A Low-Latency Full-Duplex Audio over IP Streamer

Asbjørn SÆBØ and U. Peter SVENSSON
Centre for Quantifiable Quality of Service in Communication Systems
NTNU – Norwegian University of Science and Technology
O.S. Bragstads plass 2E
N-7491 Trondheim
Norway
asbjs@q2s.ntnu.no, svensson@q2s.ntnu.no

Abstract

LDAS (Low Delay Audio Streamer) is software for transmitting full duplex high-quality multi-channel audio with low end-to-end latency over IP networks. It has been designed and implemented as a tool for research into distributed multimedia interaction and quality of service in telecommunications. LDAS runs on Linux, using the ALSA sound drivers and libraries. It uses UDP as its transport protocol. A flow control scheme is used to keep the sender and receiver synchronised and to deal with transmission errors. Tests have shown end-to-end latencies (from analog input to analog output) down to around five milliseconds over a minimal network.

Keywords

audio, streaming, latency, IP network

1 Introduction

The field of telecommunications is changing, with new technologies making new services possible. One aspect of this is the increasing use of various kinds of transmission of audio and multimedia over packet-switched networks using the Internet Protocol (IP).

Related to this, "Quality of Service" (QoS) is a topic of current interest in telecommunications. Traditionally, this deals with technical specifications and guarantees. A wider interpretation may also take into account the quality of service as experienced by the user. In this respect, an increased quality of service may not only be an increase in the quality of a given service (like voice communication), but to improve the user experience by offering a service of inherently higher quality, something that is not only better, but more. (An example might be to replace a one-channel voice communication service with a voice communication service with 3D-audio.)

There is an untapped potential for services utilising audio and multimedia over IP that would give a better quality of service, as experienced by the user. The ultimate telecommunications system would enable one to achieve virtual *presence* and true *interaction* between the endpoints of the communication. Current common systems, like internet telephony (Voice over IP, VoIP) and video conferencing do not fully achieve this. A primary aspect of this topic is how such services should work, on a technical level. Further, and perhaps even more interesting, is how they may be used, and how the user will experience them.

Our aim is therefore to explore and investigate more advanced and demanding communication services, focusing on the audio side of *distributed multimedia interaction*. The general situation would be one where two acoustical situations are connected through the network in such a way as to achieve transmission of the complete acoustic environments. Examples of such applications, and our immediate targets, are ensemble playing and music performance over the network, and transmission of various kinds of 3D audio (binaural, Ambisonics, multichannel). This has been demonstrated in a number of studies, e.g. (Woszczyk et al., 2005).

Latency is the most interesting and demanding of the factors limiting these kinds of services. Available data storage capacity, data transmission capacity (bandwidth) and processing capacity are all steadily increasing, with no hard bounds in sight. Transmission time, however, is fundamentally limited by the the speed of light. And, as will be further discussed below, low latency is in many cases important and necessary in order to achieve interaction. In addition to high quality overall, low latency should therefore be emphasised.

For our exploration, a tool suitable for such services was needed. This tool should be capable of transmission of high quality, low latency audio. It should also be open, to facilitate control over all aspects and parameters of the transmission process. Several such applications exist, from the simple ones to the very advanced

ones. For many of these, source code is not accessible, making studies of their internals or changing of their workings next to impossible. Others do, in various ways, not meet our needs, although StreamBD, used at CCRMA, (Chafe et al., 2000), might have had adequate properties. The development of such a tool would give valuable insights into the specific problems associated with these topics, give full control of all aspects of the tool and its usage and be a good way to start the explorations. So, the task of developing LDAS - the Low Delay Audio Streamer, was undertaken, using previous work done at NTNU (Strand, 2002) as a starting point.

2 Requirements and specifications

The two immediate target applications are networked ensemble playing and transmission of acoustical environments. The first involves musicians at different locations, connected by a network, performing music together. The second involves the transmission of enough information from one site to another to be able to recreate the acoustic environment of the first site at the second site in a satisfactorily manner. The main requirements of these applications are discussed below.

2.1 Audio quality

Audio quality should be high. For simplicity, the lower limit for quality has been set equal to that of an audio Compact Disc. This means a sampling frequency of 44.1kHz (or 48kHz) or higher, and a sample depth (word length) of at least 16 bits for uncompressed PCM.

The number of channels available should be at least two (which will allow for transmission of stereo or binaural signals), but preferably eight or more (which will allow for various kinds of multi-channel and three-dimensional audio formats like Ambisonics).

2.2 Latency considerations

Latency is important for interaction. In particular, it is known that excessive inter-musician latency is detrimental to ensemble playing, one of our target applications (Bargar et al., 1998). The fundamental latency requirement is therefore that the latency should be so low that it will not hinder successful ensemble playing across the established connection.

Some investigations into what constitutes tolerable delay has been done, but few definite conclusions have been given. Experiments, using hand-clapping as the musical signal, have found an "optimal" delay (with respect to tempo stability) of 11.6 milliseconds (Chafe and Gurevich, 2004). Based upon these data and their own experiences, (Woszczyk et al., 2005) suggest that latencies of 20ms to 40ms are "easily tolerated", and that even higher latencies may be acceptable after training and practice.

Experiments conducted at the acoustics group at NTNU have concluded that latencies lower than 20 milliseconds do not seem to influence the ensemble playing much, while latencies above 20 milliseconds may may lead to a less "tight" rhythm, with the two musicians not following each other as well (Winge, 2003). On the other hand, (Lago and Kon, 2004) claims that latencies up to at least 30 milliseconds should be considered normal and in most situations acceptable, and that latencies of this order will not impair musical performance. Further, informal evidence, based upon the experience of a number of practising musicians, indicates that latencies of 30 ms and maybe up to 50ms may be tolerable.

For comparison: Sound in air at room temperature travels at approximately 345 meters per second. This gives a latency of about three milliseconds per meter. Two musicians two meters apart will experience a delay of six milliseconds. The width of a typical sympony orchestra on stage may be around 15 to 20 meters, corresponding to a latency between the outermost musicians on the order of 50 milliseconds. (It should be noted that an orchestra also has external synchronisation in the form of a conductor.)

Obviously, for networked ensemble playing, network transmission time may make up a large part of the total latency. (With factors like A/D and D/A conversion, buffering in the sound card, data processing and handling by application and OS and travel time for sound waves in air making up for the rest.) Whether a sufficiently low latency is possible is therefore to a large degree dependent upon the "network distance" between the participants.

Based upon the indications above, the operating requirement was set to have an end to end latency (analog signal to analog signal) of less than 20 milliseconds for transmission over a campus-wide LAN. To avoid unnecessary latency, it was decided that uncompressed audio should be transferred. While expending effort into keeping the latency low, it should be re-

membered that latency requirements must be balanced against the robustness of transmission. This compromise should be tunable.

2.3 Network protocol

Using IP (Internet Protocol) as the network layer protocol is a given, since the area for which LDAS is intended is audio over IP.

The two main transport protocols used over IP are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP provides a reliable, connection-oriented transmission mechanism. Lost packets are retransmitted, and flow and congestion control is part of the protocol. UDP is a connectionless protocol with very limited delivery guarantees. Packets sent via UDP may not arrive at the destination at all, they may arrive out of order, or they may arrive as duplicates. The only guarantee given is that if a packet does arrive, its contents are intact. UDP supports broadcasting and multicasting (which TCP does not), but does not have built in flow and congestion control. (Stevens et al., 2004).

For networked ensemble playing, the reliability of TCP is not called for. A low latency is deemed more important than totally reliable delivery of all data. Some data loss may be acceptable, and retransmission of lost packets may be a waste of time and capacity, as they may well be discarded due to arriving too late when they finally arrive. Neither will TCP's flow and congestion control be the optimal way to rate-limit LDAS traffic. Methods taking into account the nature of the service, built into the application itself, will be preferable. (Until flow control is implemented in LDAS, LDAS traffic will not be TCP-friendly. But, at least for the research purposes and situations, this is acceptable.)

As multi-way communication is envisioned, it is possible, maybe also probable, that LDAS may be extended to multicast, for which UDP is needed. As the distinguishing features of TCP are not necessary for LDAS, not choosing TCP as the transport protocol is not a disadvantage. On the other hand, the multicast feature of UDP may be essential. On the basis of this, UDP was chosen as the transport protocol.

It is necessary for LDAS that the order of packets be maintained and the relative placement of packets in time be correct. This feature is not delivered by UDP. LDAS should therefore implement its own protocol, on top of UDP, to provide the necessary features for packet stream tracking and control.

Alternative protocols are DCCP (Datagram Congestion Control Protocol, `http://www.icir.org/kohler/dcp/`) and RTP (Real Time Protocol, `http://www.faqs.org/rfcs/rfc3550.html`). DCCP is intended as a substitute for UDP for applications like these, but is still a work in progress. RTP sits on top of UDP, and is a transport protocol for real time applications. It was, however, found to be more complex than needed for this case.

2.4 Synchronisation

It is mandatory that the sending and the receiving parts be kept synchronised, keeping the latency as low and as constant as circumstances will allow. To minimise the effects of network transmission time jitter, the receiving part should maintain a buffer of received audio data. The amount of data to be held in this buffer should be setable, so it can be balanced against the latency requirements.

3 Implementation

The high-level system architecture is currently a "pairwise peer-to-peer" system, i.e. two equal programs sending and receiving data to and from each other.

The structure of the application is shown in figure 1. There are three threads running in parallel, a recorder/sender, a receiver and a playback task. The recorder/sender thread is running independently of the two other threads, while the receiver and the playback threads are linked through a common data structure, the *receiver queue*. All three threads are running scheduled as SCHED_FIFO tasks.

3.1 The data stream and the packet format

Audio is a continuous signal, which is digitised by the audio interface into a stream of samples. It is, however, practical to handle chunks of the stream as units when processing and transmitting the data. The audio stream is therefore divided into a series of application level *packets*.

This packet stream is the connection between the sender and the receiver. The packet format is quite simple. Each packet consists of one *period*[1] of audio data, as delivered from the audio interface. (Currently, the ALSA interleaved format is used.) To these data is appended a

[1] In the ALSA sense.

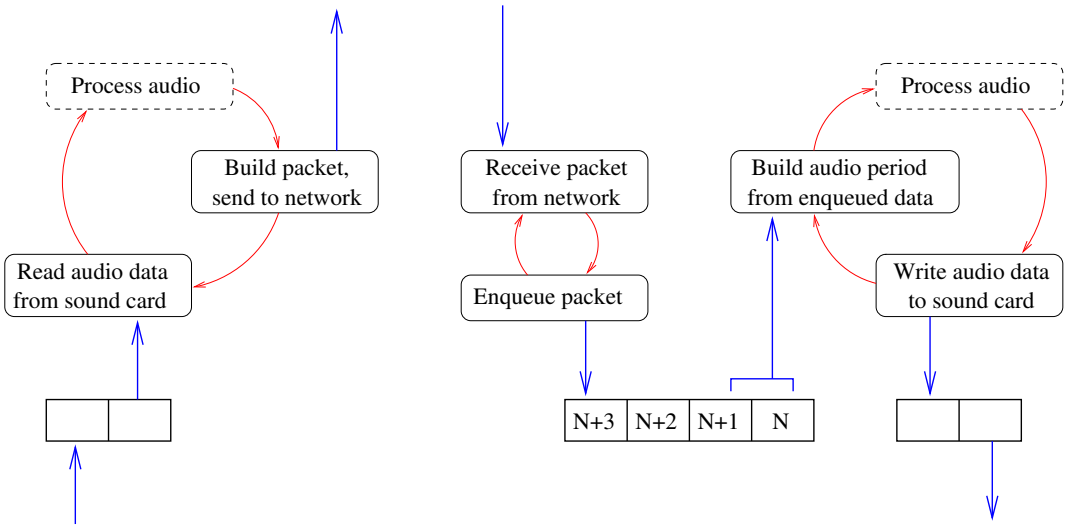

Figure 1: Application architecture and conceptual flowchart. Blue (straight, vertical) lines show the flow of the audio data, red (curved) lines show program flow. From left to right, there is the recorder/sender thread, the receiver thread and the playback thread. Audio is acquired and played back via the audio interfaces, at the bottom of the figure, and sent to, and received from, the network at the top of the figure. The square boxes at the bottom of the figure are, from left to right, the input sound card buffer, the receiver queue and the output sound card buffer. The dashed boxes indicates the possibility for additional processing, currently not present, of the audio data.

sequence number and a time stamp, as shown in figure 2.

Positions in the audio stream are identified by the sequence number of the packet and a frame-level offset into the packet. Of particular interest is the *playback position* kept by the receiver. This is the position of the data that is "next in turn" to be sent to the playback sound card.

3.2 The receiver queue

The receiver queue is the central data structure of the program. In this queue, received packets are temporarily stored until their audio data have been played back. (No copying of data takes place during enqueuing. The queue is implemented as an array of pointers to packets, addressed modulo the length of the array to give a circular buffer.) The queue, containing the packet corresponding to the playback position and any more recent packets, acts as a sliding window onto the incoming packet stream.

A primary purpose of the queue is to buffer the incoming packet stream, absorbing the effects of network transmission time jitter. The setting of the *nominal length* of the queue, i.e. the number of audio periods we try to keep in the queue, allows for the trading of latency for robustness. More data in the queue will give a higher latency, but also more leeway for the occasional late arriving packet to still come in time.

The queue, together with the packet format, implicitly defines a simple protocol that makes the application capable of handling the shortcomings of UDP. Packets are inserted into the queue in correct order, according to their sequence numbers. Lost (or missing) packets are detected, and dummy data (currently silence) substituted in their place. Duplicate packets, and packets arriving too late, are detected and rejected. Altogether, this gives a data stream that is ordered and without gaps.

The queue is further central in the synchronisation of sender and receiver, as discussed below.

3.3 Synchronisation and drift adjustment

There are two kinds of synchronisation issues, synchronisation on a large scale and drift adjustment. Both are handled in the receiver queue.

The first, synchronisation, is handled by the receiver thread. As mentioned above, the receiver queue is a sliding window onto the packet stream. The purpose of the synchronisation is to position this window correctly. This is done by monitoring the sequence numbers of the incoming packets. A packet is said to be "early" if

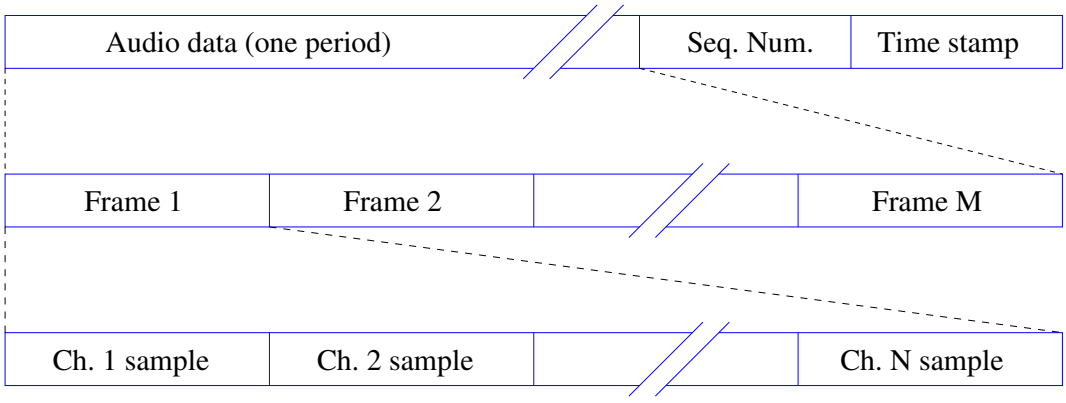

Figure 2: LDAS packet format, the payload of the UDP packet.

its sequence number is so high that it would fall outside of the queue. If an early packet arrives, this is a sign that the receiver is lagging, and the queue is resynchronised. Resynchronisation resets the queue, setting a new playback position, so that the length of the queue, as measured from the playback position to the end of the queue (including the early packet) equals the nominal queue length.

Similarly, "late" packets are packets arriving after they should have been played back, i.e. with a sequence number lower than the sequence number corresponding to the current playback position. From the number of recent late arriving packets, a "lateness factor" is computed. If this value gets too high, the receiver is assumed to be leading, and the receiver queue is resynchronised.

Although nominally equal, the sampling frequencies of the sound cards at the sending and receiving ends will in practice differ slightly. Over time, this difference will lead to more samples being produced than consumed, or vice versa, and this aggregation or starvation of data will cause the sender and the receiver to drift apart. To compensate for this, more smoothly than would be done by large scale synchronisation alone, the playback thread does drift adjustment. From the nominal queue length, an upper and a lower limit is computed. If the low pass filtered actual queue length increases or decreases beyond those limits, single samples are skipped or reused to adjust the rate of data consumption as appropriate to counteract the drift. There is at most one sample of adjustment per period, giving a range of adjustment that is inversely proportional to the period size.

A consequence of this adjustment is that the audio periods sent to the sound card will in general not correspond to those of the received packets, but be built from audio data from more than one packet.

4 Similar solutions

For comparison purposes, this section gives a brief overview of other available open source software similar to LDAS. The information is gleaned from available documentation and various other sources. Except for llcon, the authors of this paper have not done actual experiments with these other solutions.

Netjack (Hohn et al., 2006) is a mechanism for transporting realtime audio over an IP Network using UDP. It is fully integrated with Jack. Full duplex transmission of uncompressed stereo data between two computers is offered. Netjack slaves one jackd to the other, and synchronises the slave jack using the incoming packet data. Which technique is used for the synchronisation is not clear from the available documentation.

An earlier solution is jack.udp (Drape, 2005). This is an udp packet transport that sends jack audio data, using UDP, over local networks between two jack enabled computers. For correct operation, jack.udp needs some form of external synchronisation between the soundcards of the computers.

The StreamBD software (SoundWIRE Group, CCRMA, Stanford, 2002) may, among other things, be used for transmission of audio over a network. It uses ALSA or OSS audio drivers, and provides multiples channels of uncompressed audio. TCP is used for one way transmission, UDP or non-guaranteed variants of TCP for bidirectional transmission. Whether, or how, synchronisation is achieved is

not described in the available documentation.

The llcon (Fischer, 2006) software differs from the rest in that it is aimed at situations with limited bandwidth, like 256kbps DSL lines. It may also connect more than two endpoints. Endpoint clients connect to a central server, which mixes all incoming signals and returns the mix to the clients. At the clients, stereo is input and mixed to one channel before sending the data to the server. The sampling frequency is 24 kHz, and IMA-ADPCM coding, which gives a coding delay of one sample, is used to compress the audio data rate to 96 kbps.

5 Latency measurements

For initial testing and latency measurements of LDAS, two PCs were used. PC-1 was equipped with an Intel P4 CPU, PC-2 had an Intel P3 CPU at 1Ghz. The computers had 512 MB of RAM each, and were equipped with M-Audio Delta44 audio interfaces. Both computers were running Linux set up for real time operation, with 2.6.12 multimedia kernels and associated setup from DeMuDi 1.2.0. (PC-2 was a regular DeMuDi installation, while PC-1 was "sidegraded" from Debian Sarge.)

The computers were connected via a switch of the regular campus LAN. The network round trip time, as reported by "ping" was 0.1 ms. For the measurements, LDAS was version 0.1.1 plus a small fix for a hard coded period size. Audio transmission was two-channel full duplex, using the ldas_mate executable. PC-2 was controlled via SSH logins from PC-1, with the corresponding network traffic between the two computers following the same network route as the LDAS transmissions.

Latency measurements were done on one channel, one way, using an impulse response measurement system, a computer running the WinMLS (Morset, 2004) measurement software. Responses were measured from the analog input of PC-1's sound card to the analog output of PC-2's sound card. Several combinations of period size and nominal receiver queue length were tried. For each combination, six impulse response measurements were taken. From the impulse responses, the total latency through the transmission chain was computed. The results are given in table 1.

While measuring, audio from a CD player was transmitted on the remaining channel from PC-1 to PC-2, and also on both channels in the opposite direction. The quality of the transmission was subjectively evaluated by monitoring the received audio signals (mostly one direction at a time) for audible drop-outs and other forms of distortion or quality reduction. For the transmission from PC-1 to PC-2, combinations A, B and D were robust, with no artifacts noted. For combinations C and E, a few distortions (over the course of a few minutes) were noted. The distortions had the character of brief crackles. Transmission from PC-2 to PC-1 seemed somewhat less robust. Infrequent distortions were noted for combination D, and more frequent distortions for E, F and G. Especially for the lower latency setups, transmission from PC-2 to PC-1 was susceptible to other use of PC-1, with e.g. a "find /usr" with output to a terminal window leading to quite a bit of "stuttering" in the transmitted audio.

	Period size	Queue length	Mean latency	Standard deviation
A	128	3	15.2	0.8
B	128	1	11.0	0.8
C	128	0.1	8.2	0.6
D	64	1	5.8	0.4
E	64	0.1	4.9	0.4
F	32	0.1	3.1	0.2
G	16	1	2.6	0.1

Table 1: Total latency for audio transmission with LDAS. The latency is measured from analog input to analog output for various combinations (marked A to G) of audio period size and nominal receiver queue length. The latency mean value and standard deviation are given in milliseconds. Period size is measured in frames (samples), and the nominal length of the receiver queue in periods. Sampling frequency is 48kHz.

6 Conclusions

A software tool for transmission of high-quality multichannel audio over IP networks has been designed and implemented. As one target application of the tool is networked ensemble playing, emphasis has been placed on achieving low latency. Testing has shown that the tool is capable of achieving robust full duplex transmission while keeping latencies below our stated goal of 20 milliseconds on a local area network. If small transmission artifacts are accepted, even lower latencies may be obtained. While not finished, the software is in a state where it may be

used for experimental work and research into user experienced quality of service in telecommunications.

7 Acknowledgements

Richard Lee Revell, of Mindpipe Audio, has written parts of LDAS, in cooperation with Asbjørn Sæbø. We are grateful for his assistance. The authors also thank Svein Sørsdal and Georg Ottesen of SINTEF, Jon Kåre Hellan of UNINETT and Paul Calamia, currently at Rensselaer Polytechnic Institute, for valuable discussions and advice.

8 Availability

The LDAS software has been released under the GNU General Public License. It may be downloaded from `http://www.q2s.ntnu.no/~asbjs/ldas/ldas.html`.

References

Robin Bargar, Steve Church, Akira Fukuda, James Grunke, Douglas Keislar, Bob Moses, Ben Novak, Bruce Pennycook, Zack Zettel, John Strawn, Phil Wiser, and Wieslaw Woszczyk. 1998. Technology report TC-NAS 98/1networking audio and musing using internet2 and next-generation internet capabilities. Technical report, Technical Council, Audio Engineering Society, Inc.

Chris Chafe and Michael Gurevich. 2004. Network time delay and ensemble accuracy: Effects of latency, assymetry. In *117th Audio Eng. Soc. Convention Preprints*. Audio Eng. Soc., October. Preprint 6208.

Chris Chafe, Scott Wilson, Randal Leistikow, Dave Chisholm, and Gary Scavone. 2000. A simplified approach to high quality music and sound over IP. In *COST-G6 Conference on Digital Audio Effects (DAFx-00)*, December 7 – 9.

Rohan Drape. 2005. jack.udp. `http://www.slavepianos.org/rd/sw/sw-23`.

Volker Fischer. 2006. Low latency (internet) connection. `http://llcon.sourceforge.net/`.

Torben Hohn, Dan Mills, and Robert Jonsson. 2006. Netjack v. 0.7. `http://netjack.sourceforge.net/`.

Nelson Posse Lago and Fabio Kon. 2004. The quest for low latency. In *International Computer Music Conference*, Miami, November.

Lars Henrik Morset. 2004. WinMLS 2004. `http://www.winmls.com`.

SoundWIRE Group, CCRMA, Stanford. 2002. StreamBD 1.0b. `http://ccrma.stanford.edu/groups/soundwire/software/newstream/docs/`.

W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. 2004. *UNIX Network Programming*, volume 1. Addison-Wesley, third edition.

Ola Strand. 2002. Distribuert multimedia samhandling. Master's thesis, Norwegian University of Science and Technology, NTNU, Dept. of Telecomm., Acoustics Group. (In norwegian).

Håkon Liestøl Winge. 2003. Musikksamspill over IP. Term project, NTNU, Norwegian University of Science and Technology, September. (In norwegian).

Wieslaw Woszczyk, Jeremy R. Cooperstock, John Roston, and William Martens. 2005. Shake, rattle and roll: Getting immersed in multisensory interactive music via broadband networks. *J. Audio Eng. Soc.*, 53(4):336 – 344, April.