Silence Is Golden? - The Effects of Silence Deletion on the CPU Load of an Audio Conference

Mark Claypool, *claypool@cs.umn.edu*
John Riedl, *riedl@cs.umn.edu*
Department of Computer Science, University of Minnesota
+1 612-624-7372

Abstract

This paper seeks to identify improvements that reduce audioconference CPU load. A major contribution is the comparison of the performance benefits of five potential audioconference improvements: faster CPU, faster communication, better compression, Digital Signal Processing (DSP) hardware, and Silence Deletion. To compare audioconference CPU load, we develop a model that identifies components of a typical audioconference. We hypothesize that silence deletion will improve the scalability of audio more than any of the above four improvements. We parameterize our model with measurements of the actual component performance. Overall, we find audioconference CPU loads with silence deletion scale better than audioconference CPU loads with any of the other four improvements. Techniques based on DSP hardware alone do not scale as well as silence deletion alone. However, DSP based silence deletion and compression together scale better than any other technique. These results hold even when using compression and even for ten times faster processors, networks and DSP hardware.

1 Introduction

Today, there are many computer applications that use audio. Electronic mail includes audio along with text [Thom85]. Multimedia editors enhance text documents with audio annotations [Cave90]. Internet Talk Radio spreads audio across the world. Movies, containing audio in addition to video, are starting to grace consoles everywhere [Rowe92]. And audioconferences synchronously link workstations [Ried93, Mash93, Schu92].

Why are audioconferences becoming so important? Hearing is one of our is one of our strongest senses. Thus, sound is one of our most powerful forms of communication. If we wish to use the flexibility and power of computers to support communication and collaboration, then they must support audio data.

Audioconferences frequently support other collaborative tasks that are themselves CPU-intensive. Therefore, efficient CPU use is essential. Our goal is to identify improvements that reduce audioconference CPU load. The major contribution of this paper is an experimentally-based comparison of the CPU performance benefits of five potential audioconference improvements:

- *Faster CPU.* How much do faster processors benefit audioconference CPU load?
- *Faster communication.* How much do more efficient network protocols and faster network speeds benefit audioconference CPU load?
- *Better compression.* How much do improved compression techniques benefit audioconference CPU load?
- *Hardware support.* How much does Digital Signal Processing (DSP) hardware reduce audioconference CPU load?
- *Silence deletion.* How much does removing the silent parts from a conversation reduce audioconference CPU load?

We focus particularly on a comparison of the first four areas to silence deletion. Silence deletion detects silence, only transmitting the sound of the person who is presently speaking. Although silence deletion algorithms take additional processing time, they may yield a net savings by reducing the amount of data that must be communicated. Therefore, we hypothesize silence deletion will improve the scalability of audioconferences more than any of the other four improvements.

Our analysis can help direct research in networks, multimedia and operating systems to techniques that will have a significant impact on audioconferences. In addition, our approach may generalize to video and other forms of multimedia.

Section 2 of this paper describes related work in distributed multimedia. Section 3 introduces our model of audioconference CPU load. Section 4 details micro experiments that measure the CPU load of each component. Section 5 describes macro experiments that test the accuracy of the predictions based on the micro experiments. Section 6 analyzes the experiments and projects the results to future environments. And Section 7 summarizes our conclusions and future work.

2 Related work

2.1 Audioconference experiments

There has been a variety of experimental audioconference work. Casner and Deering perform a wide-area network audioconference using UDP multicast [Casn92]. They find disabling silence suppression increases average bandwidth and eliminates the gaps between packets that give routers a chance to empty their queues. They recognize that experimenters need better tools to measure audioconference performance. Our model may be one of the tools they seek. They conjecture that ubiquitous multicast routing support can greatly reduce network and CPU loads. In addition, they described several on-going experiments in which readers can participate.

Terek and Pasquale implement an audioconference with an Xwindow server [Tere91]. They describe the structure and performance of their system. In particular, they describe a strategy for dealing with real-time guarantees.

Gonsalves predicts that without software or protocol overhead, a three Mbps Ethernet could support 40 simultaneous 2-way 64Kbps conversations [Gons83]. Thus, if our results show what is needed to enable the CPUs to handle the conversation loads, the networks can.

Riedl, Mashayekhi, Schnepf, Frankowski and Claypool measure network loads of audioconferences using silence deletion [Ried93]. They find silence deletion significantly reduces network loads. We analyze how silence deletion affects CPU loads.

2.2 Measuring CPU load

Jeffay, Stone and Smith discuss a real-time kernel designed for the support of multi-media applications [Jeff92]. They achieve some real-time guarantees through utilizing close to eighty percent of the CPU. Our results may indicate methods that can trim audioconference CPU loads, while achieving the same guarantees.

Lazowska parameterizes queuing network performance models to assess the alternatives for diskless workstations [Lazo86]. He uses a counter process to measure CPU loads. We use a similar process (see Section 4.1 of the present document).

Riedl, Mashayekhi, Schnepf, Frankowski and Claypool measure the CPU load of an audio conference with no floor control and no silence deletion using a counter process [Ried93]. They show the CPU loads quickly become prohibitive under increasingly large audioconferences. We provide further analysis and a component breakdown of the CPU load.

2.3 Analyzing UDP

Cabrera measures throughput for UDP and TCP for connected Sun workstations [Cabr88]. In analyzing the call-stack for UDP in detail, he determines the checksum at the receiving end takes the most CPU time; this checksum may not be needed for adequate quality sound.

Kay and Pasquale measure delay at the CPU end for sending UDP packets for a DEC 5000 [Kay93]. They find checksumming and copying dominate the processing time for high throughput applications.

Bhargava, Mueller and Riedl, divide communications delay in Sun's implementation of UDP/IP into categories such as buffer copying, context switching, protocol layering, internet address translation, and checksum implementation [Bhar91]. They find socket layering and connection are the most expensive categories. We use their analysis of their kernel buffering techniques in defining our experiment.

2.4 Using silence deletion

Rabiner views voice as a measure of energy and presents an algorithm for discovering the endpoints of words [Rabi75]. Henning Schulzrinne implements an audioconferencer with silence deletion [Schu92]. We adapt and build upon their view of voice and silence deletion algorithms.

2.5 Digital audio compression

Pan surveys techniques used to compress digital audio signals [Pan93]. He discusses µlaw, Adaptive Differential Pulse Code Modulation (ADPCM) and Motion Picture Experts Group (MPEG), compression techniques which are not specifically tuned to human voice. Gerson and Jasiuk describes techniques to improve performance of Code Excited Linear Prediction (CELP) Type coders, compression techniques tuned to human voice [Kroo92]. We predict the CPU load of audioconferences using such compression techniques.

3 Model

Figure 1 depicts our model of an audioconference CPU load. Our model is based on the components of reading,

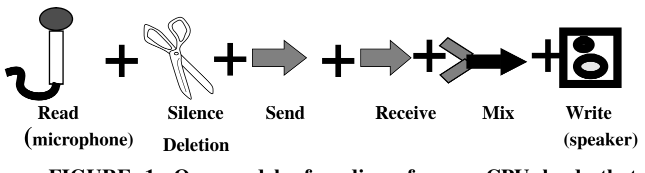

FIGURE 1. Our model of audioconference CPU load, that includes reading from the audio device, silence deletion, sending and receiving packets, mixing sound packets that should be played simultaneously and writing to the audio device.

silence deletion, sending, receiving, mixing and writing. Reading is the CPU load for taking the digitized sound samples from the audio device. Silence deletion is the CPU load for applying one of the deletion algorithms to the read sample. Sending is the CPU load for packetizing the sample and sending it to all other stations. Receiving is the CPU load for processing all incoming packetized samples. Mixing is the CPU load for combining sound packets that arrive simultaneously. Writing is the CPU load for delivering the incoming samples to the audio device. Our model asserts we can predict audioconference CPU load from the sum of the above components.

Silence deletion removes silent parts from speech. Experiments have shown that silence deletion substantially reduces network load for two reasons: [Ried93]

- In a typical `N` person conversation, at any given time one person is talking and `N-1` are silent. With silence deletion, only the talking person's packets are sent; each workstation must send only `1/N` of the packets on average. Without silence deletion, all packets are sent; each workstation sends N of the packets. A linear increase in participants results an N^2 increase in network load.
- Pilot tests suggest about 1/3 to 2/3 of the digitized speech data can be identified as pauses between words or sentences. Silence deletion may remove these pauses.

Although silence deletion algorithms themselves take additional processing time, they may yield a net savings in total CPU load by decreasing the CPU costs of the communication.

We consider four common silence deletion algorithms [Clay93]:

1. HAM uses the energy in each byte (`b[n]`).

   $$e(n) = b[n]$$

   It removes chunks with enough consecutive byte energies below a threshold. HAM decreases a counter for each byte with energy below the threshold. When the counter reaches zero, the chunk is not sent on. Any byte above the threshold resets the counter to a maximum value.

2. Exponential also uses the energy in each byte. It removes all exponentially weighted byte energies that are below a threshold. The most recent bytes are weighted most heavily. Exponential decreases a counter value according to the decay value. When the counter drops below the threshold, the bytes are not sent on. The decay determines the exponential change. When decay is small, the counter fluctuates quickly. When decay is large, the counter fluctuates slowly. Our pilot tests indicate that this algorithm often yields poor sound quality.

3. Absolute uses the average energy over many bytes to determine silence.

   $$P(n) = \left(\frac{1}{size}\right) \sum_{i=1}^{size} |b[i]|$$

   It removes all chunks with energy less than a threshold. It is effective when signal-to-noise ratio is very high. This may occur in a recording studio or with very high fidelity magnetic tape. However, it is not practical in real-world situations[Rabi75]. Pilot tests of our own showed that this algorithm does, indeed, often yield poor sound quality.

4. Differential uses the changes in energy of each byte.

   $$\Delta e(n) = b[n] - b[n-1]$$

   All chunks with changes below a threshold are interpreted as silence. The algorithm keeps a counter of the number of non-changes. If there are too few changes (the counter gets higher than a threshold), then differential does not send the chunk on. When a change is encountered, the counter is reset to zero.

4 Micro experiments

4.1 Design

Our micro experiments were designed to measure the CPU load of audioconference components. We chose two Sun workstations, the 20 Mhz SLC and the 40 Mhz IPX, to test if the components of the audioconference scale with processor speed.

We used a process that increments a `double` variable in a tight loop to measure the CPU load of the individual components. We did not use the Unix `time` command because the reporting of per-process CPU consumption by most operating systems is unreliable. Often, the system gives an incorrect account of interrupt level processing and fails to capture processor degradation from DMA [Lazo93]. In our pilot tests, the counter process attributed 10 out of 200 seconds to sending packets, while the `time` command only credited 0.3 seconds, a factor of 30 discrepancy.

To obtain a baseline, we ran the counter process on a bare machine. This gave us the CPU potential for the machine. The difference in the bare count and the count with another process is the process-induced load. To verify that the counter process does indeed accurately report loads of CPU bound processes with which it runs concurrently, we ran the counter process with 1, 2, 3 and 4 other counter processes. We expect the counter value to be `(count on bare machine) × 1/(N+1)`, where N is the number of other counters running. Figure 2 shows the result of this experiment. The predicted values were within the confidence intervals for the measured values.

To obtain the CPU load for each component of the model, we ran the counter process in conjunction with a

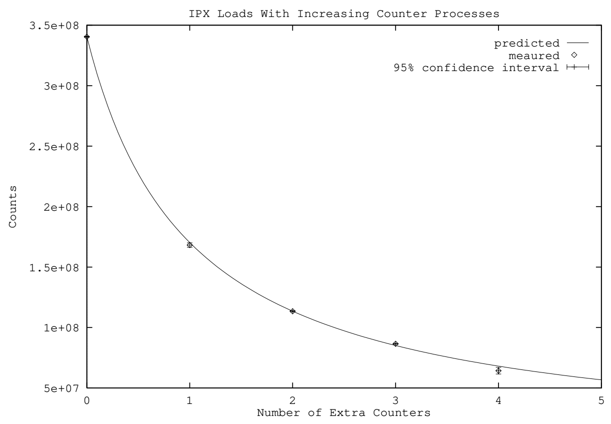

FIGURE 2. IPX loads with increasing counter processes. The points are from 5 samples each with 95% confidence intervals. The largest interval is 8% of the measure value. The curving line is the predicted value.

process for each of the components of the audioconference: read, deletion, send, receive, mix and write (see [Clay93] for the process descriptions).

4.2 Data collection

Since the counter process measurements are sensitive to other processes, we performed the experiments on machines in single user mode. In single user mode, the CPU runs a bare minimum of system processes and no other user processes.

We ran the counter process for 200 seconds to amortize start-up costs. To determine the number 200, we ran the counter process for increasing times and recorded the standard deviation of its counts. The standard deviations level out just before 200. At this point, the standard deviation is only 0.001% of the mean. Thus, we chose one 200 second counter run as one iteration. Pilot tests indicated that five iterations for each machine at each packet size were needed to achieve reasonable confidence intervals.

The process we used to measure the deletion algorithm was memory intensive to avoid I/O costs. In order to avoid measuring unwanted paging, we recorded the total page faults during the experiments.The number of page faults during the deletion iterations is almost always three, which we accepted as the baseline case. We decided to discard cases that had more than three page-faults, as they incur extra CPU load from paging. In our experimental runs, this situation never occurred.

The sound chunk size and the threshold levels determine how the deletion algorithms perform. We fixed them such that they deleted about 1/3 to 2/3 of live, audioconference speech (in accordance to the sound pilot tests mentioned above). We confirmed that the speech with the sound deleted was still very understandable.

The kernel changes between large and small buffers at various packet sizes has a direct influence on packet sending and receiving times [Bhar91 and Clay93]. User messages are copied into buffers in kernel space. The kernel may use one large buffer and one copy for certain messages, while it uses several smaller buffers and several copies for a slightly smaller buffer. To avoid possible nonlinearities, we collected data on 515 to 2000 byte packets, which we assume covers most audioconference packet sizes.

4.3 Results

The 200 second counts on bare machines are shown in Table 1.

TABLE 1. Bare Counts for SLC and IPX with a 95% Confidence Interval (Indicated by the Left and Right Endpoints).

Machine	Bare Count	Left Endpoint	Right Endpoint
SLC	123924595	123920808	123928381
IPX	340539937	340204150	340875723

Figure 3 shows the line equations obtained from the counter measurements for the deletion algorithms on the IPX. We can obtain similar graphs for the SLC and for other components of the model (See [Clay93]), but to avoid redundancy we do not do so here.

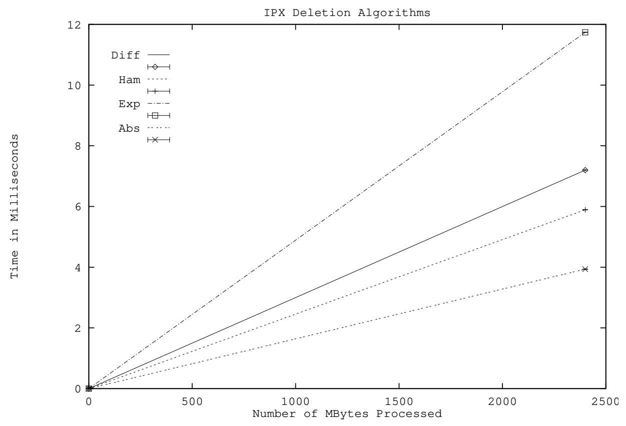

FIGURE 3. CPU time for deletion algorithms on the IPX. Th four deletion algorithms are shown for their time to process 30 seconds worth of sound. All points are shown with 95% confidence intervals.

Table 2 shows the values for the line equations for each of the audioconference components for each machine type.

TABLE 2. Values for SLC and IPX Line Fits. Units are in milliseconds.

Operation	SLC per-packet	SLC per-byte	IPX per-packet	IPX per-byte
Read	0.810	0.0145	0.597	0.00169
Absolute	0.00	0.00302	0.000	0.000164
Differential	0.00	0.00563	0.000	0.00300
Exponential	0.00	0.0130	0.000	0.00489
Ham	0.00	0.00454	0.000	0.00245
Send	0.807	0.000194	0.210	0.000100
Receive	0.910	0.000129	0.187	0.000103
Mix	0.00	0.00546	0.000	0.00245
Write	1.26	0.0137	0.726	0.00103

They `per-packet` and `per-byte` terms above pertain to the equations:

Load(component) = per-packet + per-byte * bytes

The equations are the CPU costs for each component of an audioconference from which we can project the cost of a complete audioconference (see Section 5).

5 Macro experiments

5.1 Design

In order to test the ability of our model to predict audioconference CPU loads, we measured the performance of a simple audioconferencer, *Speak*. Speak is two person, uses UDP, can employ any of the four deletion algorithms, and has little extra user-interface overhead.

We used Internet Talk Radio (ITR) files rather than real conversants. This made our experiments more reproducible and gave us a large conversation sample space from which to chose. Since the ITR files have one person speaking most of the time, the silence deletion algorithms typically deleted only 10% of the packets. The actual audio data used in these experiments does not matter, since our model is parameterized by the amount of silence deleted.

We did experiments on the five possible silence deletion methods on the SLC and two such methods on the IPX. A shell script initiated a remote Speak process and a local Speak process. One two hundred second conversation was one data point. We repeated each data point 5 times. We predict the load from the speak processes by using the micro experiment results. From the conversation length, the read size and the sample rate, we calculate the total packets read. By profiling the sound files with the deletion algorithms, we know the number and size of the packets sent, received and written. Because sound only arrives from one other Speak process, there is no mix component.

5.2 Results

Table 3 shows the results of the seven macro-experiments.

TABLE 3. Predicted and Measured Loads from the Macro Experiments with 95% Confidence Intervals (Indicated by Left and Right Endpoints). Loads are in seconds. Maximum load is 200 seconds.

Machine	Algorithm	Predicted	Actual	Left Endpoint	Right Endpoint
SLC	Absolute	45.55	44.84	44.76	44.91
SLC	Differential	54.92	49.02	48.93	49.11
SLC	Exponential	54.08	50.00	49.91	50.09
SLC	HAM	49.34	48.45	48.45	48.46
SLC	None	49.92	46.49	46.44	46.55
IPX	Differential	10.63	12.50	12.49	12.56
IPX	None	6.96	8.81	8.78	8.85

The discrepancy in predicted and actual values may be due to the unforeseen costs that occur when putting micro components together. In most cases, the predicted values are within 10% of the actual values. We therefore consider the projected results presented in Section 6 to be significant only if the differences are larger than 10%.

6 Analysis

The CPU loads in the macro experiments compared to the CPU loads predicted by adding the components of the micro experiments suggests that our model may be a reasonable way to predict the CPU load of audioconferences. It is difficult and expensive to run experiments with many people on a large number of workstations. Instead, from our micro experiments we extrapolate our results to conversations that have more silence and audioconferences that have more participants. Furthermore, we adjust the pieces of our model to compare the performance benefits of four potential audioconference improvements: faster CPU, faster communication, better compression, and Digital Signal Processing (DSP) hardware. We also compare the benefits of better silence deletion algorithms.

All analysis can be done for both the SLC and IPX and with any of the four silence deletion algorithms. To avoid repetition, we do our extrapolations on one machine type (IPX) with one deletion algorithm (Differential). We use the fastest CPU we studied to improve the quality of our extrapolations to even faster machines. Our results are largely independent of the type of silence deletion.

6.1 Increasing participants

What happens when we have an increasing number of audioconference participants? We can extrapolate the loads at each workstation to an N person audioconference. The load depends upon the number of participants, the total conversation time, the packet size, the percentage of packets deleted and the percentage of sound removed from the remaining packets. The load without silence deletion does not have the deletion algorithm component. The load with silence deletion does not have the mix component since we assume only one person speaks at a time and the deletion algorithm removes all sound of those who are not speaking.

Figure 4 shows the predicted load for an audioconference without silence deletion and an increasing number of participants. The audioconference components are displayed, each line representing the sum of the particular piece and the pieces below it. The total load is the value of the line labeled `write` since it is the sum of all the components. The communication components, send, receive and mix, all increase as the number of participants increases. The mix component increases the fastest, accounting for approximately 70% of the CPU load with 8 participants.

Figure 5 shows the predicted load for an audioconference with silence deletion and an increasing number of participants. Again, the audioconference components are all displayed. There is no mix component since we assume only one participant is speaking at a time. The communication components are only approximately 2% of the total load for all audioconferencing. The read and write from the audio device account for about 50% of the audioconference load, which seems disproportionately high compared to the communication components. The silence deletion component is also large, accounting for just under 50% of the audioconference load. Section 6.3.4 investigates the effects of reducing this component through software optimization or DSP hardware.

Figure 6 shows the total loads with silence deletion and the total loads without silence deletion. Here and in all subsequent graphs, only the total loads are displayed. For three or more participants, silence deletion reduces CPU load compared to not using silence deletion.

6.2 Increasing silence

What happens when the conversation has more silence in it? The amount of silence will vary, as conversation characteristics such as speakers and topics change. The percentage of bytes in packets after deletion has an inconsequential effect on CPU load because most cost is per-packet [Bhar92], but the percentage of packets after deletion may have a greater effect.

Figure 7 shows the result of our extrapolations to an increasing amount of silence. For a two person audioconference, the CPU load without silence deletion is always less than the load with silence deletion. Thus, for a two

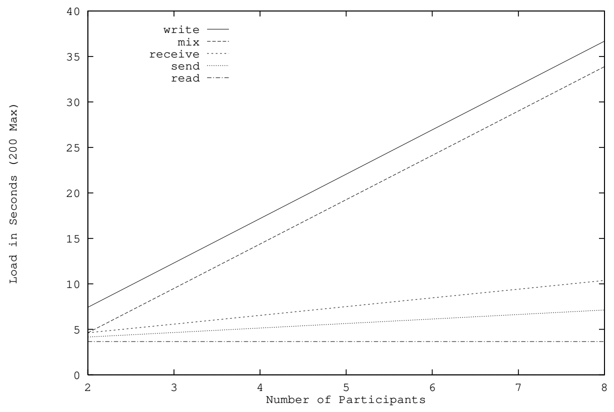

FIGURE 4. The IPX CPU load without silence deletion. The graph reads from the bottom. Each piece is the sum of the pieces below it. Thus, the total load is indicated by the write piece at the top. The maximum load is 200 seconds.

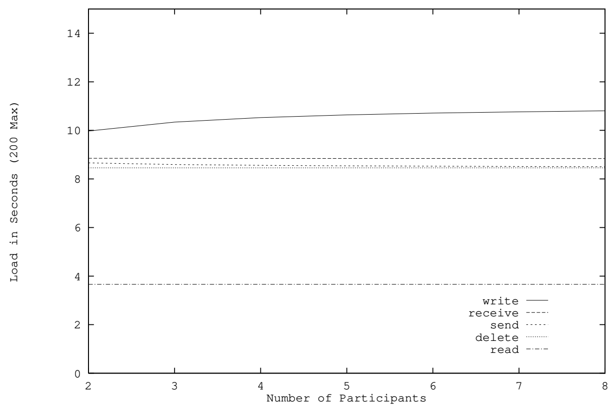

FIGURE 5. The IPX CPU load with silence deletion. The graph reads from the bottom. Each piece is the sum of the pieces below it. Thus, the total load is indicated by the write piece at the top. Since only one person sends packets at a time, there is no mix component. The maximum load is 200 seconds.

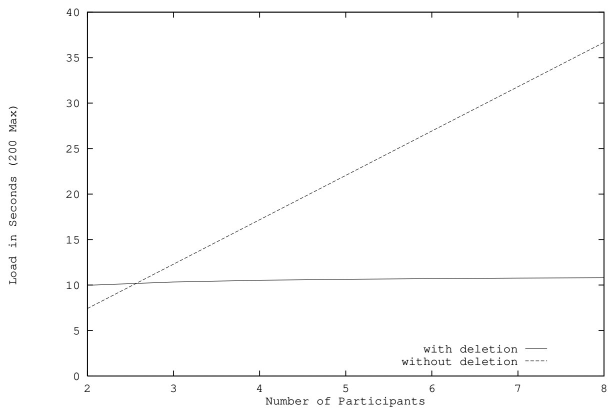

FIGURE 6. Comparison of CPU loads with and without silence deletion. Here, only the total load is indicated. The maximum load is 200 seconds.

person audioconference, there is never a CPU benefit from

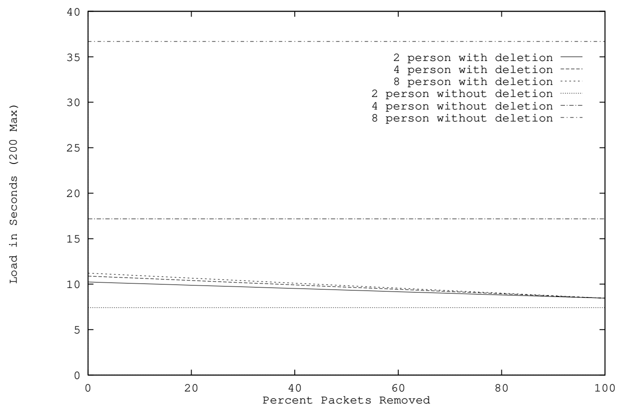

FIGURE 7. Affect of percentage of packets removed on conversations with silence deletion for 2, 4, and 8 participants. For comparison, conversations without silence deletion are plotted (horizontal lines).

using silence deletion, even when 100% of the packets are removed. This is because it takes less CPU time to send a packet once than it does to remove silence from it. For 4 and 8 person audioconferences, the load without silence deletion is always greater than the load with silence deletion. Thus, for audioconferences of 3 or more, any silence deletion results in a reduction in CPU load over no silence deletion. The largest part of this effect is that there is only one person speaking at a time, and even basic silence deletion algorithms delete all of the speech of silent participants.

6.3 Potential audioconference improvements

We adjust the components of our model to compare the performance benefits of four potential ways technology could be used to improve audioconference performance: faster CPU, faster communication, better compression, and Digital Signal Processing (DSP) hardware.

6.3.1 Faster CPU

What happens when the processor becomes faster? Faster processors decrease the per-byte times and the per-packet times of all components. Figure 8 shows the effects of faster processors for a 6-person audioconference. Notice that even the conversations without silence deletion use less than 5% of the CPU with an 8 times faster processor. With a fast enough CPU, the load from audioconferences may be insignificant with or without silence deletion. Similar results will hold for 4 to 8 participants.

6.3.2 Faster communication

What happens when the network protocols become more efficient? A more efficient network protocol decreases the amount of CPU time required to send and receive the sound packets. Figure 9 shows the predicted effect of an 8 times faster network protocol. Since the CPU load of the network protocol is so slight, the audio-

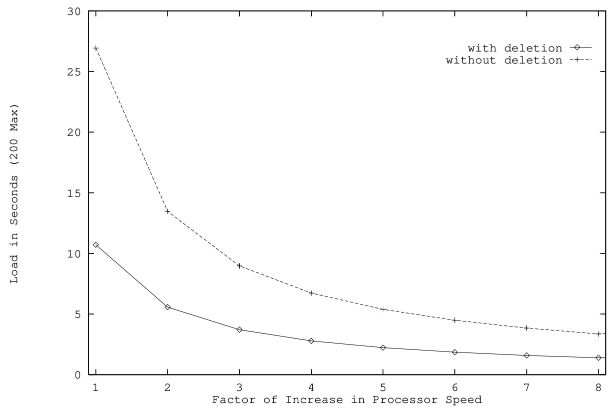

FIGURE 8. The effects of faster processors on audioconferences with and without silence deletion. The number of participants is fixed at 6. Maximum load is 200 seconds.

conference CPU load does not improve much despite a significant decrease in network protocol load.

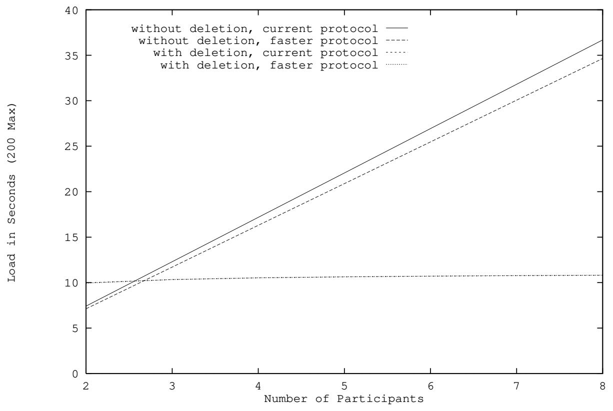

FIGURE 9. Audioconference CPU load with and without silence deletion with an 8 times faster network protocol. The CPU loads under the current network protocol is shown for comparison. With silence deletion, the two lines overlap. Maximum load is 200 seconds.

What happens when multicasting is used in place of unicasting? With multicasting, only one send is required for each sound packet regardless of the number of participants. Thus, the send component decrease by a factor of N-1. However, the receive, mix and write components remain unchanged. Figure 10 shows the predicted effect of multicasting on audioconferences with and without silence deletion. Since the CPU load for sending packets is small compared to the mix and write components, multicast routing reduces CPU load only slightly.

6.3.3 Better compression

How do forms of compression other than silence deletion affect audioconference CPU loads? Compression reduces the packet size but not the number of packets. The silence deletion algorithms we used are fairly simple. We

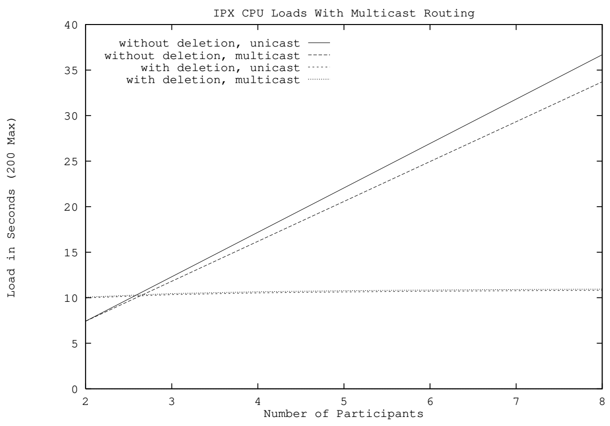

FIGURE 10. Audioconference CPU load with and without silence deletion for multicasting. The loads under unicasting are shown for comparison. With silence deletion, the two lines overlap. Maximum load is 200 seconds.

assume they are a lower bound on the CPU complexity of most compression and decompression algorithms. We estimate the CPU load of compression from the CPU load of silence deletion and use a range of values for the amount audio is compressed. Audioconference loads with other forms of compression will have full-sized packets for reading and compressing, and reduced packets for sending, receiving, mixing and writing. In addition, they will have an additional uncompressing component.

Figure 11 shows the predicted effects of other forms of compression. Only algorithms that compress sound bytes by 70% or more perform better than silence deletion for more than 4 people. It appears unlikely that compression algorithms better than this can be expected to run in real-time. Audioconference CPU load using compression scales worse than audioconference CPU load using silence deletion in every case.

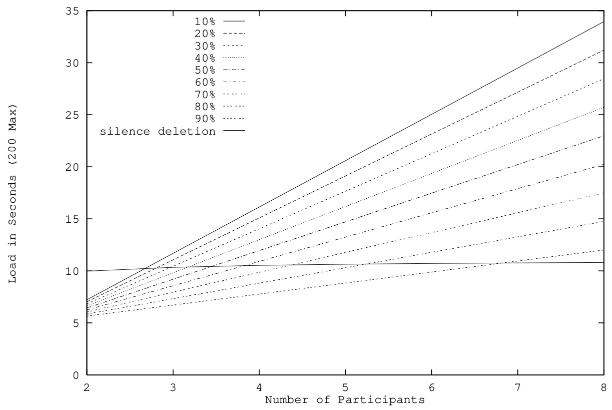

FIGURE 11. The effects of other forms of compression on audioconference CPU load. For comparison, CPU load with silence deletion is displayed. The maximum load is 200 seconds.

6.3.4 Digital signal processing hardware

In Section 6.1, we observe that the silence deletion component accounts for almost half the CPU load. A DSP chip might completely remove the load of silence deletion from the CPU. Such hardware silence deletion could be available at the user level through memory mapping, performing silence deletion at near-zero CPU cost. Or, the chip could be in the kernel, removing silence before the user makes the read call, reducing the number of bytes in each read call. Figure 12 shows the predicted effects of the two hardware silence deletion designs. Hardware silence deletion always reduces CPU load compared to no silence deletion. In addition, hardware silence deletion halves CPU load compared to software silence deletion. There is no significant difference between the two hardware deletion implementations.

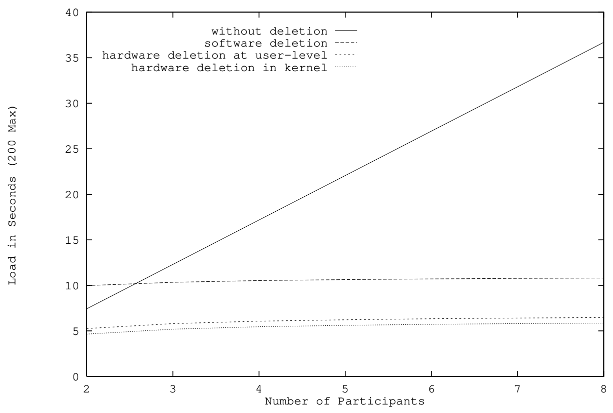

FIGURE 12. Audioconference CPU load with silence deletion done in hardware, compared with software and without silence deletion. The two different hardware deletion designs, having a zero-cost user level call, or having the hardware in the kernel, are presented. The maximum load is 200 seconds.

What happens when we have both compression and silence deletion in hardware? DSP chips could completely remove the load of silence deletion and compression from the CPU. Since we observed that a zero-cost user level call to the DSP or the DSP in the kernel perform similarly (Figure 12), we only present data for the DSP in the kernel. Figure 13 shows the CPU load for hardware silence deletion and compression. Having both silence deletion and compression in hardware decreases total CPU load by 60%.

In Section 6.1, we note that in conversations without silence deletion the mixing component accounts for over 70% of the CPU load. Hardware mixing can be done with a DSP chip or on a multi-channel audio-board. When mixing is done in hardware, the CPU does not have to mix. The CPU writes all incoming sound packets to the audio device, so the load of writing then scales with the number of participants. Figure 14 shows the predicted effects of hardware mixing. Hardware mixing significantly reduces

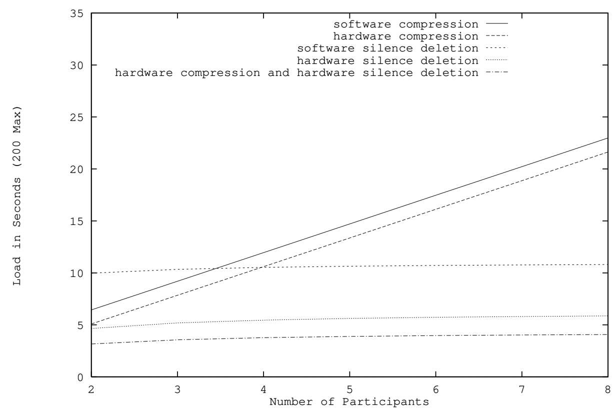

FIGURE 13. Audioconference CPU load with hardware silence deletion and hardware compression. For comparison, software and hardware silence deletion and software and hardware compression are shown. All hardware implementations have the DSP chip in the kernel. Maximum load is 200 seconds.

non-silence deletion audioconference CPU loads. However, such loads still increase rapidly with the number of participants, becoming larger than the loads under silence deletion at four or more people.

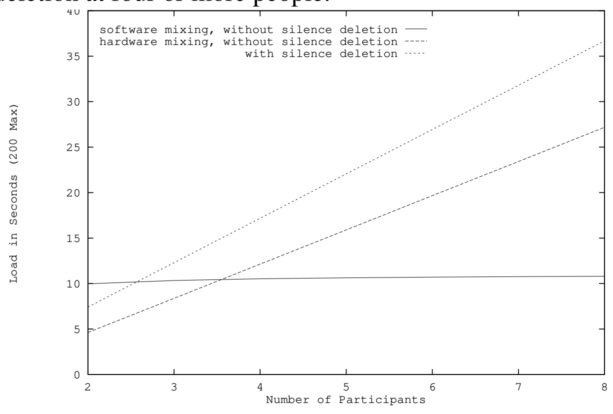

FIGURE 14. Audioconference CPU load with packet combining done in hardware compared with packet combining done in software. The silence deletion conversations are unaffected by hardware mixing. Maximum load is 200 seconds.

We observed in the micro experiments (section 4) that the audio component appeared comparatively large. Measurement of three classes of Suns' audio devices show that the audio device efficiency is improving disproportionately to CPU speed (Figure 15). Perhaps the audio device will eventually be made to read and write as efficiently as sending and receiving packets (a 10 fold increase). Figure 16 shows the CPU load for an 8 times faster audio device. For audioconferences with silence deletion, a faster audio device can reduce total Sun CPU load by almost one-half.

6.3.5 Improved silence deletion algorithms

The above experiments and equations all used cases with a deletion algorithm that removed only 15% of the

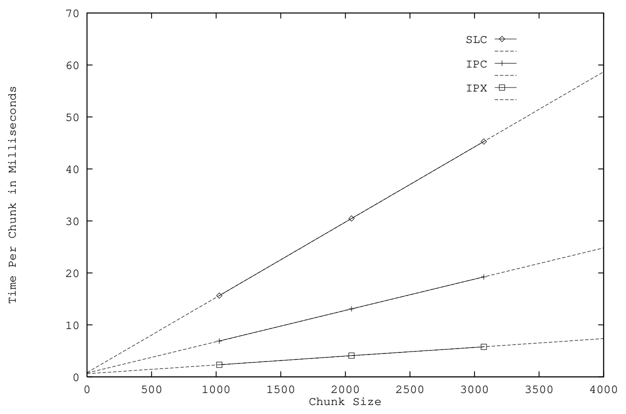

FIGURE 15. Read times per byte for different classes of machines. We would expect the slopes to scale according to the 20-25-40 Mhz clock speeds but they actually scale in a 20-48-171 ratio. The improved performance on the IPC and IPX may be due to better audio hardware. The dashed lines represent extrapolations beyond the measured data.

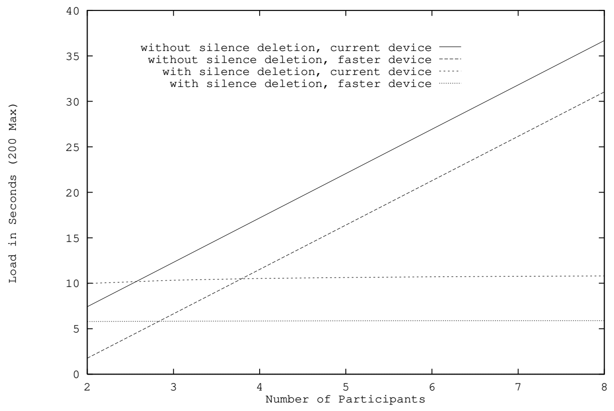

FIGURE 16. Effects of an 8 times faster audio device on CPU loads. For comparison, loads under the current device are shown. Maximum load is 200 seconds.

packets. Our pilot tests suggest that, theoretically, deletion algorithms could remove as much as 60% of speech packets. Figure 17 shows the predicted effects of increasing the amount of silence deleted from the speaker's speech. Improving the deletion algorithm to remove as much as 90% of the packets improves the CPU load by only approximately 10%. The biggest reduction in load is in removing the silence of the non-speakers; doing better than that improves CPU load only a little.

7 Conclusions

Our goal was to identify improvements that reduce audioconference CPU load. We developed a model for audioconference CPU load and presented experimentally-based analysis on the effects of improving each component of the model.

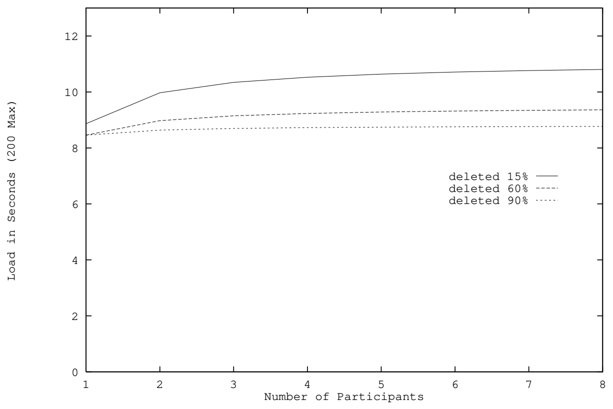

FIGURE 17. Audioconference CPU load with different amounts of silence deleted. The 15% is based on a fairly poor algorithm. 60% is the maximum observed during our pilot tests. We consider 90% deletion to be a conservative upper bound on silence deletion algorithms. The maximum load is 200 seconds.

Based on our analysis of individual components, silence deletion appears to improve the scalability of audio more than all other improvements. This result holds even when using compression and even for ten times faster processors, networks and Digital Signal Processing (DSP) hardware. Further, the simple silence deletion algorithms we studied achieve 90% of the potential benefit from silence deletion for audioconference CPU load. Simple silence deletion algorithms are sufficient because the biggest reduction in CPU load from silence deletion is in removing the silence of the non-speakers which even the basic algorithms do well.

Sending and receiving packets are already relatively low-cost, so reducing the cost further has relatively low benefit.With or without silence deletion, better network protocols, faster networks and even multicasting reduce load only slightly. Techniques based on DSP hardware alone do not scale as well as silence deletion alone. However, DSP based silence deletion and compression together scale better than any other technique. Silence deletion done in a DSP chip can reduce CPU load by 50%. Compression and silence deletion in a DSP chip can reduce CPU load by an additional 30%.

There are important issues still to explore in the CPU performance of multimedia on packet-switched networks, including uses of audio other than audioconferences, total CPU load of workstations and routers over wide-area networks and CPU load of videoconferences.

8 References

[Bhar91] *Communication in the Raid Distributed Database System*, Bharat Bhargava, Enrique Mafla, and John Riedl, International Journal on Computers and ISDN Systems, 21(1991).

[Cabr88] *User-Process Communication Performance in Networks of Computers*, Luis-Filipe Cabrera, Edward Hunter, Michael J. Karels, and David A. Mosher, IEEE Transactions on Software Engineering, Vol. 14, No. 1, January 1988.

[Casn92] *First IETF Internet Audiocast*, S. Casner, S. Deering, ACM SIGCOMM, Computer Communication Review, pages 92-97, 1992.

[Cave90] *A Visual Design for Collaborative Work: Columns for Commenting and Annotation*, Todd Cavalier, Ravinder Chandhok, James Morris, David Kaufer, and Chris Neuwirth, Proceedings of HICSS, IEEE 1990.

[Clay93] *Silence is Golden? - The Effects of Silence Deletion on the CPU Load of an Audioconference*, Mark Claypool and John Riedl, University of Minnesota Department of Computer Science TR, July 1993.

[Gons83] *Packet-Voice Communication on an Ethernet Local Computer Network: an Experimental Study*, Timothy A Gonsalves, ACM 1983.

[Jeff92] *Kernel support for live digital audio and video*, K. Jeffay, D.L. Stone and F.D. Smith, Computer Communications, Vol. 15, No. 6, pages 388-95, July/August 1992.

[Kay93] *Measurement, Analysis, and Improvement of UDP/IP Throughput for the DECstation 5000*, Jonathan Kay and Joseph Pasquale, University of California, San Diego, Department of Computer Science TR 1993.

[Kroo92] *A High-Quality Multirate Real-Time CELP Coder*, Peter Kroon and Kumar Swaminathan, IEEE Journal on Selected Areas in Communications, Vol. 10, No. 5, June 1992.

[Lazo93] *File Access Performance of Diskless Workstations*, Edward D. Lazowska, John Zahorjan, David R. Cheriton and Willy Zwaenepoel, ACM Transactions on Computer Systems, Vol 4, No. 3, p. 238-268, August 1986.

[Mash93] *Distributed Collaborative Software Inspection*, Vahid Mashayekhi, Janet Drake, Wei-Tek Tsai, and John Riedl, IEEE Software, p. 66-75, September 1993.

[Pan93] *Digital Audio Compression*, Davis Yen Pan, Digital Technical Journal, Vol. 5, No. 2, p. 28-40, Spring 1993.

[Rabi75] *An Algorithm for Determining the Endpoints of Isolated Utterances*, L. R. Rabiner and M. R. Sambur, The Bell System Technical Journal February 1975.

[Ried93] *SuiteSound: A System for Distributed Collaborative Multimedia*, John Riedl, Vahid Mashayekhi, James Schnepf, Mark Claypool, and Dan Frankowski, IEEE Transactions on Knowledge and Data Engineering, August 1993.

[Rowe92] *A Continuous Media Player*, Lawrence A. Rowe and Brian C. Smith, Proc. 3rd Int. Workshop on Network and OS Support for Digital Audio and Video, November 1992.

[Schu92] *Voice Communication Across the Internet: A Network Voice Terminal*, Henning Schulzrinne, University of Massachusetts Department of Electrical Engineering, August 6, 1992. Nevot is available via anonymous ftp.

[Tere91] *Experiences with Audio Conferencing Using the X Window System, UNIX, and TCP/IP*, R. Terek and J. Pasquale, USENIX, Summer 1991, pages 405-417.

[Thom85] *Diamond: A Multimedia Message System Built on a Distributed Architecture,* Robert H. Thomas, Harry C. Frisked, Terrence R. Crowley, Richard W. Schaaf, Raymond S. Tomlinson, Virginia M. Travers and George G. Robertson, IEEE Computer, December 1985.