An Empirical Study of Delay Jitter Management Policies*

Donald L. Stone   *Kevin Jeffay*
University of North Carolina at Chapel Hill
Department of Computer Science
Chapel Hill, NC   27599-3175 USA
{*stone*, *jeffay*}@*cs.unc.edu*

July 1994

Abstract: This paper presents an empirical study of several policies for managing the effect of delay jitter on the playout of audio and video in computer-based conferences. The problem addressed is that of managing the fundamental tradeoff between display with low latency and display with few gaps. We describe a particular policy called *queue monitoring* which observes delay jitter over time and dynamically adjusts display latency in order to support low-latency conferences with an acceptable gap rate. Queue monitoring is evaluated by comparing it with two policies from the literature in a study based on measurements from a computer-based conferencing system. Our results show that queue monitoring performs as well or better than the other policies over the range of observed network loads. More importantly, we show that queue monitoring performs better on those network loads for which the other policies exhibit poor performance.

1. Introduction

This work concerns network and operating system support for computer-based video conferencing. Our goal is to support high-performance conferences using today's networks based on asynchronous communications (*e.g.*, ethernet, token ring, FDDI, T3, *etc*.). In this environment, transmission times vary and hence audio and video data may occasionally arrive at their destination after the time at which they should have been displayed. When this occurs, a "gap" appears in the playout that can adversely affect users' perception of the conference. The problem we consider here is that of ameliorating the effect of variable network transmission delays in order to support good quality audio and video playout. In particular, we are interested in supporting small internetworks consisting of several physical networks connected via bridges and routers (*e.g.*, a building or campus-area network). Support for such networks will be necessary if, as we believe, conventional LANs continue to be widely used in the "last mile" to the desktop.

For the audio and video hardware considered in our work, data units called *frames* are acquired from an audio or video source at a precise rate (*e.g.*, one frame every 33 ms. for

---

* This work supported by the National Science Foundation (grant numbers CCR-9110938 and ICI-9015443), and by the IBM and Intel Corporations.

In: *Multimedia Systems*, Volume 2, Number 6, (January 1995) pages 267-279.

video), and must be delivered to a conference participant's display at the same precise rate. Prior to delivery, each frame must be digitized, compressed, transmitted over the network, and decompressed as shown in Figure 1. Once decompressed, a frame is buffered in a *display queue* until it can be displayed.

For each conference receiver, the *display latency* of a frame is defined as the total time from acquisition at the sender to display at the receiver. For purposes of discussion, it is convenient to divide display latency into two components: the total time from acquisition to decompression, called the *end-to-end delay* of a frame, and the time a frame is buffered between decompression and display, called the *display queuing delay*. If we assume that the time required to digitize, compress, or decompress frames is not constant, or if we assume that the delays incurred transmitting frames across the network can vary, then the end-to-end delays experienced by frames will vary. Variance in end-to-end delay is called *delay jitter*.

Ideally, frames should be displayed continuously with each successive frame displayed immediately after its predecessor. Since frames are generated and displayed at fixed intervals, continuous playout can occur only if each frame is played with the same display latency. Thus, if frames are to be played continuously, jitter in the end-to-end delay of each frame must be compensated for by varying the display queuing delay of the frame.

However, continuous playout is not always possible. For example, consider a case where a frame incurs a particularly long end-to-end delay. As a result, the frame may not be ready to be displayed at the time when the display of the preceding frame is complete. In such a case, there will necessarily be a *gap* in the playout (*e.g.*, for a video stream, this would mean that no video frame is displayed for a 33 ms. interval). More precisely, a gap will occur whenever a frame arrives with an end-to-end delay greater than the display latency of the previous frame.

In general, the lower the display latency, the higher the probability of encountering an

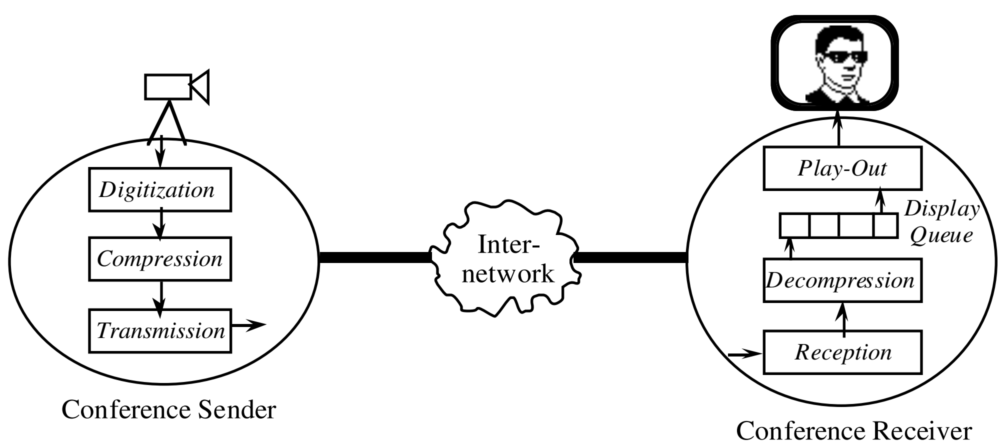


Figure 1: System overview.

end-to-end delay sufficient to cause a gap. Thus there is a fundamental tradeoff between display latency and gap frequency. A conferencing system must explicitly manage this tradeoff so as to provide good quality playout to the user.

One approach to managing this tradeoff is to prevent delay jitter completely by using a network that provides transmission with constant delay. This is the approach used in computer-based conferencing systems based on networks that provide isochronous services (such as ATM). Another approach is to minimize delay jitter by reserving resources in the network so as to provide guaranteed bounds on delay and delay jitter. For example, Ferrari [1] presents a scheme in which clients requiring real-time communication services specify their traffic characteristics and performance requirements. In response, the system reserves sufficient resources at each node on the network (*e.g.*, processor capacity, buffer space) to guarantee the performance requirements of the client.

Unfortunately, isochronous delivery and resource reservation are not available in the network environments we wish to support. More importantly, even if the network supported transmission with constant delay, jitter in the end-to-end delay can occur as a result of variations in the time required to digitize, compress, and decompress frames, as well as variations in the time required to communicate data between the audio/video subsystem and the network transport system (all of which can be affected by operating system scheduling decisions). For each of these reasons, we assume that delay jitter is a fundamental phenomenon and therefore the tradeoff between display latency and gap frequency must be explicitly managed in any conferencing system.

In this paper, we evaluate three policies for managing the tradeoff between display latency and gap frequency. Two policies, the I-policy and the E-policy, are taken from the literature; the third, *queue monitoring*, is a new policy that we have developed. The performance of these policies is evaluated with an empirical study based on an experimental computer-based video conferencing system (described in [3]). In the study, traces of the end-to-end delays experienced by frames are recorded during the execution of the conferencing system under a variety of (real) network loads. Each trace is used as input to a simulator which determines the display latency and gap rate that would result from applying each policy to a conference with the corresponding sequence of end-to-end delays. The simulation results demonstrate that queue monitoring always performs at least as well, and often performs better than either the I-policy or the E-policy over the range of observed network conditions.

The following section discusses the basic principles of managing the tradeoff between display latency and gap frequency, and describes the I- and E-policies. The queue monitoring policy is presented in Section 3. Section 4 discusses the metrics we propose for evaluating delay jitter management policies and the problems inherent in formulating such metrics. Section 5 describes the experiments and presents the results of the trace-driven simulations. We conclude in Section 6 with a summary of our findings.

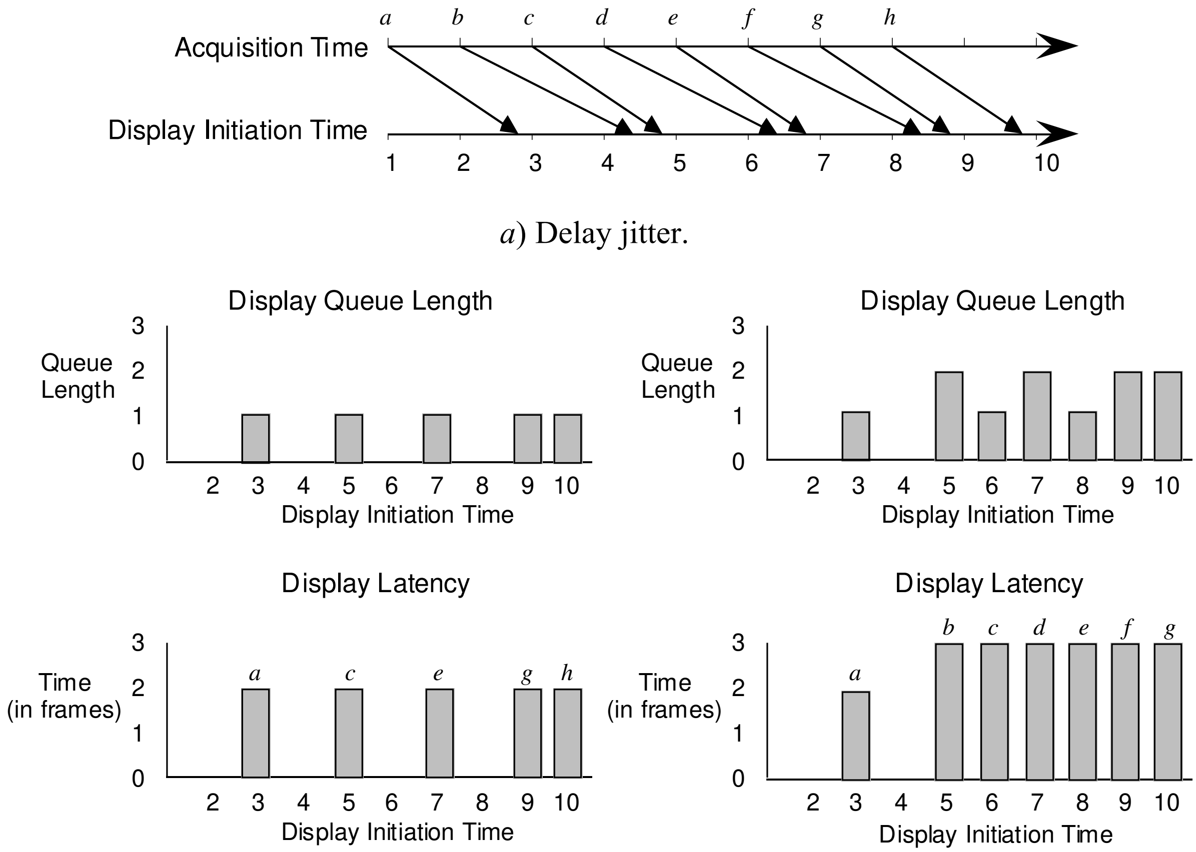
2. The Effect of Delay Jitter

In order to sustain continuous playout without gaps, every frame must be played with a fixed display latency that is greater than the worst-case end-to-end delay that will be encountered during a conference. It is not clear however that our primary goal should be playout with no gaps. Display latency is only one important factor in determining the perceived quality of the playout [2]. It is likely that in many conferencing applications, as long as gaps occur infrequently, playout with low latency and some gaps will be preferable to playout with high latency and no gaps. Therefore, if we always play frames with a display latency greater than the worst-case delay and if the worst-case delay is rarely observed in practice, then we will display most frames with latency higher than necessary to support good quality playout.

If we are willing to accept gaps in the playout, then we can choose to play frames with a display latency less than the worst-case end-to-end delay. However, we must now address the issue of what should be done with a frame that arrives after the time at which it should have been displayed. There are two choices: either the frame can be discarded or it can be displayed. These choices define two delay jitter management policies which Naylor and Kleinrock call the *I-Policy* and the *E-Policy* [6]. Under the I-policy, frames are displayed with a single fixed display latency which is a parameter of the policy, and each frame that arrives with an end-to-end delay greater than this latency is discarded. Under the E-policy, late frames are displayed at the next opportunity. This causes the display of all frames after the late frame to be delayed and thus the E-policy has the effect of increasing the display latency of all subsequent frames.

Figure 2 illustrates the behavior of the I- and E-policies in response to late frames. Figure 2a shows the acquisition and display times for eight frames. Tick marks on the upper timeline indicate *acquisition times*, the times at which new frames are acquired. Tick marks on the lower timeline indicate *display initiation times*, the times at which the new frames are displayed. Each diagonal arrow represents the end-to-end delay of an individual frame, extending from the time at which it was acquired to the time it is placed in the display queue. Throughout these examples, the acquisition time of a frame (*i.e.*, points *a*, *b*, *c*, *etc*.) is used to refer to individual frames.

Figure 2b shows the effect of executing the I-Policy on the pattern of acquisition times, display initiation times, and end-to-end delays shown in Figure 2a. In this example, the display latency parameter of the I-Policy is 2 frame times. (For simplicity in the examples, time is represented as multiples of the time taken to acquire or display a frame). The top graph in Figure 2b shows the display queue length at each display initiation time. The bottom graph shows the display latency of the frame being displayed at each display initiation time. In addition, each latency bar is labeled with the acquisition time of the frame that is displayed at that display initiation time. In this example, frames *b*, *d*, and *f* arrive with end-to-end delays longer than two frame times and are discarded. Thus, use of the I-policy results in three gaps occurring in the playout at display initiation times 4,6, and 8.



a) Delay jitter.

b) I-policy.

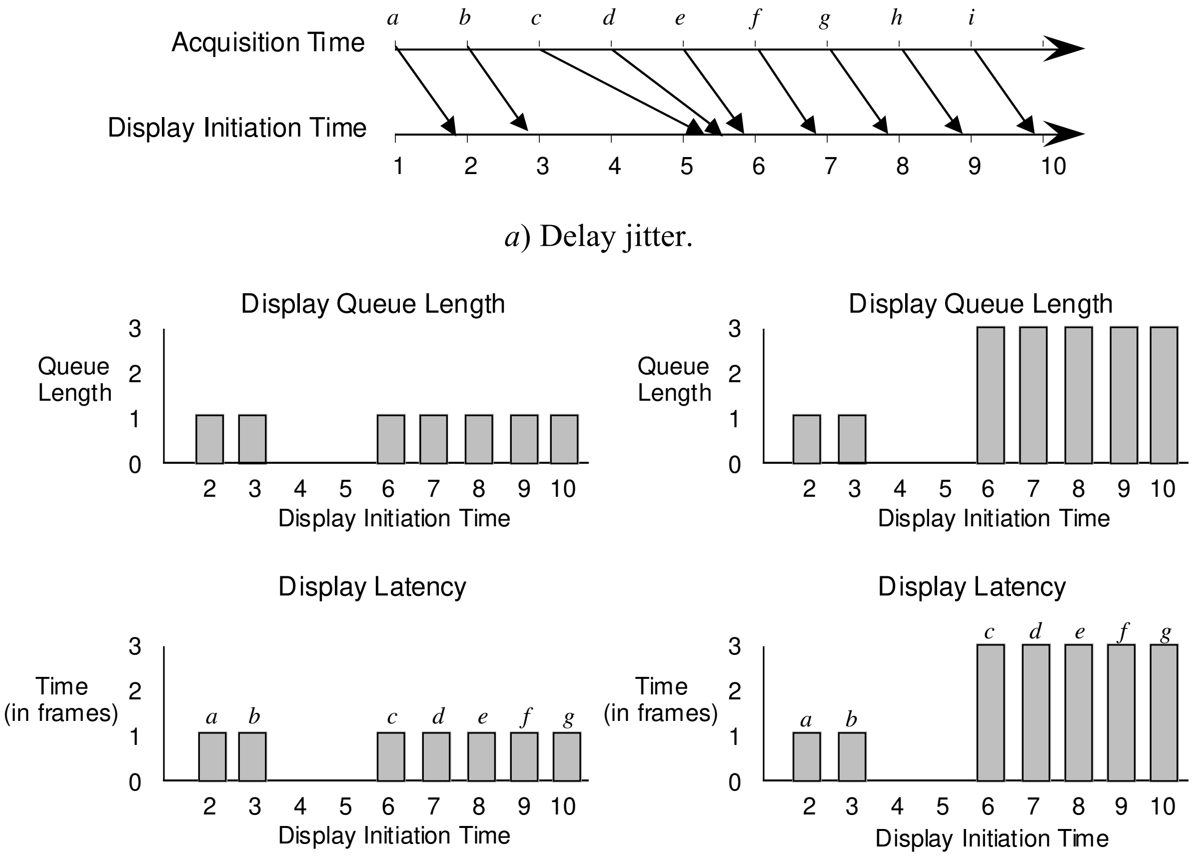
c) E-policy.

Figure 2: Behavior of I-policy and E-policy.

Figure 2c shows the effect of executing the E-policy. As with the I-policy, the late arrival of frame *b* causes a gap at display initiation time 4. However, after frame *b* arrives, it is placed in the display queue and eventually played at display initiation time 5 with a latency of 3 frame times. This delay results in each frame after *b* also being played with a display latency of 3 frame times. Because 3 frame times is longer than the end-to-end delay experienced by any frame after *b*, there are no gaps after display initiation time 4.

The example shown in Figure 2 illustrates an advantage of the E-policy. The E-policy starts playing frames with the lowest possible initial display latency and then adjusts display latency upward in response to delay jitter. The overall effect of the E-policy is to find a display latency which is sufficient to play frames without gaps by dynamically adjusting the latency to be higher than any end-to-end delay yet observed.

Figure 3 illustrates a situation in which the I-policy performs better than the E-policy. In this example, all frames except frames *c* and *d* arrive with an end-to-end delay of less than a frame time and experience negligible delay jitter. Frames *c* and *d* arrive late because of some temporary increase in network activity. Each policy results in gaps at display initiation times 4 and 5, however, the I-policy with a display latency of one frame time



a) Delay jitter.

b) I-policy.

c) E-policy.

Figure 3: Behavior of the I-policy and E-policy after a "burst."

plays the frames after the gap with low latency while the E-policy plays the frames after the gap with high latency. Under the E-policy, a single "burst" of activity on the network that causes a few frames to arrive late results in a permanent increase in display latency.

In each policy, we are choosing a display latency at which to play frames, either explicitly in the case of the I-policy, or implicitly in the case of the E-policy. A good choice for display latency will depend on many factors. First, the acceptable rate of gaps and the acceptable display latency may vary depending on the application (*e.g.*, speech and music may have different gap and latency requirements) and the current requirements of the user (*e.g.*, a surgeon viewing an operation will have different requirements than participants in a televised lecture). Second, the display latency required to maintain an acceptable gap-rate will depend on the expected level of delay jitter, which will vary as a result of congestion both at participants' machines and on the network. The dynamic nature of these factors motivates the design of a delay jitter management policy which dynamically chooses an appropriate display latency so as to adapt to new requirements and conditions.

3. Queue Monitoring

The problem of choosing a display latency has been extensively studied in the context of packet audio systems (systems for the transmission of audio streams across packet-switched networks). In many applications, audio is modeled as a sequence of "talkspurts" (some period of time in which audio data must be acquired, transmitted, and played) separated by "silent periods" (some period of time in which there is no significant audio activity, so audio need not be acquired or played). Naylor and Kleinrock proposed that a display latency be chosen at the beginning of each talkspurt by observing the transmission delays of the last m audio fragments, discarding the k largest delays, and choosing the greatest remaining delay. For their particular model of audio quality, they stated a rule of thumb for choosing m and k ($m > 40$ and $k = .07*m$) which usually resulted in good quality audio. More recent work on practical solutions for the Internet (as exemplified by Nevot [7]), has also used a scheme by which a display latency is chosen at the beginning of each talkspurt on the basis of observed delay jitter.

More generally, consider an oracle that has perfect knowledge of the end-to-end delays of future frames and hence can choose the best display latency at which to play each frame. Such an oracle can adjust display latency in response to changes in delay jitter (perhaps due to changes in network congestion) in order to achieve the best possible balance between display latency and gaps. Display latency can be adjusted upward by artificially introducing a gap (*i.e.*, delaying the playout of the next frame) and can be adjusted downward by discarding frames.

If we assume that the delay jitter in the near future can be predicted by observing the delay jitter in the recent past, we can construct delay jitter management policies that are approximations to the oracle. (This is analogous to the working set concept of page replacement in virtual memory management.) The Naylor and Kleinrock policy for choosing a display latency is one example of such a policy. This policy, however, is difficult to implement as measuring end-to-end delays at runtime requires synchronized clocks.

Instead of measuring end-to-end delays, we can directly measure the impact of delay jitter at a receiver by observing the length of the display queue over time. Once every frame time, a frame is removed from the display queue to be played. For example, for video frames displayed at 30 frames per second, a frame is removed every 33 ms. Since frames are also acquired and transmitted once per frame time, *on average* one frame will arrive and be placed in the display queue and one frame will be removed from the display queue during each frame time. If end-to-end delays are constant, the queue length measured at each display initiation time should be constant (as it was between display initiation times 7 and 10 in the example shown in Figure 3c). If end-to-end delay increases sufficiently, it may happen that no frame arrives during the playout of a frame. In that case, the length of the display queue will decrease by 1 frame (*e.g.*, display initiation time 6 in Figure 2c). If end-to-end delays decrease sufficiently, more than one frame may arrive during the playout of a frame, and the length of the display queue will increase (*e.g.*, display initiation time 6 in Figure 3c).

Over time, the length of the display queue will vary depending on the range of end-to-end delays encountered by frames. If we assume the level of delay jitter in the near future will be the same as the level in the recent past, then while end-to-end delays may vary, they will not vary outside the range that we have observed recently. This implies that the length of the display queue will remain at least as long as the minimum length that has been observed in the recent past. Furthermore, as long as the display queue contains at least one frame at each display initiation time, there will be no gaps in the playout. Thus, if the minimum display queue length observed recently was at least two frames, we can discard a frame to reduce display latency without causing a gap.

A policy for decreasing display latency based on observing queue lengths has been used to govern the behavior of the audio display queue in the Pandora system [5]. Whenever frames are added to the display queue (called the *clawback buffer*), the length of the queue is checked against a target value. In the Pandora system, the target is 2 frames (which corresponds to 4 ms. of audio data). If the length of the display queue is greater than this target for a sufficiently long interval (8 seconds in the Pandora system), incoming audio frames are discarded. Because this has the effect of shortening the display queue, audio data that arrives after this time will be played with a lower display latency.

We propose a display queue management policy called *queue monitoring* that is a variation of the policy used in Pandora. For each possible display queue length we define a threshold value. The threshold value for queue length n specifies a duration (measured in frame times) after which, if the display queue has continuously contained more than n frames, we will conclude that display latency can be reduced without increasing the frequency of gaps. Since we expect that large variations in end-to-end delay will occur infrequently, and small variations will occur much more frequently, threshold values for long queue lengths specify short durations while those for short queue lengths specify long durations. This policy has the effect of reducing display latency quickly after long delays due to large bursts of network traffic, but approaching minimal latency slowly.

To implement queue monitoring, we associate an array of counters and threshold values with the display queue. Each time we wish to display a new frame, we first perform a "thresholding operation." If the display queue has length m, then counters 2 through $m - 1$ are incremented and all other counters are reset. If any counter exceeds its associated threshold value, all the counters are reset and the oldest frame is discarded from the display queue. The oldest remaining frame is then displayed.

An important principle in this implementation is that the thresholding operation will never discard frames unless the display queue contains more than two frames. The last frame in the display queue should never be discarded because there must be a frame available for display after the thresholding operation completes. Similarly, if the second-to-last frame in the display queue were discarded, then even minimal delay jitter could potentially cause a gap. For example, this can occur in a situation where a frame arrives immediately before the thresholding operation and results in a display queue with length 2. If one of those frames is discarded and the other is displayed, then the queue would be empty. Then, if the next frame has a slightly larger end-to-end delay, it may not arrive in time to be displayed. Therefore, in the following discussion, thresholds will only be defined for queue lengths greater than two.

4. Evaluating Delay Jitter Management Policies

The remainder of the paper is a study comparing the effectiveness of the queue monitoring policy to that of the I- and the E-policy. We are hampered in this task by the lack of metrics for comparing the performance of delay jitter management policies. Clearly, if policy A results in lower display latency and less gaps than policy B, it is performing better. However, if policy A results in a lower display latency and a higher gap rate as compared with policy B, which has performed better? The answer will depend on a number of factors including the display latency, the gap rate, the resolution of the display, and the user's particular audio and video requirements. It may also depend on the distribution of gaps throughout the measurement interval, the number of display latency changes, and the distribution of periods of high and low display latency throughout the interval. A proper standard for comparing policies would take each of these factors into account in making a judgment.

Unfortunately, we are not aware of such a standard. Therefore we have adopted a simple and arbitrary comparison rule for the analysis in this paper. The performance of a policy is evaluated along two dimensions: average display latency and average gap rate. We assume that only differences in display latency of more than 15 ms. and differences in gap rate of more than 1 gap per minute are significant. Under these rules policy A is declared to have done better than policy B if it is better in one dimension and the same or better in the other dimension. Two policies are declared to have done equally well if they are the same in both dimensions and are declared to be incomparable if each has done better in one dimension.

Given this comparison rule, we can evaluate and compare the effectiveness of policies for a particular execution of the system. However, it is still difficult to compare results of multiple executions. One fundamental difficulty arises because the video hardware that acquires frames at the sender is not synchronized with the display at the receiver. To illustrate the effect this has on display latency, assume there is no end-to-end delay (*i.e.*, acquisition and arrival of frames are simultaneous). Despite this fact, we must wait until the next display initiation time to display each new frame. Depending on the synchronization difference between the video hardware acquiring the frames and the display, each frame may have to wait up to one frame time before being displayed. This synchronization time is a random variable and varies between executions. Therefore, when comparing results of multiple executions, differences in latency of as much as one frame time are not significant.

The second difficulty in comparing multiple executions arises from our working definition of the I-policy. Ideally, the I-policy should enforce a particular display latency. However, this would require that the clocks at the acquisition and display workstations be synchronized. In our work we only assume synchronized clocks for measurement purposes (*i.e.*, we do not use synchronized clocks to guide the execution of the system), so we cannot completely implement such a policy. Instead, we use a variant of the I-policy which buffers the first frame for a fixed number of frame times before displaying it and then displays all subsequent frames with the same display latency. The effect of this definition is to make the display latency enforced by a particular I-policy in an execution, a function of the end-to-end delay of the first frame that is received (*i.e.*, a random variable).

The goal of the study presented in this paper is to determine which of several policies results in the best quality playout. Because of the difficulties involved in comparing the results of multiple executions, and because our comparison rule determines relative, rather than absolute performance, we restrict direct comparisons to determining the relative performance of two policies on single conference executions. This allows us to conclude only that one policy outperforms another on a particular execution. To show that one policy outperforms another in general, we must show that it performs better on some executions and as well or better on all executions. This method of pairwise comparison is the basis of the performance evaluations presented in the next section.

5. The Study

In this section, we present the results of a study that we have performed to gauge the effectiveness of the queue monitoring policy in a building-sized internetwork. In this study, we compared the performance of the queue monitoring policy with the performance of the I- and E-policies on video conferences transmitted over the main network supporting our department. The experiments presented were performed using the audio portion of an experimental video conferencing system that we have developed. In each experiment, we have recorded a trace of the end-to-end delay experienced by each frame. These traces are then used as input to a simulator which determines the average display latency and average gap rate that would result from applying each policy to a conference with the corresponding sequence of end-to-end delays. The results are compared using the comparison rule defined in Section 4.

The video conferencing system used for the experiments runs on IBM PS/2 workstations. These workstations run a real-time operating system and network transport software specially tailored for real-time communications. Network packets use the UDP/IP format. Acquisition and display of audio and video data is performed using IBM-Intel ActionMedia 750 hardware at a rate of 30 video frames per second (256x240 resolution and 8-bit color pixels; each compressed video frame occupies around 6000-8000 bytes) and 60 audio frames per second (a 2-channel audio stream generated at a bit rate of 128 Kb per second and packaged into 16.5 ms "frames"). The full system has been extensively described elsewhere [3,4].

The building network consists of several 10 Mb Ethernets and 16 Mb token rings interconnect by bridges and routers. It supports approximately 400 UNIX workstations and Macintosh personal computers. The workstations share a common filesystem using a mix of NFS and AFS. The application mix running on these workstations should be typical of most academic computer science departments. We are linked to the Internet through the campus internetwork which also includes a number of other departmental networks (generally smaller than the one just described) connected via a central broadband network. In all, this environment should be typical of those generally found in the "last mile" to the desktop.

In each experiment, we acquired, transmitted, and displayed audio and video for a series of 10 minute intervals. Audio frames were transmitted in individual packets and video frames were broken into 1350 byte fragments. Each packet was routed across a lightly loaded private token ring to a gateway, through a segment of the departmental ethernet to a bridge, through a second segment of the departmental ethernet to another gateway, and back across the private token ring to the display machine.

Twenty-four runs of the system were performed over the course of a typical day (between 6 am and 5 pm) covering lightly and heavily loaded periods. Four additional runs were performed between midnight and 1 am. For each run, we recorded a trace of the acquisition time and the arrival time of each audio frame. In addition, we recorded each display initiation time (*i.e.*, the times at which new frames were displayed). Before each run, we ran a protocol to measure the difference in clock times between the acquisition and display machines. After each run, the recorded times were adjusted to account for the difference in clock times and to account for clock drift (measured in a separate experiment). Finally, the traces were used as input for a trace-driven simulation of the display-side of a conference. For the given sequence of arrivals and display initiation times, the simulator determined the time at which each frame would be displayed under each of the three policies. The output of the simulator was the average display latency and average gap rate that would result from using each policy on the run.

Figure 4 gives some basic data on the 28 runs. "Time of Day" is the time the run was initiated. Average and maximum delays are calculated from the end-to-end delays experienced by audio frames. Lost and duplicate frames are counts of lost and duplicated packets which contained an audio frame. No out of order packets were observed.

Figure 5 provides a more detailed look at two runs, one with low delay jitter and one with high delay jitter. These figures are histograms of the end-to-end delays experienced by audio frames in runs 2 and 24. The y-axis shows a count of the number of frames with end-to-end delays within each 5 ms. interval (*e.g.*, a count of frames with an end-to-end delay of 30-35 ms.). Note that the y-axis is plotted on a log scale.

Figure 6 shows the simulation results for each of the 28 runs. For each, we simulated the queue monitoring policy with a threshold of 120 frame times at all queue lengths. The effect of these threshold settings is to reduce display latency by one audio frame time (*i.e.*, 16.5 ms.) whenever the display queue contains more than 2 audio frames for 120 continuous frame times (*i.e.*, 2 seconds). We also simulated the E-policy and the (variant) I-policy with 2 and 3 frame times of display latency. In the table, these policies are labeled QM-120, E, I-2, and I-3 respectively. For each policy, the table shows the resulting average display latency (in ms.) and the average gap rate (in gaps per minute).

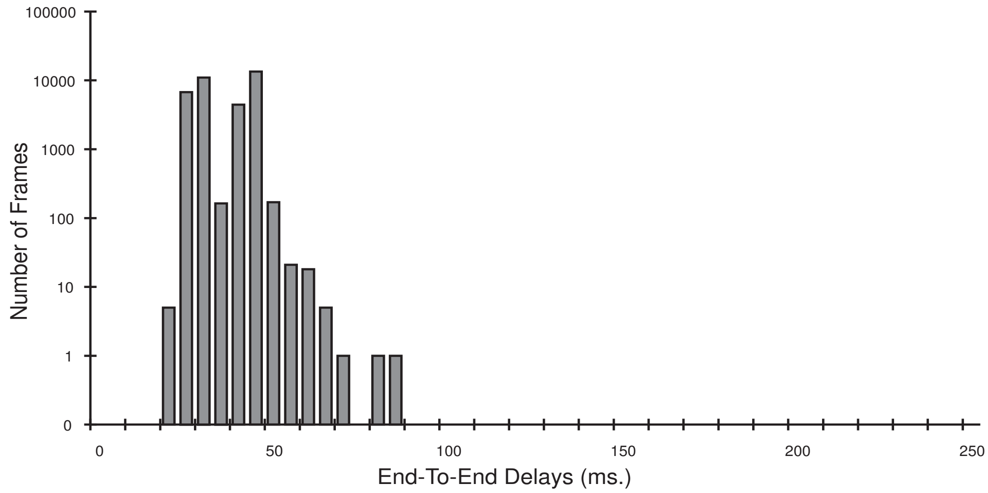
Run	Time of Day	Avg. Delay ms.	Max. Delay ms.	Lost Frames	Duplicate Frames
1	06:03	38	76	1	0
2	06:25	38	88	3	0
3	06:36	37	171	5	0
4	06:47	37	105	1	0
5	08:03	38	115	1	0
6	08:14	37	73	2	0
7	08:25	38	184	7	0
8	08:36	39	157	1	0
9	10:02	41	186	23	0
10	10:16	40	124	4	0
11	10:31	41	213	7	0
12	10:49	40	140	6	0
13	11:57	39	110	5	0
14	12:08	41	138	5	0
15	12:19	41	133	3	0
16	12:34	40	187	11	0
17	14:02	41	189	11	0
18	14:13	42	141	3	0
19	14:42	39	107	4	0
20	14:54	40	131	12	0
21	16:01	39	171	9	0
22	16:21	39	128	2	0
23	16:33	39	86	2	1
24	16:55	42	242	14	1
25	00:05	38	80	4	0
26	00:16	38	128	0	0
27	00:27	38	134	8	0
28	00:38	38	83	2	0

Figure 4: All runs, basic data.

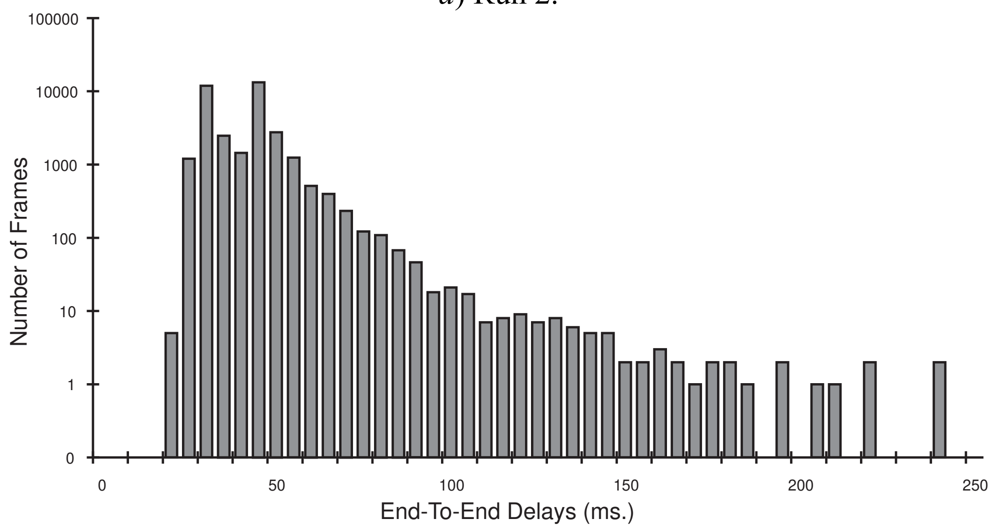
For each run, the rightmost columns show the comparison between the queue monitoring policy and the other policies (using the comparison rule defined in Section 0). A '+' indicates that queue monitoring did better, a '0' means the two were equivalent, a '-' means that queue monitoring did worse, and a blank space means the two policies were incomparable.

Figure 6 shows that the QM-120 policy performed as well or better than the I-2 policy for every run except run 11. For that run, QM-120 is incomparable with I-2 because it has a somewhat lower latency and a much higher gap rate. In particular, the gap rate produced by the I-2 policy is extremely high (approximately 1 gap every 2 seconds, compared with about 1 gap every 12 seconds for the QM-120 policy). From these results, we can conclude that in general, the QM-120 policy is more effective than the I-2 policy.

The QM-120 policy also performed as well or better than the I-3 policy for every run except run 15. On that run, QM-120 policy is judged to perform worse than I-3 because



a) Run 2.



b) Run 24.

Figure 5: Distribution of end-to-end delays.

the difference between the display latency produced by the QM-120 policy and that produced by the I-3 policy (13 ms.) is not significant (according to the comparison rule), while the difference in gap rates (1.2 gaps per minute) is significant. Thus, we conclude that, in general, the QM-120 policy is more effective than the I-3 policy. Furthermore, QM-120 always results in a display latency lower than that produced by I-3. Therefore, all I-policies with display latencies larger than 3 frame times cannot perform better than the QM-120 policy (by our comparison rule). However, because gap rates for the I-policies with larger display latency will decrease, these policies may be incomparable with QM-120. Nevertheless, the QM-120 policy resulted in relatively low gap rates for every run.

Overall then, we conclude that if our goal in managing delay jitter is to minimize display latency with an acceptable gap frequency, then the QM-120 policy outperforms all I-policies. For other goals, we make no conclusions.

Run	I-Policy 2 (I-2) Latency ms.	I-Policy 2 (I-2) Gaps /min.	I-Policy 3 (I-3) Latency ms.	I-Policy 3 (I-3) Gaps /min.	E-Policy Latency ms.	E-Policy Gaps /min.	QM (QM-120) Latency ms.	QM (QM-120) Gaps /min.	QM vs. I2	QM vs. I3	QM vs. E
1	80	0.1	97	0.1	75	0.2	66	0.3	0	+	0
2	75	0.5	91	0.3	72	0.5	66	0.6	0	+	0
3	69	3.6	86	2.8	140	0.9	68	1.4	+	+	+
4	65	0.7	82	0.4	104	0.6	65	0.6	0	+	+
5	71	0.6	88	0.4	93	0.5	68	0.5	0	+	+
6	70	0.3	86	0.2	76	0.4	70	0.5	0	+	0
7	73	2.9	90	1.6	106	1.2	72	1.9	+	+	+
8	62	5.1	79	2.4	106	0.9	75	1.3	+	+	+
9	81	23.0	98	12.6	118	2.8	87	7.6	+	+	
10	70	14.6	87	3.6	113	0.8	78	3.9	+	0	
11	66	25.2	83	6.9	133	1.4	83	4.8		+	
12	71	9.6	87	3.4	114	0.9	76	2.7	+	0	
13	67	9.6	84	2.8	96	0.8	72	2.1	+	0	
14	72	15.1	88	3.9	101	1.1	80	3.9	+	0	
15	76	4.4	92	1.7	117	0.9	79	2.9	+	-	
16	68	18.6	85	8.0	114	1.8	80	6.6	+	+	
17	77	22.0	93	12.1	146	1.8	88	7.5	+	+	
18	76	13.0	93	4.1	131	0.7	85	4.8	+	0	
19	66	5.0	82	1.3	87	0.9	72	1.8	+	0	+
20	73	11.3	90	4.1	98	1.6	77	3.7	+	0	
21	70	12.8	87	6.1	159	1.5	76	3.6	+	+	
22	79	1.4	95	0.5	100	0.6	77	1.0	0	+	+
23	75	0.4	91	0.2	84	0.4	74	0.6	0	+	0
24	77	39.6	93	15.0	104	1.8	87	5.2	+	+	
25	65	0.8	81	0.4	66	0.7	65	0.7	0	+	0
26	64	5.4	81	0.9	122	0.6	69	1.5	+	0	+
27	70	7.8	87	3.8	107	1.3	73	3.5	+	0	
28	76	0.3	93	0.2	75	0.3	73	0.4	0	+	0
							QM Better		18	18	8
							QM Equivalent		9	9	6
							QM Worse		0	1	0
							Incomparable		1	0	14

Figure 6: Results for all runs.

With respect to the E-policy, the QM-120 policy performs as well or better on 14 runs and is incomparable on 14 runs. The runs in which QM-120 performs as well or better tend to be the runs with low delay jitter (*i.e.*, the early morning and late evening runs), while the incomparable runs are those with high delay jitter. In every run, QM-120 resulted in smaller display latency and a somewhat higher gap rate (5.2 gaps per minute is the largest difference). So again we conclude that if the goal in managing delay jitter is to minimize display latency with an acceptable gap frequency, then the QM-120 policy outperforms the E-policy. For other goals, we make no conclusions.

In the second part of the study, we investigated the effect of varying the threshold parameter on the effectiveness of the queue monitoring policy. For each of the 28 runs, we simulated the queue monitoring policy with a single threshold defined for all queue

Run	QM (30) Latency ms.	QM (30) Gaps /min.	QM (60) Latency ms.	QM (60) Gaps /min.	QM (120) Latency ms.	QM (120) Gaps /min.	QM (600) Latency ms.	QM (600) Gaps /min.	QM (3600) Latency ms.	QM (3600) Gaps /min.	120 vs. 30	120 vs. 60	120 vs. 600	120 vs. 3600
1	64	0.3	65	0.3	66	0.3	73	0.3	75	0.2	0	0	0	0
2	65	0.7	65	0.7	66	0.6	66	0.6	67	0.6	0	0	0	0
3	67	1.7	67	1.4	68	1.4	74	1.4	103	1.1	0	0	0	+
4	65	0.6	65	0.6	65	0.6	69	0.6	83	0.6	0	0	0	+
5	67	0.5	68	0.5	68	0.5	69	0.5	81	0.5	0	0	0	0
6	70	0.5	70	0.5	70	0.5	70	0.5	76	0.4	0	0	0	0
7	70	2.3	71	1.9	72	1.9	77	1.7	95	1.4	0	0	0	+
8	68	2.0	70	1.5	75	1.3	83	1.0	97	1.0	0	0	0	+
9	77	13.1	83	9.0	87	7.6	102	4.9	117	3.0	+	+	-	
10	72	6.6	75	5.0	78	3.9	89	1.6	98	1.0	+	+	-	
11	72	8.3	76	6.3	83	4.8	98	3.4	124	1.7	+	+		
12	72	5.3	74	3.3	76	2.7	86	1.9	103	1.2	+	0	0	
13	69	3.5	70	2.7	72	2.1	82	1.4	91	1.0	+	0	0	
14	74	6.7	76	6.0	80	3.9	92	1.8	99	1.2	+	+	-	
15	76	3.1	77	3.1	79	2.9	89	1.7	106	1.2	0	0	-	
16	71	8.6	74	7.7	80	6.6	96	2.9	112	1.9	+	+		
17	79	10.0	83	8.9	88	7.5	106	3.9	131	2.2	+	+		
18	78	7.5	81	6.2	85	4.8	100	2.5	123	1.0	+	+		
19	68	3.0	69	2.3	72	1.8	80	1.1	86	0.9	+	0	0	0
20	74	5.0	74	4.3	77	3.7	85	2.3	94	1.7	+	0	-	
21	72	5.0	73	4.4	76	3.6	88	2.4	121	1.6	+	0	-	
22	70	1.6	72	1.3	77	1.0	79	0.9	90	0.7	0	0	0	0
23	73	0.6	74	0.6	74	0.6	74	0.6	81	0.5	0	0	0	0
24	81	8.5	83	6.4	87	5.2	97	2.7	104	1.9	+	+	-	
25	65	0.7	65	0.7	65	0.7	66	0.7	66	0.7	0	0	0	0
26	66	1.7	67	1.5	69	1.5	79	0.9	94	0.6	0	0	0	+
27	70	3.9	71	3.5	73	3.5	82	2.9	101	1.6	0	0	0	
28	73	0.4	73	0.4	73	0.4	73	0.4	74	0.4	0	0	0	0

	120 vs. 30	120 vs. 60	120 vs. 600	120 vs. 3600
QM-120 Better	13	8	0	5
QM-120 Equivalent	15	20	17	9
QM-120 Worse	0	0	7	0
Incomparable	0	0	4	14

Figure 7: Effect of varying thresholds.

lengths. We considered thresholds of 30 frame times (1/2 second), 60 frame times (1 second), 120 frame times (2 seconds), 600 frame times (10 seconds), and 3600 frame times (1 minute). Figure 7 shows the simulation results. In the figure, these policies are labeled QM-30, QM-60, QM-120, QM-600, and QM-3600 respectively. The rightmost columns show the comparison between QM-120 and the other policies.

Figure 7 shows that QM-600 performs best relative to QM-120. On 24 runs, QM-600 is judged to have provided better or equivalent performance. On the other four incomparable runs, QM-120 produced slightly better average display latency while QM-600 produced approximately one-third to one-quarter the number of gaps. QM-120 performed as well or better than QM-30 and QM-60 on every run. Finally, QM-120 performed as well or better than QM-3600 on half the runs and was incomparable on the remainder.

It is interesting to compare the performance of QM-30 and QM-60 with that of the I-2 policy. The QM-30 policy and QM-60 policy produced average display latencies that were similar to or better than those produced by I-2. Furthermore, both produced similar or better gap rates. For example, on run 9, the I-2 policy produced a display latency of 81 ms. with a gap rate of 23.0 gaps/min. The QM-60 policy produced a display latency of 83 ms. with a gap rate of 9.0 gaps/min. Thus, by choosing an appropriate threshold value, we can use queue monitoring to produce behavior similar to that produced by an I-policy, but with better performance on runs where an I-policy behaves badly.

The results for the QM-3600 policy are also interesting. In general, the QM-3600 policy produced results similar to those produced by the E-policy, with a slightly lower average display latency and a slightly higher gap rate. Thus, the QM-3600 policy is incomparable with the QM-120 policy on exactly the same runs that the E-policy was incomparable with the QM-120 policy. Again, by choosing an appropriate threshold value, we can use queue monitoring to produce behavior similar to that produced by the E-policy, but with better performance on runs where the E-policy behaves badly.

From these results, we conclude that thresholds are a useful tunable parameter for the general queue monitoring policy. A range of thresholds produces a range of results, from low-latency with many gaps to high-latency with few gaps. For our comparison rule and for the threshold values we examined, a threshold of 10 seconds for all queue lengths performs best. Given a particular network environment and a particular comparison rule, it should be possible to find an optimal threshold value.

The final part of the study examined the performance of the general queue monitoring policy. Recall that individual thresholds can be defined for each queue length, whereas in the previous simulations we used the same threshold value for all queue lengths. These thresholds can be arbitrary, but for purposes of this study, we defined a particular rule for setting the threshold values. This rule has two parameters: a threshold value for a queue of length 3, referred to as the *base threshold*, and a decay factor which specifies a rate at which the thresholds decrease with increasing queue length. For example, a queue monitoring policy with a base threshold of 3600 and a decay factor of 2 would have the threshold values: 3600 for queues of length 3, 1800 for queues of length 4, 900 for queues of length 5, *etc*. In Figure 8, we show the results for three policies: base threshold 120 with decay factor 2, base threshold 600 with decay factor 2, and base threshold 3600 with decay factor 2. In the figure these are labeled QM-(120,2), QM-(600,2) and QM-(3600,2). The figure also shows the QM-120 policy which is the base for comparison with the other policies.

With a base threshold of 120 frame times, decreasing the thresholds for longer queue lengths did not improve the QM-120 policy. One run (run 24) was worse, and the runs that were equivalent according to the comparison rule generally had comparable latencies and slightly worse gap rates. On the other hand, for a base threshold of 600 frame times, decreasing the thresholds did improve the QM-600 policy. Relative to the QM-120 policy, QM-(600,2) performed as well or better on every run. Furthermore, it performed better than QM-120 on 11 runs, where QM-600 only performed better than QM-120 on

Run	QM (120) Latency ms.	QM (120) Gaps /min.	QM (120,2) Latency ms.	QM (120,2) Gaps /min.	QM (600,2) Latency ms.	QM (600,2) Gaps /min.	QM (3600,2) Latency ms.	QM (3600,2) Gaps /min.	120 vs. 120,2	120 vs. 600,2	120 vs. 3600,2
1	66	0.3	66	0.3	73	0.3	75	0.2	0	0	0
2	66	0.6	66	0.6	66	0.6	67	0.6	0	0	0
3	68	1.4	67	1.6	68	1.4	78	1.4	0	0	0
4	65	0.6	65	0.6	68	0.6	82	0.6	0	0	+
5	68	0.5	68	0.5	68	0.5	72	0.5	0	0	0
6	70	0.5	70	0.5	70	0.5	76	0.4	0	0	0
7	72	1.9	71	1.9	72	1.8	82	1.5	0	0	0
8	75	1.3	74	1.5	79	1.0	89	1.0	0	0	0
9	87	7.6	85	8.5	97	5.7	113	3.2	0	-	
10	78	3.9	78	4.2	88	1.7	97	1.0	0	-	
11	83	4.8	81	5.2	91	3.6	110	2.1	0	-	
12	76	2.7	75	2.8	82	2.0	94	1.3	0	0	
13	72	2.1	72	2.1	81	1.4	91	1.0	0	0	
14	80	3.9	79	4.0	90	2.0	99	1.2	0	-	
15	79	2.9	78	2.9	85	1.8	100	1.3	0	-	
16	80	6.6	77	7.1	90	3.6	106	2.1	0	-	
17	88	7.5	82	8.2	95	5.5	119	2.9	0	-	
18	85	4.8	84	4.9	98	2.7	120	1.1	0	-	
19	72	1.8	72	1.8	79	1.1	83	0.9	0	0	0
20	77	3.7	76	3.7	85	2.3	94	1.7	0	-	
21	76	3.6	74	3.9	82	2.6	98	1.8	0	-	
22	77	1.0	77	1.0	78	0.9	86	0.8	0	0	0
23	74	0.6	74	0.6	74	0.6	81	0.5	0	0	0
24	87	5.2	83	6.3	90	4.0	100	2.2	+	-	-
25	65	0.7	65	0.7	66	0.7	66	0.7	0	0	0
26	69	1.5	69	1.5	78	1.0	89	0.6	0	0	+
27	73	3.5	72	3.5	78	3.1	92	2.0	0	0	
28	73	0.4	73	0.4	73	0.4	74	0.4	0	0	0
							QM-120 Better		1	0	2
							QM-120 Equivalent		27	17	12
							QM-120 Worse		0	11	1
							Incomparable		0	0	13

Figure 8: Effect of multiple thresholds.

7 runs. In general, QM-(600,2) produced lower display latencies than QM-600 with only slightly higher gap rates. Decreasing the thresholds also helped to improve the performance of the QM-3600 policy. On a number of runs, the QM-(3600,2) policy produced significantly lower display latencies than those produced by QM-3600 with only slightly higher gap rates. Relative to the QM-120 policy, QM-(3600,2) performed somewhat better than QM-3600.

From these results, we conclude that the principle of decreasing thresholds has some utility. For low base thresholds, decreasing thresholds does not help, and may even hurt performance. But for higher base thresholds, decreasing the thresholds seems to help a great deal for some network conditions, without adversely affecting performance for other network conditions.

6. Summary and Conclusions

If we wish to support computer-based video conferences transmitted over interconnected local-area networks based on ethernets and token rings, we must address the effect of variable network transmission delays (*i.e.*, delay jitter) on the perceived quality of audio and video playout. The fundamental effect of delay jitter is to necessitate a tradeoff between display with low latency and display with few gaps. We have presented a new policy for managing this tradeoff called queue monitoring. This policy operates by observing delay jitter over time and dynamically adjusting display latency in order to support low-latency conferences with an acceptable gap rate. Overall, we conclude that queue monitoring can be an effective policy for ameliorating the effect of delay jitter in a building-sized internetwork.

In this paper, queue monitoring was evaluated by comparing it with two policies from the literature: the I-policy and the E-policy. The performance of these policies was evaluated by recording the end-to-end delays experienced by audio frames during video conferences run over our building network and simulating the effect of each policy on the resulting trace. In these conferences, the end-to-end delays experienced by most audio frames were in the range of 35-40 ms (including acquisition and processing time as well as network transmission time), but the variation in end-to-end delays was as much as 200 ms. Furthermore, even at times of day when there was little other activity on the network, end-to-end delays varied by as much as 80 ms. Over this range of network loads, queue monitoring consistently performed as well or better than either the I-policy or the E-policy. Our results have shown that queue monitoring with a base threshold of 600 frame times (10 seconds) and a decay factor of 2 resulted in the best performance.

Two runs best illustrate the advantages of the queue monitoring policy over the E-policy and the I-policy. Run 3 is an example of a run on which the E-policy performed poorly, producing an average display latency of 140 ms. with a gap rate of 0.9 gaps/minute. In contrast, the QM-(600,2) policy produced an average display latency of 68 ms. with a gap rate of 1.4 gaps/minute. Run 24 is an example of a run on which the I-3 (and the other I-policies) performed poorly, producing an average display latency of 93 ms. with a gap rate of 15.0 gaps/minute. For this run, QM-(600,2) produced an average display latency of 90 ms. with a gap rate of 4.0 gaps/minute. Together, these examples show that the queue monitoring policy can adapt to many network conditions to produce good quality playout.

Queue monitoring is also a flexible and tunable policy. Experiments with a range of thresholds produced a range of behaviors, from low display latency with frequent gaps, to high display latency with few gaps. As a result, we believe that queue monitoring can be tuned to meet the requirements of particular applications and to adapt to the demands of particular network environments. In addition, queue monitoring has a simple and efficient implementation. A particularly important feature of this implementation is that it operates without feedback from the audio or video source and without synchronized clocks. Thus queue monitoring can be used effectively in computer-based conferences with many participants each of which may experience differing network delays.

There are a number of issues still to be addressed in evaluating the queue monitoring policy. First, in this paper we have compared queue monitoring to the I-policy in which the display latency remains fixed over an entire run. An alternative would be an I-policy for which we choose a new display latency at regular intervals, perhaps based on observations of delay jitter in the recent past.

Second, we have compared policies using average display latency and average gap rate. There are many other factors that can influence perceived quality including the distribution of gaps throughout a conference, the number of display latency changes, and the distribution of periods of high and low display latency throughout a conference. Ideally, policies should be compared using a more general measure of audio and video quality which accounts for all the factors which determine perceived quality.

Third, the study presented in this paper is based on audio and video data transmitted over a building-area network. We are also interested in determining the extent to which the queue monitoring technique scales. Future work will include repeating the study in this paper for a succession of larger networks. Such a study will help to identify the types of networks which can be supported without resorting to new specialized network services.

Fourth, the emphasis in this work has been on managing delay jitter. Another aspect of our work is a forward error correction (FEC) strategy for minimizing frame loss [4]. By decreasing loss, FEC will support the assumptions underlying queue monitoring, namely that display queue length only decreases as a result of late arrivals. However, frames that are recovered through FEC will incur longer end-to-end delays, thus increasing delay jitter. Because of these effects, we must investigate the impact of FEC and other mechanisms for reducing loss (*e.g.*, timeouts and retransmission) on queue monitoring and the other policies.

Fifth, work must be done on choosing good threshold values, but this work will require a new quality measure. Simulation using a variety of threshold values indicates that large changes in threshold values may only produce small changes in average display latency and average gap rate. As such, work on choosing threshold values will involve making tradeoffs which result in small changes to display latency and gap rate. While simple measures of quality may be sufficient to evaluate the gross performance characteristics of a policy, proper evaluation of small tradeoffs will require a better quality measure than we currently have available.

Finally, the use of decreasing thresholds for longer queue lengths must be investigated further. While the study has shown that the principle of decreasing thresholds can improve performance, it is not yet clear what strategy should be used to set those thresholds. Among the possibilities are the geometric decrease in threshold values we used in this study, and a linear decrease in threshold values suggested by Jones and Hopper [5].

7. References

[1] Ferrari, D., "Delay Jitter Control Scheme For Packet-Switching Internetworks," Computer Communications, Vol. 15, No. 6 (July/August 1992), pp. 367-373.

[2] Issacs, E., Tang, J.C., "What Video Can and Can't Do for Collaboration: A Case Study," Proc. AC M Intl. Conf. on Multimedia, Anaheim, CA, August 1993, pp. 199-205.

[3] Jeffay, K., Stone, D.L., Smith, F.D., "Kernel Support for Live Digital Audio and Video," Computer Communications, Vol. 15, No. 6 (July/August 1992), pp. 388-395.

[4] Jeffay, K., Stone, D.L., Smith, F.D., "Transport and Display Mechanisms for Multimedia Conferencing Across Packet-Switched Networks," Computer Networks and ISDN Systems, Vol. 26, No. 10 (July 1994), pp. 1281-1304.

[5] Jones, A., Hopper, A., "Handling Audio and Video Streams in a Distributed Environment," Proc. ACM Symp. on Operating Systems Principles, Asheville, NC, December 1993, ACM Operating Systems Review, Vol. 27, No. 5, pp. 231-243.

[6] Naylor, W.E., Kleinrock, L., "Stream Traffic Communication in Packet-Switched Networks: Destination Buffering Considerations," IEEE Trans. on Communications, Vol. COM-30, No. 12 (December 1982), pp. 2527-2534.

[7] Schulzrinne, H., "Voice Communication Across the Internet: A Network Voice Terminal," Technical Report, Univ. of Massachusetts 1992.