MDRS: A LOW COMPLEXITY SCHEDULER WITH DETERMINISTIC PERFORMANCE GUARANTEE FOR VBR VIDEO DELIVERY

BY

LAI HIN LUN

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF INFORMATION ENGINEERING

THE CHINESE UNIVERSITY OF HONG KONG

JUNE 2001

Abstract

Variable-bit-rate (VBR) video encoding provides a more consistent visual quality when compared to constant-bit-rate (CBR) video encoding. However, the long-range bit-rate variations require the network and video server to adapt to the large fluctuations in bandwidth requirement in a video-on-demand (VoD) system. We investigate a new scheduling algorithm with monotonic-decreasing rate allocations for scheduling disk retrievals and data transmissions. This scheduler enables the use of a minimal complexity admission-control algorithm that can provide deterministic performance guarantee. We also study and compare the performance of several other schedulers. Simulation results based on a large number of VBR video traces (307 DVD movies) show that despite long-range bit-rate variations within individual videos,

the proposed scheduler can achieve a disk efficiency of over 93% when compared to the maximum achievable disk utilization of a system serving multiple concurrent video streams. With a trade off in buffer requirement which can be easily accommodated by today's set-top-boxes and PCs, our proposed scheduler achieves much lower admission complexity than existing methods, yet providing very good disk utilization.

摘要

可變比特率視頻編碼提供的視覺效果比固定比特率視頻編碼所能提供的更爲穩定。但是其中大範圍的比特率變化須要視頻點播系統中的視頻伺服器及網路來配合這種巨大的帶寬變動。我們提出了一個新的排程器，以純遞減形式的分配來編排硬盤讀取和資料傳輸。這個排程器能以最低複雜度的接入控制演算法來提供確定的性能保證。我們更將其性能和其他幾個排程器作研究和比較。基於大量可變比特率視頻編碼的視頻記錄（307 套 DVD 電影）之模擬測試結果顯示，儘管在每個視頻中均有大範圍的比特率變化，我們提議的排程器相對一個同時有多個視頻流的系統所能達到最高的硬盤利用率而言，仍可達到超過百份之九十三的效率。雖然我們提議的排程器須用上頗大的緩衝區，但現今的機頂盒或者個人電腦已能輕易地符合這個要求。而且它比現存方法的接入複雜度要低得多，更能達到相當好的硬盤利用率。

Acknowledgement

I would like to express my sincerest gratitude to my supervisor, Professor L. K. Chen, for his continued encouragement, guidance and support to my study.

I would also like to thank my co-supervisor, Prof. Jack Lee, for his insights and invaluable advices for my research work.

Thanks also go to all the fellow researchers in the department of Information Engineering, whose opinions and informative discussions helped develop my research work and this thesis into their final forms.

Table of Contents

List of Figures

Chapter 1

Introduction

Future broadband networks will support a wide variety of services with very different traffic characteristics. Multimedia applications such as video-on-demand are expected to consume a significant portion of the bandwidth. The efficient transmission of delay-sensitive VBR video data [1] is likely to be one of the key challenges in the resource management of such networks. Apart from the frame-by-frame bit-rate fluctuations that are also found in CBR videos, VBR videos tend to exhibit long-range bit-rate variations in time scale of minutes. Example VBR and CBR video bit-rate profiles are shown in Figure 1 & Figure 2 respectively.

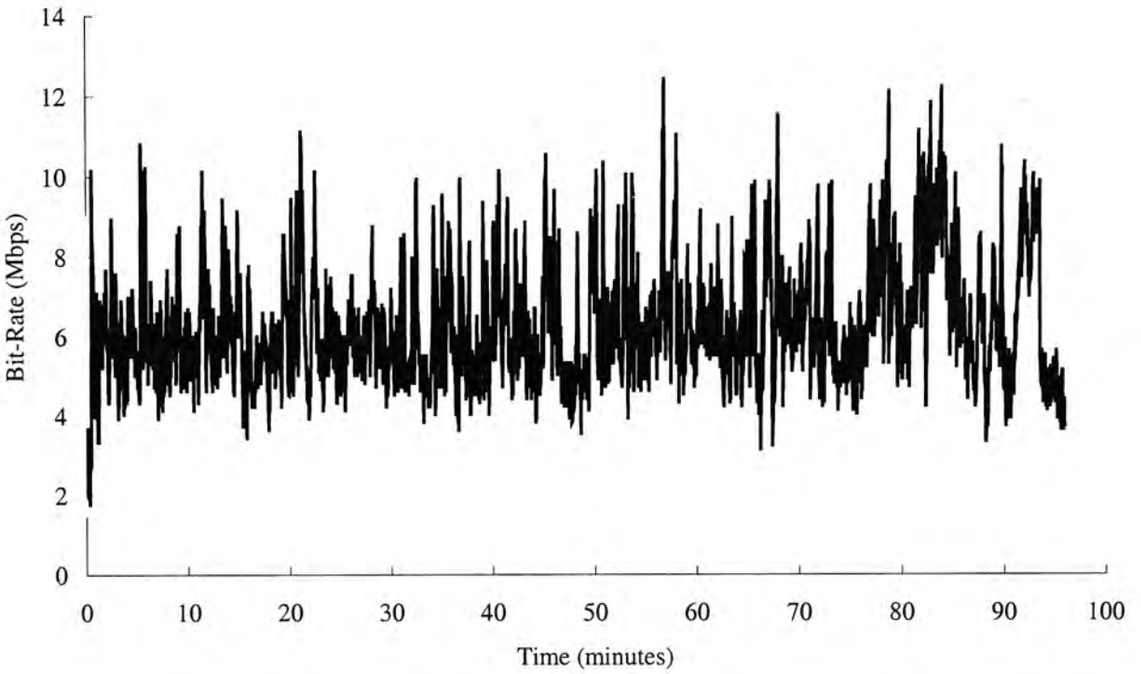


Figure 1 Sample VBR video trace

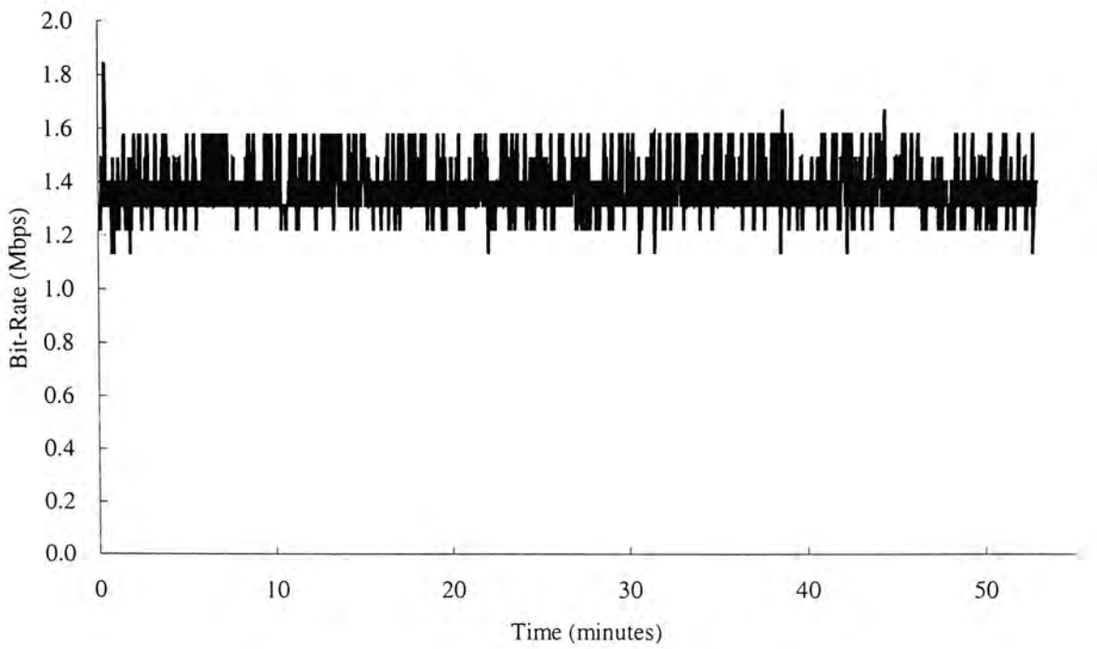


Figure 2 Sample CBR video trace

To alleviate this problem, researchers have conducted extensive studies trying to reduce the burstiness of VBR videos [2]-[6]. These include smoothing techniques applied to the encoder [7], online video compression with network utilization feedback to the encoder to control the encoding parameters [8]-[9], and the smart scheduling of video data transmission to minimize the magnitude or the number of bandwidth changes [10].

Most of these studies put the main focus on the network aspect. In this paper, we investigate a much more complete solution that addresses the admission control, as well as the scheduling of data transmission/retrieval in both the network and the disk storage. The simulation results given in Chapter 4 are mainly based on disk retrieval. In scheduling network transport, bit-rates can be of arbitrary numbers, and the aggregation of multiple data streams has little or no switching overhead. On the contrary, in disk retrieval, bit-rates are limited to the multiples of the base unit of one sector per round, and the switching between concurrent data streams induces significant overhead.

Consider the popular disk scheduling algorithm, SCAN [11]-[12], which schedules

the read requests in rounds and read one block for each stream while the disk head scans back and forth across the disk surface. This algorithm is designed for achieving higher throughput. Given the long-range bit-rate variations of VBR-encoded videos, it is still very difficult to perform admission control to provide deterministic performance guarantee. One may need to perform a heavy amount of disk I/O time estimations before granting a user request; or may have to rely on the aggregated sum of traffic characteristics of the video streams to provide statistical performance guarantee [13].

For example, consider a server using SCAN with fixed round length and variable block size for scheduling disk retrievals. Assume the server is serving S existing streams and a request for a new stream of length equivalent to R disk rounds arrives. To provide deterministic performance guarantee, the server can admit this stream only if none of the R rounds will be overloaded by the new stream. Consequently, the server will have to perform resource allocation by estimating the disk I/O time required for the R disk rounds, and then add to the estimations for the other S streams round by round to ensure there is no overload. Given that a typical video server can serve over 100 streams and a movie can span over 7200 rounds (for a 2-hour movie

served with 1-second round time), the amount of computations involved is substantial. The server must be able to precisely estimate the I/O time in order to guarantee performance. Furthermore, I/O time is a complex function of disk seeking, head switching, and latency, given by:

$$T_{read}(n) = \alpha + \beta\sqrt{n} + T_{lantency} + \frac{Q}{R_{disk}} \quad (1)$$

where: α – fixed overhead (sec)

β – seek time constant (sec)

n – number of tracks to seek

$T_{lantency}$ – rotational latency (sec)

Q – size of data to read (byte)

R_{disk} – disk transfer rate (bytes/sec)

This will only further increase the computational complexity.

Our simple simulation shows that it takes about 10 ms to perform 7200 disk I/O time estimations on a P-III 500 PC. However, this optimistic result does not mean a system can admit a maximum of 100 requests in one second. It is because the result is

obtained from a simple program loop on which the CPU can work at full speed with all code and data in its primary cache and no context switching or other overhead. Secondly, if we assume a realistic server can afford about 5% of CPU time for admission control, then the server can only process less than 5 admissions per second. This will clearly become the bottleneck in large-scale systems or systems with bursty arrivals.

Even if one can efficiently perform the admission process, the scheduling of video data transmission is not trivial. Long-range bit-rate variations not just complicates admission control, it also means a number of bandwidth renegotiations within the playback duration which, when the effects of a large number of streams combine, can overwhelm the network system. Network bandwidth renegotiations are especially undesirable when there are other traffics in the network, in which upwards renegotiations may not be successful.

Seeing that disk storage is becoming cheap and abundant, this study tackles all the previous problems with a new scheduling algorithm by trading off client side buffer requirement with monotonic-decreasing rate allocations for scheduling disk retrievals

and data transmissions. This scheduler enables the use of a minimal complexity admission-control algorithm that can provide

- *deterministic performance guarantee,*
- *zero startup delay,*
- *no loss of video quality*
- *good disk efficiency, and*
- *the capability to co-exist with ABR data service*

In the next chapter, we introduce several other solutions, and discuss about their advantages and limitations. Then we present our new scheduling algorithm in Chapter 3 and show the simulation setup and performance results in Chapter 4. Finally, Chapter 5 concludes the paper.

Chapter 2

Related Works

The study of VBR video delivery basically falls in to two big categories, source-model based and trace-based. In both cases, different methods provide either statistical guarantee or deterministic guarantee. Our proposed scheduler is a trace-based method providing deterministic guarantee. Here, we briefly introduce methods that try to tackle the problems from different perspectives.

2.1 Source Modeling

Compressed videos posses self-similar characteristics [14], and many video traffic source models have been proposed, which are useful in network design and performance analysis. Such models are helpful in many areas such as predicting packet loss ratio and locating the bottleneck in a network. Most models try to exploit the frame-by-frame correlation or the effects of scene changes and average scene lengths. The variance of bits per frame is modeled, and some use autoregressive models to predict future bandwidth [15]-[16].

The advantage of using source models in traffic engineering is that a small set of parameters can represent a range of video traffics. Adjustments on such parameters can adapt the generated traffic with particular network designs. Source models for videos with very different traffic characteristics are also available, which include models for videos with lower activity (such as teleconferences) [17], models for active video sequences [18], models for videos that are encoded with a particular structure or pattern (such as MPEG encoding with the well-known IPB frames encoding and fixed GOPs) [19], and models for multiplexed video streams [20]. The

main drawback of using source models is that no model can perfectly represent actual video traffic, so a system maybe overloaded or under-utilized when the traffic is under- or over-estimated. There are also worst-case models which give guaranteed bounds [21] to eliminate traffic underestimation, and thus system overloading. However, poor utilization would be expected when such models are incorporated.

2.2 CBR Scheduler for VBR Delivery

A simple way to make use of the bit-rate trace of VBR video traffic is to adapt it for CBR transport. Admission, retrieval and transmission can be based on a constant bit-rate like the peak, average or critical rate (critical rate is the minimum constant rate to schedule a video delivery with neither startup delay nor data underflow, see Figure 4).

Peak Rate Scheduler:

The most apparent way performing admission control is to base on the peak rate of the requested video and reserve that bandwidth throughout the playback of the entire video, although the actual requirement is lower most of the time. In this case, admission control is very simple: a new stream is admitted only if the server has extra bandwidth larger than the peak rate of the requested video. Video data are then retrieved and transmitted according to the exact bit-rate profile. This scheduler enables videos to be delivered with zero startup delay and virtually zero client side buffer space (depending on the encoding algorithm and implementation of the

decoder, a number of frames may need to be buffered for inter-frame decoding). The main disadvantages are the reduced bandwidth utilization and the large amount bit-rate renegotiations required in the transmission.

Average Rate Scheduler:

To improve the bandwidth utilization over the Peak Rate Scheduler, we can perform admission control and schedule a video stream to be served at its average rate. However, a disadvantage of this scheduler is that pre-play buffering may be required for some videos. As shown in Figure 3, there maybe points at which the cumulative data consumption function $A(t)$, which represents the amount of data that need to be accumulated at the client side at time t, is higher than the average rate curve where data underflow will occur. In that case, we would have to right shift the cumulative data consumption function (delaying the start of playback) until it is completely under the average rate schedule to ensure that no data underflow will occur. The amount of shifting is then equal to the pre-play buffering delay.

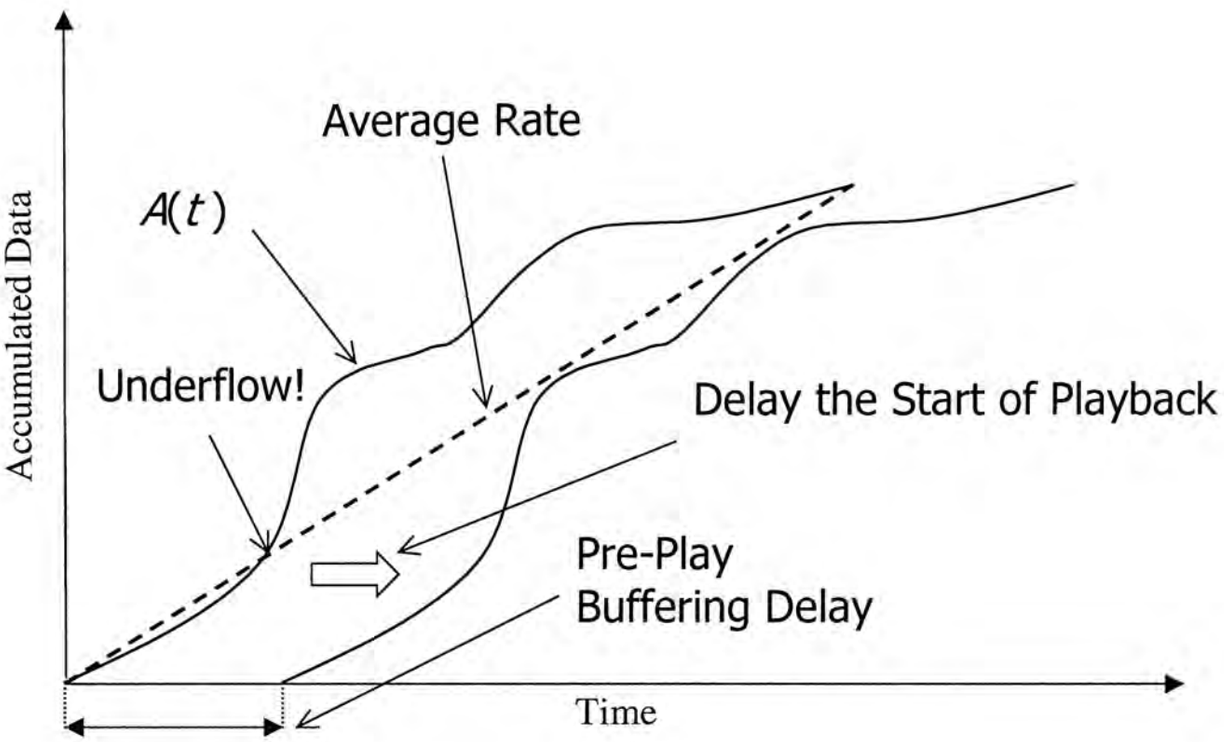


Figure 3 Average rate scheduler with pre-play buffering delay

Critical Rate Scheduler:

To remove the buffering delay introduced in Average Rate Scheduler, we can also perform admission control and schedule video data retrieval and transmission based on the critical rate of a video. The critical rate of a video is defined as the minimum constant rate to schedule a video delivery with neither startup delay nor data underflow. It is shown graphically in Figure 4, to be a straight line passing through the

origin, with the whole cumulative data consumption function underneath. The advantage of this scheduler over the Average Rate Scheduler is the zero startup delay, but the drawback is the larger buffer requirement.

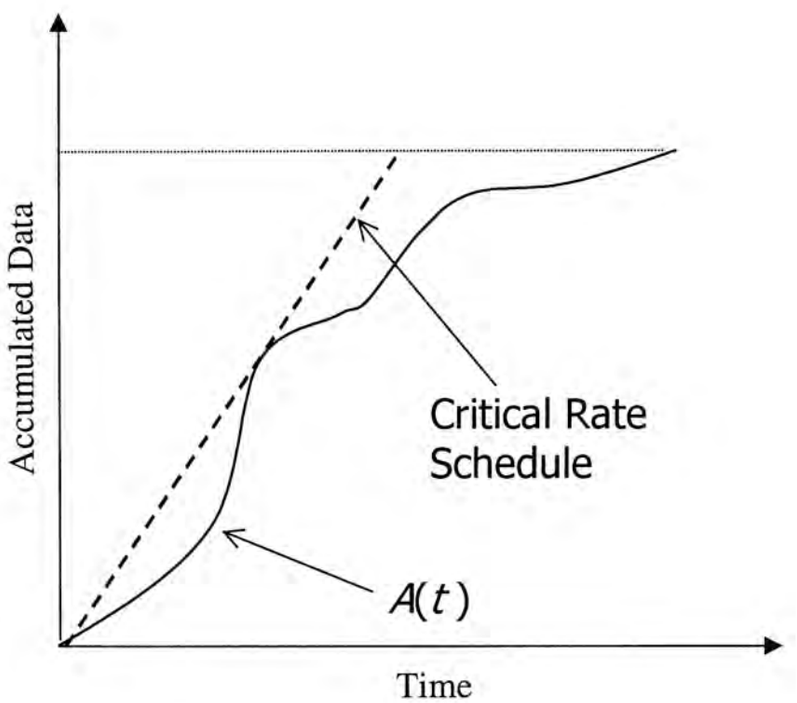


Figure 4 Critical rate scheduler

2.3 Brute Force Scheduler:

With the exact bit-rate trace of the pre-recorded videos known, it is also possible to perform admission control by exhaustively computing the load of all future disk rounds affected by the new stream. The stream is admitted only if none of these rounds are overloaded. Virtually zero client side buffer is required, but as discussed in Chapter 1, the admission control process is very computationally expensive. For the scheduling of disk retrieval and network transmission, we again need to exactly follow video bit-rate trace. This means network bandwidth renegotiations are required for each and every disk round.

2.4 Temporal Smoothing Scheduler:

Temporal smoothing is a technique to reduce the burstiness in the retrieval and transmission of VBR encoded videos. One typical method is to transmit at a bit-rate higher than the playback bit-rate during periods of smaller frame sizes. Excess video data are buffered at the client side so that the chance of having an increase in transmission bit-rate can be reduced when it comes to periods of larger frame sizes, as the client can draw video data from the buffer at that time. Figure 5 illustrates the effect of sliding window smoothing [3] with a window size of 5 seconds. At any time, the transmission rate is equal to the data rate averaged over the next 5 seconds. One can observe from the figure that this method can reduce the bit-rate fluctuations, as well as the peak data rate for considerable amounts.

Other methods include the use of hopping window smoothing [3], which applies smoothing on video segments of the same length, suitable for encoding video data with repetitive patterns like MPEG with GOPs; online smoothing that requires buffering on the server side as well [6], suitable for real time videos; and some other smart transmission algorithms that take into account the client buffer requirements

[4]-[5] and aim at minimizing the number of rate changes/increases in the transmission [10], or to schedule the transmission of multiplexed streams [5] without data underflow.

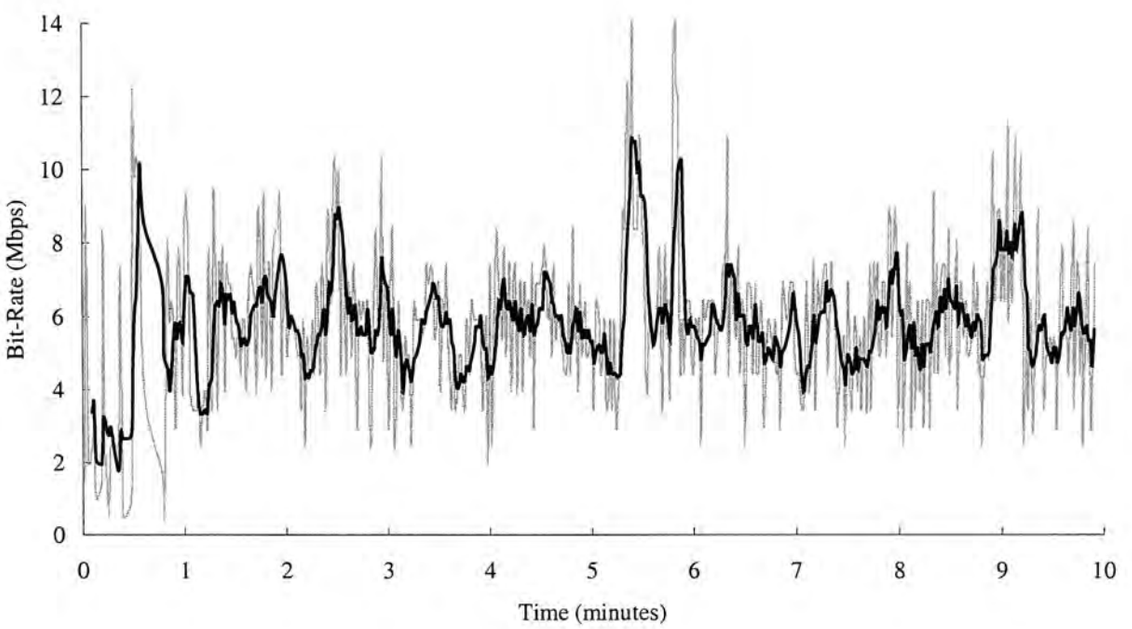


Figure 5 Effect of temporal smoothing with a sliding window of 5 seconds

The goal of temporal smoothing is to schedule the retrieval and transmission of video data to ensure playback continuity without starvation at the client. Define $A(t)$ as the cumulative data consumption function (see Figure 6), which represents the amount of data that need to be accumulated at the client side at time t. Apparently, any feasible

transmission schedules, denoted by $S(t)$, must be higher than $A(t)$ for all t so that the client will not run out of video data during playback:

$$S(t) \geq A(t) \tag{1}$$

In temporal smoothing algorithms, a system is often assumed to be buffer constrained or delay constrained, so another bounding function $B(t)$ is constructed, which is an upward shift of $A(t)$ by buffer size b, as shown in Figure 6(a); or a rightward shift of $A(t)$ by a delay bound d, as shown in Figure 6(b):

$$B(t) = A(t) + b, \qquad \text{for buffer constrained case} \tag{2}$$

$$B(t) = A(t - d), \qquad \text{for delay constrained case} \tag{3}$$

The transmission schedule $S(t)$ must be always lower than the bounding function $B(t)$ to satisfy the buffer or delay constraints:

$$S(t) \leq B(t) \tag{4}$$

At any time if $S(t)$ is higher than $B(t)$, the system will suffer from a client buffer overflow or a missed delay bound guarantee. In any case, a schedule $S(t)$ that satisfies the buffer or delay constraints (or both) and at the same time having no data underflow

must lie between the functions $A(t)$ and $B(t)$:

$$B(t) \geq S(t) \geq A(t) \tag{5}$$

The transmission (or retrieval) rate of the schedule generated in this way may increase or decrease within the length of the served video, as illustrated in Figure 6.

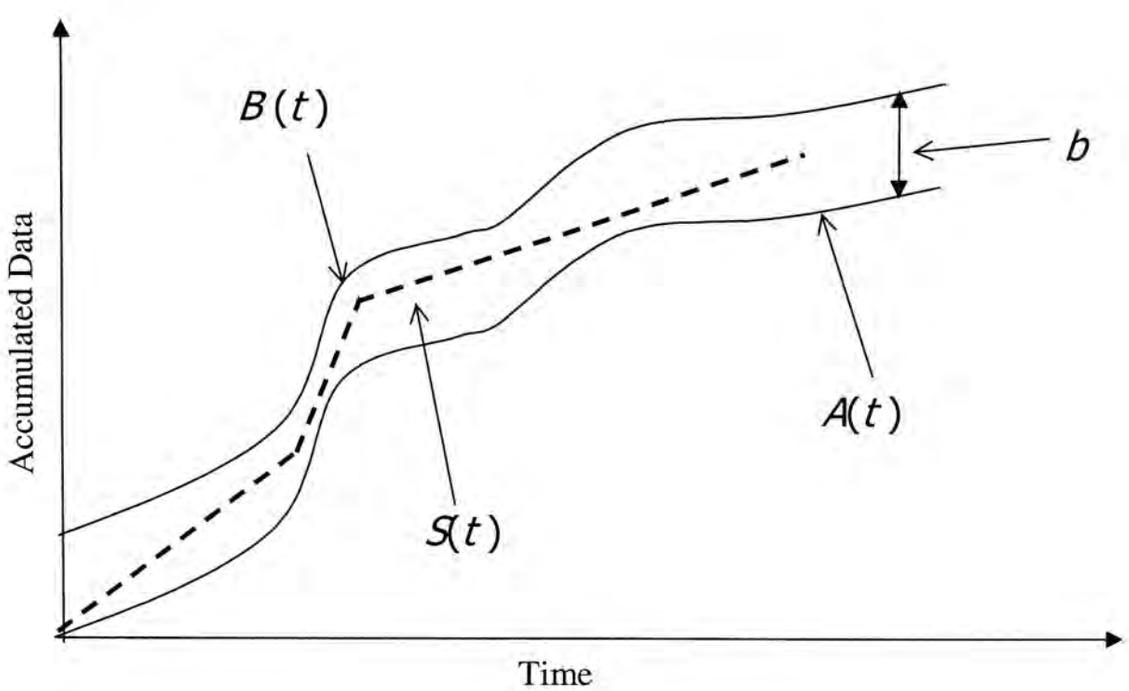


Figure 6 (a) Constructing a buffer constrained temporal smoothing schedule

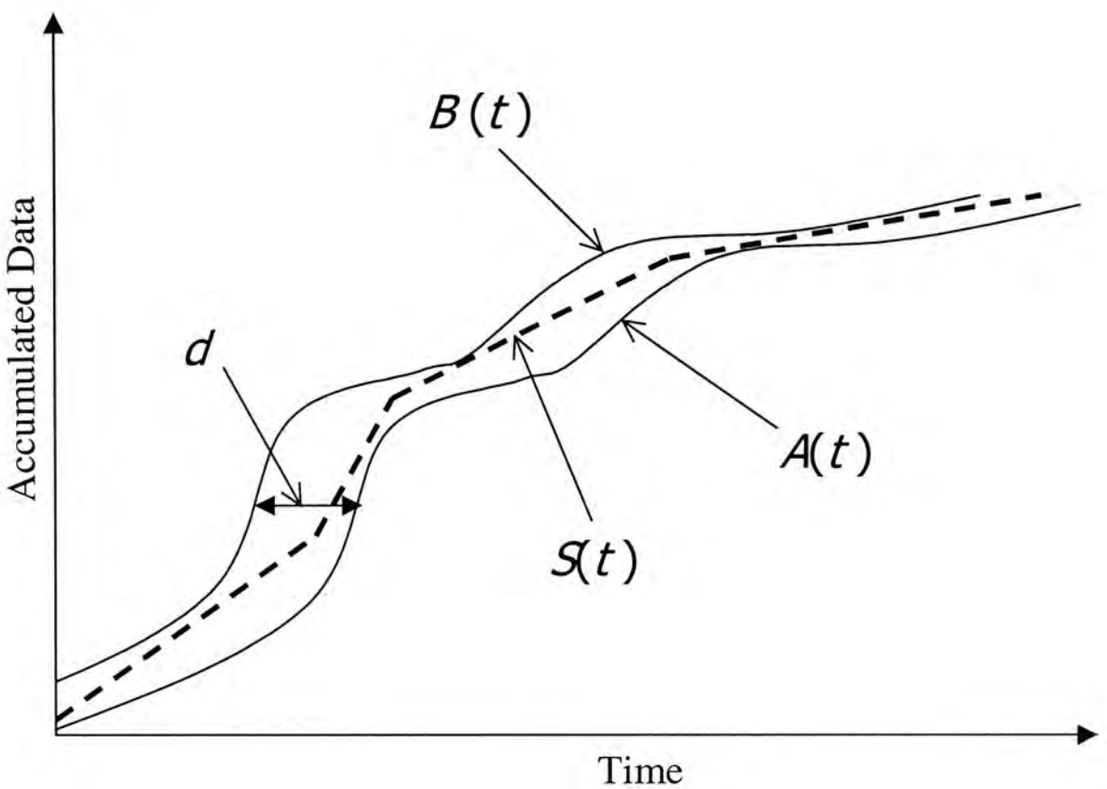


Figure 6 (b) Constructing a delay constrained temporal smoothing schedule

Many studies [2]-[6], [10] have been conducted to smooth this transmission schedule $S(t)$. Some recent development of network calculus [23]-[26] also addresses on the video smoothing problems. However, even after smoothing, part of the burstiness remains. As the efficient transmission of video data depends on the statistical multiplexing gain, server overload cannot be eliminated and one has to resort to statistical rather than deterministic performance guarantee. Last but not least, we observe that real-world VBR-encoded videos have vastly different long-range bit-rate

variations, rendering the statistical multiplexing gain to be highly dependent on the video mix.

On the other hand, to provide deterministic guarantee, one can check to ensure there is no overload for the entire duration of the new video stream on the fly during the admission process. This involves the superimposing of smoothed schedules for the new video stream and the existing streams to find the combined number of bit-rate changes in the aggregated traffic. Then each segment within the aggregated traffic must be checked against the system capacity to guarantee performance. The complexity depends on the shape of the cumulative data consumption function for individual videos and the number of videos that are currently being served. We study the Optimal Smoothing Algorithm [4], which is a buffer constrained temporal smoothing scheduler that minimizes the number of rate changes by extending each segment as far as possible. Performance evaluation results are shown in Chapter 4

Chapter 3

Decreasing Rate Scheduling

In this study, we propose a new Monotonic-decreasing Rate Scheduler (MDRS) to address the problems in VBR video delivery mentioned in the previous chapters. With MDRS, VBR videos can be delivered with no loss of visual quality, zero delay, minimum admission complexity and deterministic performance guarantee. Rather than using generic source models, MDRS computes the retrieval and transmission schedules offline, based on the bit-rate profile of each individual video to guarantee performance.

Unlike traditional temporal smoothing schedulers that generate schedules with both

bit-rate increases and decreases, MDRS assigns rates in a monotonic-decreasing manner, with the first assigned rate being the highest, and each subsequent rate being lower. The purpose of such a design is to ensure that once a stream is admitted, its load offered will only decrease with time. Hence during the admission control process, we only need to ensure the remaining network bandwidth is higher than the first rate r_0 of the requested video and that the first disk round is not overloaded after the new stream is added, then both the disk and network are guaranteed to have no overload during the entire playback duration. In this way, admission of a new stream requires the computation of only one disk round and the inspection of only the initial bit-rate in the schedule, thereby greatly simplifies the admission control process.

Another important advantage of MDRS is the capability of utilizing the remaining bandwidth to provide available bit-rate (ABR) service together with the VBR video service. If VBR video service is scheduled with other methods like temporal smoothing in which allocated bandwidth may increase or decrease, the originally scheduled upward bandwidth renegotiations of the video traffic may fail when other traffics are included in the system. On the other hand, with MDRS, it is safe to fully utilize any residual bandwidth unused by the video traffic without affecting the on

going video streams, as there are only downward bandwidth renegotiations within the schedules.

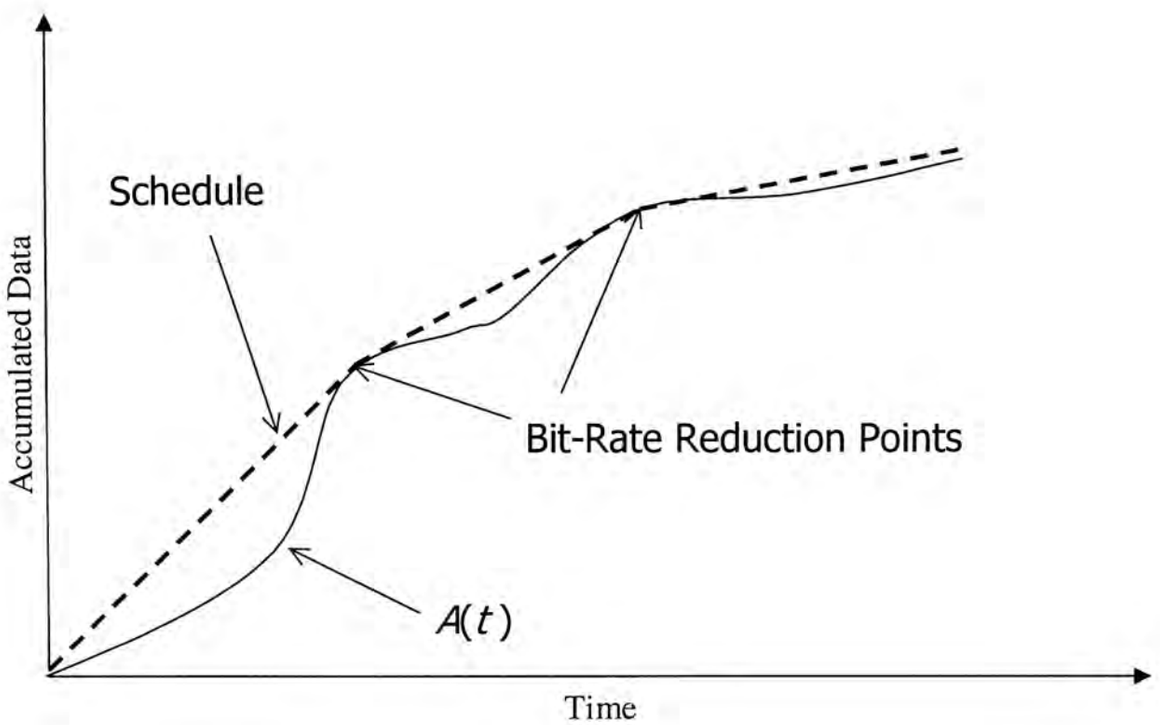


Figure 7 Constructing a decreasing rate schedule with minimal buffer requirement

3.1 MDRS with Minimum Buffer Requirement

We first present a general algorithm for off-line computation of the retrieval and transmission schedule with minimum buffer requirement. The algorithm assumes that the client has infinite buffer space. While this is clearly not true in practice, we show in Chapter 4.1 that the resultant buffer requirement is modest for real VBR-encoded videos, and also well within the capacity of a small hard disk, common in most set-top boxes and PCs.

To determine the schedule, we begin at the origin and scan through the data-consumption curve $A(t)$ to look for a point with the greatest slope when connected to the starting point (see Figure 7). This is then defined as the first bit-rate reduction point and will be used as the starting point to repeat the procedures until the end of the video is reached. The resultant output are a number of {bit-rate, time} tuples, which define the current bit-rate and the time for switching to the next bit-rate. Figure 8 shows the pseudo code of this scheduler.

$t = 0, i = 0$

while $(\tau < L)$ {

$$r_i = \max_{t_0 > \tau} \frac{A(t_0) - A(\tau)}{t_0 - \tau}$$

$T_i = t_0$

$\tau = T_i, i = i + 1$

}

output : $(r_0, T_0), (r_1, T_1) \ldots (r_n, T_n)$

L is the length of the video in seconds,
r_i and T_i are the bit-rate and transition time
for the i^{th} segment in the schedule.

Figure 8 Pseudo code for generating an MDRS Schedule

We now show that the schedule $S(t)$ generated with the algorithm we discussed has monotonic-decreasing rate allocations, and has the minimum buffer requirement among all monotonic-decreasing rate schedules.

Theorem 1: The MDRS algorithm generates schedules with monotonic-decreasing rate.

Proof: Let r_i and r_{i+1} are the i^{th} and $(i+1)^{th}$ segment of $S(t)$ generated with the MDRS algorithm, and let r_i' be the slope connecting $S(T_{i-1})$ and $S(T_{i+1})$. According to the MDRS algorithm, r_i should has the largest slope connecting $S(T_{i-1})$ to any point on $S(t)$ with t larger then T_{i-1}.

Assume $r_i < r_{i+1}$, i.e., the rate allocated are not monotonic-decreasing. Then we have:

$$\frac{S(T_i)-S(T_{i-1})}{T_i-T_{i-1}} < \frac{S(T_{i+1})-S(T_i)}{T_{i+1}-T_i} \tag{6}$$

Or:

$$[S(T_i)-S(T_{i-1})](T_{i+1}-T_i) < [S(T_{i+1})-S(T_i)](T_i-T_{i-1}) \tag{7}$$

Expand (7) to give:

$$\begin{aligned} &S(T_i)T_{i+1}-S(T_i)T_i-S(T_{i-1})T_{i+1}+S(T_{i-1})T_i < \\ &\quad S(T_{i+1})T_i-S(T_{i+1})T_{i-1}-S(T_i)T_i+S(T_i)T_{i-1} \end{aligned} \tag{8}$$

Cancel the $S(T_i)T_i$ term on both sides and re-arrange:

$$S(T_i)T_{i+1}-S(T_{i-1})T_{i+1}-S(T_i)T_{i-1} < S(T_{i+1})T_i-S(T_{i+1})T_{i-1}-S(T_{i-1})T_i \tag{9}$$

Add $S(T_{i-1})T_{i-1}$ on both sides to give:

$$\begin{aligned} &S(T_i)T_{i+1} - S(T_{i-1})T_{i+1} - S(T_i)T_{i-1} - S(T_{i-1})T_{i-1} < \\ &\quad S(T_{i+1})T_i - S(T_{i+1})T_{i-1} - S(T_{i-1})T_i - S(T_{i-1})T_{i-1} \end{aligned} \tag{10}$$

Factorize (10) and we get:

$$[S(T_i) - S(T_{i-1})](T_{i+1} - T_{i-1}) < [S(T_{i+1}) - S(T_{i-1})](T_i - T_{i-1}), \tag{11}$$

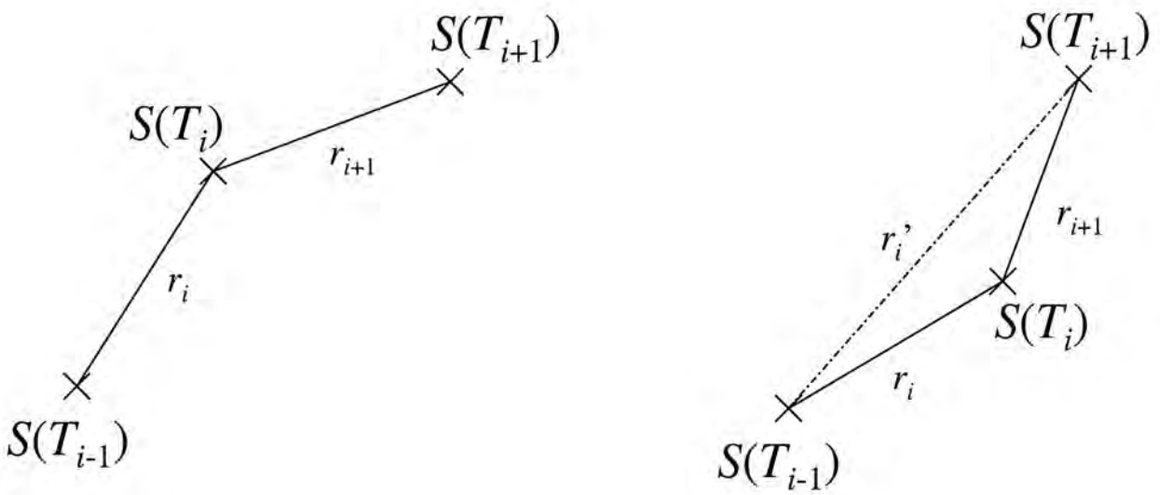
which is equal to:

$$\frac{S(T_i) - S(T_{i-1})}{T_i - T_{i-1}} < \frac{S(T_{i+1}) - S(T_{i-1})}{T_{i+1} - T_{i-1}} \tag{12}$$

This gives:

$$r_i < r_i' \tag{13}$$

This contradicts with our previous assumption that r_i has the largest slope, and we conclude that the rates allocated by MDRS are monotonic-decreasing.



(a) (b)

Figure 9 Graphical illustration of the rate monotonicity proof

(a) $r_i > r_{i+1}$ (b) If $r_i < r_{i+1}$, i^{th} segment should have been r_i' instead.

Theorem 2: The MDRS generates schedules with minimum buffer requirement among all monotonic-decreasing rate schedules.

Proof: We proof by contradiction. Let $X(t)$ be a monotonic-decreasing rate schedule, which is always above the cumulative data consumption function $A(t)$:

$$X(t) \geq A(t) \tag{14}$$

Assume $X(t)$ has lower buffer requirement than $S(t)$, so:

$$S(t_0) > X(t_0) \geq A(t_0), \qquad \forall\ t_0 \tag{15}$$

$X(t)$ cannot be lower than $S(t)$ at the bit-rate reduction point T_i's:

$$t_0 \neq T_i, \qquad \text{for } i = 0, 1, 2, \ldots \tag{16}$$

As $S(t)$ coincides $A(t)$ at the bit-rate reduction points:

$$S(T_i) = A(T_i), \qquad \text{for } i = 0, 1, 2, \ldots \tag{17}$$

But $S(t)$ is constructed with straight lines connecting the bit-rate reduction points, so $X(t)$ cannot be lower than $S(t)$ in between 2 bit-rate reduction points:

$$t_0 \notin (T_{i-1}, T_i), \qquad \text{for } i = 1, 2, 3, \ldots \tag{18}$$

Otherwise $X(t)$ has to be concave in the range (T_{i-1}, T_i), which contradicts with the assumption that $X(t)$ has monotonic-decreasing rates. We conclude that function $X(t)$ does not exist and $S(t)$ is minimum among all functions above $A(t)$ with monotonically decreasing slope.

3.2 *2*-Rate MDRS

The previous MDRS algorithm generates the schedule with the minimum buffer requirement (we refer it to MDRS-Min hereafter). A side effect is that many rate reductions maybe required for the schedule. We note that each rate reduction also requires network-bandwidth re-negotiation to reduce the amount of bandwidth reserved. Since bandwidth negotiation requires a network control node to participate, too many renegotiations can overwhelm the network. To circumvent these problems, we can limit the number of bit-rate changes in the schedules to be fixed at two. We evaluate and compare the performance with the MDRS-Min in Chapter 4.

The only single requirement in constructing a *2*-Rate schedule is that the first rate r_0 assigned, which is also the highest rate, needs to be higher than or equal to the critical rate of the video. Then given two predetermined rates, r_0 and r_1, ($r_0 > r_1$ and $r_0 \geq$ critical rate), the transition point T_0 for switching the bit-rate r_0 to r_1 that attains the minimum buffer requirement is the earliest possible transition point. Delaying the transition can never reduce the buffer requirement. As illustrated in Figure 10, shifting segment r_1 to the right (delaying the transition point) is equivalent

to shifting it upwards, which can only increase the buffer requirement.

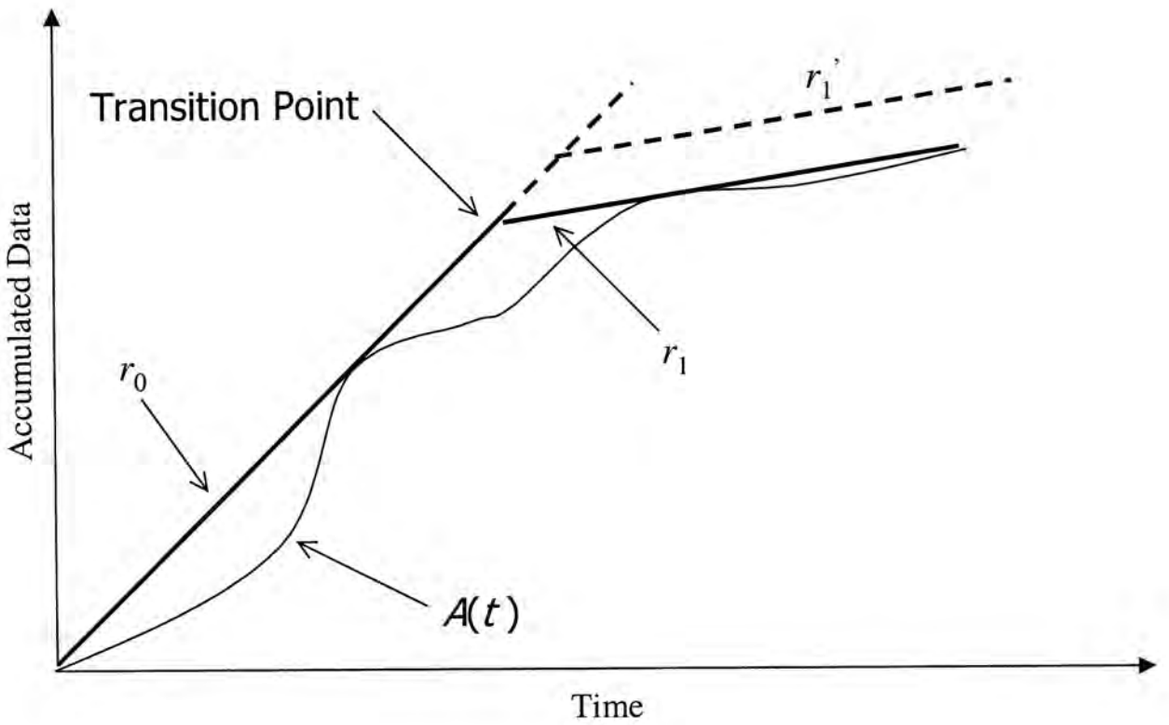


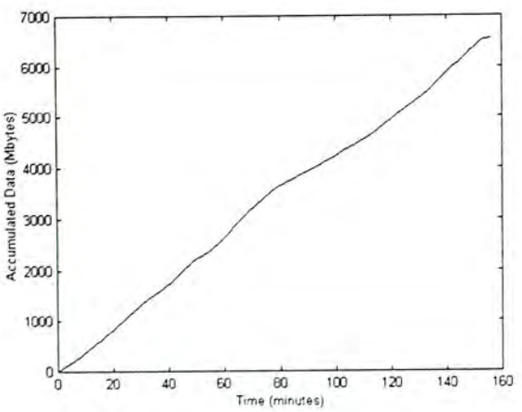
Figure 10 Finding the earliest transition point in a 2-Rate MDRS
This figure shows that delaying the transition to r_1 segment is essentially the same as upward shifting the r_1 segment.

Since the schedule can be generated offline, we can use exhaustive search of all possible bit-rates combinations r_0 & r_1, with step size equals to the lowest bit-rate when only one sector is read in a disk round to find the optimal bit-rates and the corresponding transition point that minimizes the client size buffer requirements.

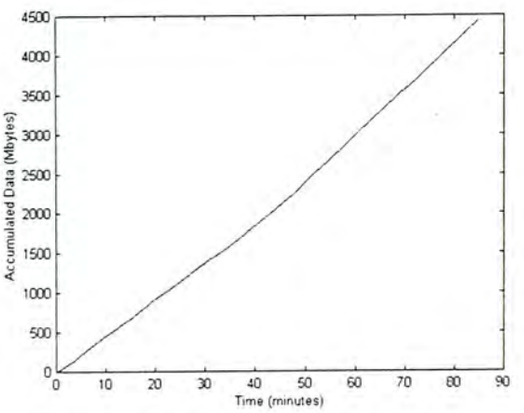
Chapter 4

Performance Evaluation

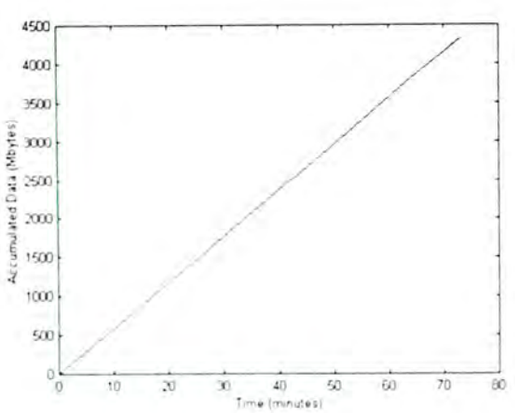
To obtain realistic performance results, we built a VBR video bit-rate archive with a wide-variety of 307 different video titles for simulation. These are full-length, MPEG-2 encoded movies with bit-rate variations ranging from 0.408 Mbps to over 18.757 Mbps and an average length of 6120 seconds. Long-range (tens of minutes) bit-rate variations are common in these real MPEG-2 encoded videos. Figure 11 shows some sample plots of cumulative data consumption functions in different shapes from four different movies.



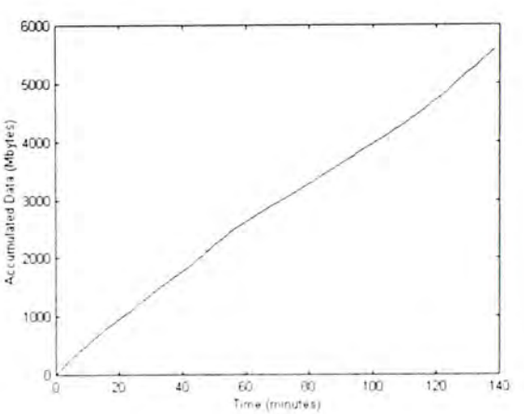
(a) irregular



(b) convex



(c) relatively linear



(d) concave

Figure 11 Sample plots of cumulative data consumption functions

4.1 Buffer Requirement

We first compute the client buffer requirements serving titles in our VBR video bit-rate archive with the MDRS-Min, 2-rate MDRS, Critical Rate, Average Rate, Peak Rate and Brute Force Schedulers, and summarize the results in Figure 12. The buffer requirement of the Temporal Smoothing Scheduler is not compared, as one can explicitly specify it to be any value. The effect of different buffer size for this scheduler is discussed in later chapters. As expected, the 2-Rate MDRS requires more client buffer than the MDRS-Min, though the requirements are in fact very close, and both require much less buffer than the Critical Rate and Average Rate Schedulers. On average, the 2-Rate MDRS requires 89.61 Mbytes of client buffer and the MDRS-Min requires 82.57 Mbytes, compared to 458.61 Mbytes and 133.44 Mbytes for the Critical Rate and Average Rate Schedulers respectively. Note that the buffer requirement of the Average Rate Scheduler is close to the 2-Rate and MDRS-Min only in modest cases. The worst cases' buffer requirement for the Average Rate Scheduler is very high, due to the need of a pre-play buffering time, which builds up a considerable amount of buffer. On the other hand, the Peak Rate and Brute Force

Schedulers require virtually no buffer space, as discussed in the previous chapter.

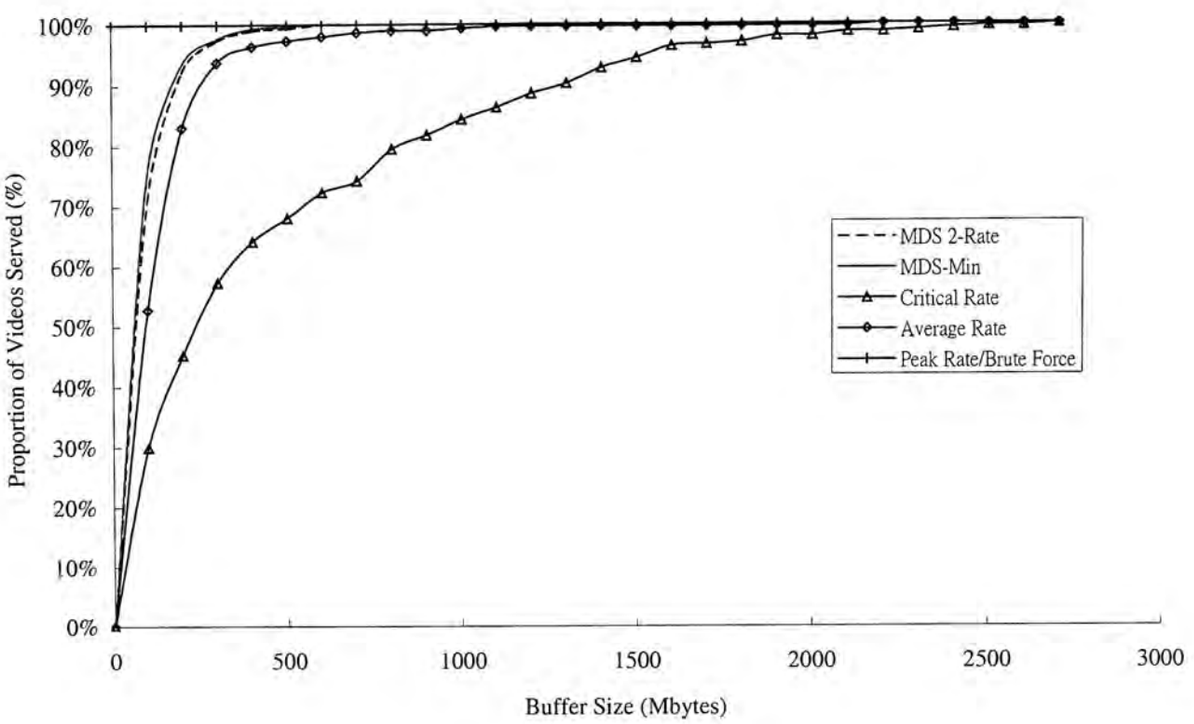


Figure 12 Number of videos that can be served with a given buffer size

(total 307 VBR videos tested)

For the 2-Rate and MDRS-Min, out of 307 tested videos, 75% of the tested videos require about 100M bytes of buffer, and over 99% (304 and 305 for 2-Rate MDRS and MDRS-Min respectively) require no more than 400 Mbytes of client buffer. The exceptions in the 2-Rate MDRS require 457.05, 596.26 and 597.24 Mbytes, while

those in the MDRS-Min require 456.78 Mbytes and 596.91 Mbytes. Compared with the Critical Rate and Average Rate Schedulers, with 100 Mbytes of buffer we can only serve about 30.1% and 52.9% of videos respectively. With 400 Mbytes of buffer, we can serve about 64.4% and 96.7% of videos. The worst-case buffer requirements are 597.24, 596.91, 2605.25 and 2128.56 Mbytes for 2-Rate MDRS, MDRS-Min, Critical Rate and Average Rate Schedulers respectively.

Given that future set-top boxes will serve as entertainment centers equipped with web browsing, gaming and other multimedia services, a local hard disk is needed anyway and hence the buffer required for our proposed schedulers can easily be accommodated.

4.2 Startup Delay

As we have mentioned in Chapter 4.1, the Average Rate Scheduler requires additional pre-play buffering delay during startup, which is not required by other schedulers we studied. We computed the delay with the bit-rate profiles in our VBR video bit-rate archive, and found that it requires 115.07 seconds of pre-play buffering delay on average. This is very undesirable, and worst still, the largest delay we found is over 4865 seconds. We observe the bit-rate profiles that generate average rate schedules among the worst-case pre-play buffering delays, and find that such schedules with pre-play buffering delays of tens of minutes generally have concave shaped cumulative data consumption curves. Apparently, such videos are better suited with MDRS.

4.3 Disk Utilization

To evaluate the disk utilization achieved by the proposed scheduler, we simulated a system with 10 Quantum Atlas10K 9GB hard disks in a spindle synchronized disk array, storing 20 full-length videos that totals 90 GB, which are sequentially arranged. The average video bit-rate is 5.86 Mbps and the transfer rate of the hard disk drives specified by the manufacturer is 144 Mbps (so the transfer rate of our simulated disk array is 1.44 Gbps). User requests are generated according to a Poisson process with mean inter-arrival time of 25 seconds. Each request selects a random video with uniform probability. The simulation is then run for a simulated time of one day and multiple simulations run with randomly selected random seeds are averaged to obtain the final results.

For simulation purpose, we ignore the computation time required by the admission process, and schedule all streams to start in the disk round right after the requests are received. We observe the average disk throughput, the number of concurrent video streams served and the disk utilization calculated from the disk throughput and the transfer rate of the disk drives. However, we noticed that it is not adequate to compare

the utilization just with the disk transfer rate, as it is the guaranteed disk-retrieval rate reading a continuous block. Parameters like seeking overhead in the multi-stream scenario have not been taken into account. Therefore, we employ the Maximum Utilization Scheduler (see Appendix) to find the maximum achievable disk throughput (131.45 Mbps) when concurrent streams are served, and use that as another indicator to compare the disk efficiency of the various schedulers. Note that this variant of the Critical Rate Scheduler, which has the highest disk throughput among our tested schedulers (see Figure 13), only exploits the unused round time to increase disk throughput, and will not affect the fairness or policy of admission.

Figure 13 tabulates the performance of the various schedulers. The numbers for the Temporal Smoothing Scheduler are averaged from the results of its 20, 40, 60, 80 and 100-Mbytes cases. It is shown that the buffer size used in Temporal Smoothing Scheduler does not affect the disk utilization. Therefore, we use the average value to compare the performance with other schedulers.

Scheduler	MDRS-Min	MDRS 2-Rate	Average Rate	Critical Rate	Peak Rate	Brute Force	Temporal Smoothing[3]
Disk Throughput (Mbps)	122.54	122.58	121.75	125.90	46.88	118.18	122.59
Utilization[1]	85.09%	85.12%	84.55%	87.43%	32.56%	82.07%	85.13%
Efficiency[2]	93.22%	93.25%	92.62%	95.78%	35.66%	89.90%	93.26%
Average Buffer Requirement (Mbytes)	89.61	82.57	133.44	458.61	0.00	0.00	---
Maximum Buffer Requirement (Mbytes)	596.91	597.24	2128.56	2605.25	0.00	0.00	---
Average Startup Delay (Seconds)	0.00	0.00	115.07	0.00	0.00	0.00	0.00
Maximum Startup Delay (Seconds)	0.00	0.00	4865.48	0.00	0.00	0.00	0.00

1. With respect to the transfer rate of the hard disk (144Mbps)
2. With respect to the maximum achievable disk throughput (131.45Mbps)
3. Averaged values

Figure 13 Performance comparison of various schedulers

Among the schedulers we tested, the Critical Rate Scheduler has the best performance, and the Peak Rate Scheduler has the worst performance. On the other hand, the 2-Rate and MDRS-Min we proposed perform almost equally well, with over 85% utilization with respect to the transfer rate of the hard disk drives, and over 93% efficiency with respect to the maximum achievable disk throughput. The MDRS has only about 2.5% lag in performance compared to the Critical Rate Scheduler, which has the worst

buffer requirement, and has performance better than the Average Rate and Brute Force Schedulers, which requires substantial startup delay and high complexity respectively. The performance of the Temporal Smoothing Scheduler is very close to our proposed schedulers, but has higher complexity (see Chapter 4.4).

4.4 Complexity

Finally, we compare the complexity of the different schedulers. The admission complexity, as well as the scheduling complexity will both be compared. Admission complexity means the number bit-rate changes in the aggregated video traffic that we need to inspect for overloading when the request of a new video stream arrives. Scheduling complexity means the number of bit-rate changes that exists in the aggregated video traffic, which also means the number of bandwidth renegotiations sent to the network. Apparently, both complexities are related to the number of bit-rate changes within the schedules. So we first look at the number of segments required by the different schedulers.

The number of segments, and thus the bit-rate changes in temporal smoothing schedules, decreases with increased buffer size, and reaches the floor when the buffer size exceeds a certain limit. Among the videos we tested, at a buffer size of 20 Mbytes, there are 24.1 segments on average and the maximum number is 79. At a buffer size of 100 Mbytes, the average number is 9.4 and the maximum number is 31. The average number and maximum number of segments both exhibit very little decrease when the

buffer size is above 160 Mbytes, at which the numbers are 8.8 and 24 respectively. In both the average and maximum cases, the asymptotic values are very close to the MDRS-Min, which requires 9.2 segments on average, and 24 segments in the worst case. The number required by the 2-Rate MDRS is, of course, always 2. One should also note that in generating temporal smoothing schedules, we fix the available buffer size and then find the corresponding schedule and number of turning points from each video bit-rate profile, as opposed to the MDRS-Min case where the number of turning points and buffer size required by a schedule depend solely on shape of that particular video bit-rate profile. (The number of turning points in a 2-Rate schedule is deliberately limited to 2)

Next, we find the number of bit-rate changes required by the schedulers, which represents the scheduling complexity. Assume a server serves S simultaneous streams on average, and each stream has an average of K segments over the length of the stream equivalent to R disk rounds. In each round, the probability of a *stream* having a bit-rate change is K/R, so the expected number of bit-rate changes in the *system* within one round is SK/R. Then the total number of retrieval rate changes or bit-rate renegotiations required over an average stream length of R rounds is SK. The Brute

Force Scheduler and Peak Rate Schedulers are two exceptions, which are the worst-cases that require re-negotiation every round, so the number is *R*. The scheduling complexities of different schedulers are tabulated in Figure 15

For the admission complexity, we need to find the number of segments in the aggregated video traffic to be inspected when a request to a new video stream arrives. This is equal to the number of bit-rate changes of the new stream added to the remaining number of bit-rate changes of the on-going streams. So for the Temporal Smoothing Scheduler, the number of segments is equal to half the number of bit-rate changes of the whole system over one stream length (*SK/2*). With MDRS or the three constant bit-rate schedulers (Average Rate, Peak Rate and Critical Rate Schedulers), no bit-rate increase will exist within the schedules. This means that when a new stream commences, it is only necessary to inspect the first segment to guarantee there is no overload throughout the entire playback duration. Therefore, the number of inspections required is only one. The number of inspections for the Brute Force Scheduler is again equal to the number of rounds *R*. This summarized in Figure 14.

The MDRS achieves the lowest possible admission complexity of one bandwidth

comparison against the system capacity for each stream, the same with that of the CBR schedulers. The Brute Force Scheduler, as expected, has the highest admission complexity, which requires an average of over 6000 times of computations compared to the MDRS. The Temporal Smoothing Scheduler, at buffer requirement as low as 20 Mbytes, requires about 12S computations. With higher buffer requirement of about 100 Mbytes which is already enough for serving about 75% of the test videos with MDRS as shown in Chapter 4.1, the Temporal Smoothing Scheduler requires about 4.7S computations, and the smallest possible number of computations (4.4S) is achieved at buffer requirement of 160 Mbytes or above, at which further increasing the buffer requirement will not reduce the admission complexity. At such high buffer requirement requirements, disk storage is very probably needed to do the buffering anyway, so it is more sensible to use MDRS, which achieves lower admission complexity.

Scheduler	Avg. Number of Segments per Stream	Max. Number of Segments per Stream	Avg. Admission Complexity
MDRS-Min	9.2	24	1*
MDRS 2-Rate	2	2	1*
Peak Rate	1	1	1*
Average Rate	1	1	1*
Critical Rate	1	1	1*
Brute Force	6120	14586	6120
Temp. Smoothing (20 Mbytes buffer)	24.1	79	12.0*S*
Temp. Smoothing (100 Mbytes buffer)	9.4	31	4.7*S*
Temp. Smoothing (160 Mbytes buffer)	8.8	24	4.4*S*
Temp. Smoothing (600 Mbytes** buffer)	8.8	24	4.4*S*

*These are worst case numbers

**The approximate buffer size required for MDRS to serve all tested videos (*S* is the average no. of concurrent streams in system)

Figure 14 Admission complexities of different schedulers

For scheduling complexity, the Brute Force Scheduler again requires much heavier computations. The MDRS-Min, although not as low as the ideal case of CBR schedulers, achieves a complexity (9.2*S*) close to the Temporal Smoothing Scheduler with about 100 Mbytes buffer requirement (9.4*S*), and about 4.5% more than the

minimum achievable complexity of 8.8*S* at buffer requirement 160 Mbytes or above. A further reduction in complexity of about 78% (from 9.2*S* to 2*S*) can be achieved with the *2*-Rate MDRS, which suffers only from a tiny trade-off in buffer requirement.

Scheduler	Avg. Number of Segments per Stream	Max. Number of Segments per Stream	Avg. Scheduling Complexity
MDRS-Min	9.2	24	9.2*S*
MDRS *2*-Rate	2	2	2*S*
Peak Rate	1	1	*S*
Average Rate	1	1	*S*
Critical Rate	1	1	*S*
Brute Force	6120	14586	6120
Temp. Smoothing (20 Mbytes buffer)	24.1	79	24.1*S*
Temp. Smoothing (100 Mbytes buffer)	9.4	31	9.4*S*
Temp. Smoothing (160 Mbytes buffer)	8.8	24	8.8*S*
Temp. Smoothing (600 Mbytes** buffer)	8.8	24	8.8*S*

**The approximate buffer size required for MDRS to serve all tested videos

(*S* is the average no. of concurrent streams in system)

Figure 15 Scheduling complexities of different schedulers

Chapter 5

Conclusion

By allocating disk and network bandwidth in a monotonic-decreasing manner, we addressed three key challenges in VBR video delivery, namely minimizing admission control complexity, eliminating startup delay, and providing deterministic performance guarantee. Through extensive simulations using real-world VBR-encoded videos, the proposed schedulers are shown to achieve good disk efficiency that is comparable to the Temporal Smoothing Scheduler, which requires higher scheduling and admission complexities; and also comparable to CBR schedulers like Critical Rate and Average Rate Schedulers, which require unrealistic

buffer size and substantial startup delay respectively. The proposed monotonic-decreasing rate schedulers enable the use of a very simple admission controller, have zero start-up delay, and still provide deterministic performance guarantees. Moreover, it is possible to exploit the remaining bandwidth from the VBR video service to provide ABR data service without affecting the scheduled video transmissions. Given that local storage is likely to be abundant in future set-top boxes, the proposed scheduler provides an efficient solution for delivering high-quality, VBR-encoded videos in future video-on-demand services.

Appendix

Maximum Utilization Scheduler

The Maximum Utilization Scheduler is based on the Critical Rate Scheduler to find the maximum achievable disk utilization under realistic multi-streams scenario. Like the Critical Rate Scheduler, it performs admission control and schedules retrieval based on the Critical Rate of the requested video. The difference is that it will keep track of the video buffer accumulated at the client side. Then in each round, besides serving the scheduled retrievals, the stream with the least buffer accumulated will exhaust all remaining round time to boost disk utilization. This scheduler may have possible problems of large client side buffer requirement and unfairness among different streams, but for comparison purposes, we studied this scheduler to see how

MDRS compare to the maximum achievable disk utilization.

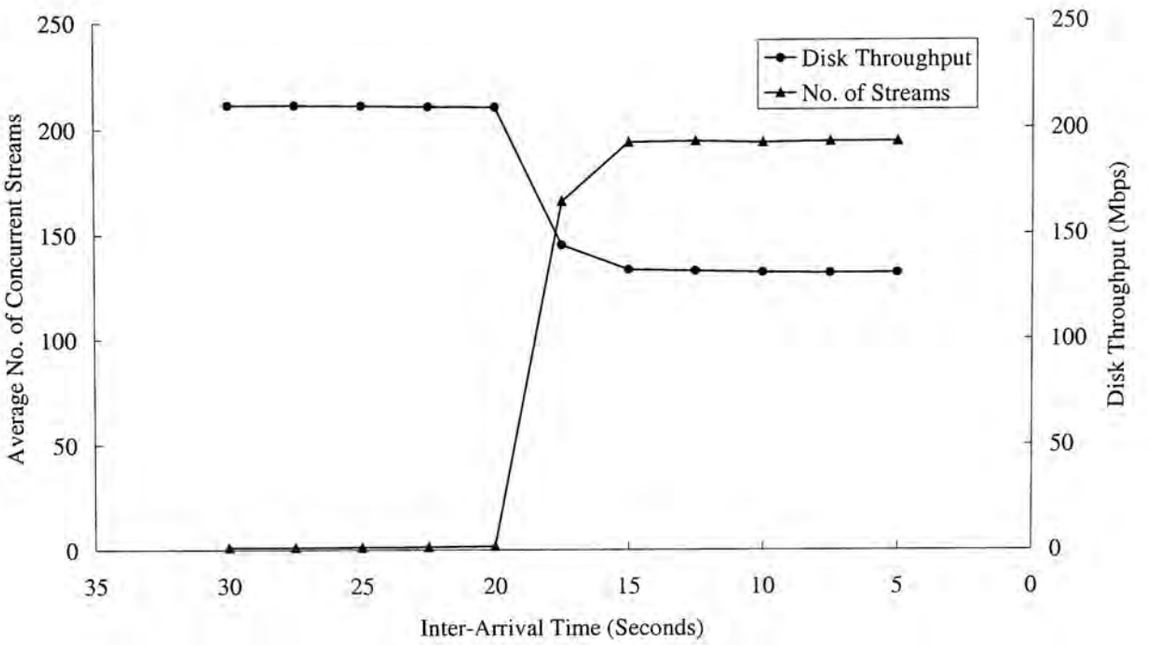


Figure 16 Performance of the maximum utilization scheduler

Figure 16 shows the maximum achievable disk throughput at different inter-arrival times. With long inter-arrival times, a video stream can be completed before the request to new stream comes. The server is essentially serving only one client most of the time, doing sequential reading. To find the maximum achievable disk throughput in a realistic multi-stream scenario, we reduce the inter-arrival time until the system is

saturated. Observing that both the average number of concurrent streams and the average disk throughput level off when the inter-arrival time is smaller than 10 seconds, we conclude that the maximum achievable disk throughput is approximately 131.45 Mbps.

Bibliography

[1] T. V. Lakshman, Antonio Ortega and Amy R. Reibman, "VBR Video: Tradeoffs and Potentials", *Proceedings of the IEEE*, Vol. 86, No. 5, May 1998.

[2] Dallas E. Wrege, Edward W. Knightly, Hui Zhang and Jörg Liebeherr, "Deterministic Delay Bounds for VBR Video In Packet-Switching Networks: Fundamental Limits and Practical Trade-Offs", *IEEE/ACM Transactions on Networking*, pp.352-362, vol.4(3), 1996.

[3] J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic and D. Towsley, "Online Smoothing of Live, Variable-Bit-Rate Video", *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 249-258, May, 1997.

[4] James D. Salehi, Zhi-Li Zhang, James F. Kurose, and Don Towsley, "Supporting Stored Video: Reducing Rate Variability and End-To-End Resource Requirements Through Optimal Smoothing", *Proceedings of the ACM SIGMETRICS Conference on Measurement & modeling of computer systems*, pp.222-231, 1996.

[5] Wei. Zhao and Satish K. Tripathi, "Bandwidth-Efficient Continous Media Streaming Through Optimal Multiplexing", *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, pp. 13-22,

1999.

[6] Simon S. Lam, Simon Chow and David K. Y. Yau, "A Lossless Smoothing Algorithm for Compressed Video", *IEEE/ACM Transactions on Networking*, pp.697-708, vol.4(5), 1996.

[7] S. C. Liew and Derek C. Y. Tse, "A Control-Theoretic Approach to Adapting VBR Compressed Video for Transport over a CBR Communications Channel", *IEEE/ACM Transactions on Networking*, pp.42-55, vol.6(1), 1998.

[8] P. P. Mishra, "Fair Bandwidth Sharing for Feedback Controlled VBR Video Traffic", *IEEE Global Telecommunications Conference*, 1995, pp. 1105-1108, vol. 2, 1995.

[9] N. G. Duffield, K. K. Ramakrishnan and Amy R. Reibman, "SAVE: An Algorithm for Smoothed Adaptive Video Over Explicit Rate Networks", *IEEE/ACM Transactions on Networking*, pp.717-728, vol.6(6), 1998.

[10] W. C. Feng and J. Rexford, "Performance Evaluation of Smoothing Algorithms for Transmitting Prerecorded Variable-Bit-Rate Video", *IEEE Transactions Multimedia*, pp.302-313, Sept. 1999.

[11] D. J. Gemmell, H.M. Vin, D. D. Kandlur, P. V. Rangan, and L.A. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, pp.40-49, vol.28(5), May 1995.

[12] H. J. Lee and D. H. C. Du, "The Effect Of Disk Scheduling Schemes On A Video Server For Supporting Quality MPEG Video Accesses", *Proceedings of the IEEE International Conference on Multimedia Computing and Systems '97*, pp.194-201, 1997.

[13] E. W. Knightly, "H-BIND: a new approach to providing statistical performance guarantees to VBR traffic", *IEEE Proceedings of INFOCOM '96*, vol. 3, pp. 1091 –1099, 1996.

[14] Mark W. Garrett and Walter Willinger, "Analysis, Modelling and Generation of Self-Similar VBR Video Traffic", *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, pp. 269-280, 1994.

[15] Dongying, Xuyong, Zhanglicui and Liuyan, "Modeling and the simulation study of MPEG video traffic", *Proceedings of the 3rd World Congress on Intelligent Control and Automation 2000*, pp. 2158 –2162, vol.3, 2000.

[16] H.S. Wang and Y.H. Wang, "A Generic Video Source Model and Its Statistical Multiplexing Properties," *IEEE Workshop on Visual Signal Processing and Communications*, September 19-20, 1994.

[17] D. P. Heyman, "The GBAR source model for VBR videoconferences", *IEEE/ACM Transactions on Networking*, pp. 554 –560, vol. 5, issue 4 , 1997.

[18] D. P. Heyman, T. V. Lakshman, "Source Models for VBR Broadcast-Video Traffic", *IEEE/ACM Transactions of Networking*, pp. 40-48, vol. 4, issue 1, 1996.

[19] Hai Liu, N. Ansari, Y. Q. Shi, "A simple model for MPEG video traffic", *2000 IEEE International Conference on Multimedia and Expo*, pp. 553–556, vol. 1, 2000.

[20] A. A. M. Ibrahim, "Efficient connection admission control for real-time video multiplexing", *1st IEEE International Conference on ATM*, pp. 66 – 70, 1998.

[21] Edward W. Knightly and Hui Zhang. "D-BIND: an accurate traffic model for providing QoS guarantees to VBR traffic", *IEEE/ACM Transactions on Networking*, pp. 219-231, vol. 5, issue 2, 1997.

[22] D.T. Hoang and J.S. Vitter, "Multiplexing VBR Video Sequences onto a CBR Channel with Lexicographic Optimization", *International Conference on Image Processing '97*, pp. 369-372, vol.1, 1997.

[23] Patrick Thiran, Jean-Yves Le Boudec, Frederic Worm, "Network Calculus Applied to Optimal Smoothing", To appear on *IEEE Proceedings of INFOCOM 2001.*

[24] Jean-Yves Le Boudec and O. Verscheure, "Optimal Smoothing for Guaranteed Service," *IEEE/ACM Transactions on Networking*, pp. 689-696, vol. 8, issue 6, December 2000.

[25] H. Sariowan, R. L. Cruz, G. C. Polyzos, "SCED: a generalized scheduling policy

for guaranteeing quality-of-service", *IEEE/ACM Transactions on Networking*, pp.669 –684, vol. 7, issue 5, October 1999.

[26] C. S. Chang, "Performance Guarantees in Communication Networks", Chapter 3, Springer-Verlag, 2000