Paper # 2916-31, Proceedings of Multimedia Storage and Archiving Systems, SPIE Photonics East '96 Symposium, Boston, Nov 1996

A New Class of Constant Data Length Retrieval Algorithms for Video Servers with VBR Streams

Ernst W. BIERSACK, Frédéric THIESSE
Institut Eurécom, 2229 Route des Crêtes,
06904 Sophia-Antipolis — France
Phone: +33 93002611
FAX: +33 93002627
email: erbi@eurecom.fr

ABSTRACT

We consider the problem of data retrieval from disk storage in video server where the video data are read in constant size blocks. Retrieval algorithms of this type are referred to as Constant Data Length (CDL) retrieval algorithms We recently introduced a new retrieval algorithm called GCDL [3] that generalizes the CDL retrieval algorithm: GCDL reads for a video stream *i* during $k \cdot m_i$, $k \in \{1, 2, 3, \ldots\}$, *consecutive* disk rounds a constant size block from the disk, which may result in a large read-ahead requiring a large amount of buffer. In this paper, we propose two new retrieval algorithms called static and dynamic GCDLb that minimize the number of reads during consecutive disk rounds while still maintaining continuous delivery to the client. Compared to GCDL, we show that GCDLb requires less buffer per client and can admit more clients.

Keywords: Video server, disk storage, disk retrieval, admission control, quality of service, variable bit rate video.

1. INTRODUCTION

Video servers store digitized, compressed continuous media information on secondary or tertiary storage. The secondary storage devices allow random access and provide short seek times compared to tertiary storage. Video server design differs significantly from that of traditional data storage servers due to the large size of the objects stored and the real-time requirements for their retrieval. A video server must meet the requirements that stem from the continuous nature of audio and video and must guarantee the delivery of continuous media data in a timely fashion. The critical resources in a video server are disk bandwidth, storage volume, and main memory, see figure 1.

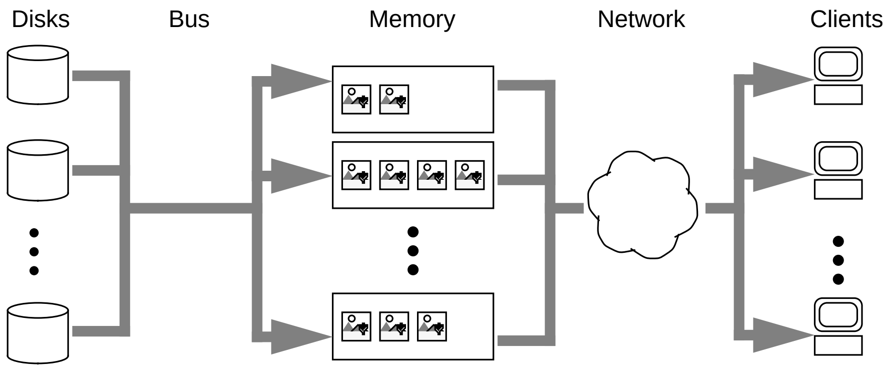

Figure 1. Components of a video server system

Given a fixed amount of these resources, a video server can only deliver a limited number of video streams simultaneously. Besides the two limited resources, namely disk bandwidth and size of main memory, there are other factors, discussed in the following, that determine how many clients can be admitted.

1.1 Quality of Service

The quality of service in a video server is determined by the timely delivery of the video information, which is encoded as a **variable bit rate stream** (VBR) of *constant* quality. In figure 2 we see the video data arrival and consumption process. If the arrival of data falls behind the consumption, the client will experience **starvation** and the quality of service will be affected negatively. If the arrival of data is ahead of the consumption, the difference between the amount of data arrived and the amount of data consumed, which is referred to as **backlog**, must be buffered until consumption.

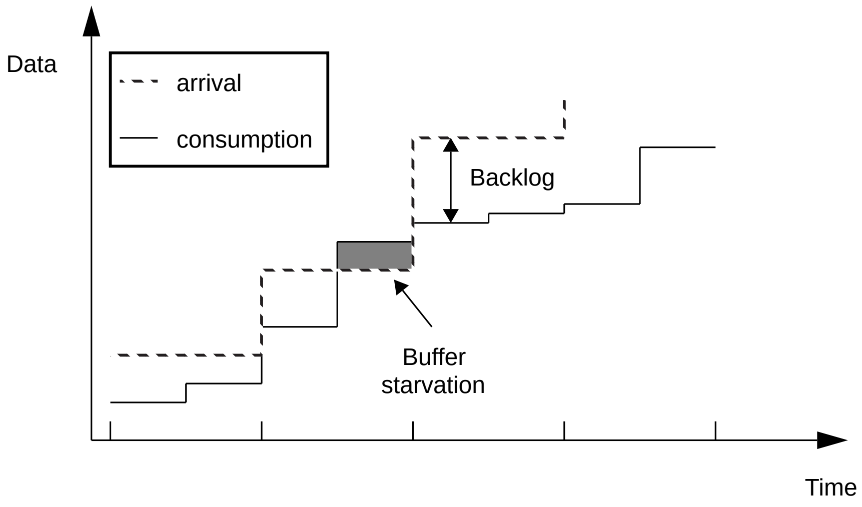

Figure 2. Video data arrival and consumption process

We will consider two types of quality of service:

- **Deterministic**, which guarantees a 100% service without any loss or delay of frames at any time. To assure deterministic service, we need a deterministic, i.e. worst case characterization of the video traffic. To provide deterministic quality of service (QOS) for VBR video, one must employ *worst-case assumptions* about the data rate of the VBR video when computing the number of streams to be admitted. We use as deterministic traffic model the so-called empirical envelope presented in [7] that provides a deterministic traffic constraint function for a given video trace. If $A_i[t, t+\tau]$ denotes the amount of video data consumed by a stream s_i in the interval $[t, t+\tau]$, an upper bound on A_i can be given by the **empirical envelope function** $\varepsilon_i(\tau)$ that is defined as:

$$\varepsilon_i(\tau) = \max_t A_i[t, t+\tau], \forall t \in [0, T_{total} - \tau] \qquad (1)$$

- **Statistical,** where starvation may occur with a certain probability: The decision whether a new session will be admitted or not depends on the starvation probability that the client is willing to accept. To assure statistical service guarantees, we use a probabilistic model for the traffic load due to the read operations.

1.2 Disk Scheduling

The scheduling of the requests to read from disk determines the order in which the requests are served and influences the disk I/O efficiency and the buffer requirement. We assume that the reads from disk are organized in rounds and starvation is avoided by *reading ahead* an amount of data that lasts in terms of playback duration at least until the end of the next round (see figure 3). Throughout the paper we assume SCAN [8] scheduling that minimizes the seek overhead between adjacent retrievals of a single round.

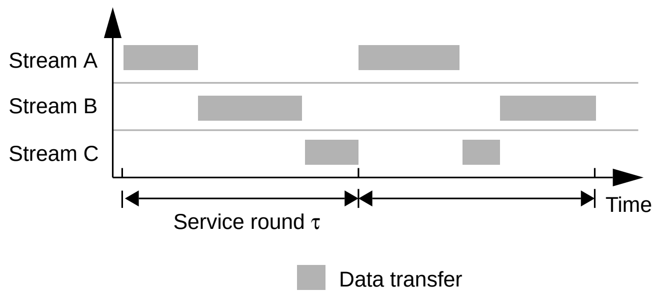

Figure 3. Sequence of service rounds with SCAN scheduling

1.3 Data Retrieval Techniques

The data retrieval technique determines the way data is read from the disk during a service round.

Using VBR as data model for a video, one can map video data onto **data blocks** stored on the disk in two ways:

- Variable size blocks of constant playout duration, referred to as **constant time length** (**CTL**) and
- Fixed size blocks of variable playout duration, referred to as **constant data length** (**CDL**) [4].

Throughout the paper we assume CDL retrieval, for a comparison between CDL and CTL see [5].

Constant data length (CDL) retrieval performs *non-periodic* retrieval of *constant* amounts of data from the disk (see figure 4, traditional CDL). To make CDL compatible with round-based disk retrieval, we introduce the restriction that the time distance between two retrieval operations must be a *multiple of a service round* τ, which will yield a sequence of **active** and **idle** rounds. During an active round, a *constant size* data block is read from the disk. Since the data must always (even in the worst case) be sufficient to supply the client with sufficient video data during the following round, the data block retrieved is of size $\varepsilon_i(\tau)$. During an idle round, no data at all is retrieved. The decision, whether a round will be active or not, can be made on-line: If there is still enough data in the buffer for the current and the next round, the current round is idle, otherwise it must be active.

We have recently proposed [5] to distinguish the

- **disk service round**, during which data for each stream are read exactly once from disk, and the
- **smoothing interval,** for which we compute the **peak consumption rate,** which is defined as $\varepsilon_i(\tau)/\tau$.

and to make the smoothing interval a *multiple* of a disk service round. The resulting retrieval algorithm is referred to as **GCDL** retrieval and works as follows:

- The disk scheduling and data retrieval still proceeds in rounds of length τ.
- However, we use a set $T = \{\tau_1, \ldots, \tau_n\}$ of smoothing intervals with $\tau_i \geq \tau$. To avoid starvation, we require that the amount of data retrieved for stream s_i from the disk during each interval τ_i must last for at least a period of τ_i. The smoothing interval duration τ_i is an *integer multiple* m_i of the disk service round duration τ (see figure 4). When $\varepsilon_i(\tau_i)$ is the deterministic upper bound on the amount of data retrieved for stream s_i during any period τ_i, we require that the amount of data retrieved during *each* of the $m_i = \tau_i/\tau$ disk service rounds is the same, namely $\varepsilon_i(\tau_i)/m_i$.

The separation of disk service round and smoothing interval reduces the peak consumption rate, which decreases with increasing τ. The possibility to choose an optimal smoothing interval for each stream, significantly reduces the buffer demand and the start-up latency, while admitting the same number of clients. A sequence of m_i consecutive disk service rounds where data for stream s_i are retrieved is also called an **active CDL round**. When during m_i consecutive disk service rounds no data are retrieved, that sequence is called an **idle CDL round** (see figure 4).

During an active CDL round a fixed amount $\varepsilon_i(\tau_i)$ of data is read, during an idle CDL round no data is read. Note that $\varepsilon_i(\tau_i)/m_i \leq \varepsilon_i(\tau)$, i.e. the amount of data that must be retrieved during an active disk service round becomes smaller, since increasing the smoothing interval τ_i reduces the peak consumption rate. As the worst-case server load $\varepsilon_i(\tau_i)/m_i$ per disk round for stream s_i becomes smaller, less disk bandwidth must be reserved for that particular stream.

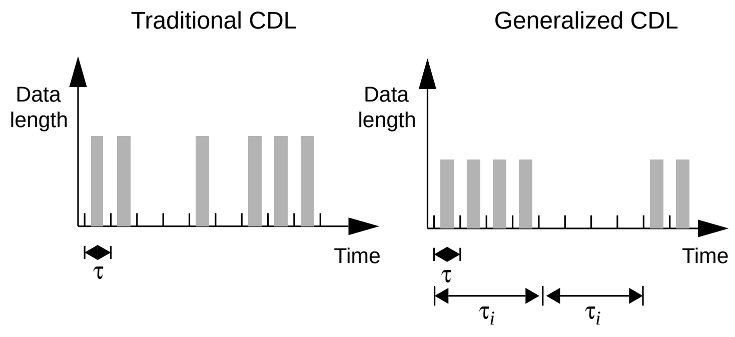

Figure 4. Traditional and generalized CDL

In the following, we will refer to the CDL retrieval where smoothing interval and disk round have the same length ($\tau = \tau_i$) as **traditional** CDL. When smoothing interval and disk round have different lengths ($\tau \neq \tau_i$), the scheme is referred to as **generalized CDL (GCDL)** retrieval. The traditional CDL can be regarded as a special case of GCDL with $m_i = 1$.

1.3.1 Admission Control Strategies

Before admitting a new client, a video server must use an admission control algorithm to check if there are enough resources for serving the additional client without degrading the quality of service (QoS) of all other clients that are already admitted. Depending on the type of QOS, we have:

- Deterministic Admission Control for GCDL Retrieval

The number of streams admitted is limited by the length of a disk service round, the available buffer space and the disk bandwidth. If we assume that the buffer space is not a scarce resource, the admission control criterion for GCDL, when SCAN scheduling is used, is given by [3]:

$$\sum_{i=1}^{n} \left\lceil \frac{\varepsilon_i(\tau_i)}{m_i} \right\rceil \cdot r_{disk}^{-1} + \sum_{i=1}^{n} \left\lceil \left\lceil \frac{\varepsilon_i(\tau_i)}{m_i} \right\rceil \cdot c_{cyl}^{-1} \right\rceil \cdot t_{track} + n \cdot (t_{track} + t_{rot}) + t_{seek} \leq \tau \qquad (2)$$

In this formula $r_{disk} = 24 \cdot 10^6$ bit/s denotes the disk bandwidth, $c_{cyl} = 4 \cdot 10^6$ bit equals the capacity of a single cylinder and $t_{track} = 1.5$ ms, $t_{rot} = 11.11$ ms and $t_{seek} = 20.0$ ms denote the track-to-track seek time, the rotational latency and the maximum seek time for a complete scan over the entire disk, and m_i denotes the number of disk service rounds within a single CDL round[1].

- Statistical Admission Control for GCDL Retrieval

A deterministic service can be assured at any time during the playback by using the *worst case* traffic characterization, given by $\varepsilon_i(\tau_i)$, for the admission control criterion. Deterministic service results an inefficient use of the server's resources, such as disk bandwidth and buffer space: We observed [2] during the majority of all CDL rounds, a client consumes much less video data than the envelope function $\varepsilon_i(\tau_i)$ (worst-case consumption) suggests, i.e. in most rounds the server allocates much more bandwidth than necessary. Therefore, a high number of CDL rounds will be *idle*[2].

2. GCDLB

Our previous work [5,1,2] demonstrated that both, GCTL and GCDL retrieval, have advantages and disadvantages (see figure 9)

- GCDL has lower start-up latencies than GCTL and admits more streams than GCTL under statistical admission control.

[1] The actual values given are the ones for the Micropolis A4110 AV disk and are used in simulations for which we present the results later.
[2] See Eq. (10) for a definition of the probability p_i of an active round and table 3 for the actual values for p_i.

- GCTL requires significantly less buffer per stream than GCDL since the data retrieved during a GCTL round will be consumed *entirely* during the next GCTL round.

In this paper we will present a modified GCDL retrieval scheme, called **GCDLb**, that tries to combine the advantages of GCTL and GCDL without their disadvantages.

2.1 Start-up Latency and Buffer Requirement

Besides the number of clients admitted, the two most important performance measures for comparing the different retrieval techniques are start-up latency and also buffer requirement per stream**.** The **start-up latency** is defined as the time elapsed between user interaction and feedback by the server. The start-up latency of a video server is given by the time that passes by from the reception of a playback request until the first frame is submitted to network. Note that the delay introduced by the network and by buffering at the client site, e.g. the time to synchronize several streams, is not being considered. The server has to wait until the beginning of the next disk service round before the request can be processed. In the worst case, this takes a time τ. Second, the server must wait another period $\hat{t}_i$ until enough data has arrived in the buffer to guarantee that buffer starvation is avoided during the playback of the video. Consequently, the total start-up latency is given by $\tau_i + \hat{t}_i$.

We assume that the service under consideration allows for full VCR functionality, which implies that a video can be started at an *arbitrary position* T_s. Note that T_s influences the arrival of frames because the retrieval of data blocks always depends on the data retrieved in the past. Let $a_i^{late}(T_s, t)$ be the **cumulative arrival function** for the latest possible arrival of data blocks, i.e. if a GCDL round starts at time t, data arrive at times $t + \tau, t + 2 \cdot \tau, \ldots, t + m_i$. The difference between $a_i^{late}(T_s, t)$ and the **cumulative consumption function** $c_i(T_s, t)$ for a delay $\hat{t}_i$ between the two is called the **backlog function** $b_i(T_s, t)$:

$$b_i(T_s, t) \;=\; a_i^{late}(T_s, t) - c_i(T_s, t - \hat{t}_i) \tag{3}$$

It is obvious that $b_i(T_s, t)$ must be equal or greater than 0 at all times in order to avoid buffer starvation (see figure 2). The determination of the optimal start-up latency can then be described as a linear minimization problem:

$$\hat{t}_i \rightarrow min \qquad \text{for } \hat{t}_i \in [\tau, \tau_i] \tag{4}$$

$$b_i(T_s, t) \geq 0 \qquad \text{for } t \in [T_s, T_{total}] \tag{5}$$

The **deterministic buffer requirement** is closely related to the playout delay $\hat{t}_i$. If the play-out of stream s_i is delayed by $\hat{t}_i$, the minimum buffer level will be equal to 0. The worst-case maximum buffer level $B_i^{max}(T_s)$ is then given by expression (6) where $a_i^{late}(T_s, t)$ states the cumulative arrival function for the earliest possible arrival of data blocks:

$$B_i^{max}(T_s) \;=\; \max_{t \in [T_s, T_{total}]} \left(a_i^{early}(T_s, t) - c_i(T_s, t)\right) \tag{6}$$

2.2 Static GCDLb

When we did the step from traditional to generalized CDL, we distinguished disk service rounds from CDL rounds and introduced the parameter m_i to describe the relationship between the two. We assumed that during *all* disk service rounds within a GCDL round the same amount of data were retrieved for best performance. This implies that the active/idle-mechanism is applied to *complete* GCDL rounds. As we saw in the analysis of GCDL [2], the resulting amount of retrieved data is quite large compared to the average demand, which leads to small start-up latencies at one hand, and the need for large buffer at the other hand.

The basic idea behind the new GCDLb scheme is to schedule only as many active disk service rounds during a GCDL round as it requires to supply the client with frames for a time period of τ_i. Since the new retrieval strategy is derived from GCDL, we call it GCDLb. In figure 4 we see that GCDLb fills up the GCDL rounds from the beginning with read requests, i.e. active disk service rounds. Unlike to GCDL, it is much rarer for GCDLb that there is a GCDL round with *no read request at all*.

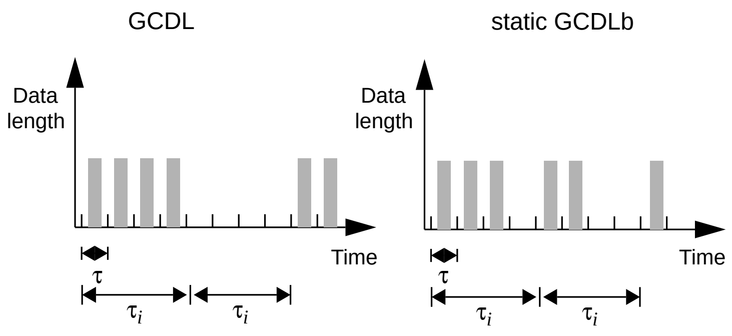

Figure 5. GCDL and static GCDLb retrieval

Determining the number of active disk rounds during each GCDL round is straightforward and will be done on-line during the playout of the video data. Let $n_i(T_s, t)$ denote the **number of active rounds during a GCDLb round** that starts at time t for a playback starting at T_s. Furthermore, let $A_i[t, t+\tau_i]$ denote the client's demand and $b_i(T_s, t)$ denote the buffer level at the beginning of that particular round. Since the size of a single data block equals $\varepsilon_i(\tau_i)/m_i$, $n_i(T_s, t)$ can be computed as follows:

$$n_i(T_s, t) = \left\lceil \frac{(A_i[t, t+\tau_i] - b_i(T_s, t)) \cdot m_i}{\varepsilon_i(\tau_i)} \right\rceil \tag{7}$$

In order to reduce the start-up latency, we assume that always the *first* $n_i(T_s, t)$ disk service rounds of a GCDLb round are set to active and the following $m_i - n_i(T_s, t)$ ones are idle, which is referred to as **static GCDLb**. It can easily be shown that the start-up latency of static GCDLb is the same as for GCDL.

Table 1 compares GCDL and static GCDLb in terms of buffer requirement. We see that static GCDLb outperforms GCDL for any combination of τ and τ_i because it has got the same start-up latency but also a data retrieval that adapts better to the consumption rate. GCDLb also outperforms GCTL: compared to GCTL, GCDLb has got a better start-up latency and a buffer requirement that differs only slightly from GCTL. Note that traditional GCDLb ($m_i = 1$) is identical to traditional GCDL. Since a worst-case round is the same for all schemes under consideration, the number of admitted streams is the same, too.

τ	τ_i	Start-up Latency			Buffer requirement per stream		
		GCTL	static GCDLb	GCDL	GCTL	static GCDLb	GCDL
1	1	2.0	2.0	2.0	673,470	1,070,602	1,070,602
1	2	2.7	2.2	2.2	901,445	1,126,456	1,414,468
1	3	3.2	2.2	2.2	1,069,069	1,126,939	1,750,577
1	4	3.4	2.2	2.2	1,142,250	1,169,607	2,069,847
1	6	3.8	2.2	2.2	1,259,659	1,344,177	2,690,487

Table 1: Start-up latency and buffer requirement of the GCTL and GCDL retrieval schemes for the 'MTV' trace.

2.3 Dynamic GCDLb

We have seen that static GCDLb fills a GCDL round from the beginning with active disk service rounds until sufficient data have been read during that particular GCDL round. This behavior can be regarded as **static** because it does not consider the scheduling of other parallel streams.

The dynamic GCDLb algorithm combines *both off-line and on-line scheduling*:

- It uses off-line scheduling only to determine *how many* disk service rounds must be active during a GCDLb round.
- The question which disk service rounds are set to active is decided *on-line* for all streams by a central scheduler. This is possible since all necessary information is known at the beginning of a GCDLb round. The scheduler distributes the active rounds *evenly* over the GCDL round in order to equalize the disk bandwidth required during each disk round.

Since this retrieval strategy reacts *on-line* to external influences, it will be referred to as **dynamic GCDLb**.

Figure 6 illustrates dynamic GCDLb in comparison to static GCDLb. The requests of a stream are always placed in slots where the disk load is minimal. If all requests are placed for one stream, the scheduler moves on to the next stream.

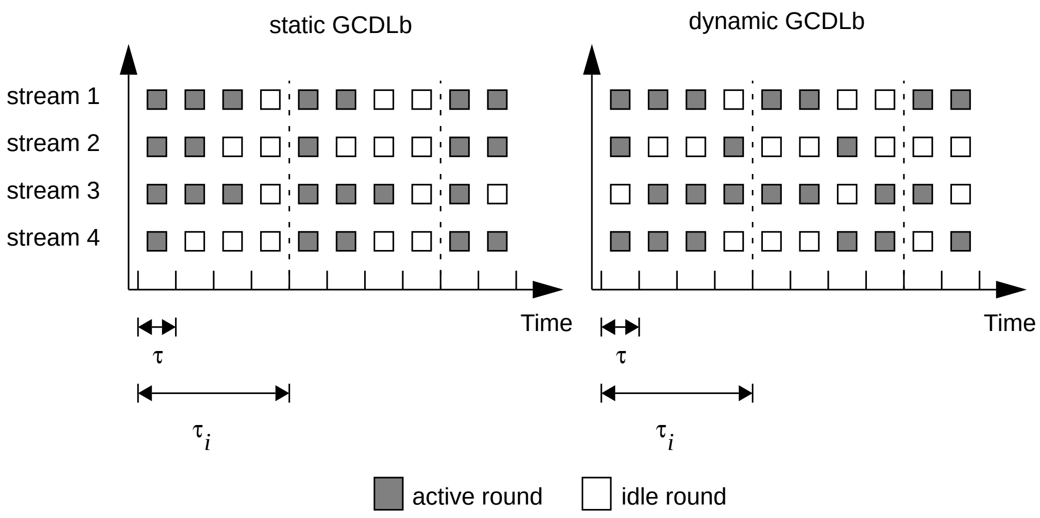

Figure 6. Scheduling of active rounds for static and dynamic GCDLb

The scheduling algorithm of dynamic GCDLb can be generalized for varying values of m_i, arbitrary starting times of GCDLb rounds, and heterogeneous streams with different retrieval block sizes:

- If m_i is not the same for all streams, the scheduling can not be done for all streams at the same time but only for those that start a disk service round at the particular time. All other overlapping GCDLb rounds that have already started must be regarded as static whereas all those that start later are unknown. The algorithm still finds a good distribution under these conditions but not an optimal one because there is always some unknown information about future server loads.
- The results of the algorithm do not depend on the order in which streams are processed, i.e. it does not matter whether we schedule active rounds for stream s_1 before or after s_2 because the resulting server load is the same in both cases. If we drop the assumption that GCDLb rounds are synchronized, we only change the order in which streams are processed, which is now given by the starting times of the GCDLb rounds.
- If we move from the homogeneous to the heterogeneous case, we can no longer consider just active and idle rounds since the size of data blocks differs for each video. The scheduling algorithm can then be regarded as a procedure that solves the knapsack problem for a set of disk service rounds, which is known from operations research [6] and can be solved by various exact and heuristic algorithms.

We must examine the consequences of the extensions made for dynamic GCDLb in terms of our performance measures. Since the position of active rounds is not fixed any more, we must assume that video data can arrive anytime during a GCDLb round.

- Start-up latency: It is possible that all necessary data for a GCDL round are retrieved in one active round *at the end* of that particular GCDLb round. In the worst case, a client has to wait one round τ until the request can be processed and τ_i until the first frames can be played. Therefore, the maximum start-up latency is given by $\tau + \tau_i$. It is also possible that this data block is retrieved at the very beginning of a GCDLb round, which is equivalent to the way static GCDLb retrieves data.
- Buffer requirement: The maximum buffer requirement B_i^{max} can then be computed the same way as we did for static GCDLb. However, the higher start-up latencies of dynamic GCDLb will lead to a higher buffer requirement as compared to static GCDLb.

$$B_i^{max} = \max_{t \in [0, \tau_i], t_s \in [0, T_{total} - \tau_i]} \left(b_i^{max}(t_s) + a_i^{early}(t_s, t) - c_i(t_s - \hat{t}_i, t - \hat{t}_i) \right) \quad (8)$$

In this formula, $b_i^{max}(t_s)$ denotes the maximum buffer level at the beginning of a GCDLb round if the playback of the video starts at position t_s . The cumulative arrival function in the worst case is given by $a_i^{early}(t_s, t)$ and $c_i(t_s, t)$ denotes the consumption function.

2.4 Performance comparison of the data retrieval schemes for homogeneous case

2.4.1 Start-up latency and maximum buffer requirement

A comparison of the different CDL-based schemes in terms of start-up latency and maximum buffer requirement is presented in table 2.

	τ	τ_i	Start-up Latency			Buffer requirement per stream		
			dynamic CDLb	static CDLb	CDL	dynamic CDLb	static CDLb	CDL
generalized	1	1	2.0	2.0	2.0	1,070,602	1,070,602	1,070,602
	1	2	3.0	2.2	2.2	1,348,748	1,126,456	1,414,468
	1	3	4.0	2.2	2.2	1,691,894	1,126,939	1,750,577
	1	4	5.0	2.2	2.2	1,988,577	1,169,607	2,069,847
	1	6	7.0	2.2	2.2	2,586,265	1,344,177	2,690,487
traditional	1	1	2.0	2.0	2.0	1,070,602	1,070,602	1,070,602
	2	2	4.0	4.0	4.0	1,992,015	1,992,015	1,992,015
	3	3	6.0	6.0	6.0	2,955,418	2,955,418	2,955,418
	4	4	8.0	8.0	8.0	3,660,027	3,660,027	3,660,027

Table 2: Start-up latency and buffer requirement of the GCDL retrieval schemes for the ‘MTV’ trace.

Dynamic GCDLb does not offer any advantages in the case of deterministic admission control when compared to static GCDLb:

- Its start-up latency is the highest of all presented schemes and

- the buffer requirement is only a little lower than the one we get for GCDL.

For the *deterministic* admission control, we get the *same maximum number* of admitted streams because the admission control criterion (see Eq (2)) is the same for all schemes. These results are not surprising because for deterministic admission control, dynamic GCDLb cannot handle the worst case data load any better than GCTL, GCDL, or static GCDLb. In this worst case, m_i subsequent active disk service rounds are necessary to retrieve all demanded frames. Since there is no idle round, no better sequence can be found.

2.4.2 Number of streams admitted under statistical admission control

The comparison between the GCDL schemes looks different for the statistical service. To calculate how many streams can be admitted under statistical admission control for dynamic GCDLb, we need to calculate the histograms for *the number of active rounds* during a GCDLb round. This number depends on the rest $\tilde{r}$ of data in the buffer at the end of a GCDL round, whose size can only lie somewhere in the interval $[0, \varepsilon_i(\tau_i)/m_i)$. If $\tilde{r}$ is greater than $A_i[t, t+\tau_i] \bmod (\varepsilon_i(\tau_i)/m_i)$, it takes $\lfloor A_i[t, t+\tau_i]/(\varepsilon_i(\tau_i)/m_i) \rfloor$ active rounds; if $\tilde{r}$ is smaller, it takes one additional round, i.e. $\lceil A_i[t, t+\tau_i]/(\varepsilon_i(\tau_i)/m_i) \rceil$ rounds. We assume that $\tilde{r}$ is uniformly distributed over the interval, which makes it simple to compute the probabilities for these two events.

The **maximum number of active disk service rounds during a GCDL round** that can be served is given by the following expression:

$$\left\lfloor \frac{\tau - t_{seek}}{\sum_{i=1}^{n} \left\lceil \frac{\varepsilon_i(\tau_i)}{r_{disk} \cdot m_i} \right\rceil + \sum_{i=1}^{n} \left\lceil \frac{\varepsilon_i(\tau_i)}{c_{cyl} \cdot m_i} \right\rceil \cdot t_{track} + n \cdot (t_{track} + t_{rot})} \cdot m_i \right\rfloor \tag{9}$$

The overload probability can then be calculated by histogram convolution as we demonstrated in [2]. Figure 7 compares the number streams admitted by the different retrieval schemes for the ‘MTV’ trace. As one can see, dynamic GCDLb performs better than all other schemes, especially for a high disk transfer rate. Note that the scheduling of active rounds for static GCDLb is a subset of all possible solutions that the scheduling algorithm for dynamic GCDLb produces.

Surprisingly, dynamic GCDLb yields a higher number of streams for low overload probabilities but GCDL/static GCDLb get better for overload probabilities greater than about 50%. This does not affect the practical use of the presented schemes because no application is known that is able to handle an overload in every second service round. Nevertheless, it is interesting since we expected dynamic GCDLb to perform better than a static scheme for any number of streams.

The explanation can be found when we remember that we considered only the overload event itself but not the amount of data that could not be read. Dynamic GCDLb tries to distribute active blocks evenly over a GCDLb round but it cannot prevent overloads at all. Because of this even distribution, many overloads can happen during a playback but each one of them causes only the loss of little data whereas for GCDL and static GCDLb a few overloads occur that cause the loss of entire data blocks

Before we further compare the different schemes we need to introduce a few definitions:

- If mbr_i denotes the **average bit rate** of stream s_i, the **probability** p_i **of an active round** for stream s_i is given by

$$p_i = \frac{mbr_i \cdot \tau_i}{\varepsilon_i(\tau_i)} \tag{10}$$

- If all clients request the same video (homogenous case), the maximum number of parallel requests that can be served under GCDL retrieval during one disk service round is equivalent to the maximum number of simultaneous streams in the deterministic case N_{det}, and can be stated as follows (see Eq (2)):

$$N_{det} = \left\lfloor \frac{\tau - t_{seek}}{\left\lceil \frac{\varepsilon_i(\tau_i)}{r_{disk} \cdot m_i} \right\rceil + \left\lceil \frac{\varepsilon_i(\tau_i)}{c_{cyl} \cdot m_i} \right\rceil \cdot t_{track} + t_{track} + t_{rot}} \right\rfloor \tag{11}$$

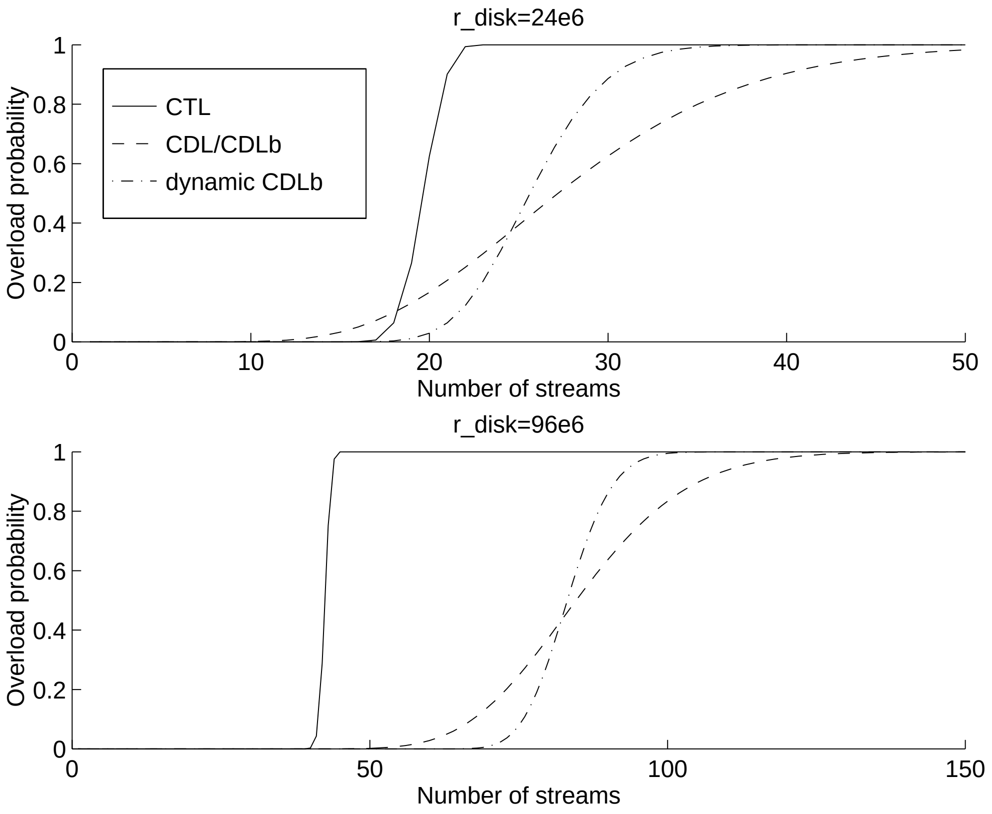

Figure 7. Overload probabilities for the different retrieval schemes for $r_{disk} = 24 \cdot 10^6$ and $r_{disk} = 96 \cdot 10^6$

- $N_{10^{-4}}$, denotes the number of streams that can be admitted without exceeding an overload probability of 10^{-4}. An overload probability of 10^{-4} means that on the average every 10^4-th disk service round an overload occurs. For $\tau = 1$ s, this is equivalent to an overload event every 2.8 hours. To compare the deterministic and the statistical admission control, we use the gain $G_{10^{-4}}$, which is defined as the improvement in terms of the number of streams admitted in the statistical case when compared to the deterministic case:

$$G_{10^{-4}} = \frac{N_{10^{-4}} - N_{det}}{N_{det}} \qquad (12)$$

We calculated the maximum number of streams admitted for 12 different videos and for two disk transfer rates. To obtain these results we used video traces made available by the University of Würzburg, Germany. Our results are summarized in tables 3 and 4. Note that all computations were done for a special case where all clients request *the same video* and retrieve it during *synchronized GCDLb rounds*, i.e. the rounds have the same length and start at the same time.

Video	Det. admiss.	Statistical admission					$\frac{\varepsilon_i(\tau_i)}{\tau_i}$ [Mbit/sec]	p_i
		GCDL/ GCDLb		dyn. GCDLb		$\frac{G^{dyn}_{10^{-4}}}{G^{static}_{10^{-4}}}$		
	N_{det}	$N_{10^{-4}}$	$G_{10^{-4}}$	$N_{10^{-4}}$	$G_{10^{-4}}$			
Lambs	19	36	0.89	51	1.68	1.89	0.85	0.24
StarWars	17	28	0.65	40	1.36	2.09	0.97	0.27
Terminator	25	29	0.16	34	0.36	2.25	0.57	0.54
Movie2	16	20	0.25	27	0.69	2.76	1.04	0.39
News	10	15	0.50	27	1.70	3.40	1.89	0.23
MrBean	10	13	0.30	21	1.10	3.67	1.75	0.28
Simpsons	13	15	0.15	22	0.69	4.60	1.33	0.40
MTV2	8	11	0.38	18	1.25	3.29	2.47	0.22
Asterix	11	12	0.09	17	0.55	6.11	1.61	0.39
MTV	7	8	0.14	14	1.00	7.14	2.53	0.27
Fuss	11	11	0.00	15	0.36	-	1.66	0.46
Race	9	9	0.00	12	0.33	-	1.98	0.44

Table 3: Number of streams admitted for the GCDL data retrieval schemes

The videos are listed in increasing order of their average bit-rate, $r_{disk} = 24 \cdot 10^6$.

Video	Det. admiss.	Statistical admission					$\frac{\varepsilon_i(\tau_i)}{\tau_i}$ [Mbit/sec]	p_i
		GCDL/ GCDlb		dynamic GCDLb		$\frac{G^{dyn}_{10^{-4}}}{G^{static}_{10^{-4}}}$		
	N_{det}	$N_{10^{-4}}$	$G_{10^{-4}}$	$N_{10^{-4}}$	$G_{10^{-4}}$			
Lambs	41	99	1.41	127	2.10	1.49	0.85	0.24
StarWars	39	84	1.15	107	1.74	1.51	0.97	0.27
Terminator	48	62	0.77	71	1.03	1.34	0.57	0.54
Movie2	38	60	0.58	74	0.95	1.64	1.04	0.39
News	28	64	1.29	90	2.21	1.71	1.89	0.23
MrBean	29	56	0.93	75	1.59	1.71	1.75	0.28
Simpsons	34	52	0.53	65	0.91	1.72	1.33	0.40
MTV2	23	49	1.31	68	1.96	1.50	2.47	0.22
Asterix	30	45	0.50	56	0.87	1.74	1.61	0.39
MTV	23	42	0.83	59	1.57	1.89	2.53	0.27
Fuss	30	40	0.33	49	0.63	1.91	1.66	0.46
Race	27	36	0.33	45	0.67	2.03	1.98	0.44

Table 4: Number of streams admitted for different GCDL data retrieval schemes

The disk transfer rate was changed to $r_{disk} = 96 \cdot 10^6$, compared to table 3.

We see that the gains vary widely for the different streams, which can be explained by the different probabilities p_i for an active round and the sizes of the retrieved data blocks $\varepsilon_i(\tau_i)/m_i$ (see figure 8). The gains are highest for low values of p_i that allow for a better scheduling of active rounds, because more bandwidth is wasted.

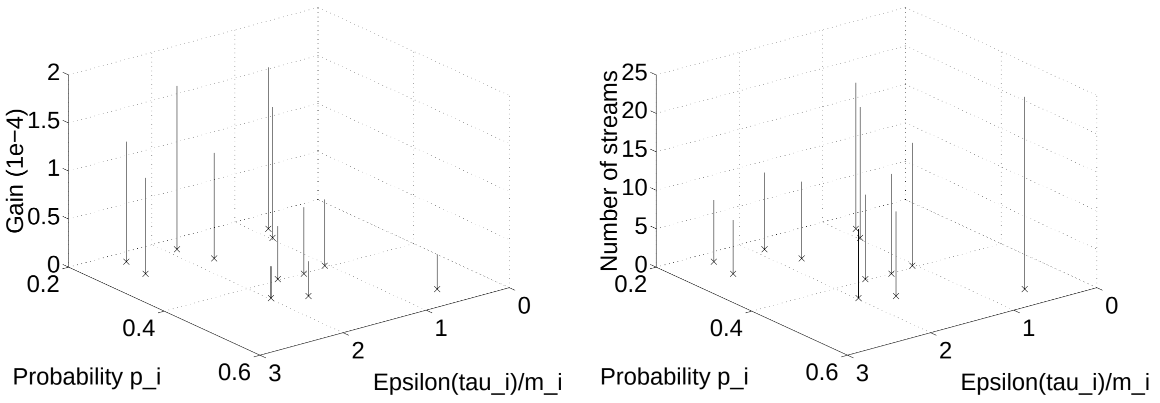

Figure 8. Gain and N_{det} as function of $\varepsilon_i(\tau_i)/m_i$, and p_i for statistical dynamic GCDLb. All computations were done for $\tau = 1$ s, $\tau_i = 4$ s, and $r_{disk} = 24 \cdot 10^6$ bit/s. Values for the gain are given for an overload probability of 10^{-4}.

Figure 10 compares for statistical admission control the number of streams admitted for the different retrieval schemes. Since the GCDL schemes retrieve a whole video with fewer read operations than GCTL they occur less overhead and can, in particular for a high disk I/O rate admit more streams than GCTL.

Figure 9 summarizes the effect of different smoothing intervals τ_i on the start-up latency and the buffer requirement, These results are the same for each scheme for both, deterministic and statistical admission control.

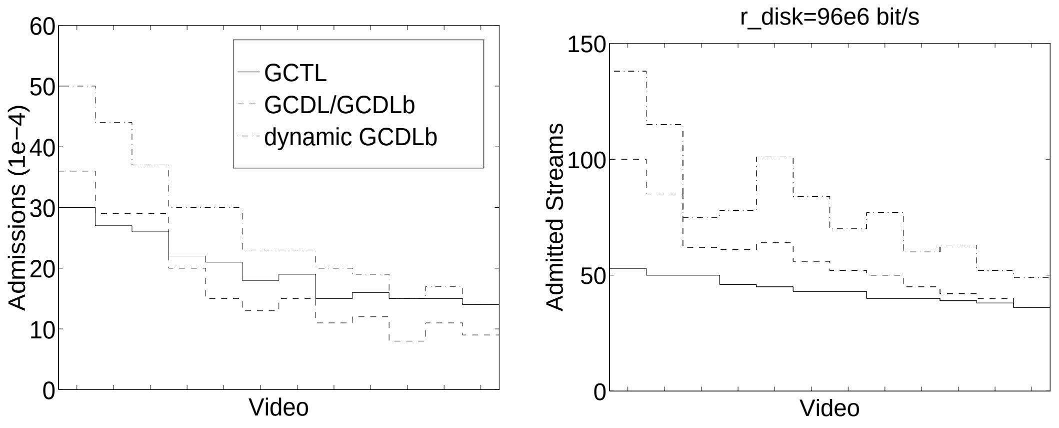

Figure 10. Statistical admission control for GCTL, GCDL, static and dynamic GCDLb All computations were done for the considered videos with or $\tau = 1$ s, $\tau = 4$ s, $r_{disk} = 24 \cdot 10^6$ and $r_{disk} = 96 \cdot 10^6$ bit/s at an overload probability of 10^{-4}.

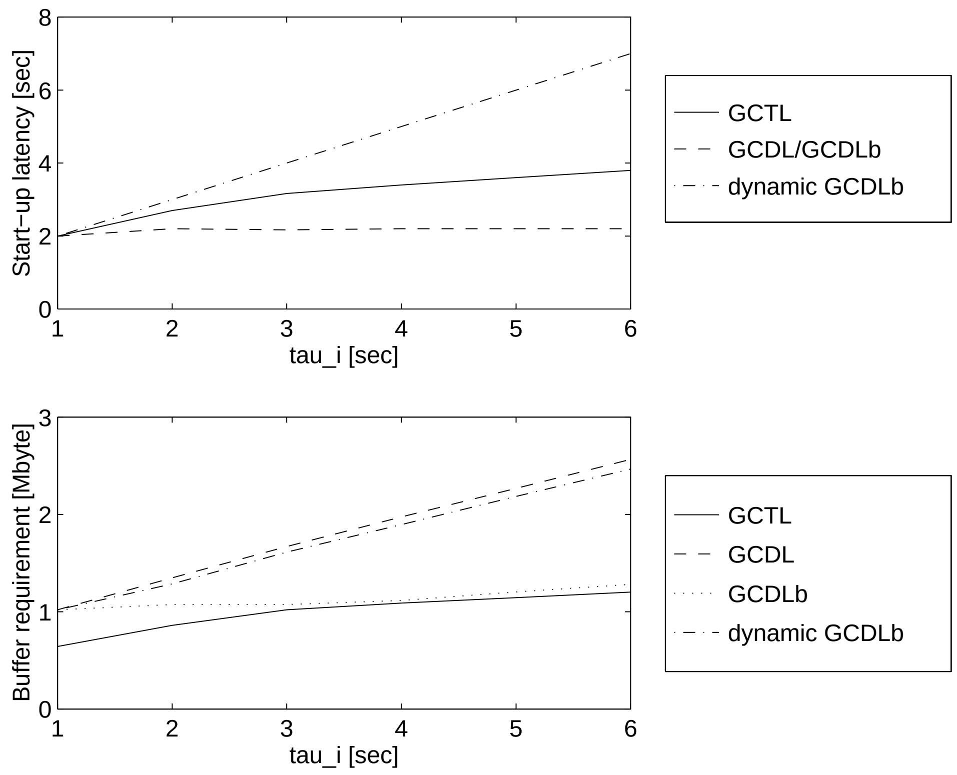

Figure 9. Start-up latency and buffer requirement for GCTL, GCDL, and GCDLb
All computations were done for the 'MTV' trace with or $\tau = 1$ s.

2.5 Conclusion

We have introduced a two new GCDL retrieval schemes and evaluated their performance improvement in terms of start-up latency, buffer requirement, and additional streams admitted. We saw that static GCDLb combines the best characteristics of GCTL (low buffer requirement) and GDCL (low start-up latency and a high number of admitted streams).

The second scheme called dynamic GCDLb can, under statistical admission control, further increase the number of streams admitted, however at the price of a higher start-up latency and a much larger buffer requirement than GCDL and static GCDLb.

The overall cost of a video server is determined by the cost for the disk storage and cost for the buffer memory. Taking into account the high cost of buffer memory compared to disk storage, static GCDLb with statistical admission control is the most suitable retrieval strategy for building a cost-efficient video server.

References

[1] E. W. Biersack and F. Thiesse. Statistical admission control for video servers with variable bit rate streams and constant time length retrieval. In *Euromicro 96*, Prague, Sept. 1996.

[2] E. W. Biersack and F. Thiesse. Statistical admission control in video servers with constant data length retrieval of VBR streams. In *Third International Conference on Multimedia Modeling*, Toulouse, France, Nov. 1996.

[3] E. W. Biersack, F. Thiesse, and C. Bernhardt. Constant data length retrieval for video servers with variable bit rate

streams. In *Proc. of the IEEE Conf. on Multimedia Systems*, pages 151–155, Hiroshima, Japan, June 1996.

[4] E. Chang and A. Zakhor. Admission control and data placement for VBR video servers. In *Proceedings of the 1st International Conference on Image Processing*, Austin, Texas, November 1994.

[5] J. Dengler, C. Bernhardt, and E. W. Biersack. Deterministic admission control strategies in video servers with variable bit rate streams. In B. Butscher, E. Moeller, and H. Pusch, editors, *Interactive Distributed Multimedia Systems and Services, European Workshop IDMS'96, Berlin, Germany*, volume 1045 of *LNCS*, pages 245–264. Springer Verlag, Heidelberg, Germany, Mar. 1996.

[6] W. Domschke and A. Drexl. *Operations Research*. Springer-Verlag, Berlin, Heidelberg, New York, Tokio, second edition, 1991.

[7] E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang. Fundamental limits and trade-offs of providing deterministic guarantees to VBR video traffic. In *Proceedings Sigmetrics '95 / Performance '95*, volume 23 of *Performance Evaluation Review*, pages 98–107, Ottawa, Canada, May 15-19 1995.

[8] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1995.