WPI-CS-TR-98-17 July 1998

Quality Planning for Distributed Collaborative Multimedia Applications

by

Mark Claypool and John Riedl

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

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

Quality Planning for Distributed Collaborative Multimedia Applications

Mark Claypool
CS, Worcester Polytechnic Institute
claypool@cs.wpi.edu

John Riedl
CS, University of Minnesota
riedl@cs.umn.edu

August 12, 1998

Abstract

The tremendous power and low price of today's computer systems have created the opportunity for exciting applications rich with graphics, audio and video. Despite this potential, planning computer systems to support the intensity of these multimedia applications is an extremely difficult task. We have developed a flexible model and method that allows us to predict multimedia application performance from the user's perspective. Our model takes into account the components fundamental to multimedia application quality: latency, jitter and data loss. In applying our method to three specific applications, we have identified some general traits: 1) processors are the bottleneck in performance for many multimedia applications; 2) networks with more bandwidth often do not increase the quality of multimedia applications; and 3) performance for many multimedia applications can be improved greatly by shifting capacity demand from computer system components that are heavily loaded to those that are more lightly loaded.

1 Introduction

Planning is the first fundamental step in developing a software system. Accurate planning is the key to success in building distributed, collaborative multimedia applications that are robust, scalable and meet the needs to today's and tomorrow's users. Unfortunately, planning computer systems to support the intensity of high-quality, collaborative multimedia is an extremely difficult task. Planning for acceptable performance of these applications may require computing capabilities that are perhaps not even available today. Several researchers have looked at capacity planning, the study of computer resources needed to meet expected computer demand, for such applications. However, while capacity planning may help plan growth it is unable to identify whether a user would be satisfied with the quality of the application.

In this paper we address the performance of distributed, collaborative multimedia applications from the user's perspective of distributed, collaborative multimedia applications. We have developed a quality method orthogonal to capacity planning which helps users plan for *acceptable* quality. To apply our planning method, we start from the perspective of the user. The user describes the distributed collaborative multimedia application and defines a set of requirements that need must be met for the application performance to be acceptable. We then simulate the user, the application, the computer system and a measure of application quality. At the heart this method is a flexible model, adjustable to applications with different user requirements and tunable to systems with different system designs and hardware. Our model uses a quality metric as a means of measuring the application performance on a distributed computer system.

In summary, our quality planning is composed of: a *quality metric* to quantify the quality of an application from the user's perspective; a *quality model* to do quality planning for users; *micro experiments* to measure the fundamental components of the application; *macro experiments* to test the accuracy of the analytic model based on the micro experiments; a *calibration mechanism* to tune the quality model; and a *prediction method* to predict the application quality as the various components of the model change.

In this paper we present:

- a detailed description of our quality metric
- the general method of applying our quality metric
- application of our method to audioconferences
- highlights of the application of our method to two other applications
- trends in distributed collaborative multimedia application performance

The rest of this paper is organized as follows: Section 2 presents our metric for measuring multimedia quality; Section 3 describes the methods and models we use to apply our metric; Section 4 details the application of our method to audioconferences; Section 5 presents the highlights and summary of results of quality planning for two additional applications; and Section 6 summarizes our conclusions.

2 Multimedia Quality

> *"There is an old network saying: 'Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed – you can't bribe God.' "* David Clark, MIT

One indication of the performance of an entire computer system is the users' opinions on the *multimedia quality* of the applications they run. Multimedia quality is a measure of the performance of a multimedia application based on the requirements expected by the user. 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 video frame on the screen. The quantity and timing of these events give us measures that affect application quality. Based on previous multimedia application research [SW93, RS94, MS94, NK82, AFKN95], we use three measures to determine quality for most distributed multimedia applications:

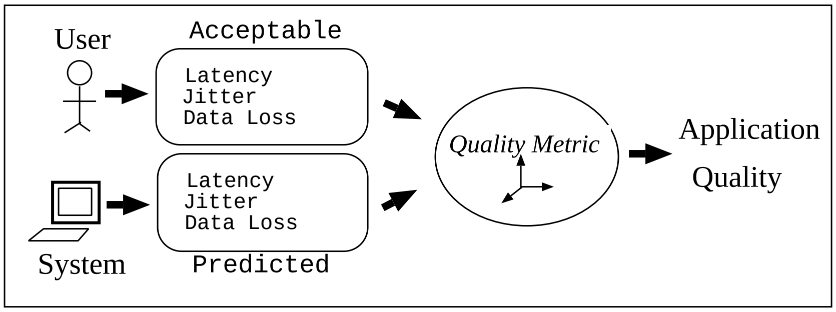


Figure 1: The Process for Computing Application Quality. The user defines the acceptable latency, jitter and data loss and the system determines the predicted values. Based on the acceptable values specified in the user requirements, a quality metric computes the application quality from the predicted values.

- *Latency*, the time it takes information to move from the server through the client to the user
- *Jitter*, the variation in latency, can cause gaps in the playout of a stream such as in an audioconference, or a choppy appearance to a video display
- *Data Loss* which can take many forms such as reduced bits of color, pixel groups, smaller images, dropped frames and lossy compression.

Ideally, we would prefer that 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 1. 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 that is compliant with the mathematical definitions of a metric. We further define a *multimedia quality metric* as having several other important properties:

1. It incorporates the three fundamental multimedia quality components: latency, jitter and data loss.
2. It treats the fundamental components equally, which seems appropriate in the absence of user studies to the contrary.
3. It produces a convex region of acceptable quality. This fits our intuition about changes in quality: the measure increases total quality with any increase in quality along one axis. 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.

To form our quality metric, we build upon the work of Naylor and Kleinrock [NK82]. Naylor and Kleinrock developed a model for measuring the quality of an audioconference based on the probability of playout gaps and end-to-end delay. The quality of the audioconference was computed by taking the normalized distance of the audioconference's delay and gaps from the origin in the

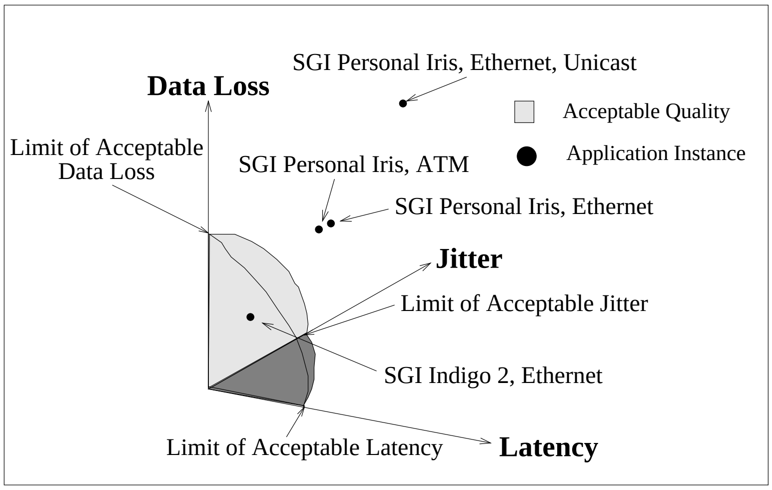


Figure 2: Multimedia Application Quality Space. The user defines the acceptable latency, jitter and data loss. These values determine a region of acceptable application quality, depicted by the shaded region. All points inside the shaded region have acceptable quality, while those outside the region do not. An instantiation of the application and the underlying computer system lies at one point in this space. Four application configuration instantiations are shown.

delay-gap plane. We extend this model by using latency, jitter and data loss as axes, creating a multi-dimensional quality space. We place the best quality value for each axis at the origin and normalize 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. The location of the point is determined by our predictions of the amount of latency, jitter an data loss that would occur with the given system configuration. In order to satisfy the mathematical properties of a metric, 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 2 depicts a 3-d quality space for multimedia applications. The user requirements determine a region of acceptable application quality, depicted by the shaded region. Each instantiation of the application and the underlying computer system is a 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. However, the rest of our model is independent of the quality metric chosen. If new metrics are developed and validated with user testing, they can be used in place of our quality metric.

3 Methods

In order to use the quality metric presented in Section 2, we must predict the amount of latency, jitter and data loss for the distributed, collaborative multimedia application being studied. We have developed a method that enables us to verify the accuracy of our model and predict quality bottlenecks as various model components change. Our predictions are based on a detailed model

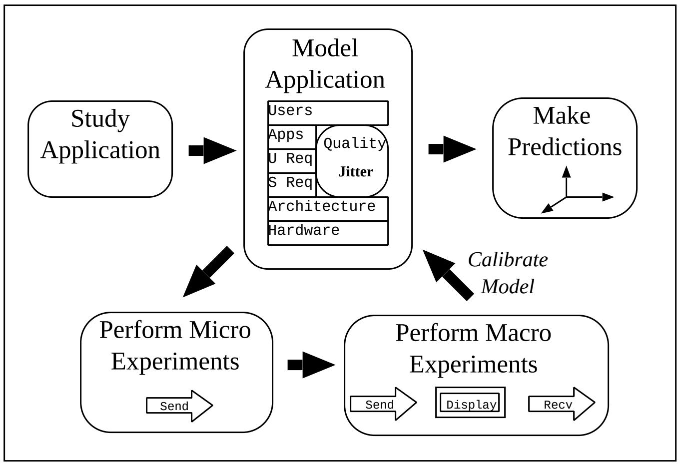


Figure 3: Quality Planning Method and Model. We have developed a method for applying our model to distributed multimedia applications. We start with an application, develop our model, perform micro and macro experiments and make quality predictions.

of the user, application and computer system. In this section, we present our method and model, depicted in Figure 3.

Study Application Our method begins by studying the application to obtain information on the users and their requirements. The application is founded on a set of user requirements that need to be fulfilled for the application to be effective for the user. The user requirements include information such as frame rate and frame size, acceptable latency and jitter and tolerance of data loss.

Model Application We use the information about the users and their requirements in our model. Our model for the quality of a distributed multimedia application incorporates: *Users*: the users of the application are those we used during the "Study Application" phase of our method as described above; *Applications:* the applications are the software programs the users will run; *User Requirements:* the user requirements are the user's interface to our model. The requirements they specify may drive the selection of the underlying system in order to make the application acceptable for the user; *System Requirements:* the user requirements impose a series of requirements on the system. Some of these include network bandwidth, disk throughput and processor power; *Architecture:* architecture is the structure of the distributed program which determines the location of data and the distribution of the processing; *Hardware:* given the system requirements and architecture, the hardware needed to support the application can be determined; *Quality:* the variations in hardware, architecture, system requirements, user requirements and the application all effect the application quality as perceived by the user.

As a brief example to better illustrate how we might use our model, suppose we wish to predict the performance of a proposed voice mail system that will allow a group of software engineers browse their archived voice-mail [BFJ+96]. We first determine the quality of the audio required by the users, either by user testing or by an analogy to similar applications. The audio quality determines the user requirements. The system requirements are derived from the user requirements, with key system components used to examine tradeoffs. For example, we might vary the number of users, the amount of compression or the network protocol. We choose an architecture and hardware on which to analyze the system. For example, we might pick Sun Sparc 5 workstations connected via a 10-baseT Ethernet cable. As described in Section 2, we build a quality model based on the user requirements. The system requirements, architecture and hardware are all used in the quality model to determine if the proposed configuration is acceptable to the users. We can then iterate by modifying the component parameters and determining a new application quality.

Perform Micro Experiments Experiments that measure performance of the fundamental processor components of an application we call *micro experiments*. We do micro experiments to allow us to predict the effects of systems on applications built with those components. Some fundamental components for many multimedia applications include: *Record* data from the microphone or video codec; *Play* data to the speakers; *Render* a frame to be displayed; *Display* a frame to the screen; *Read* data from a disk; *Write* data to a disk; *Compress* data; *Decompress* data; *Send* a data packet to a client; and *Receive* a data packet from a server.

After carefully measuring the processor load of each component, we can predict the processor load of an application built with those components. Changes in application configuration or changes in hardware are represented by modifying the individual components and observing how that affects performance.

Perform Macro Experiments Experiments that measure performance of applications built with micro experiment components we call *macro experiments*. We do macro experiments to test the accuracy of micro experiment-based predictions of application performance. For example, assume we have a two-person audioconference that lasts for three minutes. Each component of the audioconference (record, send, receive and play) processes the three minutes of audioconference data. We predict the total processor load from our micro experiment measurements of the record, send, receive and play loads. In addition, we predict the network load based on the audio data rate of the workstations. In our macro experiments, we run a two-person audioconference and carefully measure the processor and network load. We then compare these measured values to the predicted values in an attempt to test the accuracy of our prediction methods.

Make Predictions By modifying the fundamental application components, we can predict performance on alternate system configurations. This allows us to evaluate the potential performance benefits from expensive high-performance workstations and high-speed networks before installing them. Moreover, we can investigate possible performance benefits from networks and workstations that have not yet been built. Our approach for evaluation of each alternative system is the same: we modify the parameters of our performance model to fit the new system, then evaluate the resulting model to obtain performance predictions. These analyses are intended to provide a sense of the relative merits of the various alternatives, rather than present absolute measures of their performance.

Our micro and macro experiments are done on only a handful of platforms. However, we would like our predictions to be accurate for untested platforms, and even future, as yet unbuilt hardware. In order to attempt these extrapolations we rely on research in benchmarks that compare the performance among systems and alternate system configurations. In particular, we rely upon SPEC benchmarks results to predict the performance of application components on untested workstations [spe]. We rely upon landmark studies in network and disk performance to predict performance on alternate networks [BMK88, LHD+95, LHDM94, RO94, SH80].

4 A Detailed Example: Audioconferences

In this section, we present a detailed example of applying our quality planning method to audioconferences.

Study Application Audioconferences have been a popular topic for multimedia research on the Internet, especially over the MBone. We chose to study a small group, peer-to-peer audioconference with voice-quality sound with each user on a separate workstation.

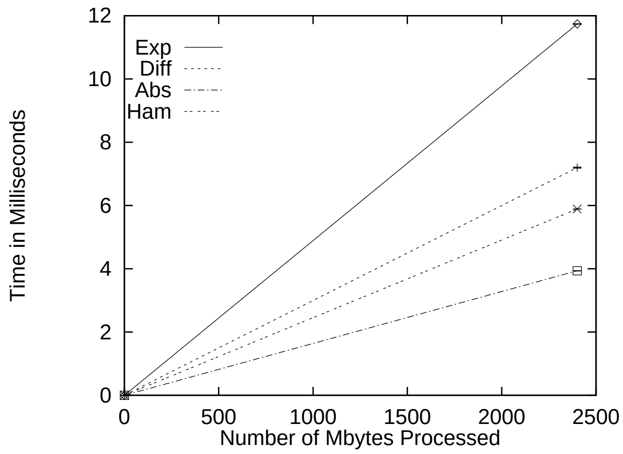
Model Our model of an audioconference is based on the components of recording, silence deletion, sending, receiving, mixing and writing. *Recording* is the processor load for taking the digitized sound samples from the audio device. *Silence deletion* is the processor load for applying one of the deletion algorithms to the recorded sample. [1] *Sending* is the processor load for packetizing the sample and sending it to all other stations. *Receiving* is the processor load for processing all incoming packetized samples. *Mixing* is the processor load for combining sound packets that arrive simultaneously. *Writing* is the processor load for delivering the incoming samples to the audio device.

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

We use a process that increments a `long integer` variable in a tight loop to measure the processor load for the individual components: Record, Deletion, Send, Receive, Mix and Play. To obtain a baseline for our counter, we run the counter process on a quiet machine. This gives the processor potential for the machine. We then run the counter process with each component in the model. The difference in the bare count and the component count is the component-induced load. In [CR93], we verify that the counter process does indeed accurately report loads of processor-bound processes with which it runs concurrently.

Figure 4 [Left] shows the line equations obtained from the counter measurements for different silence deletion algorithms on the IPX. We have similar graphs for the SLC and for other components of the model [CR93], but to avoid redundancy we do not present them here.

[1] Silence deletion removes silent parts from speech. Experiments have shown that silence deletion substantially reduces network load [RMS+93].



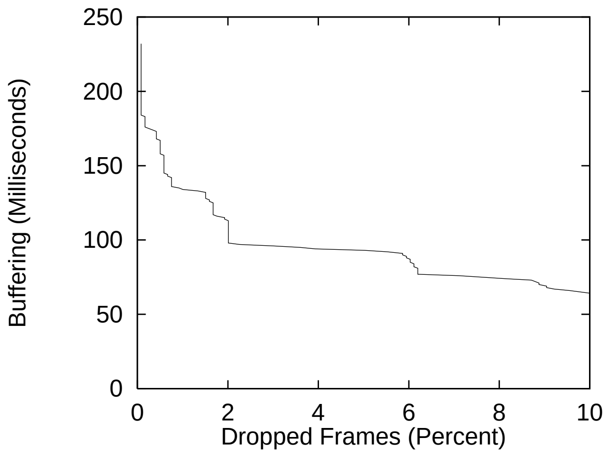


Figure 4:
[Left] Processor Time for Deletion Algorithms on the Sun IPX. The four deletion algorithms are shown for their time to process 300 seconds worth of sound. All points are shown with 95% confidence intervals.

[Right] 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.

Table 1 shows the values for the line equations for each of the audioconference components for each machine type.

The per-packet and per-byte terms above pertain to the equations: `Load(component) = per-packet + per-byte * bytes`. The equations are the processor costs for each component of an audioconference from which we can project the cost of a complete audioconference.

Macro Experiments In order to test the accuracy of our model in predicting audioconference processor loads, we measured the performance of a simple audioconferencer, *Speak*. Speak is two person, uses UDP, can employ any of the five deletion algorithms (Absolute, Differential, Exponen-

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

Table 1: Values for Sun SLC and Sun IPX Line Fits for audioconference components. Units are in milliseconds.

tial, Ham or None), and has little extra user-interface overhead.

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

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

The complete results are given in [CR93], but for brevity, we summarize the results here. In most cases, the predicted values are within 10% of the actual values. We therefore consider the predicted results to be significant only if the differences are larger than 10%.

4.1 Predictions

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

4.1.1 The Region of Acceptable Audioconference Quality

To determine the region of acceptable audioconference quality, we need to define acceptable limits for audioconferences along each of the latency, jitter and data loss axes. According to [FM76], fewer than 6% gaps in an audio stream playout and 230 milliseconds or less of delay resulted in acceptable audio quality. Audioconference quality is then the Euclidean distance from the origin to a point represented by delay milliseconds normalized over 230 and the percentage of audio gaps normalized over 6%. Any quality value under 1 is considered acceptable.

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, 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. 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 [NK82]. 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.

Figure 4 [Right] depicts the tradeoff between dropped frames and buffering as a result of jitter. We generated the graph by first recording a trace of audio frame 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 4 [Right], 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 6-Kbyte 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.

4.1.2 Determining Jitter

Our previous experiments measuring the effectiveness of several jitter reduction techniques give us the relationship between load and jitter for faster processors and networks [CHR97]. We use these results as the basis for determining the jitter in the audioconference under various system configurations.

4.1.3 Determining 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. In our micro experiments, we measured the latency from all the micro experiments. 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 audio frame; performing silence deletion; sending the audio frame to the the other users; receiving the audioframe from the other users; buffering in the jitter compensation curve; and playing the audio frame to the speakers.

4.1.4 Determining 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 audioconference, we assume data loss only in the form of dropped frames, when an application chooses to discard late frames in order to keep playout latency low and constant, or when either the network or the processor do not have sufficient capacity to transmit data at the required frame rate.

4.1.5 Determining Quality

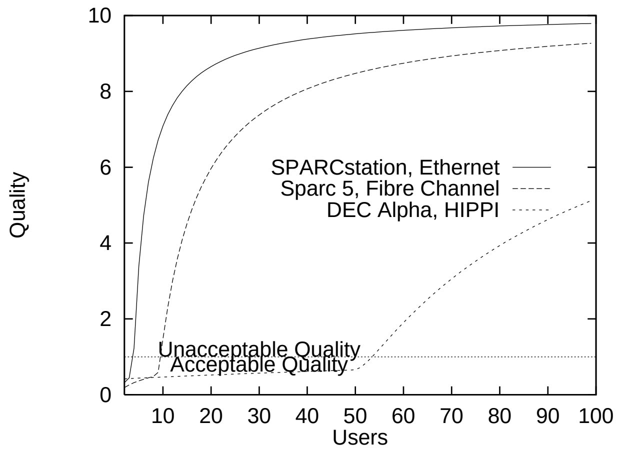
We can now use our metric to explore audioconference quality under different system configurations. We can quantify how effectively today's computer systems support multi-person audioconferences. We can determine when today's systems will fail due to too many users. We can evaluate the benefits of expensive 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 determine audioconference quality for two scenarios: 1) increasing users; and 2) high-performance processors and high-speed networks. For all of our audioconference quality predictions we assume multicast routing and audio hardware for capturing and displaying frames.

Users Tomorrow's processor improvements promise to support more and more users. But how many more? How do more and more simultaneous audioconference users affect application quality? Figure 5 [Left] depicts the predicted effects of increasing users on audioconference quality. We predict audioconference quality for three different audioconference 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). However, workstations such as Sun Sparc 5s connected by fast networks such as a Fibre Channel can support up to 10 users. Very high-performance workstations such as DEC Alphas connected by a high-speed network such as a HIPPI can support over 50 users.

High-Performance Processors and High-Speed Networks Our previous experimental results showed that both high-performance processors and high-speed networks reduce jitter [CHR97]. However, which reduces jitter more? And more importantly, which improves application quality more?

We assume we have five audioconference participants. In the previous analysize, we use our model to evaluate quality for a variable number of users, but here we evaluate a likely audioconference 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. We use the Standard Performance Evaluation Corporation (SPEC) benchmarks to make predictions about quality on more powerful workstations [spe]. Figure 5 [Right] shows these predictions. For five users, increasing the processor power to a SPEC Cint95 of 3 or greater results in acceptable audioconference quality. At no time does increasing the network bandwidth result in an acceptable quality. In this scenario, we conclude that processor power influences audioconference quality more than does network bandwidth.



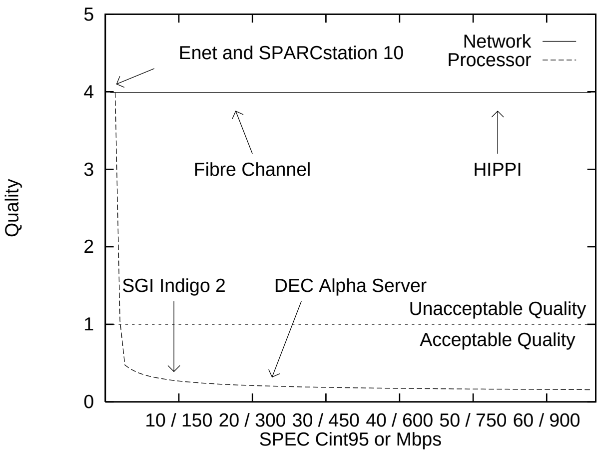


Figure 5: Audioconference Quality.

[Left] Increasing 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 SPARCstation 10 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 audioconference quality.

[Right] Processor or Network Increase. The horizontal axis is the SPEC Cint95 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 SPARCstation (SPEC Cint95 = 1.5), 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 audioconference quality.

5 Highlights of Applications

In Subsections 5.1 and 5.2, we present a few select details from the application of our method and model to two emerging multimedia applications: a "flying" interface to a zoomable database, and a flight simulator for combat training called the Virtual Cockpit.

5.1 Flying through the Zoomable Database

Neuroscientists from diverse disciplines plan to collaborate across distances in exploring various aspects of brain structure [CRG+94]. Their design includes a zoomable multimedia database of images of the brain tissue. High-resolution magnetic resonance imaging (MRI) shows 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 3-d 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 to allow neuroscientists to communicate effectively.

Figure 6 [Left] depicts our quality predictions for multi-processor flying clients. The individual points are all SGI Indigo 2 clients with a different number of processors. The curve represents an acceptable level of quality; all points inside the curve will have acceptable quality while points outside will not. In this figure, we have not assumed any specialized flying hardware. We assume that the servers will be able to provide the bandwidth requested by all the clients to simplify the computation. At least an 8-processor client is required in order to have acceptable flying quality.

Figure 6 [Right] depicts quality versus the number of clients for both flying with compression and flying without compression. Clients are assumed to be 20 processor Indigo 2's without specialized hardware. The server is assumed to be an SGI Indigo 2 workstation with specialized hardware (see [CRC+95] for more information). The arrows indicate points at which the server can no longer keep up with the bandwidth requests by the clients. At this point, application performance decreases, as depicted by the increasing quality values. For fewer than 4 clients, compression decreases client-side quality, mostly because of the latency increase from the clients decompressing the images. However, for 5 or more clients, compression increases application quality because the server can meet the bandwidth requirements of more clients.

5.2 The Virtual Cockpit

The Virtual Cockpit is a flight simulator prototype built by the Air Force Institute of Technology and designed to integrated into Distributed Interactive Simulation (DIS) [MSA+94]. DIS applications are designed to enable soldiers to engage in simulated combat [Com94]. 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. 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

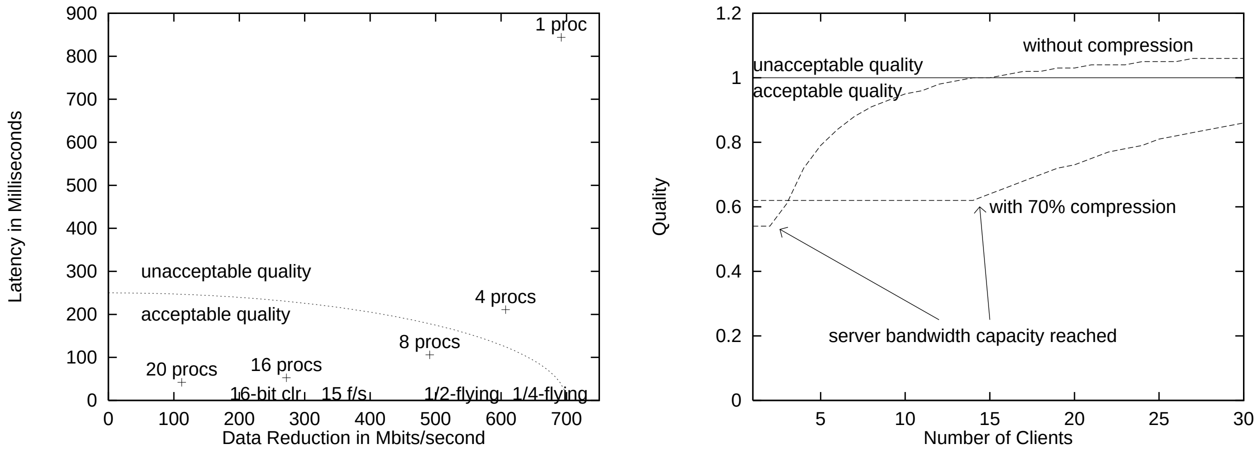


Figure 6: Flying Quality.

[Left] Multi-processor Clients. The horizontal axis is the number of Mbits/second of data reduction received by the client. The vertical axis is the latency added by the client. The points are SGI Indigo 2's clients with different numbers of processors. The curve represents an acceptable level of quality; all points inside the curve will have acceptable quality while points outside will not. Note that the clients are not equipped with any special flying hardware.

[Right] Effects of Compression. Clients are 100 processor SGI Indigo 2 workstations with no specialized hardware. The server is an SGI Indigo 2 workstation with specialized hardware. The arrows indicate points at which the server can no longer keep up with the bandwidth requested by the clients. At this point, application quality gets worse as depicted by the increasing quality values.

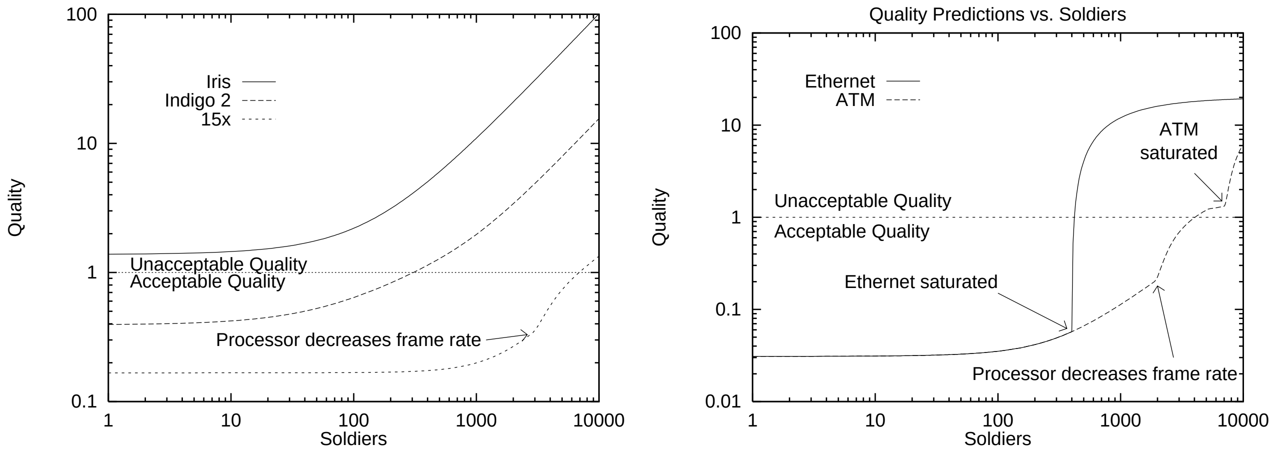


Figure 7: Virtual Cockpit Quality versus Soldiers.

[Left] High-Speed Networks. Virtual Cockpit Quality versus Soldiers. The two curves represent the quality predictions for an Ethernet and an ATM network. The horizontal line represents the acceptable quality limit. Both the horizontal and vertical axes are in log scale.

[Right] High-Performance Processors. The three curves represent the quality predictions for three different processors. The top curve is an SGI Personal Iris. The second curve is an SGI Indigo 2. The bottom curve is a processor 15 times more powerful than the Indigo 2. The horizontal line represents the acceptable quality limit. Both the horizontal and vertical axes are in log scale.

such as DIS applications will stress all parts of a computer system.

Our work in [CR96] showed that low-end SGI Personal Iris workstations do not provide acceptable Virtual Cockpit quality for any number of users. Is there further benefit from higher-performance processors? We assume the network has sufficient bandwidth to handle all necessary updates in order to minimize the effects of the network. We compare the quality of the Virtual Cockpit with SGI Personal Irises and SGI Indigo 2s to the quality of the Virtual Cockpit with processors 15 times more powerful than the Indigo 2.[2] Figure 7 [Left] shows the quality predictions for the Virtual Cockpit with different processors. The top curve is an SGI Personal Iris. The second curve is an SGI Indigo 2. The bottom curve is a processor 15 times more powerful than the Indigo 2. The horizontal line represents the acceptable quality limit. The "knee" in the curve for the 15x processor is where the processor decreases the frame rate in order to handle the updates from the other soldiers. High-performance processors are crucial for acceptable Virtual Cockpit quality. SGI Personal Iris' are unable to deliver acceptable application quality. More powerful SGI Indigo 2s can deliver acceptable application quality for up to 500 soldiers. 15x's provides better application quality than Indigo 2s and can deliver acceptable application quality for up to 7000 soldiers.

With the Virtual Cockpit running on a processor 15 times more powerful than the SGI Indigo 2, a T1 network will become saturated while supporting just 100's of soldiers. How much quality benefit will then be gained from a high-speed network? We compare the quality of the Virtual Cockpit with an Ethernet to that of the Virtual Cockpit with an ATM network. The ATM network

[2]Processor performance has approximately doubled every year for the last 5-10 years. If this trend continues, the 15x processor will come along in about 8 years.

transmits the update packets faster (155 Mbits/second versus 10 Mbits/second for an Ethernet). Past work has found jitter and missed updates in the ATM network are the same as jitter and missed updates in the Ethernet [HR96]. We assume jitter and missed updates remain the same in high-speed networks.

Figure 7 [Right] shows the quality predictions for the Virtual Cockpit with different networks. The top curve is the quality predictions for an Ethernet. The lower curve is the quality predictions for an ATM. The steep increase in the Ethernet curve occurs when the Ethernet becomes saturated. At this point, the Virtual Cockpit begins to increasingly miss updates. The first bend in the ATM curve occurs when the processor must decrease the frame rate in order to process all updates. The second bend in the ATM curve occurs when the ATM becomes saturated. High-speed networks are unimportant for the Virtual Cockpit quality until existing networks reach saturation. The quality prediction curves for the Ethernet and the ATM are indistinguishable until the Ethernet becomes saturated. At this point, the ATM network greatly increases scalability.

6 Conclusions

Despite the real and potential benefits of multimedia, there are several obstacles that need to be overcome in designing multimedia applications and systems. Multimedia and multi-user applications are more resource intensive than traditional text-based, single-user applications. In addition, multimedia applications have different performance requirements than do text-based applications. Text-based applications are sensitive to latency and loss, while multimedia applications are sensitive to latency and jitter. The bottlenecks to text-based application performance might lie in those components that induce latency, while the bottlenecks to multimedia applications might lie in the those components that induce the jitter. New techniques must be developed to identify bottlenecks in multimedia application performance.

We have developed a quality planning method for distributed collaborative multimedia applications that allows us to investigate potential bottlenecks in application quality. At the heart of our method is a model that allows us to predict the application performance from the user's perspective. Our model takes into account the components fundamental to multimedia applications: latency, jitter and data loss. Our model allows us to investigate application bottlenecks by being adjustable to: the number of users; new hardware and architectures; alternate quality metrics; and different applications.

Using our model, we can explore the performance tradeoffs for a variety of multimedia applications as the underlying computers systems change. In Section 4, we have shown a detailed example of how our model can be applied to audioconferences and in Section 5, a summary of the results to Flying in a zoomable database [CRC+95], and the Virtual Cockpit [CR96]. There are three general results common to all the applications we studied:

1. Processors are the bottleneck in performance for many multimedia applications. Audioconferences, Flying and the Virtual Cockpit all saw a dramatic increase in application performance with an increase in processor power.
2. Networks with more bandwidth often do not increase the quality of multimedia applications.

For Audioconferences, a faster network did very little to improve application quality. In Flying, more network bandwidth did increase the performance for one user, but once that user's requirements were met, there was little benefit from more bandwidth. For the Virtual Cockpit, more bandwidth did not noticeably affect application quality at all, but it did allow more simultaneous users to train for combat when existing networks became saturated. Networks with more bandwidth do not benefit few-person multimedia applications but serve only to increase the scalability of the applications by allowing more simultaneous users.

3. Application capacity requirements are not equally distributed across computer systems. Performance for many multimedia applications can be greatly improved by shifting capacity demand from computer system components that are heavily loaded to those that are more lightly loaded. Shifting capacity demand is *crucial* as the number of application users increases. For audioconferences, silence deletion transfered load from the network to the processor. While this decreased application quality for two audioconference users, it greatly increased application quality for three or more users. For Flying, application performance was totally unacceptable unless capacity demand was shifted from the processor to specialized hardware. For the Virtual Cockpit dead reckoning shifted capacity demand dramatically from the network, enabling current networks to support the tens of thousands of soldiers required for effective combat training.

Our objective in identifying application bottlenecks is to understand the system limits that will prevent applications from meeting users' needs. After identifying each bottleneck, we explore ways to reduce the effect of the bottleneck through improving system resources. We then explore the new bottlenecks that arise in the enhanced system. Our analysis at each stage is likely to overstate system performance, because we assume maximum possible performance of each system component. However, the bottlenecks we identify are likely to be bottlenecks in practice, and the design principles suggested by the analysis should ameliorate these bottlenecks in practice.

To conclude, the major contributions of this paper are:

- A multimedia quality metric that provides a quantitative means to measure multimedia application performance from the users perspective.
- A model and method that uses our multimedia quality metric and enables the prediction of application performance and evaluation of system design tradeoffs.
- Detailed performance predictions for three distributed collaborative multimedia applications: a Audioconference, a "flying" interface to a 3D scientific database and a collaborative flight simulator called the Virtual Cockpit.
- The effects of system improvements on the performance of these multimedia applications.

References

[AFKN95] Ronnie T. Apteker, James A. Fisher, Valentin S. Kisimov, and Hanoch Neishlos. Video acceptability and frame rate. *IEEE Multimedia*, pages 32 – 40, Fall 1995.

[BFJ+96] M. Borwn, J. Foote, G. Jones, K. Sparck Jones, and S. Young. Open-vocabulary speech indexing for voice and video mail retrieval. In *Proceedings of the Fourth ACM International Multimedia Conference*, Boston, MA, November 1996. ACM.

[BMK88] 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.

[CHR97] Mark Claypool, Joe Habermann, and John Riedl. The effects of high-performance processors, real-time priorities and high-speed networks on jitter in a multimedia stream. Technical Report TR-97-023, University of Minnesota Department of Computer Science, June 1997.

[Com94] 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.

[CR93] Mark Claypool and John Riedl. Silence is golden? The effects of silence deletion on the CPU load of an audio conference. Technical Report TR-93-81, University of Minnesota Department of Computer Science, 1993.

[CR96] Mark Claypool and John Riedl. A quality planning model for distributed multimedia in the virtual cockpit. In *Proceedings of ACM Multimedia*, pages 253 – 264, November 1996.

[CRC+95] 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.

[CRG+94] 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.

[FM76] J.W. Forgie and C.W. McElwain. Some comments on NSC note 78 'effects of lost packets on speech intelligibility.'. Technical Report Network Speech Compression Note 92, M.I.T., Lincoln Laboratory, March 1976.

[HR96] Joe Habermann 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.

[LHD+95] 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.

[LHDM94] 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.

[MS94] Michael J. Massimino and Thomas B. Sheridan. Teleoperator performance with varying force and visual feedback. In *Human Factors*, pages 145 – 157, March 1994.

[MSA+94] 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.

[NK82] William E. Naylor and Leonard Kleinrock. Stream traffic communication in packet switched networks: Destination buffering considerations. *IEEE Transactions on Communications*, COM-30(12):2527 – 2534, December 1982.

[RMS+93] 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.

[RO94] Thomas M. Ruwart and Matthew T. O'Keefe. Performance characteristics of a 100 megabyte/second disk array. In *Proceedings of the Storage and Interfaces '94*, Santa Clara, CA, January 1994.

[RS94] 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, 1994. IEEE Computer Society Press.

[SH80] John F. Shoch and Jon A. Hupp. Measured performance of an Ethernet local network. *Communications of the ACM*, 23(12):711–720, December 1980.

[spe] *SPEC– The Standard Performance Evaluation Corporation.*
Internet site (`http://www.spec.org/`).

[SW93] 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.