WPI-CS-TR-98-19 July 1998

The Effects of High-Performance Processors, Real-Time Priorities and High-Speed Networks on Jitter in a Multimedia Stream

by

Mark Claypool
Joe Habermann and John Riedl

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

The Effects of High-Performance Processors, Real-Time Priorities and High-Speed Networks on Jitter in a Multimedia Stream

Mark Claypool
claypool@cs.wpi.edu
Worcester Polytechnic Institute
Computer Science Department

Joe Habermann
John Riedl
{*habermann,riedl*}*@cs.umn.edu*
University of Minnesota
Computer Science Department

August 12, 1998

Contents

Abstract

Multimedia applications have the potential to enhance work for teams of users collaborating across distances. Jitter hampers the effectiveness of these multimedia applications. Jitter is the variation in the end-to-end delay of data sent from one user to another. Jitter can cause silent gaps in the playout of an audio stream such as in an audioconference, or a choppy appearance to a video stream for a videoconference. We experimentally measure the effects of three jitter reduction techniques: high-performance processors, real-time priorities and high-speed networks. We incorporate our jitter measurements into a general model for multimedia application quality. Our model allows us to explore how advances in networks and processors will improve application quality compared with real-time priorities. As an example, we apply our model to a videoconference. We find high-performance processors, real-time priorities and high-speed networks all significantly reduce jitter under conditions of heavy processor and network load. For the next five years, processor and network improvements alone will not reduce jitter enough to eliminate the need for application buffering techniques. However, for multimedia on a LAN, real-time priorities can reduce jitter enough to eliminate the need for application buffering today. On a WAN, especially the Internet, real-time priorities may not be available on all routers, reducing the effectiveness of real-time priorities in reducing jitter. In this case, buffering techniques may still be needed.

1 Introduction

There are many exciting new distributed multimedia applications. Today, two to tens of users can communicate through a computer audioconference. Tomorrow, tens to hundreds of neuroscientists will explore and contribute to a distributed brain database [5]. Soon, tens, hundreds and perhaps even thousands of soldiers will train for combat in a distributed interactive simulation [10]. In particular, there is an increasing interest in the use of packet-switched networks for performing computer-based multimedia teleconferencing:

- Teleconferencing saves the time and trouble of traveling to collaborate.
- Multimedia is needed because people communicate best when they can draw pictures and use voice inflections and body language rather than simply type text.
- General-purpose workstations can have advantages over the use of specialized hardware: corporate and academic environments have ready-access to necessary hardware; and teleconferencing can be enhanced when computers are used through the use of record/playback, on-screen speaker identification, floor control and subgroup communication.
- Audio and video streams similar to those in a teleconference are often integrated into larger distributed multimedia applications. For example, a shared editor may allow several users to simultaneously collaborate on a document from separate workstations. Audio and video links coupled with the shared editor enhance the editing process by making it more like face-to-face collaboration.
- Most existing workstations are connected by packet-switched networks.

There are many examples of multimedia applications that are driving this interest in packet-switched multimedia teleconferencing:

Distributed Interactive Simulation. Distributed Interactive Simulation (DIS) applications are designed to enable soldiers to engage in simulated combat [10]. The DIS protocol allows participation from soldiers at military bases across the country using current packet-switched networks, saving the time and trouble of traveling for combat training. Many DIS developers are designing simulators that use off-the-shelf general purpose hardware [38]. In order for the combat to be realistic, the simulators use high-quality graphics and allow communication among the soldiers with audio and video. With the high multimedia system requirements and many users, applications such as DIS will stress all parts of a computer system.

Scientific Visualization. Neuroscientists from diverse disciplines plan to collaborate across distances in exploring various aspects of brain structure [5]. Their design includes a zoomable multimedia database of images of the brain tissue. High-resolution magnetic resonance imaging (MRI) show the entire brain in a single dataset. Even higher resolution confocal microscope images are anchored to these MR images in three dimensions. The user starts a typical investigation by navigating through the MR images in a coarse 3D model of the brain to a site of interest. The user then zooms to higher resolution confocal images embedded in the MRI landscape. This real-time navigating and zooming is called "flying." In order to be an effective collaboration tool, flying must provide high-resolution images and a high-frame rate as well as high-quality audio and video to allow neuroscientists to communicate.

Audioconferences. Audioconferences have been shown to enhance collaborative work among users on distributed workstations [44, 36, 46]. 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.

In order to build systems that will adequately support such applications, it is important to predict application performance as the number of users increases and evaluate performance and cost tradeoffs for different system designs. One indication of the performance of an entire computer system is the user's opinion on the *multimedia quality* of the applications they run. Multimedia quality is a measure of how closely the performance of a multimedia application meets the requirements expected by the user. If the user performance requirements are met, application quality will be acceptable. If the user performance requirements are not met, application quality will be unacceptable. We are developing a quality planning model to aid in designing systems that meet users' quality requirements for multimedia applications in the future.

Although we often think of a multimedia application as a continuous stream of data, computer systems handle multimedia in discrete events. An event may be receiving an update packet or displaying a rendered frame on the screen. The quantity and timing of these events give us measures that affect application quality. We have identified three measures that determine quality for most distributed multimedia applications:

- *Latency.* The time it takes information to move from the server through the client to the user we call *latency.* Latency decreases the effectiveness of applications by making them less like real-life interaction [53, 22, 45, 13].
- *Data Loss.* Any data less than the amount determined by the user requirements we call *data loss.* Data loss takes many forms such as reduced bits of color, jumbo pixels, smaller images, dropped frames and lossy compression [2, 40, 37, 49]. Data loss may be done voluntarily by either the client or the server in order to reduce load or to reduce jitter and/or latency.
- *Jitter.* Distributed applications usually run on non-dedicated systems. The underlying networks are often packet-switched and the workstations are often running multiple processes. These non-dedicated systems cause variance in the latency, which we call *jitter.* Jitter can cause gaps in the playout of a stream such as in an audioconference, or a choppy appearance to a video display [27, 43, 28].

The effects of latency on a user's perception of an application is well-understood and well-researched [53, 22, 45, 13]. Similarly, there is a clear relationship between data loss and application quality deterioration [2, 40, 37, 49]. Methods to ameliorate the effects of jitter have been explored by many researchers [48, 43, 16]. The tradeoff between buffering and jitter has also been explored [31, 47]. However, to the best of our knowledge, the effects of high-performance workstations, real-time priorities and high-speed networks on jitter has not been thoroughly investigated. Moreover, the effects of jitter on the overall quality of a multimedia application has not been established.

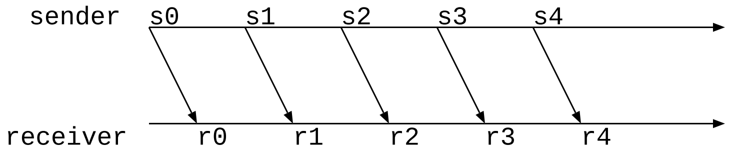

Figure 1: A Jitter-Free Stream. The above figure is a model of a jitter-free stream. Each si is the time at which the send process initiates the transmission of frame i. Each ri is the time at which the receiving process plays frame i.

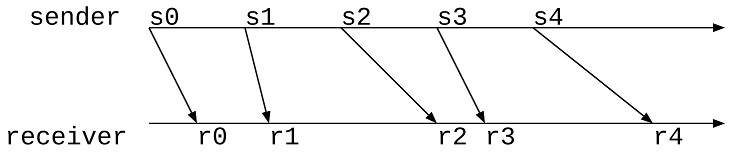

Figure 2: A Stream with Jitter. The above figure is a model of a stream with jitter. Each si is the time at which the send process initiates the transmission of frame i. Each ri is the time at which the receiving process plays frame i.

1.1 Contributions

The major contributions of this paper are:

- A comparison of measures of jitter that have been used by jitter researchers.
- An experimentally-based study of the effects of high-performance processors, real-time priorities and high-speed networks on jitter.
- A multimedia application quality model that enables the prediction of application performance and evaluation of system design tradeoffs.
- Performance predictions for videoconferences with high-performance processors and networks, multiple users and increasing system load.
- Predictions on the importance of application jitter reduction techniques in the future.

1.2 Jitter

In a distributed multimedia application, a multimedia stream is generated at the sending workstation and sent over a network to the receiving workstation. The data is generated and sent in fixed-sized quantities called frames. The end-to-end frame delay is the time between a frame's generation on the sender and the time it is processed by the receiver. Variation in this end-to-end delay we call *jitter*. We have observed jitter on the order of a few hundred milliseconds when sending a multimedia stream using unloaded workstations on a quiet Ethernet local area network.

How does jitter affect a multimedia stream? In the absence of jitter, the frames can be played as they are received, resulting in a smooth playout, depicted by Figure 1. However, in the presence of jitter, interarrival times will vary, as depicted in Figure 2. In Figure 2 the third frame arrives late at r2. In the case of audio speech, the listener would experience an annoying pause during this period. In the case of video, the viewer would see the frozen image of the most recently delivered frame.

Human ears and eyes can smooth over occasional glitches in audio and video streams, so typically some latency, jitter and data loss is acceptable. The amount of latency, jitter and data loss will determine the application quality. The more, the worse the quality. The less, the better the quality. The acceptable amount of latency, jitter and data loss varies from application to application. In Section 8, we study the acceptable amount of latency, jitter and data loss for a videoconference application.

Level	Possible Jitter Reducing Technique
Application	Application tuning
	Buffering
System	Disk layout/scheduling
	Alternate network protocols
	Operating system priorities
Hardware	High-performance processors
	High-speed networks
	Fast disks

Table 1: Hierarchy of Possible Jitter Reducing Techniques

1.3 Potential Jitter Reduction

To summarize, jitter is one of the fundamental components for multimedia application quality. Jitter can cause gaps in the playout of an audio or video stream. Therefore, if we decrease jitter, we will have a less choppy playout and better application quality. In seeking to decrease jitter, we can tune the application, the operating system or the underlying hardware. This gives us a hierarchy of possible jitter reducing techniques. Table 1 depicts this hierarchy. Application techniques for reducing jitter include application tuning and buffering. System techniques for reducing jitter include disk layout/scheduling, alternate network protocols and operating system priorities. Hardware techniques for reducing jitter include hardware improvements, such as high-performance processors, high-speed networks and fast disks. We consider the merits of conducting research on each jitter reducing technique:

Application Tuning and Buffering. Application-level techniques for reducing jitter are done in the context of compensating for the jitter produced by the underlying computer system. There has been a lot of research done on application techniques [15, 46, 35]. Controlled frame playout through buffering, in particular, has been well studied [31, 43, 50, 48, 16]. With buffering, transmitted frames are held in memory by the receiver for a period of time. Then, the receiver plays out each frame with a constant latency, achieving a steady stream. If the buffer is made sufficiently large so that it can hold all arriving data for a period of time as long as the tardiest frame, then the user receives a complete, steady stream. However, the added latency of data can be disturbing [41]. So there is a tradeoff between ameliorating the effects of jitter and minimizing the amount of latency due to buffering.

Computer systems continue to get faster while human perceptions remain the same. Future system improvements may remove enough of the underlying jitter such that application-level jitter reduction techniques are unnecessary. How far in the future will this be? In this work, we seek to answer this question by experimentally measuring the effects of system improvements on jitter. Then, based on the rate of hardware improvements, we predict when the jitter levels contributed by the underlying system will drop below human perception.

Disk Layout/Scheduling and Fast Disks. Fast disks, and disk layout/scheduling strategies may be crucial for reducing jitter for some multimedia applications. However, in a "live" teleconference, data is generated and consumed in real-time, so in this work, we do not consider the effects of disk layout or scheduling on jitter.

Alternate Network Protocols. Alternate network protocols may reduce jitter over traditional network protocols. There has been a lot of research into network protocols designed for handling multimedia data

[33, 51, 45]. Our work experimentally measures the effects on jitter of two high-speed network protocols: Fibre Channel and HIPPI.

Operating System Priorities. There is evidence to suggest that a significant source of jitter in the transmission of multimedia may be found in the operating system of the sending and receiving workstations [18]. The principal deficiency in process scheduling in modern operating system is that user-level processes are not preempted while in kernel mode. As a result, increased system activity can increase response time without bound. In addition, modern operating systems allow priority inversions to occur whereby a low priority process excludes high-priority processes from accessing time-critical data, thus causing the high-priority processes to miss deadlines. These deficiencies can result in latency on the order of 100 milliseconds [30]. Real-time scheduling allows the operating system to accurately time events and allocation of memory to the process and provide for priority scheduling. This feature can reduce latency to the order of a few milliseconds [30]. Our work experimentally compares the benefits of real-time scheduling to normal scheduling to determine if real-time scheduling does reduce jitter.

High-Performance Processors and High-Speed Networks. High-performance processors have higher throughput and a faster context switch time than typical processors resulting in better application response time. High-speed networks deliver frames from the sender to the receiver faster than typical networks, reducing the network transmission time. Together, high-performance processors and high-speed networks networks will reduce application latency. The reduced latency should be accompanied by a reduction in latency variation, or jitter. We run experiments on SGI Challenge workstation clusters and Fibre Channel and HIPPI networks under both light and heavy load to determine the effects that hardware improvements have on jitter.

1.4 Hypotheses

Given the above discussion, we make the following hypotheses:

1. *High-performance processors reduce jitter.*
2. *Real-time operating system priorities reduce jitter.*
3. *High-speed networks reduce jitter.*

Processor performance approximately doubles every year [20]. As processor performance increases, latency decreases. This decrease in latency may be accompanied by a decrease in jitter. However, in order to significantly improve application quality, any jitter reduction from high-performance processors needs to be large compared to jitter contributed by the network and operating system.

Since real-time operating system priorities have been shown to decrease latency, it seems natural to assume that they may decrease jitter, also. If real-time priorities do prove to significantly reduce jitter, application quality may be improved without expensive hardware upgrades or time-consuming application tuning.

Under heavy loads, high-speed networks should deliver multimedia frames faster than traditional networks. We would also expect high-speed networks to decrease jitter under such conditions. However under light loads, the reduced latency from high-speed networks may not significantly improve application quality.

In order to test our hypotheses, we looked at three possible performance evaluation techniques: analytic modeling, simulation and experimental measurement. Frankowski and Riedl found that analytically modeling jitter is very difficult, even on a quiet, single-hop network [17]. The contributions to jitter from the operating systems on both the sender and receiver and the contribution to jitter from the network are difficult to capture mathematically. Stein and Riedl had some success in using simulation to evaluate the effects of jitter on audioconference quality [47]. However, according to Jain, simulations are most effective when they are based on previous measurement [23]. Unfortunately, to the best of our knowledge, careful measurements of the contributions to jitter from high-performance processors, real-time priorities and high-speed networks have not been made. Simulating the effects of such components on jitter may give rise to inaccurate or misleading results. We use experimental measurement to test our hypotheses about the effects of high-performance processors, priorities and high-speed networks on jitter reduction. Future research may be able to use our performance measurements as a basis for simulations.

Although our hypotheses are simply stated, the answers to these hypotheses are not quite as simple as "true" or "false." The rest of this paper explains why. Section 2 lists related work. Section 3 details the experimental design components that were common to all experiments. Section 4 explores the effects of processor performance on jitter. Section 5 examines the effects of real-time priorities on jitter. Section 6 looks at the effects of high-speed networks on jitter. Section 7 describes how jitter fits into our a model for application quality. Section 8 applies our quality model to a videoconference. Section 9 summarizes our conclusions and Section 10 lists possible future work.

2 Related Work

2.1 Teleconferencing Systems

Several experimental teleconferencing systems have been designed to explore teleconferencing performance issues such as jitter.

Riedl, Mashayeki, Schnepf, Claypool and Frankowski developed SuiteSound [44]. SuiteSound attempted to integrate support for multimedia into the Suite programming environment. They performed experiments to determine the network and CPU load of the SuiteSound tools, including the effects of an algorithm that removes silence from digitized speech.

Hopper developed the Pandora system to investigate the potential for creating and deploying a desktop multimedia environment based on advanced digital video and audio technology [21]. Pandora offered a "tool box" to those analyzing and working on information by providing a flexible communications system that allowed effective interaction between a number of users.

Jeffay, Stone and Smith developed a transport protocol that supports real-time communication of audio/video frames across campus-area packet switched networks [26]. They demonstrated the effectiveness of their protocol by measuring the performance of their protocol when transmitting audio and video across congested networks.

Teleconferencing systems attempt to provide quality audio and video to groups of users. One component to teleconferencing quality is jitter. We provide experimental results on the effects that system improvements will have on reducing jitter. We also present a quality model that can be used to evaluate system configuration tradeoffs in predicting teleconferencing quality from the users' perspective.

2.2 Delay Buffering

Research in delay buffering has looked at ways to ameliorate the effects of jitter by controlling the playout and delivery of frames at either the sender or the receiver.

Ramjee, Kurose, Towsley and Schulzrinne compared the effects of four different buffering algorithms for adaptively adjusting the playout delay of audio packets over a wide area network [43]. They found that an adaptive algorithm which explicitly adjusts to the sharp, spike-like increases in packet delay achieved the lowest rate of lost packets.

Stone and Jeffay presented an empirical study of several buffering policies for managing the effect of jitter on the playout of audio and video in computer-based conferences [48]. They evaluated a particular policy called queue monitoring by comparing it with two policies from other literature. They showed that queue monitoring performs as well or better than the other policies over the range of observed network loads.

As opposed to the two papers above that use receiver-side buffering, Ferrari presented a scheme for buffering data at the network nodes between the sender and receiver [16]. He studied the feasibility of bounding jitter in packet-switched wide area networks with a general topology. He presented a buffering scheme that is capable of providing a significant reduction in jitter, with no accumulation of jitter along the path of a channel, and demonstrated that jitter control significantly reduces the buffer space required in the network.

Talley and Jeffay presented a method of buffering at the side of the sender, instead of the receiver [50]. They presented a framework for transmission control that describes the current network environment as a set of sustainable bit and packet transmission-rate combinations. They empirically demonstrated the validity of adapting both packet and bit-rate using simple adaptation heuristics.

Naylor and Kleinrock developed a model for measuring the quality of an audioconference based on the amount of dropped frames and client-side buffering [31]. They used their model to investigate two adaptive receiver-side buffering schemes which may be used to achieve a smooth playout. Method E expands the buffer to preserve all incoming frames. Method I ignores all late frames in order to preserve timing.

We explore alternative means to delay buffering to reduce jitter. Specifically, we examine when hardware improvements might make delay buffering techniques unnecessary. We extend the work of Naylor and Kleinrock to develop a more general model for multimedia application quality.

2.3 Processor Performance

Dongarra compared the performance of different computer systems in solving dense systems of linear equations [14]. He compared the performance of approximately 100 computers, from a CRAY Y-MP to scientific workstations such as Apollos and Suns to IBM PC's.

SPEC, the Standard Performance Evaluation Corporation, has sought to create an objective series of applications-oriented tests, which can serve as common reference points and be considered during the evaluation process [11]. The benchmarks are primarily meant for comparing processor speeds. The SPEC benchmark numbers are the ratio of the time to run the benchmarks on a reference system and the system being tested.

In our past work, we explored audioconference processor load, focusing on the effects of silence deletion [8]. We found silence deletion improves audioconference CPU load more than most alternatives, including 10 times faster processors and multicasting.

Also in our past work we found a strong inverse correlation between SPEC results and the execution times for fundamental application components, such as the time to send a packet. We use SPEC results to make predictions to processors other than the ones on which we perform experiments. Performance results from our research may also be useful to other benchmark researchers.

2.4 Real-time Performance

Govindan and Anderson conjectured that the traditional operating system goals of fairness, maximum system throughput and fast interactive response may conflict with the needs of real-time applications such as continuous media and proposed a new processor scheduling algorithm [18]. Jeffay, Stone and Smith made similar remarks and provided experimental evidence [26]. Jeffay and Stone and went so far as to build an operating system that supports real-time multimedia [24]. Khanna, Serbree and Zonowsky discussed the design, implementation and performance of the SunOS 5.0 operating system as a real-time system [30].

We experimentally measure the effects of Solaris real-time priorities and the Unix nice facility on jitter. We compare the effects of using operating system priorities with processes run under default priorities.

2.5 Network Performance

Boggs, Mogul and Kent discussed Ethernet performance [4]. Based on measurements, they show that for a wide class of applications, Ethernet is capable of carrying its nominal bandwidth of useful traffic and allocates bandwidth fairly.

Lin, Hsieh, Du, Thomas and MacDonald studied the performance characteristics of several types of workstations running on a local Asynchronous Transfer Mode (ATM) network [34]. They measured the throughput of four different application programming interfaces (API). They found the native API achieved the highest throughput, while TCP/IP delivered considerably less.

Lin, Hsieh, Du and MacDonald studied the performance of a Fibre Channel network [33]. The proposed approaches for improving the maximum achievable bandwidth and reducing end-to-end communication latency.

We experimentally measure the amount of jitter contributed by a traditional Ethernet and compare it with jitter contributed by two high-speed networks, an Fibre Channel and a HIPPI. We study jitter contributions under both light and heavy network load conditions.

3 Shared Experimental Design

We ran a series of experiments to test our hypotheses. This section details the design components that were common to all experiments.

Our experiments were all conducted on a single-hop LAN. In our first set of experiments, we attempted to measure jitter contributions on a WAN. However, getting tight confidence intervals on WAN jitter proved extremely difficult. Perhaps in the future, our LAN measurements may be used in simulations to explore the effects of a WAN on jitter.

Each experiment simulated the transmission of a multimedia stream under various conditions and measured the amount of jitter. We used pairs of user-level processes that sent and received UDP datagrams using Berkeley socket I/O. The `send` process used an interval timer to initiate frame delivery. The `receive` process took timestamps using the `gettimeofday()` system call. These timestamps are used to measure the amount of jitter.

A jitter measure must reflect the extent to which the interarrival times vary. There are several classical statistical measures of variation that have been used in jitter research:

- *Range.* The simplest measure of variation is the maximum delay between any two consecutive frames. Generally speaking, more variation is reflected in a larger range. The range also provides a maximum delay variance for providing jitter guarantees to multimedia applications. For this reason, range has been used in some past jitter research [52].
- *Variance.* The primary measure of variation determines the extent to which each frame deviates from the mean frame interarrival time. The standard way to prevent values below the mean interarrival time negate values above the mean interarrival time is to square them. Dividing by the number of frames gives the average squared deviation. This is the standard definition of variance and is used by several jitter researchers [16, 43]. The formula for variance is [12]:

$$Variance = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{(n-1)}$$

- *Standard Deviation.* By taking the square root of the variance, we get the same units as the frame delay. This is the standard deviation, the classical measure of spread. Several researchers have used the standard deviation of packet interarrival times as a measure of jitter [47, 28, 3]. The formula for standard deviation is [12]:

$$StandardDeviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{(n-1)}}$$

- *Absolute Deviation.* Variance and standard deviation have proven to be very sensitive to outliers. For this reason, some researchers have used absolute deviation as a measure of jitter because it is less sensitive to values far from the mean [46]. The formula for absolute deviation is :

$$Absdev = \sum_{i=1}^{n} |x_i - \overline{x}|$$

In addition, there has been jitter research that has incorporated derived measures of jitter:

- *Gaps.* When playing out a multimedia stream, a late frame will cause a gap in the smooth playout. The number of gaps per second can provide a measure of how much jitter there is in the stream. The number of gaps per second has been used by several jitter researchers [48, 17].
- *Jitter Compensation.* In addition to determining buffer size, the jitter compensation curve (see Figure 16) can be used to measure jitter. The more jitter in a multimedia stream, the larger the area under the jitter compensation curve. This jitter measure has been used by [19].

There are many more possible measures of jitter: interquartile range, mean length of gaps, median length and even second-order statistics such as variance on the mean length, etc. We do not consider these methods further since, to the best of our knowledge, other researchers have not used them to measure of jitter.

However, we did want to compare the equivalence of jitter measures that *had* been used in previous research. We recorded interarrival times for all experiments in Sections 4 through 6 and computed jitter values for range, variance, standard deviation, absolute deviation, gaps and jitter compensation. We then computed the Pearson's correlation coefficient [12] for each pair of jitter measures. For the *gaps* jitter measure, we assumed all late data is ignored and that there is a buffer of 250,000 microseconds, values used by other researchers [31]. Table 2 gives the correlation coefficient for each jitter measure pair.

Range does not correlate well with any of the other jitter measures. Range measures variation as the distance between the two most extreme values. Variation depends on more than just the extreme values as we can see from these two samples: $\{10,15,15,15,20\}$ and $\{10,10,15,20,20\}$ both have the same range but there is less dispersion in the first sample.

Gaps does not correlate well with Absolute Deviation and it correlates only slightly well with Area. The number of gaps per second depends upon the initial buffer chosen, so different buffer sizes might have different correlation results. The rest of the jitter measures correlate well (0.72) to to extremely well (0.99) with each other. With the exception of Gaps, the measure of jitter chosen does not matter in terms of representing the number and magnitude of late packets in a multimedia stream. We choose Variance as our measure of jitter because it is easy to understand and compute and has been used by [47, 28, 3] to measure jitter.

Audio and video have very different bandwidth needs. The Sun audio device records audio at a rate of 8000 bytes/second [1]. Video acquisition hardware typically generates 30 frames-per-second and compressed

	Range	Variance	Stddev	Absdev	Gaps	Area
Range	1.00	0.34	0.34	0.33	0.21	0.34
Variance	0.34	1.00	0.96	0.89	0.85	0.95
Stddev	0.34	0.96	1.00	0.95	0.73	0.99
Absdev	0.33	0.89	0.95	1.00	0.66	0.96
Gaps	0.21	0.85	0.73	0.66	1.00	0.72
Area	0.33	0.95	0.99	0.96	0.72	1.00

Table 2: Correlation Among Jitter Measures. This table depicts the correlation coefficients for 9 different jitter measures. Range is the maximum interarrival time. Variance is the variance in interarrival times. Stddev is the standard deviation of interarrival times. Absdev is the absolute deviation of interarrival times. Gaps are the number of playout gaps per second with 250,000 microseconds of buffering. Area is the area under the jitter compensation curve. The table is symmetric about the diagonal.

frames with a resolution of 256x240 require around 2 megabits-per-second of bandwidth [6]. We wanted to see if either packet size or packet rate changed the frequency and/or magnitude of the interarrival times.

Pairs of packets tend to reflect about the mean packet arrival time. If a packet arrives late, the next packet usually arrives early by the same time that the previous packet was late. When interarrival times become small, the packet following a late packet is unable to arrive an equal amount early, being unable to arrive in fewer than zero microseconds. Figure 3 depicts packet reflection. There are two multimedia streams shown. The top stream has a mean interarrival time of 160,000 microseconds. The lower stream has a mean interarrival time of 30,000 microseconds. When the packet first packet arrives late in the 160,000 stream, the subsequent packet arrives an equal amount early. However, when a packet arrives late in the 30,000 stream, the subsequent packet arrives almost immediately, being unable to reflect an equal amount. Mathematically, the top stream will have a larger variance, even though they both have an equal chance of a having a packet arrive late. This reflection effect is an artifact of the environment and does not accurately indicate the frequency and magnitude of late packets.

We performed three meta experiments to test if either packet size or packet rate changed the frequency and/or magnitude of the interarrival times. We subtract the mean from each interarrival time and drop all points that are below zero. This avoids the reflection effect and allows us to compare the frequency and magnitude of late packets. In the first experiment, we sent packets of three different sizes at three different rates: audio-rate (160,000 microseconds) and audio-size (1280 bytes); video-rate (33,000 microseconds) and video-size (6000 bytes); and mid-rate (100,000 microseconds) and mid-size (4000 bytes). Figure 4 depicts the results. The correlation between packet rate and variance is an extremely low 0.09.

In the second experiment, we sent packets all of the same size (1280 bytes) at four different rates: 30,000 microseconds, 70,000 microseconds, 110,000 microseconds, and 160,000 microseconds. Figure 5 depicts the results. The horizontal axis is the time between packets. The vertical axis is the variance. The correlation between packet rate and variance is an extremely low -0.02.

In both experiments, the correlation between packet rate and jitter was extremely low. This indicates that the amount of jitter we would expect to see in a high-rate video stream would be the same as the jitter from a lower-rate audio stream. For all subsequent experiments, we used the audio rate in order to avoid the reflection effect and to keep network and processor loads low. We can then measure the effects that increased network and processor load had on jitter. To simulate the transmission of audio, we sent 1280-byte datagrams every 160,000 microseconds.

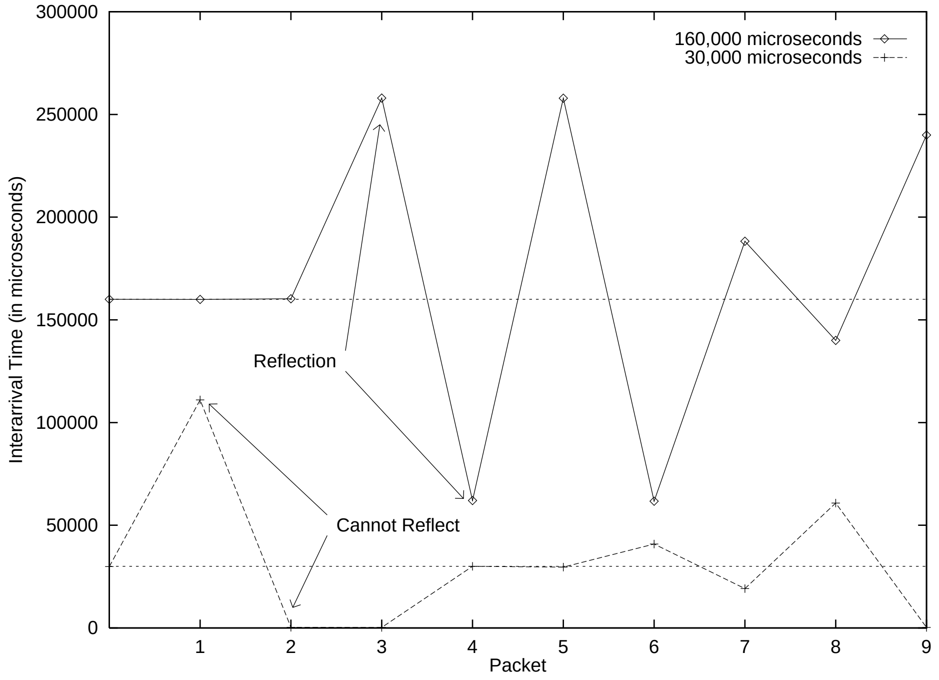

Figure 3: The Reflection Effect. The horizontal axis is the packet number. The vertical axis the interarrival time in microseconds. The zig-zag lines represent two multimedia streams with the top having a mean interarrival time of 160,000 microseconds and the bottom having a mean interarrival time of 30,000 microseconds.

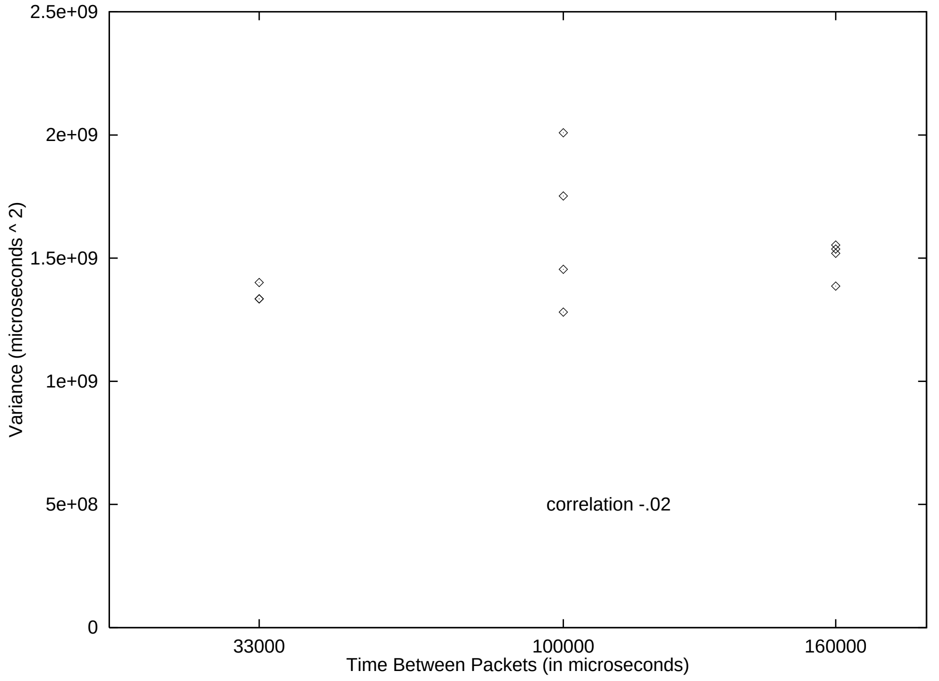

Figure 4: Jitter versus Packet Rate. The horizontal axis is the time between packets. The vertical axis is the variance. Each point represents an independent experiment. The correlation is 0.09.

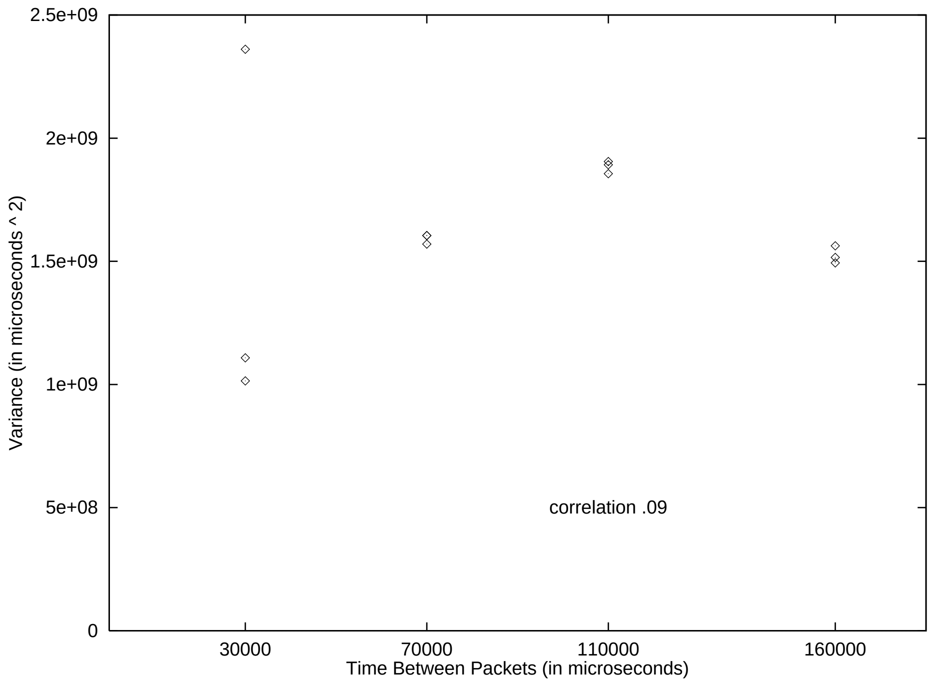

Figure 5: Jitter versus Packet Rate for 1280 Byte Packets. The horizontal axis is the time between packets. The vertical axis is the variance. Each point represents an independent experiment. The correlation is -0.02.

Workstation	MHz	SPECint92
SLC	20	8.6
IPC	25	13.8
IPX	40	21.8
Sparc 5	85	64.0

Table 3: Workstations used in Processor Experiments

4 Processor Experiments

Processor performance approximately doubles every year [20]. These high-performance processors will probably have a huge effect on improving the quality of current multimedia applications and may enable new multimedia applications. We explore the effects of processor performance on jitter, one component in multimedia application quality.

4.1 Specific Experimental Design

We used four classes of Sun processors: SLC, IPC, IPX and Sparc 5. Table 3 summarizes the workstation attributes. The workstations were connected to an 10 Mbits/second Ethernet. We used a process that increments a long integer variable in a tight loop to induce processor load.

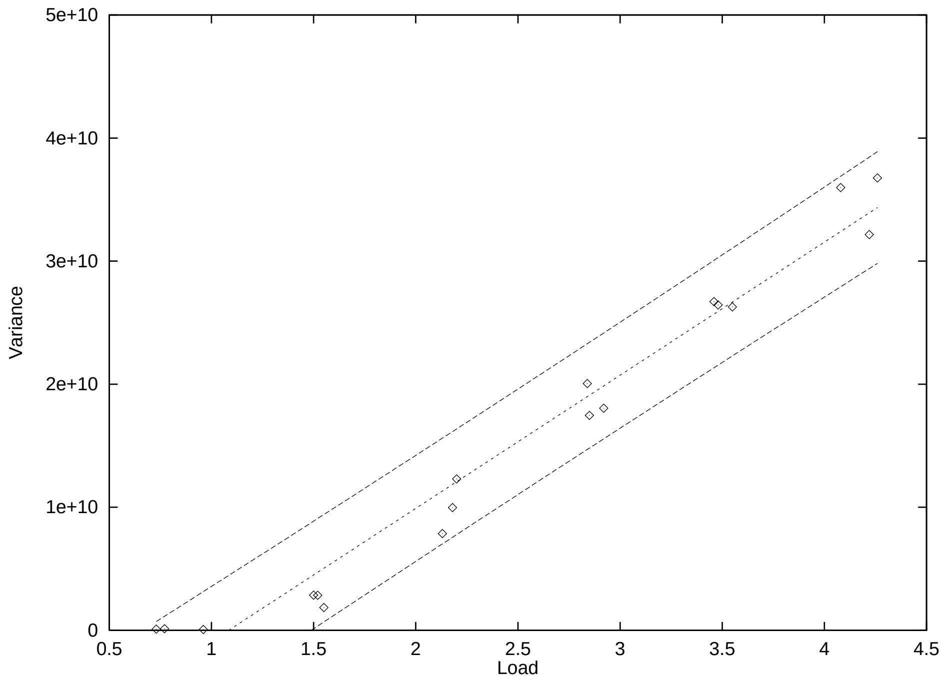

Figure 6: Sun SLC Jitter versus Receiver Load. The horizontal axis is the processor load as reported by the Unix `w` command. The vertical axis the the variance in interarrival times. The middle line is the least squares line fit. The outer two lines form a 95% confidence interval around the line. The correlation coefficient is 0.98.

4.2 Jitter versus Processor Load

We first test whether increasing processor load increases jitter. We ran an increasing number of counter processes on the receiver and obtained the processor load from the Unix `w` command at the end of each experiment run. The `w` command displays a summary of the current activity on the system, including the average number of jobs in the run queue over the last 1, 5 and 15 minutes. Lastly, we did a least squares line fit for the load versus variance and computed the correlation coefficient. Figure 6 shows the results of our experiments on the Sun SLC.

Figure 7 shows the results of our experiments on all the Sun workstations listed in Table 3. Again, we did a least-squares line fit for the load versus variance, for each class of processor. The slopes of the least squares line fits decrease as the processors get more powerful.

From the data presented, we conclude that jitter increases with processor load and jitter decreases with increased processor power.

4.3 Jitter versus Processor Power

We observed that more powerful processors decrease jitter under high loads, but what happens under conditions of more normal load?

In Figure 7, the lines come to nearly the same point at a processor load of 1. At a load of 1, we could show no correlation between jitter reduction and processor power, so we sought to isolate our experimental environment from the outside world to better observe the effects of the processor. Figure 8 depicts a series of attempts to do this. The top line in the picture represents the interarrival times for an experiment run in

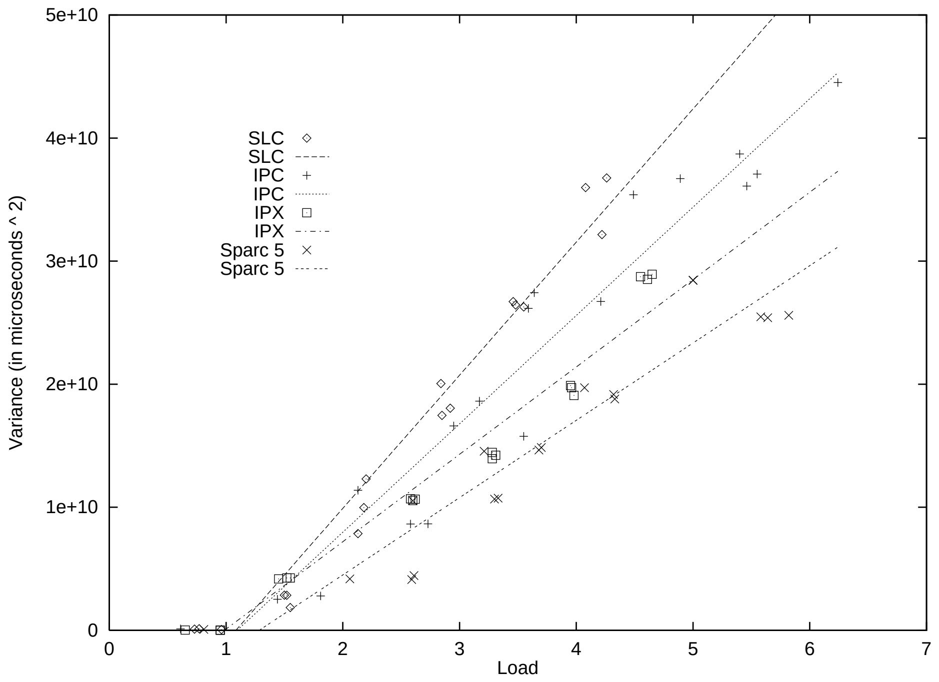

Figure 7: Jitter versus Receiver Load. The horizontal axis is the processor load as reported by the Unix `w` command. The vertical axis the the variance in interarrival times. The lines are the least squares line fits. The correlation coefficients range from 0.97 to 0.99.

normal multi-user processor mode during the day.

We wanted a quiet network to reduce effects of network on jitter. We assumed that most users do their work in the day, so there should have been less network traffic at night. We recorded interarrival times at night to see if they appeared significantly different than those recorded during the day. These times are depicted by the 2nd line from the top of Figure 8. The amount of jitter at night appears no better than the amount of jitter during the day.

We also wanted to reduce the effects of other workstation processes on jitter. We ran an experiment in single-user mode during the day and recorded the interarrival times, depicted by the 3rd line from the top of Figure 8. Still, the amount of jitter in single-user mode appears no better than the amount of jitter in multi-user mode.

We then tried a combination of single-user mode and nighttime. These interarrival times are depicted by the 2nd line from the bottom of Figure 8. The amount of jitter in this case is noticeably less than the amount of jitter from the multi-user mode experiment run during the day. However, we could still show no correlation between jitter reduction and processor power.

In our last effort to remove as much jitter as possible, we physically disconnected our workstations and subnet from the surrounding networks and ran our experiments in single-user mode. Success! These interarrival times are depicted by the bottom line in Figure 8. This nearly flat line represents a multimedia stream that is almost jitter-free.

Once we isolated the machines on a quiet network and ran our experiments in single user mode, we were able to observe the effects different power processors have on jitter under light loads. Figure 9 depicts these results. There is a strong correlation between increased processor power and decreased jitter.

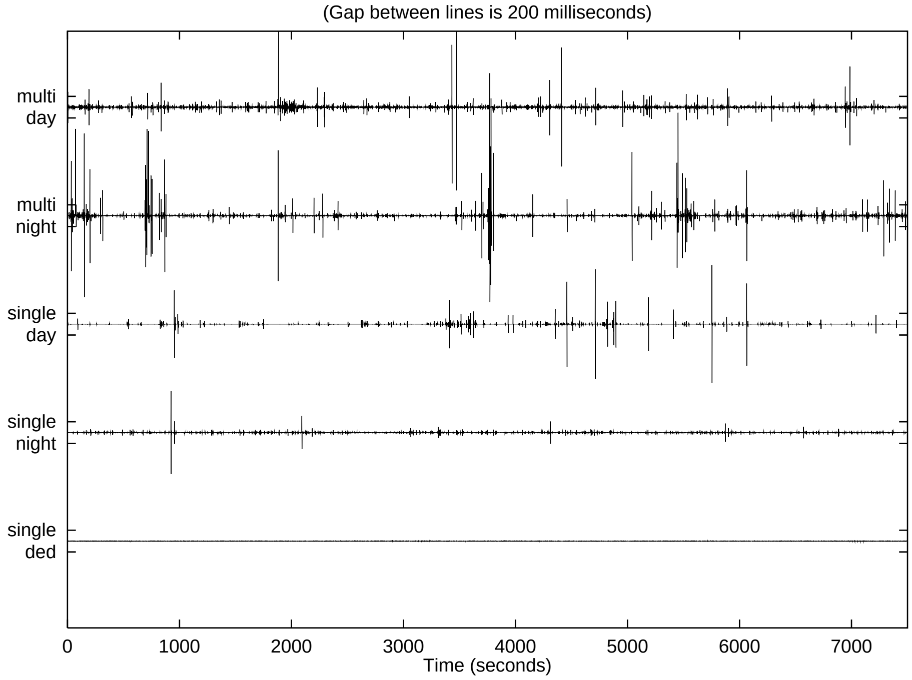

Figure 8: Effects on Jitter. There are 5 multimedia streams represented in this picture, each by a horizontal line depicting the interarrival times. "Single" is an experiment run in single-user mode. "Multi" is an experiment run in typical multi-user mode. "Day" is an experiment run in the middle of the day. "Night" is and experiment run at night. "Ded" is an experiment run on a dedicated network. Each multimedia stream is off-set from the one below it by 200 milliseconds.

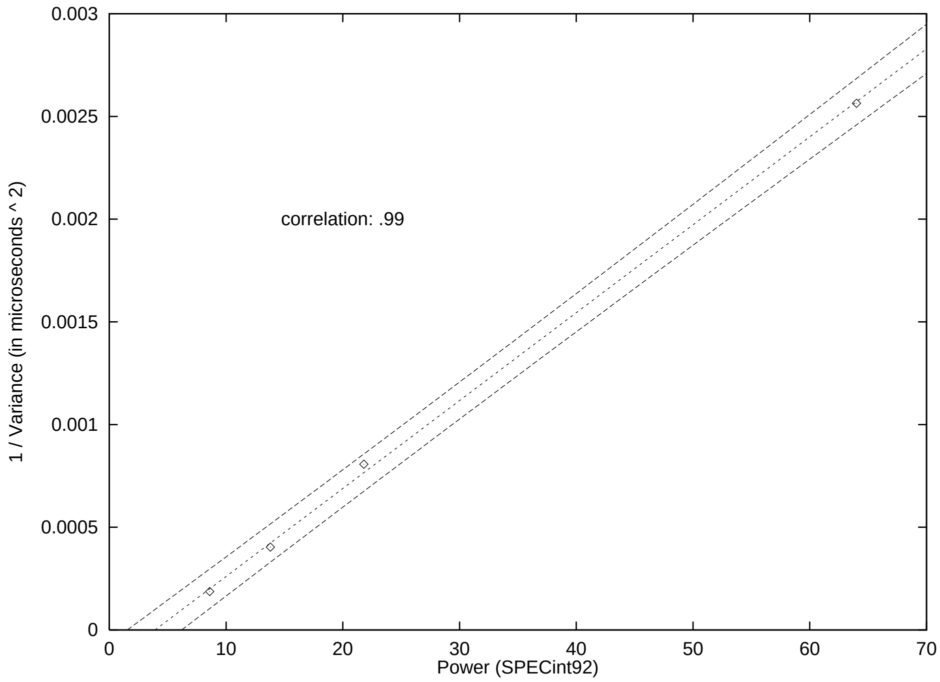

Figure 9: Jitter versus Power. The horizontal axis is the SPECint92 result of the workstation. The vertical axis is 1 / variance in packet interarrival times. The middle horizontal line is the least squares line fit. The outer two lines form a 95% confidence interval around the line. The correlation coefficient is 0.98.

4.4 Summary

Jitter correlates highly with processor load. Since multimedia applications are often processor intensive, they often force processors to run at a heavy load. Moreover, in the past, applications have tended to expand to fill (or surpass) available system capacity, making heavy-load conditions likely in the future. These heavily loaded workstations will contribute to the jitter in multimedia applications.

Fortunately, under heavy loads, faster processors reduce jitter compared to slower processors. However, under normal load, the reduction in jitter from the faster processors is overshadowed by the variance in average processor and network loads. To fully determine the benefits of faster processors to jitter, it is imperative to determine if future multimedia application will, in fact, push processor capacities to the limit. If so, future multimedia application quality will benefit from faster processors. If not, application quality should be improved by means other than processor improvement.

We can use our model in Section 7 to determine if current and future processor capacities are being consumed by multimedia applications. In addition, our model allows us quantitatively determine the benefit to application quality from high-performance processors for today's and tomorrow's multimedia applications.

5 Real-time Priority Experiments

From the previous experiments, single-user mode and a dedicated network can nearly eliminate jitter. However, in a multi-user system a dedicated processor is seldom available, and a dedicated network is even more rare. It seems unlikely that the above results alone will help users in most cases. We turn to real-time priorities to see how they compare to the benefits of a single-user mode processor under realistic load conditions.

5.1 Specific Experimental Design

We used Sun SPARC Classics running SunOS 5.4, a Unix variant. In Unix, a processes priority determines how much CPU time the process gets [39]. The kernel will decrease the priority of processes that have accumulated what it considers "excessive" CPU time. The priority of a process is not the only thing that determines when a process will be run. To determine which process should be run next, the scheduling mechanism in the kernel uses a formula that takes into account each process's priority, how much CPU time each process has gotten recently, and how long it has been since each process has run. SunOS 5.4 extends Unix process scheduling with real-time support. A real-time process runs in a separate scheduling class than normal processes. In the default configuration, a runnable real-time process runs before any other process. This gives real-time processes the highest priority.

Even without real-time extensions, Unix provides the `nice` facility. Nice allows you to change the default priority of a process. Using nice, it is possible to improve the user-mode priority of a process allowing it to run ahead of other eligible runnable user-mode processes. Processes using the nice facility may still suffer long dispatch latencies, however. Any process that is suspended waiting for I/O will, upon being re-activated, have a higher priority than any process that is in user-mode, regardless of its nice value [32].

We varied the process priority from Default (processes were run without enhanced priority), Niced (processes were invoked using the `setpriority(-19)` system call as an offset to determine the final user-mode priority of the process) and Real-Time (processes were given fixed priorities in the SunOS real-time scheduling class using `priocntl()` system call).

As in Section 4, we used a process that increments a long integer variable in a tight loop to induce processor load. We ran separate experiments with one through six of these counter processes.

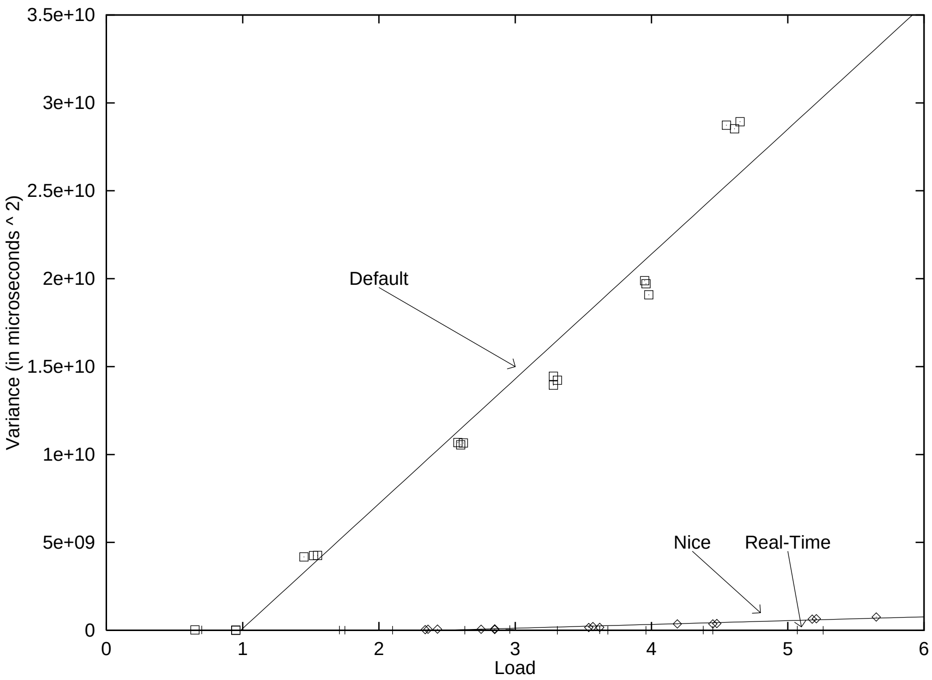

Figure 10: Jitter versus Receiver Load. The horizontal axis is the processor load as reported by the Unix `w` command. The vertical axis the the variance in interarrival times. The lines are the least squares line fits for default priorities, niced priorities and real-time priorities, as indicated.

5.2 Results and Analysis

Figure 10 compares the jitter for real-time, nice and default priorities as processor load increases. The slopes of the lines indicate how sensitive a process running under the given priority is to the effects of increases in processor load. The steeper the slope, the more jitter the process contributes as processor load increases. Both nice and real-time priorities have *significantly* gentler slopes than default priority, indicating that nice and real-time processes do not suffer nearly as much jitter as do as default priority processes as processor load increases.

Figure 11 is a close-up of the nice and real-time priorities in Figure 10. The most steeply line is for default priority. The next most steeply sloped line is nice priority and the horizontal line is real-time priority. Real-time priority has nearly a flat slope, indicating that real-time processes show almost *no* increase in jitter as processor load increases. Nice priority processes, however, do suffer from increased jitter with increased processor load, as indicated by its steep slope. Notice that the line for nice priority intersects the x-axis at about a load of two, as opposed to default priority which intersects the x-axis around a load of one. This indicates that even under light loads, nice priority processes reduce jitter as opposed to default priority processes.

5.3 Summary

Real-time priorities significantly reduce jitter as processor load increases. The severity of jitter that was observed when real-time priorities were used was extremely light compared to the jitter without real-time priorities. Real-time priorities show almost *no* increase in jitter as processor load increases. Real-time processes have priority over other processes, allowing them to respond to multimedia data with less jitter

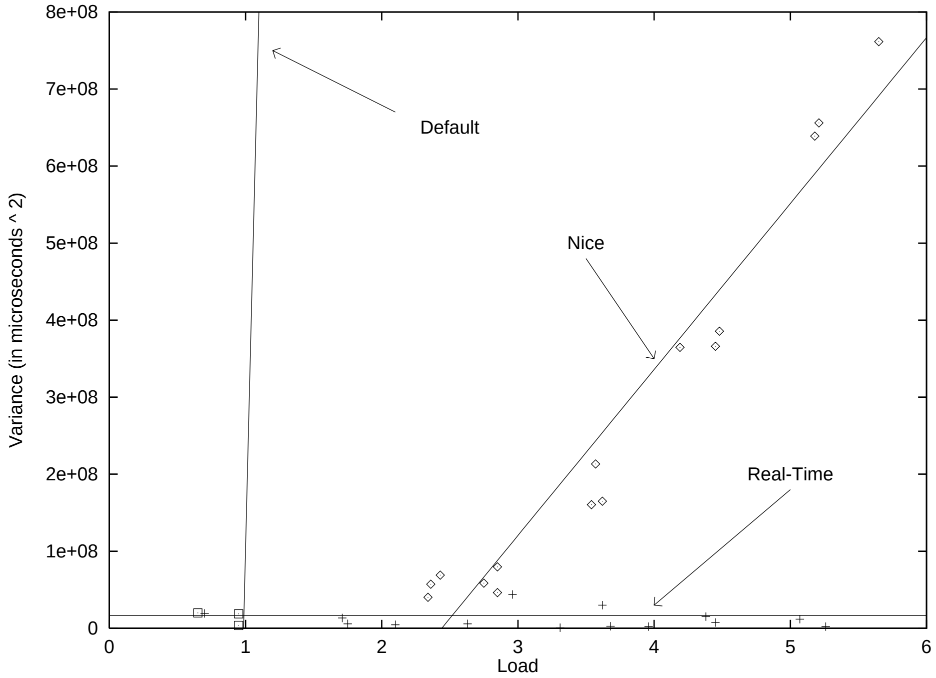

Figure 11: Zoom of Jitter versus Receiver Load. This graph is a close-up of Figure 10 by changing the upper bound on the vertical axis from 3.5e+10 to 8e+08. The horizontal axis is the processor load as reported by the Unix `w` command. The vertical axis the the variance in interarrival times. The lines are the least squares line fits for niced priorities and real-time priorities, as indicated.

than do nice or default priority processes. Note, however, that real-time priorities must be used carefully. A real-time process can starve other process of CPU time, including critical operating system processes. We found this out the hard way when we ran a counter process in real-time mode. The counter process consumed *all* of the CPU cycles, not even allowing enough CPU time to kill the process as the super-user. We were forced to reboot our computer. In addition, if many processes are run with real-time priorities the benefits in jitter reduction will most certainly be reduced compared with having just one process with real-time.

Nice priorities significantly reduce jitter as processor load increases. Under heavy loads, nice priority processes had much less jitter than did default priority processes. Even under light loads, nice priorities significantly reduced jitter versus default priorities. The improved user-mode priority of a niced process causes it to run ahead of other eligible runnable user-mode processes, allowing the niced process to respond to multimedia data with less jitter. However under heavy processor loads, nice priority processes did suffer from some increased jitter while jitter under real-time processes remained nearly constant. Even a nice priority process will run after any newly-activated process, regardless of its nice value, causing the niced process to suffer from more jitter than would a real-time process.

The amount of jitter reduction from nice or real-time priorities depends upon the processor load. We can use our model in Section 7 to determine the processor load from multimedia applications. With our model we can also explore how the reduction in jitter from nice and real-time priorities benefits the application quality as future multimedia applications push processors to the limit.

6 Network Experiments

From the experiments in Section 4 and Section 5, we know that high-performance processors and real-time priorities affect jitter. We next examine the effects that high-speed networks have on jitter.

6.1 Specific Experimental Design

We used 4 processor SGI Challenge L workstations. Each workstation was connected to three networks: a 10 Mbit/s Ethernet; a 266 Mbit/s Fibre Channel; and a 800 Mbit/s HIPPI.

We induced network load using `netperf` [29]. Netperf is a benchmark that can be used to measure the performance of many different types of networks. It provides tests for both uni-directional throughput and end-to-end latency.

Both HIPPI and Fibre Channel are point-to-point networks, unlike Ethernet which is a shared medium network. Inducing network load between two workstations that were not the sender and receiver did not increase jitter because they did not share the same physical wire. Instead, we ran the `netperf` and `netserver` processes on the same workstations as the sender and receiver, respectively. We ran the netperf processes on different processors than the sender and receiver by using the `runon` command. This minimized the jitter that might be contributed by the processor load induced by the `netperf` processes.

6.2 Results and Analysis

Figure 12 depicts the experiments results that show the interarrival times for the three networks under two different conditions of network load. The "no load" runs are the experiments on a quiet network. The "load" runs are the experiments run with `netperf`. The three "no load" lines are almost indistinguishable, indicating no correlation between jitter and network bandwidth. However, under high network load there is a noticeable difference in the interarrival times for the three networks, indicating there may be a correlation between jitter and network bandwidth.

Figure 13 shows the relationship between jitter and network bandwidth for loaded networks. We expect that jitter decreases as network bandwidth increases, so we graphed the theoretical network bandwidth versus 1/variance. There are three data points plotted, one for each of Ethernet, Fibre Channel and HIPPI. The line represents a least squares line fit through the three data points. The correlation coefficient is 0.98, indicating that there is a strong inverse relationship between jitter and network bandwidth. In other words, as network bandwidth increases, jitter decreases. Note that although the correlation coefficient is a high 0.98, the fact that there are only three data points is evident in the width of the 95% confidence intervals around the least squares line fit. It would be nice to have more data points in order to strengthen the statistical significance of our results. To do this, however, we need to have new networks on which to experiment.

6.3 Summary

Under low network loads, high-speed networks do not significantly reduce jitter. Under low loads, the network contributes little variation to the interarrival times for the multimedia frames. Most of the variance is caused outside the network, rendering any jitter reduction from the high-speed networks insignificant.

However, under heavy network loads, high-speed networks significantly reduce jitter. Under heavy loads, interfering network traffic causes increased variation in the interarrival times for the multimedia frames. High-speed networks reduce the amount of time to deliver the interfering network traffic, decreasing the variation in the multimedia interarrival times.

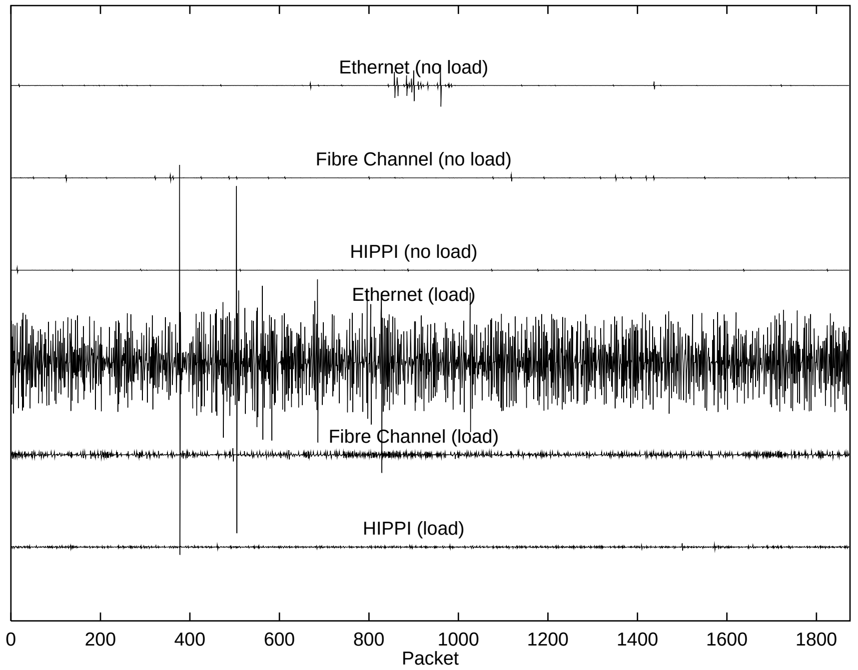

Figure 12: Interarrival Times for Different Networks and Network Loads. There are 6 multimedia streams represented in this picture, each by by a horizontal line depicting the interarrival times on the indicated network. The network was either either loaded or unloaded. Each multimedia stream is off-set from the one below it by 750,000 microseconds. The horizontal axis is the packet number.

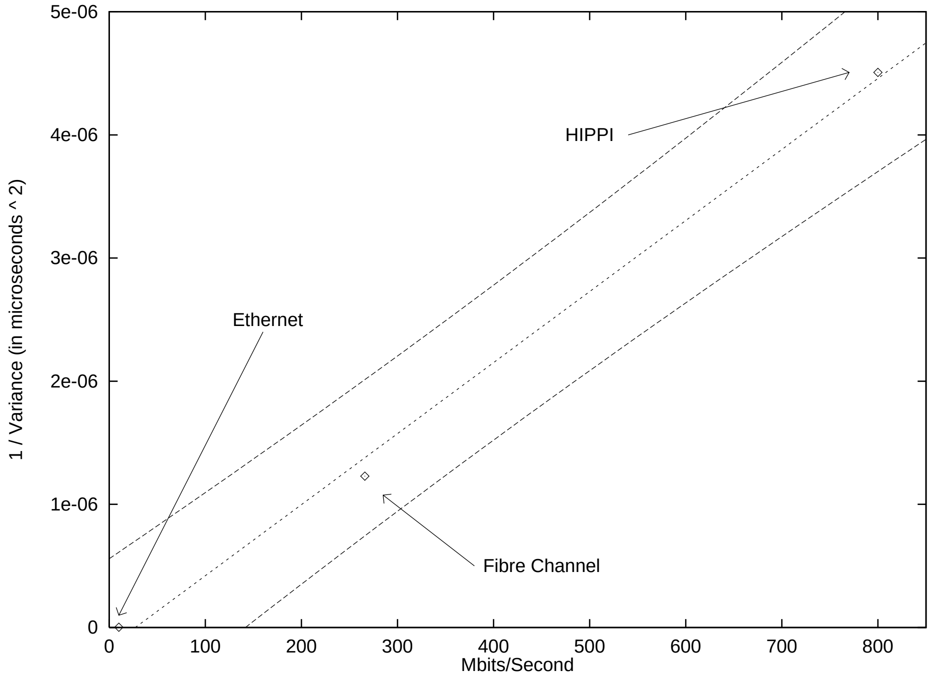

Figure 13: Jitter versus Network Bandwidth. The horizontal axis is the network bandwidth. The vertical axis is 1 / variance. The line is a least squares line fit of jitter versus theoretical network bandwidth for three networks: Ethernet, Fibre Channel and HIPPI. The correlation coefficient is 0.98.

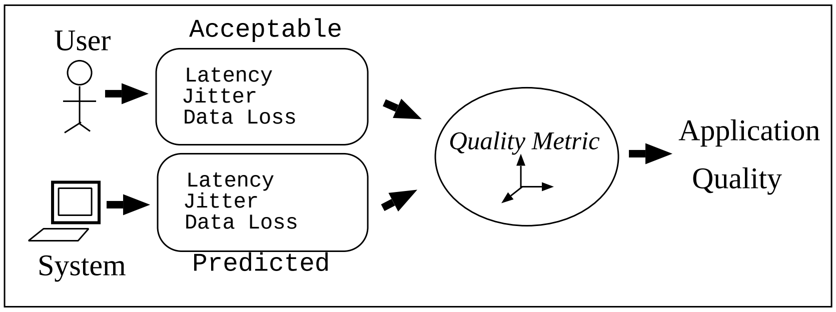

Figure 14: The Process for Computing Application Quality. The user defines the acceptable latency, jitter and data loss and the system determines the actual values. Based on the acceptable values specified in the user requirements, a quality metric computes the application quality from the actual values.

The amount of jitter reduction from high-speed networks depends upon the network load. We use our model in Section 7 to determine the network load from multimedia applications as the number of users increases. Then, using the results from this section, we can determine what affect jitter reduction from high-speed networks has on the application quality.

7 Quality

The quality of a distributed multimedia application is a measure of the application's acceptability to the user. A measure of jitter alone is not sufficient to predict the quality of an application. As described in Section 1, in addition to jitter, we have identified two other measures that determine quality for most distributed multimedia applications: latency and data loss.

There may be additional measures that affect application quality that are application specific. For example, Distributed Interactive Simulation applications use a process of computing the location of other simulators through "dead reckoning" [38]. When state update packets are dropped, the accuracy of the simulation decreases [9]. The use of dead reckoning creates an additional quality measure specific to DIS applications. In the rest of this paper, we consider only delay, jitter and data loss, but our analysis could be similarly applied to such application specific measures.

Ideally, we would like there to be no latency, jitter or data loss. Unfortunately, on a variable delay network and non-dedicated computer this can never be achieved. To compute the application quality, we use the above quality components in a process depicted by Figure 14. The user requirements for the application define the acceptable latency, jitter and data loss. The system determines the predicted latency, jitter and data loss. Acceptable and projected data are fed into a *quality metric* for the application. The quality metric is a function, based on the acceptable components and dependent upon the projected components, that computes the application quality.

In order to quantitatively compare application quality for different system configurations, we need a reasonable quality metric. To form our quality metric, we build upon the work of Naylor and Kleinrock [31]. Naylor and Kleinrock developed a model for measuring the quality of an audioconference based on the amount of dropped frames and client-side buffering. We extend this model by using each quality component as one axis, creating a multi-dimensional quality space. We place the best quality value for each axis at the origin and scale each axis so that the user-defined minimum acceptable values have an equal weight. An instantiation of the application lies at one point in this space. We compute the application quality by taking the Euclidean distance from the point to the origin. All points inside the region defined by the user-defined minimums have acceptable quality while points outside do not.

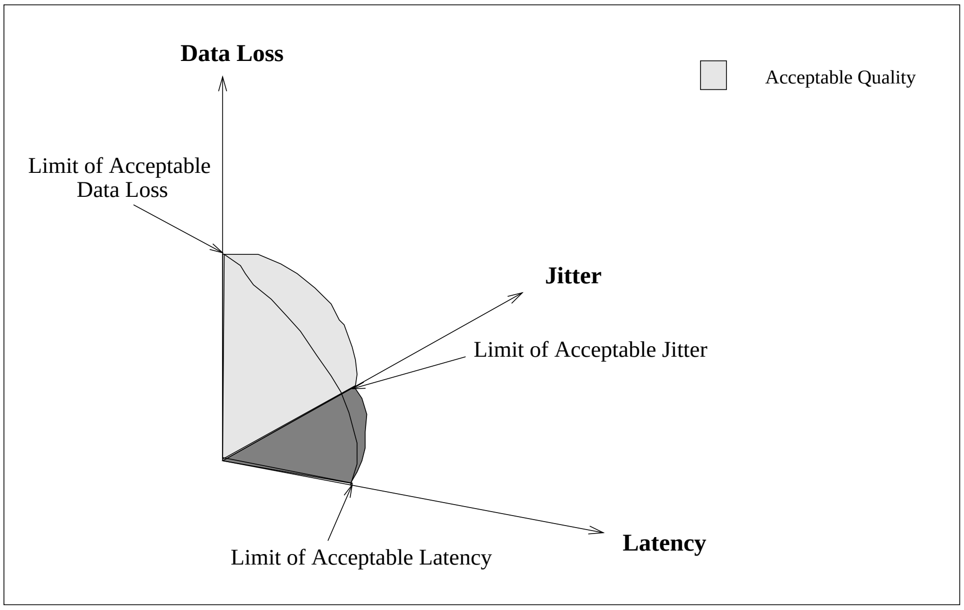

Figure 15: Application Quality Space. The user defines the acceptable latency, jitter and data loss. These values determine a region of acceptable application quality, depicted by the shaded region. All points inside the shaded region have acceptable quality, while those outside the region do not. An instantiation of the application and the underlying computer system would lie at one point in this space.

Figure 15 depicts a 3-d quality space for multimedia applications. The user requirements determine a region of acceptable application quality, depicted by the shaded region. All points inside the shaded region have acceptable quality, while those outside the region do not. An instantiation of the application and the underlying computer system would lie at one point in this space.

There can be many possible quality metrics for a given application. In fact, there may be many quality metrics that agree with a user's perception of the application. Mean opinion score (MOS) testing can be used to determine if a metric agrees with users' perception. The MOS is a five-point scale where a MOS of 5 indicates perfect quality and a score of 4 or more represents high quality. MOS has been used extensively in determining the acceptability of coded speech. MOS testing is beyond the scope of this paper, so we cannot be certain our quality metric fits user perceptions. However, the metric we chose has several useful characteristics. First, it treats the axes symmetrically which seems appropriate in the absence of user studies to the contrary. Second, the Euclidean distance fits our intuition about changes in quality: the measure increases total quality with any increase in quality along one axis. Third, the metric produces a convex region of acceptable quality, which avoids certain anomalies. For example, there are no pockets of unacceptable quality within the acceptable quality region, nor can you move from unacceptable to acceptable by any combinations of increase along the axes. The rest of our model is independent of the quality metric chosen. If new metrics are developed and validated with MOS testing, they can be used in place of our quality metric.

One limitation to quality metrics is that after scaling, the upper limits on the axes have different characteristics. The "data loss" axis has a finite upper-limit of 100%, while the "latency" and "jitter" axes each have an infinite bound. Comparing application quality for two different configurations at the upper-limit of any of the axes may not match user perception. Fortunately, this limitation only arises when comparing two unacceptable configurations. The metric is most valuable for determining whether a configuration provides "acceptable" or "unacceptable" application quality and comparing configurations within the "acceptable" region.

Note that the user-defined acceptability limits along each axis are greatly dependent upon the application and must be re-evaluated for each new application. For example, the acceptable latency for an audio broadcast application such as a radio program may be far more than the acceptable latency for an audioconference. In an audioconference, users require low latencies so that the conversation is as life-like as possible. According to Partridge, one-way delays over 400,000 microseconds are unacceptable for live audio [41]. However, in an audio broadcast program, the users do not interact, allowing a larger delay to go unnoticed. You could imagine a case where a user downloads an entire radio program overnight and then plays it back in the morning. In this case, a latency of over twelve hours might be quite acceptable.

Our quality model may be used to compare the benefits from different jitter reducing techniques, from application techniques such as buffering to hardware techniques such as high-speed networks. This allows us to find the bottleneck in reducing jitter and determine which techniques reduce jitter the most, possibly directing further jitter reduction research. In addition, we can use our model to evaluate the potential performance benefits from expensive high-performance processors and high-speed networks before installing them. We can even investigate possible performance benefits from networks and processors that have not yet been built. In the next section, we apply our model to videoconferences. We use our model to examine the effects of hardware-level jitter reducing techniques of high-speed processors and high-speed networks, and the system-level jitter reducing technique of real-time priorities.

8 An Example: Videoconference Quality

In this section, we apply the predictive powers of our quality model presented in Section 7 to a videoconference. We can learn a lot from videoconferences. Videoconferences incorporate both audio and video. Interactive videoconferences can have from two to tens of users, while videoconference broadcasts can have hundreds or perhaps even thousands of viewers. In addition, videoconferences are often integrated into larger distributed multimedia applications. Predicting quality for various system configurations to support videoconferences is valuable for business' wishing to invest in videoconference technology. Our model allows identification of computer systems that will provide acceptable videoconference quality and a comparison of their costs.

In order to apply our quality model to a videoconference under various system configurations, we must: 1) determine the region of acceptable videoconference quality; 2) predict jitter; 3) predict latency; and 4) predict data loss.

8.1 The Region of Acceptable Videoconference Quality

To determine the region of acceptable videoconference quality, we need to define acceptable limits for videoconferences along each of the latency, jitter and data loss axes. According to Jeffay and Stone, delays of 230 milliseconds or under are acceptable for a videoconference [25]. For data loss, research in remote teleoperator performance has found that task performance is virtually impossible below a threshold of 3 frames per second [37]. We use 3 frames per second as the minimum acceptable frame rate.

The presence of jitter often presents an opportunity for a tradeoff among latency and data loss. Buffering, an application-level technique for ameliorating the effects of jitter, can compensate for jitter at the expense of latency. Transmitted frames are buffered in memory by the receiver for a period of time. Then, the receiver plays out each frame with a constant latency, achieving a steady stream. If the buffer is made sufficiently large so that it can hold all arriving data for a period of time as long as the tardiest frame, then the user receives a complete, steady stream. However, the added latency from buffering can be disturbing [41], so minimizing the amount of delay compensation is desirable.

Another buffering technique to compensate for jitter is to discard any late frame at the expense of data loss.

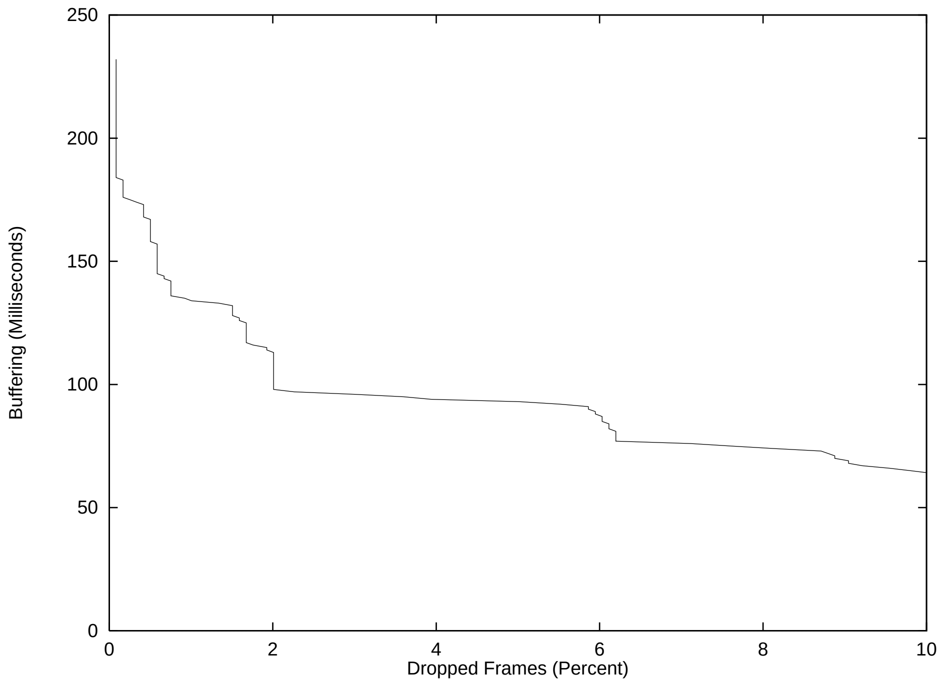

Figure 16: Jitter Compensation. This picture depicts the amount of buffering needed for a given number of dropped frames. The horizontal axis is the percentage of dropped frames. The vertical axis is the number of milliseconds of buffering needed.

Discarding frames causes a temporal gap in the play-out of the stream. Discarding frames can keep play-out latency low and constant, but as little as 6% gaps in the playout stream can also be disturbing [31]. In the case of audio speech, the listener would experience an annoying pause during this period. In the case of video, the viewer would see the frozen image of the most recently delivered frame.

Naylor and Kleinrock describe two policies that make use of these buffering techniques: the E-Policy (for Expanded time) and the I-Policy (for late data Ignored) [31]. Under the E-policy, frames are never dropped. Under the I-policy, frames later than a given amount are dropped. Since it has been observed that using a strict E-Policy tends to cause the playout latency to grow excessively and that dropping frames occasionally is tolerable [6, 48], we use the I-Policy as a means of examining needed jitter compensation for a multimedia stream.

The I-policy leads to a useful way to view the effects of jitter on a multimedia stream. Figure 16 depicts the tradeoff between dropped frames and buffering as a result of jitter. We generated the graph by first recording a trace of interarrival times. We then fixed a delay buffer for the receiver and computed the percentage of frames that would be dropped. This represents one point in the graph. We repeated this computation with buffers ranging from 0 to 250 milliseconds to generate the curved line. The graph can be read in two ways. In the first, we choose a tolerable amount of dropped frames (the horizontal axis), then follow that point up to the line to determine how many milliseconds of buffering are required. In the second, we choose a fixed buffer size (the vertical axis), then follow that point over to the line to determine what percent of frames are dropped. In Figure 16, if we wish to restrict the amount of buffering to 100 milliseconds, then we must drop about 2% of the frames since that is how many will be more than 100 milliseconds late, on average. For an 2 Mbps video stream consisting of 33 6000-byte frames per second, this equates to dropping one frame every 1.5 seconds. On the other hand, if we wish to not drop any frames, we have to buffer for over 200 milliseconds.

How much buffering should we chose? We normalize the axes from 6% gaps to 230 milliseconds buffering.

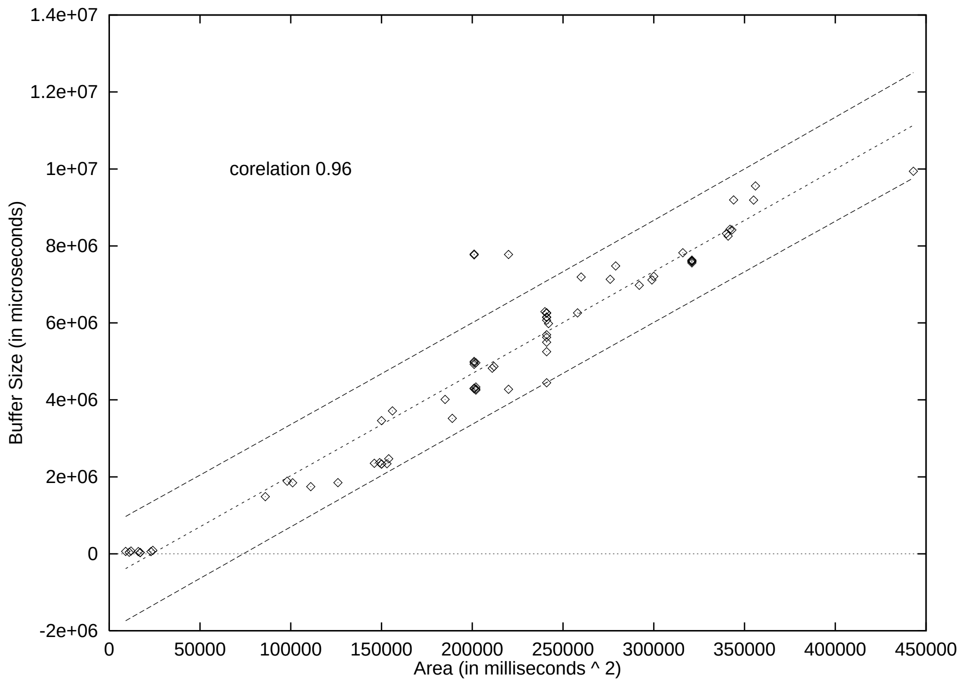

Figure 17: Jitter Compensation Area versus Buffer Size. The horizontal axis is the area under the jitter compensation curve. The vertical axis is the buffer size in microseconds. The points are each a separate experiment run. The middle line is the least squares line fit. The outer two lines form a 95% confidence interval around the line. The correlation coefficient is 0.96.

The best quality value in the jitter compensation curve is the closest point to the origin along the curve. We would like to know how much latency is added from buffering at this point. It seems natural to assume that as the area under the jitter compensation curve gets larger, the amount of buffering at the closest point along the curve gets larger. We hypothesize that there is a strong correlation between the area under the jitter compensation curve and the buffer size. If this hypothesis is true, we can determine the latency attributed to buffering from the jitter compensation area. We tested our hypothesis by generating jitter compensation curves for all data points from the experiments detailed in Sections 4, 5 and 6. We then computed the area under each curve. We computed the buffer size by normalizing the axes as described in Section 8.1 and finding the lowest Euclidean distance to the origin along the curve. We plotted buffer size versus area and computed the correlation coefficient. Figure 17 depicts these results. There is a high correlation between jitter compensation area and buffer size.

We can predict the optimal amount of buffering if we know the area of under the jitter compensation curve. How can we determine the area under the jitter compensation curve? As the amount of jitter experienced by the system gets larger, more buffering should be required to alleviate the effects of jitter and more gaps should appear in the multimedia stream. Thus, the area under the jitter compensation curve should get larger as jitter increases. We hypothesize that there is a strong correlation between the area under the jitter compensation curve and the variance in the packet interarrival times. If this hypothesis is true, then we can predict the area under the jitter compensation curve from the amount of jitter. Then, we can predict the buffer size from the jitter compensation area. We tested our hypothesis by computing jitter from all data points from the experiments detailed in Sections 4, 5 and 6. We then compared these jitter values to the areas under the jitter compensation curves from our previous hypothesis. We plotted area versus jitter and computed the correlation coefficient. Figure 18 depicts these results. There is a high correlation between jitter and compensation curve area.

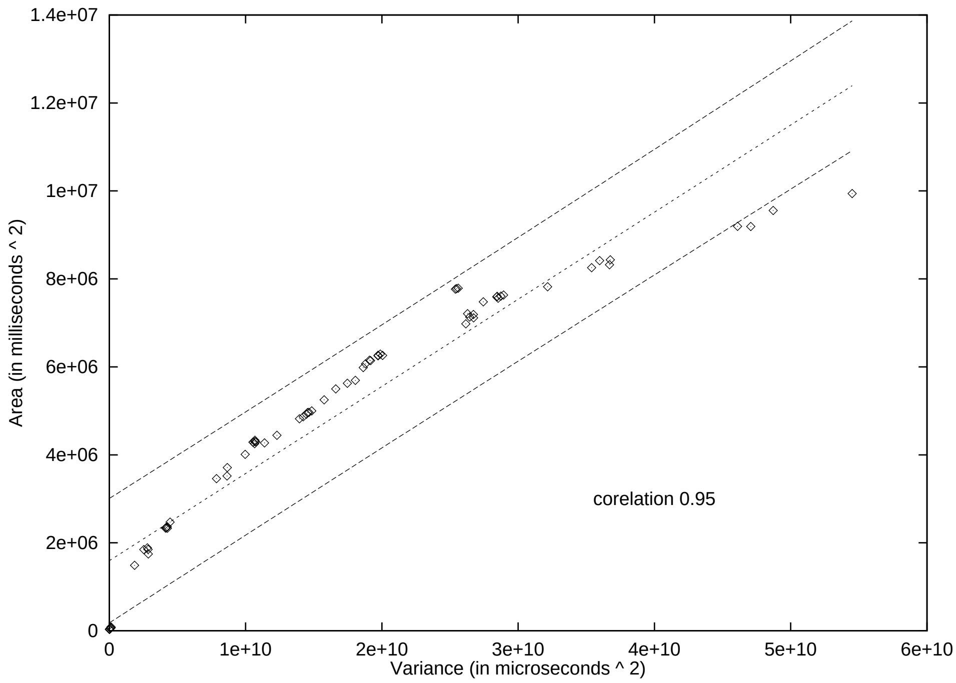

Figure 18: Jitter versus Jitter Compensation Area. The horizontal axis is the jitter (variance in packet interarrival times). The vertical axis is the area under the jitter compensation curve. The points are each a separate experiment run. The middle line is the least squares line fit. The outer two lines form a 95% confidence interval around the line. The correlation coefficient is 0.95.

From the above graphs, we have:

$$Buf. = 26.5 \times Area - 624,000$$

$$Area = 0.000198 \times Jitter + 1,590,000$$

Substituting the equation for Area into the equation for Buffer, we have:

$$Buf. = 0.00525 \times Jitter + 41,500,000$$

This last equation allows us to compute the optimal buffer size for an application given the amount of jitter.

8.2 Predicting Jitter

From our results in Sections 4 and 6, we know the relationship between load and jitter for faster processors and networks. We hypothesize that a high network load along with a high processor load increases jitter by the sum of the jitter from the network and processor. While perhaps seeming obvious, our hypothesis will be false if processor jitter and network jitter are not independent. If some of the jitter attributed to the processor is actually due to jitter from the network and/or some of the jitter attributed to the network is actually due to jitter from the processor, adding the jitter from each component will result in more jitter

Network	Processor	Network	Predicted	Actual
Ethernet	564.63	411.37	976.00	913.72
Fibre Channel	590.60	58.33	648.93	641.14
HIPPI	501.15	28.62	529.77	538.41

Table 4: Predicted versus Actual Jitter

than the system actually experiences. However, if our hypothesis is true, we can then add the predicted jitter from each component in predicting jitter for systems with both components.

We tested our hypothesis with an experiment. We first computed the jitter we would predict on a system with a loaded processor and loaded network. This prediction is based on the amount of jitter attributed from the processor as obtained in results from Section 4 and from the network as obtained in results from Section 6. We next experimentally measured the jitter with a loaded processor and a loaded network for each of the Ethernet, Fibre Channel and HIPPI networks. We then compared the predicted results to the actual results. Table 4 depicts this comparison. The predicted jitter values are within 7% of the actual jitter values. It seems appropriate to add the jitter attributed to processor load alone with the jitter attributed to network load alone to predict the jitter attributed to processor load and network load together.

8.3 Predicting Latency

We can predict the amount of latency from the jitter compensation buffer by using predictions on the amount of jitter. In addition to the buffering latency, there is the additional latency from the sender processing, the network transmitting and the receiver processing. From our previous experiments, we measured the latency from recording and playing video [9]. From other previous experiments, we measured the latency attributed to sending and receiving packets [8]. We can compute the latency from the network based on the frame size and network bandwidth. To predict the total latency, we add the latencies from: recording the video frame; sending the video frame to the client; receiving the video frame from the receiver; buffering in the jitter compensation curve; and playing the video frame.

8.4 Predicting Data Loss

In order to predict data loss, we need to identify what form data loss may take and when data loss may occur. In general, data loss can take many forms such as reduced bits of color, jumbo pixels, smaller images, dropped frames and lossy compression. For a videoconference, we assume data loss only in the form of dropped frames or reduced frame rate. For a videoconference, we assume data loss under three conditions:

- *Voluntary.* As described in Section 8.1, an application may chose to discard late frames in order to keep playout latency low and constant. We assume the videoconference chooses to discard enough frames to achieve the best quality.
- *Saturation.* When either the network or the processor do not have sufficient capacity to transmit data at the required frame rate, data loss occurs. For example, if the network has a maximum bandwidth of 5 Mbps and the videoconference required 10 Mbps there will be a 50% data loss. We can compute when systems reach capacity based on our previous work measuring processor capacities [8, 9] and theoretical network bandwidths.
- *Transmission Loss.* In our previous experiments, we found that typically about 0.5% packets on the average are lost when the network is running under maximum load [9]. We assume a maximum lost data rate of about 0.5% due to network transmission.

8.5 Predicting Quality

At last! We have built and validated an experiment-based model that will allow us to explore videoconference quality under different system configurations. We can quantify how effectively today's computer systems support multi-person videoconferences. We can predict when today's systems will fail due to too many users or too much load on the processors or networks. We can see how much using real-time priorities will help videoconference quality. We can evaluate the benefits of expensive high-performance processors and high-speed networks before installing them. We can even investigate possible performance benefits from networks and processors that have not yet been built. Let's go exploring!

We predict application quality for three scenarios: 1) high-performance processors and high-speed networks; 2) increasing users; and 3) increasing load.

8.5.1 Present Videoconference Assumptions

For all of our videoconference quality predictions we assume:

- *Multicast.* Our previous work has found that multicast is crucial for many-person multimedia applications [9]. Using unicast routing, multi-person multimedia applications saturate existing networks for even a few participants. Multicast routing dramatically increases the user scalability of multi-person applications.
- *Specialized Hardware.* The CPU load for processing video frames can be substantial [7]. We assume specialized hardware that does most of the computation required for video frame processing.

8.5.2 Future High-Performance Processors and High-Speed Networks

Our results in Sections 4 and 6 showed that both high-performance processors and high-speed networks reduce jitter. However, which reduces jitter more? And more importantly, which improves application quality more?

We assume we have five videoconference participants. In Section 8.5.3, we use our model to evaluate quality for a variable number of users, but here we evaluate a likely videoconference configuration that has interesting quality predictions. We compute quality under two different scenarios. In the first, processor load remains constant while the network bandwidth increases. In the second, network bandwidth remains constant while processor power increases. Figure 19 shows these predictions. For five users, increasing the processor power to a SPECint92 of 40 or greater results in acceptable videoconference quality. At no time does increasing the network bandwidth result in an acceptable quality. In this scenario, we conclude that processor power influences videoconference quality more than does network bandwidth.

8.5.3 Future Users

While today's computer systems may struggle to support even five videoconference participants, tomorrow's processor improvements promise to support more and more users. But how many more? How do more and more videoconference users affect application quality? Figure 20 depicts the predicted effects of increasing users on videoconference quality. We predict videoconference quality for three different videoconference configurations: a low-end workstation with a typical network (Sun IPX and Ethernet), a mid-range workstation with a fast network (Sun Sparc 5 and Fibre Channel), and a high-performance workstation with a high-speed network (DEC Alpha and HIPPI). As we saw in Section 8.5.2, today's typical workstations and networks cannot support even five videoconference participants. However, powerful workstations such as Sun Sparc 5s connected by fast networks such as a Fibre Channel can support up to 10 users. Very high-performance

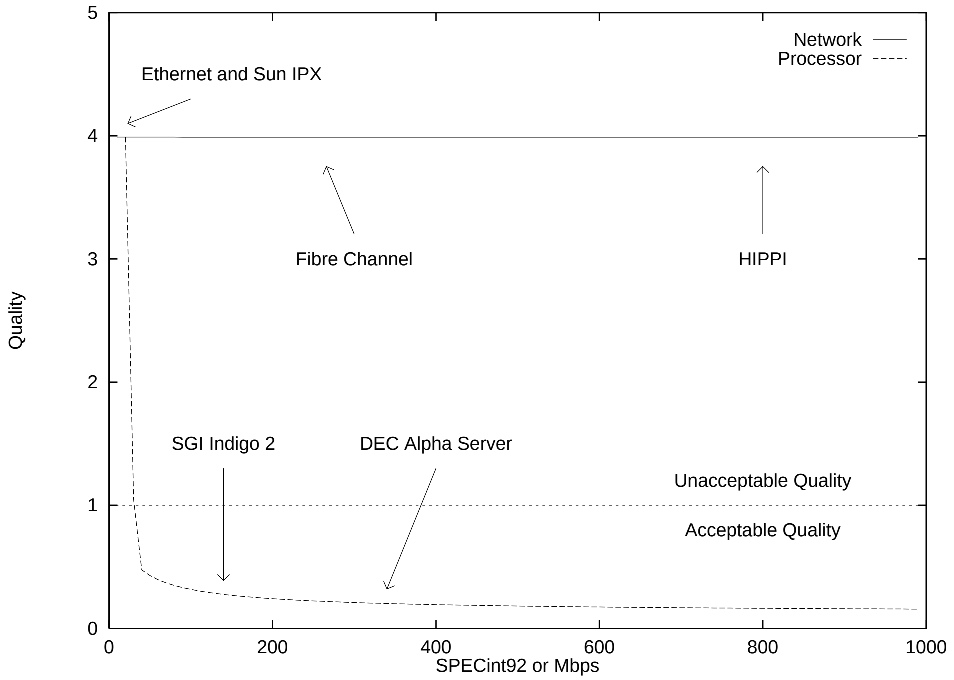

Figure 19: Videoconference Quality versus Processor or Network Increase. The horizontal axis is the SPECint92 power of the workstation or the network Mbps. The vertical axis is the predicted quality. There are two scenarios depicted. In the first, the processor power is constant, equivalent to a Sun IPX (SPECint92 = 22), while the network bandwidth increases. This is depicted by the solid curve. In the second scenario, the network bandwidth is constant, equivalent to an Ethernet (10 Mbps), while the processor power increases. This is depicted by the dashed curve. The horizontal line marks the limit between acceptable and unacceptable videoconference quality.

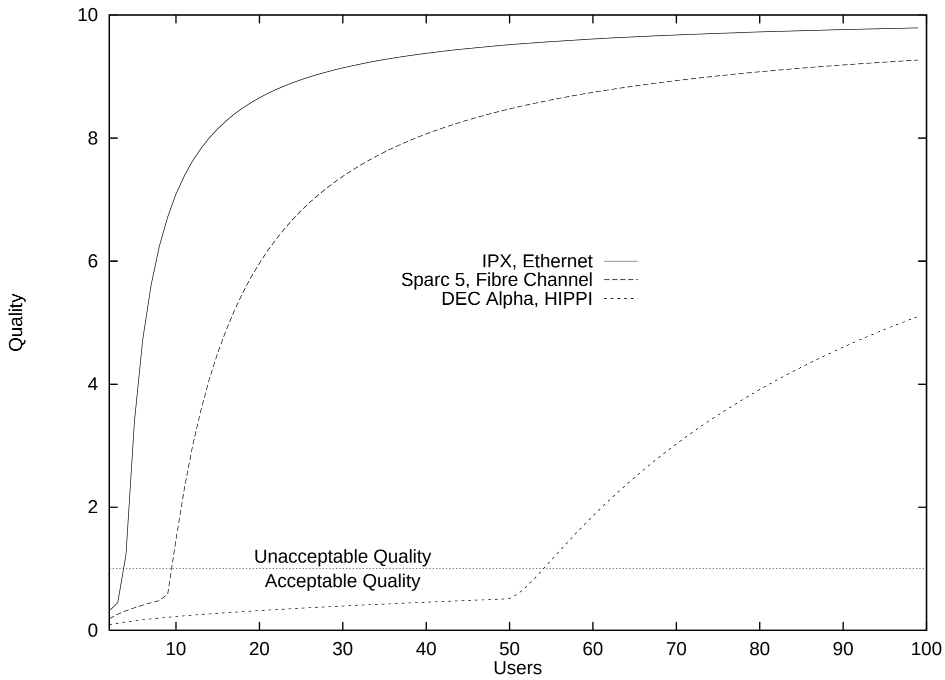

Figure 20: Videoconference Quality versus Users. The horizontal axis is the number of users. The vertical axis is the predicted quality. There are three scenarios depicted. In the first, the processors is a Sun IPXs connected by an Ethernet. In the second, the processors is a Sun Sparc 5s connected by a Fibre Channel. In the third, the processors are DEC Alphas connected by a HIPPI. The horizontal line marks the limit between acceptable and unacceptable videoconference quality.

workstations such as a DEC Alpha connected by high-speed networks such as a HIPPI can support over 50 users.

8.5.4 Future Processor and Network Load

Videoconferences are resource intensive, forcing processors and networks to run at a heavy loads. In addition, videoconference streams are often integrated into larger distributed multimedia applications. In the past, applications have tended to expand to fill (or surpass) available system capacity. As system capacities increase, videoconference users will demand higher frame rates and better resolution, making heavy-load conditions likely in the future. We predict the effects of increasing load on videoconference quality.

Figure 21 depicts the predicted effects of load on videoconference quality (remember, we are assuming specialized hardware for video processing and multicast routing). There are three classes of systems depicted. A traditional system has Sun IPXs connected by an Ethernet. A high-end system has Sun Sparc 5 connected by a Fibre Channel. A very high-end system has DEC Alphas connected by a HIPPI. The predictions for videoconference quality are almost identical for the three systems. We saw in Section 8.5.2 that the processor is more crucial than network for videoconference quality. Increasing processor load has a larger effect on decreasing videoconference quality than does improving the network speed and processor power.

Figure 21 also depicts Sun IPXs connected by an Ethernet but using real-time priorities instead of default priorities, shown by the bottom line. With real-time priorities, videoconference quality does not suffer from increased jitter from the processor as processor load increases. For conditions of increasing load, real-time priorities have a greater effect on improving quality than do faster processors and faster networks.

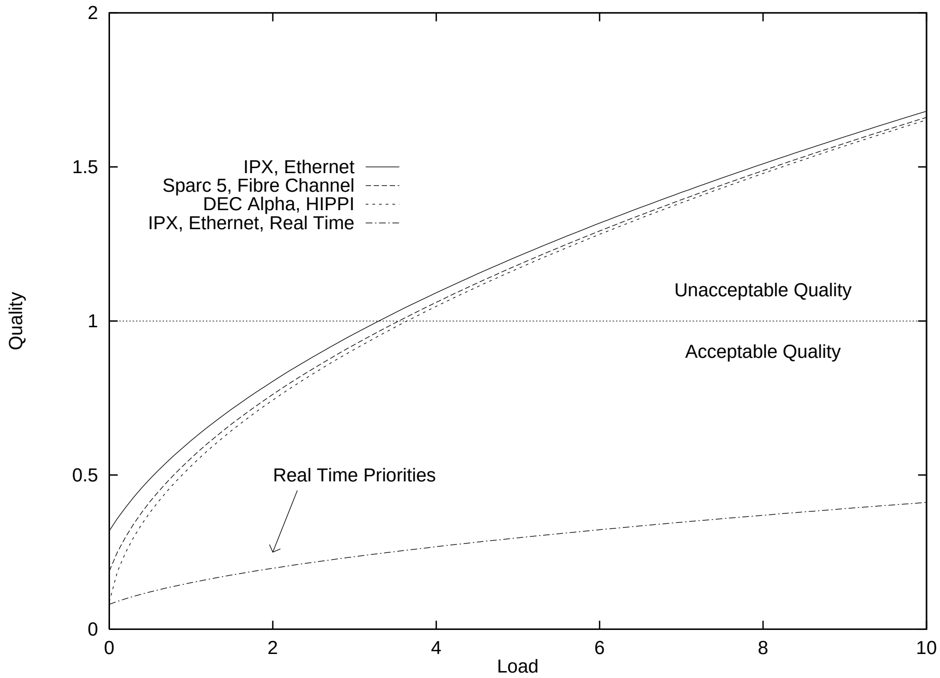

Figure 21: Videoconference Quality versus Load. The horizontal axis is the processor load. The vertical axis is the quality prediction. There are four system configurations depicted. In the first, the processors are Sun IPXs connected by an Ethernet. In the second, the processors are Sun Sparc 5s connected by a Fibre Channel. In the third, the processors are DEC Alphas connected by a HIPPI. In the fourth, the processors are again Sun IPXs connected by an Ethernet, but they are using real-time priorities instead of default priorities. The upper horizontal line marks the limit between acceptable and unacceptable videoconference quality.

9 Conclusions

Most of the work people do is in groups. People communicate best when they can draw pictures and use voice and body language. Computers supporting collaborative work can provide realistic person-to-person communication by using multimedia. Multimedia applications can also provide collaboration in synthetic environments through virtual reality. Today's explosive growth in fast networks and powerful workstations has the potential to support and even enhance group work through multimedia.

Jitter hampers computer support for multimedia. Jitter is the variation in the end-to-end delay for sending data from one user to another. Jitter can cause gaps in the playout of a stream such as in an audioconference, or a choppy appearance to a video display for a videoconference. There are several techniques that can be used to reduce jitter. In this work, we have experimentally measured the effects of three jitter reduction techniques: high-performance processors, real-time priorities and high-speed networks.

We find high-performance processors, real-time priorities and high-speed networks all significantly reduce jitter under conditions of heavy load. Multimedia applications, tending to be resource intensive, are likely to push processors capacities to the limit, making conditions of heavy load likely. As the growth in distributed collaborative applications continues, multimedia applications will push network bandwidths to the limits, also. Thus, high-performance processors, real-time priorities and high-speed networks will all be effective in reducing jitter.

Computer systems continue to get faster while human perceptions remain the same. Future system improvements may remove enough of the underlying jitter such that application-level jitter reduction techniques are unnecessary. How far in the future will this be?

We saw in Section 8.1 that as the area under the jitter compensation curve decreased, the required buffer decreased. From Figure 17, if we had an area of 75,000 square microseconds or less, we would require virtually no buffering in order to achieve acceptable quality. From Figure 18, we can predict that this will happen if jitter is 10^9 square microseconds or less. From our results in Section 4, Section 6 and Section 8.2, we can predict how powerful hardware must become in order to achieve this low jitter rate. We can determine when this will be if we assume both processor power and network bandwidth double each year, as has been the trend in the past [20]. Figure 22 depicts these predictions.

For the next five years, hardware improvements alone will not reduce the effects of jitter enough to eliminate the need for application buffering techniques. However, if we implement real-time priorities in scheduling our multimedia stream, we can reduce jitter enough to eliminate the need for application buffering today.

However, improving jitter alone is not sufficient to guarantee improving application quality. In addition to jitter, the application quality also depends upon *latency*, which decreases the application realism, and *data loss*, which reduces the application resolution. We have developed a model that allows us to evaluate the effects of jitter reduction in the larger context of a user's perception of the multimedia application. Our model allows us to show how advances in networks and processors will improve application quality without tuning the operating system or the application.

As an example, in Section 8 we applied our model to a videoconference, a fundamental multimedia application. Videoconference applications can have from two to hundreds of users, support both audio and video and are often integrated into larger multimedia applications.

In applying our model we found that for some configurations, high-performance processors will improve videoconference quality more than will high-speed networks. However, for some system configurations, the typical Ethernet network is the bottleneck in application quality as the number of users increases. Thus, even high-performance processors will not sufficiently improve application quality when increasing users saturate existing networks. The traditional Ethernet network will become the bottleneck in approximately a year when workstation performance has doubled.

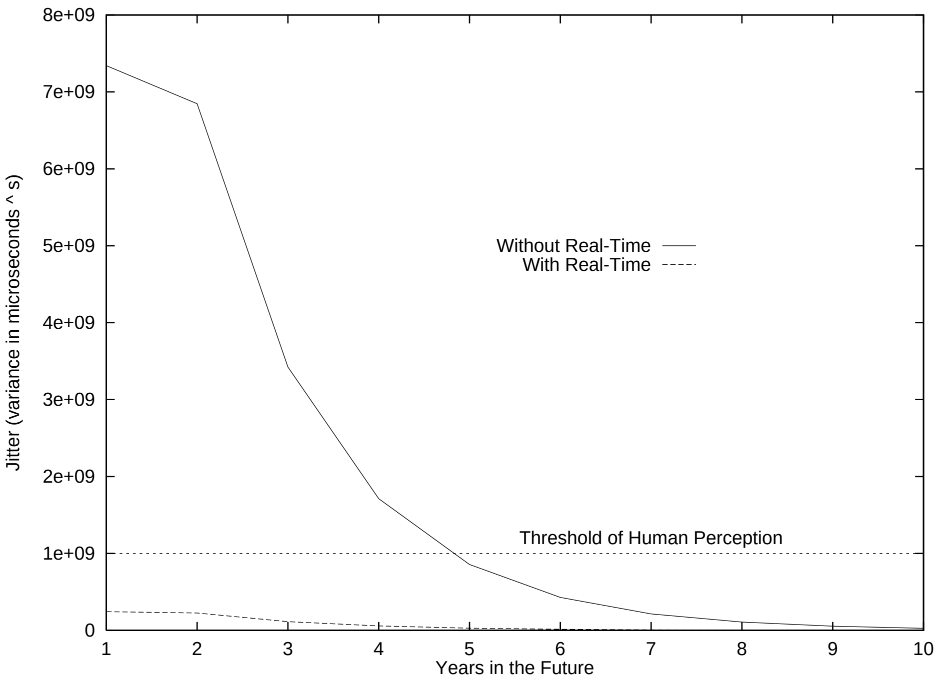

Figure 22: Jitter versus Years. The horizontal axis is the number of years in the future. The vertical axis is the amount of jitter. There are two sets of predictions. The top, slightly curved line depicts the amount of jitter in a system without real-time priorities. The lowest, slightly curved line depicts the amount of jitter in a system with real-time priorities. The horizontal line represent the threshold below which application buffering would not be needed.

When multimedia applications are running under conditions of increasing load, real-time priorities have a greater effect on improving quality than do faster processors and faster networks. In fact, hardware improvements alone will not reduce jitter enough to eliminate the need for application buffering techniques. However, for multimedia on a Local Area Network (LAN), real-time priorities we can reduce jitter enough to eliminate the need for application buffering today. On a Wide Area Network (WAN) especially the Internet, real-time priorities will not be available on all routers, reducing the effectiveness of real-time priorities in reducing. In this case, buffering techniques may still be needed.

10 Future Work

We studied the effects of real-time priorities on jitter when used at both the sender and receiver. Real-time priorities may not significantly reduce jitter when used at only the sender or at only the receiver. If this is true, real-time priorities may only be effective in reducing jitter for a local area network, without an intervening router. On a WAN, especially the Internet, real-time priorities may not be available on all routers, reducing the effectiveness of real-time priorities on the sender and receiver. In this case, buffering techniques will be needed.

Another possible potential jitter reduction technique would be a real-time network protocol. Future work includes experiments to measure the effects of real-time network protocol on jitter. However, network delivery of data within strict jitter bounds does not significantly help the sender and receiver. They must still worry about internal jitter due to queuing in the operating system. Stamping out jitter in the network does not eliminate the need for jitter management code in the hosts.

In our analysis of a videoconference, we assumed a constant bit rate. Video is almost invariably transmitted in a compressed form for the simple reason that compression is so effective. Commercial compression algorithms can achieve 25-to-1 or better reductions in the average number of bits transmitted. However, the amount of data that must be sent for each video frame can vary widely, depending upon the amount changes from frame to frame. Transmission schemes that generate variable amounts of data for each frame are termed *variable bit-rate* (VBR) video schemes, such as MPEG [42]. MPEG codes video using three frame formats. The effects of losses in the MPEG data stream depend upon which type of frame is lost. In the absence of results determining the effects of VBR frame rate losses on quality, we assumed that all frames are of equal importance in determining video quality. Future work might determine the effects on video quality for losses of each type of frame.

References

[1] Sparcstation audio programming. Technical report, Sun Microsystems, 1991.

[2] Ronnie T. Apteker, James A. Fisher, Valentin S. Kisimov, and Hanoch Neishlos. Video acceptability and frame rate. *IEEE Multimedia*, pages 32 – 40, Fall 1995.

[3] Barberis and Pazzaglia. Analysis and optimal design of a packet voice receiver. *IEEE Transactions on Communication*, February 1980.

[4] David R. Boggs, Jeffrey C. Mogul, and Christopher A. Kent. Measured capacity of an Ethernet: Myths and reality. In *Proceedings of the SIGCOMM Conference*, August 1988.

[5] J. Carlis, J. Riedl, A. Georgopoulos, G. Wilcox, R. Elde, J. H. Pardo, K. Ugurbil, E. Retzel, J. Maguire, B. Miller, M. Claypool, T. Brelje, and C. Honda. A zoomable DBMS for brain structure, function and behavior. In *International Conference on Applications of Databases*, June 1994.

[6] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in and integrated services packet network: Architecture and mechanism. *Computer Communication Review*, 22(4), July 1992.

[7] M. Claypool, J. Riedl, J. Carlis, G. Wilcox, R. Elde, E. Retzel, A. Georgopoulos, J. Pardo, K. Ugurbil, B. Miller, and C. Honda. Network requirements for 3D flying in a zoomable brain database. *IEEE JSAC Special Issue on Gigabit Networking*, 13(5), June 1995.

[8] Mark Claypool and John Riedl. Silence is golden? The effects of silence deletion on the CPU load of an audio conference. In *Proceedings of IEEE Multimedia*, Boston, May 1994.

[9] Mark Claypool and John Riedl. A quality planning model for distributed multimedia in the virtual cockpit. In *Proceedings of ACM Multimedia*, November 1996.

[10] The DIS Steering Committee. The DIS vision - a map to the future of distributed interactive simulation. Technical report, Institute for Simulation and Training, May 1994.

[11] Standard Performance Evaluation Corporation. SPEC primer. July 1994. The SPEC primer is frequently posted to the newsgroup comp.benchmarks. SPEC questions can also be sent to spec-ncga@cup.portal.com.

[12] Jay Devore and Roxy Peck. *Statistics – The Exploration and Analysis of Data*. Wadsworth, Inc., second edition edition, 1993.

[13] Spiros Dimolitsas, Franklin L. Corcoran, and John G. Phipps Jr. Impact of transmission delay on ISDN videotelephony. In *Proceedings of Globecom '93 – IEEE Telecommunications Conference*, pages 376 – 379, Houston, TX, November 1993.

[14] Jack J. Dongarra. Performance of various computers using standard linear equations software. Technical Report CS-89-85, University of Tennessee, February 1994. To obtain a postscript copy, send email to netlib@ornl.gov with message body: send performance from benchmark.

[15] Chip Elliot. High-quality multimedia conferencing through a long-haul packet network. In *Proceedings of the First ACM International Conference on Multimedia*, pages 91 – 98, New York, NY, 1993.

[16] Domenico Ferrari. Delay jitter control scheme for packet-switching internetworks. *Computer Communications*, 15(6):367–373, July 1992.

[17] Daniel Frankowski and John Riedl. Hiding jitter in an audio stream. Technical Report Technical Report 93-50, University of Minnesota Department of Computer Science, 1993.

[18] R. Govindan and D. Anderson. Scheduling and IPC mechanisms for continous media. *ACM Operating Systems Review*, 25(5), October 1991.

[19] Joe Haberann and John Riedl. Using real-time priorities to eliminate jitter in a multimedia stream. Technical report, University of Minnesota Department of Computer Science, January 1996.

[20] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, Inc., 1990.

[21] A. Hopper. Pandora – an experimental system for multimedia applications. *ACM Operating Systems Review*, 24(2), April 1990.

[22] Satoru Iai, Takaaki Kurita, and Nobuhiko Kitawaki. Quality requirements for multimedia communcation services and terminals – interaction of speech and video delays. In *Proceedings of Globecom '93 – IEEE Telecommunications Conference*, pages 394 – 398, Houston, TX, November 1993.

[23] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, Inc., 1991.

[24] K. Jeffay, D. Stone, and D. Poirier. YARTOS – kernel support for efficient, predictable real-time systems. In *Joint IEEE Workshop on Real-Time Operating System and Software and IFAC/IFIP Workship on Real-Time Programming*, pages 8 – 13, May 1991.

[25] K. Jeffay, D. Stone, and F. Smith. Kernel support for live digital audio and video. *Computer Communications*, 15(6), 1992.

[26] K. Jeffay, D. Stone, and F. Smith. Transport and display mechanisms for multimedia conferencing across packet-switched networks. *Computer Networks and ISDN Systems*, 26(10):1281 – 1304, July 1994.

[27] K. Jeffay, D. Stone, T. Talley, and F. Smith. Adaptive, best-effort, delivery of audio and video data across packet-switched networks. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1992.

[28] Saimin Jin, Dhadesugoor R. Vaman, and Divyendu Sina. A performance mangement framework to provide bounded packet delay and variance in packet switched networks. *Computer networks and ISDN Sytems*, pages 249 – 264, September 1991.

[29] Rick Jones. *Netperf.* Hewlett-Packard, 1995. The netperf home page can be found at http://www.cup.hp.com/netperf/NetperfPage.html.

[30] S. Khanna, M. Serbree, and J. Zolnowsky. Realtime scheduling in SunOS 5.0. In *Proceedings of the Winter '92 Usenix Conference*, 1992.

[31] Kleinrock and Naylor. Stream traffic communication in packet switched networks: Destination buffering considerations. *IEEE Transactions on Communications*, COM-30(12):2527 – 2534, December 1982.

[32] S. Leffler, M. McKusick, M. Karels, and J. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System.* Addison-Wesley Publishing Company, 1989.

[33] Mengjou Lin, Jenwei Hsieh, David Du, and James MacDonald. Performance of high-speed network I/O subsystems: Case study of a Fibre Channel network. In *Proceedings of Supercomputing '94*, November 1994.

[34] Mengjou Lin, Jenwei Hsieh, David H.C. Du, Joseph P. Thomas, and James A. MacDonald. Distributed network computing over local ATM networks. *ATM LANs: Implementation and Experience with An Emerging Technology*, 1995.

[35] Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Paul T. Barham, and Steven Zeswitz. NPSNET: A network software architecture for large scale virtual environments. *Presence*, 3(4):265 – 287, October 1994.

[36] Vahid Mashayekhi, Janet Drake, Wei-Tek Tsai, and John Riedl. Distributed, collaborative software inspection. *IEEE Software*, 10(5), September 1993.

[37] Michael J. Massimino and Thomas B. Sheridan. Teleoperator performance with varying force and visual feedback. In *Human Factors*, pages 145 – 157, March 1994.

[38] W. Dean McCarty, Steven Sheasby, Philip Amburn, Martin R. Stytz, and Chip Switzer. A virtual cockpit for a distributed interactive simulation. *IEEE Computer Graphics and Applications*, January 1994.

[39] Evi Nemeth, Garth Snyder, and Scott Seebass. *Unix System Adminstration Handbook*. Prentice Hall, 1989.

[40] Judith S. Olson, Gary M. Olson, and David K. Meader. What mix of video and audio is useful for remote real-time work? In *Proceedings of CHI'95 – Proceedings of the Conference in Human Factors in Computing Systems*, pages 362 – 368, 1995.

[41] Craig Partridge. *Gigabit Networking*. Addison-Wesley, 1994.

[42] Ketan Patel, Brian C. Smith, and Lawrence A. Rowe. Performance of a software mpeg video decoder. In *Proceedings of ACM Multimedia*, Anaheim, 1993.

[43] Ramachandran Ramjee, Jim Kurose, Don Towsley and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the 13th Annual Joint Conference of the IEEE Computer and Communications Societies on Networking for Global Communciation. Volume 2*, pages 680–688, Los Alamitos, CA, USA, June 1994. IEEE Computer Society Press.

[44] John Riedl, Vahid Mashayekhi, Jim Schnepf, Mark Claypool, and Dan Frankowski. SuiteSound: A system for distributed collaborative multimedia. *IEEE Transactions on Knowledge and Data Engineering*, August 1993.

[45] Radhika R. Roy. Networking contraints in multimedia conferencing and the role of ATM networks. *AT&T Technical Journal*, July/August 1994.

[46] Henning Schulzrinne. Voice communications across the internet: A network voice terminal. Technical report, University of Massachussetts Department of Electrical Engineering, August 1992.

[47] Michael V. Stein and John T. Riedl. The effects of transport method on the quality of audioconferences with silence deletion. Technical report, University of Minnesota Department of Computer Science, June 1995.

[48] D. Stone and K. Jeffay. An empirical study of delay jitter management policies. *ACM Multimedia Systems*, 2(6):267 – 279, January 1995.

[49] Merryanna Swartz and Daniel Wallace. Effects of frame rate and resolution reduction on human performance. In *Proceedings of IS&T's 46th Annual Conference*, Munich, Germany, 1993.

[50] T. Talley and K. Jeffay. Two-dimensional scaling techniques for adaptive, rate-based transmission control of live audio and video streams. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 247 – 254, October 1994.

[51] Claudio Topolcic. Experimental internet stream protocol, version 2 (ST-II). *RFC 1190*, October 1990.

[52] Dinesh C. Verma, Hui Zhang, and Domenico Ferrari. Delay jitter control for real-time communication in a packet switching network. *IEEE Computer*, pages 35 – 43, 1991.

[53] J.A. Zebarth. Let me be me. In *Proceedings of Globecom '93 – IEEE Telecommunications Conference*, pages 389 – 393, Houston, TX, November 1993.