PHYSICAL MODEL SYNTHESIS WITH APPLICATION TO INTERNET ACOUSTICS

Chris Chafe, Scott Wilson, Daniel Walling

Stanford University
Center for Computer Research in Music and Acoustics
Stanford, California 94305
(cc, rswilson, dwalling @ccrma.stanford.edu)

ABSTRACT

Distributed physical models of musical instruments have been used to acoustically "ping" Internet connections between two network hosts. Sound waves propagated through Internet acoustics behave just as in air, water or along a stretched string. In this case, a musical synthesis technique creates waves on the Internet path between two hosts. When waves recirculate between two endpoints, a musical tone is created if the round trip travel time lies within the range of our pitch sense (roughly $250\mu s$ to $50ms$). The resulting tones provide a quick and intuitive evaluation of quality of service (QoS), displaying its significant aspects including latency, jitter and packet loss. Stable, clear tones with high pitch indicate good end-to-end capabilities that a path must support for immersive, real-time applications. A "network harp" recently demonstrated 320 synthesized strings oscillating across the Western half of the Internet2 Abilene Network.

1. INTRODUCTION

As a point of departure, this paper describes in detail a plucked string algorithm which uses the Internet as its acoustic medium. The new method is quite general and can be adapted to any kind of acoustical system whose physics include wave propagation through a vibrating medium. The algorithm is able to "pluck the Internet" using a simplified 1D waveguide technique, but can be extended to 2D and 3D systems, such as drum membranes [9] and reverberation chambers [5], [8].

Sound is transmitted along an Internet path in the form of small-sized packets of audio samples emitted at regular intervals [1]. The smallest possible packets carry individual samples and flood the path with new packets entering at the audio sampling rate (approximately $25\mu s$ at $fs = 44.1kHz$). Present implementations require that packets contain multiple samples and enter the path at intervals determined by one of two parameters, either the computer host's audio buffer size, or the round trip time (RTT) of the path. This choice of structure distinguishes the two architectures to be described.

This work sponsored by NSF Award 9977246, "QoS Evaluation with SoundWIRE."

RTT for small packets is reported by the Internet Control Message Protocol (ICMP) *ping* command [3]. A measurement is instigated by *host A* when it sends an echo request message to *host B*, who then returns an echo reply message. The initiator typically clocks several echo times received from a series of requests emitted at $1s$ intervals and prints statistics that include average time, high and low values and percent packet loss.

The advantage of using the present acoustic technique, SoundWIRE (short for "Sound Waves on the Internet from Real-time Echoes"), is that we can apply our listening capabilities to analyze network behavior at much finer granularities than the standard ICMP ping is suited for. SoundWIRE reveals the fine structure of network jitter and is a real-time monitoring technique. Fine-grained temporal behavior is mostly lost when statistics are averaged or falls out of real time when "post-processed" for graphical analysis by eye. A discussion of the advantages of sonification (auditory display) of network behavior appears in [2].

One aspect that distinguishes *next-generation networking* is the ability to link spatially separated real-time processes and create a "virtual computer" in which the network functions as a computer backplane or signal bus. SoundWIRE works in this way, substituting network delay for on board memory and distributing its signal processing across separate hosts. Until now, physical models of musical instruments have been computed on a single processor, simulating acoustic delay with delaylines in memory.

2. PHYSICAL MODEL SYNTHESIS

A simple plucked string physical model is comprised of two unit generators, a delayline and a low-pass filter, forming a feedback loop which becomes a hi-Q, IIR resonator (UG is a term used in music synthesis for a signal processing element). An input is provided to excite the loop resonance, Fig. 1. Excitation is a "pluck-like" pulse signal which is triggered either in real-time by user interaction, or from a process such as a musical score. Software synthesizers can run the model in real time sending a stream of samples to the computer's sound output.

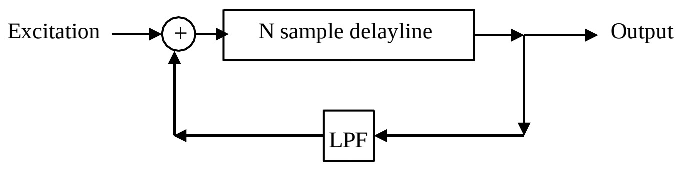

Fig. 1. Plucked string synthesis patch.

The strength of the physical analogy for plucked string synthesis depends on model detail. Sophisticated models accurately calibrate traveling wave motion, physical distance, dependence on sound velocity, acoustic impedance, etc. The abstract model above makes no such analogies and simply treats the length of the delayline as its "physical length."

The two UG's represent the linear components of a stretched string resonator separated into a pure delay (with length equal to one round trip on the string) and a low-order filter (the string's damping characteristics). The model is an early lumped-circuit computer music "patch" and was first implemented by Karplus and Strong [6] with many extensions since [7], [4]. Any of these versions can be realized as SoundWIRE implementations, which simply "folds the network" into a new kind of delay UG, Fig. 2. Other patches which lend themselves to becoming Internet instruments consist of a feedback loop around a pure delay which governs the pitch of the synthesis. Examples include bowed strings and wind instruments without valves or tone holes, e.g. trombone, slide flute.

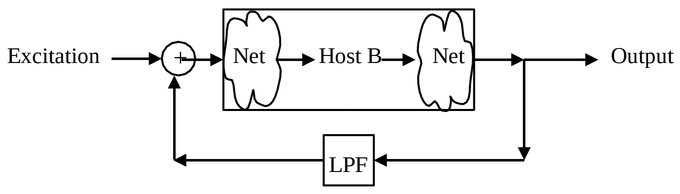

Fig. 2. Network audio path in place of delayline.

3. TWO SOUNDWIRE METHODS

The vibrating medium which SoundWIRE sets into oscillation has a unique property: its speed of sound changes unpredictably. Sound velocity in air changes only slightly (and very slowly) with temperature and humidity, in water with depth, temperature and salinity. String players tune their instruments by altering tension, but except for "whammy bars" don't continue to change the sound velocity during performance. As a way of understanding the difference from physical acoustics, imagine a sequence of packets emitted into the medium at the sampling rate, each containing a single audio sample. In a synchronous network, each would arrive at the destination properly spaced in time. However, the asynchronous nature of Internet Protocol (IP) networking makes it likely that samples will arrive late or early relative to a regular sample clock (playback buffers are used to sort this out and produce properly clocked samples at the receiving end).

The time of travel for a wave across the plucked string's delayline in Fig. 1 depends on sampling rate, fs and length N_{samps} in samples. Let $v = fs$ samples per second, then the

$$\text{Delay period:} \quad t = \frac{N_{samps}}{v} \tag{1}$$

Feedback of the delayline output to its input creates a resonance whose

$$\text{Frequency:} \quad f = \frac{1}{t} Hz \tag{2}$$

For example, a delayline buffer of $N_{samps} = 441$ samples introduces a $10ms$ delay at a sampling rate of $44.1kHz$. A plucked string tone obtained by looping around the delayline would create an oscillation frequency of $100Hz$.

3.1. Speed of sound on the net

A wave traveling from host A to host B and back in Fig. 2 travels one round trip of length L. The time this takes can be expressed in terms of round trips per second (L/s). The theoretically fastest speed corresponds to receiving an audio sample back from the network delay one sample tick later, in other words with a velocity at which, in one second, it would traverse $fs\ L$ units.

$$\text{Velocity:} \quad v = fs\frac{L}{s} \tag{3}$$

$$\text{Network delay period:} \quad t = \frac{L}{v} \tag{4}$$

In present implementations, non-unity packet sizes P_{samps} add a fixed delay, and a playback buffer of N_{packs} packets may lengthen the delay.

$$\text{Packet delay:} \quad t_{packs} = \frac{P_{samps} * N_{packs}}{fs} \tag{5}$$

And a variable source of delay t_{rtt} is added by the network, for a total:

$$t_{total} = t_{packs} + t_{rtt} \tag{6}$$

such that the oscillating frequency for SoundWIRE is

$$f = \frac{1}{t_{total}} Hz \qquad (7)$$

As an example, the network harp test (SC2001 Conference, Denver) was run with $P_{samps} = 16$ and an "expandable" playback buffer which varied from $N_{packs} = 50$ up to $N_{packs} = 100$ depending on load. t_{packs} varied from a minimum delay of $18ms = (16 * 50)/44100$ up to a maximum of twice that amount, $36ms$.

Adding the path delay from Denver to Stanford, nominally observed to be $t_{rtt} = 25ms$, the resulting oscillation period, seen in Fig. 3 is in the predicted range, varying from 44 to $58ms$ (sounding frequencies of $22.7$to $17.2Hz$).

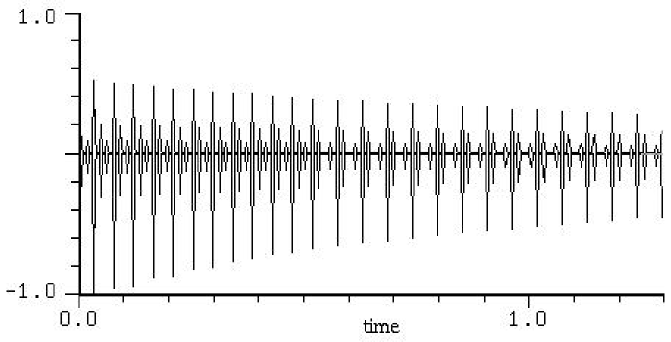

Fig. 3. Plucked Internet path, Denver, CO to Stanford, CA.

3.2. Time quantization and packet size

SoundWIRE is continuously displaying to the ear the time for a wave to complete a trip around the path. In a perfect network v is constant and the resulting tone is perfectly periodic. Absent the jitter in real networks, lack of frequency fluctuation is quite apparent.

Because SoundWIRE is implemented with various buffers, the amount of change to RTT that can be detected is bounded. At its theoretical best, quanta correspond to the sample rate.

$$t_{rtt}min = \frac{1}{fs} \qquad (8)$$

Actual time resolution is a function of packet size and playback buffer dynamics and corresponds to t_{packs} in eq. 5.

Playback buffer dynamics determine how the application responds to conditions of packet underrun or overrun on the receiving side. A number of packets can be elastically buffered, but conditions which exceed the buffer capacity may be caused by network slowing or speeding. The extent to which elasticity absorbs RTT jitter is an application parameter.

3.3. Implementations

Fig. 4 graphs the (recirculating) time delay of the plucked string in Fig. 1. A client / server version is shown in Fig. 5, in which packet size is matched to the host A's real-time audio buffer size (ABS) (typically chosen to lie in the range of $16-256$ samples with equivalent rates of $300\mu s-5ms$, this rate determines how often the application's sound thread will service the sound hardware). A circular buffer for received packets adds to the total delay, and frequency follows eq. 7, above. UDP packets are emitted at regular intervals from the send side shown and a networking thread handles reception, asynchronously.

The second implementation is a modified version of ICMP ping and does not require a server on the far end. A succession of *ICMP_ECHO* messages contain packets of audio in their data segments. The packet length is continuously adjusted according to instantaneous RTT measured using the message immediately preceding. The method sets a limit on adjustment step size and can minimize quantization to single samples as shown in eq. 8.

Both methods have parameters for *number of strings, number of audio channels, additional per-string delay UG, low-pass filter coefficient, pluck amplitude, pluck position,* and *pluck rate.* Additionally, the ABS implementation provides parameters for audio buffer size ($= P_{samps}$), N_{packs}, and assignment of any UG processing to either the client or server.

In practice, at $fs = 44.1kHz$, ABS is suited for WAN RTT's which are significantly longer than t_{packs}, roughly, $1ms$ to $150ms$. The ICMP RTT-based method operates well for LAN paths in the $> 50\mu s$ range.

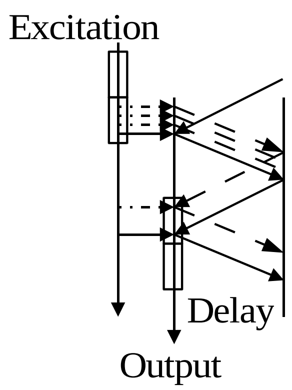

Fig. 4. Plucked string delay is constant (from Fig. 1)

3.4. Network Harp

Network Harp / SoundWIRE testing at the SC2001 SCinet Bandwidth Challenge exceeded expectations in terms of sheer number of simultaneous audio channels that were sustained

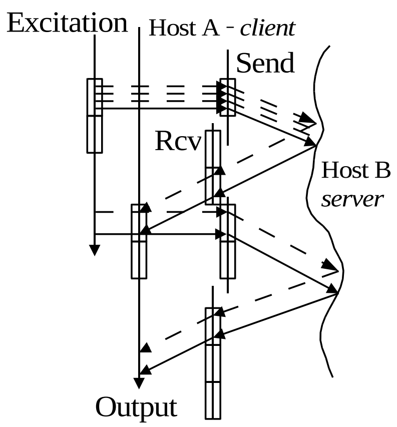

Fig. 5. SoundWIRE with ABS packets, packet delay adds to network delay. Network sound velocity is varying.

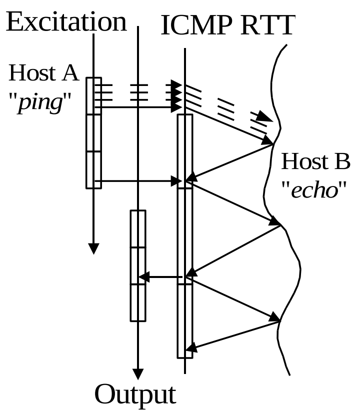

Fig. 6. SoundWIRE over ICMP with variable RTT-sized packets.

between Denver and Stanford. The goal was to find an upper limit to the number of channels (using the ABS method). The mark we hit was 320 strings ($451.6Mbps = 16bit * 44.1kHz * 2directions * 320$) running 3 client / server pairs. Multiple strings can be instantiated on a given CPU by multiplexing their signals within each packet and mixing their outputs into the audio output. There is reason to believe UDP load on the network could have been increased further before we would have noticed audio packets being dropped for network reasons.

4. CONCLUSION

The Bandwidth Challenge test also demonstrated that synthetic sound is indeed a quick, revealing and intuitive measure for network performance. For example, a single channel (single harp string) sound was extremely stable, but (and this is a good thing) would occasionally alter its pitch slightly as jitter became significant. With eight channels, this was much more apparent. We also found that our application was able to gracefully recover after severe loss from congestion.

5. ACKNOWLEDGMENTS

Thanks are due Fernando Lopez-Lezcano, Juan Reyes and Ron Roberts for making the network harp test a success.

6. REFERENCES

[1] C. Chafe, et al., "A Simplified Approach to High Quality Music and Sound Over IP," in *Proc. COST-G6 Conference on Digital Audio Effects (DAFx-00)*, Verona, 2000.

[2] C. Chafe, R. Leisikow, "Levels of Temporal Resolution in Sonification of Network Performance," in *Proc. 2001 Intl. Conference on Auditory Display (ICAD)*, Helsinki, 2001.

[3] D. Comer, *Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture*, Prentice Hall, 2000.

[4] J. O. Smith, "Efficient Synthesis of Stringed Musical Instruments," in *Proc. 1993 Intl. Computer Music Conference (ICMC)*, Tokyo, pp. 64–71, Computer Music Association, 1993.

[5] J. O. Smith, "A New Approach to Digital Reverberation Using Closed Waveguide Networks," in *Proc. 1985 Intl. Computer Music Conference (ICMC)*, Vancouver, pp. 47–53, Computer Music Association, 1985.

[6] K. Karplus and A. Strong, "Digital Synthesis of Plucked-String and Drum Timbres," in *Computer Music Journal*, 7(2), reprinted in C. Roads (ed.), *The Music Machine*, MIT Press, 1989.

[7] C. R. Sullivan, "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback," in *Computer Music Journal*, 14(3):26-37, 1990.

[8] S. Van Duyne and J. O. Smith, "The 3D Tetrahedral Digital Waveguide Mesh with Musical Application," in *Proc. 1996 Intl. Computer Music Conference (ICMC)*, Hong Kong, Computer Music Association, 1996.

[9] S. Van Duyne and J. O. Smith, "Physical Modeling with the 2D Digital Waveguide Mesh," in *Proc. 1993 Intl. Computer Music Conference (ICMC)*, Tokyo, pp. 40–43, Computer Music Association, 1993.