Investigation and modeling of traffic issues in immersive audio environments

Jeremy McMahon, Michael Rumsewicz
TRC Mathematical Modelling
University of Adelaide
South Australia 5005, AUSTRALIA
jmcmahon, mrumsewicz@trc.adelaide.edu.au

Paul Boustead, Farzad Safaei
Telecommunications and Information Technology Research Institute
University of Wollongong
New South Wales 2522, AUSTRALIA
paul@snrc.uow.edu.au, farzad@uow.edu.au

***Abstract*—A growing area of technical importance is that of distributed virtual environments for work and play. For the audio component of such environments to be useful, great emphasis must be placed on the delivery of high quality audio scenes in which participants may change their relative positions. In this paper we describe and analyze an algorithm focused on maintaining relative synchronization between multiple users of such an environment and examine the subjective quality of service achieved.**

Keywords– audio synchronization; traffic modeling.

I. INTRODUCTION

The use of the Internet as a means of enabling group working and multi-party communications is ever increasing. A key factor in the acceptance of such applications is the ability of the Internet to deliver high quality, real-time, audio services, that is, the ability to enable users to carry out effective conversations.

Most literature regarding audio quality on the Internet focuses on important issues such as jitter handling and maintaining acceptable end-to-end delays (see, for example [3] and references therein), or the synchronization of audio and video streams in multimedia environments (see, for example, [5], [6], [7]).

In a multi-party audio environment, as might be found in a conference call or multi-player game with audio support, those factors are still important, but are supplemented by issues related to the multi-user nature of the application. For example, from the point of view of a specific user it may be important to maintain relative synchronization between all of the audio streams arriving to the user. By maintaining synchronization the user is ensured a more stable audio environment, which will assist in maximizing the usability of the application especially when the application characteristics are highly dynamic.

In this paper we develop an algorithm that maintains the relative synchronization between audio streams being mixed for playback to an individual in such environments. A key element of the algorithm is that it is able to adapt quickly to gross changes in the underlying delays of each stream traversing the network while not being overly sensitive to short term variations in delay resulting from audio stream packets being queued at routers.

This is an area that has not undergone significant research in the literature. Most of the available literature regarding synchronization in multimedia relates to synchronizing audio with video playout (e.g. [5], [6], [7]) and / or requires a global synchronization clock such as GPS (e.g. [6]). We consider the use of GPS timing to be unnecessary and somewhat excessive for the application considered herein. Zarros, Lee and Saadawi [8] examine the question of interparticipant synchronization without the use of a global timing mechanism. The analysis presented there focuses on synchronization of streams at the playout point and is measurement intensive in that it requires measurements of the maximum and minimum jitter for each source as well as determination of a so-called reference time. The algorithm presented herein does not require global clock synchronization, relying only on simple to maintain counters, and can be either employed at a central audio server or at the point of user playout at the client.

In order to determine the effectiveness of the proposed scheme we require a measure of audio quality in a group environment. At this time there does not appear to be any directly relevant literature to this question. A significant body of research exists concerning measuring the quality of service, in terms of packet delay, jitter and loss, required to support one to one conversation in packet audio environments (see, for example, [1], [2], [3], [4] and references therein). Consequently, we have extended the notion of Audio Rating Factors, as defined in [1], to group environments in order to gain a handle on overall quality across users of a multi-person audio application environment.

This paper is structured with system architecture outlined in Section 2 and the stream processing mechanisms in Section 3. Section 4 briefly discusses sound quality measurement. The simulation environment is described in Section 5 and Section 6 contains results of the simulation and discussion.

II. SYSTEM ARCHITECTURE

In this paper we analyze a pure peer-to-peer model, that is, an environment in which each client sends an audio stream to every other client. In this paper we are not concerned with the efficiency of the audio delivery mechanism but focus rather on the issue of mixing multiple audio streams for playout to a

single user.

Each client utilizes an Audio Stream Processor. This processor consists of a buffer for each of the incoming audio streams and a CPU that performs tasks on the streams, including relatively synchronizing and mixing. Once mixed, the single output stream is then sent to the client playback buffer. The architecture is shown in Fig 1.

III. STREAM PROCESSING MECHANISMS

Each client sends its audio stream to several other clients in a series of sequentially numbered packets. Each client uses the same audio encoding scheme and sends packets with the same (deterministic) interpacket times.

We now discuss the issues in delivering a coherent mixed audio stream to a client.

A. Audio Stream Processor Timeout

In order to meet the near real-time delay constraint for audio mixing of streams (typically, of the order of 150ms), it is necessary for the audio stream processor to "give up" on packets that have been delayed excessively or dropped by the network. To do this the audio processor utilizes a timer and any packets from the incoming audio streams that do not arrive before the timer expires are assumed lost and the audio processor mixes the packets within the group that have arrived.

The structure of the timeout is based on when the processor deems it should have produced a mixed audio packet. Once the first mixed packet is produced, a packet ideally should be produced at deterministic intervals matching the interpacket generation time at the sources. From these "ideal" times, the audio processor will wait a maximum of T_o ms before producing a mixed packet regardless of how many sources' packets are present.

B. Relative Synchronization

Different minimum propagation delays from sources to the client, and even the different random elements of the transmission delay, create issues with maintaining relative synchronization of streams at the audio processor. Consider a case where source 1 has a minimum propagation delay of 105ms while the minimum propagation delay for the other sources is 5ms. If all sources begin recording and sending data at the same time with the same interpacket generation time of 10ms, the 1st packet from source 1 is expected to arrive at the audio processor around the same time as the 11th packet from each of the other sources. The processor only mixes packets from sources of which it is aware. Until the first arrival from source 1, the processor has been mixing packets from all other sources and as such, is preparing to mix it's 11th packet, containing the 11th packet from the other sources. For this reason, a sequence number offset is used by the Audio Server to map the source assigned sequence number of each incoming packet to sequence numbers related to the mixing process. Each audio stream, and hence source, consequently has its own offset value. This value is added to the sequence number of every subsequent packet that arrives from the source, effectively assigning a new sequence number to the packet. It is these "virtual" sequence numbers by which the audio processor determines which packets are to be mixed into a single packet. The audio processor attempts to mix all packets with the same virtual sequence number *n*, say, into the n^{th} mixed packet it produces.

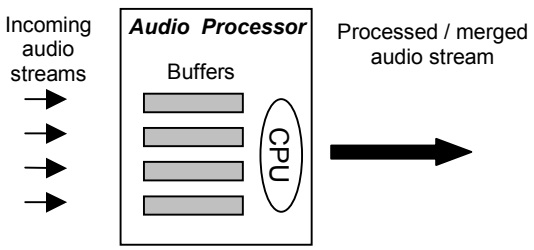

Figure 1. System architecture of Audio Stream Processor

C. Adjusting Relative Synchronization

If a packet experiences a large unexpected delay, the following packets from the same source can also be expected to experience a similar delay, for example, due to a change in path because of link failure. In this case, if adjustment to the offset is not implemented to bring the virtual sequence numbers back into line with the mixed packet sequence numbers, every packet from that source may be subsequently omitted from the mixing process at the client.

We now describe the Adaptive Synchronisation algorithm.

Adaptive Synchronization Algorithm:

Definitions:

- $s_n^{(i)}$ Source assigned sequence number of packet *n* from source *i*.
- $o^{(i)}$ The sequence number offset value for source *i*.
- $vs_n^{(i)}$ Virtual sequence number of packet *n* from source *i*.
- m Sequence number of the next mixed packet to be produced.
- $T_{+1}^{(i)}$ Number of times the missed packet trigger, as defined below, has been activated for source *i* in between packet arrivals.
- $T_{-1}^{(i)}$ Number of times the buffer limit trigger, as defined below, has been activated for source *i* in between packet arrivals.
- $S^{(i)}$ Boolean indicating whether sequencing certainty is satisfied for arrivals from source *i*.

Initialization: $m = 1$
$T_{+1}^{(i)} =$ 0 for all sources
$T_{-1}^{(i)} =$ 0 for all sources

Algorithm: Packet arrival from source *i* with sequence number $s_n^{(i)}$ occurs.

```
If (n = 1) {
    o(i) = m - s1(i) }

If (T+1(i) = 0 AND T-1(i) = 0) {
    vsn(i) = sn(i) + o(i)
    If (vsn(i) >= m) {
        Place packet in buffer. }}
```

```
Else If (T+1(i) > 0) {
    o(i) = o(i) + T+1(i)
    T+1(i) = 0
    vsn(i) = sn(i) + o(i)
    If (vsn(i) >= m) {
        Place packet in buffer. }}

Else If (T-1(i) > 0) {
    If (S(i)) {
        o(i) = o(i) - T-1(i)
        T-1(i) = 0 }
    vsn(i) = sn(i) + o(i)
    If a packet already exists in the buffer from
    source i with the same virtually assigned
    sequence number {
        Remove packet already in buffer. }
    If (vsn(i) >= m) {
        Place packet in buffer. }}
```

Once a mixed packet has been produced, it is then sent on to the appropriate client playback buffer.

The missed packet trigger is triggered initially when a given number of packets from a particular source have been sequentially missed in the mixing process. It is then triggered once for every subsequent missed packet. When a packet eventually arrives, that source's offset value is increased by the number of times the missed packet trigger has been triggered, $T_{+1}^{(i)}$. Increasing the offset reduces the number of packets missed from a particular source, at the expense of end-to-end delay of the packets.

The buffer limit trigger is triggered when a packet arrival causes the buffer for a particular source to contain a predetermined limit of packets and sequencing certainty (discussed below) is satisfied. The buffer limit trigger can at most be triggered once in between arrivals as it is triggered just after an arrival. Having a high number of packets in the buffer means that packets are waiting in the buffer longer than possibly necessary. When the offset is decremented, a packet will be discarded, having the same effective sequence number as the previous packet that arrived, but from this point the end-to-end delay for packets being mixed has been reduced.

Sequencing certainty is a measure of whether or not a specified number of packets arriving from a particular source have consecutive sequence numbers.

Using the algorithm enables the audio processor to dynamically correct relative synchronization without the need for a centralized time source.

IV. Methods for Measuring Sound Quality

The subjective quality of sound playout to the client is affected by network performance through end-to-end packet delays and losses. Sound quality is further affected by the codec employed and setup parameters related to the codec, for example, the size of playout buffers.

The ITU-T has developed a model that predicts customer perception from objective measurements of attributes such as delay, loss and the codec employed. This model, the so-called E-model, is described in ITU-T recommendation G.107 and the related recommendation G.113 [1], [2]. The E-model operates by converting a number of objective network measurements into a rating factor, *R,* on a scale of 0 to 100. Rating factor values greater than approximately 70 correspond to acceptable service quality.

ITU-T Recommendation P.800 [4] states that speech samples in the order of 2.5-5 seconds be used to assess speech quality. Consequently, measurements obtained over conversation periods of time should not be used to determine a single *R* factor for a call. In the analysis section, 3 second time periods are used to calculate R-values for a particular audio stream.

V. Description of Simulation Environment

The simulation environment consists of N clients sending audio and one client receiving audio. At each of the N clients there is a user, whose speech is recorded, encoded, packetized, and sent into the data network. Packets sent into the network contain 10ms of audio samples, consistent with the PCM codec information found in G.113 [2].

Once in the network, the packet experiences a transmission delay comprising two components, $d_p^{(i)}$ and $d_q^{(i)}(n)$. The first, $d_p^{(i)}$, is defined as the minimum propagation delay from client *i* to the destination client. The latter, $d_q^{(i)}(n)$, is the random element of the network delay for the n^{th} transmitted packet resulting from queuing at routers and switches.

Under normal circumstances, all packets from source *i* to the destination client follow the same path. As a result, the random element of network delay is correlated for consecutive packets within a stream.

We generate the correlated interpacket delays using the following relation:

$$d_q^{(i)}(n) = d_q^{(i)}(n-1) + f(d_q^{(i)}(n-1))$$

where $f(\cdot)$ has the property that $f(x) > -min(y, x)$ for packets containing *y* milliseconds of audio.

The simulation also allows for occurrences of network failure. In such instances, packets from a particular source are forcibly rerouted along a different path. This results in a potentially large change of the minimum propagation delay $d_p^{(i)}$ for that source. When this change occurs, packets from a particular source may arrive out of sequence at the destination client.

Once the packet has arrived at the Audio Stream Processor, it is assigned a virtual sequence number using the synchronization algorithm, and enters the corresponding stream buffer.

Packets from different sources with the same virtually assigned sequence number are mixed into a single packet and placed into the client's playback buffer.

VI. Numerical Results

The simulation environment consists of 5 sources sending PCM encoded audio in packet sizes containing 10ms of audio samples. We assume a 3ms encoding/decoding delay for this codec and a minimum propagation delay of 25ms from all sources to the destination client. A 40ms timeout is used at the processor and once produced, the mixed packets are passed

into an 80ms playback buffer. As parameters to the synchronization algorithm, a missed packet trigger of 1 packet, a buffer limit trigger of 7 packets and a sequencing certainty value of 3 packets are used.

For the missed packet trigger, the higher the value, the more slowly the algorithm will react to changes in the transmission delay. Having a slower reaction time is not necessarily detrimental to performance but it does increase packet loss percentage. Smaller values mean the algorithm reacts quickly, possibly when not needed due to a small random variation, and can drive up the end-to-end delay while lowering loss rates. When setting the buffer limit trigger, larger values tend to keep the system more stable by maintaining larger end-to-end delays for longer periods than perhaps necessary.

It is best to minimize end-to-end delays and loss percentages at the same time, however, a compromise between the two trends is chosen to continue with the performance analysis.

Traffic model: The specific form of $f(\cdot)$, used in the calculation of the random element of propagation delay, has the form:

$f(x)$ drawn from *Exponential*(5) – 10, for $x > 5$ms,
$f(x)$ drawn from *Pareto*(2,2) , for $x <= 5$ms,

where the distribution function for *Pareto*(α,β) is given by

$$F(t) = 1 - (\beta/t)^{\alpha}.$$

The expected value of the Pareto in this case is 4ms delay and it has infinite variance. To avoid excessive increases in the propagation delays for successive packets within the simulation, $f(\cdot)$ is bounded above by 1000ms. It is not, however, desirable to have these delay spikes occurring too frequently and hence the use of 5ms as an upper bound for use of the Pareto generated random variables. The negative exponential part of the combined distribution has an expected value of -5ms. This is so that after a large random element of delay, it is more probable that the delay will reduce itself than continue to increase. A sample of possible network delays experienced by sequential packets in the network is shown in Fig 2. Here, the minimum propagation delay of the network path begins at 25ms, with a path change after 1 second increasing the minimum propagation delay to 60ms for a further 1 second at which time, the original network path is restored and utilized.

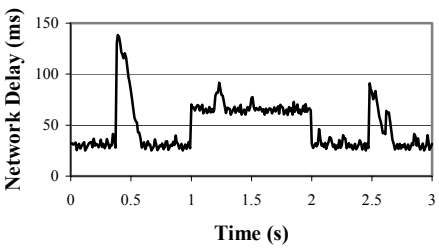

Figure 2. Sample network delay experienced in the simulated network.

Synchronisation Analysis - Deterministic Environment: To test the synchronization algorithm, the simulation was set to run for 9 seconds. For the first 3 seconds, the propagation delays for all audio streams are 25ms. Once 3 seconds passes, a change in network topology is simulated and the minimum propagation delay for source 1 increases instantaneously to 100ms (without any loss of packets in transmission). This delay is experienced by source 1 until a further 3 seconds passes, at which time, the minimum propagation delay returns to its original value of 25ms. The delays continue at this level until the termination of the simulation.

To simplify the resulting output, this was first simulated with no random effects of propagation delay, creating a deterministic environment, making it easier to visualize the effects of the algorithm.

From time 0-3 seconds, a packet is generated from each source every 10ms. The transmission delay, including recording, encoding and propagation delays, for each of these packets is 38ms. Once at the client's audio processor, the packets are mixed immediately into a mixed packet. In this particular example, the offset for source 1 is initialized within the simulation to 0 and all other offsets are initialized to 1, due to the processor's conditions for initiating mixing processes. The simulated network change, and hence the change in network propagation delay, affects every subsequent packet from source 1. Immediately following the change, 50ms passes until the next mixed packet is produced, which does not contain any audio information from source 1. This 50ms consists of the 10ms that would usually have passed until the next mix and the 40ms processor timeout. Due to the way the timeout is structured, a mixed packet is produced every 10ms until it is once again possible to include all sources in the mixing process. Fig 3 shows the event times of arrivals and mixing processes at the audio processor with corresponding sequence numbers around the transition period, assuming source sequence numbers begin at 1.

It is not until 4 mixed packets have been produced that the next arrival from source 1 occurs. The first mixed packet that is produced without information from source 1 triggers the missed packet trigger once for source 1. For every subsequent

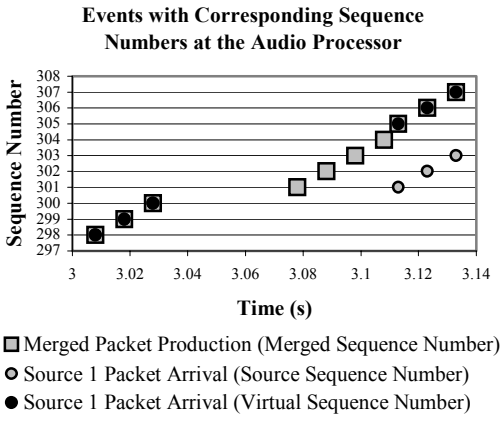

Figure 3. Processor events.

omission, the trigger is triggered until a packet from source 1 is included. When the arrival occurs, the offset value of source 1 increases by 4, which is sufficient such that the packet from source 1 is assigned a virtual sequence number that corresponds to the sequence number of the next mixed packet to be produced.

The new offset value of 4 remains, in this deterministic environment, until the change back to the original network propagation delays. During this period, the end-to-end delay for source 1 is increased by 40ms (10ms for each packet omitted). Due to the relatively high buffer limit trigger, there is not a sufficient jump in network propagation, using these inputs, for the other 4 streams to be affected. Instead, for nearly all of this period, the packets from the other 4 sources arrive to see 4 other packets already in their respective buffers. As such, their offsets remain at 1 and hence their end-to-end delays remain as they were initialized.

When the original network topology is restored, the only changes that eventuate are the number of packets that arriving packets see in the buffer. For source 1, this number increases to 3 packets for every packet arrival. All other sources' packets arrive to empty queues. End-to-end delays are not affected in any way, as the buffers never contain the trigger capacity of 7 packets.

Synchronisation Analysis – Random Environment: The same inputs were used to perform a simulation of network failure as above, but also with the inclusion of random elements of delay. Now, it is not only the change in network topology that can cause packets to be omitted from the mixing process.

The R-values calculated from the simulation are shown in Table I. It is worth noting that the maximum R-value that can be obtained using this particular codec is 93.1928. Sources 2 – 5 can be seen to be near the best audio quality attainable for the entirety of this particular simulation. The higher loss percentage and average end-to-end values both contribute to the lower rating factor value for source 1 in the 2nd 3 second block.

TABLE I. RATING FACTOR VALUES FOR A SIMULATION OF NETWORK CHANGE USING PCM ENCODING

Time	*source 1*	*source 2*	*source 3*	*source 4*	*source 5*
0 – 3	91.5023	91.7203	90.2211	91.5023	91.5023
3 – 6	82.24	91.5023	91.3867	91.5023	91.5023
6 – 9	91.3851	90.1695	92.7474	92.7474	92.7474

Nevertheless, ITU-T G.107 states that at an R-value lower limit of 70, user satisfaction is still rated as acceptable. This means that using the relative synchronization algorithm, given the current system parameters, all users listening to the mixed output at the destination client would have been satisfied to very satisfied with the quality of audio received from each source.

VII. CONCLUSION

This paper provides an efficient algorithm that can achieve and maintain relative synchronization between audio streams in a real time audio mixing environment. The algorithm does not attempt to maintain absolute synchronization between audio streams, as this would require the use of global timing. Rather, the algorithm attempts to maintain consistency within the mixing process such that the alignment of audio samples from each stream remains as constant as possible given the random elements of network delay. The algorithm is able to adapt quickly to gross changes in the underlying delays of each stream, as might result from network link failures. At the same time, the proposed algorithm is robust to short term variations in delay resulting from audio stream packets being queued at routers.

ACKNOWLEDGMENT

The support of the Cooperative Research Centre for Smart Internet Technology (CRC-SIT, www.smartinternet.com.au) for this work and permission to publish this paper is hereby acknowledged.

REFERENCES

[1] ITU-T Recommendation G.107 (07/2002), The E-model, a computational model for use in transmission planning.

[2] ITU-T Recommendation G.113 Appendix 1 (05/2002), Provisional planning values for the equipment impairment factor Ie and packet-loss robustness factor Bpl.

[3] A.P. Markopolou, F.A. Tobagi and M.J. Karam, Assessing the quality of voice communications over Internet backbones, IEEE/ACM Transactions on Networking, vol. 11, no. 5, October 2003.

[4] ITU-T Recommendation P.800 (08/96), Methods for subjective determination of transmission quality.

[5] C-M Huang, H-Y Kung, J-L Yang, Synchronization and flow adaptation schemes for reliable multiple stream transmission in multimedia presentations, The Journal of Systems and Software 56 (2001) 133-151.

[6] P V Rangan, S Ramanathan, T Kaeppner, Performance of inter-media synchronization in distributed and heterogeneous multimedia systems, Computer Networks and ISDN Systems 27 (1995) 549-565.

[7] L Bertoglio, R Leonardi, P Migliorati, Inter-media synchronization for videoconference over IP, Signal Processing: Image Communication 15 (1999) 149-164.

[8] P N Zarros, M J Lee, T N Saadawi, Interparticipant synchronization in real-time multimedia conference using feedback, IEEE/ACM Transactions on Networking, vol. 4, no. 2, April 1996.

