JACKTRIP/SOUNDWIRE MEETS SERVER FARM

Juan-Pablo Cáceres & Chris Chafe
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
{jcaceres,cc}@ccrma.stanford.edu

ABSTRACT

Even though bidirectional, high-quality and low-latency audio systems for network performance are available, the complexity involved in setting up remote sessions needs better tools and methods to asses and tune network parameters. We present an implementation of a system to intuitively evaluate the Quality of Service (QoS) on *best effort* networks.

In our implementation, musicians are able to connect to a multi-client server and tune the parameters of a connection using direct "auditory displays." The server can scale up to hundreds of users by taking advantage of modern multi-core machines and multi-threaded programing techniques. It also serves as a central "mixing hub" when network performance involves several participants.

1 INTRODUCTION

Systems for real-time, high-quality and low-latency audio over the Internet that take advantage of high-speed networks are available and have been used in the last several years for distributed concerts and other musical applications [12]. The difficulty of setting up one of these distributed sessions is, however, still very high. Most musicians have experienced the disheartening amount of time that can be lost in rehearsal, where most of the time is spent adjusting the connection rather than playing music.

Keeping delay to a minimum is one of the main goals when tuning network parameters. Delay is known to be disruptive in musical performance [4], so a sensible goal is to minimize it as much as possible. Often, there is a tradeoff with audio quality. The longer the latency, the better the audio (i.e., less dropouts) if facing problematic network conditions. For most users that are not familiar with TCP/IP network protocols [1] and delivery, understanding the meaning of these parameters can be daunting.

We present here a server-based application that can be of use to intuitively tune these parameters using "auditory displays" [5]. With it, musicians tune their network connection much like they do their instruments, using their ears. The implementation is part of the JackTrip application [3], a software for low-latency, high quality and multi-channel audio streaming over TCP/IP Wide Area Networks (WAN).

The design and architecture is first geared towards implementation of this QoS evaluation method. The architecture has also been extended to provides other types of service. In particular, a central "mixing hub" to control audio in a concert where multiple locations are involved.

[1] In particular, we use the User Datagram Protocol (UDP) which is part of the TCP/IP protocol suite.

2 QoS EVALUATION METRICS

Cromer gives a good definition of QoS:

> "The term *Quality of Service (QoS)* refers to statistical performance guarantees that a network system can make regarding loss, delay, throughput, and jitter." [7, p. 510]

Most of the networks available today are *best effort delivery*, i.e., don't provide any specific level of QoS. As such, this infrastructure can be problematic since sound is unforgiving in regard to packet loss and jitter; any lost data is immediately audible. In evaluating a particular connection, we want to know "instantaneous" QoS, i.e., assessing its quality at any given moment. Users should be able to adjust their settings to achieve the optimal quality given the current bandwidth and congestion conditions. This should be convenient and a conscious part of setting up, it should also be monitored with regard to longer-term changes: a connection that is perfectly clean at 1:00 a.m. can become congested at 9:00 a.m. A bad connection today can be a surprisingly good one a year from now when intermediate network upgrades are put in place, or when the user asks that their service be enhanced.

A connection is presently either tuned by trial and error, or is set automatically by an adaptive mechanism that changes the data rate depending on bandwidth availability [11]. Adaptive methods are typically found in unidirectional streaming and have a disadvantage for bidirectional high-quality audio. Latency is a parameter we want to keep constant. To accommodate changing amounts of jitter, adaptive methods can arbitrarily increase and decrease the local buffering, affecting total latency in a way that is very disruptive for musical performance.

We describe an implementation of a tool that let musicians tune a connection completely by ear. Parameters like

buffer size, sampling rate, packet size, and packet redundancy among others, can be adjusted using this "auditory display" mechanism.

2.1 *Pinging* the network, acoustically

The advantages of evaluating very fine-grained jitter and packet loss using these "auditory displays" have been previously discussed in the literature [5]. The method consists of listening to a pitched sound in order to assess delay, jitter, and loss. The procedure produces a tone by recirculating audio in the network path and thus allows for fine-grained listening of the packet flow [2] . The acronym, SoundWIRE, describes the technique used in this project, sound waves on the Internet from real-time echoes. In principle, it uses the Karplus-Strong plucked string synthesis algorithm [9] and simply replaces string delay lines running in local host memory with network memory.

This technique can be extended to incorporate different-sounding auditory pings using other physical models [6], but the underlying approach is the same. In the case of, e.g., a string physical model, musicians want to tune their connection to get a sounding instrument that has the highest possible pitch (low delay) without vibrato (jitter). Users also want to minimize extraneous impulses coming from packet loss.

In the next section (Sec. 3), we present an architecture of a server that clients can use to evaluate and tune their connection solely based on auditory feedback, much like guitar players tune their instruments.

3 MULTI-CLIENT CONCURRENT SEVER

We extended the JackTrip platform to include a system for QoS evaluation. The new architecture provides a multi-client concurrent server that can be used to provide QoS evaluation service, or a central network/mixer hub, among other uses. Taking advantage of multi-core computers, it is possible to run concurrently hundreds of clients with uncompressed real-time audio and processing plugins.

3.1 Server architecture

The User Datagram Protocol (UDP) is a connection-less protocol and consequently identification of a new client's IP number has to be done on a packet-per-packet basis. Several techniques to deal with multiple clients connecting are discussed in the literature [13], but no standard exists as in the case of Transmission Control Protocol (TCP) servers (see [7] or [10] for a good description of the differences between TCP and UDP protocols.)

[2] The granularity is determined by the sampling rate and the packet size. e.g., at 48kHz and 64 samples/packet, the granularity is 1.3 milliseconds.

Two different but related techniques are implemented. The first relies on a "smart" client which can change to a new server port number after being assigned one for exclusive communication. The second uses Linux's *iptables* rules to route clients into local sockets. The former technique has the advantage of being portable (works currently on both Linux and OS X, and should be easily portable to Windows), is lightweight and doesn't require root privileges. In turn, it expects its clients to change connection ports. The latter technique (Linux only) requires *iptables* privileges but provides a mechanism whereby "dumb" clients (e.g., embedded systems) can connect to a unique IP/Port pair without any change to their behavior. It is also computationally more expensive because the kernel has to perform a port redirect by source IP number for every packet.

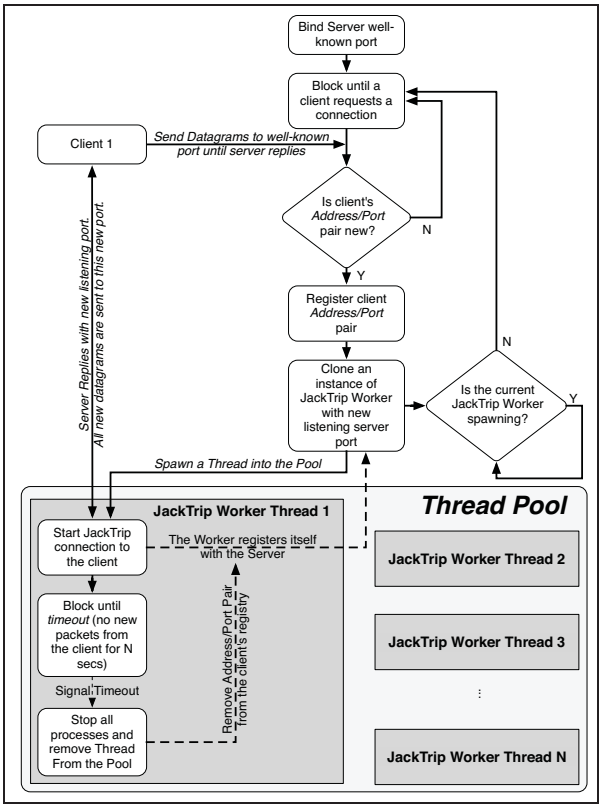

Figure 1. Multi-client concurrent server algorithm

Figure 1 describes the architecture of the system. The server listens on a well-known port for client connection requests. For every new request, the server has to check if the originating address/port pair is new. If it is, it registers it in an array of active address/port pairs and blocks the requests of new clients while this one is being processed. It then allocates a new port to communicate exclusively with this client, and informs it of the new port. The client then

stops sending packets to the well- known port and starts to send them to its own assigned one. From then on, the whole JackTrip process is sent to a thread pool and runs independently, in its own thread. The server is freed to wait for new client requests. The thread runs until the client stops sending packets (or the server doesn't receive them) for a certain amount of time. At that point a signal is emitted and the server deletes the client IP/port pair from the active clients registry and removes the process from the thread pool. The implementation is written in C++ using the Qt libraries [1] for networking and multithreading.

The architecture that uses Linux's *iptables* is similar, except that all port determination work is on the server side, and packets are redirected to a local IP/Port pair assigned exclusively to the client. It doesn't need to be notified of a new port.

4 SERVER APPLICATIONS

4.1 Quality of Service (QoS) evaluation

Each connection between the client and the server recirculates audio and implements a Karplus-Strong string model [9]. This configuration has been discussed in detailed previously [6]. Figure 2 shows a basic implementation of the algorithm. The ipsi-lateral host (which in our system corresponds to the server) generates excitations (plucks or noise bursts) that are "echoed" back from the contra-lateral host (the client) recirculating in a loop that includes a low-pass filter (LPF).

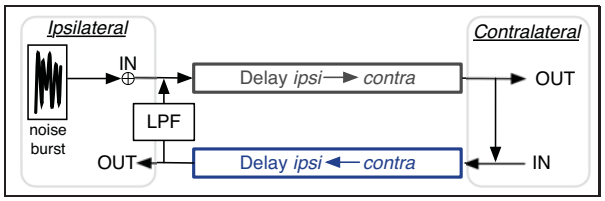

Figure 2. Karplus-Strong algorithm implemented in the network path recirculating audio.

To test a connection, a client connects against a known server IP number (e.g., CCRMA at Stanford). The path is sonified with this string model. As the network delay increases, the pitch of the sting will be lower. Variances in the latency will be perceived as vibrato of the string model. Packet losses are translated into impulsive types of sounds (for the case when the receive plays zeros when it doesn't receive a packet) or into wavetable type of sound (for the mode when the system keeps looping through the last receiving packet) [3] .

Providing this service for intuitive and quick evaluation of connection QoS is the original intended application of this technology. By connecting to the server and "listening" to the path, users can tune their connection to its optimal settings. As mentioned above, there's a trade-off between latency and sound quality. In the presence of jitter/vibrato, the local buffering has to be increased to avoid late packets, but at the same time we don't want to increase it too much (to avoid unnecessary latency). Doing this by trial and error requires experience and can be frustrating for new users. If, in turn, musicians can listen and tune the connection in the same way they tune an instrument, the setup is much faster and intuitive. Again the goal for the musician, is that they want to tune their pitch to be as high as possible (lower latency) with the smallest possible vibrato (jitter).

4.2 Star topology connection/mixing hub

Mixing and managing a remote connection when more than two sites are involved can be very complicated. Engineers have to deal with audio channels coming from different places (sometimes on confusingly different channels), all with different levels. They also need to make sure local audio is sent to the peer with proper gains. A solution to centrally manage these types of situations designates a master location which can mix and/or relay all the channels and send them back to the respective connected peers.

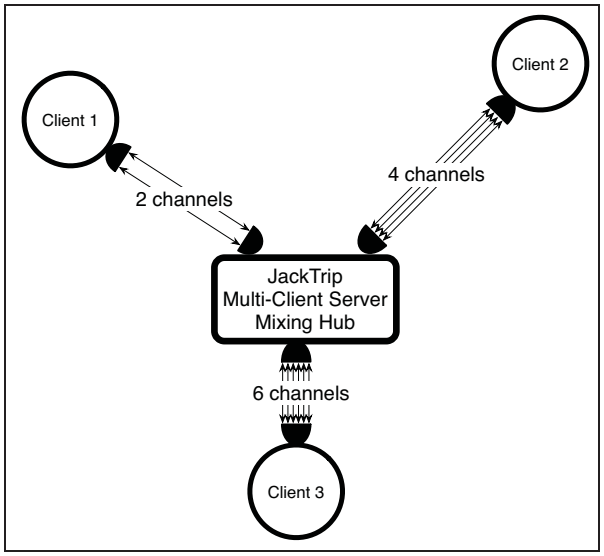

Figure 3. Multi-client server as a hub

The present sever implementation allows a server to dynamically connect and disconnect audio from different clients. Each client can have a different number of channels and different network tuning parameters. [4] In this case

[3] More details on these two modes can be found in [3].

[4] JackTrip presently uses Jack [2] as its audio host. This has the limitation that sampling rate and buffer are fixed at *Jack start-time* and cannot be tuned after the server has started.

the server will act as a "hub" between several locations.

Figure 3 illustrates this for an example with three clients. The server can mix and re-route all the audio channels between the clients, hence allowing a multi-site performance with one site acting as a master relay service and/or mixer.

5 CONCLUSIONS AND FUTURE WORK

The first decade of the 21st century evidenced a dramatic increase in the speed and reliability of high-speed networks. This increase is expected to continue. We have provided a system for musicians to tune and optimize their connections against a reference server in a way that lets them adapt to their given network situation. The server can also be used to interconnect multiple sites with arbitrary number of channels, and be a "mixing hub" that distributes audio to all the locations from a central place.

Scalability in network performance is a big issue that still needs to be solved. Learning how to connect hundreds or even thousands of remote locations for a global-jam session is a pending goal. Multicast at the network layer would provide a solution for a fully connected peer-to-peer mesh. Clients would select from a list of peers they want to connect with, and then send just one packet via multicast (using its underlying network layer implementation). Network routers and switches determine when a copy needs to be made. AccesGrid implements this [8] for a fixed number of audio channels, however this infrastructure is not yet ubiquitous. Furthermore, when the number of audio channels and other settings differ among the clients, a new and consistent solution is required so that they can inter-operate.

Scaling up and distributing physical models embedded in the network path can also serve to perform "global string network symphonies", where the global network becomes the instrument itself, an instrument distributed throughout the world.

6 ACKNOWLEDGMENTS

This work was carried out in cooperation with MusicianLink, Inc. and funded by National Science Foundation Award Grant No. IIP-0741278 with a sub-award to CCRMA. See the online Final Report, Technical Research Summary. [5] Fernando Lopez-Lezcano and Carr Wilkerson from CCRMA have provided continuous assistance in the implementation and server infrastructure setup.

7 REFERENCES

[1] (2008–2009) Qt Software. [Online]. Available: http://www.qtsoftware.com/

[2] (2009) JACK: Connecting a world of audio. [Online]. Available: http://jackaudio.org/

[3] Cáceres, J.-P. and Chafe, C., "JackTrip: Under the hood of an engine for network audio," in *Proceedings of International Computer Music Conference*, Montreal, 2009.

[4] Chafe, C. and Gurevich, M., "Network time delay and ensemble accuracy: Effects of latency, asymmetry," in *Proceedings of the AES 117th Convention*, San Francisco, 2004.

[5] Chafe, C. and Leistikow, R., "Levels of temporal resolution in sonification of network performance," in *Proceedings of the 2001 International Conference on Auditory Display*. Helsinki: ICAD, 2001.

[6] Chafe, C., Wilson, S., and Walling, D., "Physical model synthesis with application to internet acoustics," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Orlando, 2002.

[7] Comer, D. E., *Internetworking with TCP/IP, Vol 1*, 5th ed. Prentice Hall, Jul. 2005.

[8] Daw, M., "Advanced collaboration with the access grid," *Ariadne*, vol. 42, Jan. 2005. [Online]. Available: http://www.ariadne.ac.uk/issue42/daw/intro.html

[9] Karplus, K. and Strong, A., "Digital synthesis of Plucked-String and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, 1983.

[10] Peterson, L. L. and Davie, B. S., *Computer Networks: A Systems Approach, 3rd Edition*, 3rd ed. Morgan Kaufmann, May 2003.

[11] Qiao, Z., Venkatasubramanian, R., Sun, L., and Ifeachor, E., "A new buffer algorithm for speech quality improvement in VoIP systems," *Wireless Personal Communications*, vol. 45, no. 2, pp. 189–207, Apr. 2008.

[12] Renaud, A. B., Carôt, A., and Rebelo, P., "Networked music performance: State of the art," in *Proceedings of the AES 30th International Conference*, Saariselkä, Finland, 2007.

[13] Stevens, W. R., Fenner, B., and Rudoff, A. M., *Unix Network Programming, Volume 1: The Sockets Networking API (3rd Edition)*, 3rd ed. Addison-Wesley Professional, Nov. 2003.

[5] http://ccrma.stanford.edu/~cc/pub/pdf/qosServer-nsfFinalReport.pdf