To appear in the IEEE Journal on Selected Areas in Communication, Special Issue on High Definition Television and Digital Video Communication, Vol. 11, No. 1, January 1993

Designing a Multi-User HDTV Storage Server

Harrick M. Vin and **P. Venkat Rangan**

Multimedia Laboratory
Depaartment of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093

E-mail: {vin,venkat}@cs.ucsd.edu, Phone: (619) 534-5419, Fax: (619) 534-7029

Abstract

Future advances in networking coupled with the rapid advances in storage technologies will make it feasible to build a HDTV-on-demand server (that provides services similar to those of a neighborhood videotape rental store) on a metropolitan-area network. In this paper, we present a quantitative study of designing a multi-user HDTV server, and present efficient techniques for (1) storing multiple HDTV videos on disk, and (2) servicing multiple subscriber requests simultaneously, both under the constraint of guaranteeing HDTV playback rates.

We develop a model that relates disk and device characteristics to the HDTV playback rate, and derive a storage pattern for HDTV video streams that guarantees their real-time retrieval. Given multiple HDTV streams, we develop mechanisms for merging their individual storage patterns together. We propose an off-line merging algorithm that can be applied a priori, and an on-line algorithm suitable for merging a new HDTV stream into a set of already stored HDTV streams, both of which yield a large improvement in space utilization over storing each of the streams independently. We study various policies, such as, round robin and quality proportional for servicing multiple subscribers simultaneously. The quality proportional algorithm retrieves video frames at a rate proportional on an average to the HDTV playback rates of subscribers, but uses a staggered toggling technique in which successive numbers of retrieved frames are fine tuned individually to achieve the servicing of an optimal number of subscribers simultaneously. The algorithm is powerful enough to accommodate bounded availability of HDTV display buffers, and permits dynamic additions and deletions of subscriber requests in a transparent manner (i.e., without causing discontinuity in the retrieval of any of the existing subscribers). In summary, our studies provide a quantitative demonstration of the technological feasibility and economic viability of HDTV-on-demand servers on metropolitan area networks.

1 Introduction

Recent developments in networking coupled with the advent of high capacity storage devices will make it feasible to support HDTV-on-demand servers over metropolitan-area networks, such as B-ISDN, that are expected to permeate residential and commercial premises in a manner similar to existing cable TV or telephone networks [10, 11]. A HDTV-on-demand storage server, which we will refer to as a **HDTV server** in the rest of this paper, provides services similar to those of a neighborhood videotape rental store. It digitally stores HDTV video such as entertainment movies, educational documentaries, advertisements, etc., on a large array of extremely high-capacity storage devices such as optical or magnetic disks, that are random accessible with a short seek time, and

are permanently on-line. The HDTV server is connected to display devices (such as TVs) belonging to residential subscribers via a high-speed metropolitan area network (see Figure 1). Subscribers can make a selection of a video through a variety of indices such as the video's subject title, and request its retrieval for real-time playback on their display devices. The HDTV server, if it has the necessary resources (such as service time and buffer space), satisfies the subscriber's request by connecting to his/her chosen display device(s), and transmitting the chosen video segment. The retrieval is *interactive*, in the sense that subscribers can stop, pause, resume, and even record[1] and edit the video if they have permissions to do so. Thus, a HDTV server also subsumes the functions of VCRs, video tapes, audio recorders, etc., and can serve varying sizes of clientele: from individual households to entire neighborhoods, and from commercial organizations and educational institutions to national services.

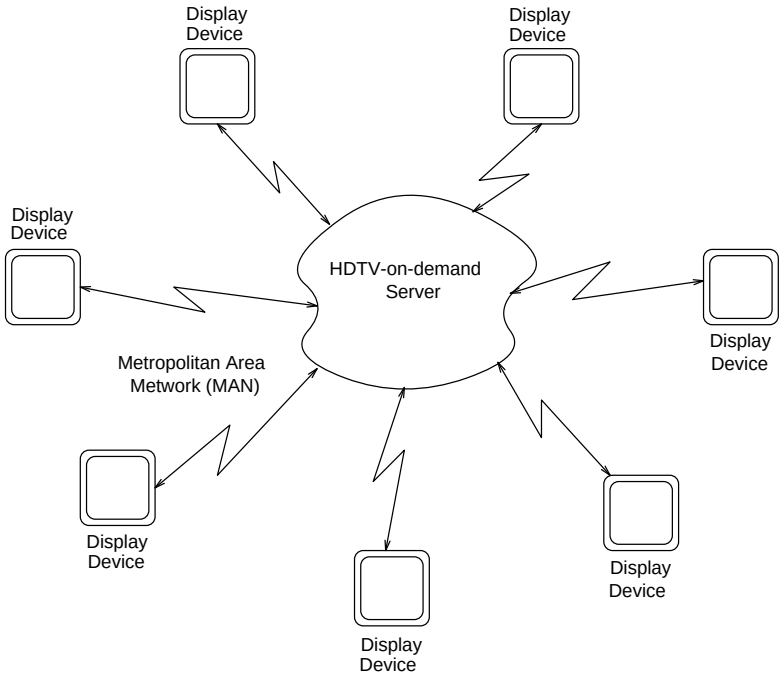

Figure 1: Configuration of a large scale HDTV-on-demand server

The above architectural vision of a HDTV server is feasible within the next several years (rather than decades). To see why, consider the storage and transmission capacities required for a HDTV server. Assuming HDTV video to require a data rate of about 2 Mbytes/s [3], a 100 minute long movie requires 12 Gbytes. Storage of 1000 such videos requires a capacity of 12 terabytes. In comparison, the capacity of currently available disks are 10 GBytes, and are expected to increase to 100 Gbytes in a few years. Thus, with an array of 120 disks, a HDTV server can store 1000 popular movies simultaneously. As for the transmission capacities, fiber-optic networks offering gigabyte bandwidths are already in place, and those offering terabyte bandwidths are conjectured to be only a few years away. However, a key deciding factor for the feasibility of such a HDTV server is its economic

[1] Throughout this paper, we will use the terms playback and retrieval, as well as recording and storage synonymously.

viability. Assuming costs of about $ 4,000 per disk, the total expected installation cost of a HDTV server will be $ 0.48 million, which when amortized over 1000 subscribers is about $ 480 per subscriber, making it a viable alternative to owning a VCR. The design of a high-performance storage server that can satisfy a large number of HDTV-on-demand retrievals from multiple users simultaneously is the subject matter of this paper.

There are two important questions that need to be addressed in designing such a multi-subscriber HDTV server[2]: (1) how should multiple HDTV videos be laid out on disk storage, and (2) how can multiple retrieval requests from multiple subscribers be serviced simultaneously by the same HDTV server? These two problems are inter-related: the storage pattern of a video (i.e., the relative positions of successive stored blocks of digitized data comprising the video) governs the rate at which each individual video can be retrieved, which must equal at least the HDTV playback rate even when the server is multiplexing itself among multiple subscribers simultaneously.

In this paper, we present a quantitative study of designing a multi-user HDTV server. We present algorithms for (1) collocational storage of maximum number of HDTV videos on disk, and (2) simultaneous servicing of maximum number of subscribers' requests. We develop a model that relates disk and device characteristics to the HDTV playback rate, and derive a storage pattern for HDTV videos that guarantees their real-time retrieval. To maximize the storage utilization of multiple HDTV videos, we develop mechanisms for merging their individual storage patterns together. We propose both an off-line merging algorithm that can be applied a priori to the storage of a set of HDTV videos before any of them have been stored on disk, and an on-line algorithm suitable for merging a new HDTV video into a set of already stored HDTV videos.

We study various policies (such as, round robin and quality proportional) for servicing multiple subscribers simultaneously, and propose algorithms by which a HDTV server can enforce these policies without violating the real-time retrieval rates of any of the subscribers. The quality proportional algorithm retrieves video frames at a rate proportional on an average to the HDTV playback rates of requests, but uses a staggered toggling technique by which successive numbers of retrieved frames are fine tuned individually to service an optimal number of subscribers simultaneously. The algorithm is powerful enough to accommodate bounded availability of HDTV display buffers, and permits dynamic additions and deletions of subscriber requests in a transparent manner (i.e., without causing discontinuity in the retrieval of any of the existing subscriber requests). We evaluate the performance of the quality proportional multi-subscriber servicing algorithm, and show that it is an order of magnitude scalable compared to straightforward multiplexing techniques such as servicing one subscriber per disk head and round robin servicing of subscribers.

The rest of this paper is organized as follows: In Section 2, we present HDTV storage techniques, in Section 3, we develop multi-subscriber servicing algorithms, and in Section 4, we present their performance evaluation. Section 5 identifies related work in this area, and finally, Section 6 concludes the paper.

[2] Building a dedicated, single-subscriber HDTV server does not offer very many design choices, and is relatively straightforward.

Symbol	*Explanation*	*display unit*
$\mathcal{R}_{pl}$	HDTV playback rate	display units/sec
$\mathcal{R}_{dr}$	Disk data transfer rate	bits/sec
$\mathcal{R}_{ds}$	Disk scan rate	bits/sec
η_{vs}	Granularity of HDTV storage	display units
s_{vf}	Size of a HDTV display unit	bits/display unit
l_{ds}	Scattering parameter	sec

Table 1: Symbols used in this paper. A *display unit* represents a frame for video, and a sample for audio

2 Efficient Storage of HDTV Video

Digitization of HDTV video yields a sequence of frames, and that of its accompanying audio yields a sequence of samples. We call a sequence of continuously recorded HDTV video frames or audio samples a *Strand*. A HDTV server must divide video and audio strands into blocks while storing them on a disk. Most existing storage server architectures employ unconstrained allocation of blocks on disk. Such storage servers cannot handle HDTV strands because, separations between blocks of a strand may not be constrained enough to guarantee bounds on access and latency times of successive blocks of the strand. At the other end of the spectrum, contiguous allocation of blocks of a strand can guarantee continuous access, but it is fraught with inherent problems of fragmentation and can entail enormous copying overheads during insertions and deletions. Constrained block allocation, on the other hand, can keep the access time within HDTV requirements without entailing the above disadvantages.

There are two questions that need to be answered in constrained allocation of blocks of a media strand: (1) What should the size of the blocks (i.e. the *granularity*) be? and (2) What should the separation between successive blocks (i.e. the *scattering parameter*) of a strand be? The guiding factor in determining the block size and separation is the requirement of continuous retrieval at HDTV rates. Table 1 defines the symbols for these parameters, using which, it can be seen that the playback duration of a HDTV block is given by $\frac{\eta_{vs}}{\mathcal{R}_{pl}}$. Retrieval at HDTV rates requires that the total delay to read each HDTV block from disk (given by $l_{ds} + \frac{\eta_{vs} * s_{vf}}{\mathcal{R}_{dr}}$) be bounded by the its playback duration:

$$l_{ds} + \frac{\eta_{vs} * s_{vf}}{\mathcal{R}_{dr}} \leq \frac{\eta_{vs}}{\mathcal{R}_{pl}} \tag{1}$$

The relative values of granularity (η_{vs}) and the scattering parameter (l_{ds}) for each HDTV strand must satisfy Equation (1). Since there are two parameters and one equation, one of these parameters, namely the granularity can be fixed based on the hardware environment and the amount of buffer space available at the display devices. Having fixed the granularity, the upper bound on the scattering parameter, l_{ds} can be obtained by direct substitution in Equation (1). Using the values of η_{vs} and l_{ds}, the size of each data block M and the separation between successive blocks G for a HDTV strand can be derived as:

$$\begin{aligned} M &= \eta_{vs} * s_{vf} \\ G &= l_{ds} * \mathcal{R}_{dr} \end{aligned} \tag{2}$$

The pair (M, G) defines the storage pattern of a HDTV strand, and the strand itself consists of repetitions of its storage pattern. For example, if a HDTV video strand is digitized at 0.5 Mbits/frame and recorded at 60 frames/s on a disk transfer rate of 1 Gbits/s, then choosing the granularity η_{vs} to be 1 frame/block yields a scattering parameter $l_{ds} \leq 16.16ms$, which together go to define the strand's storage pattern (M, G) to be (0.5 Mbits, 16.16 Mbits) (see Figure 2).

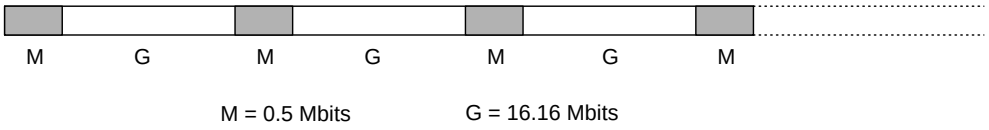

Figure 2: Storage pattern of a HDTV strand $S = (M, G)$ when $\mathcal{R}_{dr} = \mathcal{R}_{ds} = 1$ Gbits/s, $\mathcal{R}_{pl} = 60$ frames/s, $\eta_{vs} = 1$, and $s_{vf} = 0.5$ Mbits

2.1 Storage of Multiple HDTV Strands

A HDTV server needs to store thousands of video strands on disk. If there are sufficiently large empty regions on the disk, each strand may be stored exactly in accordance with its storage pattern. However, storing each strand independently entails the unusability of all the gaps in its storage pattern, resulting in an occupancy of $\frac{M}{M+G}$. To utilize the disk space much more efficiently, data blocks of a new strand may have to be stored in the gaps of already existing strands on the disk. We refer to this process as *merging*. Intuitively, if we assume that the length of strands can be unbounded, then a set of strands $S_1, S_2, ..., S_n$ can be merged together if the sum of the fractions of space occupied by data blocks of $S_1, S_2, ..., S_n$ does not exceed 1:

Merge Condition: A set of HDTV strands $S_1, S_2, ..., S_n$ with storage patterns (M_1, G_1), (M_2, G_2), ..., (M_n, G_n), respectively, can be merged together only if:

$$\frac{M_1}{M_1 + G_1} + \frac{M_2}{M_2 + G_2} + \cdots + \frac{M_n}{M_n + G_n} \leq 1 \tag{3}$$

For HDTV strands with storage pattern (0.5 Mbits, 16.16 Mbits), Equation (3) permits 32 strands to be stored in a merged form in the space spanned by one strand.

When storing data blocks of a new HDTV strand with storage pattern (M, G) in the gaps of existing strands, it may not be possible to maintain the storage pattern of the new strand. Since strand patterns are constructed so as to exactly satisfy real-time retrieval rates of HDTV, nonconformance with a strand pattern can result in violation of HDTV rate requirements during retrieval. Such a violation can be avoided by introducing finite buffering between the HDTV server and the display device, and prefetching a finite number of data blocks of the merged strand and storing them in memory buffers before initiating its playback. This is feasible only if the relative ratio of the number of data blocks and gaps for the merged strand is maintained to equal $\frac{M}{G}$ at least on an average over a finite length, in which case, buffering can nullify the effects of jitter in the pattern, and result in the relaxation of the condition for retrieval at HDTV rates. The exact prefetch and buffering requirements depend on the placement of the new strand's data blocks in the gaps of the strands already stored on the disk.

We now propose placement strategies for a strand to be merged so as to guarantee its retrieval at HDTV rates even after merger, while at the same time minimizing the accompanying prefetch and buffering requirements. Consider two HDTV strands S_1 and S_2 with storage patterns (M_1, G_1) and (M_2, G_2), respectively[3]. Let S_1 be laid out on the disk in accordance with its storage pattern. Merging S_2's pattern into that of S_1 is straight-forward if $G_1 = M_2$ and $M_1 = G_2$, in which case, each data block of S_2 will exactly fit into a gap of S_1. However, this can be very restrictive. In general, if the merge condition (Equation (3)) is satisfied, a simple derivation yields that over a length $L = LCM(M_1 + G_1, M_2 + G_2)$, the data blocks of strand S_2 are guaranteed to fit in the gaps of S_1. That is, if there are p_1 patterns of S_1 and p_2 patterns of S_2 that can span a length L, then:

$$p_2 * M_2 \leq p_1 * G_1$$

where, $p_1 = \frac{L}{M_1+G_1}$ and $p_2 = \frac{L}{M_2+G_2}$. After the merger, L represents the length of a cycle over which S_2's merged pattern repeats. Since the number of data blocks of S_2 in a merge cycle L is the same as the number of blocks of S_2 in its unmerged pattern of length L (see Figure 3(b)), and since retrieval at HDTV rates is guaranteed for the unmerged pattern, playback of S_2's merged pattern at HDTV rates can be guaranteed over an average of length L. Specifically, prefetching all of the data blocks (p_2 in number) of S_2 within the first merge cycle, and initiating playback just after beginning the transfer of the next set of p_2 blocks from the second merge cycle, guarantees that S_2's playback proceeds continuously at its HDTV rate.

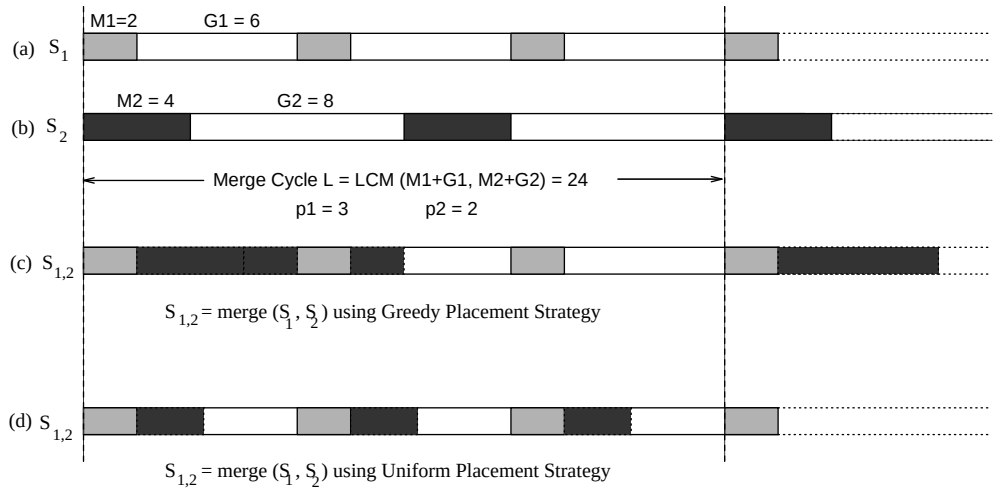

Figure 3: Merging the storage of strands $S_1 = (M_1, G_1)$ and $S_2 = (M_2, G_2)$

Under these conditions, the simplest approach to place S_2's data blocks is to fill S_1's gaps continuously starting from the very first gap, yielding a greedy placement policy (see Figure 3(c)). After storing the blocks (p_2 in number) belonging to each merge cycle, the remaining gaps up to the end of that merge cycle would be left

[3]Each strand may be associated with different values of M and G due to variation in the data rates, which are a result of variations in recording rates, rates of compression, and differences in media (e.g., audio and video).

free. However, if S_2's pattern is sparse compared to the empty space available in S_1, then the greedy placement policy causes a large number of data blocks of S_2 to be read earlier than their time of display leading to peaks in buffering requirements, which at HDTV data rates can be quite large. Buffering needs can be reduced if, instead, the p_2 data blocks of S_2 in each merge cycle are uniformly distributed across all the gaps of S_1 in that merge cycle (see Figure 3(d)).

The above binary merging techniques can be easily extended to three or more strands. For instance, consider the process of merging strands $S_1 = (M_1, G_1)$, $S_2 = (M_2, G_2)$, and $S_3 = (M_3, G_3)$. Let $S_{1,2}$ be the composite strand obtained by merging the patterns of S_1 and S_2. Then, in order to merge the patterns of S_3 with $S_{1,2}$, we derive the effective block size $M_{1,2}$ and gap size $G_{1,2}$ of the composite strand $S_{1,2}$ to be:

$$\begin{aligned} M_{1,2} &= p_1 * M_1 + p_2 * M_2 \\ G_{1,2} &= L_{1,2} - M_{1,2} \end{aligned}$$

where $L_{1,2}$ = LCM$(M_1 + G_1, M_2 + G_2)$, $p_1 = \frac{L_{1,2}}{M_1+G_1}$, and $p_2 = \frac{L_{1,2}}{M_2+G_2}$, and then repeat the binary merging procedure for $S_{1,2}$ and S_3.

As more and more strands are merged together, gaps become more and more scarce, as a result of which larger cycle lengths L are necessary for storing the data blocks of newer strands. Since the buffer space needed increases directly with the number of blocks p_2 of S_2 within a merge cycle, increase in the merge cycle length L yields higher buffer space requirement. On the contrary, if none of the strands that need to be stored by a HDTV server have been physically placed on the disk, the patterns of each of the strands can be determined so as to be exactly mergeable, thereby eliminating pattern deviations for any of the strands when they are stored in a merged form, and consequently reducing the buffer space requirements. Such an off-line merging technique, a fitting application of which is in the placement of HDTV strands on write-once optical disks (such as, WORMs and CLVs), is elaborated next.

2.2 Off-line Merging

Suppose that HDTV strands $S_1, S_2, ..., S_n$ with storage patterns (M_1, G_1), (M_2, G_2), ..., (M_n, G_n), respectively, are to be stored in a merged form on the disk. Let the strands be placed on disk such that p_1 blocks of S_1, p_2 blocks of S_2, ..., p_n blocks of S_n follow each other, and the sequence repeats indefinitely (see Figure 4).

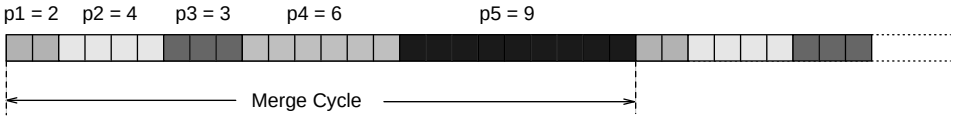

Figure 4: Off-line merging

Guaranteeing retrieval at HDTV rates for each strand S_i requires that the space occupied by blocks of all other strands does not exceed the maximum gap space permitted by S_i's pattern for its blocks in the sequence

(which is G_i for each block of size M_i, as given by Equation (2)). That is,

$$\forall i \in [1,n]: \quad \sum_{j \in [1,n], j \neq i} p_j * M_j \leq p_i * G_i \tag{4}$$

The values of $p_1, p_2, ..., p_n$ satisfying the above system of n equations define a HDTV *merge cycle*. We now propose a *scaled placement policy*, in which the number of consecutive blocks p_i of a strand S_i placed in a merge cycle is inversely scaled by its pattern length (i.e., $M_i + G_i$). That is, $\forall i \in [1,n]$:

$$p_i = \frac{p}{M_i + G_i}$$

where, p is a constant. The following theorem proves that the scaled placement policy will always yield a solution if one exists, thereby showing that it is complete in its effectiveness.

Theorem 1 *Whenever the merge condition (Equation (3)) is satisfied, the scaled placement policy always yields a HDTV merge cycle.*

Proof: In the scaled placement policy, the number of consecutive blocks p_i of a strand S_i placed in a merge cycle is given by $p_i = \frac{p}{M_i+G_i}$. The scaled placement policy yields a solution if Equation (4), which reduces to:

$$\forall j \in [1,n]: \quad \sum_{i \in [1,n], i \neq j} \frac{p * M_i}{M_i + G_i} \leq \frac{p * G_j}{M_j + G_j}$$

$$\Rightarrow \forall j \in [1,n]: \quad \sum_{i \in [1,n], i \neq j} \frac{M_i}{M_i + G_i} \leq \frac{G_j}{M_j + G_j} \tag{5}$$

is satisfied. Substituting $\frac{G_j}{M_j+G_j} = 1 - \frac{M_j}{M_j+G_j}$ in Equation (5), which is surprisingly independent of p, and rearranging terms, we get:

$$\sum_{i=1}^{n} \frac{M_i}{M_i + G_i} \leq 1$$

which is nothing but the merge condition (Equation (3)), which goes to prove that, the scaled placement policy yields a solution whenever the merge condition is satisfied.

□

As explained in the derivation of Equation (4), for each strand S_i, fetching its p_i blocks within each merge cycle is sufficient to guarantee HDTV rate retrieval for the duration of the merge cycle. Hence, no prefetch is required to initiate playback, and a buffer space of at most p_i blocks is sufficient. However, choosing a value of p (from which the values of p_i are derived) so as to satisfy the buffering constraints may result in non-integral values for p_i's. Truncating or rounding off the real values of p_i so obtained may not guarantee that Equation (4) is satisfied, and hence may not guarantee that the retrieval of each of the merged strands proceeds at HDTV rates. In Section 3.3, we describe a technique for toggling between $\lfloor p_i \rfloor$ and $\lceil p_i \rceil$ for each strand in a staggered manner between successive merge cycles, so as to guarantee retrieval at HDTV data rates for each of the merged strands.

3 Servicing Multiple Subscribers Simultaneously

Till now, we have investigated techniques for optimizing space utilization in a HDTV server. We shall now develop techniques for optimizing the service time so as to satisfy the maximum number of subscriber requests simultaneously in real time. In the best scenario, all the subscribers request the retrieval of the same video (for instance, a popular movie), in which case, the HDTV server needs only to retrieve the video once from the disk and then multicast it to all the subscribers. However, more often than not, different subscribers may request the retrieval of different videos, or even when it is a popular movie being requested by multiple subscribers, there may be phase shifts among their requests, i.e., each subscriber viewing a different part of the movie at the same time. A simple mechanism to guarantee that the real-time retrieval rates of none of the requests are violated is to dedicate each disk head to service one request. This limits the total number of simultaneous requests to the number of disk heads, which is about 120 in the configuration proposed in Section 1, in contrast to the estimated 1000 subscribers that are needed to make the HDTV server economically viable.

In this section, we develop algorithms to support the maximum number of subscriber requests simultaneously, under the constraint that each of their retrievals must be guaranteed to proceed at its HDTV rate. In order to precisely formulate this requirement, let us suppose that a HDTV server is servicing n subscribers, each of whom is retrieving his/her own HDTV strand. Let $\eta_{vs}^1, \eta_{vs}^2, ..., \eta_{vs}^n$ denote the granularities of the n strands being retrieved, $l_{ds}^1, l_{ds}^2, ..., l_{ds}^n$ denote their scattering parameters, and $\mathcal{R}_{pl}^1, \mathcal{R}_{pl}^2, ..., \mathcal{R}_{pl}^n$ their playback rates. These differences among strands (even though each contains HDTV video) may arise due to differences in their levels of compression and encryption (employed possibly for copyright protection), differences in frame rates, and differences in picture quality. The HDTV server multiplexes among all the n subscribers, transferring a finite number of blocks k_i of each request $i \in [1, n]$, before switching to the next request. Each sequence of transfers $k_1, k_2, ..., k_n$ constitutes a *service round*, and the HDTV server repeatedly executes service rounds until completion of the requests. Whereas the rate of transfer of successive blocks of each request is governed by the granularity and scattering parameters of its strand, switching from one request to another may entail an overhead of up to the maximum disk seek time plus the maximum rotational latency, to move the disk head from a block of the first strand to a block of the second strand (since the layout does not constrain the relative positions of two different strands). Thus, the total time spent retrieving k_i blocks of i^{th} request in a service round can be said to consist of:

1. θ_i^1: The overhead of switching from the previous request to the ith request, and then transferring the first block of ith request. Since the overhead due to seek time is bounded by l_{seek}^{max}, and since the rotational latency to access a media block placed on a track is bounded by l_{rot}^{max}, we get:

$$\Rightarrow \theta_i^1 = l_{seek}^{max} + l_{rot}^{max} \tag{6}$$

2. θ_i^2: The time to transfer remaining $(k_i - 1)$ blocks of this request in this service round.

$$\Rightarrow \theta_i^2 = \sum_{j=1}^{k_i - 1} (l_{ds}^i + \frac{\eta_{vs}^i * s_{vf}^i}{\mathcal{R}_{dr}}) \tag{7}$$

Hence, the total time spent servicing ith request in a round is

$$\theta_i = \theta_i^1 + \theta_i^2 \tag{8}$$

and the total time spent servicing one round of all the n requests is given by:

$$\Theta = \sum_{i=1}^{n} \theta_i = n * (l_{seek}^{max} + l_{rot}^{max}) + \sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} \left(l_{ds}^i + \frac{\eta_{vs}^i * s_{vf}^i}{\mathcal{R}_{dr}} \right) \tag{9}$$

The HDTV retrieval rate for each of the requests can be satisfied if and only if the service time per round does not exceed the minimum of the playback durations of all the requests. That is,

$$n * (l_{seek}^{max} + l_{rot}^{max}) + \sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} \left(l_{ds}^i + \frac{\eta_{vs}^i * s_{vf}^i}{\mathcal{R}_{dr}} \right) \leq \min_{i \in [1,n]} \left(k_i * \frac{\eta_{vs}^i}{\mathcal{R}_{vp}^i} \right) \tag{10}$$

In order to determine whether a HDTV server can provide *deterministic* service guarantees to each of the n subscribers, the values of l_{ds}^i and s_{vf}^i, $\forall i \in [1, n]$, in Equation (10) must be set to their respective maximum values. However, this may be very pessimistic, since media block allocation policies and variable rate compression techniques (such as, JPEG and MPEG) may yield l_{ds} and s_{vf} significantly smaller than their respective maximum values. Consequently, the number of subscribers that can be serviced simultaneously can be increased by considering the variations in l_{ds} and s_{vf}, and providing *statistical* service guarantees to each of the subscribers. Specifically, if l_{ds}^i represents the random variable characterizing the separation between successive media blocks, and if s_{vf}^i represents the random variable characterizing the bursty bit size distribution of frames yielded by compression techniques such as JPEG and MPEG, then the term

$$\sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} \left(l_{ds}^i + \frac{\eta_{vs}^i * s_{vf}^i}{\mathcal{R}_{dr}} \right)$$

in Equation (10) denotes the sum of $2 * \sum_{i=1}^{n} (k_i - 1)$ independent random variables, and can itself be represented as a random variable (say χ). Hence, Equation (10) reduces to:

$$\chi \leq \min_{i \in [1,n]} \left(k_i * \frac{\eta_{vs}^i}{\mathcal{R}_{pl}^i} \right) - n * (l_{seek}^{max} + l_{rot}^{max}) \tag{11}$$

Thus, if F_χ is the probability distribution function of χ, then guaranteeing simultaneous continuous playback of n HDTV video strands with a probability greater than π necessitates that:

$$F_\chi \left(\min_{i \in [1,n]} \left(k_i * \frac{\eta_{vs}^i}{\mathcal{R}_{pl}^i} \right) - n * (l_{seek}^{max} + l_{rot}^{max}) \right) \geq \pi \tag{12}$$

The HDTV server can service all the n requests simultaneously if and only if $k_1, k_2, \ldots, k_n$ can be determined such that either Equation (10) (in the case of deterministic guarantees) or Equation (12) (in the case of statistical guarantees) is satisfied. Since both of these formulations contain n parameters and only one equation, determination of the values of $k_1, k_2, \ldots, k_n$ require additional policies. The goal of all of such policies is to satisfy maximum number of subscribers simultaneously.

We will henceforth study deterministic servicing policies; extensions to statistical servicing can be carried out fairly easily. To simplify the analysis, we assume that the variations in granularity can be absorbed into the variation of recording rate, and therefore substitute $\eta^1_{vs} = \eta^2_{vs} = \cdots = \eta^n_{vs} = \eta_{vs}$ in Equation (10), yielding:

$$n * (l^{max}_{seek} + l^{max}_{rot}) + \sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} \left(l^i_{ds} + \frac{\eta_{vs} * s^i_{vf}}{\mathcal{R}_{dr}} \right) \leq \min_{i \in [1,n]} \left(k_i * \frac{\eta_{vs}}{\mathcal{R}^i_{pl}} \right) \tag{13}$$

The simplest policy for the choice of $k_1, k_2, \ldots, k_n$ is to use the same value for all of them, yielding what is generally referred to as a *round robin* servicing algorithm. Formally, if $k_1 = k_2 = \cdots = k_n = k$, Equation (13) reduces to

$$n * (l^{max}_{seek} + l^{max}_{rot}) + n * (k-1) * \left(l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}} \right) \leq k * \min_{i \in [1,n]} \left(\frac{\eta^i_{vs}}{\mathcal{R}^i_{pl}} \right)$$

$$\Rightarrow k \geq \frac{n * \left((l^{max}_{seek} + l^{max}_{rot}) - \left(l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}} \right) \right)}{\min_{i \in [1,n]} \left(\frac{\eta_{vs}}{\mathcal{R}^i_{pl}} \right) - n * \left(l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}} \right)}$$

Since any media block can be retrieved from disk within time $(l^{max}_{seek} + l^{max}_{rot})$ starting from any other location on disk, it is guaranteed that $\left(l^{avg}_{ds} + \frac{\eta_{ms} * s_{mu}}{\mathcal{R}_{dr}} \right) \leq (l^{max}_{seek} + l^{max}_{rot})$. Hence, for k to be non-negative, the denominator must be positive, yielding:

$$\min_{i \in [1,n]} \left(\frac{\eta_{vs}}{\mathcal{R}^i_{pl}} \right) > n * \left(l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}} \right)$$

Rearranging the above equation, we obtain the maximum number of subscribers that can be serviced in a round robin algorithm to be:

$$n^c_{max} = \frac{\min_{i \in [1,n]} \left(\frac{\eta_{vs}}{\mathcal{R}^i_{pl}} \right)}{\left(l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}} \right)} \tag{14}$$

Clearly, the number of subscribers that can be serviced by the round robin algorithm is limited by the request with maximum playback rate. This certainly may not be the optimal number of subscribers, because, whereas the subscriber with the maximum playback rate will have retrieved exactly the number of data blocks it needs for the duration of a service round, other subscribers whose playback rates are smaller will have retrieved more data blocks than they need in each service round. Consequently, by reducing the number of data blocks retrieved per service round for such subscribers, it may be possible to accommodate more number of subscribers. We now propose an algorithm that tries to allocate values to k_i proportional to the playback rate of the strand requested by subscriber i, and show that it supports the optimal number of subscribers (that is, it always yields values of k_i so as to satisfy Equation (13) whenever a solution exists for the given number of subscribers).

3.1 Quality Proportional Multi-Subscriber Servicing

In the *Quality Proportional Multi-subscriber Servicing (QPMS)* algorithm, the number of blocks accessed during each round for each subscriber request is proportional to its playback rate[4]. That is,

$$\forall i \in [1,n]: \quad k_i \propto \mathcal{R}_{pl}^i$$

Let k be the proportionality constant, using which, we get, $k_1 = k * \mathcal{R}_{pl}^1$, $k_2 = k * \mathcal{R}_{pl}^2$, ..., $k_n = k * \mathcal{R}_{pl}^n$. Under these conditions, Equation (13) reduces to:

$$n * (l_{seek}^{max} + l_{rot}^{max}) + k * \sum_{i=1}^{n} \mathcal{R}_{pl}^i * (l_{ds}^i + \frac{\eta_{vs} * s_{vf}^i}{\mathcal{R}_{dr}}) - \sum_{i=1}^{n} (l_{ds}^i + \frac{\eta_{vs} * s_{vf}^i}{\mathcal{R}_{dr}}) \leq k * \eta_{vs} \tag{15}$$

In the above equation, l_{ds}^i denotes the scattering parameter, and, by Equation (1), it is inversely proportional to the playback rate $\mathcal{R}_{pl}^i$. Hence, using basic algebra[5], it can be shown that:

$$\sum_{i=1}^{n} \mathcal{R}_{pl}^i * (l_{ds}^i + \frac{\eta_{vs} * s_{vf}^i}{\mathcal{R}_{dr}}) \leq n * \mathcal{R}_{pl}^{avg} * (l_{ds}^{avg} + \frac{\eta_{vs} * s_{vf}^{avg}}{\mathcal{R}_{dr}})$$

If we define:

$$\alpha = l_{seek}^{max} + l_{rot}^{max} \tag{16}$$

$$\beta = \mathcal{R}_{pl}^{avg} * (l_{ds}^{avg} + \frac{\eta_{vs} * s_{vf}^{avg}}{\mathcal{R}_{dr}}) \tag{17}$$

$$\gamma = l_{ds}^{avg} + \frac{\eta_{vs} * s_{vf}^{avg}}{\mathcal{R}_{dr}} \tag{18}$$

$$\delta = \eta_{vs} \tag{19}$$

then Equation (15), which represents the requirement of HDTV retrieval rates of all the subscribers, reduces to:

$$n * \alpha + k * n * \beta - n * \gamma \leq k * \delta \tag{20}$$

$$\Rightarrow \begin{array}{lcll} k & \geq & \frac{n(\alpha-\gamma)}{(\delta-n\beta)} & \text{if } \delta > n\beta \\ k & \leq & \frac{n(\alpha-\gamma)}{(n\beta-\delta)} & \text{if } \delta < n\beta \\ k & = & \infty & \text{if } \delta = n\beta \end{array} \tag{21}$$

[4]This policy is the time analog of the scaled placement policy presented for spatial merging in Section 2, but as we will see shortly, the algorithms to implement the policy are different and much more complex.

[5]Consider two number sequences $A = \{a_i | i \in [1,n]\}$ and $B = \{b_i | i \in [1,n]\}$. If the elements of A are sorted in the ascending order (i.e., $a_i \leq a_j$ if $i > j$), and the elements of B are sorted in the descending order (i.e., $b_i \geq b_j$ if $i > j$) , then

$$\sum_{i=1}^{n} a_i * b_i \leq n * \bar{a} * \bar{b}$$

where $\bar{a} = \frac{\sum_{i=1}^{n} a_i}{n}$ and $\bar{b} = \frac{\sum_{i=1}^{n} b_i}{n}$.

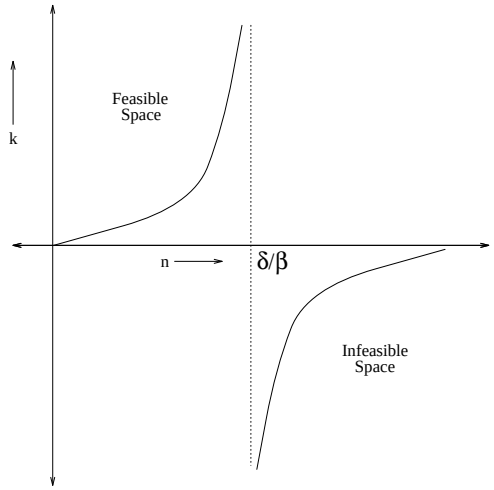

Figure 5: Variation of the number of blocks (k) per service round with respect to the number of requests (n)

Figure 5 shows the variation of k with n. The value of k obtained from Equation (21) will be positive (and hence, meaningful) if and only if $\delta > n\beta$, which yields the maximum number of simultaneous subscribers that can be serviced to be:

$$n^p_{max} \leq \lfloor \frac{\eta_{vs}}{\mathcal{R}^{avg}_{pl} * (l^{avg}_{ds} + \frac{\eta_{vs} * s^{avg}_{vf}}{\mathcal{R}_{dr}})} \rfloor \tag{22}$$

For a HDTV video stream retrieved at 30 frames/s and frame size 0.5 Mbits/frame on a disk of transfer bandwidth 1 Gbits/s, choosing $\eta_{vs} = 1$ frame/block and $l_{ds} = 1$ ms yields $n^p_{max} \leq 22$. Hence, the configuration proposed in Section 1 consisting of 120 such disks can support 2640 simultaneous subscriber requests, which is more than the 1000 required to make the configuration economically viable.

Given a value of the number of subscribers $n \leq n^p_{max}$, Equation (21) can be used to determine k, from which, the number of blocks of each subscriber retrieved during each service round can be obtained to be: $k_1 = k * \mathcal{R}^1_{pl}$, $k_2 = k * \mathcal{R}^2_{pl}$, ..., $k_n = k * \mathcal{R}^n_{pl}$. Furthermore, the values of k_i's thus derived can determine bounds on the buffer space requirement for continuous playback. Specifically, if the retrieval of a media strand is initiated from an independently decodable frame (such as, an intra-coded frame (I) in a MPEG encoded video strand), then since k_i blocks are retrieved during each service round, the buffer space requirement for each subscriber can be bounded by $2 * k_i$. However, initiating the playback starting from any other type of frame (such as, a predicted (P) or an interpolated (B) frame in a MPEG encoded video strand) necessitates reading ahead all the frames essential for its decoding, and hence, increases the buffer space requirement by at most the number of frames separating successive independently decodable frames (which, in the case of MPEG, is equal to the number of predicted (P) and interpolated (B) frames separating successive intra-coded (I) frames).

In the above analysis, we have assumed that a HDTV video strand is retrieved from disk in units of blocks, each containing an integral number of frames (since η_{vs} is assumed to be an integer). Since the size of a video frame (namely, s^i_{vf}) varies from frame to frame, the above formulation requires that media information be stored and retrieved from disk in variable sized blocks. However, in the case of disks with fixed size blocks, a disk block

may not contain an integral number of frames, and hence, disk block and frame boundaries may not coincide. In such a scenario, the requirement for servicing n subscribers simultaneously can be formulated in terms of bit rate requirement (instead of frame rate requirement, as in Equation (10)), and is given by:

$$n * (l_{seek}^{max} + l_{rot}^{max}) + \sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} \left(l_{ds}^{i} + \frac{D}{\mathcal{R}_{dr}} \right) \leq \min_{i \in [1,n]} \left(\frac{k_i * D}{\mathcal{R}_{br}^{i}} \right) \tag{23}$$

where D is the size of the disk block, and $\forall i \in [1, n]$, $\mathcal{R}_{br}^{i}$ denotes the bit rate requirement of the ith subscriber. Servicing optimal number of subscribers simultaneously requires that the number of blocks accessed during each round for each subscriber request be proportional to that subscriber's bit rate requirement. That is,

$$\forall i \in [1, n]: \quad k_i \propto \mathcal{R}_{br}^{i}$$

Thus, if k is the proportionality constant, then substituting $k_1 = k * \mathcal{R}_{br}^{1}$, $k_2 = k * \mathcal{R}_{br}^{2}$, ..., $k_n = k * \mathcal{R}_{br}^{n}$ in Equation (23), and repeating the analysis presented in this section, we can compute the number of blocks of each subscriber retrieved during each service round.

3.2 Optimality of Quality Proportional Multi-Subscriber Servicing

The QPMS algorithm can support a much larger number of subscribers compared to the round robin algorithm. The relative increase in the number of subscribers is given by:

$$\frac{n_{max}^{p}}{n_{max}^{c}} = \frac{\mathcal{R}_{pl}^{max}}{\mathcal{R}_{pl}^{avg}} \tag{24}$$

which can be significant if there is a large variation in the playback rates of the videos requested by subscribers. In fact, the QPMS algorithm can be shown to be optimal, that is, it can support the largest possible number of subscribers simultaneously:

Theorem 2 *The QPMS algorithm supports the optimal number of subscribers simultaneously.*

Proof : Without any loss of generality, let us assume that

$$\mathcal{R}_{pl}^{1} \geq \mathcal{R}_{pl}^{2} \geq \cdots \geq \mathcal{R}_{pl}^{n}$$

where,

$$\mathcal{R}_{pl}^{1} = a_2 * \mathcal{R}_{pl}^{2} = a_3 * \mathcal{R}_{pl}^{3} = \cdots = a_n * \mathcal{R}_{pl}^{n}$$

for the n strands requested by n subscribers.

In order to prove this theorem, we show that, given any number of subscribers n, if there is a set of $k_1', k_2', ..., k_n'$ satisfying the HDTV retrieval rate equation (Equation (13)), then $\exists k_1, k_2, ..., k_n$ that are proportional to their respective playback rates, i.e.,

$$k_1 = a_2 k_2 = \cdots = a_n k_n$$

and which satisfy the same Equation (13). In such a case, $k = \frac{k_i * a_i}{\mathcal{R}^1_{pl}}$ is guaranteed to satisfy the QPMS equation (Equation (20)), and is, hence, a solution that will be found by the QPMS algorithm.

In order to obtain the values of $k_1, k_2, ..., k_n$, the following procedure is used to increment the value of each k'_i, the end result of which is, $\forall i \in [1,n]: \; a_i * k_i = \max_{j \in [1,n]}(a_j * k'_j)$.

1. Let

$$a_{m_1} * k'_{m_1} = \min_{i \in [1,n]} a_i * k'_i$$

and

$$a_{m_2} * k'_{m_2} = \min_{i \in [1,n] \; and \; i \neq m} a_i * k'_i$$

Incrementing the value of k'_{m_1} to k_{m_1} such that $a_{m_1} * k_{m_1} = a_{m_2} * k'_{m_2}$ (i.e., setting $k_{m_1} = \frac{a_{m_2} * k'_{m_2}}{a_{m_1}}$) will make the number of blocks of HDTV strands m_1 and m_2 proportional to their respective playback rates. However, such an increase is permissible only if it does not cause the HDTV retrieval rate equation (Equation (13)) to be violated. Since $a_{m_1} * k'_{m_1} = \min_{i \in [1,n]} a_i * k'_i$ and $\mathcal{R}^i_{pl} = \frac{\mathcal{R}_{pl} 1}{a_i}$, Equation (13) reduces to:

$$\begin{aligned} n * (l^{max}_{seek} \quad + l^{max}_{rot}) + \textstyle\sum_{i=1}^{n} \sum_{j=1}^{k'_i - 1} (l^i_{ds} + \frac{\eta_{vs} * s_{vf}}{\mathcal{R}_{dr}}) \leq \\ \min_{i \in [1,n]}(k'_i * \frac{\eta_{vs}}{\mathcal{R}^i_{pl}}) = \frac{\eta_{vs}}{\mathcal{R}^1_{pl}} * \min_{i \in [1,n]}(k'_1, a_2 * k'_2, ..., a_n * k'_n) = \frac{a_{m_1} * k'_{m_1} * \eta_{vs}}{\mathcal{R}^1_{pl}} \end{aligned} \tag{25}$$

Since Equation (1) guarantees that

$$l^{avg}_{ds} + \frac{\eta_{vs} * s_{vf}}{\mathcal{R}_{dr}} \leq \frac{\eta_{vs}}{\mathcal{R}^{m_1}_{pl}} = \frac{a_{m_1} * \eta_{vs}}{\mathcal{R}^1_{pl}}$$

incrementing k'_{m_1} results in a greater increase in the RHS of Equation (25) than that in LHS. Hence, the value of k'_{m_1} can be safely increased to k_{m_1}. After the increase, the display durations of both m_1 and m_2 will be identical, yielding the RHS of Equation (25) as

$$\min_{i \in [1,n]}(k'_i * \frac{\eta_{vs}}{\mathcal{R}^i_{pl}}) = \frac{a_{m_1} * k_{m_1} * \eta_{vs}}{\mathcal{R}^1_{pl}} = \frac{a_{m_2} * k'_{m_2} * \eta_{vs}}{\mathcal{R}^1_{pl}}$$

If we set $\mathcal{M}_{prop}$ to denote the set of media streams whose k_i's become proportional to their respective playback rates, at this step, $\mathcal{M}_{prop}$ would become:

$$\mathcal{M}_{prop} = \{m_1, m_2\}$$

2. As long as there are requests that are not yet in $\mathcal{M}_{prop}$, i.e., $|\mathcal{M}_{prop}| < n$, determine the next request m not in $\mathcal{M}_{prop}$, whose $a_m * k'_m$ is the minimum:

$$a_m * k'_m = \min_{i \in [1,n] \; and \; i \notin \mathcal{M}_{prop}} a_i * k'_i$$

and, increase the values of $k_{m_1}, k_{m_2}, ...,$ of all the requests in $\mathcal{M}_{prop}$ such that:

$$\forall m_i \in \mathcal{M}_{prop} : \; k_{m_i} = \frac{a_m * k'_m}{a_{m_i}}$$

Again, such an increase is permissible only if it does not cause the HDTV Equation (13) to be violated. In order to show that it is indeed not violated, consider Equation (25) with the LHS terms belonging to requests within and without $\mathcal{M}_{prop}$ separated:

$$n*(l_{seek}^{max}+l_{rot}^{max})+\sum_{i\in\mathcal{M}_{prop}}\sum_{j=1}^{k_i-1}(l_{ds}^{i}+\frac{\eta_{vs}*s_{vf}}{\mathcal{R}_{dr}})+\sum_{i\notin\mathcal{M}_{prop}}\sum_{j=1}^{k'_i-1}(l_{ds}^{i}+\frac{\eta_{vs}*s_{vf}}{\mathcal{R}_{dr}}) \leq \frac{k_{m_i}*\eta_{vs}}{\mathcal{R}_{pl}^{i}} \quad (26)$$

Hence, increasing the number of blocks of each of the subset of requests in $\mathcal{M}_{prop}$ by the same proportion multiplies the RHS by a factor that exceeds 1, and only the second term of the LHS by that same factor. Hence, the increase in the RHS is more than that in the LHS, and the inequality is still maintained. Hence, the number of blocks of each of the subset of requests in $\mathcal{M}_{prop}$ can be safely increased so as to equal in proportion to that of request m. After the increase, m is added to $\mathcal{M}_{prop}$:

$$\mathcal{M}_{prop} = \mathcal{M}_{prop} \cup \{m\}$$

If there are no more requests outside $\mathcal{M}_{prop}$, that is, $|\mathcal{M}_{prop}| = n$, it implies that $k_1 = a_2 * k_2 = \cdots = a_n * k_n$, and the procedure can be terminated.

□

In the QPMS algorithm, the values of k_i obtained from a chosen value of k may be non-integral. Truncating or rounding off the real values so obtained may not guarantee that Equation (13) is satisfied, and hence may not guarantee that the retrieval of each of the requests proceeds at HDTV rates. We now present techniques to go from real values to integral values so as not to violate HDTV rate requirements.

3.3 QPMS with Integral Quanta

Display of HDTV strands proceeds in terms of quanta such as frames. Assuming each block to contain a display quantum, if k_i, the number of blocks retrieved for a request i in a service round, is not an integer, then retrieval of a fraction of a block cannot be used for display, causing the display to starve until the remaining fraction arrives, possibly in the next service round. Such scenarios can be avoided if the number of blocks $\{k_1, k_2, ..., k_n\}$ retrieved in a service round are all integers, techniques for deriving which we now elaborate, starting from the real values yielded by the QPMS algorithm.

Let the values of $\{k_1, k_2, ..., k_n\}$ yielded by the QPMS algorithm be:

$$\forall i \in [1, n] : \quad k_i = I_i + F_i$$

where I_i and F_i are the integer and the fractional parts, respectively, of k_i. If $I = \sum_{i=1}^{n} I_i$ and $F = \sum_{i=1}^{n} F_i$, then $(I + F)$ denotes the average number of blocks that need to be retrieved in each service round. In the technique that we present, the number of blocks transferred for a strand S_i during a service round toggles between the floor and the ceiling of k_i, so that on an average, the transfer rate for each request i is $I_i + F_i$ blocks/round. Specifically, for each round r, the HDTV server must determine the set $\mathcal{K}_j = \{k_1^r, k_2^r, \cdots k_n^r\}$ of the sequence of number of blocks of the n subscribers to be transferred during round r, where k_i^r can equal either I_i or $(I_i + 1)$. However, in doing so, both the service time and buffer space constraints, that would have been met had the transfer rate been $I_i + F_i$ strictly for every round, must continue to be satisfied:

- *HDTV rate constraint:* The cumulative slack time at the HDTV server, which is the sum of the differences between the RHS and the LHS of Equation (13) for each round, must be non-negative so as to ensure that none of the subscribers are starved during a service round. Denoting the time to access a block by $\tau = \frac{\eta_{vs} * s_{vf}}{R_{dr}}$, the inherent slack time ($\mathcal{S}_t^{in}$) that would have been available in each round, if exactly k_i blocks of strand i were transferred, is given by the difference between RHS and LHS in the HDTV rate Equation (13):

$$\mathcal{S}_t^{in} = \min_{i \in [1,n]} \left(\frac{\eta_{vs} * s_{vf}}{\mathcal{R}_{pl}^i}\right) - \left(n * \left(l_{seek}^{max} + l_{rot}^{max}\right) + \sum_{i=1}^{n} \sum_{j=1}^{k_i - 1} (l_{ds} + \tau)\right) \geq 0$$

  If, instead, the number of blocks transferred toggles between I_i and $I_i + 1$, the slack time during a round can be higher or lower, respectively. The cumulative slack time $\mathcal{S}_t(R)$ available during the Rth round is given by:

$$\mathcal{S}_t(R) = R * \mathcal{S}_t^{in} + \tau * \sum_{i=1}^{n} (R * k_i - \sum_{r=1}^{R} k_i^r)$$

  For HDTV rate constraint to be met, the cumulative slack time must never become negative, i.e., $\mathcal{S}_t(R) \geq 0$.

- *Buffer space constraint:* The slack buffer space at the HDTV server, which is the difference between the available buffer space and the used buffer space, must be non-negative.

  The transfer of $I_i + 1$ instead of k_i blocks during a service round can increase the buffer space requirements. If the needed additional buffering is unavailable, the toggling up of number of blocks transferred from I_i to $I_i + 1$ of one request must be matched by a toggling down from $I_j + 1$ to I_j of another request. If $\mathcal{S}_s = B - I$ is the slack buffer space available during each round, then the following buffer space constraint must be met:

$$\sum_{i=1}^{n} (k_i^r - I_i) \leq \mathcal{S}_s$$

  It can be shown that, the slack buffers $\mathcal{S}_s$ must at least equal $\lceil F \rceil$, yielding a value of at least $(I + \lceil F \rceil)$ for the total number of buffers B. If not, buffering may become insufficient to hold all the blocks transferred during a service round. (This is because, since we do not make any assumptions about the order of requests within a service round, the blocks transferred during one service round, which are $I + F$ in number on an average, are not assumed to be available for playback until the immediately following service round, and need to be buffered until then).

We now present an algorithm in which toggling of I_i to $(I_i + 1)$ for requests are dynamically staggered so as never to exceed available slack time and slack space in each service round.

During each round, either I_i or $(I_i + 1)$ blocks of subscriber i are retrieved. During rounds in which I_i blocks are retrieved, there must be sufficient accumulation of data to maintain continuity of HDTV display, and the accumulation is resumed during rounds in which $(I_i + 1)$ blocks are retrieved. Furthermore, an initial prefetching of blocks is also necessary to guarantee continuity during the first few rounds (since not all requests i can have $I_i + 1$ transferred during the first few rounds). The accumulation at the end of round R for subscriber i is the sum of differences between k_i^r and $(I_i + F_i)$ during the r rounds plus the prefetched number of blocks $\mathcal{P}_i$, and is given by:

$$\mathcal{D}_i(R) = \mathcal{P}_i + \sum_{r=1}^{R}(k_i^r - (I_i + F_i)) \tag{27}$$

Since every round consumes $(I_i + F_i)$ blocks of request i on an average, during a round R_i, if $\mathcal{D}_i(R_i) < F_i$, a shortage of blocks would occur during the next round; hence, round R_i is the *deadline* for accessing $(I_i + 1)$ blocks of request i. During each round, if there is sufficient slack time available to transfer extra blocks, requests are ordered with earliest deadline round first, and $(I_i + 1)$ blocks are transferred for each such request i until the exhaustion of the slack time.

A schedule generated by the above algorithm will satisfy HDTV rate constraints if there is sufficient slack time in the deadline round of each request to transfer its extra block. By Equation (27), the deadline round of request i is related to F_i by:

$$\mathcal{P}_i + \sum_{j=1}^{R_i}(k_i^j - (I_i + F_i)) < F_i$$

The deadline round will occur earliest if I_i (and not $I_i + 1$) blocks have been transferred in all the previous rounds, i.e., $\forall r \in [1, R_i] : k_i^r = I_i$, in which case, the above equation becomes:

$$\mathcal{P}_i - R_i * F_i < F_i$$

yielding the equation for deadline round as:

$$R_i = \frac{\mathcal{P}_i}{F_i} - 1 \tag{28}$$

In order for sufficient slack time to be available during round R_i to transfer subscriber i's extra block, the accumulated slack time due to request i must exceed the time to transfer an extra block. In each round in which I_i is transferred, a slack time of $F_i * \tau$ is accumulated towards request i, yielding an accumulated slack time of $R_i * F_i * \tau$ in R_i rounds. If the HDTV rate constraint is to be met, then after retrieving an extra block of request i, the net slack time should be non-negative. That is,

$$R_i * F_i * \tau - \tau > 0$$

$$\Rightarrow R_i > \frac{1}{F_i}$$

Substituting for R_i from Equation 28, we obtain the condition for satisfiable deadline to be:

$$\frac{\mathcal{P}_i}{F_i} - 1 > \frac{1}{F_i}$$

$$\Rightarrow \mathcal{P}_i > 1 + F_i \tag{29}$$

In the deadline round, the accumulation $\mathcal{D}_i$ decreases to F_i, but an extra block transferred restores $\mathcal{D}_i$ to back to at least $(1 + F_i)$, and this cycle repeats. Since, $F_i < 1$, $\mathcal{P}_i \geq 2$ is guaranteed to satisfy Equation (29), and hence, a prefetch of two or more blocks of every request will guarantee the availability of sufficient slack time in the deadline rounds of all requests, thereby meeting the HDTV rate constraint.

Since each round can potentially produce a slack time of $\tau * F$, the HDTV server can transfer the extra blocks of at least $\lfloor F \rfloor$ requests in the order of earliest occurring deadline first, and whenever sufficient slack time accumulates, transfer the extra blocks of $\lceil F \rceil$ requests. Such a policy allows the deadline requirements of the maximum number of requests to be satisfied as much in advance as possible, while at the same time limiting the maximum extra buffering needed during each round to $\lceil F \rceil$. Hence, the buffer space constraint is also met by the algorithm. As pointed out earlier, $(I + \lceil F \rceil)$ is the minimal buffer space requirement that cannot be avoided by any implementation of the proportional policy.

3.4 Dynamic Admission of New Subscribers

While servicing an existing set of n subscribers, if a HDTV server receives a new $(n + 1)$th subscriber, it must now decide whether to admit the new subscriber or not in the QPMS algorithm. If $n + 1 \leq n_{max}$ derived from Equation (22), the HDTV server can compute the new values of α, β, γ, and δ (see Section 3.1), and then compute k_{new} (from Equation (21)) necessary for satisfying $(n + 1)$ subscribers.

If $k_{new} = k_{old}$ (where, k_{old} is the value of k using which the HDTV server has been servicing the existing n subscribers), then the HDTV server can immediately admit the $(n + 1)$th subscriber. However, if $k_{new} \neq k_{old}$, then $k_{new} > k_{old}$ (see Figure 5), and the HDTV server has to begin transferring $k_i^{new} = k_{new} * \mathcal{R}_{pl}^i$ blocks of each of the earlier n requests, and of the new $(n + 1)$th request. During the first round that includes servicing of the $(n+1)$th subscriber, the number of blocks of subscriber i being transferred is k_i^{new}, whereas, the number of blocks available for display are those of the previous round, which is k_i^{old}. Since it may be the case that $k_i^{new} > k_i^{old}$, for each subscriber i, the time spent to transfer its k_i^{new} blocks will exceed the playback duration of its k_i^{old} blocks, leading to a violation of its requirement of HDTV rate display. In other words, Equation (20) guarantees HDTV retrieval rates are maintained only in steady state, and not during transitions.

In order to guarantee a smooth and transparent transition, we propose the following modification to Equation (20). Suppose the HDTV server makes a transition from k_i^{old} to k_i^{new} in steps of 1 before beginning to service the $(n + 1)$th request. When it performs a transition from k_i^{old} to $(k_i^{old} + 1)$, the time to transfer $(k_i^{old} + 1)$ blocks must not exceed the minimum playback duration of k_i^{old} blocks. Thus, if we use the time to transfer $(k_i + 1)$ blocks instead of k_i in the left hand side of Equation (13) but use k_i in the right hand side, and then solve for k_i, a

transparent transition from k_i^{old} to $(k_i^{old} + 1)$ is guaranteed. Specifically, this substitution changes Equation (20) to

$$n * \alpha + n * k * \beta \leq k * \delta \tag{30}$$

Furthermore, since $\delta \geq n\beta$,

$$n\alpha + nk\beta \leq k\delta \Rightarrow n\alpha + n(k+1)\beta \leq (k+1)\delta$$

Hence, for all $i \in [1, n]$, a transition from $k_i^{old} + 1$ to $k_i^{old} + 2$, $k_i^{old} + 2$ to $k_i^{old} + 3$, ..., $k_i^{new} - 1$ to k_i^{new} are also automatically guaranteed. Thus, using Equation (30) (instead of Equation (21)) to determine k (from which, the values of $k_1, k_2, ..., k_n$ can be obtained using $k_i = k * \mathcal{R}_{pl}^i$), and increasing it in steps of 1, yields a QPMS algorithm that guarantees *both* transient and steady state maintenance of HDTV retrieval rates.

4 Experience and Performance Evaluation

So far, we have presented algorithms and techniques for collocational storage of maximum number of HDTV videos on disk, and simultaneous servicing of maximum number of subscribers' requests. In order to experimentally evaluate their performance, we are implementing a prototype HDTV server at the UCSD Multimedia Laboratory. We have carried out preliminary performance simulations of merging and QPMS algorithms assuming a configuration consisting of an array of 120 disks, each with a data transfer bandwidth of 1 Gbits/s at the HDTV server. For our simulations, the HDTV data rate is assumed to be about 15 Mbits/s at a frame rate of 30 frames/s, which is what is yielded by terrestrial HDTV broadcasting systems such as, DigiCipher, DCS-HDTV, ADTV, and ATVA-P [3]. The granularity of storage (i.e., η_{vs}) is fixed at 1 frame/disk block. At a recording rate of 30 frames/s, the bound on the scattering parameter (l_{ds}) as evaluated by using Equation (1) comes out to be 16.66 ms. Therefore, the storage pattern (M, G) of a HDTV strand is given by (0.5 Mb, 16.66 Mb). Under these constraints, when the strands are stored independently (i.e., without merging), then the average storage efficiency is found to be about 3%. On the contrary, both on-line and off-line merging techniques yield a storage efficiency of about 95%, thereby providing an evidence of the significant improvements in storage utilization that can be expected due to merging.

Within this environment, assuming various compression models, we have evaluated the relative performance of policies that provide deterministic and statistical service guarantees for servicing multiple subscribers simultaneously. In order to perform this analysis, we characterized compression models by specifying: (1) the types of frames (and their respective sizes) generated by the compression technique, and (2) the relative frequency of occurrence for each type of frame. For instance, still picture compression techniques, such as JPEG, do not exploit the temporal redundancy present in motion video, and yield only one type of frames (namely, intra-coded frames). In contrast, MPEG, which achieves significant bit rate reduction by motion-compensated interpolation, yields three different types of frames, namely, intra-coded (I), predicted (P), and bidirectional interpolated (B) frames, each with a different relative frequency. Our analysis demonstrates that applying QPMS policy for providing statistical service guarantees to HDTV video streams encoded using JPEG or MPEG compression techniques

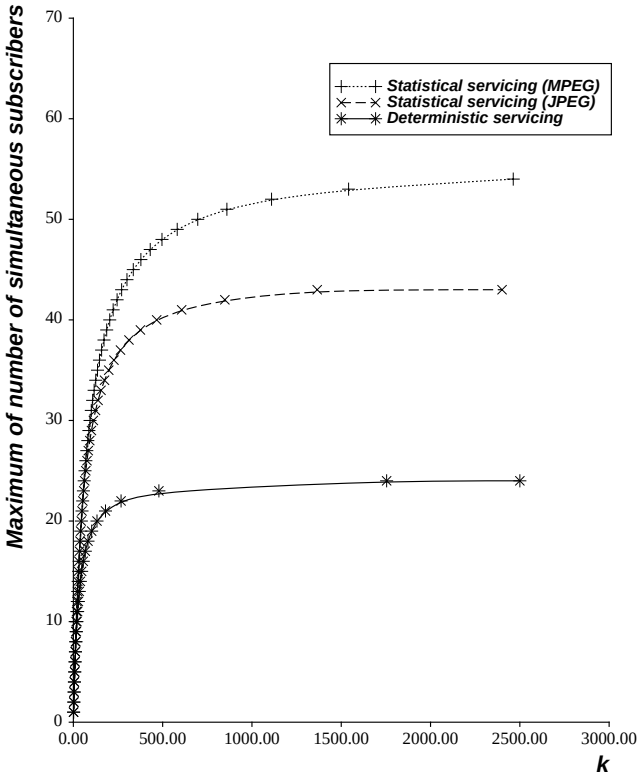

Figure 6: Variation in k with respect to the number of simultaneous subscribers (n) for providing deterministic or statistical service guarantee

yields smaller values of k (and hence, impose smaller buffer space requirement), and can service a larger number of subscribers simultaneously, as compared to its deterministic counterpart (see Figure 6).

Figure 7 shows the maximum number of simultaneous subscribers that can be supported (at various scattering parameters) by such a HDTV server using the QPMS algorithm. As expected, the number of subscribers increases with the decrease in the playback rate. The maximum number of subscribers reaches a highest value of 8000 for the QPMS algorithm, which is two orders of magnitude greater than 120 subscribers supported by straightforward multiplexing techniques such as one subscriber per disk head, thereby demonstrating the immense scalability of the QPMS algorithm.

Figure 8 illustrates that the gain in the maximum number of simultaneous subscribers in the QPMS as compared to the round-robin algorithm. Higher the asymmetry among the playback rates of the subscribers, greater is the advantage of employing the QPMS algorithm.

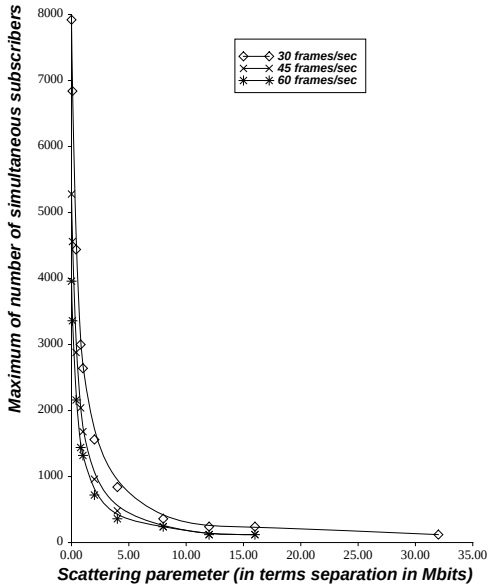

Figure 7: Variation of the maximum number of simultaneous subscribers with the scattering parameter, at various playback rates, in the QPMS algorithm

5 Relation to Previous Work

In the recent past, many research projects have investigated storage systems for still images and/or audio [1, 7]. Work by Mackay and Davenport [5], and Rangan and Swinehart [8] support video filing, but video is stored in an analog form on consumer electronic devices. The Matsushita's Real Time Storage System [6] has investigated some of the low level storage mechanisms for digital video. Anderson et al. [2], and Gammell and Christodoulakis [4] have described file system designs for supporting multiple audio channel playback, and have proposed techniques for providing hard performance guarantees. A model for the design of a file system for storing real-time video and audio streams individually on magnetic disks have been presented by Rangan and Vin [9]. A qualitative proposal

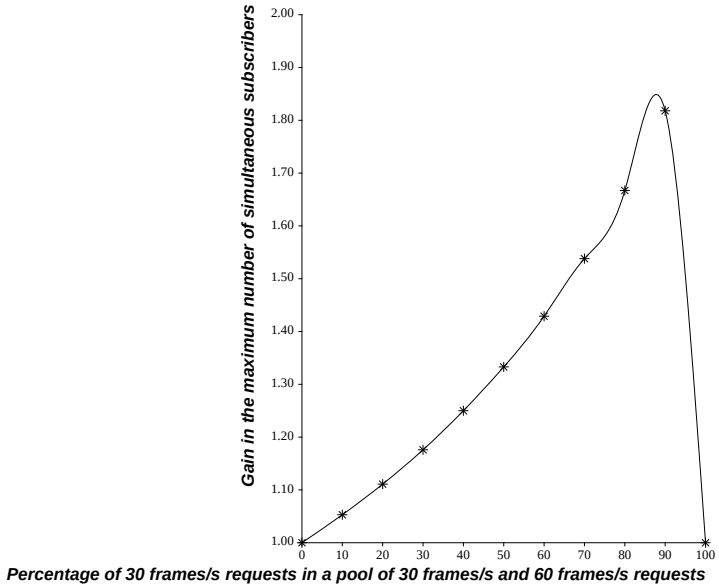

Figure 8: Increase in the maximum number of simultaneous subscribers in QPMS as compared to the round-robin algorithm

for a video-on-demand service is presented by Sincoskie in [11]. However, a quantitative study and optimizing algorithms for designing multi-user HDTV storage servers have not received much attention.

6 Concluding Remarks

We have presented techniques for designing HDTV-on-demand server that can satisfy a large number of subscribers simultaneously. Using a model that relates disk and device characteristics to the HDTV playback rate, storage patterns for HDTV video streams are obtained, and multiple streams each with its own storage pattern are merged so as to utilize disk space efficiently. We have proposed both an optimal off-line merging algorithm suitable for archival storage, and an on-line merging algorithm suitable for intermediate cache storage. In order to service multiple subscribers simultaneously, we have developed a Quality Proportional Multi-subscriber Servicing (QPMS) algorithm that retrieves video frames at a rate proportional on an average to the HDTV playback rates of requests. The algorithm uses a staggered toggling technique by which successive numbers of frames retrieved are fine tuned individually to achieve the servicing of an optimal number of subscribers simultaneously, without violating the HDTV rate requirements of any of the subscriber. The QPMS algorithm is also powerful enough to accommodate bounded availability of HDTV display buffers, and permits dynamic additions and deletions of subscriber requests in a transparent manner (i.e., without causing discontinuity in the retrieval of any of the existing subscribers). Its performance indicates that it is two orders of magnitude scalable compared to straightforward techniques such as servicing one subscriber per disk head and round robin servicing of subscribers.

In summary, our studies provide a quantitative demonstration of the technological feasibility and economic viability of HDTV-on-demand servers (that provide services similar to those of neighborhood videotape rental stores) on metropolitan area networks.

REFERENCES

[1] C. Abbott. Efficient Editing of Digital Sound on Disk. *Journal of Audio Engineering*, 32(6):394–402, June 1984.

[2] D. Anderson, Y. Osawa, and R. Govindan. A File System for Continuous Media. *ACM Transactions on Computer Systems*, 10(4):311–337, November 1992.

[3] G. Y. Beakley. Channel Coding for Digital HDTV Terrestrial Broadcasting. *IEEE Transaction on Broadcasting*, 37(4):137–140, December 1991.

[4] J. Gemmell and S. Christodoulakis. Principles of Delay Sensitive Multimedia Data Storage and Retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[5] W. E. Mackay and G. Davenport. Virtual Video Editing in Interactive Multimedia Applications. *Communications of the ACM*, 32(7):802–810, July 1989.

[6] Y. Mori. Multimedia Real-Time File System. Technical report, Matshushita Electric Industrial Co., February 1990.

[7] B.C. Ooi, A.D. Narasimhalu, K.Y. Wang, and I.F. Chang. Design of a Multi-Media File Server using Optical Disks for Office Applications. *IEEE Computer Society Office Automation Symposium, Gaithersburg, MD*, pages 157–163, April 1987.

[8] P. Venkat Rangan and D. C. Swinehart. Software Architecture for Integration of Video Services in the Etherphone Environment. *IEEE Journal on Selected Areas in Communication*, 9(9):1395–1404, December 1991.

[9] P. Venkat Rangan and Harrick M. Vin. Designing File Systems for Digital Video and Audio. In *Proceedings of the 13th Symposium on Operating Systems Principles (SOSP'91), Operating Systems Review, Vol. 25, No. 5*, pages 81–94, October 1991.

[10] P. Venkat Rangan, Harrick M. Vin, and Srinivas Ramanathan. Designing an On-Demand Multimedia Service. *IEEE Communications Magazine*, 30(7):56–65, July 1992.

[11] W. D. Sincoskie. System Architecture for a Large Scale Video on Demand Service. *Computer Networks and ISDN Systems, North-Holland*, 22:155–162, 1991.