MEMORY MANAGEMENT FOR LARGE SCALE DATA STREAM RECORDERS [*]

Kun Fu and Roger Zimmermann
Integrated Media System Center
University of Southern California
Los Angeles, California 90089

Keywords: Memory management, real time, large-scale, continuous media, data streams, recording.

Abstract: Presently, digital continuous media (CM) are well established as an integral part of many applications. In recent years, a considerable amount of research has focused on the efficient retrieval of such media. Scant attention has been paid to servers that can record such streams in real time. However, more and more devices produce direct digital output streams. Hence, the need arises to capture and store these streams with an efficient data stream recorder that can handle both recording and playback of many streams simultaneously and provide a central repository for all data.

In this report we investigate memory management in the context of large scale data stream recorders. We are especially interested in finding the minimal buffer space needed that still provides adequate resources with varying workloads. We show that computing the minimal memory is an $\mathcal{NP}$-complete problem and will require further study to discover efficient heuristics.

1 INTRODUCTION

Digital continuous media (CM) are an integral part of many new applications. Two of the main characteristics of such media are that (1) they require real time storage and retrieval, and (2) they require high bandwidths and space. Over the last decade, a considerable amount of research has focused on the efficient retrieval of such media for many concurrent users (Shahabi et al., 2002). Algorithms to optimize such fundamental issues as data placement, disk scheduling, admission control, transmission smoothing, etc., have been reported in the literature.

Almost without exception these prior research efforts assumed that the CM streams were readily available as files and could be loaded onto the servers offline without the real time constraints that the complementary stream retrieval required. This is certainly a reasonable assumption for many applications where the multimedia streams are produced offline (e.g., movies, commercials, educational lectures, etc.). However, the current technological trends are such that more and more sensor devices (e.g., cameras) can directly produce digital data streams. Furthermore, many of these new devices are network-capable either via wired (SDI, Firewire) or wireless (Bluetooth, IEEE 802.11x) connections. Hence, the need arises to capture and store these streams with an efficient data stream recorder that can handle both recording and playback of many streams simultaneously and provide a central data repository.

The applications for such a recorder start at the low end with small, personal systems. For example, the "digital hub" in the living room envisioned by several companies will in the future go beyond recording and playing back a single stream as is currently done by TiVo and ReplayTV units (Wallich, 2002). Multiple camcorders, receivers, televisions, and audio amplifiers will all connect to the digital hub to either store or retrieve data streams. An example for this convergence is the next generation of the DVD specification that also calls for network access of DVD players (Smith, 2003). At the higher end, movie production will move to digital cameras and storage devices. For example, George Lucas' "Star Wars: Episode II Attack of the Clones" was shot entirely with high-definition digital cameras (Huffstutter and Healey, 2002). Additionally, there are many sensor networks that produce continuous streams of

[*]This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and IIS-0082826, and an unrestricted cash gift from the Lord Foundation.

data. For example, NASA continuously receives data from space probes. Earthquake and weather sensors produce data streams as do web sites and telephone systems.

In this paper we investigate issues related to memory management that need to be addressed for large scale data stream recorders (Zimmermann et al., 2003). After introducing some of the related work in Section 2 we present a memory management model in Section 3. We formalize the model and compute its complexity in Section 4. We prove that because of a combination of a large number of system parameters and user service requirements the problem is exponentially hard. Conclusions and future work are contained in Section 5.

2 RELATED WORK

Managing the available main memory efficiently is a crucial aspect of any multimedia streaming system. A number of studies have investigated buffer and cache management. These techniques can be classified into three groups: (1) *server buffer management* (Makaroff and Ng, 1995; Shi and Ghandeharizadeh, 1997; Tsai and Lee, 1998; Tsai and Lee, 1999; Lee et al., 2001), (2) *network/proxy cache management* (Sen et al., 1999; Ramesh et al., 2001; Chae et al., 2002; Cui and Nahrstedt, 2003) and (3) *client buffer management* (Shahabi and Alshayeji, 2000; Waldvogel et al., 2003). Figure 1 illustrates where memory resources are located in a distributed environment.

In this report we aim to optimize the usage of server buffers in a large scale data stream recording system. This focus falls naturally into the first category classified above. To the best of our knowledge, no prior work has investigated this issue in the context of the design of a large scale, unified architecture, which considers both retrieving and recording streams simultaneously.

3 MEMORY MANAGEMENT OVERVIEW

A streaming media system requires main memory to temporarily hold data items while they are transferred between the network and the permanent disk storage. For efficiency reasons, network *packets* are generally much smaller than disk *blocks*. The assembly of incoming packets into data blocks and conversely the partitioning of blocks into outgoing packets requires main memory buffers. A widely used solution in servers is double buffering. For example, one buffer

Table 1: Parameters for a current high performance commercial disk drive.

Model	ST336752LC
Series	Cheetah X15
Manufacturer	Seagate Technology, LLC
Capacity C	37 GB
Transfer rate R_D	See Figure 2
Spindle speed	15,000 rpm
Avg. rotational latency	2 msec
Worst case seek time	$\approx$ 7 msec
Number of Zones Z	9

is filled with a data block that is coming from a disk drive while the content of the second buffer is emptied (i.e., streamed out) over the network. Once the buffers are full/empty, their roles are reversed.

With a stream recorder, double buffering is still the minimum that is required. With additional buffers available, incoming data can be held in memory longer and the deadline by which a data block must be written to disk can be extended. This can reduce disk contention and hence the probability of missed deadlines (Aref et al., 1997). However, in our investigation we are foremost interested in the minimal amount of memory that is necessary for a given workload and service level. Hence, we assume a double buffering scheme as the basis for our analysis. In a large scale stream recorder the number of streams to be retrieved versus the number to be recorded may vary significantly over time. Furthermore, the write performance of a disk is usually significantly less than its read bandwidth (see Figure 2b). Hence, these factors need to be considered and incorporated into the memory model.

When designing an efficient memory buffer management module for a data stream recorder, one can classify the interesting problems into two categories: (1) *resource configuration* and (2) *performance optimization*.

In the resource configuration category, a representative class of problems are: *What is the minimum memory or buffer size that is needed to satisfy certain playback and recording service requirements?* These requirements depend on the higher level QoS requirements imposed by the end user or application environment.

In the performance optimization category, a representative class of problems are: *Given certain amount of memory or buffer, how to maximize our system performance in terms of certain performance metrics?* Two typical performance metrics are as follows:

i Maximize the total number of supportable streams.

ii Maximize the disk I/O parallelism, i.e., minimize the total number of parallel disk I/Os.

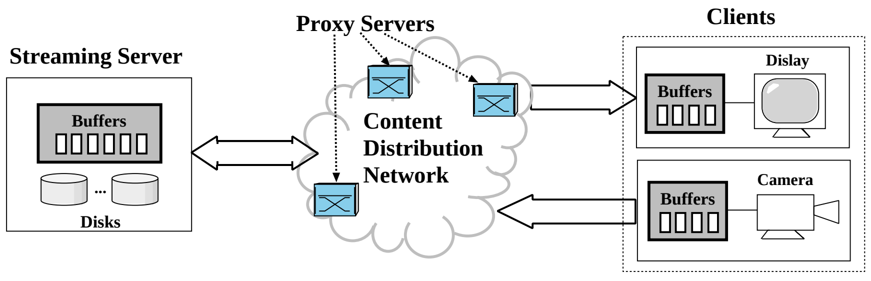

Figure 1: Buffer distribution in a traditional streaming system.

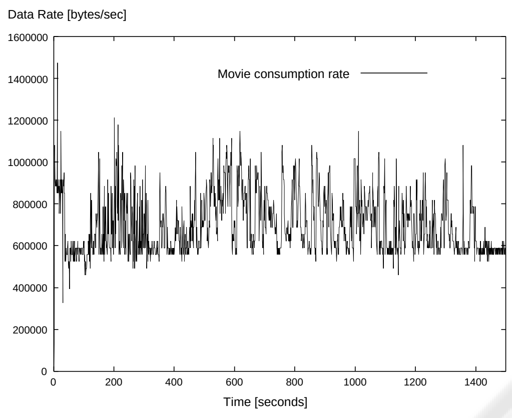

Figure 2a: The consumption rate of a movie encoded with a VBR MPEG-2 algorithm (“Twister”).

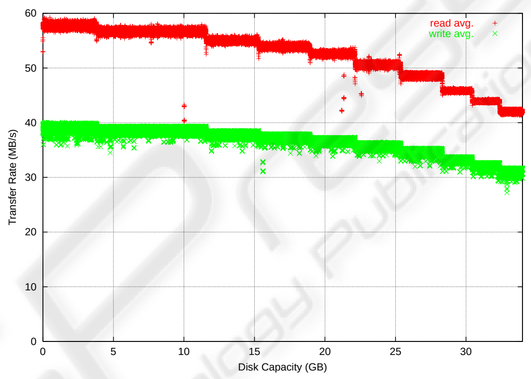

Figure 2b: Maximum read and write rate in different areas (also called *zones*) of the disk. The transfer rate varies in different zones.i The write bandwidth is up to 30% less than the read bandwidth.

Figure 2: Variable bit rate (VBR) movie characteristics and Disk characteristics of a high performance disk drive (Seagate Cheetah X15, see Table 1).

We focus on the resource configuration problem in this report, since it is a prerequisite to optimizing performance.

4 MINIMIZING THE SERVER BUFFER SIZE

Informally, we are investigating the following question: *What is the minimum memory buffer size* S^{buf}_{min} *that is needed to satisfy a set of given streaming and recording service requirements?*

In other words, the minimum buffer size must satisfy the *maximum* buffer resource requirement under the given service requirements. We term this problem the *Minimum Server Buffer* or MSB. We illustrate our discussion with the example design of a large scale recording system called HYDRA, a High-performance Data Recording Architecture (Zimmermann et al., 2003). Figure 3 shows the overall architecture of HYDRA. The design is based on random data placement and deadline driven disk scheduling techniques to provide high performance. As a result, statistical rather than deterministic service guarantees are provided.

The MSB problem is challenging because the media server design is expected to:

i support multiple simultaneous streams with different bandwidths and variable bit rates (VBR) (Figure 2a illustrates the variability of a sample MPEG-2 movie). Note that different recording devices might also generate streams with variable bandwidth requirements.

ii support concurrent reading and writing of streams. The issue that poses a serious challenge is that disk drives generally provide considerably less write than read bandwidth (see Figure 2b).

iii support multi-zoned disks. Figure 2b illustrates how the disk transfer rates of current generation

Table 2: List of terms used repeatedly in this study and their respective definitions.

Term	Definition	Units
B_{disk}	Block size on disk	MB
T_{svr}	Server observation time interval	second
ξ	The number of disks in the system	
n	The number of concurrent streams	
p_{iodisk}	Probability of missed deadline by reading or writing	
$\overline{R_{Dr}}$	Average disk read bandwidth during T_{svr} (no bandwidth allocation for writing)	MB/s
p_{req}	The threshold of probability of missed deadline, it is the worse situation that client can endure.	
$\overline{R_{Dw}}$	Average disk write bandwidth during T_{svr} (no bandwidth allocation for reading)	MB/s
$t_{seek}(j)$	Seek time for disk access j, where j is an index for each disk access during a T_{svr}	ms
$R_{Dr}(j)$	Disk read bandwidth for disk access j (no bandwidth allocation for writing)	MB/s
$\mu_{t_{seek}}(j)$	Mean value of random variable $t_{seek}(j)$, where j is an index for each disk access during a T_{svr}	ms
$\sigma_{t_{seek}}(j)$	Standard deviation of random variable $t_{seek}(j)$	ms
β	Relationship factor between R_{Dr} and R_{Dw}	
$\overline{t_{seek}}$	The average disk seek time during T_{svr}	ms
$\mu_{t_{seek}}$	Mean value of random variable $\overline{t_{seek}}$	ms
$\sigma_{t_{seek}}$	Standard deviation of random variable $\overline{t_{seek}}$	ms
α	Mixed-load factor, the percentage of reading load in the system	
m_1	The number of movies existed in HYDRA	
D_i^{rs}	The amount of data that movie i is consumed during T_{svr}	MB
μ_i^{rs}	Mean value of random variable D_i^{rs}	MB
σ_i^{rs}	Standard deviation of random D_i^{rs}	MB
n_i^{rs}	The number of retrieving streams for movie i	
m_2	The number of different recording devices	
D_i^{ws}	The amount of data that is generated by recording device i during T_{svr}	
μ_i^{ws}	Mean value of random variable D_i^{ws}	MB
σ_i^{ws}	Standard deviation of random D_i^{ws}	MB
n_i^{ws}	The number of recording streams by recording device i	
N_{max}	The maximum number of streams supported in the system	
S_{min}^{buf}	The minimum buffer size needed in the system	MB

drives is platter location dependent. The outermost zone provides up to 30% more bandwidth than the innermost one.

iv support flexible *service requirements* (see Section 4.1 for details), which should be configurable by Video-on-Demand (VOD) service providers based on their application and customer requirements.

As discussed in Section 3, a double buffering scheme is employed in HYDRA. Therefore, two buffers are necessary for each stream serviced by the system. Before formally defining the MSB problem, we outline our framework for service requirements in the next section. Table 4 lists all the parameters and their definitions used in this paper.

4.1 Service Requirements

Why do we need to consider *service requirements* in our system? We illustrate and answer this question with an example.

Assume that a VOD system is deployed in a five-star hotel, which has 10 superior deluxe rooms, 20 deluxe rooms and 50 regular rooms. There are 30 movies stored in the system, among which five are new releases that started to be shown in theaters during the last week. Now consider the following scenario. The VOD system operator wants to configure the system so that (1) the customers who stay in superior deluxe rooms should be able to view any one of the 30 movies whenever they want, (2) those customers that stay in deluxe rooms should be able to watch any of the five new movies released recently at anytime, and finally (3) the customers in the regular rooms can watch movies whenever system resources permit.

The rules and requirements described above are formally a set of service constraints that the VOD operator would like to enforce in the system. We term these type of service constraints *service requirements*. Such service requirements can be enforced in the VOD system via an admission control mechanism. Most importantly, these service requirements will affect the server buffer requirement. Next, we will describe how to formalize the memory configuration problem and find the minimal buffer size in a streaming media system.

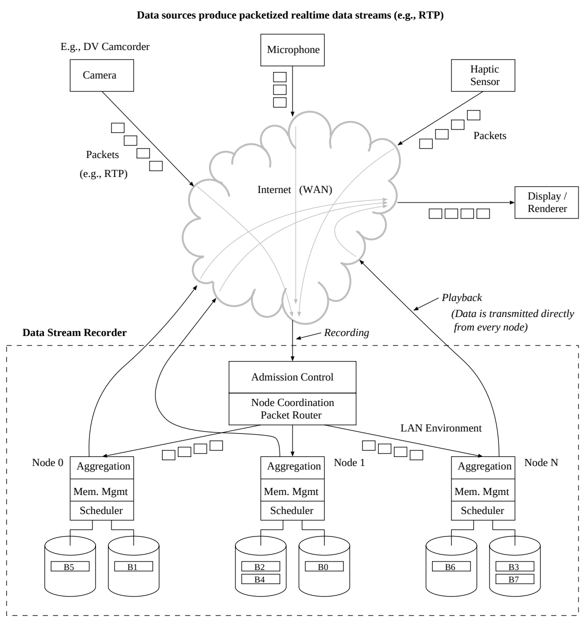

Figure 3: HYDRA: Data Stream Recorder Architecture. Multiple source and rendering devices are interconnected via an IP infrastructure. The recorder functions as a data repository that receives and plays back many streams concurrently.

4.2 MSB Problem Formulation

4.2.1 Stream Characteristics and Load Modeling

Given a specific time instant, there are m_1 movies loaded in the HYDRA system. Thus, these m_1 movies are available for playback services. The HYDRA system activity is observed periodically, during a time interval T_{svr}. Each movie follows an inherent bandwidth consumption schedule due to its compression and encoding format, as well as its specific content characteristics. Let D_i^{rs} denote the amount of data that movie i is consuming during T_{svr}. Furthermore, let μ_i^{rs} and σ_i^{rs} denote the mean and standard deviation of D_i^{rs}, and let n_i^{rs} represent the number of retrieval streams for movie i.

We assume that there exist m_2 different recording devices which are connected to the HYDRA system. These recording devices could be DV camcorders, microphones or haptic sensors as shown in Figure 3. Therefore, in terms of bandwidth characteristics, m_2 types of recording streams must be supported by the recording services in the HYDRA system. Analogous with the retrieval services, D_i^{ws} denotes the amount of data that is generated by recording device i during time interval T_{svr}. Let μ_i^{ws} and σ_i^{ws} denote the mean and standard deviation of D_i^{ws} and let n_i^{ws} represent the number of recording streams generated by recording device i. Consequently, we can compute the total number of concurrent streams n as

$$n = \sum_{i=1}^{m_1} n_i^{rs} + \sum_{i=1}^{m_2} n_i^{ws} \qquad (1)$$

Thus, the problem that needs to be solved translates to finding the combination of $< n_1^{rs} ... n_{m_1}^{rs}, n_1^{ws} ... n_{m_2}^{ws} >$, which maximizes n. Hence, N_{max} can be computed as

$$N_{max} = max(n) = max(\sum_{i=1}^{m_1} n_i^{rs} + \sum_{i=1}^{m_2} n_i^{ws}) \qquad (2)$$

under some *service requirements* described below. Note that if the double buffering technique is employed, and after computing N_{max}, we can easily obtain the minimum buffer size S_{min}^{buf} as

$$S_{min}^{buf} = 2B_{disk}N_{max} \qquad (3)$$

where B_{disk} is the data block size on the disks. Note that in the above computation we are considering the worst case scenario where no two data streams are sharing any buffers in memory.

4.2.2 Service Requirements Modeling

We start by assuming the example described in Section 4.1 and following the notation in the previous section. Thus, let $n_1^{rs}, ..., n_{30}^{rs}$ denote the number of retrieval streams corresponding to the 30 movies in the system. Furthermore, without loss of generality, we can choose $n_1^{rs}, ..., n_5^{rs}$ as the five newly released movies.

To enforce the *service requirements*, the operator must define the following constraints for each of the corresponding service requirements:

C1: $n_1^{rs}, ..., n_{30}^{rs} \geq 10$.

C2: $n_1^{rs}, ..., n_5^{rs} \geq 20$.

Note that we do not define the constraint for the third service requirement because it can be automatically supported by the statistical admission model defined in the next section.

The above constraints are equivalent to the following linear constraints:

C1: $n_1^{rs}, ..., n_5^{rs} \geq 30$.

C2: $n_6^{rs}, ..., n_{30}^{rs} \geq 10$.

These linear constraints can be generalized into the following linear equations:

$$\begin{array}{l} \sum_{j=1}^{m_1} a_{ij}^{rs} n_j^{rs} + \sum_{k=1}^{m_2} a_{ik}^{ws} n_k^{ws} \leq b_i \\ n_j^{rs} \geq 0 \\ n_k^{ws} \geq 0 \\ n_j^{rs} \text{ and } n_k^{ws} \text{ are integers} \end{array} \qquad (4)$$

where $i \in [0, w]$, w is the total number of linear constraints, $j \in [1, m_1]$, $k \in [1, m_2]$, and a_{ij}^{rs}, a_{ik}^{ws}, b_i are linear constraint parameters.

4.2.3 Statistical Service Guarantee

To ensure high resource utilization in HYDRA, we provide statistical service guarantees to end users through a comprehensive three random variable (3RV) admission control model. The parameters incorporated into the random variables are the variable bit rate characteristic of different retrieval and recording streams, a realistic disk model that considers the variable transfer rates of multi-zoned disks, variable seek and rotational latencies, and unequal reading and recording data rate limits.

Recall that system activity is observed periodically with a time interval T_{svr}. Formally, our 3RV model is characterized by the following three random variables: (1) $\sum_{i=1}^{m_1} n_i^{rs} D_i^{rs} + \sum_{i=1}^{m_2} n_i^{ws} D_i^{ws}$, denoting the amount of data to be retrieved or recorded during T_{svr} in the system, (2) $\overline{t_{seek}}$, denoting the average disk seek time during each observation time interval T_{svr}, and (3) $\overline{R_{Dr}}$ denoting the average disk read bandwidth during T_{svr}.

We assume that there are ξ disks present in the system and that p_{iodisk} denotes the probability of a missed deadline when reading or writing, computed with our 3RV model. Furthermore, the statistical service requirements are characterized by p_{req}: the threshold of the highest probability of a missed deadline that a client is willing to accept (for details see (Zimmermann and Fu, 2003)).

Given the above introduced three random variables — abbreviated as X, Y and Z — the probability of missed deadlines p_{iodisk} can then be evaluated as follows

$$\begin{aligned} p_{iodisk} &= P\left[(X,Y,Z) \in \Re\right] \\ &= \iiint_{\Re} f_X(x) f_Y(y) f_Z(z)\, dx\,dy\,dz \\ &\leq p_{req} \end{aligned} \tag{5}$$

where $\Re$ is computed as

$$\Re = \left\{ (X,Y,Z) \mid \frac{X}{\xi} > \left(\frac{(\alpha Z + (1-\alpha)\beta Z) \times T_{svr}}{1 + \frac{Y \times (\alpha Z + (1-\alpha)\beta Z)}{B_{disk}}} \right) \right\} \tag{6}$$

In Equation 6, B_{disk} denotes the data block size on disk, α is the mixload factor, which is the percentage of reading load in the system and is computed by Equation 10, and β is the relationship factor between the read and write data bandwidth. The necessary probability density functions $f_X(x)$, $f_Y(y)$, and $f_Z(z)$ can be computed as

$$f_X(x) = \frac{e^{-\frac{\left[x - (\sum_{i=1}^{m1} n_i^{rs}\mu_i^{rs} + \sum_{i=1}^{m2} n_i^{ws}\mu_i^{ws})\right]^2}{2 \times (\sum_{i=1}^{m1} n_i^{rs}(\sigma_i^{rs})^2 + \sum_{i=1}^{m2} n_i^{ws}(\sigma_i^{ws})^2)}}}{\sqrt{2\pi(\sum_{i=1}^{m1} n_i^{rs}(\sigma_i^{rs})^2 + \sum_{i=1}^{m2} n_i^{ws}(\sigma_i^{ws})^2)}} \tag{7}$$

while $f_Y(y)$ similarly evaluates to

$$f_Y(y) \approx \frac{e^{-\frac{(\sum_{i=1}^{m1} n_i^{rs}\mu_i^{rs} + \sum_{i=1}^{m2} n_i^{ws}\mu_i^{ws})}{2B_{disk}}\left[\frac{y - \mu_{t_{seek}}(j)}{\sigma_{t_{seek}}(j)}\right]^2}}{\sqrt{2\pi\sigma_{t_{seek}}^2(j)}} \tag{8}$$

with $\mu_{t_{seek}}(j)$ and $\sigma_{t_{seek}}$ being the mean value and the standard deviation of the random variable $t_{seek}(j)$, which is the seek time[2] for disk access j, where j is an index for each disk access during T_{svr}. Finally, $f_Z(z)$ can be computed as

$$f_Z(z) \approx \frac{e^{-\frac{(\sum_{i=1}^{m1} n_i^{rs}\mu_i^{rs} + \sum_{i=1}^{m2} n_i^{ws}\mu_i^{ws})}{2B_{disk}}\left[\frac{z - \mu_{R_{Dr}}(j)}{\sigma_{R_{Dr}}(j)}\right]^2}}{\sqrt{2\pi\sigma_{R_{Dr}}^2(j)}} \tag{9}$$

where $\mu_{R_{Dr}}(j)$ and $\sigma_{R_{Dr}}(j)$ denote the mean value and standard deviation for random variable $R_{Dr}(j)$. This parameter represents the disk read bandwidth limit for disk access j, where j is an index for each disk access during a T_{svr}, and α can be computed as

$$\alpha \approx \frac{\sum_{i=1}^{m1} n_i^{rs}\mu_i^{rs}}{\sum_{i=1}^{m1} n_i^{rs}\mu_i^{rs} + \frac{\sum_{i=1}^{m2} n_i^{ws}\mu_i^{ws}}{\beta}} \tag{10}$$

We have now formalized the MSB problem. Our next challenge is to find an efficient solution. However, after some careful study we found that there are two properties — integer constraints and linear equation constraints — that make it hard to solve. In fact, MSB is a $\mathcal{NP}$-complete problem. We will prove it formally in the next section.

4.3 NP-Completeness

To show that MSB is $\mathcal{NP}$-complete, we first need to prove that $MSB \in \mathcal{NP}$.

Lemma 4.1: $MSB \in \mathcal{NP}$

Proof: We prove this lemma by providing a polynomial-time algorithm, which can verify MSB with a given solution $\{n_1^{rs} \ldots n_{m_1}^{rs}, n_1^{ws} \ldots n_{m_2}^{ws}\}$.

We have constructed an algorithm called *Check-Optimal*, shown in Figure 4. Given a set $\{n_1^{rs} \ldots n_{m_1}^{rs}, n_1^{ws} \ldots n_{m_2}^{ws}\}$, the algorithm *CheckOptimal* can verify the MSB in polynomial-time for the following reasons:

[2] $t_{seek}(j)$ includes rotational latency as well.

```
Procedure CheckOptimal ($n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}$)
        /* Return TRUE if the given solution satisfies */
        /* all the constraints and maximize $n$, */
        /* otherwise, return FALSE. */
        (i) $S$={$n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}$},
            If CheckConstraint($S$) == TRUE
            Then continue;
            Else return FALSE;
        (ii) For ($i = 1; i \leq m_1; i++$)
            {
             $S' = S$; $S'.n_i^{rs}$ = $S'.n_i^{rs}$ + 1;
             If CheckConstraint($S'$) == TRUE
             Then return FALSE;
             Else continue;
            }
        (iii) For ($i = 1; i \leq m_2; i++$)
            {
             $S' = S$; $S'.n_i^{ws}$ = $S'.n_i^{ws}$ + 1;
             If CheckConstraint($S'$) == TRUE
             Then return FALSE;
             Else continue;
            }
        (iv).return TRUE;
end CheckOptimal;

Procedure CheckConstraint ($n_1^{rs}...n_{m_1}^{rs}, n_1^{ws}...n_{m_2}^{ws}$)
        /* Return TRUE if the given solution satisfies */
        /* all the constraints, otherwise return FALSE. */
        (i) $S$={$n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}$},
             If $S$ satisfies all the linear constraints defined
                 in Equation 4.
             Then continue;
             Else return FALSE;
        (ii) If $S$ satisfies the statistical service guarantee
                 defined in Equation 5.
             Then return TRUE;
             Else return FALSE;
end CheckConstraint;
```

Figure 4: An algorithm to check if a given solution $\{n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}\}$ satisfies all the constraints specified in Equation 4 and 5 and maximizes n as well.

1 Procedure *CheckConstraint* runs in polynomial time because both step (i) and step (ii) run in polynomial time. Note that the complexity analysis of step (ii) is described in details elsewhere (Zimmermann and Fu, 2003).

2 Based on the above reasoning, we conclude that procedure *CheckOptimal* runs in polynomial time because each of its four component steps runs in polynomial time.

Therefore, $MSB \in \mathcal{NP}$. ▮

Next, we show that MSB is $\mathcal{NP}$-hard. To accomplish this we first define a restricted version of MSB, termed RMSB.

Definition 4.2 : The *Restricted Minimum Server Buffer Problem* (RMSB) is identical to MSB except that $p_{req} = 1$. ▮

Subsequently, RMSB can be shown to be $\mathcal{NP}$-hard by reduction from *Integer Linear Programming* (ILP) (Papadimitriou and Steiglitz, 1982).

Definition 4.3 : The *Integer Linear Programming* **(ILP)** problem:
Maximize $\sum_{i=1}^{n} C_j X_j$
subject to
$\sum_{i=1}^{n} a_{ij} X_j \leq b_i$ for $i = 1, 2, \dots, m$, and
$X_j \geq 0$ and X_j is integer for $j = 1, 2, \dots, n$.
▮

Theorem 4.4: *RMSB is $\mathcal{NP}$-hard.*

Proof: We use a reduction from ILP. Recall that in MSB, Equation 5 computes the probability of a missed deadline during disk reading or writing p_{iodisk}, and p_{iodisk} is required to be less than or equal to p_{req}. Recall that in RMSB, $p_{req} = 1$. Therefore, it is obvious that $p_{iodisk} \leq (p_{req} = 1)$ is always true no matter how the combination of $\{n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}\}$ is selected. Therefore, in RMSB, the constraint of statistical service guarantee could be removed, which then transforms RMSB into an ILP problem. ▮

Theorem 4.5: *MSB is $\mathcal{NP}$-hard.*

Proof: By *restriction* (Garey and Johnson, 1979), we limit MSB to RMSB by assuming $p_{req} = 1$. As a result – based on Theorem 4.4 – MSB is $\mathcal{NP}$-hard. ▮

Theorem 4.6: *MSB is $\mathcal{NP}$-complete.*

Proof: It follows from Lemma 4.1 and Theorem 4.5 that MSB is $\mathcal{NP}$-complete. ▮

4.4 Algorithm to Solve MSB

Figure 5 illustrates the process of solving the MSB problem. Four major parameter components are utilized in the process: (1) *Movie Parameters* (see Section 4.2.1), (2) *Recording Devices* (see Section 4.2.1), (3) *Service Requirements* (see Section 4.2.2), and (4) *Disk Parameters* (for details see (Zimmermann and Fu, 2003)). Additionally, there are four major computation components involved in the process: (1) *Load Space Navigation*, (2) *Linear Constraints Checking*, (3) *Statistical Admission Control*, and (4) *Minimum Buffer Size Computation*.

The *Load Space Navigator* checks through each of the possible combinations $\{n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}\}$ in the search space. It also computes the temporary maximum stream number N_{max} when it receives the results from the admission control module. Each of the possible solutions $\{n_1^{rs} \dots n_{m_1}^{rs}, n_1^{ws} \dots n_{m_2}^{ws}\}$ is first checked

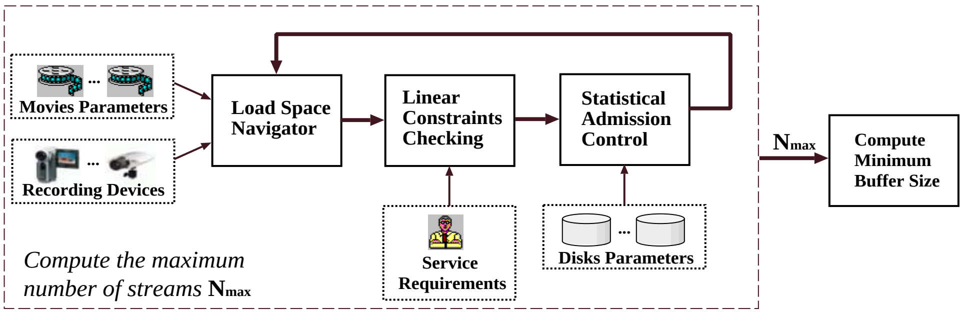
Movies Parameters
Recording Devices
Load Space Navigator
Linear Constraints Checking
Statistical Admission Control
Service Requirements
Disks Parameters
Compute the maximum number of streams Nmax
Nmax
Compute Minimum Buffer Size

Figure 5: Process to solve the MSB problem.

by the *Linear Constraints Checking* module, which enforces the service requirements formulated in Section 4.2.2. The solutions that satisfy the service requirements will be further verified by the *Statistical Admission Control* module described in Section 4.2.3, which provides the statistical service guarantees for the recording system. After exhausting the search space, the load space navigator forwards the highest N_{max} to the *Minimum Buffer Size Computation* module, which computes the minimal buffer size S^{buf}_{min}.

We conclude by providing an algorithm that solves the MSB problem in exponential time shown in Figure 6, based on the process illustrated in Figure 5.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

We have presented a novel buffer minimization problem (*MSB*) motivated by the design of our large scale data stream recording system HYDRA. We formally proved that MSB is $\mathcal{NP}$-complete, and we also provided an initial exponential-time algorithm to solve the problem. As part of our future work, we will focus on finding an approximation algorithm which solves the MSB problem in polynomial-time. Furthermore, we plan to evaluate the memory management module in the context of the other system components that manage data placement, disk scheduling, block prefetching and replacement policy, and QoS requirements. Finally, we plan to implement and evaluate the memory management module in our HYDRA prototype system.

REFERENCES

Aref, W., Kamel, I., Niranjan, T. N., and Ghandeharizadeh, S. (1997). Disk Scheduling for Displaying and Recording Video in Non-Linear News Editing Systems. In *Proceedings of the Multimedia Computing and Networking Conference*, pages 228–239, San Jose, California. SPIE Proceedings Series, Volume 3020.

Chae, Y., Guo, K., Buddhikot, M. M., Suri, S., and Zegura, E. W. (2002). Silo, rainbow, and caching token: Schemes for scalable, fault tolerant stream caching. *Special Issue of IEEE Journal of Selected Area in Communications on Internet Proxy Services*.

Cui, Y. and Nahrstedt, K. (2003). Proxy-based asynchronous multicast for efficient on-demand media distribution. In *The SPIE Conference on Multimedia Computing and Networking 2003 (MMCN 2003), Santa Clara, California*, pages 162–176.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to Theory of NP-Completeness*. W.H.Freeman and Company, New York.

Huffstutter, P. J. and Healey, J. (2002). Filming Without the Film. *Los Angeles Times*, page A.1.

Lee, S.-H., Whang, K.-Y., Moon, Y.-S., and Song, I.-Y. (2001). Dynamic Buffer Allocation in Video-On-Demand Systems. In *Proceedings of the international conference on Management of data (ACM SIGMOD'2001), Santa Barbara, California, United States*, pages 343–354.

Makaroff, D. J. and Ng, R. T. (1995). Schemes for Implementing Buffer Sharing in Continuous-Media Systems. *Information Systems, Vol. 20, No. 6.*, pages 445–464.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632.

Ramesh, S., Rhee, I., and Guo, K. (2001). Multicast with cache (mcache): An adaptive zero delay video-on-demand service. In *IEEE INFOCOM '01*, pages 85–94.

Sen, S., Rexford, J., and Towsley, D. F. (1999). Proxy prefix caching for multimedia streams. In *IEEE INFOCOM '99*, pages 1310–1319.

Shahabi, C. and Alshayeji, M. (2000). Super-streaming: A new object delivery paradigm for continuous media servers. *Journal of Multimedia Tools and Applications*, 11(1).

Shahabi, C., Zimmermann, R., Fu, K., and Yao, S.-Y. D. (2002). Yima: A Second Generation Continuous Media Server. *IEEE Computer*, 35(6):56–64.

Shi, W. and Ghandeharizadeh, S. (1997). Buffer Sharing in Video-On-Demand Servers. *SIGMETRICS Performance Evaluation Review*, 25(2):13–20.

Smith, T. (2003). Next DVD spec. to offer Net access not more capacity. *The Register*.

Tsai, W.-J. and Lee, S.-Y. (1998). Dynamic Buffer Management for Near Video-On-Demand Systems. *Multimedia Tools and Applications, Volume 6, Issue 1*, pages 61–83.

Tsai, W.-J. and Lee, S.-Y. (1999). Buffer-Sharing Techniques in Service-Guaranteed Video Servers. *Multimedia Tools and Applications, Volume 9, Issue 2*, pages 121–145.

Waldvogel, M., Deng, W., and Janakiraman, R. (2003). Efficient buffer management for scalable media-on-demand. In *The SPIE Conference on Multimedia Computing and Networking 2003 (MMCN 2003), Santa Clara, California*.

Wallich, P. (2002). Digital Hubbub. *IEEE Spectrum*, 39(7):26–29.

Zimmermann, R. and Fu, K. (2003). Comprehensive Statistical Admission Control for Streaming Media Servers. In *Proceedings of the 11th ACM International Multimedia Conference (ACM Multimedia 2003)*, Berkeley, California.

Zimmermann, R., Fu, K., and Ku, W.-S. (2003). Design of a large scale data stream recorder. In *The 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers - France*.

```
**Procedure** FindMSB
    /* Return the minimum buffer size */
    (i) $N_{max}$ = FindNmax; /* Find the maximum number of supportable streams */
    (ii) Compute $S_{min}^{buf}$ using Equation 3.
    (iii) return $S_{min}^{buf}$;
**end** FindMSB;

**Procedure** FindNmax
    /* Return the maximum number of supportable streams */
    (i) Considering only statistical service guarantee $p_{req}$, let $N_i^{rs}$ denote the maximum of supportable
        retrieving streams of movie $i$ without any other system load. Find the $N_i^{rs}$, where $i \in [1, m_1]$.
    (ii) Considering only statistical service guarantee $p_{req}$, let $N_i^{ws}$ denote the maximum of supportable
        recording streams of generated by recording device $i$ without any other system load.
        Find the $N_i^{ws}$, where $i \in [1, m_2]$.
    (iii) $N_{curmax} = 0$; $S_{curmax}$={$0 \ldots 0, 0 \ldots 0$}
    (iv) For ($X_1^{rs} = 1$; $X_1^{rs} \leq N_1^{rs}$; $X_1^{rs} ++$)
         . . . . . .
         For ($X_{m_1}^{rs} = 1$; $X_{m_1}^{rs} \leq N_{m_1}^{rs}$; $X_{m_1}^{rs} ++$)
         For ($X_1^{ws} = 1$; $X_1^{ws} \leq N_1^{ws}$; $X_1^{ws} ++$)
         . . . . . .
         For ($X_{m_2}^{ws} = 1$; $X_{m_2}^{ws} \leq N_{m_2}^{ws}$; $X_{m_2}^{ws} ++$)
         {
           $S' = \{X_1^{rs} \ldots X_{m_1}^{rs}, X_1^{ws} \ldots X_{m_2}^{ws}\}$;
           If CheckConstraint($S'$) == TRUE /* CheckConstraint is defined in Figure 4 */
           Then
           {
             If $\sum_{i=1}^{m_1} X_i^{rs} + \sum_{i=1}^{m_2} X_i^{ws} > N_{curmax}$
             Then
               $N_{curmax} = \sum_{i=1}^{m_1} X_i^{rs} + \sum_{i=1}^{m_2} X_i^{ws}$; $S_{curmax}$={$X_1^{rs} \ldots X_{m_1}^{rs}, X_1^{ws} \ldots X_{m_2}^{ws}$}
           }
         }
    (v) return $N_{curmax}$;
**end** FindNmax;
```

Figure 6: Algorithm to solve MSB problem.