A Multi-Threshold Online Smoothing Technique for Variable Rate Multimedia Streams *

Roger Zimmermann, Kun Fu, Mehrdad Jahangiri and Cyrus Shahabi
Integrated Media Systems Center and
Computer Science Department
University of Southern California
Los Angeles, California 90089
*(*rzimmerm,kfu,jahangir,shahabi@usc.edu*)*

Abstract. Variable bit rate (VBR) compression for media streams allocates more bits to complex scenes and fewer bits to simple scenes. This results in a higher and more uniform visual and aural quality. The disadvantage of the VBR technique is that it results in bursty network traffic and uneven resource utilization when streaming media. In this study we propose an online media transmission smoothing technique that requires no a priori knowledge of the actual bit rate. It utilizes multi-level buffer thresholds at the client side that trigger feedback information sent to the server. This technique can be applied to both live captured streams and stored streams without requiring any server side pre-processing. We have implemented this scheme in our continuous media server and verified its operation across real world LAN and WAN connections. The results show smoother transmission schedules than any other previously proposed online technique.

Keywords: Continuous Media Delivery, Continuous Media Servers, Smoothing, Video on Demand

1. Introduction

Many multimedia applications, such as news-on-demand, distance learning, and corporate training, rely on the efficient transfer of pre-recorded or live multimedia streams between a server and a client. These media streams are captured and displayed at a predetermined rate. For example, video streams may require a rate of 24, 29.97, 30, or 60 frames per second. Audio streams may require 44,100 or 48,000 samples per second. An important measure of quality for such multimedia communications is the precisely timed playback of the streams at the client location.

Achieving this precise playback is complicated by the popular use of variable bit rate (VBR) media stream compression. VBR encoding algorithms allocate more bits per time to complex parts of a stream and fewer bits to simple parts to keep the visual and aural quality at near constant levels. For example, an action sequence in a movie may require more bits per second than the credits that are displayed at the end. As a result, different transmission rates

* This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), and IIS-0082826, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash/equipment gifts from NCR, IBM, Intel and SUN.

may be required over the length of a media stream to avoid starvation or overflow of the client buffer. As a contradictory requirement we would like to minimize the variability of the data transmitted through a network. High variability produces uneven resource utilization and may lead to congestion and exacerbate display disruptions.

The focus of this study is on achieving high quality media playback by reducing the variability of the transmitted data and hence avoiding display disruptions due to data starvation or overflow at the client. We propose a novel technique that adjusts the multimedia traffic based on an end-to-end rate control mechanism in conjunction with an intelligent buffer management scheme. Unlike previous studies, we consider multiple signaling thresholds and adaptively predict the future bandwidth requirements. With this *Multi-Threshold Flow Control* (MTFC) scheme, VBR streams are accommodated without a priori knowledge of the stream bit rate. Furthermore, because the MTFC algorithm encompasses server, network and clients, it adapts itself to changing network conditions. Display disruptions are minimized even with few client resources (e.g., a small buffer size).

To verify the effectiveness of the MTFC protocol we have implemented it in our *Yima* [31, 40] continuous media server and clients. The Yima server is based on a scalable cluster design. Each cluster node is an off-the-shelf personal computer with attached storage devices and, for example, a Fast Ethernet connection. The server software manages the storage and network resources to provide real-time service to the various clients that are requesting media streams. The clients run on either Windows or Linux and utilize either a hardware or software decoder to display media streams. We have implemented a number of different clients that support media of varying display bandwidths from less than 1 Mb/s (e.g., MPEG-4) to more than 20 Mb/s (e.g., HDTV).

For this study we have conducted extensive real world experiments in both LAN and WAN network environments, with a range of buffer sizes and prediction windows. The test video streams were of DVD quality with a consumption rate of approximately 6 Mb/s. We have further compared MTFC with two recently published online rate control algorithms [20, 14]. The experimental results show that our scheme outperforms these algorithms in terms of traffic smoothness with similar or less signaling frequency. We summarize the contributions of our work as follows.

- We present a flexible multi-threshold buffer model that incorporates various threshold spacing strategies: equi-distant, arithmetic and geometric spacing. We evaluate the performance of the different threshold spacing strategies with real world experiments.

- We introduce a consumption prediction component that employs prediction algorithms based on: a window-based average consumption rate, a

window-based exponential average and a fuzzy exponential average. We evaluate the efficiency of these prediction algorithms with simulations.

– We designed a modular rate change computation framework in which new consumption prediction algorithms and feedback delay estimation algorithms can be easily incorporated.

The remainder of this paper is organized as follows. In Section 2 we review the related work. Section 3 describes our approach to the proposed rate-control scheme. In Section 4 we present the results of our extensive performance evaluation. Finally, Section 5 concludes the paper.

2. Related Work

A number of studies have investigated the transmission of multimedia streams when faced with variable bit rate streams and/or a constrained client buffer size. The techniques can be classified into two groups: *server-controlled* and *client-controlled*.

Server-controlled algorithms generally pre-compute a *transmission schedule* for a media stream based on a substantial knowledge of its rate requirements. The variability in the stream bandwidth is smoothed by computing a transmission schedule that consists of a number of constant-rate segments. The segment lengths are calculated such that neither a client buffer overflow nor an underflow will occur. The various approaches differ in the optimization criteria that they choose. Some examples are as follows: minimizing the number of rate changes in the transmission schedule, minimizing the utilization of the client buffer, minimizing the peak rate, or minimizing the number of on-off segments in an on-off transmission model. A good summary of the different techniques can be found in [2]. All these techniques require that complete or partial traffic statistics are known a-priori.

These offline server-controlled techniques have several disadvantages. They do not work with live streams where only a limited rate history is available. They cannot adjust to changing network conditions and they may get disrupted when users invoke interactive commands such as pause, rewind, and fast forward.

Consequently, client-controlled techniques seem more appropriate in a dynamic environment. Involving the client has the drawback of feedback overhead and response delays (for a discussion see [5]). However, there are several significant advantages. For example, such a technique adapts itself to changing network conditions. In addition, the server does not need to be aware of the content format of the stream, which results in a simpler and more flexible architecture. Therefore, new media types such as haptic data can automatically be supported without modification of the server software [30].

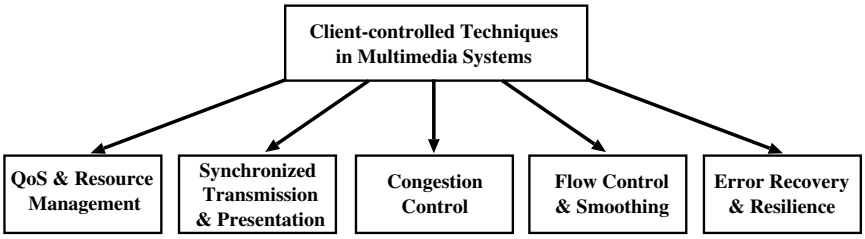


Figure 1. Classification of applications of client controlled feedback techniques in multimedia systems.

A number of studies have investigated client-side feedback control techniques in multimedia systems. These previous studies can be classified into the following groups according to their applications (as shown in Fig. 1):

- QoS and Resource Management [22, 4, 34, 36, 32, 33].
- Synchronized Transmission and Presentation [27, 26, 13, 7, 35].
- Congestion Control [11, 1, 12, 10].
- Flow Control and Smoothing [20, 14, 15, 21].
- Error Recovery and Resilience [25, 24, 28, 19, 39, 9, 23, 8].

Only a handful of studies [20, 14, 15, 21] have focused on client-side feedback control *smoothing* techniques. In paper [21], the authors proposed a feedback rate control technique in best effort mobile packet networks. The proposed technique is quite similar to the approach introduced in [20]. Both techniques trigger rate changes based on two pre-configured thresholds at the receiver buffer. However, in [21] the authors applied the flow control between the gateway and the base station router in stead of between the streaming server and client as in [20] (to avoid buffer underflow and overflow conditions.) In Section 4.4 we will compare our technique with two approaches proposed in [14, 20] ([14] is an extension of [15] and we consider them the same technique).

3. Multi-Threshold Flow Control (MTFC) Technique for Stream Smoothing

The design of a new rate control algorithm was motivated in part by our continuous media server implementation. We required an adaptive technique that could work in a real-world dynamic environment with minimal prior knowledge of the multimedia streams to be served. We identified the following desirable characteristics for our new algorithm:

- **Online operation**: Required for live streaming and also desirable for stored streams.

Table I. List of terms used repeatedly in this study and their respective definitions.

Term	Definition	Units
B	Buffer size	bytes
m	Number of threshold levels	
TH_i	i-th Threshold, $TH_i > TH_{i-1}$	bytes
TH_O	Overflow protection threshold	bytes
TH_U	Underflow protection threshold	bytes
TH_R	Resume threshold, e.g. $\frac{B}{2}$	bytes
TH_N	Buffer target level, e.g. $\frac{B}{2}$	bytes
$\triangle t_{obsv}$	Sampling interval	seconds
$t_{obsv}(i)$	i-th observation point	
$t_{ctrl}(i)$	i-th feedback control point	
w_{obsv}	Observation window size, i.e. the number of observation points	
R^S	Observed server transmission rate window, $\langle r^s_1, r^s_2, \ldots, r^s_{w_{obsv}} \rangle$	
r^s_i	i-th latest observed server transmission rate, $r^s_{w_{obsv}}$ is the latest observed server transmission rate	bytes/second
R^C	Observed client consumption rate window, $\langle r^c_1, r^c_2, \ldots, r^c_{w_{obsv}} \rangle$	
r^c_i	i-th latest observed client consumption rate, $r^c_{w_{obsv}}$ is the latest observed client consumption rate	bytes/second
B_{obsv}	Observed client buffer status window, $\langle b_1, b_2, \ldots, b_{w_{obsv}} \rangle$	
b_i	i-th latest observed client buffer status, $b_{w_{obsv}}$ is the latest observed client buffer status	bytes
w_{pred}	Prediction window size, i.e. the number of predicted points	
$\hat{R}$	Predicted client consumption rate window, $\langle \hat{r}_1, \hat{r}_2, \ldots, \hat{r}_{w_{pred}} \rangle$	
$\hat{r}_i$	i-th predicted future client consumption rate (consumption rate at time $t_{obsv}(w_{obsv}) + \triangle t_{obsv} \times i$), and $\hat{r}_{w_{pred}}$ is the furthest predicted future client consumption rate	bytes/second
r_{new}	The computed new server transmission rate	bytes/second
w_{fcd}	Observation feedback control delay window size	
$SCR[i]$	i-th latest smoothed consumption rate	bytes/second
$\hat{r}$	Predicted client consumption rate	bytes/second
α_{cr}	Client consumption rate predict parameter	
$t_{feedback}$	Feedback control delay	second
r_t	Observed client consumption rate at time t	bytes/second
$\hat{r_t}$	Predicted client consumption rate at time t	bytes/second
C	Consumption component, i.e., data consumed during the prediction window	bytes
$\triangle r$	Rate change	
μ	Average consumption rate	bytes/second
σ	Standard deviation of consumption rate	

– **Content independence**: An algorithm that is not tied to any particular encoding technique will continue to work when new compression algorithms are introduced.

– **Minimizing feedback control signaling**: The overhead of online signaling should be negligible to compete with offline methods that do not need any signaling.

– **Rate smoothing**: The peak data rate as well as the number of rate changes should be lowered compared with the original, unsmoothed stream. This will greatly simplify the design of efficient real-time storage, retrieval, and transport mechanisms to achieve high resource utilization [37, 29].

Considering these objectives, we designed our novel Multi-Threshold Flow Control (MTFC) technique. It distinguishes itself from previously proposed algorithms by incorporating the following: (1) a multi-threshold buffer model, (2) a consumption prediction component and (3) a modular rate change computation framework in which new consumption prediction algorithms and feedback delay estimation algorithms can easily be incorporated.

The client playout buffer is a crucial component of any feedback control paradigm. The server is the data producer that places data into the buffer while the media decoder is the consumer that retrieves data. If the production and consumption rates are exactly the same then the amount of data in the buffer does not change. If there is a mismatch, however, data will either accumulate or drain. If the buffer overflows or underflows then display disruptions will appear. Hence, the goal of managing the buffer is to keep the data level approximately at half the buffer size such that fluctuations in either direction can be absorbed. In an online feedback scheme, when the data level sufficiently deviates from the buffer midpoint, a correction message is sent to the server to adjust the sending rate. We will first describe our buffer management scheme, then elaborate on the rate adjustment calculation, and finally explain our consumption prediction component.

3.1. Multi-Threshold Buffer Model

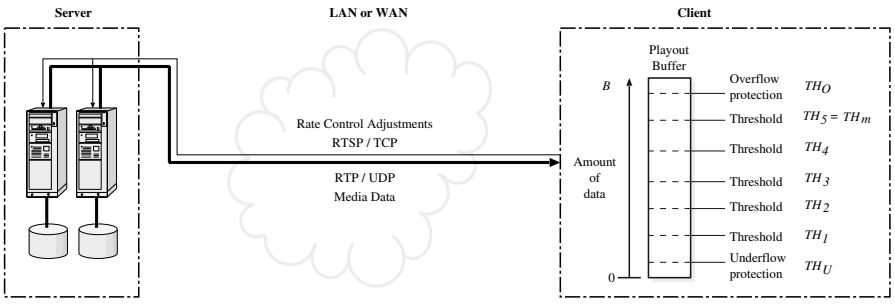


Figure 2. Components of our streaming media architecture experimental setup.

When placing thresholds within the client buffer there are two parameters that need to be chosen: (1) the number of thresholds (denoted m) and (2) the spacing between multiple thresholds.

We have investigated the following spacing policies: equi-distant, based on an arithmetic series, and based on a geometric series. First, we will describe the system behavior of the multi-threshold buffer model with an equi-distant threshold spacing strategy. Subsequently, we will discuss the details of the other two approaches.

3.1.1. *System Behavior with Different Thresholds*

As an example, Figure 2 illustrates how the client buffer (size B) is segmented with m equally spaced thresholds TH_i ($m = 5$ in the example) where TH stands for threshold. Table I lists all the terms used in this study. Additionally there exists an *overflow protection* threshold TH_O and an *underflow protection* threshold TH_U. The thresholds which are closest to the protection thresholds TH_O and TH_U are also called warning thresholds (TH_m and TH_1 in the example). Equation 1 illustrates the calculation of the thresholds with 5% protection levels.

$$\begin{aligned} TH_U &= 0.05 \times B \\ TH_O &= 0.95 \times B \\ TH_i &= TH_U + i \times \frac{TH_O - TH_U}{m+1} \quad \text{for } 1 \leq i \leq m \end{aligned} \tag{1}$$

Whenever a threshold TH_i is crossed, a new server sending rate is calculated (details are described in the next section) and a $\triangle r$ rate adjustment is sent to the server through an RTSP feedback command. If the TH_O overflow or TH_U underflow protection thresholds are crossed then the rate adjustment is more aggressive. In case of TH_O, the server is paused (i.e., the sending rate is set to zero). It remains paused until the buffer level reaches the resume threshold TH_R, e.g., the midpoint. In case of TH_U, the sending rate is increased to one and a half times the average sending rate. The maximum rate of 1.5 times the average rate (μ) was chosen because we empirically observed that even highly variable streams typically cross this limit for less than 5% of the playback time. Fig. 3**(a)** illustrates the consumption rate distribution for the movie "Twister." (Note that the movie consumption rate is measured periodically at 1 second intervals). Similarly, the consumption rate distribution for two other movies encoded with MPEG-4 and HDTV MPEG-2 are shown in Figs. 3(b) and (c), respectively. Table II provides the numerical consumption rate statistics for the three sample clips. The statistical data shows that 11% of the time the data rate is outside of $\mu \pm 0.5\mu$ for this particular example of MPEG-4. In fact, using a maximum and a minimum rate of $\mu \pm 2\sigma$ (σ denotes the standard deviation) is a safer choice because the

data rate stays within this limit for more than 95% of the movie duration for all three sample movie files.

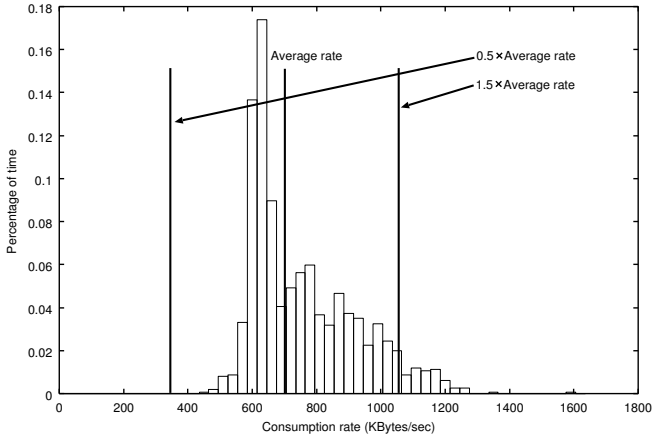


Fig. 3(a): DVD movie "Twister."

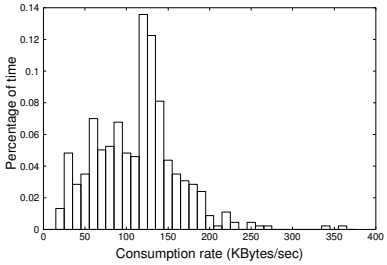


Fig. 3(b): **MPEG-4 movie.**

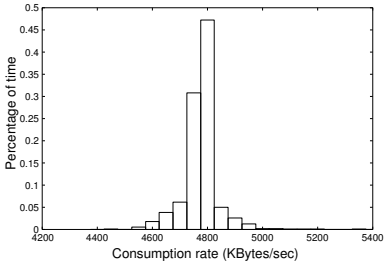


Fig. 3(c): **HDTV MPEG-2 movie.**

Figure 3. Consumption rate distribution for **different movie types**.

Table II. **Consumption rate characteristics for sample media format.**

Movie Type	μ[1] [Bytes/sec]	σ^2 [Bytes/sec]	$\mu \pm 0.5\mu$[3]	$\mu \pm 2\sigma$[4]
MPEG4	107753.1	49193.0	11.0%	2.76%
DVD	693744.9	138442.3	1.7%	4.85%
HDTV	4752540.8	83658.9	0.0%	1.44%

[1] μ is the average consumption rate of the movie

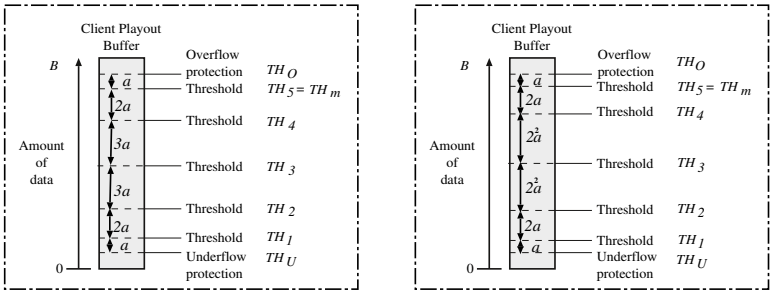
[2] σ denotes the standard deviation of the movie consumption rate

[3] $\mu \pm 0.5\mu$ denotes the percentage of the movie duration that has a consumption rate $\notin [0.5\mu, 1.5\mu]$

[4] $\mu \pm 2\sigma$ represents the percentage of the movie duration that has a consumption rate $\notin [\mu - 2\sigma, \mu + 2\sigma]$

The TH_O and TH_U buffer limits are very rarely reached when the rate adjustment calculation is done properly and these aggressive actions are invoked only to avoid display disruptions.

3.1.2. *Threshold Spacing Strategies*



(a) Arithmetic spacing **(b) Geometric spacing**

Figure 4. Two non-uniform threshold spacing strategies.

Our MTFC have with three different threshold placement strategies: (1) equi-distant, (2) based on an arithmetic series (*arithmetic spacing*), and (3) based on a geometric series (*geometric spacing*). Equi-distant spacing has already been discussed in the previous section.

Figs. 4(a) and (b) illustrate how the client buffer is segmented by m thresholds ($m = 5$) with the arithmetic and geometric spacing strategies. Note that a spacing parameter α appears in both figures. α is determined by Equation 2 for geometric spacing and by Equation 4 for arithmetic spacing. Both equations use the same variable i, defined by Equation 3.

$$\alpha = \begin{cases} \frac{TH_O - TH_U}{2^{i+2} - 2^i - 2} & \text{if } m \text{ is even} \\ \frac{TH_O - TH_U}{(2^i - 1) \times 2} & \text{otherwise} \end{cases} \tag{2}$$

$$i = \left\lceil \frac{m}{2} \right\rceil \tag{3}$$

$$\alpha = \begin{cases} \frac{TH_O - TH_U}{(i+1)^2} & \text{if } m \text{ is even} \\ \frac{TH_O - TH_U}{i \times (i+1)} & \text{otherwise} \end{cases} \tag{4}$$

3.2. Rate Change ($\triangle r$) Computation

The server sending rate, the decoder consumption rate, and the buffer level are sampled every $\triangle t_{obsv}$ time instance. If the buffer level $b_{w_{obsv}}$ crosses any of the thresholds TH_i, a new server sending rate is computed based on Equation 5. It combines the following components. The target buffer level TH_N is usually

set to the buffer midpoint. The current buffer level is described by $b_{w_{obsv}}$. The expected duration $w_{pred} \times \triangle t_{obsv}$ is the time to recover the current buffer level back to the target buffer level TH_N. This parameter is adjustable and we will discuss its impact on the system performance in Section 4. The amount of data consumed during $w_{pred} \times \triangle t_{obsv}$ is C, as predicted with Equation 6.

$$r_{new} = \frac{TH_N - b_{w_{obsv}} + C - r^s_{w_{obsv}} \times t_{feedback}}{w_{pred} \times \triangle t_{obsv} - t_{feedback}} \tag{5}$$

with

$$C = \sum_{i=1}^{w_{pred}} (\hat{r}_i \times \triangle t_{obsv}) \tag{6}$$

We investigated several different prediction algorithms (more on this later). The last component is the round-trip feedback message delay $t_{feedback}$. In Equation 6, the consumption prediction component is denoted $\hat{r}_i$, the future consumption rate at time $t_{obsv}(w_{obsv}) + \triangle t_{obsv} \times i$. $\hat{r}_i$ is estimated using one of the three prediction algorithms discussed in Section 3.3. When crossing the thresholds TH_m and TH_1, a rate change of $\triangle r$ may not be enough to avoid reaching the overflow or underflow protection thresholds, TH_O or TH_U. This is due to the error margin of the prediction algorithms. To decrease the possibility of crossing the protection thresholds, a dynamically calculated mean absolute percentage error (MAPE) is added or subtracted from $\hat{r}_i$ as shown in Equation 7. The error is computed using Equation 8 where N is the number of prediction samples up to the current prediction time.

$$\hat{r}_i = \begin{cases} \hat{r}_i \times (1 - MAPE) & \text{if } TH_m \text{ is reached} \\ \hat{r}_i \times (1 + MAPE) & \text{if } TH_1 \text{ is reached} \end{cases} \tag{7}$$

$$\text{MAPE} = \frac{\sum_{t=1}^{N} |r_t - \hat{r}_t|}{N} \quad (N \text{ is the total number of predictions}) \tag{8}$$

In a final step the rate change $\triangle r$ is determined from the new server sending rate r_{new} and the current server sending rate $r^s_{w_{obsv}}$:

$$\triangle r = 1 - \frac{r_{new}}{r^s_{w_{obsv}}} \tag{9}$$

3.3. Consumption Rate Prediction

Complete knowledge of the future consumption rate of a stream enables the client buffer management to be optimal. However, one of our objectives was not to assume any a-priori knowledge of the stream bandwidth requirements in order to support live streams. Consequently, we studied several prediction algorithms to approximate the optimal case as closely as possible without excessive computational complexity.

3.3.1. *Window-Based Prediction Algorithms*

Prediction algorithms commonly estimate future values based on information collected within a past time window. We investigated several algorithms that observe the w_{obsv} most recent rate samples to predict w_{pred} samples into the future. We use the following notation: observation window R^C with samples $\langle r^c_1, r^c_2, \ldots, r^c_{w_{obsv}} \rangle$ and prediction window $\hat{R}$ with samples $\langle \hat{r}_1, \hat{r}_2, \ldots, \hat{r}_{w_{pred}} \rangle$. The estimated future rate is denoted $\hat{r}$. We evaluated three different prediction algorithms: (a) an average consumption rate algorithm, (b) an exponential average algorithm, and (c) a fuzzy exponential average algorithm. We will now describe each one in turn.

With the first approach the *average consumption rate* of the observation window R is used to predict the average consumption rate throughout the prediction window $\hat{R}$ as shown in Equation 10.

$$\hat{r} = \frac{\sum_{i=1}^{w_{obsv}} r^c_i}{w_{obsv}} \tag{10}$$

The second approach is based on an exponential average algorithm similar to the TCP round-trip time estimation algorithm [16]. The predictor is characterized by a parameter α_{cr}, which could be intuitively interpreted as the weight given to the samples in the past time window. Here, recent samples are given more weight than older samples. The exact weight can be adjusted with the parameter α_{cr} as shown in Equation 11. Note that $SCR[1]$ is just the initialization, $SCR[i]$ are intermediate results, and $\hat{r}$ is the next predicted consumption rate in the prediction window $\hat{R}$.

$$\begin{aligned} SCR[1] &= r^c_1 \\ SCR[i] &= \alpha_{cr} \times SCR[i-1] + (1-\alpha_{cr}) \times r^c_{i-1} \\ \hat{r} &= SCR[w_{obsv}+1] \end{aligned} \tag{11}$$

There are two variations to apply this algorithm in forecasting the future consumption rates during the prediction window $\hat{R}$. These two approaches differ in the observation window size used to estimate $\hat{r}_i$. In the first approach, $\hat{r}_i$ is predicted based on an increasing window $\langle R^C, \hat{r}_1, \hat{r}_2, \ldots, \hat{r}_{i-1} \rangle$ using Equation 11. With the second, the prediction is based on a fixed, sliding window $\langle r^c_i, \ldots, r^c_{w_{obsv-1}}, r^c_{w_{obsv}}, \hat{r}_1, \hat{r}_2, \ldots, \hat{r}_{i-1} \rangle$. The first one increases the window size by one sample each time a new $\hat{r}_i$ is generated. The second approach keeps the window size constant and slides the observation window R^C forward when a new $\hat{r}_i$ is generated. Throughout the rest of the paper, we will call the first approach *expanding-window exponential average algorithm* and the second approach *sliding-window exponential average algorithm*.

The third approach, based on a fuzzy exponential average algorithm [18], combines a fuzzy logic controller with the window exponential average algorithm. The idea is to dynamically calculate the parameter α_{cr} used in the exponential average algorithm (see Fig. 5 for details).

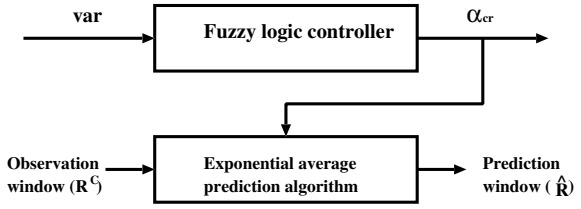


Figure 5. Schematic diagram for fuzzy exponential average algorithm.

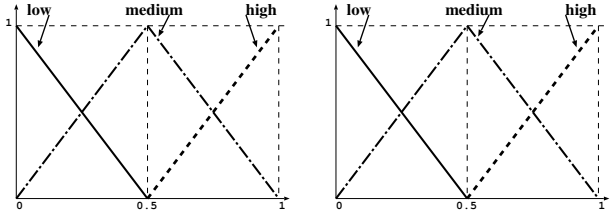


Fig. 6(a): var. Fig. 6(b): α_{cr}.

Figure 6. Membership functions for the two variables var and α_{cr}.

The parameter α_{cr} can be interpreted as representing the variability of the consumption rate in our system. To have a better prediction with a smooth stream, we want to give more weight to past samples, which means a larger α_{cr}. On the other hand, if the stream is very bursty, we would like to give more weight to recent samples, which means a smaller α_{cr} [5]. The input of the membership function for a fuzzy logic controller should be in the range of $[0, 1]$. Therefore, the variability of a stream is characterized by the normalized variance var calculated as:

$$var = min\left(1, \frac{\left| r^c_{w_{obsv}} - \frac{\sum_{i=1}^{w_{obsv}} r^c_i}{w_{obsv}} \right|}{\frac{\sum_{i=1}^{w_{obsv}} r^c_i}{w_{obsv}}}\right) \tag{12}$$

These ideas translate into the following fuzzy control rules: (1) if var is *low* then α_{cr} is *high*, (2) if var is *medium* then α_{cr} is *medium*, and (3) if var is *high* then α_{cr} is *low*. The shape and ranges of the fuzzy terms *low*, *medium* and *high* and the membership functions for the two variables var and α_{cr} are shown in Fig. 6.

[5] If a stream is smooth, then the window exponential average algorithm always generates a fairly accurate estimate. That means, the prediction error is small and α_{cr} should be large. In contrast, if the stream is bursty, the past history cannot predict the future well. In this case, it would be better to give little weight to the past history, and make α_{cr} small, so that the recent variability of the stream can be tracked.

3.3.2. *Simulation Results of Rate Prediction Algorithms*
We evaluated the effectiveness of the rate prediction algorithms with a simulation program. We chose two different MPEG-2 encoded movies as test cases. The first one was a 25 minute segment of the DVD movie "Twister," that exhibits a highly variable consumption rate, as shown in Fig. 8. The second one called "Football" is our own recording of a USC football game and it has a relatively smooth consumption rate. After we collected the consumption rate traces for these two movies we performed the three prediction algorithms on each of them. For the exponential average algorithm we chose the sliding window approach with either $\alpha_{cr} = 0.65$ or $\alpha_{cr} = 0.95$. The fuzzy exponential average algorithm we evaluated here is also based on the sliding-window exponential average algorithm. We used a number of different window sizes and measured the prediction error for each algorithm. The Mean Absolute Prediction Error (MAPE) [6] is used to evaluate the accuracy of the different prediction algorithms as shown in Equation 8.

Fig. 7 illustrates the simulation results. MAPE is shown as a function of the window size w_{obsv} (note that here $w_{obsv} = w_{pred}$). We plotted the results from our three rate prediction algorithms plus one additional reference curve that represents MAPE calculated with a priori knowledge of the average consumption rate. The error value for our smooth movie "Football" is consistently below 3% for all algorithms except the reference one, as illustrated in Figure 7(a). This is because our prediction algorithms keep track of the movie consumption rate changes much more closely than assuming the average consumption rate, which does not reflect any changes at all. For the movie "Twister," with its higher variability, the prediction error is between 15% and 20%. Here there is a visible performance difference between the various algorithms. The prediction algorithm that averages the consumption rate in the observation window outperforms all the others. One intuition is that it captures the consumption rate within the prediction window and recovers some prediction error by averaging. It is also the simplest measure to compute and therefore the least complex to compute in real-time. Because of its good performance we chose this algorithm for the rest of our study and implemented it in our prototype system.

4. Performance Evaluation

We integrated the MTFC technique into our operational distributed continuous media architecture which serves as the platform for testing the effectiveness of our algorithm. Fig. 2 illustrates our experimental setup. The server consists of two Pentium II 450 MHz PCs with 384 MB of memory. Each PC is connected to an Ethernet switch (model Cabletron 6000) in our laboratory via a 100 Mb/s network interface. Movies are striped over two 18 GB Seagate Cheetah

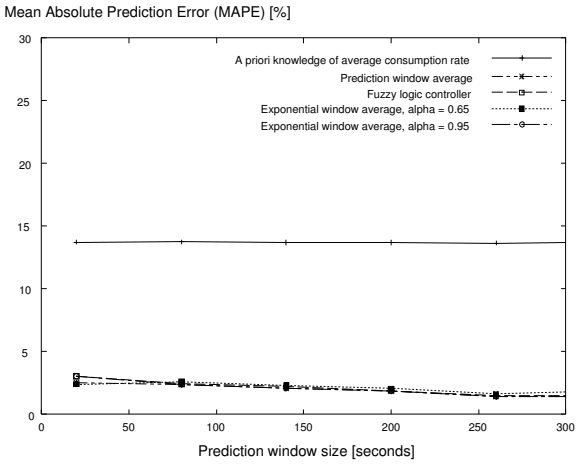


Fig. 7(a): MAPE for movie “Football.”

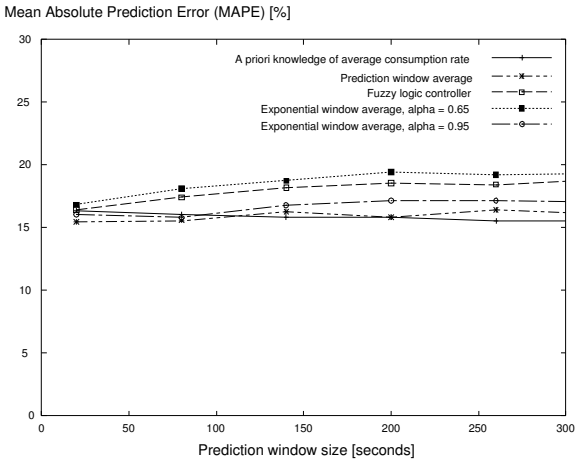


Fig. 7(b): MAPE for movie “Twister.”

Figure 7. The Mean Absolute Prediction Error (MAPE) of different prediction algorithms for two MPEG-2 encoded movies “Football” and ”Twister.”

disk drives (one per server node). The disks are attached through Ultra2 low-voltage differential (LVD) SCSI connections that can provide 80 MB/s throughput. Red Hat Linux 7.0 is used as the operating system for each server PC. The client is based on a Pentium III 933 MHz PC. The player software that includes the buffer management and rate prediction algorithms runs under Red Hat Linux 7.0. Table III lists the input parameters and corresponding values

Table III. Input Parameters used in the experiments.

Input Parameters	Configurations
Test movie "Twister"	MPEG-2 video (NTSC resolution), AC-3 audio
Average bandwidth	698594 bytes/sec
Length	25 minutes
throughput std. dev.	308283.8
Client buffer sizes (B)	8, 16, 32 MB
Number of thresholds (m)	2, 3, 5, 9, 17
Sampling interval ($\triangle t_{obsv}$)	1 second
Prediction window size (w_{pred})	45, 90, 180 samples
	i.e. 45, 90, 180 seconds

Table IV. Parameters measured in the experiments.

Measurement Parameters	Definitions
x	Throughput standard deviation
y	Number of rate adjustments
z	Number of times protection threshold crossed
u	Number of buffer overflow or underflow
v	Overhead of rate adjustments(ratio of message size over total transmitted data size)

that we used in our experiments[6] Note that we chose a one second sampling interval since it is small enough to capture the system status (e.g., consumption rate fluctuations) without utilizing too much of the system resources (e.g., CPU and memory). Table IV lists the all the parameters we measured in our experiments.

The round-trip feedback message delay ($t_{feedback}$ in Equation5) is a very important factor. In the current implementation, it is manually configured as a conservatively estimated constant delay for our experimental setup. However, the algorithm could be extended with a dynamically estimated value based on a prediction algorithm [3, 17] to adapt to various network conditions (note that estimation of RTT is not the focus of this paper). To be more general, we conducted the experiments with two different types of networks: (1) a LAN

[6] The movie "Twister" used in our original experiments was obtained from a DVD. DVD movies are usually optimized for constant visual quality with little concern for any bandwidth changes (because playback is local). This means they generally exhibit extremely variable bit rates.

Table V. End-to-end route from one Yima client (located at USC campus, Los Angeles) to the Yima server (Metromedia Fiber Network data center, El Segundo, CA). The distance between these two is approximately 1000 km because two of the routers are located in San Jose, CA.

Hop#	Router
1	imsc-gw (128.125.163.254)
2	rtr-gw-43 (128.125.254.43)
3	c2-12008 (128.125.251.241)
4	ISI–USC.POS.calren2.net (198.32.248.26)
5	ISI-7507–ISI.POS.calren2.net (198.32.248.22)
6	mae-la.above.net (198.32.146.21)
7	sjc1-lax1-oc3.sjc1.above.net (216.200.0.166)
8	core1-core6-oc12.sjc1.above.net (64.125.31.29)
9	core2-sjc1-oc48.sjc2.above.net (208.184.102.26)
10	lax3-sjc2-oc48.lax3.above.net (208.184.232.138)
11	main1colo34-core1-oc48.lax3.above.net (208.185.175.230)
12	64.124.204.8.ismc.edu (64.124.204.8)

where the server and client are directly connected through a Fast-Ethernet switch and the round-trip time (RTT) is usually less than 1 ms, and (2) the public Internet, where the RTT is around several hundred ms. Table V shows the data route between the client and server. Although the geographical distance between the two end points is less than 40 km, a data stream travels more than 1000 km due to the Internet topology. Since the MTFC technique adapts to the network conditions, the results from these two setups are very similar: they follow the same trend and differ from each other by less than 3% of the throughput standard deviation. Thus, we only report the LAN result set.

Before we discuss our results in detail we would like to illustrate the effectiveness of our MTFC algorithm with an example. Fig. 8**(a)** shows the unsmoothed and the smoothed transmission rate of the DVD movie "Twister" with B = 32 MB, m = 17 thresholds and w_{pred} = 180 seconds. Similarly, Figs. 8(b) and (c) show the experimental results for the MPEG-4 and HDTV contents. The variability is clearly reduced as well as the peak rate. To quantify the effectiveness of our technique, we measured the standard deviation of the transmission schedule. Table VI summarizes the experimental results that we collected from the LAN experimental setup. Note that no actual overflow or underflow occurred in these experiments. We quantified the feedback message overhead assuming that the length of each command is approximately 100 bytes. We also measured the impact of the rate adaptation on the server CPU load and the results show that the overhead is negligible (significantly less than 0.1%).

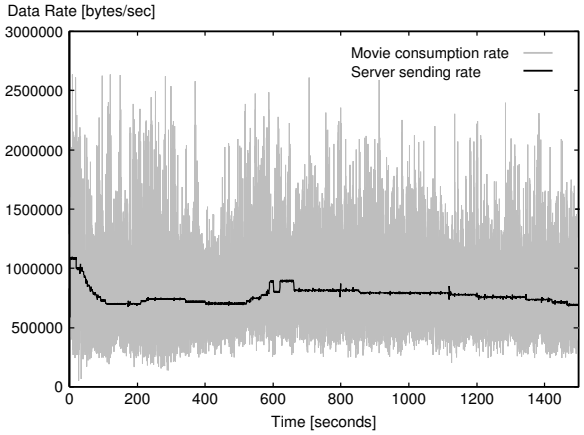


Fig. 8(a): A 25 minute segment of a typical DVD movie ("Twister").

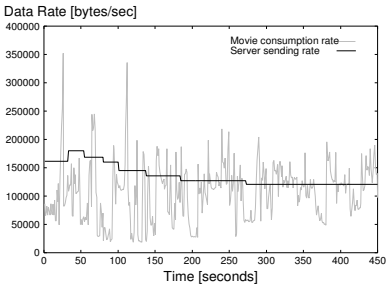


Fig. 8(b): MPEG-4 movie (640 × 480).

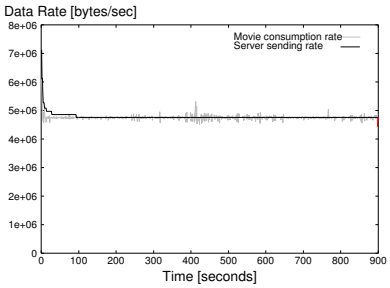


Fig. 8(c): HDTV MPEG-2 movie (1920 × 1080).

Figure 8. Real consumption rate versus smoothed sending rate for different movie type. The smoothing parameters used are as follows: 32 MB playout buffer size, 17 thresholds and 180 seconds prediction window size.

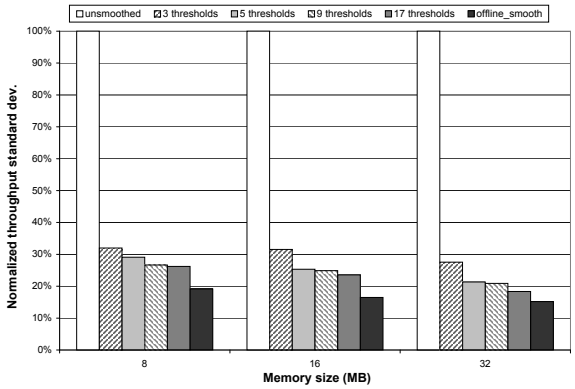


Figure 9. Reduction in rate variability of the movie "Twister" with different client buffer sizes and number of thresholds. The transmission schedule becomes smoother as the number of thresholds increases and with an increased buffer size.

Table VI. Experimental results for multiple client buffer sizes, different number of thresholds, and different prediction window sizes (DVD movie streams at 6 Mb/s).

Parameters			Measurements				
B	m	w_{pred} [samples][12]	x[7]	y[8]	z[9]	u[10]	v[11]
8 MB	3	45	103428	50	2	0	4.66E-06
	3	90	98633	43	1	0	4.00E-06
	3	180	96566	43	1	0	4.00E-06
	5	45	89599	69	3	0	6.43E-06
	5	90	88286	57	3	0	5.31E-06
	5	180	88784	43	4	0	4.00E-06
	9	45	84057	107	5	0	9.97E-06
	9	90	82210	74	5	0	6.89E-06
	9	180	85583	57	5	0	5.31E-06
	17	45	79411	119	6	0	1.11E-05
	17	90	80893	97	7	0	9.03E-06
	17	180	81139	71	9	0	6.61E-06
16 MB	3	45	98666	18	1	0	1.68E-06
	3	90	95800	23	2	0	2.14E-06
	3	180	97425	20	2	0	1.86E-06
	5	45	79459	26	1	0	2.42E-06
	5	90	76344	22	1	0	2.05E-06
	5	180	80309	27	1	0	2.51E-06
	9	45	73864	45	1	0	4.19E-06
	9	90	75791	47	1	0	4.38E-06
	9	180	75522	31	1	0	2.89E-06
	17	45	71066	93	1	0	8.66E-06
	17	90	72237	49	4	0	4.56E-06
	17	180	72060	39	3	0	3.63E-06
32 MB	3	45	88102	10	0	0	9.31E-07
	3	90	85399	9	0	0	8.38E-07
	3	180	79669	11	0	0	1.02E-06
	5	45	84403	18	0	0	1.68E-06
	5	90	66974	17	0	0	1.58E-06
	5	180	59657	9	0	0	8.38E-07
	9	45	70691	23	0	0	2.14E-06
	9	90	64037	21	0	0	1.96E-06
	9	180	51264	18	0	0	1.68E-06
	17	45	71575	42	0	0	3.91E-06
	17	90	56594	38	0	0	3.54E-06
	17	180	51192	27	0	0	2.51E-06

[7] Throughput standard deviation
[8] Number of rate adjustments
[9] Number of times protection threshold (TH_O and TH_U) crossed
[10] Number of buffer overflow or underflow
[11] Overhead of rate adjustments(ratio of message size over total transmitted data size)
[12] Since the sampling interval is 1 second, the prediction window lengths are 45, 90 and 180 seconds, respectively.

4.1. Transmission Schedule Smoothness

We examined the effectiveness of MTFC by comparing the smoothed transmission schedules to the unsmoothed schedule and the optimal offline schedule [29] for the movie "Twister." The prediction window size in these experiments is 90 seconds.

Fig. 9 presents the reduction in standard deviation achieved by MTFC, across client buffer sizes ranging from 8 MB to 32 MB and with the number of thresholds ranging from 3 to 17. In all cases, the standard deviation is reduced substantially, by 67-74% for a 8 MB client buffer, 67-76% for a 16 MB client buffer, and 72-81% for a 32 MB client buffer. This figure illustrates what is intuitively clear: an increase in the client buffer size yields smoother traffic. More importantly, it also shows that a higher number of thresholds results in smoother traffic. Note that the optimal offline schedule is only 3-8% smoother than the result of MTFC and the difference decreases with larger client buffers.

The optimal buffer size is a direct function of the bandwidth required by a stream and signaling latency between the client and the server. Our DVD test streams required 6 Mb/s (on average) and hence using less than 8 MBs of buffer resulted in less than optimal performance. However, for example with a wireless handset the stream bandwidth will most likely be much lower. Assuming a 320×240 screen, an MPEG-4 movie would require about 200-250 kb/s (note that the MPEG-4 movie used in Fig. 8(b) had a video resolution of 640×480 and required about 800 kb/s). Therefore, a buffer size of roughly 350 kB should suffice and yield similar results. In such an application one would probably choose the simplest prediction algorithm to reduce the processing power requirements.

In the next set of experiments we compared the impact of different prediction window sizes. We expected that a longer prediction window would result in a smoother transmission schedule. However, the longer the prediction window extends into the future, the less accurate the prediction becomes. Figs. 10(a), (b) and (c) show the rate variability reduction with a prediction window size of 45, 90 and 180 seconds and buffer sizes of 8 MB, 16 MB and 32 MB, respectively. With a 32 MB client buffer (Fig. 10(c)), for a given number of thresholds, a longer prediction window results in smoother traffic. However, with a smaller client buffer size (8 MB and 16 MB) the trend is less clear. A longer prediction window corresponds to a longer prediction time, which results in longer segments of a constant rate and hence smoother streams. Although this holds true for larger memory sizes, smaller buffer sizes with the same number of thresholds contain less data between two neighboring thresholds, and hence server rate changes are triggered more quickly. Thus, if the prediction window is too long, it does not necessarily extend the interval during which the server sends data at a constant rate. In our experiments, a window size of 45 seconds is already sufficient for buffer sizes of 8 MB and 16 MB. Therefore, increasing the prediction window to 90 or 180 seconds did not reduce traffic variability any further.

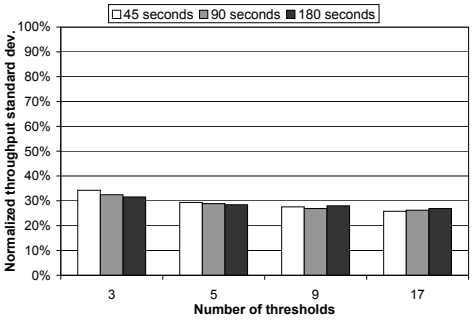


Fig. 10(a): B = 8 MB.

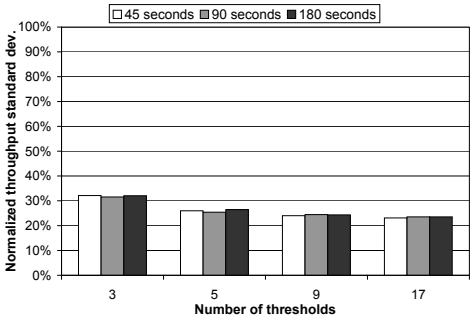


Fig. 10(b): B = 16 MB.

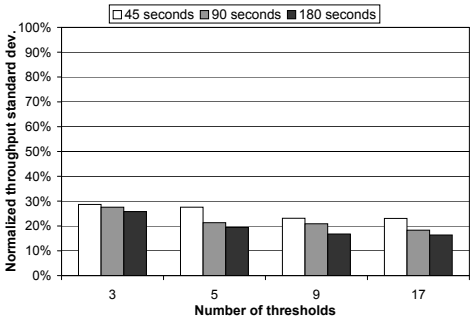


Fig. 10(c): B = 32 MB.

Figure 10. Impact of the prediction window size on the reduction in rate variability.

4.2. Rate Change Signaling Frequency

Feedback messages from the client to the server introduce overhead that should be minimized. We examined the overhead generated by our MTFC scheme in terms of the number of rate changes. The prediction window was fixed at 90 seconds in these experiments.

Fig. 11 shows the number of rate changes across client buffer sizes ranging from 8 MB to 32 MB and the number of thresholds varying from 3 to 17.

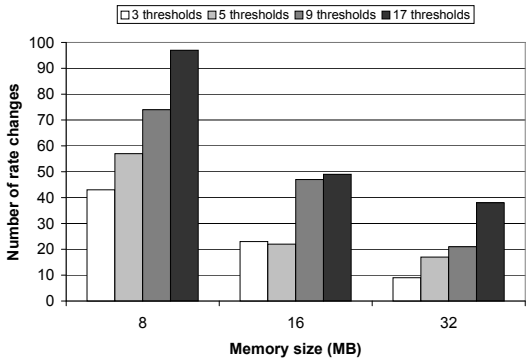


Figure 11. Number of rate changes generated by MTFC as a function of the client buffer sizes and the number of thresholds.

The figure clearly shows that an increase in the number of thresholds results in a steady increase in the number of rate changes. It also demonstrates that a larger client buffer reduces the number of rate changes. We also quantified the overhead of the feedback messages as a fraction of the total transmission bandwidth (see Table VI) and we found that it is negligible. As illustrated in Fig.12 a longer prediction window generally results in fewer rate changes.

4.3. Evaluation of Different Threshold Spacing Strategies

Recall that one of the buffer management parameters is the threshold spacing strategy. The simplest method will place thresholds at equal distances within the buffer. However, one might conjecture that feedback actions should be executed at a finer granularity when the amount of data is close to either a full or an empty buffer. To test this hypothesis we evaluated our MTFC with the three different threshold placement strategies: (1) equi-distant, (2) based on an arithmetic series (*arithmetic spacing*), and (3) based on a geometric series (*geometric spacing*).

We implemented these two non-uniform spacing strategies in our client and conducted the same experiments as we did with equi-distant spacing. We again used the rate variability and the number of rate changes as the performance measure. Note that in all these experiments, the prediction window length was 45 seconds.

The smoothing results are shown in Fig.13(a) and interestingly, in most experiments all three spacing strategies achieve very similar smoothing effects. One notable exception occurs for a small playout buffer size of 8MB or 16MB, when the number of thresholds m is high, e.g., 17. In this case, geometric spacing performs considerably worse than the other two strategies. We speculate that the density of thresholds is too high and hence rate corrections

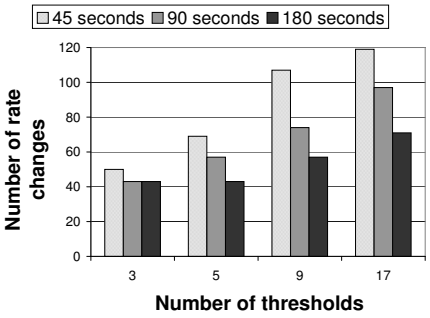


Fig. 12(a): B = 8 MB.

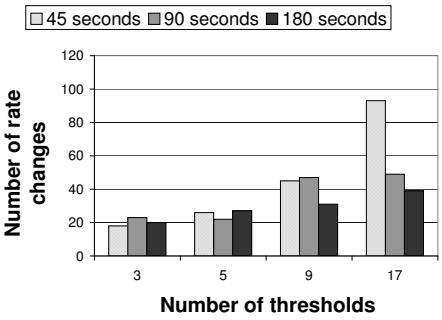


Fig. 12(b): B = 16 MB.

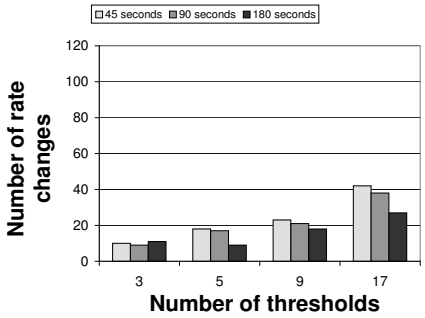


Fig. 12(c): B = 32 MB.

Figure 12. Impact of the prediction window size on the number of rate changes. As expected, a larger window size reduces the number of rate changes.

become ineffective. Also, for $B = 32MB$ and $m = 3$, both the arithmetic and geometric spacing strategies performed not as good as expected. With this big buffer size and small number of thresholds, non-uniform spacing strategies result in a low threshold density near the buffer midpoint level TH_N. Hence, the thresholds located far from TH_N may be reached infrequently, resulting in less rate adjustments. Note that the equi-distant spacing strategy performs consistently well with all parameters. Intuitively, the uniformly distributed

thresholds allow rate adjustments enough time to have the desired smoothing effect.

Fig. 13(b) shows the number of rate changes for all three spacing strategies with different client buffer sizes and different numbers of thresholds. With a small number of thresholds, say 3 or 5, all three spacing strategies result in a comparable number of rate changes since the positions of the thresholds are either exactly the same or quite similar. For larger numbers of thresholds and the client buffer is small, for example 8 MB or 16 MB, the non-uniform spacing strategies result in a higher number of rate changes because some thresholds are too close to each other. On the other hand, for large client buffer sizes, the non-uniform spacing strategies generally result in a lower number of rate changes. This is due to the lower threshold density near the buffer level midpoint TH_N, where the data level fluctuates most of the time.

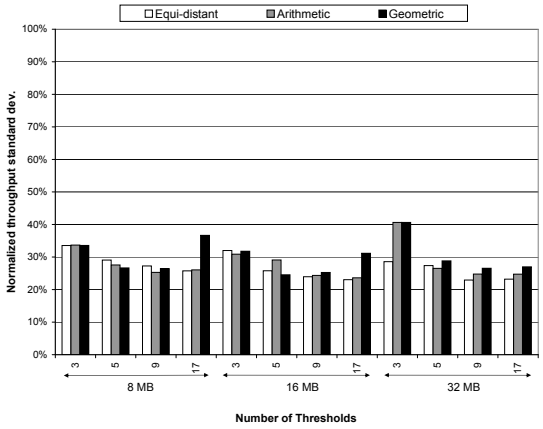


Fig. 13(a): Throughput standard deviation.

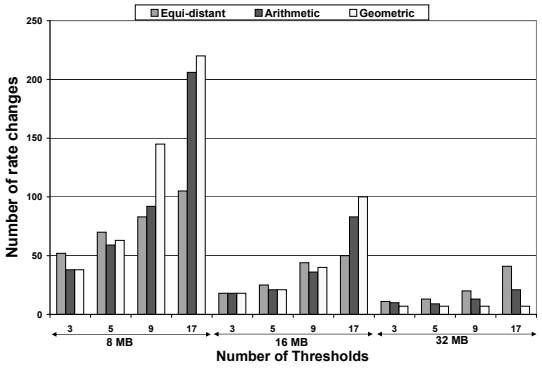


Fig. 13(b): **Number of rate changes.**

Figure 13. Comparison of three threshold spacing strategies.

4.4. Comparison with Previously Proposed Techniques

We compared our MTFC technique with two previously proposed online techniques by Hui et al. [14] and Mielke and Zhang [20]. We implemented both algorithms in our distributed continuous media architecture. We will call the two techniques FIXED3 and FIXED2, respectively, for the the number of thresholds that they use. Our performance measures were the rate variability, the number of rate changes and the number of times the protection thresholds were passed. All experiments used the movie "Twister" and the MTFC prediction window was fixed at 90 seconds.

4.4.1. *Comparison with the FIXED3 Technique*

The FIXED3 [14] technique utilizes three thresholds: low, medium and high. Whenever a threshold is crossed, the server sending rate is set to a new fixed rate, which is pre-calculated based on perfect knowledge of the average consumption rate of the movie. If the high threshold is crossed, the server sending rate is reduced to 0.5 times of the average rate. If the medium threshold is crossed, the rate is set to the average rate. If the low threshold is passed, the server sending rate is set to 1.5 times the average rate. To allow for a fair comparison, we set the three thresholds to the same levels as our MTFC technique with three thresholds. Furthermore, we added the two overflow and underflow protection thresholds TH_O and TH_U to FIXED3.

Fig. 14(a) shows the standard deviation of the server sending rate of MTFC and FIXED3 with client buffer sizes ranging from 8 MB to 32 MB. The standard deviations are normalized to the variability of the unsmoothed movie. The figure illustrates that our MTFC techniques outperform FIXED3 by between 10% to 15%. Fig. 14(c) illustrates the number of rate changes generated by both techniques as a function of different buffer sizes. Overall, MTFC results in fewer rate changes than FIXED3. Fig. 14(e) shows the number of times the protection thresholds are crossed. For this measure, FIXED3 performs better because it changes the sending rate much more aggressively at both the high and low thresholds. Since MTFC changes the sending rate based on a future consumption rate prediction, the protection thresholds are rarely crossed. Note that with both techniques the buffer never actually overflows or underflows.

4.4.2. *Comparison with the FIXED2 Technique*

The technique proposed by Mielke and Zhang uses only two thresholds and the new server sending rate is calculated solely based on the buffer threshold levels and the client buffer conditions. Hence, the major differences between our MTFC and the FIXED2 technique are the lack of a prediction algorithm and the non-existent multi-threshold buffer model of FIXED2. For a fair comparison, we set the FIXED2 thresholds at the same level as with our MTFC technique with two thresholds and we also added the two overflow and

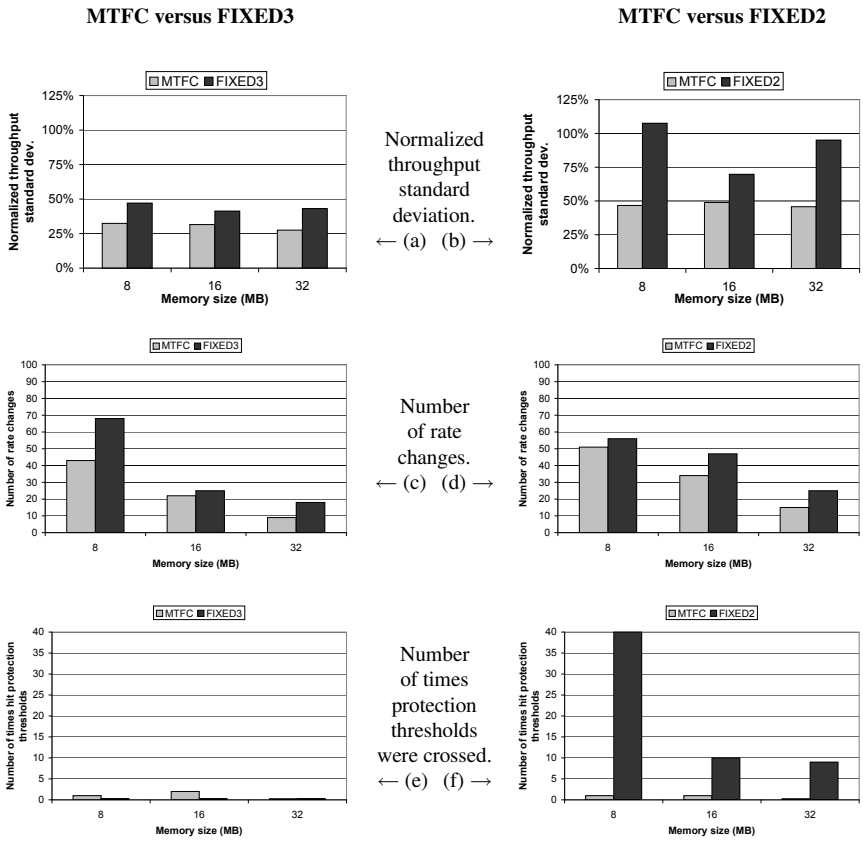


Figure 14. MTFC performance comparison of the FIXED3 versus the FIXED2 techniques.

underflow protection thresholds, TH_O and TH_U. Figs. 14(b), (d) and (f) show the normalized server sending rate variability, the number of rate changes, and the number of times both algorithms crossed the protection thresholds as a function of three client buffer sizes. As shown in Fig. 14(b), MTFC results in a 20%-60% smoother stream as compared with FIXED2. Figs. 14(d) and (f) illustrate that FIXED2 has a higher feedback overhead and crosses the protection thresholds 5-40 times more than MTFC. With both techniques, no actual buffer overflow or underflow occurred. These figures clearly indicate that the MTFC scheme outperforms the FIXED2 technique. In these experiments, since MTFC also uses two thresholds, the better performance should be attributed to the effectiveness of the prediction algorithm. Note that without consumption prediction, the variability of the server sending rate generated by FIXED2 sometimes (e.g., when $B = 8$ MB) is even higher than the unsmoothed schedule.

Figs. 14(a), (c), (e) and Figs. 14(b), (d), (f) use the same scales and can be compared directly. This demonstrates that increasing the number of thresholds from two to three lowers the rate variability by approximately 30%-40% and results in less feedback overhead. With the multi-threshold MTFC technique we have increased the number of thresholds beyond three and we showed that the sending rate variability is further reduced.

5. Conclusion and Future Research Directions

We have presented a novel online flow control technique for stream smoothing that includes a consumption rate prediction component and a multi-threshold buffer model. From our extensive experiments conducted in a real-world environment we conclude that more than three buffer thresholds reduces the variability of the data transmission and the feedback overhead. At the same time, a consumption rate prediction algorithm allows the smoothing of streams with no prior knowledge of their transmission schedule. Therefore, our technique is very well suited for highly dynamic environments that need to adapt to changing network and load conditions. Furthermore, the achieved smoothing allows for improved resource utilization by significantly reducing peak data rates.

As part of our future work we plan to extend our MTFC technique in several directions. First, we intend to combine network congestion control mechanisms with our smoothing technique to achieve even more robust media stream transmissions. Second, we plan to evaluate the MTFC technique in other types of network environments, for example mobile and wireless networks. Finally, we are working towards incorporating MTFC into our new High-speed Data Recording Architecture (HYDRA) [38] and evaluate its performance with a combination of real time recording and playback streams.

References

1. Aboobaker, N., D. Chanady, M. Gerla, and M. Y. Sanadidi: 2002, 'Streaming Media Congestion Control using Bandwidth Estimation'. In: *Proc. of IFIP/IEEE International Conference on Management of Multimedia Networks and Services*. Stockholm, Sweden.
2. Al-Marri, J. and S. Ghandeharizadeh: 1998, 'An Evaluation of Alternative Disk Scheduling Techniques in Support of Variable Bit Rate Continuous Media'. In: *Proceedings of the International Conference on Extending Database Technology*.
3. Allman, M. and V. Paxson: 1999, 'On Estimating End-to-End Network Path Properties'. In: *SIGCOMM*. pp. 263–274.
4. Amir, E., S. McCanne, and R. Katz: 1997, 'Receiver-driven Bandwidth Adaptation for Light-weight Session'. In: *Proceedings of the 5th ACM International Multimedia Conference*. Seattle, WA, pp. 415–426.
5. Chang, R., M. Chen, J. Ho, and M. Ko: 1999, 'An Effective and Effcient Traffic Smoothing Scheme for Delivery of Online VBR Media Streams'. In: *Proceedings of the IEEE INFOCOMM*.

6. Chatfield, C.: 2001, *Time-Series Forecasting*. CHAPMAN&HALL/CRC Boca Raton London New York Washington, D.C.
7. Chen, H. A., L. Qiao, and K. Nahrstedt: 2001, 'Adaptive versus Reservation-based Synchronization Protocols - Analysis and Comparison'. *Multimedia Tools and Applications Journal, num. 4, Kluwer Publisher* pp. 219–257.
8. Cheung, G., W. Tan, and T. Yoshimura: 2003, 'Double Feedback Streaming Agent for Real-time Delivery of Media over 3G Wireless Networks'. In: *IEEE Wireless Communications and Networking Conference (WCNC2003)*. New Orleans, Louisiana, USA.
9. Dogan, S., A. Cellatoglu, M. Uyguroglu, A. H. Sadka, and A. M. Kondoz: 2002, 'Error-Resilient Video Transcoding for Robust Internetwork Communications Using GPRS'. *IEEE Transactions on Circuits and Systems for Video Technology, Vol.12, No.6* pp. 453–464.
10. Feng, A. C., A. C. Kapadia, W. chun Feng, and G. G. Belford: 2002, 'Packet Spacing: An Enabling Mechanism for Delivering Multimedia Content in Computational Grids'. *The Journal of Supercomputing, 23(1)* pp. 51–66.
11. Floyd, S., M. Handley, J. Padhye, and J. Widmer: 2000, 'Equation-based congestion control for unicast applications'. In: *SIGCOMM 2000*. Stockholm, Sweden, pp. 43–56.
12. Gevros, P., J. Crowcroft, P. Kirstein, and S. Bhatti: 2001, 'Congestion control mechanisms and the best effort service model'. *IEEE Networking, Volume: 15, Issue: 3* pp. 16–26.
13. Hac, A. H. and C. X. Xue: 1997, 'Synchronization in multimedia data retrieval'. *International Journal of Network Management, Volume: 7, Issue: 1* pp. 33–62.
14. Hui, J., E. Karasan, J. Li, and J. Zhang: 1996, 'Client-Server Synchronization and Buffering for Variable Rate Multimedia Retrievals'. *IEEE Journal on Selected Areas in Communications, 14(1)* pp. 226–237.
15. Hui, J., J. Zhang, and J. Li: 1995, 'Quality of Service Control in GRAMS for ATM Local Area Networks'. In: *Proceedings of the IEEE JSAC*.
16. Jacobson, V. and M. J. Karels: 1988, 'Congestion Avoidance and Control'. In: *Proceedings of ACM SIGCOMM '88*. pp. 314–329.
17. Karn, P. and C. Partridge: 1991, 'Improving Round-Trip Time Estimates in Reliable Transport Protocols'. *ACM Transactions on Computer Systems* **9**(4), 364–373.
18. Khedkar, P. and S. Keshav: 1992, 'Fuzzy Prediction of Time Series'. In: *Proceedings of IEEE Conference on Fuzzy Systems*.
19. Loguinov, D. and H. Radha: 2001, 'On Retransmission Schemes for Real-time Streaming in the Internet'. In: *INFOCOM*. pp. 1310–1319.
20. Mielke, M. and A. Zhang: 1998, 'A Multi-Level Buffering and Feedback Scheme for Distributed Multimedia Presentation Systems'. In: *Proceedings of Seventh International Conference on Computer Communications and Networks (IC3N'98)*. Lafayette, Louisiana.
21. Morikawa, D., S. Ota, A. Yamaguchi, and M. Ohashi: 2002, 'A Feedback Rate Control of Video Stream in Best-Effort High-Speed Mobile Packet Network'. In: *The 5th International Symposium on Wireless Personal Multimedia Communications (WPMC2003)*. Sheraton Waikiki, Honolulu, Hawaii, USA.
22. Nahrstedt, K.: 1995, 'End-to-End QOS Guarantees in Networked Multimedia Systems'. *ACM Computing Survey, 27(4)* pp. 613–616.
23. Nick Feamster, H. B.: 2002, 'Packet Loss Recovery for Streaming Video'. In: *Proceedings of 12th International Packet Video Workshop*.
24. Papadopoulos, C., G. M. Parulkar, and G. Varghese: 1998, 'An Error Control Scheme for Large-Scale Multicast Applications'. In: *Symposium on Principles of Distributed Computing*. p. 310.

25. Pejhan, S., M. Schwartz, and D. Anastassiou: 1996, 'Error Control using Retransmission Schemes in Multicast Transport Protocols for Real-time Media'. *IEEE/ACM Transactions on Networking* **4**(3), 413–427.
26. Ramanathan, S. and P. V. Rangan: 1993a, 'Feedback Techniques for Intra-Media Synchronization in Distributed Multimedia Systems'. *The Computer Journal, 36(1)* pp. 19–31.
27. Ramanathan, S. and P. V. Rangan: 1993b, 'Feedback Techniques for Synchronizated Multimedia Retrieval over Integrated Networks'. *IEEE/ACM Transactions on Networking, 1(2)* pp. 246–260.
28. Rhee, I.: 1998, 'Error Control Techniques for Interactive Low-Bit Rate Video Transmission over the Internet'. In: *SIGCOMM*. pp. 290–301.
29. Salehi, J., Z.-L. Zhang, J. Kurose, and D. Towsley: 1998, 'Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing'. *IEEE/ACM Transactions on Networking* pp. 397–410.
30. Shahabi, C., G. Barish, B. Ellenberger, N. Jiang, M. R. Koladouzan, S.-R. A. Nam, and R. Zimmermann: 1999, 'Immersidata Management: Challenges in Management of Data Generated within an Immersive Environment'. In: *Proceedings of the Fifth International Workshop on Multimedia Information Systems (MIS'99)*. Indian Wells, California.
31. Shahabi, C., R. Zimmermann, K. Fu, and S.-Y. D. Yao: 2002, 'Yima: A Second Generation of Continuous Media Servers'. *IEEE Computer Magazine* pp. 56–64.
32. Tan, W., W. Cui, and J. G. Apostolopoulos: 2003, 'Playback-Buffer Equalization for Streaming Media using Stateless Transport Prioritization'. In: *The 13th International Packet Video Workshop (PV2003)*. Nantes, France.
33. Ueda, K., H. Ohsaki, S. Shimojo, and H. Miyahara: 2003, 'Design and Implementation of Real-Time Digital Video Streaming System over IPv6 Network using Feedback Control'. In: *Symposium on Applications and the Internet (SAINT2003)*. Orlando, Florida, USA, pp. 111–119.
34. Wu, D., Y. T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H. Chao: 2000, 'On End-to-End Architecture for Transporting MPEG-4 Viceo Over the Internet'. *IEEE Transactions on Circuits and Systems for Video Technology, Vol.10, No.6* pp. 923–941.
35. Zhang, A., Y. Song, and M. Mielke: 2002, 'NetMedia: streaming multimedia presentations in distributed environments'. *IEEE Multimedia, Volume: 9, Issue: 1* pp. 56–73.
36. Zhang, Q., W. Zhu, and Q.-Q. Zhang: 2001, 'Resource Allocation for Multimedia Streaming Over the Internet'. *IEEE Transcations on Multimedia, Vol.3, No.3* pp. 339–355.
37. Zhang, Z.-L., J. Kurose, J. Salehi, and D. Towsley: 1997, 'Smoothing, statistical multiplexing, and call admission control for stored video'. *IEEE Journal on Selected Areas in Communications, Volume: 16 Issue: 6* pp. 1148–1166.
38. Zimmermann, R., K. Fu, and W.-S. Ku: 2003a, 'Design of a Large Scale Data Stream Recorder'. In: *Proceedings of the Fifth International Conference on Enterprise Information Systems (ICEIS 2003)*. Angers, France.
39. Zimmermann, R., K. Fu, N. Nahata, , and C. Shahabi: 2003b, 'Retransmission-Based Error Control in a Many-to-Many Client-Server Environment'. In: *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking 2003 (MMCN 2003)*. Santa Clara, CA.
40. Zimmermann, R., K. Fu, C. Shahabi, S.-Y. D. Yao, and H. Zhu: 2001, 'Yima: Design and Evaluation of a Streaming Media System for Residential Broadband Services'. In: *VLDB 2001 Workshop on Databases in Telecommunications (DBTel 2001)*. Rome, Italy.