

Multi-party Distributed Audio Service with TCP Fairness

Milena Radenkovic
Computer Science Department
University of Nottingham, UK
+44 115 9514226
mvr@Cs.Nott.AC.UK

Chris Greenhalgh
Computer Science Department
University of Nottingham, UK
+44 115 9514221
cmg@Cs.Nott.AC.UK

ABSTRACT

Distributed Partial Mixing is an approach to creating a distributed audio service that supports optimisation of bandwidth utilization across multiple related audio streams (e.g. from concurrently active audio sources) while maintaining fairness to TCP traffic in best effort networks. Rate adaptation of streamed audio is difficult because of its rate sensitivity, the relatively limited range of encoding bandwidths available and the potential impact on the end user of rate-adaptation artefacts (such as changes of encoding). This paper describes and demonstrates how our design combines TCP-fairness with the stability that is desirable for streaming audio and other rate sensitive media. In particular, our design combines: a distributed multi-stream management/mixing architecture, loss event and round-trip time monitoring, rate limiting based on a TCP rate equation, tuned increase and decrease strategies and a loss-driven network probing mode. Experimental validation is performed over a wide range of network conditions including against various congesting levels, TCP and independent DPM traffic.

Keywords

Audio, mixing, distributed partial mixing, multi-party audio, congestion control, adaptation, TCP-fairness.

1. INTRODUCTION

This paper is concerned with supporting ‘natural’ audio communication in collaborative environments across the Internet. Recent experience with Collaborative Virtual Environments, for example to support large on-line communities and highly interactive social events, suggest that in the future there will be applications in which many users speak at the same time (see section 2). Such applications may generate large and dynamically changing volumes of audio traffic that can cause congestion and hence packet loss in the network and so seriously impair audio quality. There is no current approach to audio distribution that can combine support for large number of simultaneous speakers with

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Multimedia '02, December 1-6, 2002, Juan-les-Pins, France.
Copyright 2002 ACM 1-58113-620-X/02/0012...\$5.00.

TCP-fair responsiveness to congestion.

This paper describes a model for audio distribution called Distributed Partial Mixing (DPM) that dynamically adapts both to varying numbers of active audio streams in collaborative environments and to congestion in the network. The particular focus of this paper is the way in which we have sought to combine fairness to TCP with the relative stability of bandwidth that is appropriate for rate-sensitive data flows such as audio.

Section 2 explains the problem domain and identifies some key design and use decisions that have shaped our approach. Section 3 describes the general approach to multi-party audio distribution used in Distributed Partial Mixing and section 4 compares it to previous work. Section 5 explains the specific design decisions and experimentally-derived constants used in our prototype of Distributed Partial Mixing to achieve fairness to competing TCP traffic in the face of congestion. Section 6 presents results from experiments with this implementation, considering adaptation and fairness to TCP and other DPM traffic. Section 7 gives final conclusions and identifies potential generalizations of this work.

2. MULTI-SOURCE AUDIO

Most applications of streaming networked audio to date have concentrated on applications in which there is typically only one (or a very small number) of simultaneous audio sources. These applications include small-scale conferencing and broadcasting. They are efficiently supported by relatively established networking techniques such as network-supported multicasting [17].

Due to the nature of these applications, some protocol developers have explicitly disregarded the possibility of multiple simultaneous speakers and have focused on supporting static, controlled, small scale and slow pace dynamic models of interaction. For example, the Real Time Protocol (RTP) standard for continuous media traffic – such as audio – on the Internet includes sophisticated algorithms to control the amount of management traffic placed on the network, but assumes that audio traffic will not be a problem:

“For example, in an audio conference the data [audio] traffic is inherently self-limiting because only one or two people will speak at a time...” [[25], section 6.1]

Similarly, the multicast backbone network (MBone) [17] guidelines for use, and the available resources (at least historically), assume that each audio session will only have one active speaker at a time (indeed, many of the larger MBone sessions have been effectively broadcasts rather than conferences, e.g. feeds from the NASA shuttle launches).

In contrast, there are many situations in everyday life in which large groups of people “speak” at the same time. Audiences and crowds of spectators at performances and sports events provide the most extreme examples, with thousands of people engaging in activities such as cheering, chanting and singing together. However, any relaxed social setting that involves sizeable groups of people is also likely to involve significant numbers of simultaneous speakers; for example, interruptions, non-verbal speech cues [7], shouting out answers to questions, dynamically forming conversing sub-groups within a large space [15].

Similarly, emerging large scale networked applications such as collaborative virtual environments (CVEs) can support large on-line communities and highly interactive social events such as multi-player games and inhabited television [9]. For example, recent work on inhabited television has focused on enabling public participation in on-line TV shows within shared virtual worlds [9]. As part of a public experiment in inhabited TV called Out of This World (OOTW), patterns of user activity, including audio activity, were studied by statistically analysing system logs [8]. The results showed that overlapping audio transmissions from several participants were common for this event. The results showed that overlapping audio transmissions from several participants were common for this event. Indeed, during a 45 minute show, there were several minutes when all 10 of the participants were generating audio traffic at the same time. Similar analyses of patterns of audio activity in other CVE applications and platforms, for example virtual conferencing in the DIVE system [6], have also revealed significant periods when several participants are simultaneously generating audio traffic.

Not all of this activity is necessarily verbal; it may also include non-verbal utterances and background noise. However, we argue that non-verbal utterances form a significant part of human communication. For example, studies of social interaction in CVEs have noted how communication in a virtual world can be influenced by events in participants’ local physical environments [1]. Obtaining awareness of these events through overheard background audio might help participants account for actions in the virtual world. Similarly, when discussing patterns of usage for the Thunderwire audio media-space, Hindus *et al.* note that it might be better to consider the number of live microphones, rather than the number of active participants [13].

For these reasons we have rejected approaches to audio management that restrict the use of the audio channel, including explicit and implicit floor control, and explicit control actions (e.g. push to talk). Instead we assume that our ideal audio service will use silence suppression or even continuous transmission, and we have designed our audio service to cope with potentially many simultaneous audio streams (although, as described in the conclusions, the same approach may also be applied in other domains and to other forms of streaming data from multiple sources).

We have chosen the following criteria to guide our design and evaluation:

1. Support for natural audio communication between users (i.e. open microphones).
2. Support for maintaining acceptable audio quality (especially minimising packet loss but also considering delays, even with network and host heterogeneity, and congestion).

3. Support for tailored audio presentations for each user (i.e. to allow each individual receiver to have its own mix or spatialisation of the various audio streams).
4. Support for heterogeneous unmanaged large-scale networks, end systems and wide area user distribution (e.g. Internet).
5. Support for efficient distribution of audio streams in the network.
6. Support for adaptability and responsiveness of the audio traffic in the network including inter-protocol fairness

The first three criteria address user issues whereas the last three criteria are network oriented.

3. DISTRIBUTED PARTIAL MIXING

3.1 Mixing

As the name suggests, mixing (of audio streams) plays a central role in our approach. Mixing multiple streams involves digitally summing the corresponding audio samples from the incoming streams and then normalising the result. In general, performing mixing (e.g. in a server, filter or proxy) in the network is the only mechanism that can actually decrease the number of audio streams. Mixing can thus drastically reduce network traffic for many simultaneous speakers. This makes it efficient according to its support for efficient distribution of audio streams in the network (criterion 5). Mixing also imposes no constraints on individual speakers (compared with floor control’s “gagging” of users). This means that the system’s view of “speaking” is the same as the user’s. This makes it effective according to supporting the most natural audio communication (criterion 1).

However there are also negative aspects to mixing, which mainly concern the user’s point of view:

- The mixed signal is noisier than the separate streams (and/or has reduced dynamic range) since it includes the noise from all of the separate streams in fewer bits.
- Mixing may not work well with some ultra-low-bandwidth CODECs due to their compression algorithms;
- Mixing introduces additional delay compared to direct communication, as audio streams have to be processed by mixers en route from speakers to listeners; and
- Mixing introduces more components (servers) or complexity (e.g. client roles in peer-based systems) in the network, with corresponding hardware, organisational and real-time control requirements.

A further major disadvantage of mixing considered in this paper is that mixing introduces loss of potential spatialisation and other aspects of individual listener control over individual audio streams (criterion 3). Previous research has shown that providing spatialised audio is a significant factor in engendering a sense of presence in a virtual environment. For example, Hendrix and Barfield conducted experiments to compare the presence or absence of spatialised sound alongside the stereoscopic display of a virtual environment [12]. They found that the addition of spatialised sound did significantly increase the sense of presence in the virtual environment. The mixed stream does not retain any distinction between the component signals (the effect is similar to that when the “mono” button on a stereo hi-fi or radio is pressed, which mixes the left and right audio channels together). Note that

mixing is irreversible and the individual streams cannot be split at a later stage.

Consequently our design has to respond to two primary challenges. First, how can the audio streams be mixed in a way that is adaptive and friendly towards other traffic in the network, supports heterogeneous networks and minimises the packet loss experienced? Second, how can we do no more mixing than is absolutely necessary i.e. keeping the maximum possible number of independent streams while having an efficient aggregation of audio streams?

3.2 Distributed Partial Mixing

Consider a multi-party audio application in which there are a number of audio senders and receivers ('clients') distributed around a network, and where each receiver wishes to receive the audio streams from all of the senders (senders can send either via unicast or multicast). Distributed partial mixing can be performed by a number of Partial Mixing components that might already exist in the network (e.g. as dedicated servers) or as additional roles played by other clients. The clients typically communicate via these Partial Mixer components, sending their audio to – and receiving other clients' audio from – a 'nearby partial mixer. The partial mixers in turn form a fully connected network (e.g. a tree) to achieve end-to-end distribution of the audio streams.

Partial mixing extends the traditional concept of mixing to be more dynamic and flexible. Unlike total mixing (which is based on mixing the whole set of received streams into a single output stream) partial mixing dynamically chooses to mix only a subset of the available audio streams at any given time and forwards this along with the rest of the un-mixed streams. In this way instead of producing a single output stream in all cases, partial mixing produces varying numbers of streams in different situations.

Figure 1 illustrates a scenario in which there are multiple partial mixing stages between the end systems. The figure shows the streams that get mixed at different stages in the system. For example, the network between Partial Mixer (PM) A and B is experiencing low loss rates, and so all five streams (*a-e*) are forwarded without mixing. However the network between PM C and D is experiencing higher loss rates, and is mixing the three audio streams (*f-h*) into a single combined stream. Client *j* is receiving a single fully mixed stream from PM E, perhaps because of tail-link congestion, or because of client processing limitations. Whereas client *i* is receiving maximally separated audio streams (only mixed where this was forced earlier in the network). Note that in this example partial mixers A and B are client machines, and partial mixers C, D and E are dedicated servers.

In order to decide which and how many streams to mix each partial mixer needs to exchange control and feedback information (describing network conditions and application preferences and end system bottlenecks) with other partial mixers, end systems or network components. Therefore there are always two channels of communication between the neighboring partial mixers: the channel that contains the audio streams as well as the channel that contains the control and/or feedback information. Figure 1 shows only half duplex communication of audio streams and controlling data for reasons of clarity and simplicity. However, the reader should assume that the communication is full duplex

In this paper we focus on a single representative connection between two DPM units. However, we have also explored various organization and deployment schemes for DPM including: a static, centralised, subscription-based DPM service suitable for fully managed networks, and a fully distributed, self-organised DPM service (e.g. based on self-organised transcoding scheme proposed by Kouvelas et al in [16]) suitable for unmanaged networks (such as the Internet). The first model has very limited support for dynamic user membership and changes in network topology. For example, partial mixers C, D and E in Fig. 1 are assumed to be pre-configured servers positioned on well-suited places in the network. In this case, central authority is assumed to have global knowledge of the whole system at every point of time including link reliability, link bandwidth, link cost, possibility of bottlenecks, time delays, topological delay, complete information about subscribed clients, their locations and other properties. The second model allows users to join and leave the application at any time, and dynamically responds to changes in the underlying network topology. For example, partial mixers A and B in Fig. 1 can be clients that are elected by other clients (on-demand) to perform partial mixing. The basic idea adopted here is to use self-organisation to form groups out of co-located receivers, which share loss events, and to provide local adaptation through the use of a DPM service. The idea behind self-organisation usually is to have the end clients self-organise into a multi-level hierarchy of multicast groups, where each individual group corresponds to the homogeneous regions within the heterogeneous multicast tree regions. In this way the resulting topology is congruent with the underlying multicast tree, and one large heterogeneous and difficult problem is reduced to many small, homogeneous and simple sub-problems. The metric used to identify groups in this particular model is based on shared loss patterns among end clients. This and other similar metrics are discussed in greater detail in [16], [23]. More detailed discussion and evaluation of large scale dynamic DPM topologies is beyond the scope of this paper.

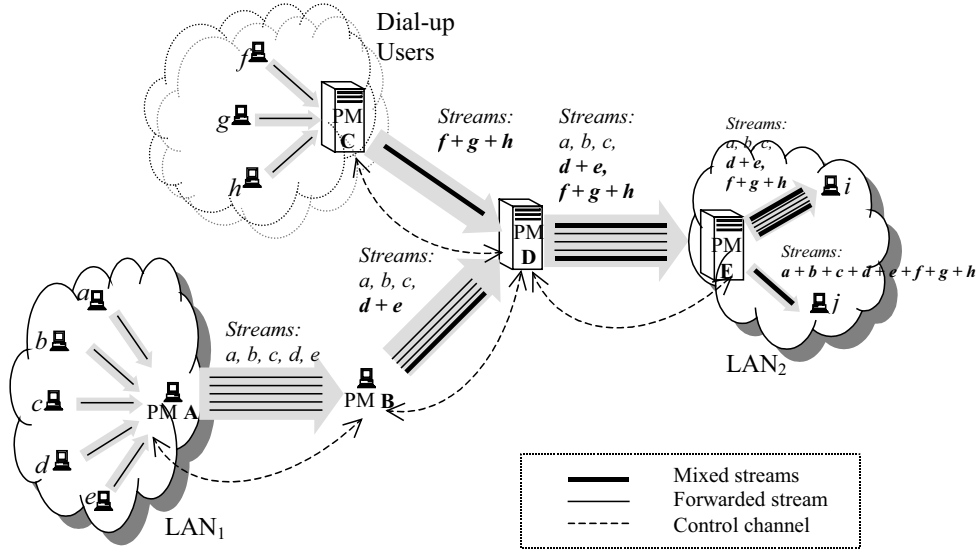


Figure 1. Example Distributed Partial Mixing scenario

3.3 Functional Architecture of a Partial Mixer

The functional architecture of a partial mixing component is shown in figure 2. It comprises the following elements:

- Receiver, which receives audio streams from other audio components in the network, including clients and other partial mixers (and gives feedback to the sending entities).
- Selector, which determines which of the incoming audio streams should be forwarded as is and which should be combined into sub-mixes before forwarding in order to meet a bandwidth target specified by the Rate Adapter.
- Mixer, which takes designated audio streams from the selector and produces mixed audio stream(s) from them.
- Transmitter, which sends mixed and forwarded audio streams on to other components in the networks (clients and/or other partial mixers), and which also receives congestion-related feedback messages (e.g. ACKs).
- Congestion monitor, which continuously estimates Round Trip Time and packet loss event rate for the transmitted audio streams, based on the feedback received by the transmitter.
- Rate Adapter, which uses the current estimates of network characteristics and the partial mixers' history to determine a target sending bandwidth which is adaptive to congestion and fair to TCP.
- Database, which provides additional information for the selection process, e.g. guiding the Selector's

prioritization of the incoming audio streams (to decide which should be mixed first).

- Note that there may be multiple instances of this functionality in each Partial Mixing component, potentially one for each client and neighbouring Partial Mixer (each of which may be experiencing unique network conditions and require its own specific mix).

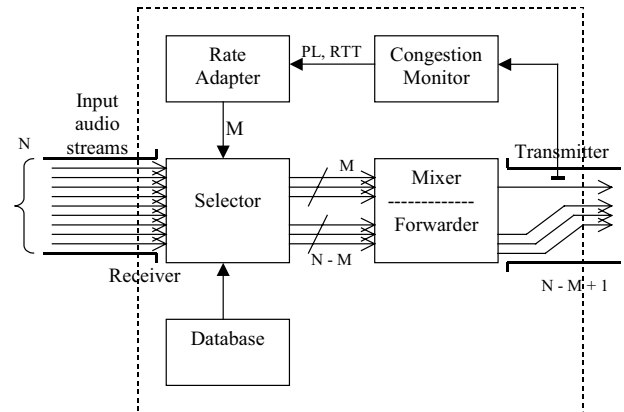


Figure 2. Functional architecture of a partial mixer.

This design separates the Rate Adapter – which determines the outgoing bandwidth target – from the Selector – which chooses exactly which of the possible streams should be mixed together to meet this target. This separation is important because the Rate Adaptation process is essentially application independent, and is responsible for responding to changing network conditions, whereas the Selection process may be highly application dependent, and should also take into account application and user requirements. For example, in one application there may be an identifiable ‘chairperson’ or ‘commentator’, whose audio should be maintained separately as long as possible. In another application there may be distinct ‘teams’, each of which can have its audio mixed together with relatively little impact on the user’s

experience. In our work to date – and in this paper – we have focused on the application independent networking aspects of DPM.

4. Related Work

This section compares the approach of Distributed Partial Mixing with other approaches to audio distribution.

4.1 Single Stream Techniques

There are a range of techniques that are concerned with efficient distribution of single audio streams. These include network supported multicasting, e.g. as used for various multiparty audio and video tools (e.g. vat [14], RAT [11]), layered multicast, which is based on layered media encodings [18], and various established compression algorithms.

There are also various proposals that – like DPM – include processing elements at intermediate points within the media distribution tree. Kouvelas et al in [16] present a self-organised scheme to form groups out of co-located receivers with bad reception, where receivers with better reception can provide a customised transcoded version of the session stream. Pasquale et al in [20] propose the use of self-propagating filters over a dissemination tree whereby filters (expressing requirements) propagate from receivers back towards the sender. [27] proposes using filters primarily to reduce data transmitted to receivers that either cannot use it or do not wish to use it. In the proxy model proposed by [5], media gateways are situated at strategic points within the network and actively transform media streams to mitigate bandwidth heterogeneity and client diversity.

However these techniques affect only the bandwidth of a single stream and do not operate across multiple streams: they must deliver all of the audio streams separately to the end user. Consequently, none of these techniques can fully address the issue of many simultaneously active sources (senders). Therefore, the amount of audio data and processing for these techniques rises proportionally with the number of simultaneous speakers. Furthermore, [16] argue that although layered (or adaptive) encoding is possible for sampled speech, the transmission bandwidth range is significantly more restricted than for video, limiting the potential for bandwidth adaptation of a single audio stream.

4.2 Multi-stream Techniques

There are also a number of approaches that are based on some form of mixing to reduce the number of audio streams in the network.

In a centralised (total) mixing architecture every audio stream is sent to a centralised audio mixer that mixes multiple streams into a single stream that is then redistributed to each listener. Total mixing requires that each listener (and their nearby network) handle only one audio stream. However, this prevents each listener from creating their own personal mix. Also, the server becomes a potential bottleneck in the system, since it (and its nearby network) has to deal with, mix and distribute $O(n)$ streams.

Client-specific mixing has been proposed in the SPLINE system to allow users with low bandwidth connections to access an audio-graphical multi-user virtual world [26]. One or more low-bandwidth users connect to a specialised access server that interfaces to the main system (which uses peer-to-peer multicast audio). This access server provides a customised audio mix for

each connected user, which is sent directly to them. Unlike the previous total mixing solution this approach does provide a separate mix to each listener, giving greater flexibility, though at the cost of more work for the access servers (since sending two copies of the same mix is easier than computing and sending two different mixes). However, this approach still requires (like total mixing) that the access servers deal with and mix all of the available audio streams, and it assumes that the network between the access servers can cope with all of the audio streams. This approach can be viewed as a subset of DPM.

In order to support applications such as a tele-orchestra, which involve large numbers of simultaneously active audio streams, Rangan in [22] proposed a hierarchical mixing architecture, and showed that it is an order of magnitude more scaleable than purely centralised or distributed architectures. In this mixing hierarchy, participants constitute leaf nodes, and the mixers are non-leaf nodes. The mixer that is at the root of the hierarchy forwards the final mixed packets to each of the leaf nodes. Like centralised mixing, this approach prevents each listener from creating their own mix(es) since all of the streams will be mixed all of the time. This is not very well suited for interactive and collaborative applications where each user should have the flexibility of controlling and mixing independent audio streams. This approach also has no awareness of the underlying network conditions, and relies only on statically configured mixers.

In terms of the design criteria for DPM, previous approaches perform ‘excessive’ mixing, even in the presence of spare bandwidth, and similarly lack explicit support for network monitoring and adaptation.

5. DESIGNING FOR TCP-FAIRNESS

A significant part of our design and evaluation of partial mixing has been concerned with achieving bandwidth adaptation, and in particular fairness to competing TCP traffic. This section describes the approaches and algorithms that we have adopted, and the control constants that we have refined experimentally. The following section shows the results of experiments with our current implementation.

The main challenge has been to achieve a level of adaptation and TCP-fairness that balances responsiveness against stability. Although we can use mixing as a mechanism to modify sending rate we do not want to change the level of mixing more often than necessary, because it is likely to introduce perceptible artifacts (such as the loss or gain of audio separation, and changes in relative volume) that may adversely affect the end-users’ experience.

We consider the following issues in turn: round trip time monitoring and estimation; loss rate monitoring and estimation; rate adaptation method and management; and self-limitation.

5.1 Round trip time monitoring and estimation

Our prototype of distributed partial mixing performs round trip time estimation, packet loss detection and loss rate calculation in the sender. The receiver explicitly acknowledges every packet received by returning the acknowledgements (potentially piggybacked on audio packets as part of the audio streams going in the reverse direction) to the sender. Each sent packet and each ACK also include a sending time stamp.

When estimating the round trip times for the purpose of adapting to network conditions, the processing and buffering times at end systems needs to be subtracted from the total round trip time, and only network distance calculated. This is achieved by compensating the sending time stamp in the ACK to offset any known time spent at the receiver (e.g. waiting for a message on which to piggy-back). As in TCP, the average RTT is maintained as an exponential moving average with a constant of 7/8.

5.2 Loss rate monitoring and estimation

We use a TCP-like timeout-based mechanism to detect packet loss. All sent packets are marked with consecutive sequence numbers. When a packet is sent a timeout value for this packet is computed and an entry containing the sequence number and the timeout value is inserted into a list and kept there until the packet delivery is acknowledged or the timeout expires. If the timeout expires before the packet is acknowledged, the corresponding packet is considered to be lost. In order to adapt to varying and unpredictable network conditions, the timeout is not fixed, but computed based on the algorithm for TCP timeout computation [21]. Timeout-based packet loss detection allows more consistent and timely packet loss detection compared to, for example, identifying gaps in the packet sequence numbers. The timeout-based approach also gives stable results, even in the presence of 100% packet loss (all packets time out).

Our prototype of distributed partial mixing uses the Weighted Loss Interval Average (WLIA) approach for computing the packet loss rate. This approach was first introduced in [4]. It relies on using loss events and loss intervals for correct computation of packet loss rate and is in accordance with how TCP performs packet loss calculations. A loss event is defined as a number of packets lost within a single RTT (losses that follow an initial loss within a round trip time are explicitly ignored). The number of packets between two consecutive loss events defines a loss interval. The detailed proof and analysis of this method can be found in [4].

The method takes a weighted average of the last n loss intervals, with equal weights for the most recent $n/2$ intervals, and smaller weights for the older intervals. The value n for the number of loss intervals used in calculating the loss event rate determines the speed in responding to changes in the level of congestion, i.e. the sensitivity to noise of the calculated loss rate depends directly on the choice of n . [4] argues that a value of 8, with the most recent four samples equally weighted, is the lower bound that still achieves a reasonable balance between resilience to noise and responding quickly to real changes in network conditions [4]. The Weighted Loss Interval Average approach also employs history discounting to allow a more timely response to a sudden decrease in congestion. History discounting is used after the identification of a particularly long interval since the last dropped packet in order to smoothly discount the weight given to older loss intervals. History discounting is described in more detail in [4]. Note that without some form of history discounting the method would have rapid response only to increases in the congestion but be slow to respond when congestion decreases.

The primary reason for measuring loss event rates rather than loss rates is that this is more consistent with the way TCP responds to loss (therefore it models the best behaviour of conformant TCP implementations) while at the same time being relatively stable and resilient to noise. Using loss intervals to determine loss rate,

rather than attempting to measure loss rate directly (e.g. with a windowing technique), gives more consistent estimates of loss rate and therefore smoother throughput (see [19]).

5.3 Rate adaptation

For applications that compete in the best-effort Internet with TCP and require relatively smooth changes in sending rate, being responsive to network congestion over longer time period (seconds, as opposed to fractions of a second) is more important than the opportunistic use of increases in the available bandwidth. The particular rate adaptation approach chosen and implemented in distributed partial mixing is an equation-based approach based on the model first proposed in [4]. Equation-based rate control mechanisms generally use a control equation that explicitly gives the maximum acceptable sending rate as a function of the packet loss and round trip times in response to the feedback from the receiving partial mixing. Hence, the sending partial mixer directly adjusts its transmission rate guided by this control equation in response to the measured RTT and packet loss rates. For distributed partial mixing that competes in the best effort Internet with TCP, the appropriate control equation is the TCP response function characterizing the steady-state sending rate of a TCP flow as a function of round-trip time and steady-state loss event rate. This equation is formulated as follows:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO} \left(3\sqrt{\frac{3p}{8}} \right) p \left(1 + 32p^2 \right)}$$

where T is the upper bound on the sending rate, s is the packet size, R is round-trip time estimated as above, p is the loss event rate estimated as above, and the t_{RTO} is the TCP retransmit timeout value [4].

Implementing this TCP response function when distributing UDP audio streams ensures that UDP-based distributed partial mixing competes fairly with TCP over long time scales. This TCP response equation is believed to adequately model TCP throughput even at high loss rates (modeling timeout behavior), and so is more suitable than simpler models such as [2].

During under-load periods the sending rate is increased linearly (i.e. one stream at a time). During over-load periods the sending rate is immediately reduced to no more than the TCP-equivalent rate. Rate adaptation is performed once every 6-8 RTTs (which limits the rate at which sending bandwidth can increase to one audio stream per 6-8 RTTs). The rate adaptation equation is applied every several RTTs rather than every single RTT in order to minimise the frequency of rate changes. Frequent rate changes in DPM are undesirable for the end user because they have a direct impact on their perceived QoS in terms of the levels of spatialisation. In addition to this, [3] recommends the equation to be applied every several RTTs and not every single RTT if long-term TCP behaviour is being considered. The exact values were determined through extensive experimentation to achieve a reasonable trade-off between TCP fairness, smoothness and responsiveness of distributed partial mixing.

5.4 Self-limitation

Self-limiting behaviour is a classic problem with rate-based congestion control schemes. In window-based schemes the source stops once it has a full window worth of data on the fly. This

property makes window-based schemes intrinsically stable. However, if the source allows retransmission beyond the current window stability is lost, and so the number of retransmitted packets must also be limited. Rate-based schemes need to find some approach analogous to a finite window to bound the volume of outstanding data in the network.

Distributed partial mixing achieves self-limitation by using a timeout mechanism for loss detection, as described above. In the extreme case, when no acknowledgements are received (e.g. when the connection goes down, the network gets partitioned, the receiving end crashes, or during very severe congestion times) or when the packet loss rates are above 20% (and therefore unusable for audio), the transmission rate drops to a probing mode. Distributed partial mixing sends one dummy packet per second while it is in the probing mode. Dummy packets must be sent in order to allow distributed partial mixing to resume sending when the connection or the end system recovers (when these dummy packets begin to be acknowledged).

One packet per second is a faster probing rate than that of some other protocols. This is chosen to suit the nature of the increase/decrease policy and the decision frequency of distributed partial mixing. Note that distributed partial mixing goes directly to the probing mode from sending one stream, it needs to have a bandwidth of one or more streams to resume sending audio, and it tries to do this not more often than every 6-8 RTTs. Consequently, it is relatively easy for distributed partial mixing to go to probing mode and relatively difficult for it to recover. More frequent probing packets allow faster receipt of acknowledgments and thus quicker sending rate recovery. A slower rate for the probing mode excessively delays distributed partial mixing's recovery from the probing mode.

6. RESULTS

This section gives some details of our prototype implementation of Distributed Partial Mixing, as described above, and presents our results from experiments testing this prototype against competing non-adaptive and adaptive traffic (including TCP and other DPM traffic).

6.1 Implementation and Scenario

Distributed partial mixing was developed as an extension of the audio service in MASSIVE-3[10]. MASSIVE-3 is a collaborative virtual environment (CVE) platform that enables interaction among multiple simultaneous distributed users in a 3-D virtual world where all users have real time audio communication and its source is freely available for research.

Each MASSIVE-3 user has their own MASSIVE-3 client program to access the shared virtual world generated by the MASSIVE-3 world server. Each user's client program gives them a 3D view of the virtual world and allows them to move around within it. Each user also has their own local audio server that interfaces to the audio hardware on their computer, allowing them to talk to and be heard by the other users. Each user's MASSIVE-3 client controls their local audio server, using information in the virtual world to determine how it should send, receive and render audio streams (e.g. according to other users' positions within the virtual world). This audio server is a general-purpose audio server process developed specifically for MASSIVE. This audio server has been enhanced to include Partial Mixing functions as described above, and each instance of the audio server can act as a client and/or a

Partial Mixer. MASSIVE-3 has also been modified to provide appropriate session support for distributed partial mixing.

The experimental scenario is shown in figure 3. "LAN 1" hosts eight audio sources (which are emulated MASSIVE-3 users, generating continuous audio traffic) and a Partial Mixer. "LAN 2" hosts a second Partial Mixer, and a single receiving user. The two LANs are connected via a simulated WAN, realized using the Dummynet tool [24]. The audio encodings are all 8KHz, 8bit, Ulaw, mono. The emulated WAN has a bandwidth limit of 635000 bits/s (approximately eight audio streams), a delay of 70ms and a default buffer size of 600000 bits. The results are based on off-line analysis of packet logs captured using tcpdump.

In this scenario we are exercising a single representative connection between two DPM units. Since DPM is applied to each connection independently, this scenario is sufficient to test and validate the proposed approach in terms of all the six criteria given in section 2 even in large scale deployments.

6.2 Competition with non-adaptive traffic

In this experiment we wish to show the utility of distributed partial mixing. We introduce nine levels of competing (congestion-inducing) UDP traffic: 0, 78400, 156800, 235200, 313600, 392000, 470400, 548800 and 627200 bits/s. Each congestion stage lasts for about five minutes. This competing traffic is non-responsive (i.e. does not change in the face of packet loss).

Figure 4 shows the rate of packet loss experienced by the audio traffic and figure 5 shows the degree of spatialisation maintained (i.e. the number of independent audio streams delivered). Quantitative evaluation based on these two criteria was chosen because these two criteria are both clearly related to the end-user's experience of the system, and can also be objectively determined from measurements of the system in use (e.g. numbers of packets per second). Also shown for comparison are the same results for non-adaptive audio distribution strategies that (a) forward all streams all of the time (as multicast would) and (b) mix all streams all of the time).

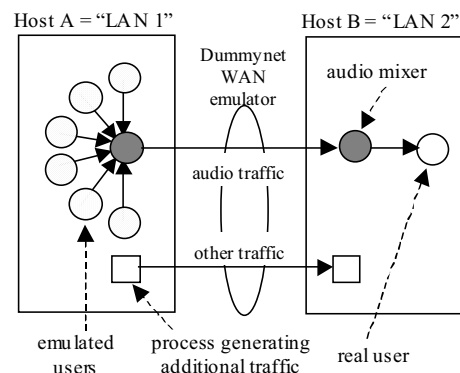


Figure 3. Experimental scenario for DPM testing.

Note that the total forwarding approach experiences packet loss rates in excess of 15% with only 156800 bits/s of additional traffic, beyond which point the received audio is likely to be useless (incomprehensible). With approximately 548800 bits/s of competing traffic, distributed partial mixing has fallen back to total mixing, with only a single stream sent over the WAN. For higher congestion levels, the graph shows that distributed partial

mixing oscillates between sending a single stream and being in the probing mode. The graph shows not only the mean value of the number of independent streams sent by distributed partial mixing within each congestion level but also the variation in bandwidth within two standard deviations of this mean value. This shows, with 95% correctness, that the maximum deviation from the mean value is approximately one stream for the test period of 5 minutes. This shows that distributed partial mixing has relatively small oscillations (due to adaptation) and therefore has reasonably stable behaviour.

Figure 6 show the temporal behaviour of distributed partial mixing as congestion levels on the link increase from zero to the full link. This figure shows the responsive and stable behaviour of distributed partial mixing while still preserving relatively high numbers of independent audio streams for each congestion level.

6.3 Competition with adaptive traffic

These experiments use essentially the same scenario, but the non-adaptive competing traffic is replaced by one TCP flow, increasing numbers of TCP flows, and another DPM flow, respectively.

Figure 7 shows the bandwidth consumed by the audio traffic sent by the partial mixer as well as a single competing TCP flow, as a function of time. Initially – with no competing traffic – the number of audio streams sent rises linearly and rapidly. After just a few seconds, it fills the link. At roughly 72 seconds a competing TCP flow starts. After 10 seconds of slow start behaviour this enters its own rapid steady state oscillation, continually probing available bandwidth. The offered traffic exceeds the capacity of the link, causing an increase in round trip time (due to buffering delays) and resulting in increased packet losses. The TCP-fair bandwidth calculated by DPM then falls, and the number of streams moderates accordingly. Distributed partial mixing takes its fair share of the bandwidth (one half or 3-4 streams) oscillating much less than TCP.

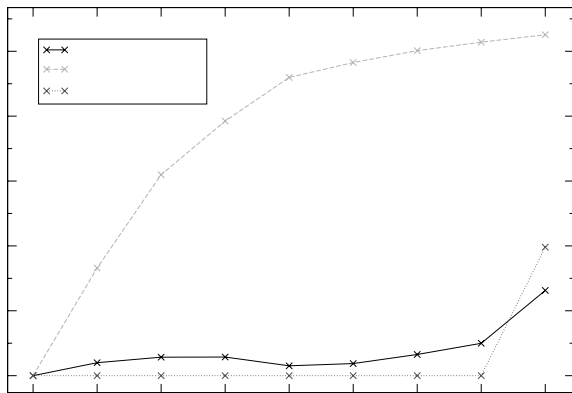


Figure 4. Rates of packet loss for varying levels of competing non-adaptive traffic.

Figure 8 shows 20-second average bandwidths for distributed partial mixing and for the competing TCP traffic (so that the TCP oscillations are smoothed out). The graph shows how DPM moderates its bandwidth usage as first one, then a second, and finally a third TCP flow is started.

To demonstrate fairness between multiple distributed partial mixing-capable applications, two such applications are started in the same LAN, each with eight users. Figure 9 demonstrates this scenario. The first DPM application is started in the beginning, and it fills in the whole link. At times 260sec, the second DPM application is started (with eight players, all simultaneously active in the audio medium). The second application would like to send all the eight audio streams, but since the link is already full with the first application sending its eight streams the second application performs partial mixing and very quickly stabilizes at half of the link bandwidth. The first application also decreases its sending rate and stabilizes at half of the link bandwidth (at around 4 streams or 313600bits/s). At time 820sec, a TCP flow is added to the link already taken fully by the two DPM applications. The graph shows that the two DPM applications and the TCP flow each adapt their rates to stabilise quickly at their fair share of the link by taking approximately one third of the link (each 2-3 streams or 192000bits/sec).

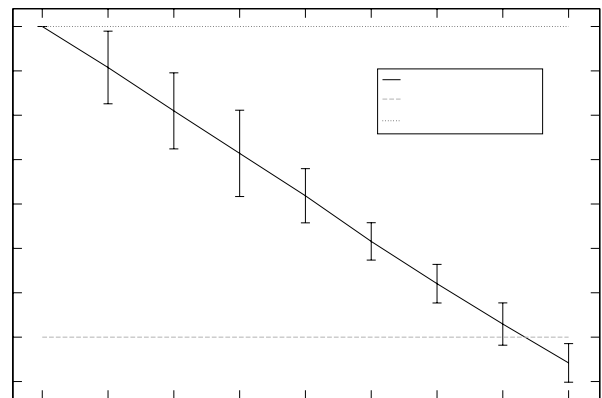


Figure 5. Degree of spatialisation for varying levels of competing non-adaptive traffic.

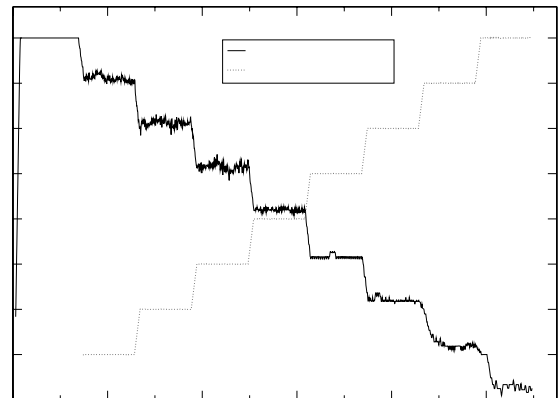


Figure 6. Temporal behaviour of DPM against non-adaptive traffic.

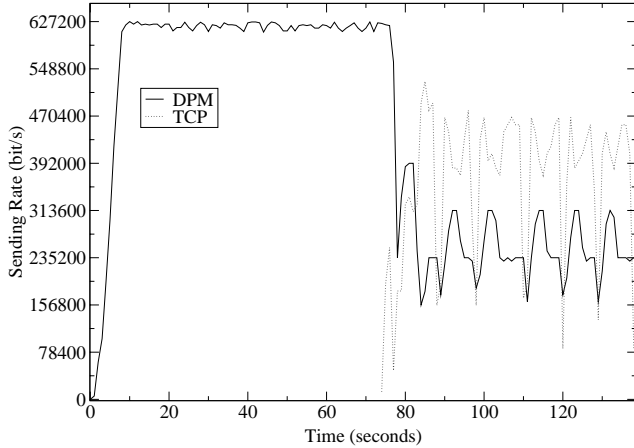


Figure 7. Detailed view of DPM vs TCP.

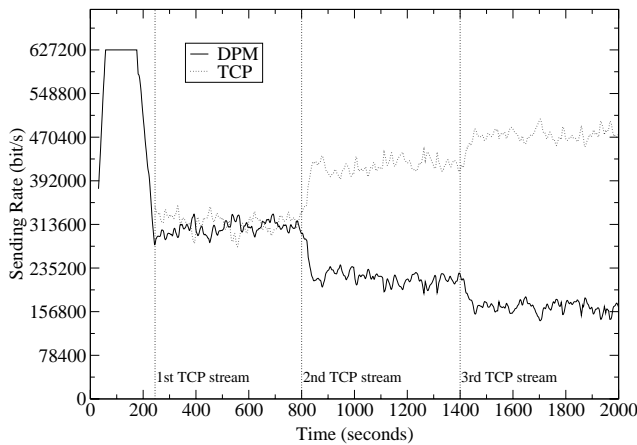


Figure 8. TCP-fair audio distribution showing medium term (smoothed) behaviour with competing TCP flows.

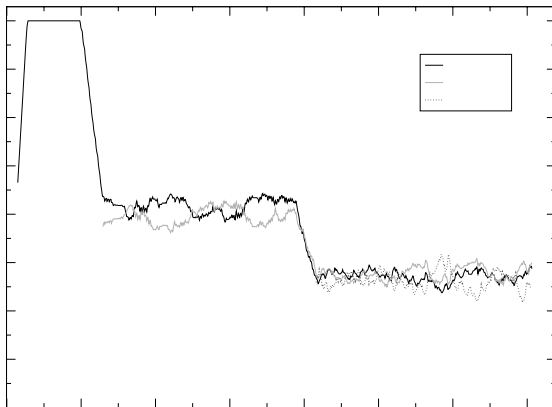


Figure 9. Fair audio distribution showing medium term (smoothed) behaviour with competing DPM and TCP flow.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have argued that there are – and will increasingly be – networked audio applications with large numbers of simultaneously active audio sources. We have described the Distributed Partial Mixing approach to distributed audio support

that uniquely supports multi-stream congestion control and bandwidth allocation. We have also explained the specific design and implementation choices that we have adopted in order to achieve TCP fairness. The results from our current prototype show the adaptation and dynamic characteristics that have been achieved over a wide range of network conditions. We have demonstrated that the chosen design for DPM can sustain high numbers of independent audio streams at acceptably low levels of packet loss while being responsive and fair towards TCP and other distributed partial mixing traffic. The results from the experiments against adaptive traffic demonstrated that distributed partial mixing co-exists acceptably well when sharing congested links with other adaptive traffic. In the long-term, distributed partial mixing consumes its fair share of the bandwidth when competing with multiple TCP flows, as well as other distributed partial mixing flows. In the short-term, DPM also consumes its fair share while oscillating significantly less than TCP. Experiments against various levels of non-adaptive traffic demonstrate that at low, moderate and high steady state congestion levels, distributed partial mixing has a relatively high mean throughput and small variations of throughput, while preserving the packet loss under 10%. Stable packet loss estimation, a decision frequency of 6 RTT, and a non-opportunistic linear increase policy towards the expected value for the sending rate result in stable and smooth sending rates for DPM that are suitable for multimedia applications.

There are many other issues involved in the wide-area realization of such a scheme, especially the establishment and coordination of the various Partial Mixing elements and the clients. We have explored both highly managed and self-organized approaches to these issues and hope to present this work on another occasion.

We believe that the approach embodied in DPM, although initially motivated by and demonstrated with distributed audio, may also be applied in other contexts. Considering the functional architecture of a partial mixer, it would be relatively straightforward to generalize the selection and mixing components to operate over other media flows (e.g. video, awareness events). Even where there is no direct analogy for mixing (i.e. no standard lossy method for combining multiple data flows) the DPM approach can still perform application-specific balancing of the various data flows in a way that is not possible with end-to-end congestion control of the individual data flows.

8. ACKNOWLEDGEMENTS

We gratefully acknowledge the support of BT and the Engineering and Physical Sciences Research Council.

REFERENCES

- [1] Bowers, J., Pycock, J., O'Brien, J, Talk and Embodiment in Collaborative Virtual Environments, In Proceedings of CHI'96, 1996.
- [2] Floyd, S., Fall, K., Router Mechanisms to Support End-to-End Congestion Control, Technical Report, 1997.
- [3] Floyd, S., Fall, K., Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, Volume 7, Number 4, IEANEP (ISSN 1063-6692), 458-472, August 1999.

- [4] Floyd, S., Handley, M., Padhye J., Widmer, J, Equation-Based Congestion Control for Unicast Applications, In Proceedings of SIGCOMM 2000, August 2000.
- [5] Fox, A., Gribble, S.D., Brewer E.A, Amir, E., Adapting to Network and Client Variability via On-Demand Dynamic Distillation, In Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, pages 160--170, Cambridge, MA, October 1996.
- [6] Frécon, E., Greenhalgh, C., Stenius, M., The DiveBone: An Application-Level Communication Infrastructure for Internet-Based CVEs, In Proceedings of VRST'99, 58-65, London, 1999.
- [7] Goodwin, C., Notes on story structure and the organization of participation, in Atkinson, J., Heritage, J. (eds.), Structures of Social Action: Studies in Conversation Analysis, 225-46, CUP, 1984.
- [8] Greenhalgh, C., Benford, S., Craven, M., Patterns of Network and User Activity in an Inhabited Television Event, In Proceedings of VRST'99, ACM, 58-65, London, Dec 20-22 1999.
- [9] Greenhalgh, C., Benford, S., Taylor, I., Bowers, J., Walker, G., Wyver, J., Creating a Live Broadcast from a Virtual Environment, In Proceedings of SIGGRAPH'99, 375-384, 1999.
- [10] Greenhalgh, C., Purbrick, J., Snowdon, D., Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring, in Proceedings of CVE'2000, ACM, 119-127, San Francisco, 2000.
- [11] Hardman, V., Sasse, A., Handley, M., Watson, A., Reliable Audio for Use over the Internet, In Proceedings of INET'95, Hawaii, 1995.
- [12] Hendrix, C., Barfield, W., Presence Within Virtual Environments as a Function of Visual Display Parameters, PRESENCE, 5(3), 274-289, MIT Press, 1996.
- [13] Hindus, D., Ackerman, M., Mainwaring, S., Starr, S., Thunderwire: A Field Study of an Audio-Only Media Space, In Proceedings of CSCW'96, USA, ACM Press, 238-247, 1996.
- [14] Jacobson, V., McCanne, S., Vat - LBNL Audio Conferencing Tool, URL: <http://www-nrg.ee.lbl.gov/vat/>, 1999.
- [15] Kendon, A., A Description of Some Human Greetings, in Conducting Interaction: Patterns of Behavior in Focused Encounters, 153-207, 190, Cambridge University Press, Cambridge, UK, 1990.
- [16] Kouvelas, I., Hardman, V., Crowcroft, J., Network Adaptive Continuous-Media Applications through Self-Organised Transcoding, In Proceedings of Network and Operating Systems Support for Digital Audio and Video, 1998.
- [17] Macedonia, M., Brutzman, D., Mbone Provides Audio and Vision Across the Internet, IEEE Computer, pp30-36, April 1994.
- [18] McCanne, S., Jacobson, V., Vetterli, M., Receiver-Driven Layered Multicast, In Proceedings of ACM SIGCOM'96, pp. 117—30, Palo Alto, CA., Aug. 1996.
- [19] Padhye, J., Kurose, J., Towsley, D., Koodli, R., A Model Based TCP-Friendly Rate Control Protocol. Network and Operating System Support for Digital Audio and Video (NOSSDAV), June 1999.
- [20] Pasquale, G., Polyzos, E., Kompella, V., Filter Propagation in Dissemination Trees, In Network and Operating Systems Support for Digital Audio and Video, November 1993.
- [21] Paxson, V., Allman, M., Computing TCP's Retransmission Timer, RFC 2988, November 2000.
- [22] Rangan, P. V., Harrick, M. Ramanathan, V. S., Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences, IEEE/ACM Transactions on Networking, Vol.1, No.1, Feb., 1993.
- [23] Ratnasamy, McCanne, S., Scaling End-to-end Multicast Transports with a Topologically-sensitive Group Formation Protocol, In Proceedings IEEE International Conference of Network Protocols, ICNP'99, 1999
- [24] Rizzo, L., An Embedded Network Simulator to Support Network Protocol Development, In Proceedings of 9th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools, St. Malo, France, LNCS-1245, pp 97-107, Springer-Verlag, June 1997.
- [25] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., RTP: A Transport Protocol for Real-Time Applications, IETF RFC 1889, January 1996.
- [26] Waters, R., Anderson, D., Barrus, J., Brogan, D., Casey, M., McKeown, S., Nitta, T., Sterns, I., Yerazunis, W., Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability, Presence, vol. 6, no. 4, pp. 461-481, August 1997.
- [27] Yeadon, N., Quality of Service Filters for Multimedia Communications, Ph.D. Thesis, Lancaster University, Lancaster, U.K., May 1996.