Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 97-053

Performance Evaluation of Media Losses in the Continuous Media Toolkit*

by: Duminda Wijesekera, Srivatsan Varadarajan, Shwetal Parikh, Jaideep Srivastava and Anil Nerode

Performance Evaluation of Media Losses in the Continuous Media Toolkit*

Duminda Wijesekera, Srivatsan Varadarajan, Shwetal Parikh,
Jaideep Srivastava and Anil Nerode†
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.
Institute for Foundations of Intelligent Systems, Cornell University, Ithaca, NY 14853†.
e-mail: {wijesek|ssparikh|varadara|srivasta}@cs.umn.edu,
nerode@hybrid.cornell.edu

Abstract

Rapid growth of multimedia systems, and accordingly research in this area requires fast prototyping environments. The *Berkeley Continuous Media Toolkit (CMT)* is a popular environment that satisfies this need. Form a human user's perspective, in order for multimedia demonstrations to be comprehensible, the number of audio or video frames dropped and the timing delays in the ones that are displayed, need to be kept to a minimum. Therefore, it is important to know the frame dropping characteristics of CMT. In a series of experiments we monitored the variation of these parameters with respect processor and network loads. It was observed that loads affect aggregate frame drops at lower rates and consecutive frame drops at higher rates. Because at a higher rates a large number of consecutive frames are dropped, the ones that are played appear in a more timely manner. As a solution to observed problems, we present some QoS based approaches to control drop and delay parameters.

Key Phrases: Quality of Service, Toolkit support for multimedia, Continuous media, Performance evaluation, Loss measurement

1 Introduction

Rapid growth of multimedia systems, and accordingly research efforts in this area, have made it necessary to have toolkit support for rapid prototyping. Being one of the most popular toolkits offered in multimedia, the *Berkeley Continuous Media Toolkit (CMT)* [SRY93, MPR97] has gone a long way in fulfilling this need. In [MPR97], Mayer-Patel and Rowe measured the performance of CMT and its system overheads. In a related experiment, we have reported [WPV+98] the quality of audio-video synchronization provided by CMT. During our measurements of synchronization quality, we noticed that CMT drops audio and video frames almost arbitrarily. In order to quantify frames drops and delays under differing load conditions, we decided to measure CMT's performance with respect to these parameters.

CMT provides objects to fetch stored media clips from files, transport them across networks, and display them at remote sites. Therefore, we have measured the quality of stream services with respect to *Quality of Service (QoS)* metrics that were designed to measure such lossy streams

*This work is partially supported by Air Force contract number F30602-96-C-0130 to Honeywell Inc, via subcontract number B09030541/AF to the University of Minnesota and DOD MURI grant DAAH04-96-10341 to Cornell University

[WS96, WS95]. In these metrics, we quantify aggregate and consecutive losses of content, and aggregate and consecutive drifts in display times.

In providing distributed continuous media (CM) services, as described in Section 3, some objects (eg. video `segment` that fetches frames from files, and `play` that is responsible for scheduling displays) have built in policies to drop frames. Because CMT provides distributed CM services, we speculate that its QoS varies according to processor and networked loads. Accordingly, we carried out two experiments: the first, which we call the *local experiment*, measures the quality of stream services against varying processor loads, and the second, which we call the *remote experiments*, measures the quality of stream services against processor and network loads.

The general trend of our results indicate that as the speed increases, the frame drops increase. Higher loads induce aggregate drops at lower speeds and consecutive drops at higher speeds. Furthermore, when a large number of frames are dropped by CMT, the remaining are displayed in a timely manner.

Since our evaluations measure perceptual quality of stream services, they should be measured external to the system that is providing the continuous media (CM) services [SNL95]. As shown in [SNL95], internal and external measurements of a stream could be different. Performance evaluations of frame drops do not depend on the external-ness of the measuring device, but the timing does. Accordingly, our timing drift parameters are in the process of being verified by external measurements.

The rest of the paper is organized as follows: Section 2 presents our metrics for continuity. Section 3 presents the design and functionality of CMT. Section 5 presents our results in the local experiment, and Section 6 presents our results in the remote experiments. Finally, Section 7 contains concluding remarks.

2 Continuity Metrics

For the purpose of describing QoS metrics for lossy media streams, we envision a CM stream as a flow of data units (referred to as logical data units - LDUs in the uniform framework of [SB96]). In our case, we take a video LDU to be a frame, and an audio LDU to constitute 8000/30, i.e. 266 samples of audio[1]. Given a rate for streams consisting of these LDUs, we envision that there is time slot for each LDU to be played out. In the ideal case a LDU should appear at the beginning of its time slot.

2.1 Metrics for Stream Continuity

Continuity of a CM stream is measured by three components: *flow rate, timing drift* and *content loss*. The ideal rate of flow and the maximum permissible deviation from it constitute our *rate* parameters. As stated, given the ideal rate and the beginning time of a CM stream, there is an ideal time for a given LDU to be displayed. The appearance time of a given LDU may deviate from this ideal. Our *drift* parameters specify aggregate and consecutive non-zero drifts from these ideals, over a given number of consecutive LDUs in a stream. For eg., first four LDUs of two example streams, with their expected and actual times of appearance, are shown in Fig. 1. In the first example stream, the drifts are respectively 0.0, 0.2, 0.2 and 0.2 seconds; and accordingly it has an aggregate drift of 0.6 seconds per 4 time slots, and a non-zero consecutive drift of 0.6 seconds. In the second example stream, the largest consecutive non-zero drift is 0.3 seconds and the aggregate

[1]SunAudio has 8-bit samples at 8kHz, and an audio frame constitutes 266 such samples, equivalent to a play time of one video frame, i.e. 1/30 seconds.

drift is 0.5 seconds per 4 time slots. The reason for a lower consecutive drift in stream 2 is that the unit drifts in it are more spread out than those in stream 1.

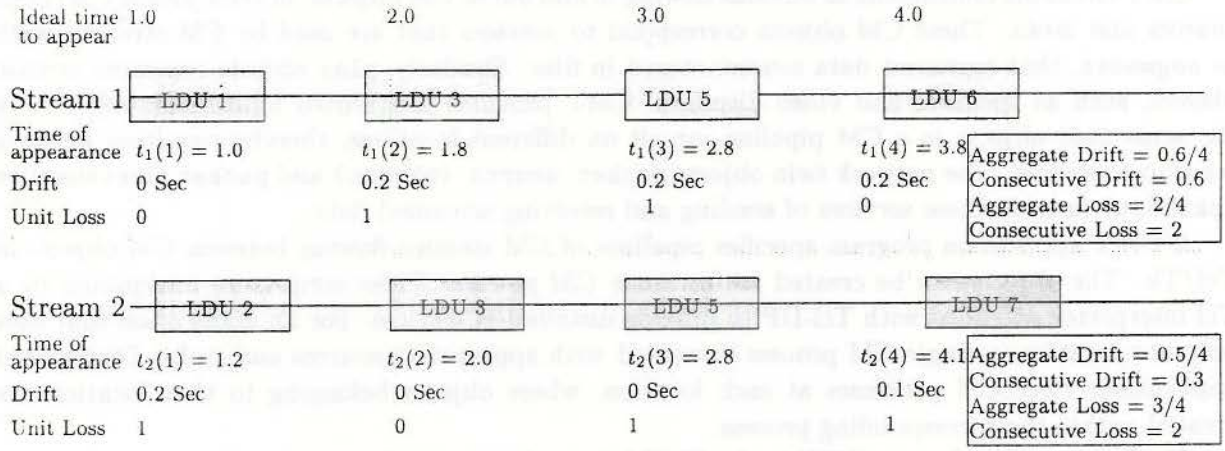

Figure 1: Two Example Streams used to Explain Metrics

In addition to timing and rate, ideal contents of a CM stream are specified by the ideal contents of each LDU. Due to loss, delivery or resource over-load problems, appearance of LDUs may deviate from this ideal, and consequently lead to discontinuity. Our metrics of continuity are designed to measure the average and bursty deviation from the ideal specification. A loss or repetition of a LDU is considered a unit loss in a CM stream. (A more precise definition is given in [WS96].) The aggregate number of such unit losses is the *aggregate loss* of a CM stream, while the largest consecutive non-zero loss is its *consecutive loss*. In the example streams of Fig. 1, stream 1 has an aggregate loss of 2/4 and a consecutive loss of 2, while stream 2 has an aggregate loss of 2/4 and a consecutive loss of 1. Once again, the reason for the lower consecutive loss in stream 2 is that its losses are more spread-out than those of stream 1.

2.2 Parameters for Continuity Metrics

In a user study [WSNF97] it has been determined that aggregate losses of upto 17/100 were imperceptible, those beyond 23/100 were annoying, and in between these two values were tolerable. The tolerable value for consecutive losses was determined to be two frames, i.e about 8000 samples. For audio this limit was about three frames. Due to the inability of precisely introducing timing drifts, the user study did not estimate any values for it.

3 CMT: Design and Functionality

The Berkeley Continuous Media Toolkit, commonly referred to as CMT, has been constructed for the purposes of application development and research in multimedia systems[MPR97]. It is an extensible system consisting of a three layered architecture. The topmost layer, referred to as the *application code layer*, consists of Tcl, Tk and Tcl-Dp [SRS93] code. The next layer, consists of a CM object model, time and synchronization services, CM event services and storage/buffer services for CM streams. The bottom most *resource layer* consists of operating system and hardware services. The complete design of CMT is explained in detail in [MPR97].

One of the outstanding features of CMT is the ease of programming in Tcl/Tk [Ous94, Wel95],

and the extensible and relatively *policy free* nature of implementation, which makes it a great testbed for experimentation.

CMT envisions multimedia as streams flowing in and out of CM objects, in their journey between *sources* and *sinks*. These CM objects correspond to *services* that are used by CM streams, such as `segments`, that represent data sources stored in files. Similarly, `play` objects represent stream players, such as speakers and video displays. CMT provides distributed multimedia services in the sense that objects in a CM pipeline can sit on different locations, thereby requiring network transport services. The network twin objects `packet source (pktSrc)` and `packet destination (pktDest)` provide these services of sending and receiving streamed data.

A CMT application program specifies pipelines of CM streams flowing between CM objects in Tcl/Tk. The objects can be created within some *CM process*. These scripts are interpreted by a Tcl interpreter extended with Tcl-DP to provide distributed services. For an application that uses only one location, a single CM process is created with appropriate sources and sinks. Distributed applications have CM processes at each location, where objects belonging to that location are created within the corresponding process.

The Logical Time System (LTS) in the CMT Library layer provides a mechanism for applications to maintain a concept of where in time the application is, and how fast time is progressing. More than one LTS can exist in one application and different CM processes can be synchronized by using the same LTS. CMT ensures that the LTS progresses according to the *rate* specified by the application program.

CMT maintains CM streams by passing data between objects by either the push model or the pull model. In our experimental setup, we used the push model, where the producer of data, typically the source object, initiates the data transfer to the object immediately downstream from it. The source object maintains a continuous streams of CM data by periodically invoking itself to fetch data from the specified file at the specified rate, and transfers data to the corresponding object downstream. The period of such data fetches are specifiable in application code. Objects that are being called by some other object upstream provides an `accept` method, that is being passed a buffer containing appropriate data by the callee. After *processing* the data in the buffer, such an intermediate object calls the accept method of its immediate downstream neighbor, thereby maintaining the flow of CM data through the specified pipeline, until a sink object such as a display device is reached. The sink objects are the final consumers of CM data, whereby it is displayed to the human viewer.

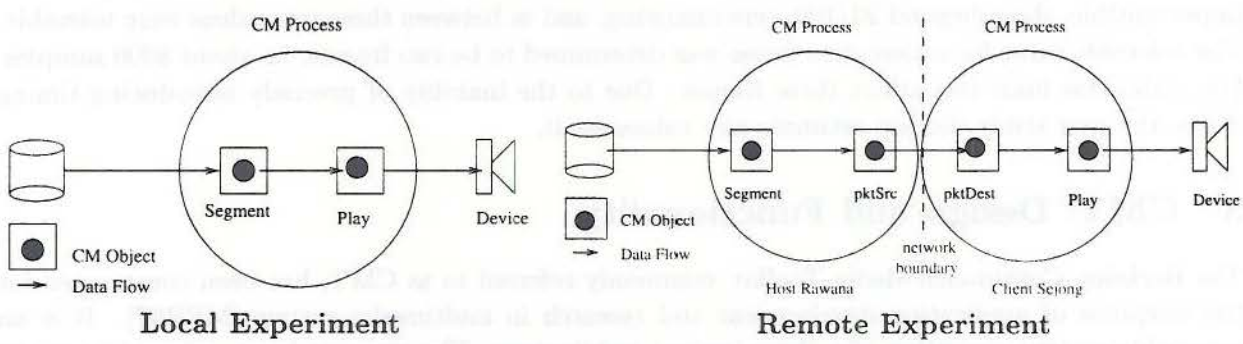

Figure 2: Local and Remote Playback Experiments in CMT

3.1 Application Code Used in Our Experiment

In our experiments, we use two application programs, referred to as *local playback application* and *remote playback application* in [MPR97], graphically represented in Fig 2. The local experiment consists of a single CM process containing four objects, namely: `mjpegSegment`, `auSegment`, `mjpegPlay` and `auPlay`. As shown in Fig 2, they are pipelined so that `mjpegSegment` passes a video stream to `mjpegPlay` to be displayed at the local screen and `auSegment` passes an audio stream to `auPlay` to be displayed at the speaker. Transparent to the application programmer, the play objects call an accept method of a low level object called `device`, that is responsible for the final display of media units.

In the *remote play* experiment, the `segments` and `play` objects are on two different machines that are connected by Ethernet. Consequently, in addition to the objects used in the local play, *packet source* and *packet destination* network twins are used.

Because the performance analysis entails the behavior of objects used in our application code, following sections are devoted to a description of relevant properties of these objects.

3.1.1 Segment Object

Given the current value of the `LTS` object, the `segment` object determines the appropriate video frames or parts of audio frames to be fetched and played in the current fetch cycle. It then schedules itself to be called back after a lapse of the *cycle time*. The application code can also set a parameter *send ahead*, that specifies the time that the server is supposed to be ahead of the client.

In the process of being recalled, based on the time it was actually supposed to be called and the time it was called, the video `segment` determines the number of frames to be dropped by using an *inverse binary ordering* scheme[Smi]. This scheme attempts to minimize the consecutive dropping of frames, so that at the end, the stream would not appear *jittery*. In dropping the frames, the `mjpegSegment` also interpolates the display times of frames adjoining the dropped frame so that, although the frames were dropped, on the time scale no time gaps appear on the stream.

3.1.2 Play Object

In the application code we used, the play object checks and discards frames that are too late to be displayed, and then displays them in order. The video play object provides hooks to select the appropriate frame out of a collection that has been handed over to itself in a buffer.

3.1.3 Device Object

These objects are transparent to the programmer and contain low level calls specific to the platform and they submit data to be played, as has been called by the play object. The `SparcAuDevice` pre-empts *leftover* audio to *re-synchronize* at a regular frequency.

3.1.4 Network Twins

The network twins, `packet-Source (pktSrc)` and `packet-destination (pktDest)` uses a *cyclic UDP* protocol, where it is responsible for its own retransmission of the lost/delayed packets. This protocol has been explained in detail, with experimental verification of appropriateness for CM transport on best effort networks, in [Smi].

4 Experimental Evaluation of Media Losses

As stated, the objective of our experiments is to measure the performance of CMT with respect to continuity losses in audio and video. Our performance evaluation, is carried out with respect to metrics set forth in Section 2.

4.1 Types of Experiments

We carried out two main experiments. In the first one the segment object and the play object reside on the same machine, which we refer to as *local* objects. In the second experiment they are distributed across two machines. By the very design of CMT, this setup requires having *network twins*, the *packet source* and the *packet destination*. In the nomenclature of [MPR97], they are the *local playback application model* and the *simple remote playback application model*, for which the stream flows are given in Fig. 2.

In our experiments we measured the performance of CMT with respect to selected metrics under varying processor loads and network loads. Accordingly, for local experiments we collected data under different processor loads, and for remote experiments we collected data under different network and processor loads. In repeating experiments, the processor loads were measured by `top`, which gives the number of processes waiting for the processor, i.e. the length of the waiting queue. The experiments were carried out only when the length of the processor queue had stabilized for the immediately preceding 1, 5 and 15 minutes. The numbers reported are these stable values. The network loads were measured by the use of a *network sniffer*, and the experiments were carried out under stable loads reported by the sniffer in terms of packets per second.

4.2 Experimental Parameters and Methodology

In all of our experiments, we displayed audio and video streams, of length 1600 frames each. We ran the local experiment with different processor loads of 0, 1 and 5, which we call *low*, *medium* and *high* loads. For network loads we ran the experiment with approximately 0, 200 and 2000 packets/second. For each load setting we repeated our experiment five times and collected average statistics over them.

Our experiments were carried out using two uni-processor Ultra Sparc 1's with 64 MB RAM each, connected by a 10 Mbps Ethernet. Our work stations were equipped with JPEG Parallax© cards, and we used the Motion-JPEG video format and the 8-bit SparcAu format for audio.

As stated in [MPR97], some care must be taken to ensure that the experimental methodology does not degrade the performance of the system, i.e. it is as *non-intrusive* as possible. In this respect, we ensured that the time recorded by each module has been taken from already existing variables, and written into global arrays that were allocated before the system started to run in a stable state. Furthermore, these arrays were written into files when the objects were destroyed, thereby incurring minimal overhead in maintaining performance statistics. We also eliminated some initial readings to ensure that any start-up overhead does not affect final measurements. Finally, in order to keep track of frame numbers, we have introduced frame numbers into header portions of video and audio frames. Statistics maintained were the frame number and the time of arrival at each relevant CM object.

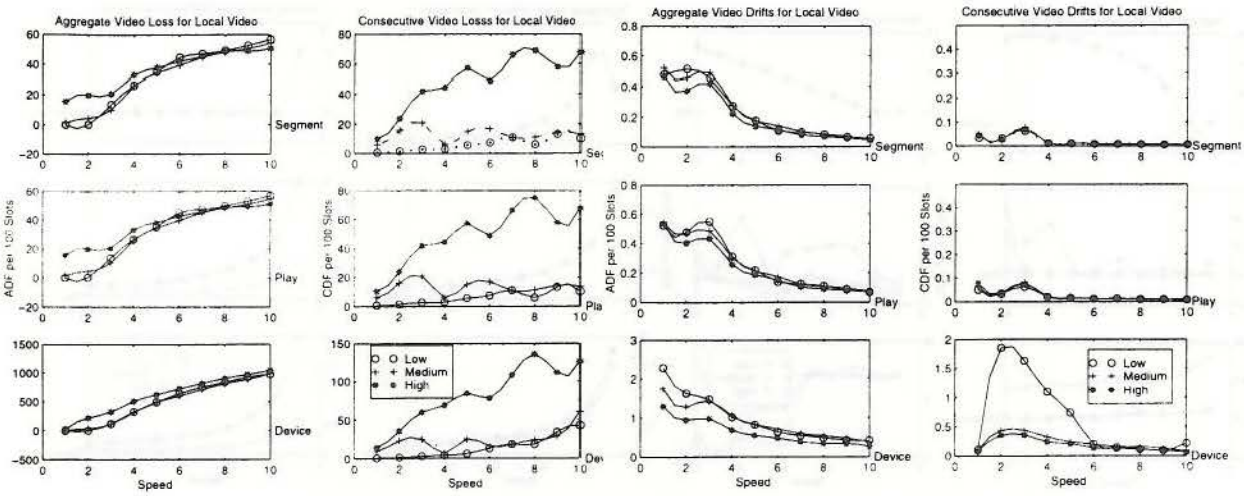

Figure 3: Video in Local Playback

5 The Local Experiment

This section presents results of the local experiment. i.e. the losses of audio and video streams where the `segment` and the `play` object reside on the same machine, and consequently there is no network involved. Our experiments were conducted with differing processor loads at speeds of 30 fps (speed = 1) to 300 fps (speed = 10) for both audio and video. The results are tabulated in terms of our metrics, i.e. aggregate and consecutive losses and drifts, for both media types.

Based on our experimental results we notice the following behavior of CMT.

1. The aggregate and consecutive frame losses increase with respect to speed for all video objects and audio segments.

2. Higher loads affect aggregate losses at low speeds and consecutive losses at higher speeds for all video objects.

3. Processor loads have no effect on audio at the `segment`. We believe that this is due to two conditions: (1) Audio data is relatively small. (2) There is no *delay based drops* for the audio stream at the `segment`, whereas in the video `segment`, drops are based on an *inverse binary order schema.*

4. At higher loads, `play` and `device` objects drop more frames than `segment` object. This is due to the fact. that they are invoked late, thereby drop frames.

5. As the speed increases, drifts drop. That means, as the speed increases, there are fewer frames present, but the ones that aren't dropped appear in time within their slot.

6. At higher loads and low speeds, audio `device` and `play` show larger drifts, indicating jitter.

6 Remote Experiment

This section presents results of our networked experiment, i.e. the losses of audio and video streams where the `segment` and the `play` object reside on two different machine connected by a 10 Mbps

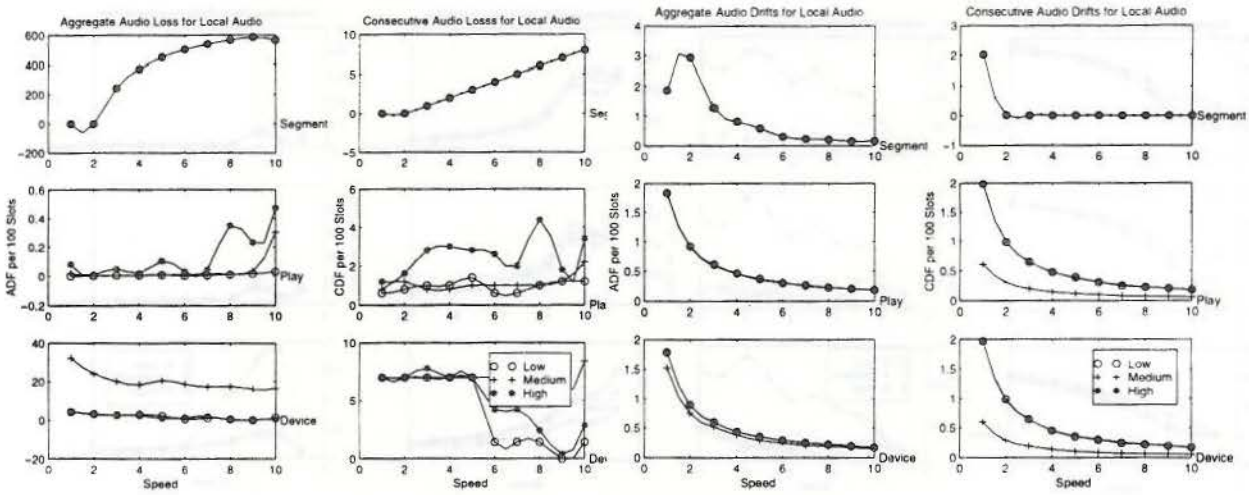

Figure 4: Audio in Local Playback

Ethernet. Accordingly, our experiments were conducted with differing processor loads and network load at speeds of 30 fps (speed = 1) to 300 fps (speed = 10) for both audio and video data.

6.1 Processor Loads

In this experiment we subjected both the server and the client to varying processor loads, measured in terms of processes waiting for the processor on a 120 MHz Sparc 1 processor. The results are tabulated in terms of our metrics, aggregate and consecutive losses and drifts for both media types.

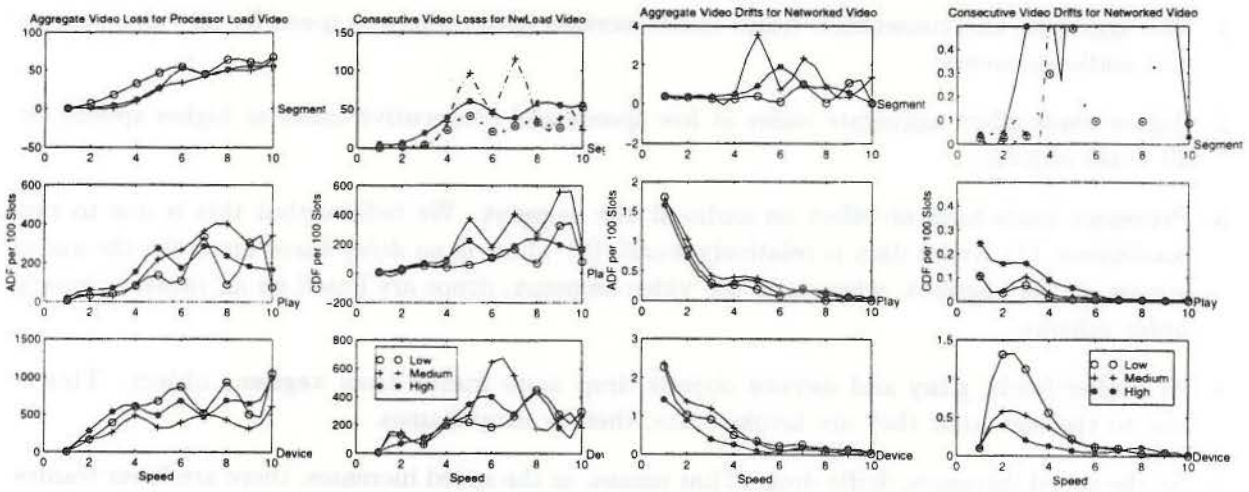

Figure 5: Video in Networked Playback

Based on our experimental results we notice the following behavior of CMT.

1. In networked video play, as in local play, playout speed increases aggregate and consecutive drops. However the variation of drops appears more wide-spread.

2. In networked audio, the `segment` object is not affected by either the load or speed, and there is wide-spread variation of drops in `play` and `device` objects.

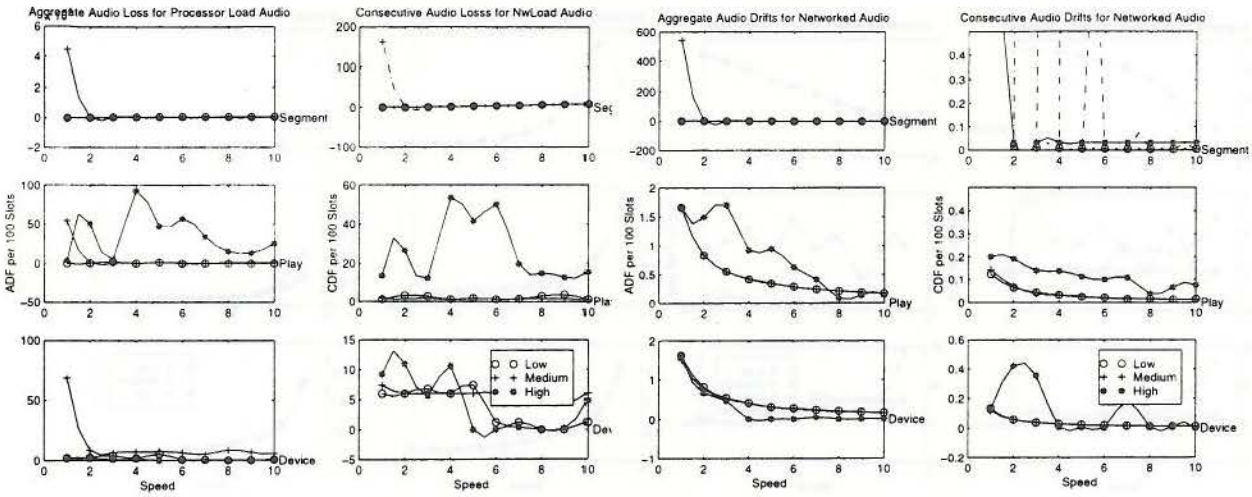

Figure 6: Audio in Networked Playback

6.2 Network Loads

In this experiment we subjected the network connecting the server and the client, to varying network loads. (measured in terms of packets per second on a 10Mbps Ethernet) between 0 to 20,000 packets/second, measured by an external *sniffer*. The results are tabulated in terms of our metrics, aggregate and consecutive losses and drifts, for both media types.

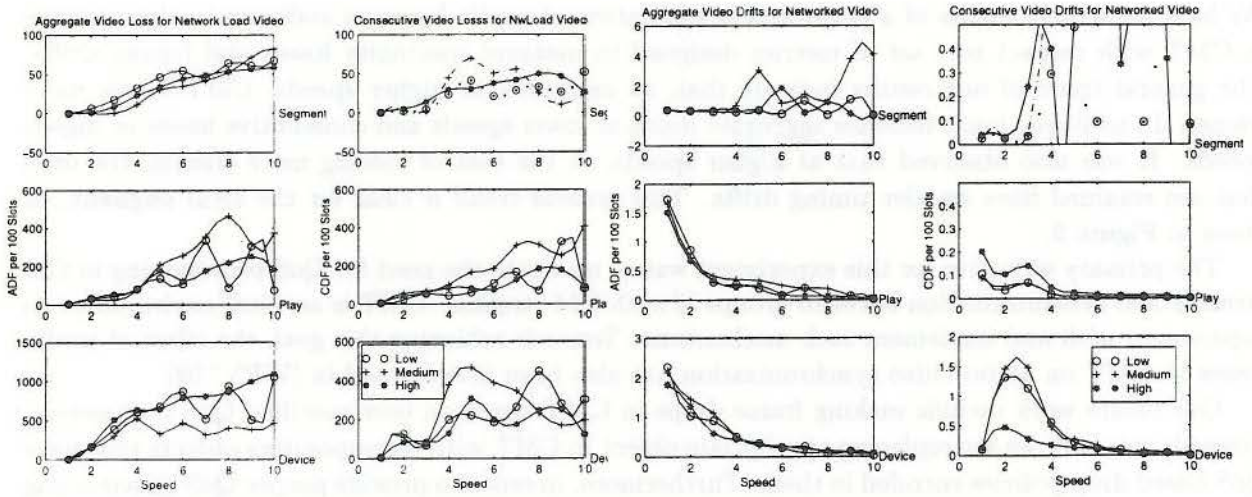

Figure 7: Video in Remote Playback against Network Loads

Based on our experimental results we notice the following behavior of CMT.

1. As in the case of processor loads, we noticed that increasing speed results in dropping frames. Our results show that in the given load ranges, the effect of network loads resulted in wide variations in drop rates.

2. In the audio `segment`, independent of the load range, increased speed result in more audio drops, leading to substantial packet drops at the remote `play` object. This was due to the lateness of audio frames.

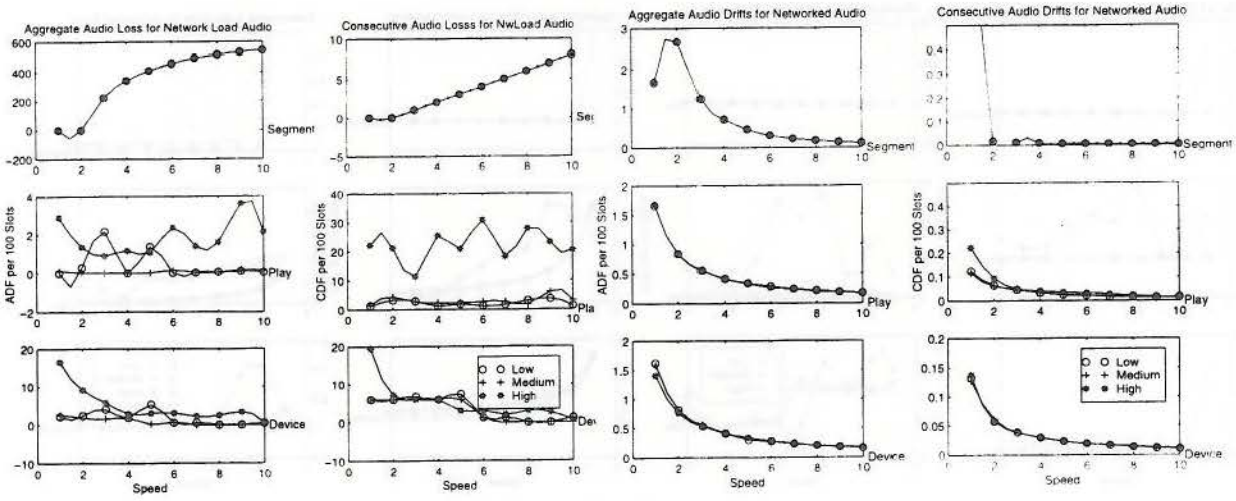

Figure 8: Audio in Networked Playback against Network Loads

3. The drifts decreased monotonically, and independently of network loads, indicating that the frames that were present did appear without much jitter, although as drops indicate many of them were not displayed.

7 Conclusions

We have presented results of a performance evaluation of media losses in audio and video streams in CMT with respect to a set of metrics designed to measure continuity losses and timing drifts. The general trend of our results indicate that, as expected, at higher speeds, CMT looses more frames. In addition, loads increase aggregate losses at lower speeds and consecutive losses at higher speeds. It was also observed that at higher speeds, at the cost of loosing more frames, the ones that are retained have smaller timing drifts. This general trend is clear for the local `segment`, as given in Figure 9.

The primary objective for this experiment was to motivate the need for QoS provisioning in CM streams and synchronization between groups of such CM streams. CMT is an ideal environment to experiment with and implement such mechanisms. Towards achieving this goal, the effect of media losses in CMT on audio-video synchronization has also been investigated in [WPV+98].

Our future work include making frame drops in CMT based on user specified QoS parameters. Towards this end, we are replacing appropriate object in CMT with corresponding objects that have QoS based drop policies encoded in them. Furthermore, in order to provide proper QoS provisioning at play objects, it is necessary for clients (`play` objects) to have a feed back mechanism toward the server (`segment`) objects.

References

[MPR97] Ketan Mayer-Patel and Lawrence A. Rowe. Design and Performance of the Berkeley Continuous Media Toolkit. In Martin Freeman, Paul Jardetzky, and Harrick Vin, editors, *Proceedings of Multimedia Computing and Networking*, pages 194–206, 1997.

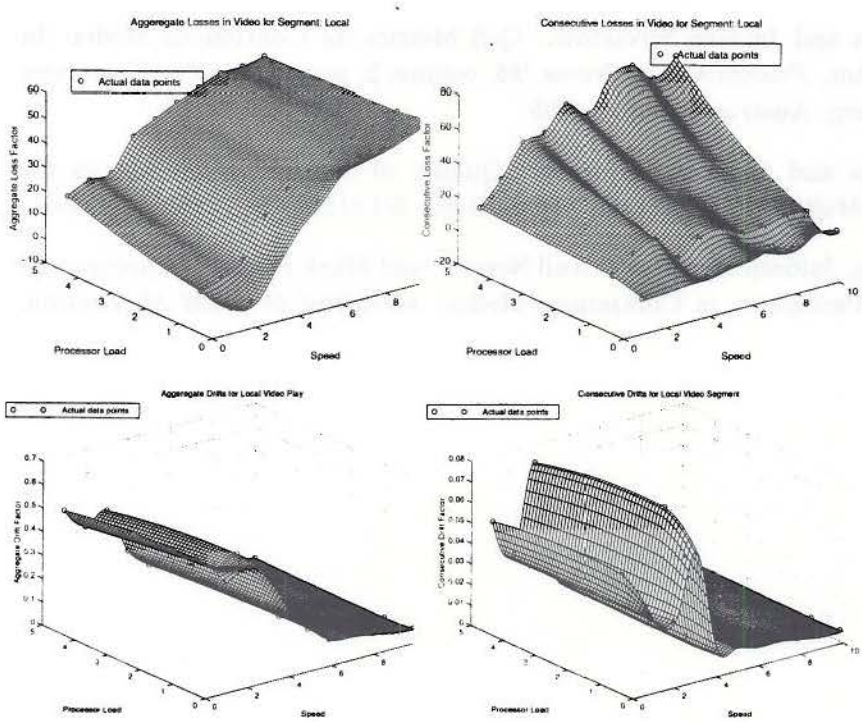

Figure 9: The General Trend: Seen from Local Video Segments

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, Massachusetts, 1 edition, 1994.

[SB96] Ralf Steinmetz and Gerold Blakowski. A media synchronization survey: Reference model, specification and case studies. *IEEE Journal on Selected Areas in Communication*, 14(1):5–35, 1996.

[Smi] Brian Smith. Cyclic-UDP: A Priority Driven Best-Effort Protocol. http://www.cs.cornell.edu/Info/Faculty/Brian_Smith.html/Publications.

[SNL95] Brian K. Schmidt, Duane Northcutt, and Monica Lam. A method and apparatus for measuring media synchronization. In T.D.C. Little, editor, *NOSDAV 95*, pages 190–201, September 1995.

[SRS93] Brian Smith, Lawrence A. Rowe, and Yen S. Tcl distributed programming. In *Proceedings of the 1993 Tcl/Tk Workshop*, 1993.

[SRY93] Brian Smith, Larry Rowe, and S Yen. A Tcl/Tk Continuous Media Player. In *Proceedings Tcl 1993 Workshop*, June 1993.

[Wel95] Brent B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall, New Jersey, 1 edition, 1995.

[WPV+98] Duminda Wijesekera, Shwetal Parikh, Srivatsan Varadarajan, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Performence Evaluation of Synchronization Losses in the Continuous Media Toolkit. In Harik Vin, editor, *submitted to IEEE Multimedia - 1998*, volume 2, University of Texas, Austin, October 1998.

[WS95] Duminda Wijesekera and Jaideep Srivastava. QoS Metrics for Continuous Media. In Ilka Miloucheva, editor, *Proceedings of Proms '95*, volume 2, pages 269–289, University of Salszberg, Salszberg, Austria, October 1995.

[WS96] Duminda Wijesekera and Jaideep Srivastava. Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications*. 3(1):127–166. September 1996.

[WSNF97] Duminda Wijesekera, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Experimental Evaluation of Loss Perception in Continuous Media. *submitted to ACM Multimedia*, July 1997.