QUEUE MONITORING
A Delay Jitter Management Policy*

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

Abstract: This paper describes *queue monitoring*, a policy for managing the effect of delay jitter on audio and video in computer-based conferences. By observing delay jitter over time, this policy 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 other policies in an empirical study of a computer-based conferencing system. Our results show that queue monitoring performs well under a variety of observed network loads.

1. Introduction

Our research is concerned with network and operating system support for computer-based conferencing 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 "late." When data arrives after the time at which it should be displayed, a "gap" appears in the display which affects the user's perception of the conference. Our goal is to provide solutions to this problem for small internetworks consisting of several physical networks connected via bridges and routers (*e.g.*, a building-area 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, frames are acquired from an audio or video source at a precise rate (*e.g.*, one frame every 33 ms. for video). Frames must then be delivered to the display at the same precise rate. Prior to delivery, each frame must be digitized, compressed, transmitted over the network, and decompressed. Once decompressed, a frame may then be buffered for some time waiting to be delivered to the display. The *display latency* of a frame is defined as the total time from acquisition to display.

The *end-to-end delay* of a frame is defined as the total time from acquisition to decompression (*i.e.*, the latency of the frame minus any time spent in buffers waiting to be displayed). If we assume that the time required to digitize, compress, or decompress frames is not constant, or if we assume that the delays incurred

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

In: Network and Operating System Support for Digital Audio and Video, Proceedings, Fourth Intl. Workshop, Lancaster, UK, November 1993, D. Shepherd, G. Blair, G. Coulson, N. Davies, F. Garcia (Eds.). Lecture Notes in Computer Science, Vol. 846, pp. 149-160, Springer-Verlag, Heidelberg, 1994.

transmitting frames over 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*.

To play frames continuously, each must be displayed with constant display latency. But if because of delay jitter, a frame arrives with an end-to-end delay greater than the display latency of the previous frame, there will necessarily be a *gap* in the display (*i.e.*, no video frame is displayed for a 33 ms. interval because we failed to deliver a frame to the display at the proper time). The lower the display latency, the higher the probability of encountering an end-to-end delay sufficient to cause a gap. Thus, there is a fundamental tradeoff between display latency and gaps.

One approach to managing this tradeoff is to prevent delay jitter by providing constant network transmission delays. This is the approach used in computer-based conferencing systems based on networks that provide isochronous services (such as ATM). Another approach minimizes delay jitter by using resource reservation 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 enough resources at each node on the network (*e.g.*, processor capacity, buffer space) to guarantee the performance requirements of the client.

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, delay jitter in the end-to-end delay can occur as a result of variations in the time to digitize, compress, and decompress frames, as well as variations in the time 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 both of these reasons, we assume that delay jitter is a fundamental phenomena and therefore that the tradeoff between display latency and gaps must be explicitly managed.

In this paper, we discuss some basic policies for managing the tradeoff between display latency and gaps. In particular, we describe a policy called *queue monitoring* which is designed to perform well over a range of network conditions and has an efficient implementation. We present an empirical study in which traces from a computer-based conferencing system (described in [2]) are used to simulate and compare queue monitoring and two policies from the literature: the I-policy and the E-policy. Lastly, we discuss the results of the study which show that queue monitoring performs as well or better than the reference policies over the range of observed network behavior.

2. Adjusting Display Latency

Given that end-to-end delays vary, if we wish to continuously display frames without gaps, we must buffer every frame for a time sufficient to ensure that each is displayed with a latency greater than the worst-case end-to-end delay. However, it is not clear

that our goal should be display with no gaps. Display latency is an important factor in determining the display quality perceived by a viewer. It is likely that in many conferencing applications, as long as gaps occur infrequently, display with low latency and some gaps will be preferable to display with high latency and no gaps. Therefore, if we are buffering to handle the worst-case delay and if the worst-case delay is rarely observed in practice, then most frames will be displayed with latency higher than necessary to support good quality display.

Once we are willing to accept gaps in the display, the problem becomes one of choosing a latency which will result in a good quality display. A good choice for display latency will depend on several factors. First, the acceptable rate of gaps in the display may vary depending on the application. Second, the display latency required to maintain a particular gap-rate will depend on the level of delay jitter, which will vary as a result of congestion and contention for resources at participants' machines and at intermediate nodes on the network.

The problem of choosing a display latency has been extensively studied in the context of packet audio (the transmission of audio streams across packet-switched networks). In many systems, audio consists of a sequence of “talkspurts” (some period of time in which audio data must be acquired 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 [4] 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 [5]), 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.

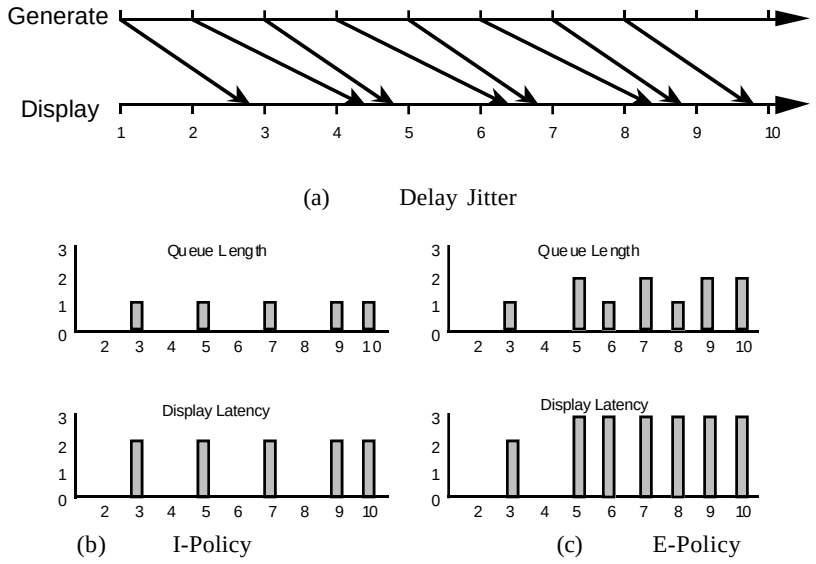
In effect, the start of a talkspurt is a convenient time to change display latency since adjustments can be made by increasing or decreasing the length of a silent period (presumably less noticeable to the listener than adding gaps or not playing part of a talkspurt). However, for the video streams that we consider, there is no concept analogous to talkspurts and silent periods since video is transmitted continuously. Furthermore, we are interested in supporting a variety of audio applications in which silence detection may not be applicable (*e.g.*, music applications). Since we do not have the “free” opportunities to adjust display latency provided by talkspurts, we instead concentrate on adjusting latency in response to gaps in the display.

When a gap occurs because a frame arrives too late to be displayed, we must decide what to do with the late frame. One choice is to discard it. This choice results in every frame that is displayed having the same display latency. Each frame with an end-to-end delay greater than that latency causes a gap. Naylor and Kleinrock call this choice the *I-Policy*. The alternative is to play the late frame at the next opportunity. Because we display frames at the same rate as they are acquired, this choice increases

the latency at which we display subsequent frames. Naylor and Kleinrock call this the *E-Policy*.

The I-policy never adjusts display latency. Rather, one chooses an initial display latency at the beginning of a conference and maintains it throughout. In contrast, the E-policy starts with the lowest possible display latency and then adjusts it upward in response to delay jitter. The effect of the E-policy is to dynamically find a display latency which is sufficient to display frames without gaps by adjusting the latency to be higher than any end-to-end delay yet observed.

Figure 1 illustrates the behavior of the I and E policies in response to late frames. In figure 1a, each frame is represented by a diagonal line from the time when it is acquired to the time it enters a queue of frames waiting to be displayed. The horizontal distance between the endpoints is the end-to-end delay of the frame. Tick marks on the lower timeline indicate *display initiation times*, the times at which the display of new frames starts. Figure 1b shows the performance of the I-Policy with a latency of two frame times (for simplicity in these examples, time is represented as multiples of the time to acquire or display a frame). The top graph plots display queue length for each display initiation time. The bottom graph plots the display latency of the frame being displayed at each display initiation time. In this example, frames 2, 4, and 6 arrive late, causing three gaps. Figure 1c shows the E-policy. Here the late arrival of frame 2 causes a gap, but the resulting increase in latency allows the following frames to be displayed without gaps. This example illustrates an advantage that the E-policy has over the I-policy. If the display latency is lower than the prevailing level of delay jitter, using the I-policy will result in a large number of gaps. If the display latency is higher than the prevailing level of delay jitter, using



(a) Delay Jitter

(b) I-Policy (c) E-Policy

Figure 1

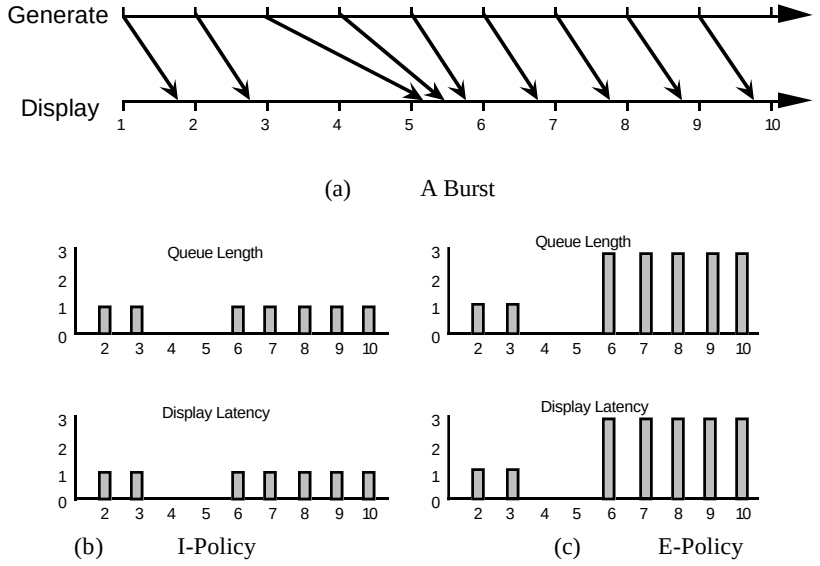
the I-policy will result in most frames being displayed with unnecessarily high latency. In contrast, the E-policy quickly adjusts to a display latency greater than the prevailing level of delay jitter.

Figure 2 illustrates a situation in which the I-policy has an advantage over the E-policy. In this example, frames 3 and 4 arrive late, but the remainder arrive with constant end-to-end delay of less than a frame time. Each policy results in gaps, but the I-policy with a display latency of 1 frame time displays the remaining frames with low latency while the E-policy displays the remaining frames with high latency. A single "burst" of activity on the network which causes a few frames to arrive late has resulted in a permanent increase in display latency.

We would like to combine the advantages of these policies. The I-policy with a low display latency works well during periods of low delay jitter and after bursts, but works poorly during periods of high delay jitter. The I-policy with a high display latency works well during periods of high delay jitter, but plays frames with unnecessary latency during periods of low delay jitter. The E-policy works well during low delay jitter and high delay jitter periods, but behaves poorly after bursts.

To combine these advantages, we propose a policy for adjusting display latency called *queue monitoring*. In short, we use the basic approach of the E-policy and increase display latency by playing late frames. However, if subsequent network activity is stable, as demonstrated by the buildup of a queue at the display, we reduce display latency by discarding frames.

In queue monitoring, we associate an array of counters and threshold values with the display queue. Each time we wish to display a frame, we first perform a "thresholding



(a) A Burst

(b) I-Policy (c) E-Policy

Figure 2

operation." If the display queue has length *m*, then counters 1 through *m*-1 are incremented; all other counters are reset. For each counter that exceeds its threshold value, the counter is reset and the oldest frame is discarded from the queue. The oldest remaining frame is then displayed. The effect of discarding an old frame is to decrease the display latency, as each succeeding frame can be displayed earlier.

This policy is based on an assumption that we can predict the level of delay jitter for the near future based on the observed level of delay jitter in the recent past. This is analogous to the working set concept of page replacement in virtual memory management. Here, our working set is the display latency required to minimize the probability of a gap occurring in the near future. Intuitively, threshold values specify a duration (measured in frame times) after which we will conclude that delay jitter has diminished below a certain level. Specifically, the threshold value for counter *n* specifies the duration after which, if the display queue has continuously contained more than *n* frames, we will conclude that frames can be displayed with reduced latency and without gaps as long as the current level of delay jitter persists.

In our current system, threshold values for large queue lengths specify a small duration (*e.g.*, about a second) and increase geometrically for shorter queue lengths. This has the effect of reducing display latency quickly after long delays due to large bursts of network traffic, but approaching minimal latency slowly. These values reflect an assumption that large variations in end-to-end delay occur infrequently and smaller variations occur much more frequently.

3. 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-policy and the E-policy. However, it is difficult to definitively conclude that one policy performed better than another for any given run of the system. Clearly, if policy A results in lower display latency and less gaps than policy B, it is performing better. However, if policy A has a lower display latency but a higher gap rate than policy B, which has performed better? The answer probably depends on a number of factors including: the level of display latency, the gap rate, the resolution of the display, and the application of the viewer. It may also depend on the distribution of gaps throughout the run, the number of display latency changes, and the distribution of high versus low display latency periods throughout the run. A proper standard for comparing policies would take each of these factors into account in making a judgment.

Unfortunately, we are not aware such a standard. Therefore we have adopted a simple and arbitrary comparison rule for the analysis in this paper. The performance of a policy for a run is evaluated on 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 per minute are significant. Given these rules for significance, 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 run. However, it is difficult to compare multiple runs. One difficulty is fundamental. The video hardware that acquires frames is not synchronized with the display. To illustrate the effect this has on display latency, assume there is no end-to-end delay (*i.e.*, acquisition and arrival are simultaneous). Nevertheless, we must wait until the next display initiation time to display a 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. This synchronization time is a random variable and varies between runs.

The second difficulty in comparing multiple runs arises from our working definition of the I-policy. Ideally, the I-policy should enforce a particular display latency. However, 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 a function of the end-to-end delay of the first frame.

4. The Study

We have performed an empirical study to gauge the effectiveness of the queue monitoring policy in a building-sized internetwork. 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 described above) connected via a central broadband network. In all, this environment should be typical of those generally found in the "last mile" to the desktop.

The study was performed using our computer-based audio and video conferencing system. Briefly, this system uses IBM PS/2 workstations as the acquisition and display machines. 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 8-bit color pixels; each compressed video frame occupies around 6000-8000

bytes) and 60 audio frames per second (the 2-channel audio stream at a bit rate of 128 Kb per second is packaged into 16.5 ms "frames"). The full system has been extensively described elsewhere [2,3].

In the study, we acquired, transmitted, and displayed audio and video for 10 minute periods. 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 an ethernet to a bridge, through a second ethernet to another gateway, and back across the private token ring to the display machine.

Before each run, we ran a protocol to measure the difference in clock times between the acquisition and display machines. Then for each run, we recorded the acquisition time and the arrival time of each audio frame and we recorded each display initiation time (*i.e.*, times at which the display of a new frame should be initiated). During analysis, the recorded times were adjusted to account for clock drift (measured in a separate experiment).

Twenty-four runs were conducted over the course of a typical day (between 6am and 5pm) covering lightly and heavily loaded periods. Four additional runs were performed between midnight and 1am during nightly backups (which are performed using a central server connected to the backup storage device). Figure 3 gives some basic data about 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 4 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 gives 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.

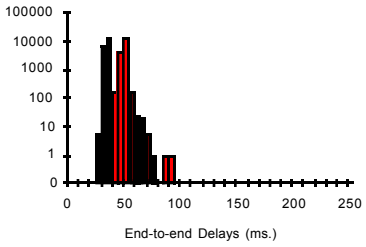
To study the effectiveness of the queue monitoring policy, we compared its behavior to that of the E-policy and the I-policy for each of the 28 runs. For each run, the recorded traces of acquisition times, arrival times, and display initiation times were used as input to a simulator. We simulated each policy on the particular sequence of arrivals and display initiation times and produced the average display latency and the number of gaps incurred for the run.

Figure 5 shows the simulation results for each of the 28 runs. For each run, we simulated the queue monitoring policy with the threshold array (Infinity, 4096, 1024, 256, 64), the E-policy, and the (variant) I-policy with 2 and 3 frame times of display latency. For each of these policies, we give the resulting average display latency (in ms.) and the average gap rate (in gaps/minute). 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 3). A '+' indicates that queue

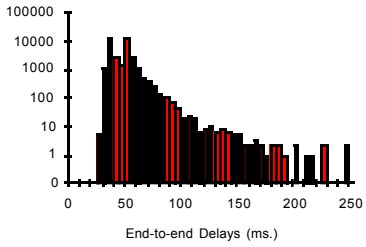
monitoring did better, a ‘0’ means the two were equivalent, a ‘-’ means that queue monitoring did worse, and a blank means the two policies were incomparable.

Run	Time of Day	Avg. Delay ms.	Max. Delay ms.	Lost Frames	Duplicate Frames
1	06:03:20	38	76	1	0
2	06:25:41	38	88	3	0
3	06:36:34	37	171	5	0
4	06:47:58	37	105	1	0
5	08:03:27	38	115	1	0
6	08:14:08	37	73	2	0
7	08:25:47	38	184	7	0
8	08:36:41	39	157	1	0
9	10:02:48	41	186	23	0
10	10:16:28	40	124	4	0
11	10:31:20	41	213	7	0
12	10:49:28	40	140	6	0
13	11:57:13	39	110	5	0
14	12:08:03	41	138	5	0
15	12:19:18	41	133	3	0
16	12:34:44	40	187	11	0
17	14:02:03	41	189	11	0
18	14:13:28	42	141	3	0
19	14:42:32	39	107	4	0
20	14:54:33	40	131	12	0
21	16:01:39	39	171	9	0
22	16:21:30	39	128	2	0
23	16:33:35	39	86	2	1
24	16:55:47	42	242	14	1
25	00:05:55	38	80	4	0
26	00:16:53	38	128	0	0
27	00:27:29	38	134	8	0
28	00:38:35	38	83	2	0

Figure 3: All Runs: Basic Data



Run 2: Delays



Run 24: Delays

Figure 4

Run	I-Policy 2 (I-2)		I-Policy 3 (I-3)		E-Policy		QM-Policy		QM vs. I2	QM vs. I3	QM vs. E
	Latency ms.	Gaps /min.	Latency ms.	Gaps /min.	Latency ms.	Gaps /min.	Latency ms.	Gaps /min.			
1	80	0.1	97	0.1	75	0.2	66	0.5	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	63	1.6	+	+	+
4	65	0.7	82	0.4	104	0.6	64	0.8	0	+	+
5	71	0.6	88	0.4	93	0.5	65	0.7	0	+	+
6	70	0.3	86	0.2	76	0.4	68	0.7	0	+	0
7	73	2.9	90	1.6	106	1.2	72	2.0	0	+	+
8	62	5.1	79	2.4	106	0.9	74	1.6	+	0	+
9	81	23.0	98	12.6	118	2.8	88	8.0	+	+	
10	70	14.6	87	3.6	113	0.8	82	3.2	+	0	
11	66	25.2	83	6.9	133	1.4	83	5.1		+	
12	71	9.6	87	3.4	114	0.9	77	3.0	+	0	
13	67	9.6	84	2.8	96	0.8	74	2.1	+	0	
14	72	15.1	88	3.9	101	1.1	83	3.7	+	0	
15	76	4.4	92	1.7	117	0.9	82	2.5	+	0	
16	68	18.6	85	8.0	114	1.8	81	5.4	+	+	
17	77	22.0	93	12.1	146	1.8	88	6.8	+	+	
18	76	13.0	93	4.1	131	0.7	88	4.2	+	0	
19	66	5.0	82	1.3	87	0.9	73	2.0	+	0	-
20	73	11.3	90	4.1	98	1.6	80	3.2	+	0	
21	70	12.8	87	6.1	159	1.5	76	3.7	+	+	
22	79	1.4	95	0.5	100	0.6	75	1.3	0	+	+
23	75	0.4	91	0.2	84	0.4	73	0.8	0	+	0
24	77	39.6	93	15.0	104	1.8	87	5.5	+	+	
25	65	0.8	81	0.4	66	0.7	65	0.9	0	+	0
26	64	5.4	81	0.9	122	0.6	72	1.5	+	0	+
27	70	7.8	87	3.8	107	1.3	76	3.4	+	0	
28	76	0.3	93	0.2	75	0.3	70	0.7	+	+	0

All Runs: Results
Figure 5

5. Conclusions

From Figure 5, we see that the queue monitoring policy performs as well or better than the I-3 policy for every run and as well or better than the I-2 policy for all but run 11 (which is incomparable). From these results, we can conclude that in general, queue monitoring is more effective than I-2 and I-3. Furthermore, the queue monitoring policy always results in a display latency lower than that of the I-3 policy. Therefore, all I-policies with larger display latencies cannot perform better than queue monitoring (by our comparison rule). However, because gap rates for the policies with larger display latency will decrease, the policies may be incomparable.

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

With respect to the E-policy, queue monitoring performs as well or better on 13 runs, worse on 1 run, and is incomparable on 14 runs. The runs in which queue monitoring 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. Queue monitoring always has smaller display latency and a higher gap rate (5.2 gaps/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 rate, then queue monitoring outperforms the E-policy. For other goals, we make no conclusions.

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 various I-policies for which the display latency remained fixed over an entire run. An alternative would be to use 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 delay latency and average gap rate computed over entire runs. The analysis needs to be expanded to account for shorter periods of high latency or high gap rate occurring in the middle of an otherwise stable run. More generally, policies should be compared using a measure of quality which accounts for application specific effects and human perception.

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 supporting somewhat larger networks, such as our campus network. Future work should 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 specialized services.

Fourth, the emphasis in this work is on managing delay jitter. Another aspect of our work is a forward error correction (FEC) strategy for minimizing frame loss. By decreasing loss, FEC will support the assumptions underlying queue monitoring, namely that display queue length only decreases as a result of late arrivals. Furthermore, 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.

Finally, work must be done on choosing proper 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. For example, queue monitoring with

thresholds (Infinity, 4096, 1024, 256, 64) on run 24 produces 87 ms. of average display latency and a gap rate of 5.5 gaps/minute while doubling these threshold values results in 89 ms. of display latency and 4.3 gaps/minute and halving them results in 82 ms. of display latency and 6.6 gaps/minute. 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.

6. References

[1] Ferrari, Domenico. "Delay Jitter Control Scheme For Packet-Switching Internetworks", Computer Communications, Vol. 15, No. 6, Jul./Aug. 1992, pp. 367-373.

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

[3] Jeffay, K., Stone, D.L., Smith, F.D. "Transport and Display Mechanisms for Multimedia Conferencing Across Packet-Switched Networks", Computer Networks and ISDN Systems, (to appear).

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

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