LULEÅ TEKNISKA UNIVERSITET

1999:04

RESEARCH REPORT

Robust Audio Transport using mAudio

Kåre Synnes

Research report

Institutionen för Systemteknik
Avdelningen för Programvaruteknik

1999:04 • ISSN: 1402-1528 • ISRN: LTU-FR--99/04--SE

Robust Audio Transport using mAudio

Kåre Synnes, Peter Parnes, Dick Schefström
Department of Computer Science / Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
{Kare.Synnes,Peter.Parnes,Dick.Schefstrom}@cdt.luth.se

ABSTRACT

IP based groupware applications, such as net-based learning environments, rely on robust audio transport for efficient communication between users. This paper therefore gives an overview and an initial evaluation of how to achieve robust transport of real-time audio streams over Internet connections without service guarantees. Due to hardware jitter and network congestion these connections face loss, packet delay and delay variation. Audio streams are especially sensitive due their real-time characteristics, where the end result is degradation of the perceived quality. Packet loss can be repaired using receiver-only, sender-initiated or receiver-initiated techniques. Depending on the actual network condition, an optimal technique can be selected using adaptive behavior together with loss-recovery techniques in the applications. Studies have shown that loss rates up to 20% can be effectively repaired using fairly simple techniques. The paper gives initial results from subjectively evaluating audio quality and presents a research prototype called mAudio that has been used to experiment with different loss recovery techniques.

Keywords: robust, audio, mStar, environment.

1. Introduction

Internet is a rapidly growing phenomenon in many perspectives, perhaps mostly due to the technical momentum created by its development. The first really large use of Internet was for interchange of messages, email, but the large boom in usage coincided with the introduction of the World Wide Web and the sudden instant global access to information. The third large step in the development of the Internet will probably be the introduction of bandwidth demanding media transport with real-time characteristics.

New services include interactive TV, Internet telephony and multi-part conferencing solutions, which all have real-time elements like streamed audio and video. These services require low delay to allow for interactiveness, as well as low loss for intelligibility. This paper is focused on how audio transport can be made robust over lossy Internet connections, since transport of real-time audio data over the Internet is particularly affected by delay and loss.

Many modern audio applications therefore include techniques for loss recovery and a few even include adaptive behavior to meet changing network conditions automatically. However, loss recovery techniques may instead increase delay and bandwidth usage. Tools should therefore be adaptable to the current requirements of a session, where either audio quality or delay is promoted (while keeping bandwidth usage in mind). For instance, a playback of a recording would select audio quality over delay, while a live conference would select the opposite.

In particular, audio traffic transported using the Real-time Transport Protocol, RTP, on the MBone is well prepared for repair algorithms [Schulzrinne 99, Deering 91]. This is due to the presence of a sequence number and a time stamp in the RTP packet header. Several research prototypes like VAT, RAT and FreePhone has shown that real-time audio transport indeed can be conducted with good result, even under lossy network conditions [Jacobson 92, Hardman 95, Bolot 98].

This paper give an overview of networking related to IP Multicast and RTP. It also provides a discussion and comparison of different methods for repairing losses of real-time audio data. These methods are either *receiver-only* (where the sender is not involved at all), or in cooperation with both sender and receiver. The latter case can be further divided into *sender-initiated* or *receiver-initiated* techniques. Observe that many of these methods are not exclusive, and that a combination of these techniques is required to achieve the best possible audio quality. The paper ends with describing a reference implementation of an adaptive audio tool, mAudio, which has been used to evaluate the different recovery techniques described together with a tool for subjective audio quality tests.

1.1 Background

The Centre for Distance-spanning Technology, CDT, at Luleå University of Technology has since the foundation in 1995 conducted research on net-based learning and collaborative teamwork environments. The result is the mStar environment, which is a platform for implementing distributed applications based on IP Multicast [Parnes 97a, Parnes 97b].

Numerous courses have been given using the mStar environment, spanning from small informal graduate courses to full-fledged under-graduate courses with hundreds of participants [Schefström 98, Synnes 98, Synnes 99]. The environment has also been in extensive use within most of the projects conducted at CDT for internal meetings and presentations.

This extensive use has shown that the most important of the different real-time media involved is audio, due to the fact that small disturbances easily can render the audio stream unintelligible. The video has mostly been used for achieving a sense of presence, and the other media are more or less non real-time (chat, whiteboard) since they use a reliable protocol for transport. Efforts have therefore been spent to study how to achieve the best audio quality during different network conditions.

An experimental audio tool, mAudio, has been implemented in order to study different techniques for repair, among them different adaptive algorithms.

2. Networking Issues

While traditional telephony networks are constructed for optimal transport of real-time audio data, Internet is inherently not. Data transported using traditional telephony services will arrive with little delay variation and low loss. The only service offered over Internet, best effort, give no guarantees on delays or delay variation. This means that Internet applications can neither rely on a guaranteed bit rate, nor assume an uncongested transport service. Several efforts are ongoing to extend the Internet architecture to support more transport services [Braden 97, Wroclawski 97, Shenker 97, Blake 98]. However, such extensions have not yet been widely deployed. In fact, it may take several years before service guarantees are globally available on the Internet. In lieu of this, IP based applications with real-time constraints should be constructed with delay, delay variation and loss in mind.

Another issue is scalability, where large sessions traditionally have been relying on special replication servers in the network. However, Deering proposed the concept of IP multicast [Deering 91], where replication is handled at network level. This alleviates the scalability problems, at least for moderately sized sessions. The IP multicast backbone is referred to as the MBone. The MBone is built with IP routers equipped with software allowing them to forward IP packets not only to a single receiver but also to a group of receivers. These loosely coupled sessions offer clear advantages in scalability over replication-based unicast services, since the amount of traffic sent over the network and the control needed at sender side are minimized.

A sender simply sends its data to the group address, without explicit knowledge about who is receiving. A receiver joins a group by listening to the group address, and is thereby forwarded the traffic by the network. Naturally this level of control is too limited for many real-time applications. The application level protocol RTP is therefore used as a protocol above UDP (Note that RTP is not limited to run over UDP only). RTP also has a control protocol built in, RTCP, which allows the receiver to report back to the sender about its networking characteristics. Each packet distributed with RTP has a timestamp and a sequence number, which makes it suited for transport of real-time data with error recovery in mind.

There have been several attempts to describe the characteristics of the Internet in general and IP multicast in particular, with parameters like loss, delay and delay variation. Due to the heterogeneous nature of the Internet, this has proven to be a hard task. The results are therefore not fully consistent but show common trends regarding IP multicast traffic [Bhattacharyya 98, Paxson 97, Schulzrinne 93, Bolot 93, Handley 97, Yajnik 96].

Research indicates that most receivers suffer from loss up to 5% due to hardware jitter[1] and light congestion, while a few experience higher degrees of loss due to greater network congestion. A reasonable design assumption is that if the number of receivers is large, then a packet is likely to be lost at least once by one receiver in the group. The relation between bandwidth usage and experienced loss is clear, which needs to be kept in mind when constructing adaptive schemes for loss recovery.

Unless the network is heavily congested, several consecutive losses are unlikely when packets are sent out evenly spaced [Bolot 95]. A uniform distribution is therefore a good approximation of losses in networks not suffering from heavy congestion. For more detailed simulations, a Markov chain would better describe the loss characteristics of a network, especially when heavy congestion is to be involved.

Lastly, packets that are late due to delays in the network might be dropped by the application. The solution is to let the application have a playout buffer that adapts to changes in delay and delay variation. This allows interactive applications to suffer from a minimal delay for a specified loss rate. Several studies have been made to construct optimal algorithms for adaptive buffers [Ramjee 94, Moon 95].

In general, the best way to combat transient losses are to apply techniques for packet repair as described in the next chapter, while long term losses require bandwidth adaptation [Kouvelas 98]. The latter will be discussed in chapter four. A possible extension is to also support balancing of reliability and interactability (loss tolerance vs. delay) [Kouvelas 98].

3. Techniques for Repair of Audio Streams

This section aims to present the different techniques for creating a loss tolerant audio application. Our focus and experience has been with these techniques applied to IP multicast based tools, but note that they can with equal benefit be applied to IP unicast tools or even regular ISDN applications.

The simplest techniques are those that does not rely on the sender for achieving loss tolerance. These *receiver-only* techniques are sufficient to cover losses induced by hardware jitter and light congestion, that is losses up to 5%. When higher amounts of loss are experienced (typically due to greater network congestion), the sender needs to take measures to lower the losses as well. These repairs are either *sender-initiated* or *receiver-initiated*.

Note that the exact percentage of tolerable losses are very subjective and varies from user to user. Also note that more complex repair schemes may increase the delay in the system.

Manipulated sound clips are used in the subsequent subchapters to visualize the effect of some of the techniques

[1] Hardware jitter is a result from faulty hardware, such as a badly connected cable or a damaged hub, which results in packet loss.

described therein. They are all based on a simple phrase, “Hello World”, which is depicted in figure 1 below. Repaired information is colored gray in the following figures.

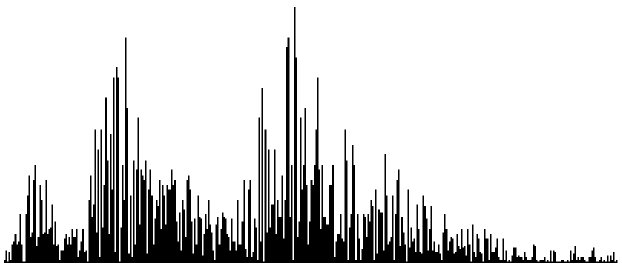


Figure 1, “Hello World”

3.1 Receiver-only techniques

These techniques are prominent when the recovery techniques that involve the sender have failed to replace a lost packet.

3.1.1 Silence Substitution

This is the simplest loss recovery technique available, where lost packets are replaced by silence. Its simplicity limits its usefulness, as it is only effective up to approximately 1 % of loss [Jayant 81, Gruber 85]. It may also cause strain, as the clipping effects are quite tiresome.

The packet size affects the effectiveness of this technique, where a packet sizes used for audio usually is either 20, 40, or 60 ms long. A phoneme is about 20ms long and loosing a full phoneme affects intelligibility, thus one lost packet means that 1 to 3 phonemes are lost. For anything but really low losses, silence substitution is therefore bad. Figure 2 shows silence substitution at 40% loss and 20 ms packets.

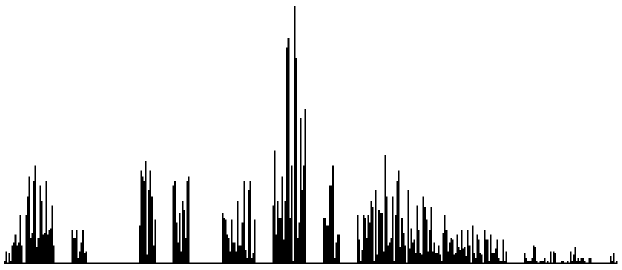


Figure 2, Silence Substitution

However, using an additional technique may improve the use of this technique further. An example of this is striping, which is a *sender-initiated* technique explained later.

3.1.2 Warping

The name of this repair technique, warping, hints that the timing is disrupted during playout. A lost packet is simply ignored and the next in line is used instead, thus resulting in a consumption of the playout buffer when loss occurs. This technique can therefore prove meaningless since the fallback when the playout buffer is consumed is silence substitution while the buffer builds up. It has therefore similar characteristics as silence substitution, but it might complicate the use of advanced adaptive buffers. The biggest downside is however a quite ugly distortion in the flow of the speech. Figure 3 shows the “Hello World” clip using warping.

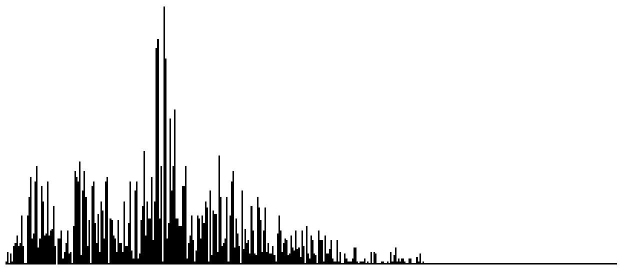


Figure 3, Warping

3.1.3 Noise Substitution

The human brain is equipped with the ability to do subconscious repairs of sound distorted with noise. This is used in noise substitution, where white (or gray) noise is used instead of silence for repair of losses. This is shown to increase the intelligibility as well as perceived quality.

A common way of deciding what noise amplitude should be used is to track the power of the received data, and then base the noise repairs power on this. This however includes the use of silence detection on the receiver side. An alternative is that the sender uses silence suppression, only sending audio when necessary, or only use silence detection to calculate a noise power that is then sent to the receiver out-of-band.

This technique is slightly better than silence substitution, and thus gives a higher loss tolerance. Note that the selection of the noise waveform is important, where selecting a too high noise may actually lessen the subjective gain in audio quality. Figure 4 shows noise substitution repairs based on the mean amplitude of previous packets.

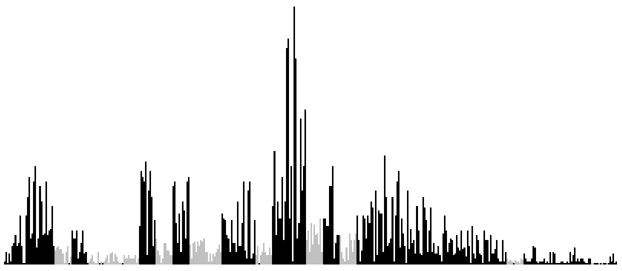


Figure 4, Noise Substitution

An alternative way would be to add a continuos noise to the signal, that act as background noise. The power of the noise should then be calculated on the loss rate and the average power of the real signal. In GSM, the term for this is comfort noise. It is also possible to only add the continuos noise when needed, that is after a certain degree of loss, in order to further minimize the disturbance caused by the overlaid noise.

3.1.4 Repetition

A technique introduced by GSM [ETSI 92] is to use previously received data for repair. Simply put, take the last packet and repeat it if the current packet is lost. GSM uses subsequent repetitions for as long as 320ms, when using a data size of 20ms (or 33 bytes). The repeated packet is slowly faded until silence.

This works quite well, since speech waveforms often exhibit a degree of self-similarity. That is, nearby located packets will show similar spectral qualities. As a result, repetition works well up to 5-10% of loss.

An advantage with repetition is that it is quite simple to implement. The disadvantage is that quite ugly reverberation effects can occur if repetition is overdone. A recommendation would be to keep the repetition small, which works fine for moderate loss due to hardware jitter or low congestion. Using 40 ms packets a repetition scheme of two subsequent repetitions with an amplitude gain shift of 50% each works quite well, while if using 20 ms packets three repetitions with an amplitude gain shift of 33% is recommended. Figure 5 shows repetition of 2 packets at 50% gain shift.

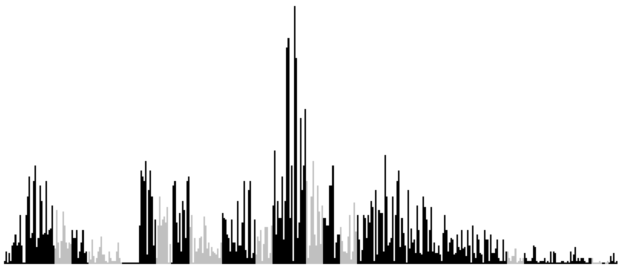


Figure 5, Repetition

3.1.5 Forward repetition

The notion of repairing a lost packet with a subsequent packet is similar to the normal repetition, but work only for small number of subsequent losses.

A combination of normal and forward repetition may be the best solution, especially when subsequent losses occur. Using the latter combined technique increases the loss tolerance above the simple repetition technique.

3.1.5 Mixing

Mixing the surrounding packets and applying an amplitude gain shift is another technique that works well for low losses. This keeps much of the spectral qualities of the lost packet, especially if using a 20 ms or smaller packet size.

If more than one packet is subsequently lost, the surrounded packets can first be gain manipulated. This in order to find the best balance when mixing the two packets, and thus achieve an as accurate as possible repair. Figure 6 shows mixing with distance balancing (where the amplitudes of the two surrounding packets are amplified depending on their respective distance to the lost packet).

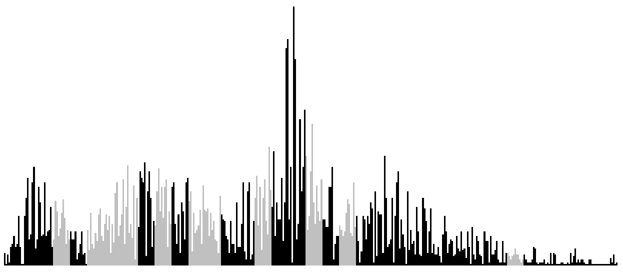


Figure 6, Mixing

3.1.6 Interpolation

This technique is based on studying the spectral qualities of the packets surrounding the loss. Goodman et al [Goodman 86] has studied interpolation limited to preceding data as well as surrounding data. This technique may be computationally demanding and does not give much better results than mixing.

Note that mixing can be seen as a simple form of interpolation, where more complex interpolation techniques uses the spectral qualities of the surrounding packets in order to achieve a more accurate repair than is possible when mixing.

3.1.7 Stretching

A lost packet can also be repaired by stretching the surrounding packets to cover the loss. This is like interpolation computationally demanding, but performs a little better. A downside may be spectral effects that are introduced when manipulating the samples, similar to warping.

3.1.8 State Interpolation

The GSM speech coder [ITU-T 96] is an example of an encoder that uses linear prediction. This makes it possible to interpolate the linear predictor state, and thus achieve a repair based on knowledge about the speech encoder. This is however a complex and very computationally demanding process and the much simpler mixing technique achieves similar results.

3.2 Receiver-initiated techniques

By *receiver-initiated* we mean that the sender is the active party for preventing loss. Observe that most of these techniques use redundant data, which may not be used at all at the receiver side and thus adds bandwidth. This may render these techniques useless under the condition of network congestion.

The optimal solution is to combine receiver-initiated techniques with bandwidth adaptation methods, as described in chapter four.

3.2.1 Striping

Striping means that data from several packets are interchanged. A single loss therefore incurs several small losses instead of one large. An example if we use a 5 packet striping technique using 20 ms packets, where not one 20 ms will be the effect but 5 4 ms losses. This greatly increases the efficiency of most *receiver-only* techniques.

One downside is that a computational overhead for moving the data around, but the gain may be greater when using striping together with for instance silence suppression than using a more complex technique like state interpolation. Another downside is increased delay. Figure 7 shows striping using a 3 packet striping technique.

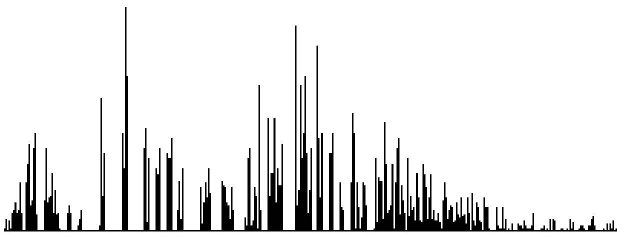


Figure 7, Striping

3.2.2 Interleaving

Interleaving is very similar to striping, but interchange hole packets. It is therefore not as computationally demanding, but does neither give as good results. The method is however quite effective for small packet sizes (~20 ms), and can have the good effect of reducing subsequent losses due to congestion. Assume that we have a sequence of packets A…F…, transmitting them like A-C-B-D… or A-D-B-E-C-F… will avoid two subsequent packets to be sent immediately after each other.

The downside is naturally increased delay, but as with striping no extra data is sent which is good for the overall bandwidth consumption.

3.2.3 Media Independent Forward Error Correction

Recent research have reused the ideas behind algebraic codes designed to detect and correct errors in a data stream to generate redundant data to increase the ability to recover from loss. Rosenberg et al has proposed parity coding and Reed-Solomon coding for inclusion in the RTP payload specification [Rosenberg 98].

The idea is quite simple, a repair packet is generated of n packets and this repair packet can be used together with n-1 packets to regenerate a single lost packet (of the n packets sent).

The biggest advantage with this technique is that is media independent, but it is also one if its biggest disadvantages. In the case of audio data, information about the data can increase the usefulness of the FEC coding. It is however fairly trivial to implement media independent FEC, given that existing coders can be reused. Another disadvantage is that it like all FEC techniques increases the bandwidth requirements.

3.2.4 Media dependent FEC

Hardman et al [Hardman] suggests transmitting audio data multiple times, and thus can use the redundancy to repair losses. This technique is very simple, but increases the bandwidth requirements like the media independent FEC. Podolsky et al [Podolsky 98] and Bolot et al [Bolot 99] have also studied this type of FEC.

Depending on the bandwidth restrictions, several different combinations of main and redundant data can be used. For instance, the main encoding can be PCM while the redundant data is encoded using GSM to decrease the bandwidth requirement. Another example is to use GSM in combination with LPC instead. The more synthetic GSM and LPC coders is well able to cover for small losses, since they preserve the frequency spectra (LPC is consider to contain ~60% of the speech information, while preserving most of the frequency spectra).

Redundant information is usually encoded using a low bit rate encoding, since these are fully capable of covering small losses. These could also be selected adaptively, to face changing network conditions, where even multiple redundancy can be used if the net suffers from high levels of hardware jitter while being otherwise uncongested.

The redundant information could be transmitted in several ways. First it can be sent as extra packets in the same data stream (or multicast group) as the main encoding. Secondly it can be piggybacked to the same packets as the main encoding, thus decreasing the overhead for headers and parsing. Lastly it can be sent in parallel as separate data streams (or multicast groups), which enable the receivers to select if they need the repairs or not. The last can be good when facing receivers with inhomogeneous networking conditions, where adding redundancy in some cases will increase congestion greatly.

An advantage is that the redundant data could either be sent immediately (if not piggybacked), or be delayed one to several packets. This allows for a sender to trade delay against increased loss tolerance (and thus better audio quality).

Note also that a receiver could select to ignore the redundant data in order to achieve a lower delay. This allows for a differentiation of the receivers, where a real-time conversation would require low delay while a recording would emphasize a high loss tolerance.

3.2.5 Simple Layering

By sub-sampling the main high-quality audio source, it is possible to achieve layered encodings. A simple example of this is to use a 32 kHz audio stream, then sub-sample and filter it to 8, 16 and 24 kHz. The receiver can then select to join the best possible stream (multicast group), depending on the current network conditions, as well as joining several streams to achieve a higher loss tolerance where a lower quality stream is upsampled and used as repairs.

Note that this can be used in combination with FEC, where higher qualities is sent directly and lower qualities uses the FEC techniques. The receiver can then better adapt to networking conditions and select the best use of the available bandwidth.

3.2.6 Wavelets

Another form of layering is the use of wavelets, which is a spectral encoding. Wavelets make it possible to separate a high quality audio stream into frequency bands. For instance, a 32 kHz stream could be separated into 0-8, 8-16 and 16-32 kHz subbands.
Each subband is the transmitted on its own stream (or multicast channel) and the receivers can simply select the best possible configuration according to the current networking conditions. The received subbands are then added and played back.

The big disadvantage of using wavelets is that it adds much delay, and thus is not suitable for interactive applications. It is also quite computationally demanding, even if it achieves a good level of compression.

3.3 Receiver-initiated techniques

In order to avoid sending redundant data that is not used anyhow, a basic idea would be to let the receivers tell the senders when loss occurred. This is a good idea, but for obvious reasons it is hard to find a useful scenario of usage for this kind of techniques. The main problems are delay and scalability.

3.3.1 Reliable Transmission

One technique for reliable transmission is the use of Scalable Reliable Multicast, SRM [Floyd 97], techniques. When a receiver detects loss, it will request a repair from the sender. This is done after a random time depending on the number of receivers and the distance from the receiver. If other receivers loose the same packet, they will suppress their request if they see another receiver's request for that same packet. Any receiver that intercepts a repair request may send a repair if available.

The huge downside to this technique is delay, since the repair can be delayed for a relatively long time. As a result, this technique is not suited for interactive applications. The worst case is also that at least one receiver loose each packet, which is possible for a large number of receivers, where the network bandwidth requirement is doubled. Naturally, the retransmitted repairs do not have to be of the same quality as the main transmission.

3.3.2 Semi-reliable Transmission

The idea with semi-reliable transmission is to time limit the repairs, thus trading loss tolerance for lesser delay. That is, if a repair has not arrived within the limited time, then ignore it and repair it with other means.

However simple, this makes it possible to differentiate the receivers much like with the FEC techniques. The usefulness is limited though, as a FEC technique is simpler for much the same effect.

4. Adaptive Applications

Due to the varying conditions of the network, applications need to adapt in order to achieve the best possible result. The need for trading delay versus loss tolerance, and bandwidth required versus loss tolerance are but two adaptations that can be made automatically.

To do so, mechanisms for detecting hardware jitter and congestion is needed. Then these mechanisms should be used to avoid congestion yet achieve the best possible perceived quality.

4.1 Network Metrics

The use of RTP allows the receivers to make several measurements on the network conditions. First, loss rates can be calculated using the sequence number. These can be calculated with short term or long term loss in mind (hardware jitter or congestion), as well as in comparison with other streams in order to decide if the loss is local or distant.

Since audio data is timed, it is also possible to use that timing information in order to decide the delay variation. So all necessary metrics are available in order to be able to adapt to changing network conditions.

4.2 Congestion control

As described in chapter three, there are many techniques for adapting to the existing bandwidth. No standard protocol for controlling the congestion exist however, even if some are suggested [McCanne 96, Vicisano 98].

A good start is to use a combination of media dependent FEC and *receiver-only* techniques, together with layered encodings based on multicast group separation. These combinations will be able to remedy most cases of congestion and hardware jitter, where the receivers play a central role for their own error tolerance. It is by that also possible to adapt to the role of the receiver, be it a telephony application or a session recorder.

Several efforts have been made to study an optimal control mechanism for real-time traffic [Busse 95, Talley 94, Cen 98, Bolot 96].

4.3 mManager

[Parnes 99a, Parnes 99b] presents the architecture and implementation of a new proposed framework for control and management of software applications. It allows applications to distribute messages in a scalable way, with regard to both the numbers of applications currently running and the available control bandwidth. This is done using IP-multicast together with a messaging platform called *the Control Bus (CB)* and a reliable multicast protocol (*SRRTP*). Note, that the whole framework is designed without any special requirements from the underlying transport mechanism as long as it is transport reliable and uses IP-multicast (unicast can be used but the framework becomes much less scalable then).

The novel usage of IP-multicast for management and control creates a mobile and scalable framework that can be used for a number of different applications including better service for distributed audio applications.

The management framework could be used within distributed real-time media applications to signal out-of-band cues to give better relative service within a session to important media senders. For instance, when a user is sending audio, video transmitters within a session should lower their targeted bandwidth. Another usage scenario is to allocate bandwidth between users and sessions depending on external input. For instance, important sessions should get more bandwidth allocated to them than less important sessions. This leads to a system where media applications cooperate both within and between sessions instead of competing for bandwidth, as is the case in today's global media sessions.

5. Subjective Evaluations of Audio Quality

One way to decide the usefulness of loss recovery techniques is to make subjective tests of the perceived audio quality. The effect of delay is already well described where a delay of more than 250 ms are generally considered to affect an interactive application negatively.

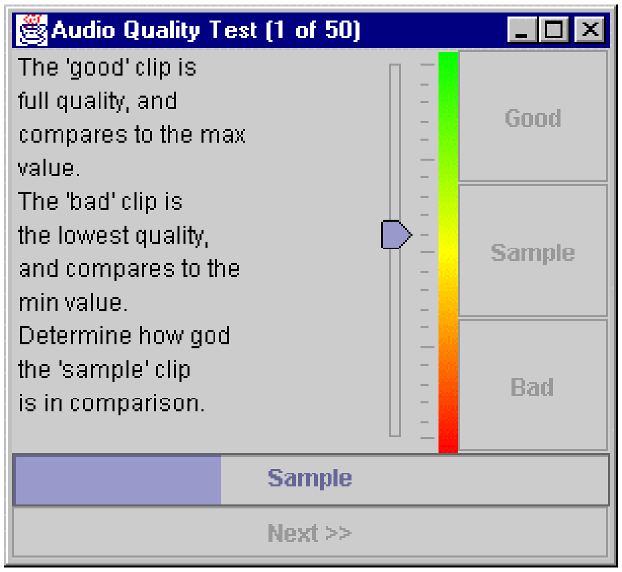
For this reason, a test bed has been implemented in order to do subjective trials of perceived audio quality with different loss recovery techniques at varying loss rates. The initial results confirm the results of other subjective tests [Hardman 95, Kouvelas 97]. However, the initial evaluations done by the authors may not be statistically assured, since too few users have been tested so far.

The most commonly used technique to subjectively evaluate audio quality is to use a full quality sample and a degraded sample, where the user then rates the degraded sample from 1 (equal to the full quality sample) to 5 (significantly distorted). If you combine this with some method for equalizing the variations among the users, then it gives a fairly good measure of the overall perceived quality. The downside is that the lower end of the scale (to decide what are significant distortions) is very personal and thereby varies a lot.

In our subjective evaluation of audio quality we therefore used three samples instead; a full quality sample, a degraded sample and a distorted sample. The user instead grades the degraded sample between the full and the distorted sample, and does so by freely replaying any of the 3 samples before deciding. This removes the uncertainty of the lower level of former method, and also lessens the need for equalizing the variations among the users.

Each user typically graded 50 degraded samples of speech, which were randomly selected and manipulated with loss and recovery methods. The loss rates used were 2, 5, 10, 15, 20, 30 and 40%. The pilot recovery methods used for this initial evaluation were silence substitution, noise substitution, single redundancy, single repetition and double repetition. These were selected as a starting point for evaluation of more advanced schemes.

The evaluation was conducted by 37 users, of which 65% were male and 85% were used to talk in mobile telephone. Figure 8 shows the user interface of the test application, where the user is asked to subjectively grade degraded samples between "*good*" quality (the original full quality samples) and "*bad*" quality (the sample distorted with 40% loss repaired with silence substitution)[2].
The metric perceived quality is ranged from 0 to 100, where 0 and 100 is equal to distorted and full quality respectively. An important question is where the level of acceptable degradation is located on this scale. This was found to be between 2 and 5 % loss, above which the distortion was clearly disturbing.


Audio Quality Test (1 of 50)

The 'good' clip is full quality, and compares to the max value.
The 'bad' clip is the lowest quality, and compares to the min value.
Determine how god the 'sample' clip is in comparison.

Figure 8, Audio Quality Test Application

[2] This conforms to the Good and Bad buttons in the tool.

The evaluation confirms that noise substitution is slightly better than plain silence substitution, in general. However, many of the users reacted on the noise as quite disturbing, which may be an effect of the actual noise selected. The noise is based on a randomization using a uniform distribution, without any filtering for higher frequencies. As a conclusion, either a prerecorded noise or a generated noise with filtering will be used for further studies. Another conclusion was that a continuos noise signal based on signal power and loss rate would increase the effect of noise substitution further, as the short bursts of noise is quite disturbing. A more uniform presence of noise would be preferable. See figure 9 for a comparison between silence and noise substitution.

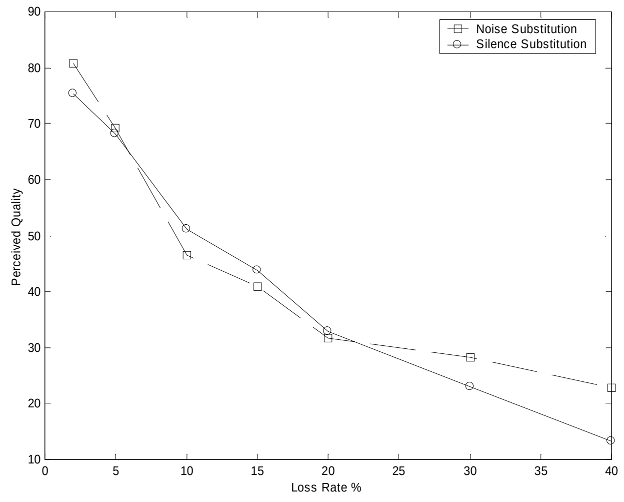


Figure 9, Silence vs. Noise Substitution

Using redundancy increases loss tolerance greatly as expected, while repetition yielded an unexpected result. The double repetition technique was considered slightly worse than the single repetition. Asking the users proofed that they perceived the audio as metallic, with 'tin can' effects. Figure 10 shows the effect of redundancy together with silence substitution and figure 11 shows single vs. double repetition.

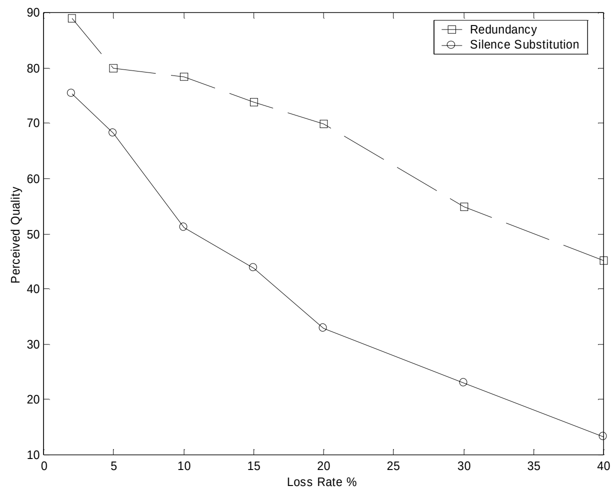


Figure 10, Redundancy

A final conclusion is that perceived quality for even moderately lossy (around 20%) audio streams can be achieved when using simple techniques as repetition and noise substitution together with redundancy. The perceived quality for this combination reaches the level of acceptable degradation.

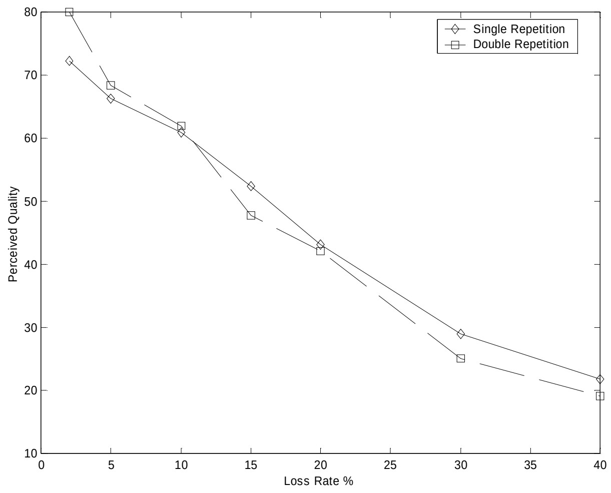


Figure 11, Single vs. Double Repetition

6. mAudio

The mAudio application is a VAT/RAT compatible research application, which has been used to evaluate different loss recovery techniques. mAudio has been used as an integrated component in the mDesk application for conferencing and net-based learning, and has also been used for recoding and traffic concentration in the mTunnel tool [Parnes 97b, Parnes 98]. It is implemented in Java and C for Solaris and Windows. Figure 12 shows an early prototype of mAudio for Solaris from 1997.

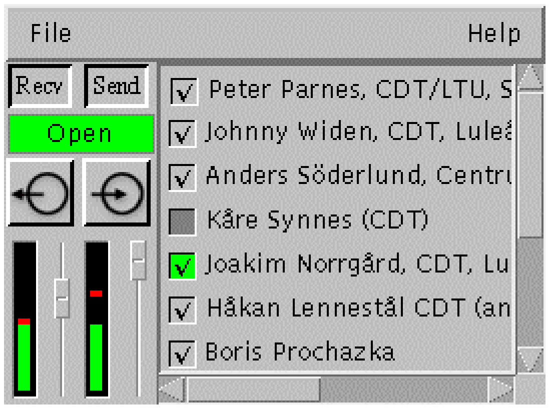


Figure 12, The mAudio Prototype

mAudio uses adaptive playout buffers together with a fairly simple loss recovery algorithm (the annex describes the algorithm used as pseudo code for 40 ms packets). The algorithm is based on packet repetition, together with noise substitution. The play-out buffer size is adjusted by tracking the sequence number of the RTP packets. This implementation is simple, but quite effective for small losses. More complex techniques, for instance mixing, striping and parallel FEC techniques were also implemented.

The block diagram of the data paths is also quite simple, and is depicted below in figure 13. The device captures audio, which is forwarded to a silence detection algorithm (based on the mean amplitude) and to the mixer for loopback. The encoder receives audio not below the silence detection level, encodes it (currently only GSM and PCM is supported) and feeds the transmitter with the encoded audio. The transmitter sends the audio data to the network using RTP. The receiver divides the traffic into several streams, and puts the data into one adaptive buffer per sender. The mixer requests data from a number of byte buffers (again one per sender), which fetches and decodes data from the adaptive buffer upon need.

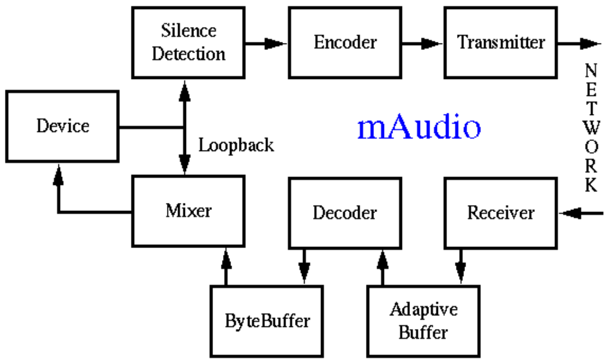


Figure 13, mAudio Block Diagram

The usage of Java showed to be a limitation however, as the Solaris implementation turned out to have severe resource problems with heavy real-time traffic. This resulted in high CPU usage (typically 20% on a SUN UltraSparc 170E using JDK 1.1 with green threads), together with severe timing problems on lesser SUN workstations (such as the Sparc5). The application was therefore kept as simple as possible, to reduce CPU usage. Recent just-in-time compilers with native thread handling have made it much better, basically lowering the CPU consumption with 30% while removing most of the timing problems on slower systems. Generally, the play-out is quite robust, while the recording is more sensitive. This is due to that the device itself also keeps a play-out buffer, while the more direct recording often lead to long delays when the machine hangs during thread switching. These tendencies are however in common with the Windows platforms, especially on the Windows 95 side. A larger play-out buffer can remedy this, but it also creates a lot of additional delay (the Windows Win32 wave device may stall for 500 ms in rare cases during capture). Using really small clips (20 ms or less) improves the smoothness of the capturing under Windows, but it takes more CPU.

The mAudio prototype is today developed further to a building block of the commercially available conferencing tool Marratech PRO. Figure 14 shows the mAudio component, integrated with a video component in Marratech[3] PRO mVideo, used in a net-based learning setting at Luleå University of Luleå.

[3] Marratech AB is a spin-off company from the Centre for Distance-spanning Technology founded in 1998 to commercialize the ideas of multicast-based conferencing tools.

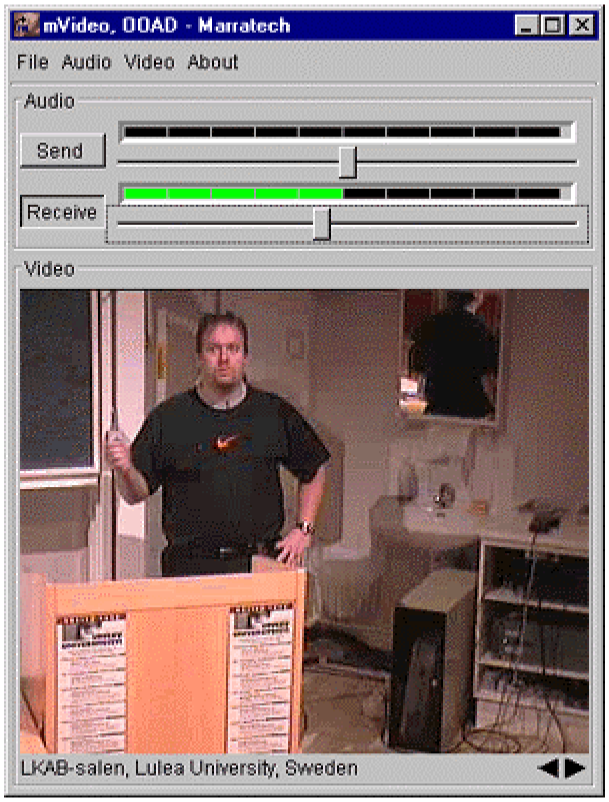


Figure 14, Marratech PRO mVideo

7. Conclusions

This paper gives an overview of available techniques for loss recovery for audio streams. It shows that a great deal can be done to achieve fairly good loss tolerance using simple techniques. The paper also points at the trade-off between loss tolerance and delay, and on the point that an application should be implemented to adapt to it's networking conditions and use. Initial results of a subjective audio quality evaluation are also presented together with an experimental prototype for audio conferencing. The results confirm previously done research, which shows that up to 20% loss can be tolerated when simple recovery techniques are used. One algorithm, used in the mAudio tool, is also presented that give a reasonably good loss tolerance under network conditions with hardware jitter and light congestion.

Creating IP-based groupware applications, such as net-based learning environments, that are resilient to loss, delay and delay variation is therefore possible.

7.1 Future Work

Clearly the work on the subjective audio quality evaluation needs to be expanded to achieve greater statistical precision, together with the addition of more complicated loss recovery techniques like media dependent FEC. The study also focuses on perceived audio quality alone, and thus give little information about intelligibility.

References

[Schulzrinne 99] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: a transport protocol for real-time applications", Internet Draft, Internet Engineering Task Force, March 1999. Work in progress.

[Deering 91] S. Deering, “Multicast routing in a Datagram Internetwork”, Ph.D. Thesis, Stanford University, CA, December 1991.

[Jacobson 92] V. Jacobson, S. McCanne, “Vat - X11-based audio conferencing tool”, Lawrence Berkeley Laboratory, University of California, Berkeley, 1992.

[Hardman 95] V. Hardman, A. Sasse, M. Handley, A. Watson, "Reliable Audio for Use over the Internet", INET '95, Honolulu, Hawaii, June 1995.

[Bolot 98] J. Bolot, S. Parisis, "Adding voice to a distributed game on the Internet", Conference on Computer Communications (IEEE Infocom), San Francisco, California, p.p. 480, March 1998.

[Parnes 97a] P. Parnes, “The mStar Environment: Scalable Distributed Teamwork using IP Multicast”, Licentiate Thesis, Luleå University of Technology, Sweden, 1997.

[Parnes 97b] P. Parnes, K. Synnes, D. Schefström, “The CDT mStar environment: Scalable Distributed Teamwork in action”, Group '97, Phoenix, Arizona, USA, November 1997.

[Schefström 98] D. Schefström, J. Widén, P. Parnes, K. Synnes, A. Söderlund, “Education Direct - Entering the World Beyond the Web”, EdMedia ‘98, June 1998, Freiburg, Germany.

[Synnes 98] K. Synnes, S. Lachapelle, P. Parnes, D. Schefström, “Distributed Education using the mStar Environment”, WebNet ’98, Orlando, Florida, USA, November 1998.

[Synnes 99] K. Synnes, P. Parnes, J. Widén, D. Schefström, “Net-based Learning for the Next Millenium”, Research Report 1999-05, Luleå University of Technology, 1999. Submitted to ACM Multimedia ’99.

[Braden 97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP), IETF RFC 2205, September 1997.

[Wroclawski 97] J. Wroclawski, Specification of the Controlled-Load Network Element Service, IETF RFC 2211, September 1997.

[Shenker 97] S. Shenker, C. Partridge, R. Guerin, Specification of the Controlled-Load Network Element Service, IETF 2212, September 1997.

[Blake 98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, Specification of Guaranteed Quality of Service, IETF RFC 2475, December 1998.

[Bhattacharyya 98] S. Bhattacharyya, D. Towsley, J. Kurose ``The Loss Path Multiplicity Problem for Multicast Congestion Control'', University of Massachusetts, CMPSCI Technical Report 98-76.

[Paxson 97] V. Paxson, “End-to-End Internet Packet Dynamics”, ACM SIGCOMM '97, Cannes, France, September 1997.

[Schulzrinne 93] H. Schulzrinne, “Reducing and characterizing packet loss for high-speed computer networks with real-time services”, Ph.D. Thesis, University of Massachusetts, Amherst, Massachusetts, May 1993.

[Bolot 93] J. Bolot, “End-to-end packet delay and loss behavior in the Internet”, ACM SIGCOMM'93, p.p. 289-298, San Francisco, September 1993.

[Handley 97] M. Handley, “An examination of MBone performance”, USC/ISI Research Report ISI/RR-97-450, April 1997.

[Yajnik 96] M. Yajnik, J. Kurose, D. Towsley, “Packet loss correlation in the MBone multicast network”, IEEE Global Internet Conference, November 1996.

[Bolot 95] Bolot et al., “Analysis of Audio Packet Loss on the Internet”, NOSSADV ‘95, p.p. 163-174, Durham, NH, April 1995.

[Rosenberg 98] J. Rosenberg, H. Schulzrinne, "An RTP Payload Format for Reed Solomon Codes", IETF Internet Draft draft-ietf-avt-reedsolomon-00.txt, November 1998.

[Rosenberg 99] J. Rosenberg, H. Schulzrinne, "A RTP Payload Format for Generic Forward Error Correction v5", IETF Internet Draft, draft-ietf-avt-fec-05.txt, February 1999.

[Schulzrinne 95] H. Schulzrinne, "QoS for real-time services: playout delay and application control," in Proceedings of 46th RACE Concertation Meeting (RCM), Brussels, Belgium, March 1995.

[Sanneck 96] H. Sanneck, A. Stenger, K. Younes, B. Girod, "A new technique for audio packet loss concealment," Proceedings of Global, London, England, p.p. 48-52, IEEE, Nov. 1996.

[Ramjee 94] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks", Computer Communications IEEE Infocom, Toronto, Canada, pp. 680-688, IEEE Computer Society Press, Los Alamitos, California, June 1994.

[Moon 95] S. Moon, J. Kurose, D. Towsley, “Packet audio playout delay adjustment algorithms: performance bounds and algorithms”, Research report, Department of Computer Science, University of Massachusetts at Amherst, Amherst, Massachusetts, August 1995.

[Kouvelas 98] Kouvelas et al, "Network Adaptive Continuous-Media Applications Through Self-Organized Transcoding", NOSDAV '98, Cambridge, UK, July 1998.

[Kouvelas 97] I. Kouvelas, O. Hodson, V. Hardman, J. Crowcroft, "Redundancy Control in Real-Time Internet Audio Conferencing", AVSPN ‘97, Aberdeen, Scotland, September 1997.

[Jayant 81] N. Jayant, S. Christenssen, “Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure”, IEEE Transactions on Communications, COM-29, p.p. 101-109, February 1981.

[Gruber 85] J. Gruber, L. Strawczynski, “Subjective effects of variable delay and clipping in dynamically managed voice systems”, IEEE Transactions on Communications, 33-8, p.p. 801-808, August 1985.

[ETSI 92] ETSI, “Substitution and muting of lost frames for full rate speech channels”, Recommendation GSM 6.11, 1992.

[Goodman 86] D. Goodman, G. Lockhart, O. Wasem, W. Wong, “Waveform substitution techniques for recovering missing speech segments in packet voice communications”, IEEE Transactions on Acoustics, Speech, and Signal Processing, 34-6, p.p. 1440-1448, December 1986.

[ITU-T 96] International Telecommunications Union (T), “Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s”, Recommendation G.723.1, 1996.

[Podolsky 98] M. Podolsky, C. Romer, S. McCanne, “Simulation of FEC-Based Error Control for Packet Audio on the Internet”, INFOCOM '98, San Francisco, CA, March 1998.

[Bolot 99] J. Bolot, A. Vega-Garcia, “The case for FEC based error control for packet audio in the Internet”, Accepted by ACM Multimedia Systems.

[Floyd 97] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang, “A reliable multicast framework for lightweight sessions and applications level framing”, IEEE/ACM Transactions on Networking, December 1997.

[McCanne 96] S. McCanne, V. Jacobson, M. Vetterli, “Receiver-driven layered multicast”, ACM SIG-COMM '96, Stanford, CA., August 1996.

[Vicisano 98] L. Vicisano, L. Rizzo, J. Crowcroft J, “TCP-like congestion control for layered multicast data transfer”, IEEE INFOCOM, 1998.

[Busse 95] I. Busse, B. Deffner, H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," in First International Workshop on High Speed Networks and Open Distributed Platforms, St. Petersburg, Russia, June 1995.

[Talley 94] Talley, K. Jeffay, “Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams”, Proceedings Second ACM International Conference on Multimedia, San Francisco, CA, October 1994.

[Cen 98] S. Cen, C. Pu, J. Walpole, “Flow and Congestion Control for Internet Streaming Applications”, Multimedia Computing and Networking (MMCN ’98), 1998.

[Bolot 96] J. Bolot, A. Vega-Garcia, “Control mechanisms for packet audio in the Internet”, IEEE INFOCOM '96, 1996.

[Parnes 99a] Peter Parnes, Kåre Synnes, Dick Schefström, “Real-Time Control and Management of Distributed Applications using IP-Multicast”, IM ’99, Boston, MA, USA, May 1999.

[Parnes 99b] Peter Parnes, Kåre Synnes, Dick Schefström, “A Framework for Management and Control of Distributed Applications Using Agents and IP-multicast”, IEEE Infocom '99, New York, USA, March 1999.

[Parnes 98] Peter Parnes, Kåre Synnes, Dick Schefström, “Lightweight Application Level Multicast Tunneling using mTunnel”, Journal of Computer Communication, Volume 21, Issue 15, p.p. 1295-1301, October 1998.

Acknowledgements

This work was sponsored by the Centre for Distance-spanning Technology, CDT, and the European projects MediaSite (IA 1012 UR) and Roxy (Esprit 28410). Thanks should go to Ulrika Wiss and Johnny Widén at CDT, for interesting comments and nice encouragement. We would also like to thank the project Education Direct, which is supported by the Swedish Foundation for Knowledge and Competence Development.

Marratech Pro is a product from Marratech AB, a spin-off company from LTU/CDT. Many thanks for the heroic efforts made by the people at Marratech.

Annex

Pseudo code for the mAudio repair algorithm for 40 ms packets:

```
int cnt = 0;                // Number of consecutive lost packets

byte[] read() {
    if (received(n)) {              // main or redundant packet
        decreaseBuffer();                   // adaptive buffering
        cnt=0;
        return decode(n);
    }
    increaseBuffer();                       // adaptive buffering
    cnt++;
    if (cnt == 1)                   // Repeat with 50% amplitude
        return amplify(n-1, 0.5);
    if (cnt == 2)                   // Repeat with 25% amplitude
        return amplify(n-2, 0.25);
    if (cnt < 10)             // Feed noise with correct amplitude
        return noise(n-cnt);
    return silence;                                // Feed silence
}
```