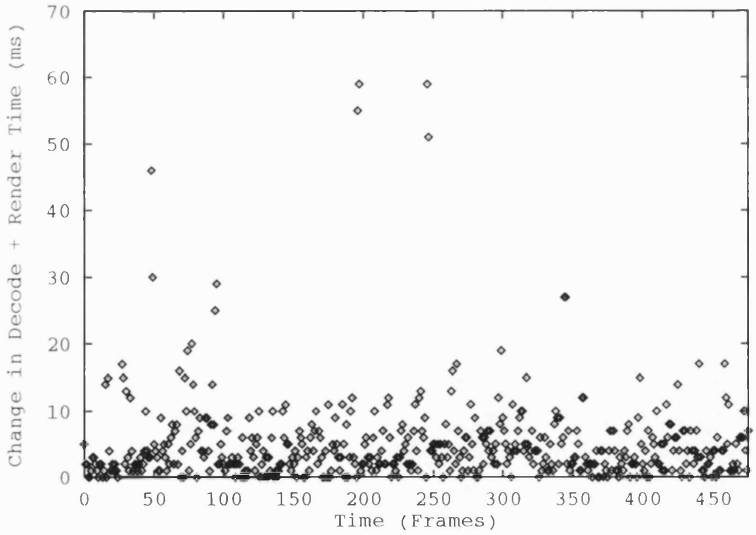


Figure 5.11: Change in decode and dither times between frames.

The experiments were performed on a pair of Sun SPARC 10 workstations, using H.261 video coding, and CIF sized frames.

The results, shown in Figure 5.12, indicate that audio and video is not perceived as being synchronised for frame rates of less than 5 fps. This was expected as the inter-frame gap for such low frame rates is far greater than the allowed skew between the media, making it hard to provide a satisfactory result. For higher frame rates the graph shows that synchronisation was achieved. The results also support the earlier conclusion that the jitter associated with video frame presentation times is below the limit of human perception.

5.5 Conclusion

This chapter 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 co-ordination 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

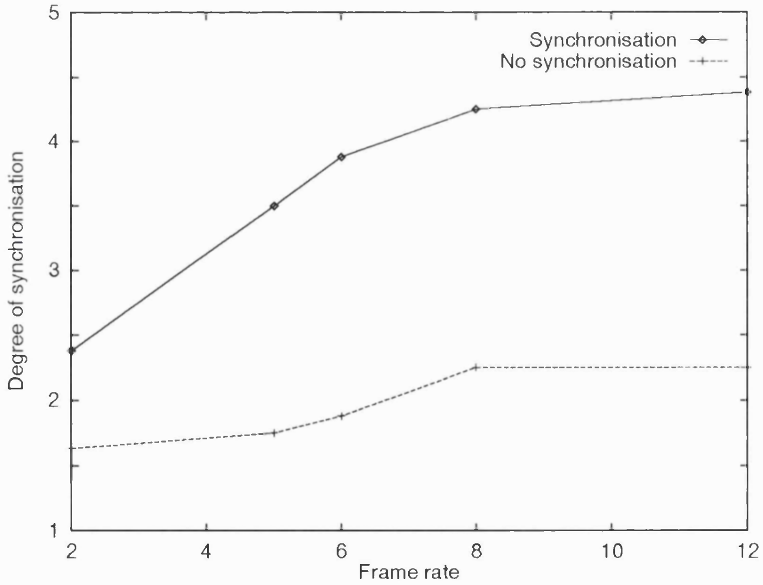


Figure 5.12: Subjective assessment of synchronisation.

design of the system. The results show that the efficient implementation is good enough to provide lip synchronisation for use over the Mbone.

Chapter 6

Network Adaptive Continuous-Media Applications Through Self Organised Transcoding

So far in this thesis we have looked at the ways in which audio conferencing quality is affected by the Internet environment and presented techniques that can be employed by end-host applications to work around problems. Adaptive buffering techniques, to smooth network delay jitter, and transmission of audio redundancy, are offered to counteract the effects of the best effort network service. End-system problems caused by the interaction with a general purpose operating system were tackled. The feasibility of synchronised presentation of audio when combined with other media was demonstrated.

The above techniques are equally well applicable in a two-way conversation and in a multi-way conference. This is an achievement of the simplicity of the multicast service model, that allows the application to transmit to an unknown receiver group using the abstraction of a multicast address. Although this offers a single handle for a potentially widely distributed group, it hides the fact that under the currently available best effort network service, due to varying link capacity and load, receivers in the group are going to experience different end-to-end delay variations and packet loss rates. The network scale and heterogeneity in available bandwidth complicate the design of network adaptive real-time multipoint applications.

The UCL Robust-Audio Tool (RAT) (presented in Chapter 3) and Freephone developed at INRIA use forward error correction (FEC) techniques [Bolot *et al.*, 1997] and successfully address the loss problem with minimal increase in stream delay. FEC used, is in the form of highly compressed low quality audio, which is piggybacked on normal audio packets. The decision on the level of FEC to use is made per source, based on receiver loss reports, and is tailored to cover for the average or highest requirements of the receiver group. This strategy is only good for a group observing similar loss rates. In a diverse group receivers observing low loss are forced to receive useless redundant information, whereas receivers with very bad loss may not be covered.

The variable network loss rates and perceived quality in different areas of a multicast distribution tree are a result of different link bandwidth availability and link load. The extent of this problem is best illustrated by the work of Handley [Handley, 1997a] in Figure 6.1. The graph represents the packet loss rates experienced by different receivers of a popular Mbone session over a period of the session. The loss measurements were collected from RTCP receiver reports (Chapter 2, Section 2.3). Each line in the graph is the loss observed at a particular receiver. It is clear that although most receivers are seeing low to moderate loss, there are a small number of sites suffering.

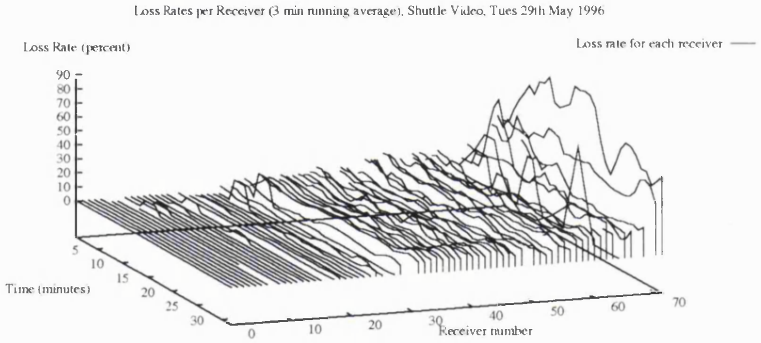


Figure 6.1: Loss rate against time for different receivers.

This indicates that a single stream addressed to the whole group cannot possibly cover the needs of all receivers. Instead the data rate and amount of redundancy has to be customised and separately distributed to problematic areas. It has been shown that sender driven schemes, that try to address the receiver heterogeneity problem, do not scale to large groups. Any scheme that attempts to separately cover for the different needs of problematic receiver subgroups has to be receiver driven to scale [McCanne *et al.*, 1996]. Subgroups of co-located receivers in a multicast delivery tree, suffering from similar problems, should co-ordinate their efforts in improving reception quality. Furthermore their attempts should not affect reception for the remaining participants in the multicast session.

Strict low delay requirements of real-time data distribution preclude solutions using retransmissions to achieve required reliability. The dynamic nature of the Mbone delivery and membership model does not allow for manually configured static schemes that work around congested links.

The solution we present in this chapter uses a self-organisation scheme to form groups out of co-located receivers with bad reception. A representative of the group is responsible for locating a suitably positioned receiver with better reception that is willing to provide a customised transcoded version of the session stream. The transcoding site thus provides local repair to the congestion problem of the group, with minimal increase in stream delay. The data rate and redundancy level of the transcoded stream are continuously modified to adapt to the bottleneck link characteristics, using reception quality feedback from a member of the formed loss group. Network friendly congestion control of the real time multicast stream can thus be achieved.

The rest of this chapter is structured as follows. Section 6.1 describes related work on congestion control for multicast distribution and attempts to solve the group reception diversity problem. In Section 6.2 we describe our self-organised transcoding solution to the problem. The proposed solution has been implemented and evaluated through simulation using the VINT network simulator [McCanne and others, 1997]. The simulation results can be found in Section 6.3.

6.1 Background and Related Work

The current multimedia conferencing architecture over the Mbone / Internet, described in Chapter 2, has the following characteristics:

- Conferencing applications use the Real-time Transport Protocol (RTP) [Schulzrinne *et al.*, 1996; Schulzrinne, 1996] to transmit information over an unreliable best-effort multipoint network.

- Receivers express interest in receiving traffic by tuning into a multicast address and the network forwards traffic only along links with downstream recipients.

- No knowledge of group membership or routing topology is available at the source or receivers.

The rest of this section discusses existing work and alternatives addressing the reliability issues for continuous media streams in a heterogeneous multicast environment.

6.1.1 Layered Encoding

McCanne *et al.* [McCanne *et al.*, 1996] combine a layered compression scheme with a layered transmission scheme to address the network heterogeneity issue. The media stream is encoded into a number of layers that can be incrementally combined to provide refined versions of varying quality of the encoded signal. The individual layers are then transmitted on separate multicast addresses. Receivers adapt to network conditions by adjusting the number of levels they subscribe to, and thus improving perceived quality, by trading off average signal quality for packet loss (Figure 6.2). The application they propose for this scheme is multicast video. For this purpose they have developed a video codec that can compress a video frame providing very fine granularity layers.

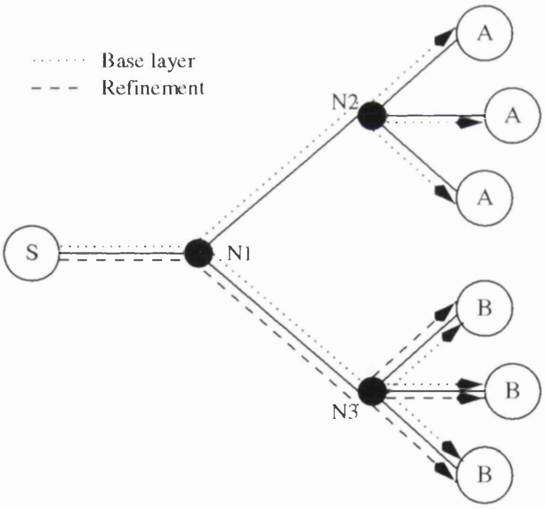


Figure 6.2: Example of layered encoding and transmission.

Although layered encoding is possible with audio, the transmission bandwidth range is significantly more restricted in comparison to that available to video applications. This is definitely the case for sampled speech. There is none or very little improvement in intelligibility to be gained by sampling speech at full CD quality (44.1 kHz stereo sampling) rather than a single channel sampled at 16 kHz [Rabiner and Schafer, 1978]. With music a wider range is available, from CD quality to a highly compressed low quality format.

Currently, available speech codecs do not render themselves naturally to layering. It is possible to modify a scheme to split up the resulting compressed block into a number of sub-blocks that can be separately decoded to provide increasing levels of quality. However, to achieve the same quality that the original not split up version of the codec provides, a larger number of bits per codeword is required [Sherif *et al.*, 1993].

An additional requirement from a layered encoding scheme, in order for it to be suitable for use with a network adaptation algorithm, is that there must be an exponential relationship between the bandwidth of different layers. Such an arrangement maximises support for network bandwidth adaptation while keeping the routing overheads due to the number of multicast groups used low. This requirement in combination with the problems listed above makes layered transmission unsuitable for real-time audio steams.

Even with a video stream depending on the application there is a target frame rate from which you cannot deviate. In video conferencing the entire range can be used, from very slow scan video to the full potential of the camera, but when watching a film full frame rate is required.

6.1.2 Simulcasting

With simulcasting a group of receivers can adapt to network conditions by having the sender transmit a new parallel stream and customise it to match their requirements. The new stream can use a different compression scheme, to reduce the bandwidth required, and employ some form of FEC to counter packet loss. This approach is likely to create congestion on links that are close to the sender, as all simulcasted streams will have to traverse them.

The bandwidth utilisation advantage of layered encoding over simulcasting is illustrated in Figures 6.2 and 6.3. Due to different bandwidth availability on links N1-N2 and N1-N3 groups of receivers A and B require different streams. With simulcasting the link between the source and N1 has to carry both the full bandwidth stream for receiver group B and a lower bandwidth stream for group A. Using layered encoding and transmission, the link between the source and N1 does not carry duplicate redundant information. Ideally, the sum in bandwidth of both layers will exactly cover the requirements of group B whereas the base layer will be customised for group A.

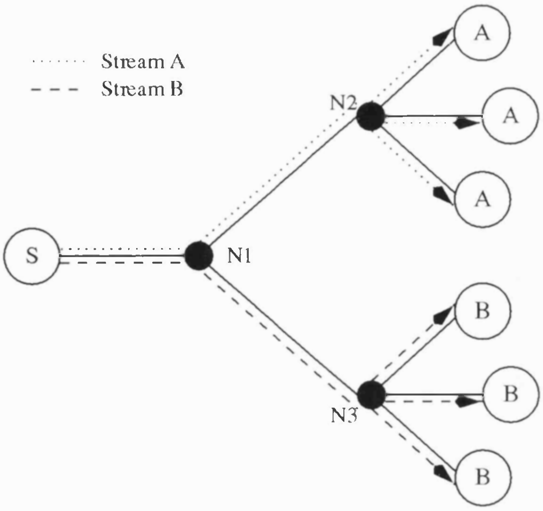


Figure 6.3: Example of simulcasting.

Simulcasting suffers from scalability problems because the sender is involved in the adaptation.

6.1.3 Retransmission Based Reliability

Proposals exist for integrating reliable multicast schemes into audio and video applications so that missing packets can be recovered from neighbours with better reception [Pejhan *et al.*, 1996; Maxemchuk *et al.*, 1997; Xu *et al.*, 1997]. This is achieved by trading off quality for delay, as any reliable multicast protocol has to request retransmission and wait for the repair. Although this may be acceptable in a real-time lecturing scenario, it becomes less useful with interactive communication. An additional undesirable side effect is that the operation of the reliable protocol creates extra control traffic.

Maxemchuk *et al.* [Maxemchuk *et al.*, 1997] propose a hierarchy of retransmission servers positioned around expensive or over utilised links. The servers operate a negative acknowledgement (NACK) based reliable protocol between them, and receivers use a similar scheme for requesting lost packets. Their proposal significantly improves reception quality but requires manual configuration of the retransmission servers.

In [Xu *et al.*, 1997], the authors describe the STORM protocol that develops parent child relationships between participants of a multicast using an expanding ring search technique. Parents are chosen according to loss statistics, so that they have a good chance of receiving packets their children are likely to request.

Streaming of stored data makes little sense unless browsing and selective playback is a requirement. For totally non real-time scenarios, a normal transport protocol and pre-fetch can be used to achieve perfect audio quality. TCP can be used in a single user scenario or a multicast congestion control protocol like RLC [Vicisano *et al.*, 1998] for multiple recipients.

6.1.4 Statically Configured Transcoders

In many situations where a group of people with limited network resources want to participate in a high bandwidth multicast conference, the use of transcoders is employed. A transcoder is an application that is placed at the far end of the low bandwidth link, to down-convert a high bit rate stream, so that it can fit through the link (Chapter 3, Section 3.3). Transcoders for video can reduce the frame rate and image quality, and audio transcoders can re-encode the audio signal using a higher compression scheme. A feature of audio transcoding is that it adds minimal delay to the signal when relaying it, as it can be done on a per packet bases. Apart from changing the bandwidth requirements of a stream, a transcoder can also introduce or remove forward error correction information to counter packet loss.

In a multicast scenario, a transcoder can be positioned on the sender's end of a problematic link, to re-encode the stream to use lower bandwidth and add FEC information. The resulting stream can be re-multicast to a new address. If all receivers beyond that link tune to receive the new customised stream, then there will be no bandwidth wasted, as the original stream will no longer traverse the problematic link. This solution is static and has problems with the dynamic nature of Mbone multicast routing.

In [Pasquale *et al.*, 1993], the authors propose the use of self-propagating filters over a dissemination tree. Leaf nodes specify to the node above them filters that can convert an incoming stream to match their requirements. When a non-leaf node has multiple output links with similar filters, the filter is propagated to a node higher up the tree. This scheme can achieve optimal network utilisation with minimal processing, but requires full knowledge of the distribution tree topology and processing capabilities at each node.

6.2 Self Organised Transcoding

By combining the simulcasting, local repair and transcoding schemes we have developed a solution that does not suffer from the above problems. What is needed, is a control scheme that automatically configures transcoders within the multicast tree to support branches with bad reception.

When a group of receivers detect loss, caused by a congested link, an upstream receiver with better reception at the far end of the bottleneck link affecting the group, needs to act as a transcoder and provide a customised version of the stream. This new stream will be multicast to a different address, to which receivers affected by the bottleneck need to switch. To achieve this in a scalable way, the suffering receivers need to elect a representative, which will attempt to locate an upstream receiver willing to serve them and co-ordinate the transcoding process (Figure 6.4).

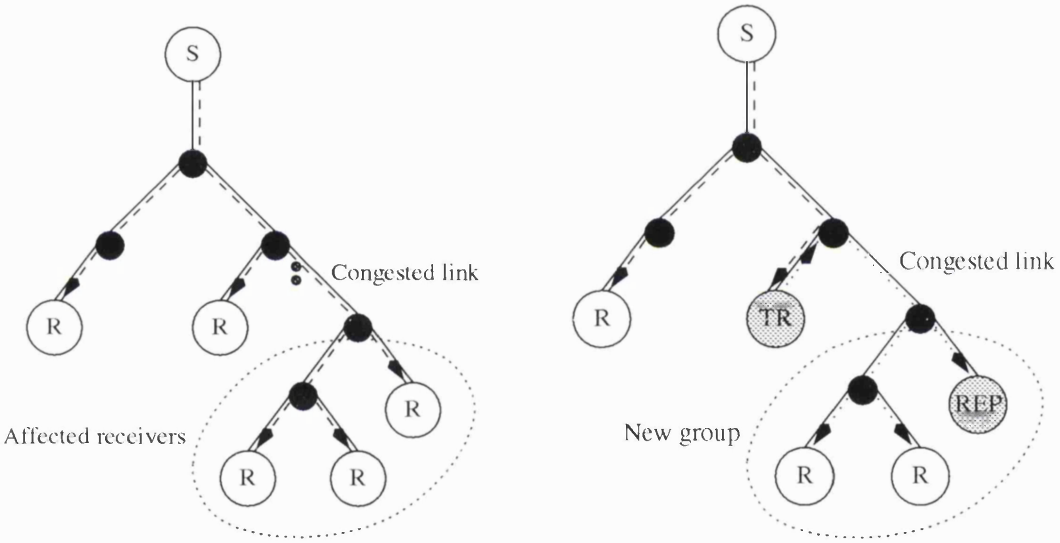


Figure 6.4: Transcoder requester and provider configuration around a congested link.

Ideally there should be no links carrying more than one of the transcoded streams (including the original encoded stream). To achieve this the following conditions must hold:

- The transcoder servicing a sub-tree should be as closely located to the sub-tree as possible. The preferable location is at the end of the bottleneck link closer to the source.

- A group of receivers behind a bottleneck link has to be co-ordinated in its actions. All the receivers responding to congestion have to switch to a new transcoded stream at the same time.

To avoid the need of processing capabilities at nodes of the tree that do not have any receivers, we co-locate the transcoders with active receivers and allow the users media tool to execute our protocol. Although this makes our proposal applicable on the currently available network infrastructure, it results in sub-optimal transcoder configurations. In the example of Figure 6.4 the ideal placement for the transcoder would be on the network node above the requesting site. The current placement of the transcoder results in wasted resources for the transcoded stream out of the transcoding site. If the transcoding site was connected to the network through a shared medium link, like an ethernet, then both the incoming and outgoing stream would have to traverse the same link possibly causing congestion.

6.2.1 Transcoding Provider Selection

When a receiver detects packet loss, it schedules a transcoder request message. To avoid multiple receivers that see the same loss simultaneously sending a request and overloading the network, requests are multicast and sending them is delayed by a time, proportional to the distance of the requester from the stream source plus an additional small random interval. If a receiver sees a request while it has one scheduled, then the request is cancelled.

The request includes a description of the loss patterns observed. Receivers of the request that have better reception from the requester can offer to provide a transcoded stream. This is achieved in a similar manner to the request. The response message is scheduled to be multicast after an amount of time proportional to their distance from the requester plus a small random interval. The response message contains a description of the loss experienced by the offering receiver and its distance from the requester. On reception of the offer, other receivers that have offers scheduled suppress their messages, unless they can provide a better service.

The quality of service that a site offering a transcoded stream can provide, is calculated as a function of the difference in loss rates observed by the requester and the offering site, and the distance between the two. The quality is better for larger loss rate differences and smaller distances.

After a short timeout period, long enough for the offers from potential transcoders to arrive, the requesting receiver multicasts a transcoding initiation message. This message serves two purposes. It notifies the offering receivers of who is going to provide the transcoded stream and instructs all other members in the loss group to switch to the new stream.

As control messages are sent to the whole receiver group, while a transcoding negotiation is in progress, control messages from other receivers are suppressed.

6.2.1.1 Receiver Distance Calculation

The timer based-scheme described above is similar to that used in the SRM [Floyd *et al.*, 1997] reliable multicast protocol for retransmission requests and repairs. In SRM round-trip times (RTT) are used as distances between receivers and are calculated from timestamps in session messages. Reporting receivers include timestamps received from other receivers plus the amount of time elapsed between receiving the stamp and sending the report. Round-trip time estimates can thus be obtained. RTP uses the same scheme with timestamps in RTCP messages just for sources, so that they can calculate the RTT to receivers. The SRM extension to obtain distance estimates between all the receivers does not scale for large sessions, since every pair of receivers has to exchange at least three messages. Puneet et al [Sharma *et al.*, 1998] have developed a hierarchical self-organising scheme, that elects top level receivers for different regions of the distribution tree, which represent their region in distance calculation estimates. This scheme significantly improves the scalability of RTT calculation in SRM.

The requirement for background session messages to build distance information is removed, if receivers use NTP [Mills, 1996] and have synchronised clocks. Although the level of deployment of NTP on current Mbone hosts is not very good, there is no reason why it should not be in use. With synchronised clocks a distance estimate can be obtained from a single timestamped message. A receiver of a request or an offer can calculate the distance from the sender by subtracting the timestamp in the message from the current time. This estimate may not be very accurate for the reverse distance from the receiver to the sender, as paths in the Internet are not always symmetric, but for our purposes it is good enough.

6.2.1.2 Multiple Offer Resolution

Depending on how the delay timers are set it is likely that the requester will receive more than one offer. Some of these may even not have originated from another receiver further up the delivery tree from the source, but by a receiver on a side branch with a better link to the requester. A control protocol can be used in such cases to measure the performance of different links and decide on which one to use.

6.2.2 Receiver Group Control

In order to reduce the amount of traffic flowing through bottleneck links when a transcoded stream is initiated, all receivers behind that link should stop receiving the original stream. The transcoding initiation message provides the synchronisation needed to co-ordinate the switching action. Receivers of the initiation message decide individually whether they belong to the group, and accordingly switch to receiving the new stream or continue without change.

The decision on whether a receiver belongs to the same loss group as the requester is based on correlated loss information between the two. The main principle behind this, is that receivers behind the same congested link will miss the same packets and see similar loss patterns.

The current Mbone media tools implementing the RTP protocol provide periodic loss measurements in RTCP receiver reports. These reports are multicast so that all receivers see reports from other receivers. By correlating the variations in loss between what is reported and what is observed locally, some grouping information can be derived. The problem with RTCP reports is that they become very infrequent, as the size of the receiver group grows, to maintain the amount of bandwidth used by control traffic small. Furthermore the period of time over which the loss is reported is not obvious. A guess at the reporting interval can be made, as the last correctly received packet is given in the report. By monitoring the frequency of reports from each receiver, we can figure out the period over which the report is referring to. The estimation process is complicated by lost report messages. To perform the correlation the loss observed locally over the same period has to be calculated. To achieve this, a history of arrival timestamps and sequence numbers of correctly received packets has to be maintained. The history has to be long enough to cover the maximum possible reporting interval. Apart from the excessive amount of storage this method requires, the results produced cannot be very accurate.

The need for background control messages to exchange locality information can be removed, by including a description of the loss pattern observed by the requester in the transcoding initiation message. The loss description can be in the form of a bitmask, representing which of the last transmitted packets were received and which not. The sequence number of the last packet in the bitmask can also be included. Other receivers can use the information in the bitmap, to correlate the loss and decide if they belong to the group or not. This alleviates the problem of infrequent RTCP reports in large sessions, as even receivers that have not had a chance to send a report will know if they belong to the group.

To perform the correlation each receiver must maintain a history of received packets. The size of the state can be as little as one bit per packet as all we are interested in is whether a packet was received or not. The length of the history does not have to be much longer than the length of the loss bitmaps as the loss data in a received initiation message will always be recent.

A received bitmap can be compared with the local log to provide a loss proximity measure. Packets that are lost in both sites, increase the likelihood that the receivers are located close to each other. Packets lost at one receiver but not at the other reduce the likelihood. The accuracy of the result can be improved by increasing the size of the bitmap at the expense of larger control messages. An optimisation would be to use a Huffman encoding for the bitmap. This way information about more packets can be packed in the same space in the message. The encoding method used can vary and be optimised for different loss rates.

The bitmap control scheme requires less state and processing and provides much more accurate results to the RTCP loss variation correlation.

6.2.2.1 Loss Bitmap Comparison

As the result of the loss bitmap comparison determines the stability of Self Organised Transcoding (SOT), it is crucial to minimise decision errors. Errors can have two outcomes:

- A receiver can decide to join a group it doesn't belong in, thus pulling the transcoded stream to some remote network location.

- A receiver that should join the transcoded group, does not do so, and the original data stream continues to flow down the bottleneck link.

Thus the decision process has to be as precise as possible and cannot be either over optimistic or under optimistic.

SOT uses three summary measures when comparing the loss bitmaps from two different receivers. These measures are the number of packets *lost in common*, the number of packets *lost at one receiver* but not the other and the overall *loss rate*. The overall loss rate is calculated as the average between the rates at the two sites, as it is assumed that in order for two bitmaps to match the two rates have to be similar.

In order for the comparison to be useful, some information has to exist in the bitmaps. A pattern full of losses is not useful, as it will be the result of a broken link. Two such patterns although identical could be the results of two different broken links. The same holds for a pattern full of received packets. In contrast, we can be almost certain that two identical bitmaps with a number of transitions between lost and received packets are the result of the same problematic link.

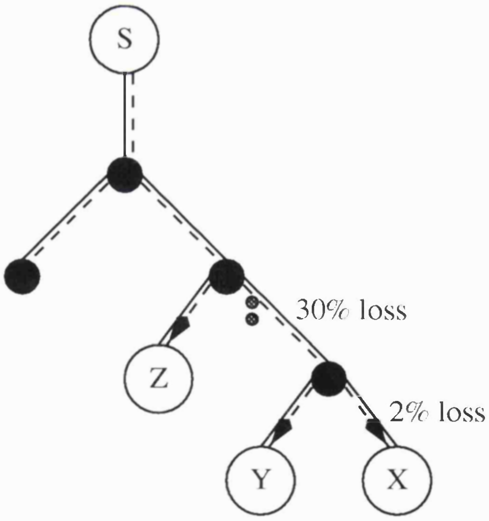


Figure 6.5: Receivers with slightly different loss affected by the same main bottleneck.

In some cases, it may be desirable to accept a small number of not common losses, in order to form groups of receivers that are affected by a common major bottleneck but also have smaller problems of their own. An example can be seen in Figure 6.5. Receiver X can be grouped together with Y and have a single stream provided by Z. The stream can be customised to cover for the additional low loss on the link to X at the expense of having the link to Y slightly under-utilised. The alternative configuration would be to have Z transcode a stream for Y and then Y transcode a stream for X. There is a trade-off between the amount of separate losses that we should allow in the comparison to achieve such configurations and the probability of error in the decision.

6.2.3 Congestion Control

The transcoder initiation protocol results in a natural pairing between requester and transcoding provider on either side of a bottleneck link (Figure 6.4). This provides a solution to the scaling problem of multicast congestion control. The original requester can represent the receiver group, and provide the feedback needed to the transcoder provider to adapt to the available bandwidth of the bottleneck. This can be achieved on similar time-scales to TCP congestion control, thus resulting in fair sharing of the network with non-multicast traffic. A possible design of a congestion control algorithm for real-time streams and results from simulation are presented in Sections 6.3.2 and 6.3.5 respectively.

6.2.4 Membership Changes

The above discussion does not address start and end time issues. Receivers may join and leave at different times during a conference. When new receivers join they have to find out if their branch of the network is being serviced by a transcoder or the real source, and which address they should join to receive the traffic.

To achieve this a new receiver has to be able to probe existing receivers in its network neighbourhood in a scalable manner. With currently deployed multicast distribution protocols, this can be achieved through the use of a time to live (TTL) based expanding ring search (ERS) algorithm. The idea is that the TTL field of the IP header can be used to limit the lifetime of multicast packets and restrict their distribution to a local part of the network. Using larger TTL values allows packets to live longer and reach further. A new receiver can send query messages to a control group, starting with a low TTL, and increasing until it receives a response.

Unfortunately increasing the TTL over the threshold necessary for the query pack-

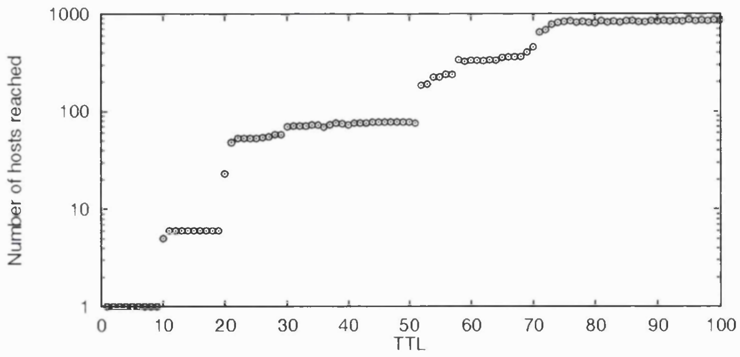


Figure 6.6: Number of hosts running sdr reachable for different TTL.

ets to live beyond some router, means that a whole new part of the network is reached and not a single potential responder. Figure 6.6 shows the number of hosts listening to the SDR [Handley, 1996] session announcement multicast address reachable from UCL for increasing TTL values. The measurements were collected in September 1997 using the multicast ping mechanism. Because the use of ping on multicast addresses causes an implosion of responses resulting in lost messages, the values in the graph are lower than the real ones, especially for larger TTL values. The graph clearly shows that as the TTL exceeds various thresholds, large groups of receivers become reachable.

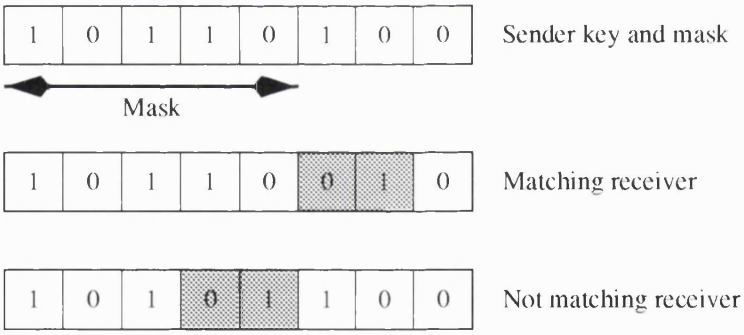


Figure 6.7: Using a sliding key to probe receiver group.

To avoid an implosion of responses to a query message that suddenly reaches a large number of receivers, an additional mechanism is required to restrict the number of potential responders. A sliding key probing mechanism (introduced in [Bolot *et al.*, 1994]) can be employed. This operates by having the query sender and receivers each choose a random key. The sender's key is included in the request, and only receivers with a matching key are allowed to respond. The number of matching keys can be controlled through the use of a mask, that specifies the number of significant bits in the key that have to match (Figure 6.7). Thus the number of responses to a request can be controlled, by selecting an initial mask length, and then re-sending the request with a reduced mask size until a response is received.

An additional problem with the locality achieved through TTL scoped ring searches, is that they cannot be constrained to work along the distribution tree from a given source. That would be desirable behaviour, as we would be able to constrain configuration of transcoding groups along the original source distribution tree, and have more predictable behaviour from our protocol. The use of subcasting, which has recently been proposed as a modification to multicast to support reliable multicast applications, can provide this functionality [Papadopoulos *et al.*, 1998].

Receivers quitting the session are not a problem except in the cases of the requester or the last member of the group leaving. By having the requester periodically multicast alive messages after group formation to the formed group, other members can detect when the requester has left. The messages are sent on a separate control address, to prevent distribution to other session participants. On detection of departure, the remaining group members schedule a message to take over the role of the requester. The transmission is delayed proportionally to the distance from the transcoder, so as to achieve election of the member closest to the bottleneck link. A transcoder can detect the departure of the last member of the group and stop transmission, by the cease of congestion feedback information.

6.2.5 Topology Changes

Link and router outages although not very frequent are quite common in the Internet / Mbone [Paxson, 1997]. As a result of an outage the multicast routing to some members of a session being serviced by a transcoder may change, resulting in a non-optimal or even problematic configuration (Figure 6.8). To recover from such situations, SOT needs to periodically repeat the initiation protocol. The task of doing this can be left up to the original requester.

The resolution of congestion problems that initially caused group formation, or the appearance of a new bottleneck splitting an existing group in half, also affect group dy-

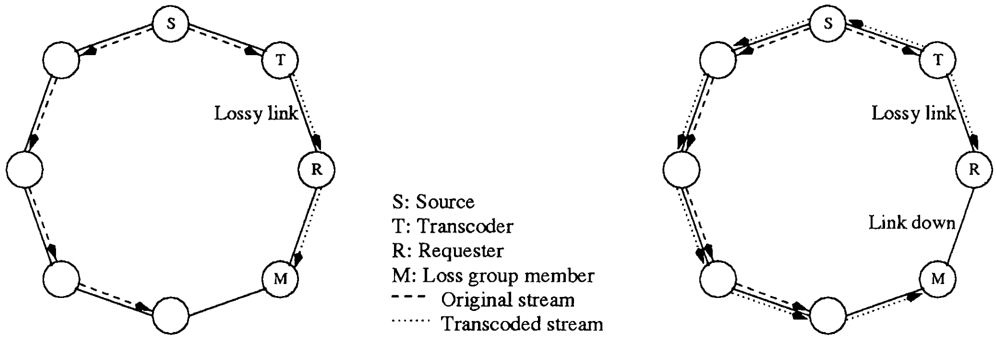


Figure 6.8: Possible effect of a topology change on a transcoder setup (before and after).

namics. The first effect can be addressed by having the transcoder dissolve a group that has been following the transmission rate of the original sender for some period of time. The introduction of a new bottleneck will cause some members of the loss group to observe different loss patterns to those of the requester. By introducing a loss bitmap in the alive messages sent by the requester, this can be detected and the affected members can rerun the initiation protocol.

6.2.6 Multicast Address Allocation

Multicast distribution trees vary for different sources in a session, so SOT adaptation has to be per source. With currently deployed multicast protocols there is no way a receiver can express interest in a particular source. Instead subscription is per group and every sender sending to this group is received. For SOT to work in this environment, each sending participant has to use a separate multicast address to transmit data. In addition, each new transcoder instantiation has to transmit on a new unused address. In sessions with a large number of participants this can be a problem. Typically very large sessions are lecture based where only a few participants transmit data and the majority are only spectators, which somewhat alleviates the problem.

The real solution to this problem is the deployment of the Internet Group Management Protocol version 3 (IGMPv3) [Cain *et al.*, 1997] which is currently under development by the IETF IDMR working group. IGMPv3 supports expression of interest in particular sources for each multicast address joined. This reduces the number of multicast addresses required by SOT to the maximum number of transcoded streams forwarded by any node.

6.3 Simulation

As part of the development of some of the ideas in SOT, and in order to evaluate its performance, we implemented the protocol in version 2.1 of the VINT network simulator ns [McCanne and others, 1997]. Ns is an event driven packet-level simulator. Within ns there are several multicast protocol implementations. We chose to use the dense mode (DM) version, as it behaves similarly to what is currently available on the Mbone. We extended the implementation by adding source-specific group membership control, which is expected to be available on the Mbone with the deployment of IGMPv3 [Cain *et al.*, 1997]. The Self Organised Transcoding protocol was implemented as an extension to ns using C++ and otcl. The protocol implementation and simulation scripts are available upon request from the authors.

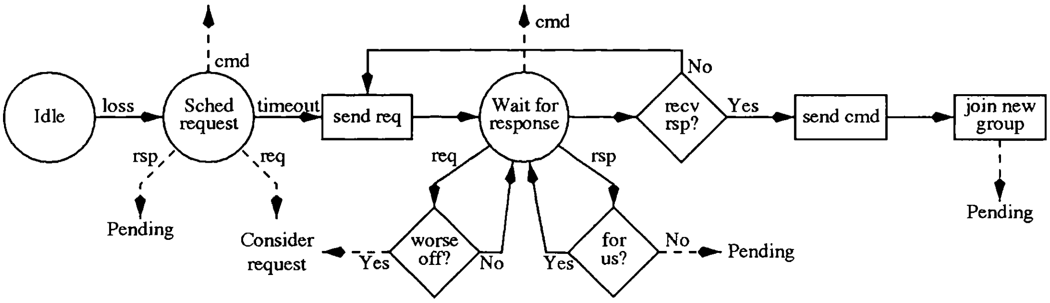


Figure 6.9: Transcoding initiation state transition diagram (requester).

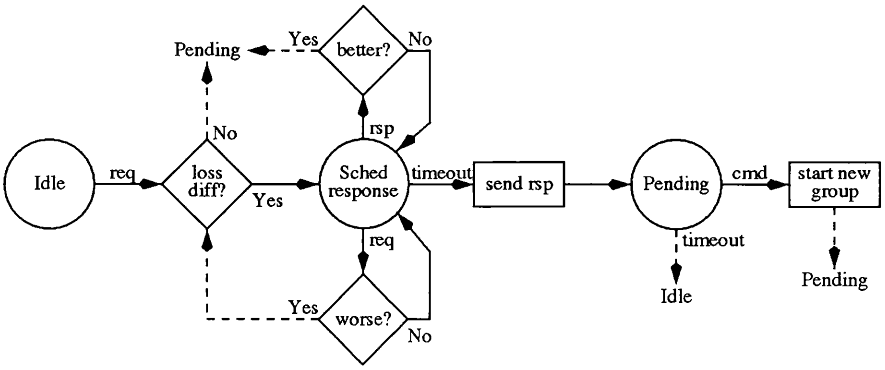


Figure 6.10: Transcoding initiation state transition diagram (transcoding offer).

6.3.1 SOT Implementation

This section describes the design of the SOT transcoder initiation implementation in ns. In the initiation stage SOT uses the following three messages:

request: Sent by the loss group representative to locate possible transcoders. This message identifies the requester and includes the observed loss rate, so that sites willing to offer a transcoded stream can compare their reception and respond only if it is better.

response: Sent by receivers that have received a request, have better reception to the requester and are willing to provide a stream. The locally observed loss rate is included, so that the requester can select the best transcoder if multiple offers are received.

command: After responses have been collected by a requester, a transcoder is selected, and this message sent to instruct the transcoder to initiate the new stream and other loss group members to switch streams. The message includes a loss bitmap for other receivers to compare against and decide if they belong to the group or not.

In addition to receiving the above messages there are two more events that can occur during protocol operation. The first is a *loss report* that is triggered by the reception of a data packet. The second is a *timeout* from the internal protocol timer.

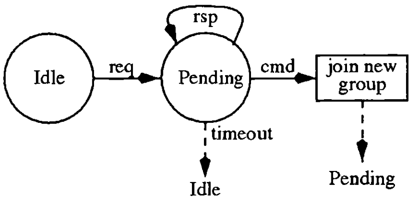


Figure 6.11: Transcoding initiation state transition diagram (loss group member).

These events cause each member of a SOT session to be in one of the following states:

Idle: This is the initial state.

Schedule request: In order to locate a transcoder after loss has been reported a request message has to be sent. To avoid message implosion, these messages are randomly delayed, as described earlier in Section 6.2. A timeout event is scheduled and the receiver waits in this state before sending the request.

Wait for response: After a request has been sent, a requester waits in this state for offers from transcoders. A timer is set to retransmit the request if no responses are received for a predefined period of time. If at the expiration of the timer an offer has been received, a command message is sent out, to initiate the transcoded stream and instruct other receivers in the loss group to receive it.

Schedule response: A site willing to offer its services as a transcoder waits in this state for a timeout before sending a response. Reception of a better offer from another transcoder before the timeout, cancels the scheduled response.

Pending: As SOT messages are multicast to the entire receiver group, in order to reduce the amount of bandwidth consumed at any instant the messages have to be spread out in time. To this end, an attempt to have only one request in progress is made. This is achieved by having all other traffic cancelled and the senders back off, when a setup with a worse loss problem is seen to be in progress. Backing off is implemented by waiting in this state for a timeout that returns the receiver to the idle state.

The transition diagrams for the states and events listed above are shown in Figures 6.9, 6.10 and 6.11 for the requester, the site offering to transcode and a loss group member respectively.

6.3.2 Congestion Control Implementation

After the initiation stage is complete, the requester provides feedback to the transcoder concerning the bottleneck behaviour. The information consists of loss / no loss signals. The transcoder uses this information to adapt the bandwidth of the transcoded stream. Although the adaptation algorithm is independent from the operation of SOT, we implemented a simple version for the purposes of our simulations. The implemented algorithm tries to behave in a manner similar to the congestion control algorithm in TCP [Stevens, 1997], by halving the stream bandwidth when loss is detected, and linearly increasing the bandwidth when no loss is signalled. When transcoders start, the initial bandwidth of the transcoded stream is set to a very low rate thus performing the equivalent of a slow start.

Bandwidth selection is achieved by varying the transcoded packet size. This is the behaviour that would be expected by an audio transcoder, when selecting a different encoding scheme for the outgoing stream. The number and frequency of outgoing packets is the same as that of incoming ones. This is true of most transcoding techniques, as each packet corresponds to a specific time interval. In video transcoding there are two different options to control the transmission rate. The simplest solution is to reduce the frame rate which will result in a smaller number of packets. A better approach is to reduce the quality of the encoded image, resulting in the same number of smaller packets. Codecs available in current Mbone tools support image quality selection.

When stream bandwidth is increased in response to a period with no loss, we are performing an experiment to see if the bottleneck can accommodate some additional traffic. If the link is full this will result in congestion and some packets will be dropped. To avoid degradation in perceived quality due to the loss, the additional bandwidth can be used to carry FEC redundant information [Bolot *et al.*, 1997] for a short period after the bandwidth increase. The switch to a higher quality encoding without redundancy can be postponed until we feel that the link can take the new traffic.

6.3.3 Simulation Metrics and Parameters

SOT was simulated on a number of simple network topologies, which were designed to include specific problematic configurations (Figure 6.12), and on larger random topologies that were created with the assistance of topology generators (Figure 6.13).

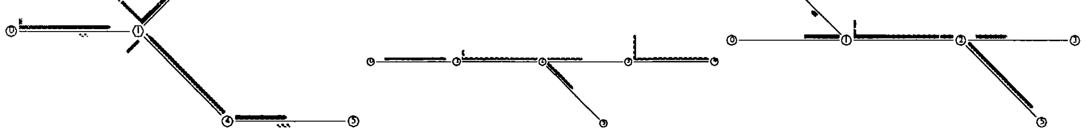


Figure 6.12: SOT simulation on specific problem topologies.

Tests included sparse and dense topologies, bottlenecks in series and introduction of additional non-real-time TCP flows during SOT sessions. In all situations the sim-

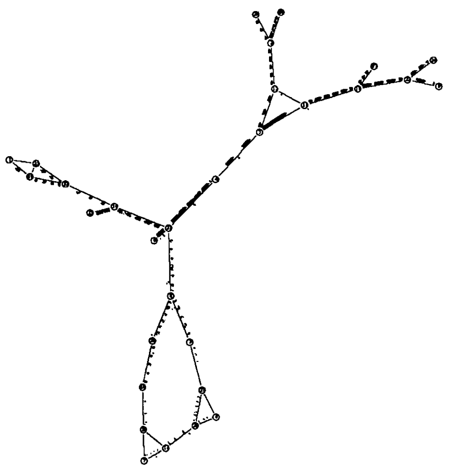


Figure 6.13: SOT simulation on randomly created topologies.

ulations performed as expected forming groups and setting up transcoders around the problem links.

In the rest of this section simulation results evaluating the performance of SOT and the congestion control algorithm are presented. The experiments performed were designed to measure the time-scales over which SOT reacts to a congestion problem, the amount of extra bandwidth used by SOT control messages and the level of network friendliness achieved by the congestion control algorithm.

In all the simulations packets are sent with a frequency of 50 pps (20 ms duration), simulating an audio source[1]. Transcoded packet sizes start from 32 bytes, and are increased in steps of 32 bytes up to the incoming stream bandwidth. Although the number of available coding algorithms is limited, and hence the number of possible packet sizes, with the combination of redundant information all the sizes used in the simulation should be possible in a real audio tool. For video transcoding things are simpler as image quality selection provides a fuller transmission range.

6.3.4 Transcoder Initiation Evaluation

The main goal of the transcoding initiation algorithm is to quickly respond to a congestion problem by setting up a transcoder. The experiments performed aim at measuring the amount of control traffic introduced to the network during initiation, and the delay

[1] The simulation in this chapter is tailored for multicast audio however the proposed scheme can be applied equally well to other types of real-time multicast streams including video.

between detecting a problem and completing initiation.

As transcoder request messages are also used to suppress further requests from other members of a loss group, only one request can be in progress at any one time. In the simulations we resolve conflicts by giving priority to the request reporting the highest loss. Back-off of competing request groups, is achieved by introducing a random delay in the message that instructs other receivers how long they should wait before retrying. This resolves a request synchronisation problem and reduces convergence time.

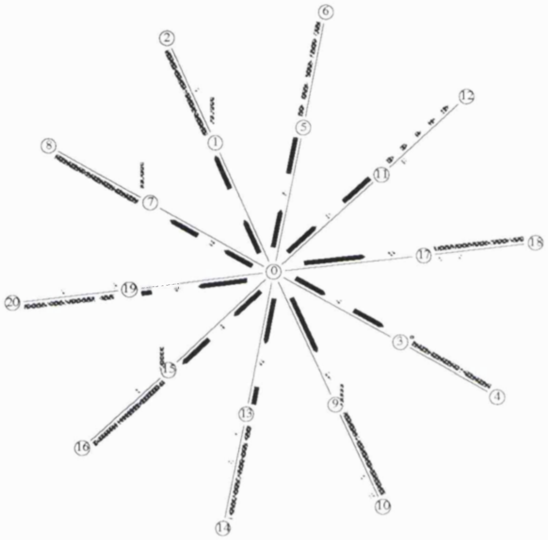


Figure 6.14: SOT simulation on session with 10 bottlenecks.

To measure the back-off algorithm performance, we used sessions with varying number of bottlenecks. The designed network topology is shown in Figure 6.14. The session source is positioned in the centre of a star topology. Each branch contains two receivers connected in series. The connection from the sender to the first receiver is a high bandwidth link (0.5 Mb/s) with delay varying between 20 ms and 40 ms. The connection between the first and second receiver is a lower bandwidth link (150 Kb/s to 200 Kb/s) with longer delay varying between 50 ms and 100 ms. The session bandwidth is set at 256 Kb/s and as a result transcoders need to be set up on all branches between the first and second receivers.

The experiment was repeated with scale varying from one to ten branches. The number of messages that were sent during initiation over the number of transcoders

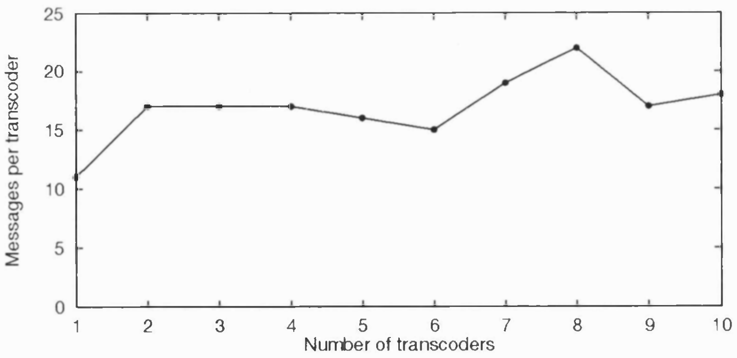


Figure 6.15: Number of SOT control messages sent during transcoder initiation for different scale simulations.

that were set up is shown in Figure 6.15 for different session sizes. The messages per transcoder is roughly constant, showing that the back-off mechanism does not get stuck in loops as a result of request conflicts.

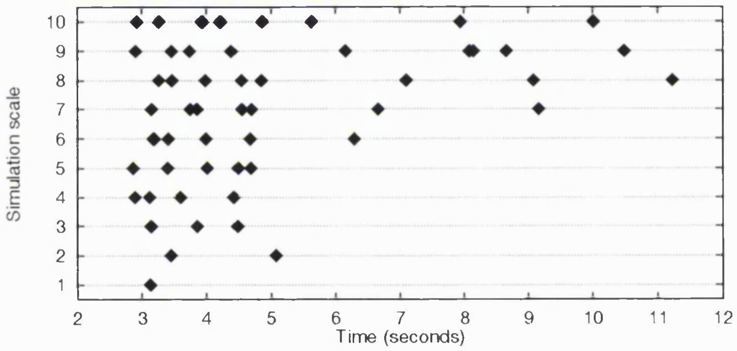


Figure 6.16: Transcoder initiation times for different scale simulations.

In all the simulations the source starts sending RTP packets at time 1.5 s and the first packets are dropped when the queues fill up somewhere around time 2 s. From then on the transcoder initiation process begins. Figure 6.16 shows the completion times for the individual transcoder initiations for each simulation. It can be seen that transcoders are set up more densely at the beginning of the simulation, and then initiations reach a steadier rate. The initial concentration can be attributed to control packet loss. The links are very congested before the transcoders start and as a result request messages may not reach other requesters behind different bottleneck links, thus allowing more than one initiation to progress in parallel. As the simulation progresses and congestion relaxes, the back-off algorithm works better and reduces conflicts by spreading out initiation times.

A reduction of initiation times can be achieved by removing the back-off algorithm and allowing multiple initiations to take place simultaneously. To ensure that other members of the same loss group are still suppressed when one member sends a request, a loss bitmap has to be included in the message. Receivers of the request can then decide if it refers to their loss problem or not. The problem with allowing simultaneous initiations, is use of excessive bandwidth with control messages at any time. This is alleviated by the following factors:

- Very few messages need to be exchanged during initiation.
- It is not important if control packets get lost and do not reach the rest of the net, as initiations should be localised close to the problem.
- The number of initiations is proportional to the number of bottlenecks and not session size, although they may have a close relation depending on geographical coverage.

Persistent Responses: Transcoder response messages serve two purposes. They suppress additional offers and double as a response to the requester. In order to reach the requester, the response has to pass through a congested link. To improve the chances of reaching the requester, the offering site can follow the multicast response with a small number of unicast copies of the message addressed to the requester. These should be spaced in time by a small random interval.

Initiation Congestion Reduction: When a new transcoder starts two new congestion problems can arise. Prunes from the receivers of the new transcoded group for the original data take time to be forwarded to the transcoding site. Hence the original group will still be forwarded for some time causing more congestion on the problematic link. This can be partially avoided by starting the transcoder at very low bandwidth (like a slow-start). The second problem is that receivers of the original group that are not interested in the new multicast traffic need time to prune it. In the meantime the additional traffic may cause problems in previously non-congested links. The slow-start will help here as well. An additional measure can be to have transcoding offers followed by a single packet in the intended new group. In this way receivers that are not interested in the new traffic can have a head start with pruning.

6.3.5 Congestion Control Algorithm Performance

In order to measure the performance of the congestion control algorithm, two different experiments were performed. In the first experiment ten SOT sessions are created that share the same bottleneck link. This arrangement is shown in the topology of Figure 6.17. Each session contains two members, the sender and one receiver. All senders are positioned on nodes on one side of the bottleneck link and all receivers on the other. The bandwidth of all sessions is set at 256 Kb/s. The available bandwidth on the bottleneck link is set at 1 Mb. The aim of the experiment is to show that transcoders are set up, and that they fairly share the bandwidth of the bottleneck link.

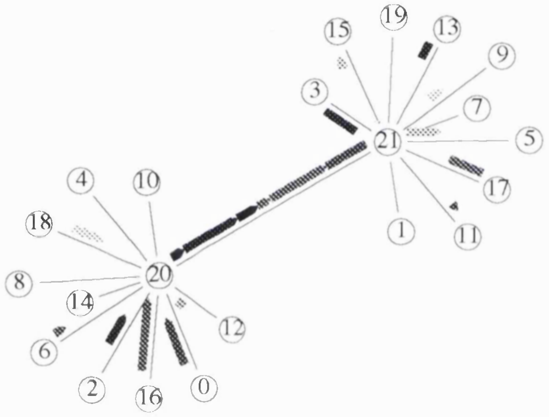


Figure 6.17: Multiple SOT sessions sharing a bottleneck.

Figure 6.18 shows the packet size variation for each of the transcoders during the simulation. There is considerable variation, due to the adaptation process of slowly increasing and then halving the bandwidth, but all the sessions oscillate around roughly the same packet size. This is better illustrated in Figure 6.19, which shows the average packet size and the standard variation during adaptation for each of the transcoders.

The percentage of simulation time that the transcoders spend sending each packet size can be seen in Figure 6.20. This graph is cumulative for all ten sessions. Although during the adaptation some extreme small and large packet sizes are reached, for most of the simulation a small number of packet sizes are used which are close to the optimum.

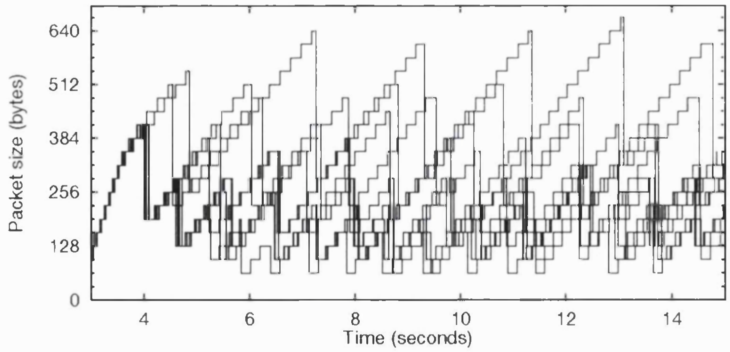


Figure 6.18: Packet size variation for each transcoder during adaptation.

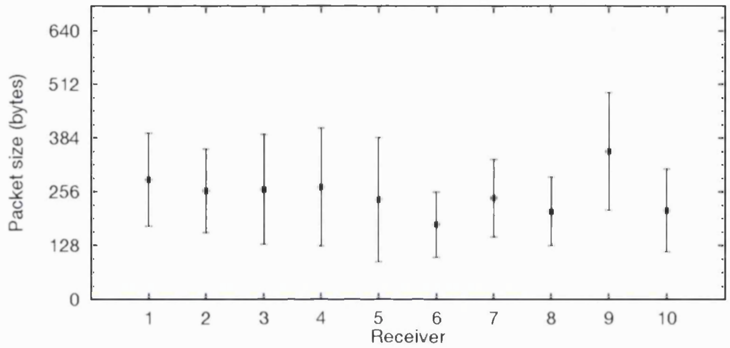


Figure 6.19: Average and standard deviation of packet size for each transcoder.

The cycle of variation between packet sizes will require switching between different codecs and levels of redundancy. Care needs to be taken in order to limit the impact of this variation on the user. With audio the changes will be very hard to perceive except for the cases where really low bandwidth codecs (like LPC) are used. However the amount of time spent by the adaptation algorithm in packet sizes requiring such codecs will be very small (a few packets). Perception experiments carried out at UCL have evaluated the impact of mixing small intervals of LPC synthetic speech with toll-quality speech for the purposes of audio redundancy [Hardman *et al.*, 1995]. Results show that for small intervals (around 40ms) intelligibility of speech does not deteriorate whereas for intervals larger than 80ms there is a slight deterioration.

The second experiment shows fair sharing of the bottleneck bandwidth with TCP. The same topology was used and the bottleneck link shared between four SOT and

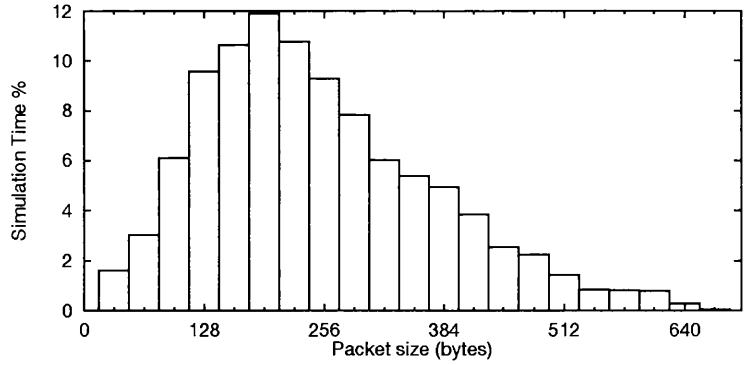


Figure 6.20: Percentage of simulation time that each packet size was used by all the transcoders.

four TCP sessions. The default ns drop-tail queueing strategy was used on the bottleneck link. Using this strategy, the queue capacity is measured in number of packets and packet size makes no difference. Figure 6.21 shows the total bottleneck bandwidth used by SOT and TCP for each second of the simulation after the SOT transcoders have started. The two curves are very evenly matched. Jain's index [Jain *et al.*, 1984] with each individual flow as a user gives 99.5 % fairness.

By modifying the simulation to use drop-tail queues that take into account packet size and have limited buffer space, the fairness drops to 96.8 % The reason for the change is that SOT packets are smaller than TCP packets and hence have a better chance of fitting in a full queue.

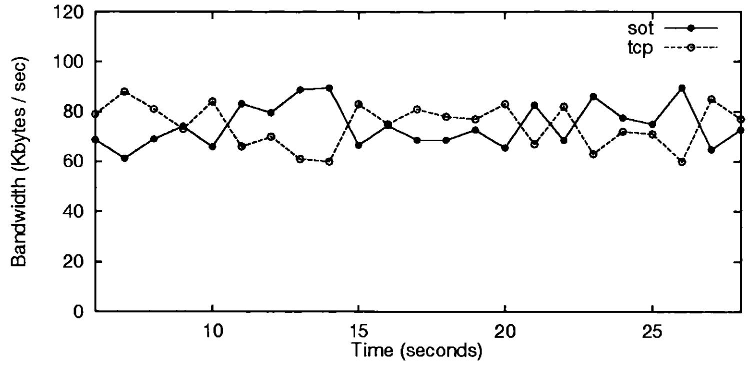


Figure 6.21: Bandwidth use of 4 SOT and 4 TCP sessions sharing a bottleneck link.

6.4 Conclusions

This chapter has presented a solution to the problem of multicast congestion control for real-time streams. The proposed scheme uses self-organisation to form groups out of co-located receivers with bad reception, and provides local repair through the use of transcoders. The receiver driven nature of the protocol ensures high scalability and applicability to large Mbone sessions. An evaluation of the proposed scheme has been conducted through simulation. The results are encouraging as they show that the protocol is viable.

The simple congestion control implementation indicates that fair sharing of bottleneck links between real-time multicast traffic and traditional unicast traffic is possible. The simulation shows that adaptation is possible even with the limited bandwidth of audio communication. The significantly broader range available with video can provide additional freedom to the adaptation algorithm.

Chapter 7

Outlook and Conclusions

Audio communication over the Internet is a new promising technology with rapid growth. Applications have evolved from web based audio file downloads, to streaming of prerecorded sessions and lately to interactive telephony. With the incorporation of cheap multimedia hardware on commonly available networked computers, this technology has become accessible to a critical mass of users.

The deployment of the Mbone and multicast audio can now allow groups of users to participate in real-time, multi-way audio conferences, supporting communication which goes beyond the possibilities of telephony or broadcast technology. When combined with video and shared workspace applications, Mbone based collaboration environments can be tailored to support the requirements of many distributed user groups.

The use however of an environment which was designed for a different class of applications does not come for free. The best-effort transmission service available over the Internet and the general purpose multitasking operating systems running in end-systems degrade the quality of real-time communication. While video and shared data are essential to many distributed tasks, sufficient quality audio is a necessary condition for almost any successful real-time interaction.

This thesis has presented design techniques for the media applications running in end-systems that counteract the problems posed by the Internet environment to real-time audio communication. The proposed solutions, which are summarised below, aim at optimising the user perceived quality of the audio conference for a given set of network and host resources available. An additional target to improved media presentation, is the smooth integration of this new service into the Internet / Mbone.

Packet loss robust audio tool. The effect of network packet loss on audio quality is currently the biggest obstacle to realisation of the full potential of multimedia conferencing over the Mbone. In Chapter 3 we have presented a new robust-audio tool (RAT), which incorporates a redundancy mechanism to counteract the effects of packet loss. The novel audio redundancy is a combined source and channel reconstruction technique particularly suited to conferencing applications. Packet loss protection is achieved by piggybacking a low-overhead highly compressed version of the signal on normal audio packets. The redundancy is used as a fill-in when the primary information is dropped by the network. Results from human perception experiments and real network measurements were used to show the potential and limitations of the audio transmission quality improvements possible with RAT. Since its first release the improved quality that RAT offers has made it popular in the Mbone community. RAT is currently being used for research project meetings, remote language teaching sessions and conference multicasts across Europe and the US.

Current and future research with RAT is aimed at further improving audio quality and providing support for emerging applications. High quality audio will improve conference speech intelligibility (through a low-bandwidth wide-band speech codec), and provide support for the multicast of music and other demanding material (using CD quality sampling and multiple audio channels). 3D sound spatialisation will enable integration of RAT into virtual reality environments (researchers at INRIA have investigated the feasibility of integration of multicast audio into networked games in [Bolot and Parisis, 1998]). All these new applications are likely to require different reconstruction techniques, to that of the presented audio redundancy, which is best suited for audio conferencing.

Workstation scheduling adaptation. Audio quality is affected when the processing capacity of a transmitting or receiving workstation is exceeded, resulting in speech quality degradation similar to that caused by packet loss. This is caused by the lack of support for real-time applications in today's general purpose workstation operating systems. The lack of a delay bound for notification on timers and external events makes it impossible to guarantee smooth delivery of the audio stream. In Chapter 4 we have presented an architecture for a real-time audio media agent that copes with these effects at the application level. The mechanism produces a continuous audio signal, despite the variable allocation of processing time a real-time application is given, by trading off signal quality for buffering delay in a controlled fashion. The proposed solution has been implemented in RAT and evaluated through simulation. The comparison between the proposed method and that used by all other audio tools has shown substantial reductions in both the average end-to-end delay, and the audio sample loss caused by the operating system.

The adaptive cushion mechanism used to smooth out the operating system effects gives accurate information on workstation load. This can be used to tune the processing requirements of media applications during a conference. By maintaining the system in a moderately loaded state, the overall performance of local media presentation can benefit, especially techniques like the inter-stream synchronisation presented in Chapter 5. Information on audio buffer levels, which is also provided by the cushion algorithm, can be used to implement sidetone monitor facilities[1] and echo cancellation in software on computers with adequate CPU power.

Lip synchronisation. Packet audio with silence suppression, variable bit-rate video and the unpredictable Mbone traffic characteristics, produce different end-to-end delays in the transmission and presentation of different media streams. As a result receivers often experience a time lag between hearing a remote user's words, and seeing the associated lip movements. Although in low frame rate applications where the video is used only for presence information this is not a problem, more demanding applications of multimedia conferencing like language teaching are adversely affected. In Chapter 5 we have presented an analysis of the requirements and implementation of the first multicast inter-stream synchronisation over the Mbone between RAT and the vic [McCanne and Jacobson, 1995] Mbone video tool. Implementation efficiency considerations heavily influenced the design chosen, since the obvious method consumes far too much processing

[1] By mixing the microphone input of a workstation to the headset output, an indication can be given to the local user of how loudly they are speaking, enabling them to better control the level of their voice.

power to make the system viable. Subjective performance results have indicated that the efficient implementation is good enough to provide lip synchronisation for demanding multimedia conferencing applications.

Scalable congestion control for multicast continuous media. The scale of the Mbone and heterogeneity in available bandwidth complicate the design of network adaptive multicast applications. The lack of transmission rate adaptation in such applications drives traditional TCP based traffic off the network and can result in network collapse due to congestion. Work in the Internet community is investigating mechanisms for policing the network and penalising unresponsive flows, which is hoped will act as an incentive to application designers to take the requirements of the network under consideration. In Chapter 6 we have presented a new scalable architecture for congestion controlled multicast real-time communication. The proposed scheme uses self-organisation to form groups out of co-located receivers with bad reception and provides local repair through the use of transcoders. The receiver driven nature of the protocol ensures high scalability and applicability to large Mbone sessions. The viability of the proposed solution has been demonstrated through simulation.

Although the simulation results are very encouraging, a real implementation of the protocol within RAT should be made and performance measurements from large sessions collected. Additional analysis is needed to determine the relationship between the bitmap size required for stable operation and the total session size. Interaction of the adaptation algorithm with future network developments should also prove interesting. In particular the proposals of the diffserv group within the IETF will enable SOT transcoders to have a guaranteed basic bandwidth, thus being able to provide a minimum quality to end-users. Active networks and positioning of specialised transcode servers at key locations in the multicast distribution tree, might also become viable solutions for large sponsored events, as the cost of CPU power decreases.

The techniques presented in this thesis and outlined above do not require any changes to be made to the existing network and end-system infrastructure and are implementable in the audio media applications. A different approach at attacking the problem has been to modify the service of the network and host operating systems to provide the guarantees required by the real-time media.

Network modifications are under design by the IETF intserv [Wroclawski, 1997; Braden *et al.*, 1997] and diffserv [Baker *et al.*, 1998] groups to offer services that can provide low delay jitter and loss. These services will eventually be available but due to the charging models that are expected to apply, the best-effort service will always be available and used by many applications including multimedia conferencing.

Operating system modifications to improve real-time application performance offer partial solutions and their deployment will ease the effort required by application adaptation techniques. New systems designed from scratch with new application requirements in mind are being investigated and by providing fine-grained access to resources will offer a significantly better platform albeit at some additional complication of application design [Leslie *et al.*, 1997].

The multicast audio work in this thesis is indicative of the potential of multimedia conferencing tools and applications over the Internet. Studying network characteristics, exploiting knowledge about human perception, and applying computational techniques can lead to considerable improvement in existing tools and the perceived quality of audio and video. Currently such techniques are necessary to work around the infrastructure shortcomings. As the cost of network capacity, CPU power and memory decreases, the approach of a best-effort infrastructure with application adaptation may continue to dominate as a low-cost solution, over the provision of QoS guarantees through network reservations and operating-system resource control.

Bibliography

[Baker *et al.*, 1998] Fred Baker, Scott Brim, Tony Li, Frank Kastenholz, Shantigram Jagannath, and John K. Renwick. IP precedence in differentiated services using the assured service. Internet draft (work-in-progress) *draft-ietf-diffserv-precedence-00.txt*, April 1998.

[Ballardie and Crowcroft, 1995] Tony Ballardie and Jon Crowcroft. Multicast-specific security threats and counter measures. In *Proceedings of Internet Society Symposium on Network and Distributed System Security*, San Diego, CA, February 1995.

[Ballardie, 1997] T. Ballardie. Core based trees (CBT version 2) multicast routing – protocol specification –. Request for Comments (Experimental) 2189, Internet Engineering Task Force, September 1997.

[Bolot and Garcia, 1996] Jean-Chrysostome Bolot and Andres Vega Garcia. Control mechanisms for packet audio in the internet. In *Conference on Computer Communications (IEEE Infocom)*, San Fransisco, California, March 1996.

[Bolot and Parisis, 1998] Jean Bolot and Sacha Fosse Parisis. Adding voice to a distributed game on the internet. In *Conference on Computer Communications (IEEE Infocom)*, San Francisco, California, March 1998.

[Bolot *et al.*, 1994] Jean-Chrysostome Bolot, Thierry Turletti, and Ian Wakeman. Scalable feedback control for multicast video distribution in the internet. In *SIGCOMM*, London, UK, August 1994. ACM.

[Bolot *et al.*, 1997] Jean-Chrysostome Bolot, Sacha Fosse-Parisis, Mark Handley, Vicky Hardman, Orion Hodson, Isidor Kouvelas, Colin Perkins, and Andres Vega-Garcia. RTP payload for redundant audio data. Request for comments (Proposed

Standard) RFC 2198, Internet Engineering Task Force, Audio / Video Transport Working Group, September 1997.

[Bolot, 1993] Jean-Chrysostome Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High-Speed Networks*, 2(3):305–323, December 1993.

[Braden *et al.*, 1997] R Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP), version 1 functional specification. Request for comments (Proposed Standard) RFC 2205, Internet Engineering Task Force, September 1997.

[Brady, 1968] Paul T. Brady. A statistical analysis of on-off patterns in 16 conversations. *Bell System Technical Journal*, 47:73–91, January 1968.

[Brady, 1971] Paul T. Brady. Effects of transmission delay on conversational behavior on echo-free telephone circuits. *Bell System Technical Journal*, 50:115–134, January 1971.

[BT, 1984] Algorithm disclosure for a wideband speech coder. Technical Report Doc No 57, CCITT SG XVIII, Rapporteurs Group on Wideband Coding within 64 kbit/s, September 1984. Source: British Telecom.

[Buckett *et al.*, 1995] J. Buckett, I. Cambell, T. J. Watson, M. A. Sasse, V. J. Hardman, and A. Watson. ReLaTe: Remote language teaching over SuperJANET. In *UKERNA 95 Networkshop*, University of Leicester, UK, March 1995.

[Cain *et al.*, 1997] Brad Cain, Steve Deering, and Ajit Thyagarajan. Internet Group Management Protocol, version 3. Technical report, Internet Engineering Task Force, November 1997. Internet Draft (work in progress).

[Carpenter, 1996] B. Carpenter. Architectural principles of the internet. Request for Comments (Informational) 1958, Internet Engineering Task Force, June 1996.

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

[Casner, 1994] Stephen Casner. Are you on the MBone? *IEEE MultiMedia, Summer 94*, pages 76–79, 1994.

[Clark and Tennenhouse, 1990] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM*, pages 200–208, Philadelphia, Pennsylvania, September 1990. ACM. Computer Communications Review, Vol. 20(4), Sept. 1990.

[Cox and Kroon, 1996] R.V. Cox and P. Kroon. Low bit-rate speech coders for multimedia communication. *IEEE Communications Magazine*, pages 34–41, December 1996.

[Deering *et al.*, 1988] S. Deering, C. Partridge, and D. Waitzman. Distance vector multicast routing protocol. Request for Comments (Experimental) 1075, Internet Engineering Task Force, November 1988.

[Deering, 1989] S. Deering. Host extensions for IP multicasting. Request for comments (Proposed Standard) RFC 1112, Internet Engineering Task Force, August 1989. Obsoletes RFC 0988 1054.

[Deering, 1991] Stephen Edward Deering. *Multicast routing in a datagram internetwork.* PhD thesis, Stanford University, Palo Alto, California, December 1991.

[Deering, 1997] S. Deering. Protocol independent multicast-sparse mode (PIM-SM): protocol specification. Request for Comments (Experimental) 2117, Internet Engineering Task Force, June 1997.

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

[Fall *et al.*, 1995] 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 Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, pages 179–182, Durham, New Hampshire, April 1995. Springer.

[Faller, 1992] 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.

[Fisher, 1992] 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.

[Floyd and Fall, 1998] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. Technical report, Network Research Group, Lawrence Berkeley National Laboratory, Berkeley, CA, February 1998. Submitted to IEEE/ACM Transactions on Networking.

[Floyd *et al.*, 1997] S. Floyd, V. Jacobson, S. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transcations on Networking*, 5(6):784–803, December 1997. An earlier version of this paper appeared in *ACM SIGCOMM 95*, August 1995, pp. 342-356.

[Frederick, 1994] Ron Frederick. Network video (nv). Software on-line, September 1994.

[Gruber and Strawczynski, 1985] John G. Gruber and Leo Strawczynski. Subjective effects of variable delay and speech clipping in dynamically managed voice systems. *IEEE Transactions on Communications*, 33(8):801–808, August 1985.

[H261, 1993] Video codec for audiovisual services at p x 64 kbit/s. ITU-T Recomendation H.261, International Telecommunication Union, 1993.

[H323, 1996] Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service. ITU-T Recomendation H.323, International Telecommunication Union, 1996.

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

[Handley and Jacobson, 1998] M. Handley and V. Jacobson. SDP: session description protocol. Request for Comments (Proposed Standard) 2327, Internet Engineering Task Force, April 1998.

[Handley *et al.*, 1993] M. Handley, P. Kirstein, and M. A. Sasse. Multimedia integrated conferencing for european researchers (MICE): piloting activities and the conference management and multiplexing centre. *Computer Networks and ISDN Systems*, 26(3):275–290, 1993.

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

[Handley *et al.*, 1998] M. Handley, J. Crowcroft, C. Bormann, and J. Ott. Very large conferences on the Internet: the Internet multimedia conferencing architecture and the Mbone. *Computer Networks and ISDN Systems*, 1998. To appear.

[Handley, 1996] Mark Handley. SAP: Session Announcement Protocol. Internet Draft, Internet Engineering Task Force, November 1996. Work in progress.

[Handley, 1997a] Mark Handley. An examination of Mbone performance. Research Report ISI-RR-97-450, USC/ISI, April 1997.

[Handley, 1997b] Mark Handley. Network Text Editor (NTE): A scalable shared text editor for the MBone. In *SIGCOMM*, pages 197–208, Canne, France, September 1997.

[Handley, 1997c] Mark Handley. *On Scalable Internet Multimedia Conferencing Systems*. PhD thesis, University College London, London, UK, August 1997.

[Hardman and Iken, 1996] Vicky Hardman and Markus Iken. Enhanced reality audio in interactive network envioronments. In *In Proceedings of the FIVE Technical Conference*, Pisa, Italy, December 1996.

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

[Hardman *et al.*, 1996] 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, England, February 1996.

[Hardman *et al.*, 1998] Vicky Hardman, Martina Angela Sasse, and Isidor Kouvelas. Successful multi-party audio communication over the internet. *Communications of the ACM*, 41(5), May 1998.

[Hardman, 1993] Victoria Jayne Hardman. *Wide-band Speech Teleconferencing over an Integrated Network.* PhD thesis, Loughborough University of Technology, UK, 1993.

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

[Jacobson, 1988] Van Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, August 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

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

[Jain *et al.*, 1984] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report DEC Research Report TR-301, DEC, September 1984.

[Jardetzky *et al.*, 1995] 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.

[Jayant and Christenssen, 1981] N.S. Jayant and S.W. Christenssen. Effects of packet losses in waveform coded speech and improvements due to odd-even sample-interpolation procedure. *IEEE Transactions on Communications*, COM-29(2):101–109, February 1981.

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

[Khanna *et al.*, 1992] Sandeep Khanna, Michael Sebrée, and John Zolnowsky. Real-time scheduling in sunos 5.0. In *Usenix Winter Technical Conference*, 1992.

[Kirstein *et al.*, 1995] P.T. Kirstein, M.A. Sasse, and Handley M.J. Recent activities in the mice conferencing project. In *International Networking Conference (INET)*, 1995.

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

[Kouvelas *et al.*, 1996] Isidor Kouvelas, Vicky Hardman, and Anna Watson. Lip synchronisation for use over the internet: Analysis and implementation. In *IEEE Globecom '96*, London, UK, November 1996.

[Kouvelas *et al.*, 1997] Isidor Kouvelas, Orion Hodson, Vicky Hardman, and Jon Crowcroft. Redundancy control in real-time internet audio conferencing. In *Proceedings of AVSPN 97*, Aberdeen, Scotland, UK, September 1997.

[Kouvelas *et al.*, 1998] Isidor Kouvelas, Vicky Hardman, and Jon Crowcroft. Network adaptive continuous-media applications through self organised transcoding. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July 1998.

[Lamont *et al.*, 1996] 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.

[Leslie *et al.*, 1997] Ian Leslie, Derek McAuley, Richard Black, Timothy Roscoe, Paul Barham, David Evers, Robin Fairbairns, and Eoin Hyden. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal on Selected Areas in Communications*, June 1997.

[Lin and Costello, 1983] S. Lin and D. J. Costello. *Error Correcting Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[Maxemchuk *et al.*, 1997] N.F. Maxemchuk, K. Padmanabhan, and S. Lo. A cooperative packet recovery protocol for multicast video. In *International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

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

[McCanne and others, 1997] Steve McCanne et al. UCB/LBNL/VINT network simulator - ns (version 2). Software and documentation available from http://www-mash.cs.berkeley.edu/ns/, November 1997.

[McCanne *et al.*, 1996] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In *SIGCOMM*, pages 117–130, Stanfort, CA, August 1996. ACM.

[Miller and Licklider, 1950] George A. Miller and J. C. R. Licklider. The intelligibility of interrupted speech. *Acoustical Society of America*, 22(2):167–173, 1950.

[Mills, 1992] David L. Mills. Network Time Protocol (version 3) specification, implementation and analysis. Request for comments (Proposed Standard) RFC 1305, Internet Engineering Task Force, March 1992.

[Mills, 1995] D. L. Mills. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Transcations on Networking*, 3(3):245–254, June 1995.

[Mills, 1996] D. Mills. Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6 and OSI. Request for comments (Proposed Standard) RFC 2030, Internet Engineering Task Force, October 1996. Obsoletes RFC 1769.

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

[Moy, 1994] J. Moy. Multicast extensions to OSPF. Request for Comments (Proposed Standard) 1584, Internet Engineering Task Force, March 1994.

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

[Papadopoulos *et al.*, 1998] Christos Papadopoulos, Guru Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *Conference on Computer Communications (IEEE Infocom)*, San Francisco, California, March 1998.

[Pasquale *et al.*, 1993] Joseph C. Pasquale, George C. Polyzos, Eric W. Anderson, and Vachaspathi P. Kompella. Filter propagation in dissemination trees: Trading off bandwidth and processing in continuous media networks. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 259–268, Lancaster, U.K, November 1993.

[Paxson, 1997] Vern Paxson. End-to-end routing behavior in the internet. *IEEE/ACM Transcations on Networking*, 5(5):601–615, October 1997.

[Pejhan *et al.*, 1996] S. Pejhan, M. Schwartz, and D. Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. *IEEE/ACM Transcations on Networking*, 4(3):413–327, June 1996.

[Rabiner and Schafer, 1978] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

[Ramjee *et al.*, 1994] 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.

[Rizzo, 1997] Luigi Rizzo. The FreeBSD audio driver. In *Proceedings of the COST 237 workshop*, Lisbon, Portugal, December 1997.

[Schulzrinne *et al.*, 1996] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. RTP: a transport protocol for real-time applications. Request for comments (Proposed Standard) RFC 1889, Internet Engineering Task Force, January 1996.

[Schulzrinne, 1992] Henning Schulzrinne. Voice communication across the Internet: A network voice terminal. Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[Schulzrinne, 1995] 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.

[Schulzrinne, 1996] Henning Schulzrinne. RTP profile for audio and video conferences with minimal control. Request for comments (Proposed Standard) RFC 1890, Internet Engineering Task Force, January 1996.

[Sharma *et al.*, 1998] Puneet Sharma, Deborah Estrin, Sally Floyd, and Lixia Zhang. Scalable session messages in SRM using self-configuration. submitted to Sigcomm '98, February 1998.

[Shenker *et al.*, 1997] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Request for comments (Proposed Standard) RFC 2212, Internet Engineering Task Force, September 1997.

[Sherif *et al.*, 1993] Mostafa Hashem Sherif, Duane O. Bowker, Guido Bertocci, Bruce A. Orford, and Gonzalo A. Mariano. Overview and performance of CCITT/ANSI embedded ADPCM algorithms. *IEEE Transactions on Communications*, 41(2):391–398, February 1993.

[Spinellis, 1998] Diomidis Spinellis. A critique of the windows application programming interface. *Computer Standards & Interfaces*, 19, 1998.

[Stankovic *et al.*, 1995] 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.

[Stevens, 1997] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. Request for comments (Proposed Standard) RFC 2001, Internet Engineering Task Force, January 1997.

[Turletti and Huitema, 1995] 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.

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

[Vicisano *et al.*, 1998] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Conference on Computer Communications (IEEE Infocom)*, San Francisco, California, March 1998.

[Warren, 1982] R.M. Warren. *Auditory Perception.* Pergamon Press Inc., 1982.

[Watson and Sasse, 1996] A. Watson and M. A. Sasse. Evaluating audio and video quality in low-cost multimedia conferencing systems. *Interacting with Computers*, 8(3):255–275, 1996.

[Wroclawski, 1997] J. Wroclawski. Specification of the controlled-load network element service. Request for comments (Proposed Standard) RFC 2211, Internet Engineering Task Force, September 1997.

[Xu *et al.*, 1997] X. Rex Xu, Andrew C. Myers, Hui Zhang, and Raj Yavatkar. Resilient multicast support for continuous-media applications. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, Missouri, May 1997.

[Yajnik *et al.*, 1996] Maya Yajnik, Jim Kurose, and Don Towsley. Packet loss correlation in the MBone multicast network. In *Proceedings of Global Internet*, London, England, November 1996.

Publications

- Isidor Kouvelas, Vicky Hardman and Anna Watson. Lip Synchronisation for use over the Internet: Analysis and Implementation. In *Proceedings of IEEE Globecom '96*, November 1996, London UK.

- Isidor Kouvelas and Vicky Hardman. Overcoming Workstation Scheduling Problems in a Real-Time Audio Tool. In *Proceedings of Usenix Annual Technical Conference*, January 1997, Anaheim, California.

- Colin Perkins, Vicky Hardman, Isidor Kouvelas and Angela Sasse. Multicast Audio: The Next Generation. In *Proceedings of INET 97*, June 1997, Kuala Lumpur, Malaysia.

- Isidor Kouvelas, Orion Hodson, Vicky Hardman and Jon Crowcroft. Redundancy Control in Real-Time Internet Audio Conferencing. In *Proceedings of AVSPN 97*, September 1997, Aberdeen, Scotland, UK.

- Colin Perkins, Isidor Kouvelas, Orion Hodson, Vicky Hardman, Mark Handley, Jean-Chrysostome Bolot, Andres Vega-Garcia, Sacha Fosse-Parisis. RTP Payload for Redundant Audio Data. *IETF Audio/Video Transport Working Group*, RFC2198, September 1997.

- Vicky Hardman, Angela Sasse and Isidor Kouvelas. Successful Multi-party Audio Communication over the Internet. In *Communications of the ACM*, 41(5), May 1998.

- Isidor Kouvelas, Vicky Hardman and Jon Crowcroft. Network Adaptive Continuous-Media Applications Through Self Organised Transcoding. In *Proceedings of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98)*, July 1998, Cambridge, UK.