Efficient Schemes for Broadcasting Popular Videos*

Lixin Gao
Department of Computer Science
Smith College
Northampton, Mass. 01060, USA
gao@cs.smith.edu

Jim Kurose
Don Towsley
Department of Computer Science
University of Massachusetts
Amherst, Mass. 01003, USA
{kurose, towsley}@cs.umass.edu

Abstract

We provide a formal framework for studying broadcasting schemes and design a family of schemes for broadcasting popular videos, called the Greedy Disk-conserving Broadcasting (GDB) scheme. We analyze the resource requirements for GDB, i.e., the number of server broadcast channels, the client storage space, and the client I/O bandwidth required by GDB. The analysis shows that GDB exhibits a tradeoff between any two of the three resources. We compare our scheme with a recently proposed broadcasting scheme, Skyscraper Broadcasting (SB). With GDB, we can reduce the client storage space by as much as 50% or the number of server channels by as much as 30% at the cost of a small additional increase in the amount of client I/O bandwidth. If we require the client I/O bandwidth of GDB to be identical to that of SB, GDB needs only 70% of the client storage space required by SB or one less server channel than SB does. In addition, we show that with small client I/O bandwidth, the resource requirements of GDB are close to the minimum achievable by any disk-conserving broadcasting scheme.

1 Introduction

Video is one of the most important communication media in all aspects of our lives. On average, each household has 1.4 television sets in the U.S.[3, Page216]. We turn to televisions to be informed, entertained, educated, or even to do business (e.g., shopping or video conferencing in offices). We wish to be able to make content available instantly at the touch of our fingertips. It was shown in [6, 7] that 80% of our demand is for a few (10 to 20) very popular videos. This drives future content providers to deliver popular videos to a large number of subscribers with the smallest latency possible. These popular videos can be "hot" news (such as the Oklahoma City bombing trial or Mars Pathfinder coverage), popular sport events (such as a U.S. Open final), taped lectures (during final exam period), or recently released movies.

In a VOD system, a server delivers videos to clients via a high speed network. Since video is an isochronous medium, a video server has to reserve a sufficient amount of network and I/O bandwidth for the video stream before committing to a client's request. We refer to the resources required to deliver one video stream while guaranteeing a client's continuous playback as a *server channel*. The simpliest scheme for scheduling server channels is to dedicate a channel to each client. However, this results in a tremendous demand for server channels in VOD systems. Therefore, two techniques for sharing a channel by more than one client have been proposed.

- **Sharing by batching requests for the same video together.** Here, a client makes requests for a video and waits until that video is eventually broadcast on a channel. When a channel becomes available, the server selects a batch of pending requests for some video according to some scheduling policy. While batching reduces the demand for server channels, it does so at the cost of introducing service latency.
- **Sharing by broadcasting a video via multiple dedicated channels.** Here, the client

* This work was supported in part by the National Science Foundation under Grant No. NCR-9729084. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

prefetches data from one of several channels for later playback. Prefetching is done in such a way as to ensure the continuous playback of the whole video. Broadcasting schemes take advantage of resources (e.g., disk) at the client end, and guarantee a service latency independent of the number of requests.

In this paper, we provide a formal framework for studying broadcasting schemes and design a family of schemes for broadcasting popular videos, called the *Greedy Disk-conserving Broadcasting* (GDB) scheme. The main idea of GDB is as follows. Each video is partitioned into segments. Each segment is broadcast periodically via a dedicated channel. We select segment sizes so as to ensure that the client can receive a segment before the playback time of the segment. Since a client can prefetch a segment during the playback of an earlier segment, the data closer to the end of the video can be broadcast less often than the data closer to the beginning of the video. Therefore, we partition the video into segments of monotonically increasing size. Since the maximum service latency (the time from the client request of the video until the start of its broadcast) is the length of the first segment, the later segments are greedily selected to be as large as possible so as to minimize the size of the first segment. Furthermore, to conserve the client disk space, a client receives a data segment at the latest possible time that ensures the continuous playback of the video.

We systematically analyze the resource requirements for GDB. In particular, we derive the number of server broadcast channels, the client storage space, and the client I/O bandwidth required by GDB. Our analysis shows that GDB exhibits a tradeoff between any two of the three resources.

We compare our scheme with the most recently proposed broadcasting scheme, *Skyscraper Broadcasting* (SB). We rigorously prove that GDB outperforms SB in terms of the resource requirements. Our simulation shows that with GDB, we can reduce the client storage space by as much as 50%, or the number of server channels by as much as 30%, at the cost of small additional client I/O bandwidth. If we require the client I/O bandwidth of GDB to be identical to that of SB, GDB needs only 70% of the client storage space required by SB or one less server channel than SB does. In addition, our data show that with small client I/O bandwidth, the resource requirements of GDB are close to the minimum achievable by *any* disk-conserving broadcasting scheme. Finally, we prove that the number of server broadcast channels required by GDB is close to the optimal achievable by any broadcast scheme.

The rest of this paper is organized as follows. In the remainder of this section, we review related work. Section 2 gives an overview of the problem. In section 3, we present and analyze disk-conserving broadcasting schemes. Sections 4, 5 and 6 propose and analyze GDB and compare it with SB. Finally, in Section 7, we conclude with ending remarks.

1.1 Related Work

The simplest scheduling scheme for a video server is to allocate a channel to each client. However, this scheme can quickly exhaust the most scarce resource in the server's system: the server network bandwidth. To avoid this, Dan et al. proposed to allocate on-demand channels for less popular videos and dedicated channels for popular videos [6, 7, 8]. The server uses on-demand channels to satisfy requests for less popular videos according to some batch scheduling policy, and uses dedicated channels to broadcast popular videos to clients. Both broadcasting and batch scheduling schemes allow a channel to be shared by more than one client as follows.

In a batch scheduling scheme, a group of requests that arrive close in time are batched together and when a channel becomes available, the server selects a batch according to some scheduling policy, satisfying the requests with the channel. Some efficient batch scheduling policies are studied in [6, 7, 2].

In a broadcasting scheme, the server broadcasts a video via multiple dedicated channels and each client follows a reception schedule to receive data from appropriate channels so as to continuously playback the whole video. The simplest broadcasting scheme is the EB scheme that divides the video into equal-length segments[6]. The EB scheme guarantees a worst case service latency equal to the length of one segment, independent of the number of of requests, since a client waits at most the length of one segment for the playback to start. Furthermore, the number of dedicated channels required for a video is inversely proportional to the guaranteed service latency.

To decrease the number of required channels, Viswanathan and Imielinski proposed an ingenious broadcasting scheme, called Pyramid Broadcasting (PB) [11]. The main idea of the PB scheme is to divide a video into segments of geometrically increasing size. Each segment is broadcast periodically via a dedicated channel that has a bandwidth greater than the playback rate. Each client saves the next segment of data while playing the current segment of video data. Since the service latency is proportional to the length of the first segment, the PB scheme ensures an exponentially decreasing latency with an increase in the number of dedicated channels. The drawback of this scheme is that a client needs to have sufficient disk space to store more than 70% of one video file.

To address the problem of the disk space requirements, Aggarwal, Wolf and Yu devised a scheme called Permutation-Based Pyramid Broadcasting (PPB) [1]. The PPB scheme divides a video object the same way as the PB scheme and multiplexes several channels among one segment of video data. The similar partition strategy limits the disk space reduction and typically requires a client disk capable of storing at least 50% of a video's data.

Recently, Kua and Sheu proposed a broadcasting scheme [10], called Skyscraper Broadcasting (SB), that further reduces the client disk space requirement. The SB scheme uses a novel data partition strategy and exhibits a tradeoff between the client storage space and the server network bandwidth. However, these schemes have been proposed in an ad hoc manner, and there is no study of the tradeoff between the client I/O bandwidth and the server network bandwidth.

2 Overview of the Problem

For simplicity of exposition, we consider the problem of broadcasting a single video, since we can easily extend the scheme to multiple videos with a proportional increase of the number of the server channels. Consider Figure 1, which depicts a video server broadcasting architecture. The server broadcasts videos via dedicated network channels according to some *broadcast schedule*. The *number of server channels* is the total number of channels required for the dedicated network channels. The network uses a multicast communication facility to transmit the video streams to clients so that a client can select which channels it wishes to receive from [5, 9].

Each client contains a set-top box, a disk, and a display monitor. A client is connected to the network with a set-top box. The set-top box selects several network channels to receive video data from according to a *reception schedule*. The reception schedule indicates which channels to select and when, so as to ensure the continuous playback of the video. In this paper, we consider only *non-workahead reception schedules*, those which require that a client first request a video before it can receive any data for that video. The received video data are either stored on the disk or sent to the display monitor for immediate playback. The display monitor can either retrieve stored data from the disk or receive data from a channel directly. The *client storage space* is the maximum disk space required throughout the client playback period. The *client I/O bandwidth* is the maximum bandwidth required to store and retrieve data from the disk at any time. The *service latency* is the maximum amount of time that a client has to wait to start the playback once it requests a video. We use the following notations throughout the paper.

B: the server bandwidth dedicated to the video

b: the video display rate at the client

K: the number of dedicated server channels

D: the length of the video in seconds

l: the latency requirement

We focus here on continuous playback starting from the beginning of the video, deferring issues introduced by providing interactivity (such as fast-forward, rewind, and pause) to a future study. Our goal is to design broadcasting schemes that guarantee a service latency l and effectively utilize all three resources: the number of dedicated server channels, the client disk space and the client I/O bandwidth.

3 Disk-Conserving Broadcasting Schemes

All proposed schemes in this paper falls into the class of *disk-conserving broadcasting schemes*. The main idea behind the disk-conserving broadcasting scheme is to conserve the client disk space by letting a client receive data as late as possible. The three components of a broadcasting scheme are 1) data partition strategy, 2) broadcast schedule, and 3) reception schedule. We formally describe each of these in turn for the disk-conserving broadcasting scheme.

Partition strategy: The dedicated bandwidth B is divided into $K = \lfloor B/b \rfloor$ logical channels of b Mb/s each. A D-second video is partitioned into K segments $T_1, T_2, ..., T_K$ according to a partition function $f(n)$. The partition function $f(n)$ is an integer function (i.e., $f(n)$ is an integer) that defines segment T_i as follows. We let one time unit be $D/\sum_{n=1}^{K} f(n)$ throughout this paper for ease of the discussion. Segment T_n contains $f(n)$ time units of video data, and its data starts at the $\sum_{m=1}^{n-1} f(m)$ th time unit of the video and ends at the $\sum_{m=1}^{n} f(m)$ th time unit of the video. As we will see later, a partition function uniquely defines a disk-conserving broadcasting scheme. Therefore, the selection of the partition function is crucial for determining the resource requirements of the scheme.

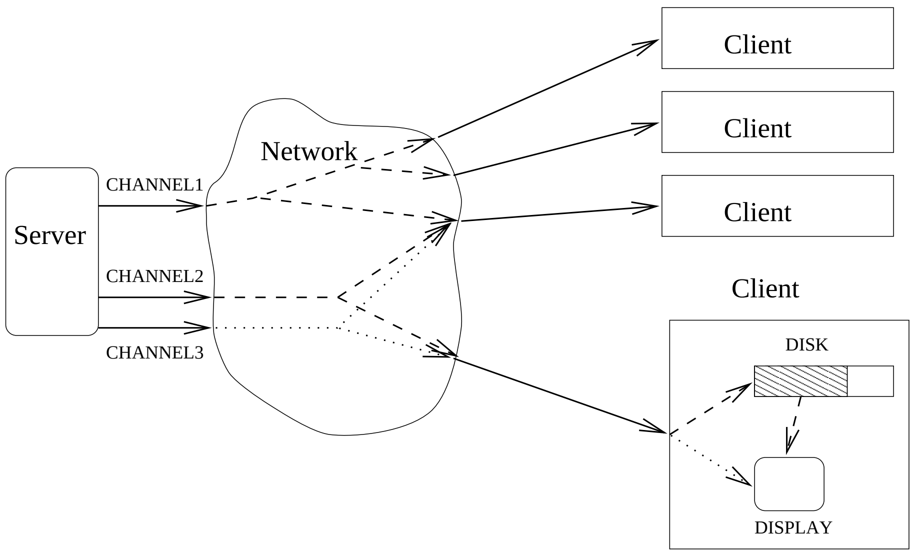

Figure 1: *Overview of the problem setting*

Broadcast schedule: The server broadcasts each segment periodically via a channel of bandwidth b starting at time 0. In other words, T_n is broadcast periodically starting at time $0, f(n), 2f(n), ...$ via channel n. For example, the τ th time unit data of T_n is broadcast at time $\tau, \tau + f(n), \tau + 2f(n), ...$ via channel n.

Reception schedule: a client arriving at time t waits until the beginning of the next broadcast cycle of T_1, at which point it can receive data from T_1 and begin playout immediately. Any other segment is received (either saved in the disk or displayed immediately) during the latest broadcast cycle prior to its playback time. To be more precise, we define the following notations.

$s(j, t)$: starting time of T_j's reception given that the client arrives at time t.
$e(j, t)$: finish time of T_j's reception given that the client arrives at time t.
$p(j, t)$: finish time of T_j's playback given that the client arrives at time t.

The first time that T_1 is broadcast after time t is

$$p(0, t) = s(1, t) = \lceil t/f(1) \rceil f(1)$$

Since the client plays out the video continuously, the client starts to play out T_j at time

$$p(j-1, t) = p(0, t) + \sum_{n=1}^{j-1} f(n) \qquad (1)$$

Therefore, the client must begin to receive segment T_j at time

$$s(j, t) = \left\lfloor \left(p(0, t) + \sum_{n=1}^{j-1} f(n) \right) / f(j) \right\rfloor f(j) \qquad (2)$$

and complete its reception at time

$$\begin{aligned} & e(j, t) \\ &= s(j, t) + f(j) \\ &= \left\lfloor \left(p(0, t) + \sum_{n=1}^{j-1} f(n) \right) / f(j) \right\rfloor f(j) + f(j) \qquad (3) \end{aligned}$$

under a disk-conserving broadcast scheme with partition function $f(n)$. Note that a disk-conserving broadcasting scheme ensures that

$$s(j, t) > p(j-1, t) - f(j) \qquad (4)$$

This property of the disk-conserving broadcast scheme is the key in determining its client I/O bandwidth and client storage space requirement as shown in Section 3.2 and 3.3.

To ensure that our reception strategy is non-workahead, we consider only the partition function $f(n)$ that satisfies

$$\sum_{n=1}^{j-1} f(n) \geq f(j) - 1 \qquad (5)$$

for any $j > 1$. It is easy to verify that this is the necessary and sufficient condition for $s(j, t) \geq t$ for any j and t, i.e.,

$$\left\lfloor \left(p(0,t) + \sum_{n=1}^{j-1} f(n) \right) / f(j) \right\rfloor f(j) \geq p(0,t) \geq t$$

for any t and j.

3.1 The Number of Server Channels

Now, we analyze the number of server channels required by a disk-conserving scheme. Since a client waits at most the length of the first segment to start the playback, the service latency guaranteed by a disk-conserving broadcasting scheme with partition function $f(n)$ is the length of the first segment, i.e., $f(1)$ time units or $f(1)D/\sum_{n=1}^{K} f(n)$ seconds. In order to guarantee a service latency l, the server must dedicate K channels to the video, where K is the smallest number that satisfies $f(1)D/\sum_{n=1}^{K} f(n) \leq l$. Therefore, we have the following theorem.

Theorem 3.1 *A disk-conserving broadcasting scheme with partition function $f(n)$ requires K channels to guarantee a service latency l, where K is the smallest number that satisfies $f(1)D/\sum_{n=1}^{K} f(n) \leq l$.*

Therefore, the "growth" of the partition function completely determines the required number of server channels. We introduce the following relation between two partition functions, $f(n)$ and $g(n)$. $f(n) \succ g(n)$ iff $\sum_{n=1}^{j} f(n)/f(1) > \sum_{n=1}^{j} g(n)/g(1)$ for all j. Let f_A and f_B denote the partition function of schemes A and B respectively. We have the following corollary.

Corollary 1 *Given two disk-conserving broadcasting schemes A and B. If $f_A \succ f_B$, then Scheme A needs no more server channels than Scheme B does for guaranteeing the same service latency.*

3.2 The Client I/O Bandwidth

We now derive a sufficient condition to ensure that the client I/O bandwidth does not exceed ib. We will see that this I/O bandwidth requirement of ib results from the client's reception of at most $i-1$ channels (each of bandwidth b) and the need to play out a stored video segment (also having a bandwidth of b).

Theorem 3.2 *The disk-conserving broadcasting scheme using partition function $f(n)$ requires a client I/O bandwidth of at most ib if $f(j) \leq \sum_{m=j-i+2}^{j-1} f(m) + \gcd(f(j), f(j-i+1))$ for all $j > i$.*

The main idea of the proof is as follows. During the playback period of any segment T_n, only segments $T_n, T_{n+1}, \ldots, T_{n+i-1}$ might be received. In fact, segment T_{n+i} is received after the playback period of segment T_n. This is because we can bound the starting reception time of T_{n+i} by Equation (4) as follows.

$$s(n+i,t) \geq p(n+i-1,t) - f(n+i) + 1$$

Figure 2 shows the earliest starting reception time of T_{n+i}. By the same reasoning, for any j, segment T_{n+i+j} is received after the playback period of T_{n+j}, thus after the playback period of T_n. Furthermore, T_n and T_{n+i-1}'s reception periods do not overlap. If they did, they would overlap for at least $\gcd(f(n), f(n+i-1))$ time units because T_n is broadcast every $f(n)$ time units and T_{n+i-1} is broadcast every $f(n+i-1)$ time units. From Figure 2, we see that the length of their overlapping period is at most $\gcd(f(n), f(n+i-1)) - 1$ time units. Therefore, at most $i-1$ segments are received simultaneously. We prove the theorem formally as follows.

First, we identify relationships between the reception and playback times for two different segments.

Lemma 3.1 *Under any disk-conserving broadcasting scheme using partition function $f(n)$, a client arriving at time t has two properties. For any $m < n$,*

(a) *if $f(n) \leq \sum_{j=m+1}^{n-1} f(j)$, then the starting reception time of T_n follows the time at which the playback of T_m completes, i.e., $s(n,t) > p(m,t)$,*

(b) *if $f(n) \leq \sum_{j=m+1}^{n-1} f(j) + \gcd(f(n), f(m))$, then T_n's reception period does not overlap T_m's, i.e., $s(n,t) \geq e(m,t)$.*

Proof : Suppose that $f(n) \leq \sum_{j=m}^{n-1} f(j) + x$, we have

$$\begin{aligned}
&s(n,t) \qquad (6)\\
&= \left\lfloor \left(p(0,t) + \sum_{j=1}^{n-1} f(j) \right) / f(n) \right\rfloor f(n) \\
&\qquad \text{(from Equation (2))}\\
&= \left\lfloor \left(p(0,t) + \sum_{j=1}^{m} f(j) + \sum_{j=m+1}^{n-1} f(j) \right) / f(n) \right\rfloor f(n) \\
&\geq \left\lfloor \left(p(0,t) + \sum_{j=1}^{m} f(j) + f(n) - x \right) / f(n) \right\rfloor f(n) \\
&= \left\lfloor \left(p(0,t) + \sum_{j=1}^{m} f(j) - x \right) / f(n) + 1 \right\rfloor f(n)
\end{aligned}$$

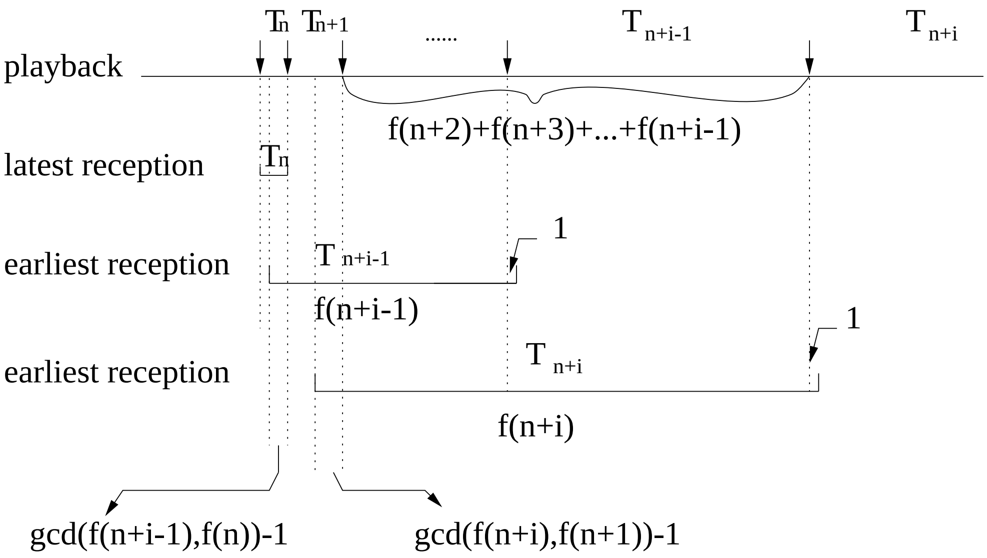

Figure 2: *The reception periods of* T_{n+i-1} *and* T_{n+i}

$$\geq \left\lceil \left(p(0,t) + \sum_{j=1}^{m} f(j) - x \right) / f(n) \right\rceil f(n) \quad (7)$$

$$\geq p(0,t) + \sum_{j=1}^{m} f(j) - x \quad (8)$$

$$= p(m,t) - x$$

Furthermore, the equality in inequality (7) is true only when $(p(0,t) + \sum_{j=1}^{m} f(j) - x) \bmod f(n)$ is not equal to zero. However, the equality in inequality (8) is true only when $(p(0,t) + \sum_{j=1}^{m} f(j) - x) \bmod f(n)$ is zero. Therefore, we have $s(n,t) > p(m,t) - x$.

Part (a) follows from the case that $x = 0$. Now, we prove part (b). Since $f(n) \leq \sum_{j=m+1}^{n-1} f(j) + \gcd(f(n), f(m))$, we have $s(n,t) > p(m,t) - \gcd(f(n), f(m))$. Since $p(m,t)$ is the finish playback time of T_m and $e(m,t)$ is the finish reception time of T_m, we know that $p(m,t) \geq e(m,t)$. Therefore, $s(n,t) > e(m,t) - \gcd(f(n), f(m))$. Since both $e(m,t) \bmod f(m) = 0$ and $s(n,t) \bmod f(n) = 0$ by the reception schedule, we have that both $e(m,t) \bmod \gcd(f(n), f(m)) = 0$ and $s(n,t) \bmod \gcd(f(n), f(m)) = 0$. Therefore, $s(n,t) \geq e(m,t)$. ∎

Proof of Theorem 3.2: It is obvious that the client needs a bandwidth of at most b for retrieving a stored segment from the disk for playout at the monitor. We thus prove the theorem by arguing that at most $i - 1$ segments are saved simultaneously at any playback time of a client.

During T_1's playback period, only $T_1, T_2, T_3, ...T_i$ can be received. This is because, for any segment T_j, where $j > i$, we know $f(j) \leq \sum_{m=j-i+2}^{j-1} f(m) + \gcd(f(j), f(j-i+1)) \leq \sum_{m=j-i+1}^{j-1} f(m)$. According to Lemma 3.1(a), $s(j,t) > p(j-i,t)$. Since $j > i$, we have $p(j-i,t) > p(1,t)$. We conclude that $s(j,t) > p(1,t)$, i.e., T_j is received after T_1's finish playback time. Therefore, only $T_1, T_2, T_3, ...T_i$ can be received during T_1's playback period. Since we never save segment T_1 to the disk, at most $(i-1)$ segments are saved during this period.

Now, consider T_n's playback period for any $n > 1$. We claim that no segment other than $T_n, T_{n+1}, T_{n+2}, ..., T_{n+i-1}$ are received during this period. We prove the claim as follows. Obviously, for any $j < n$, T_j is received before T_n's playback period. For any $j > n + i - 1$, we have $f(j) \leq \sum_{m=j-i+2}^{j-1} f(m) + \gcd(f(j), f(j-i+1)) \leq \sum_{m=j-i+1}^{j-1} f(m) \leq \sum_{m=n+1}^{j-1} f(m)$. From Lemma 3.1(a), we conclude that T_j is received after T_n's finish playback time. Furthermore, we see that T_{n+i-1}'s reception period does not overlap that of T_n from Lemma 3.1(b). Therefore, at most $i-1$ segments are received simultaneously at any time during T_n's playback period. ∎

3.3 The Client Storage Requirement

We analyze the client storage requirement of a disk-conserving broadcasting scheme as follows. For a client arriving at time t, the earliest time that the client receives T_j is $p(j-1,t) - f(j) + 1$ according to

Equation (4). Therefore, we can bound the total storage space required for segment T_j at any time. Figure 3 shows the maximum time units of data saved for each segment at the xth time unit of T_n's playback period, i.e., at time $p(n-1,t)+x$. Note that the figure shows only the case that $x > 1$, and that the client receives at most $f(n) - x$ time units of data from T_n because the client saves at most $f(n)$ time units of data and consumes x time units of data from T_n. We can derive a similar bound for the case that $x \leq 1$. Summing the storage space required for all segments, we conclude that the total amount of data saved is no greater than $f(m) - 1$ time units of data. Note that T_m is the last segment whose data is received before time $p(n-1,t)+x$. Therefore, we derive the following bound for the amount of client storage required.

Theorem 3.3
A disk-conserving broadcasting scheme with partition function $f(n)$ requires a client storage space that can store $(\max_{1\leq n\leq K} f(n) - 1)$ time units of data, i.e., $(\max_{1\leq n\leq K} f(n) - 1)Db/\sum_{n=1}^{K} f(n)$ data.

First, we state a lemma that will be useful in analyzing of the storage requirement.

Lemma 3.2 *For any $x_1, x_2, ..., x_n \geq 0$, $x_1 + \sum_{j=2}^{n}[x_j - \sum_{m=1}^{j-1} x_m]^+ \leq \max_{1\leq j\leq n} x_j$, where $[x]^+ \stackrel{\text{def}}{=} \max\{x, 0\}$.*

Proof : We prove the lemma by induction on n. It is obviously true when $n = 1$. Now, consider two cases.
Case 1. $x_n > \sum_{m=1}^{n-1} x_m$.
We have

$$
\begin{aligned}
& x_1 + \sum_{j=2}^{n}[x_j - \sum_{m=1}^{j-1} x_m]^+ \\
\leq\ & x_1 + \sum_{j=2}^{n-1}[x_j - \sum_{m=1}^{j-1} x_m]^+ + (x_n - \sum_{m=1}^{n-1} x_m) \\
\leq\ & \max_{1\leq j\leq n-1} x_j + (x_n - \sum_{m=1}^{n-1} x_m) \\
\leq\ & x_n \\
\leq\ & \max_{1\leq j\leq n} x_j
\end{aligned}
$$

by the induction hypothesis.
Case 2. $f(n) \leq \sum_{m=1}^{n-1} f(m)$.
We have

$$
\begin{aligned}
& x_1 + \sum_{j=2}^{n}[x_j - \sum_{m=1}^{j-1} x_m]^+ \\
\leq\ & x_1 + \sum_{j=2}^{n-1}[x_j - \sum_{m=1}^{j-1} x_m]^+ \\
\leq\ & \max_{1\leq j\leq n-1} x_j \\
\leq\ & \max_{1\leq j\leq n} x_j
\end{aligned}
$$

by the induction hypothesis. ∎

Proof of Theorem 3.3: We prove the theorem by analyzing the storage space needed during T_n's playback period. Let t denote the arrival time of the client. Consider the x th time unit during T_n's playback, i.e., time $p(n-1,t)+x$. First, we give a bound on the total data saved for segment T_n. If $x \leq 1$, the client saves at most $f(n) - 1$ time units of data from segment T_n. If $x > 1$, the client saves at most $f(n) - x$ time units of data from segment T_n, since $f(n)$ is the maximum time units of data saved and x is the consumed data. Therefore, the client saves at most $f(n) - 1 - [x-1]^+$ time units of data. Now we derive a bound on the storage space requirement of segment T_{n+i} at time $p(n-1,t) + x$ for any $1 \leq i \leq K - n$. We know that the client saves at most

$$
\begin{aligned}
& [p(n-1,t) + x - s(n+i,t)]^+ \\
& \quad \leq\ [p(n-1,t) + x - (p(n+i-1,t) - f(n+i) - 1)]^+ \\
& \qquad\qquad \text{(from inequality (4))} \\
& \quad =\ [f(n+i) - 1 - \sum_{j=n+1}^{n+i-1} f(j) - (f(n) - x)]^+
\end{aligned}
$$

time units of data from segment T_{n+i} at time $p(n-1,t) + x$. In other words, the client saves at most $[f(n+1) - 1 - (f(n) - x)]^+$ time units of data from segment T_{n+1} at time $p(n-1,t)+x$. The client saves at most $[f(n+2) - 1 - f(n+1) - (f(n) - t)]^+$ time units of data from segment T_{n+2} at time $p(n-1,t)+x$. .... and so on. The client saves at most $[f(K) - 1 - \sum_{j=n+1}^{K-1} f(j) - (f(n) - t)]^+$ time units of data from segment T_K at time $p(n-1,t)+x$. Therefore, a client saves at most

$$
\begin{aligned}
& f(n) - 1 - [x-1]^+ \\
+\ & [f(n+1) - 1 - (f(n) - x)]^+ \\
+\ & [f(n+2) - 1 - f(n+1) - (f(n) - x)]^+ \\
+\ & \ldots \\
+\ & [f(K) - 1 - \sum_{j=n+1}^{K-1} f(j) - (f(n) - x)]^+ \\
\leq\ & f(n) - 1 - [x-1]^+ - (f(n) - x) \\
+\ & f(n) - x \\
+\ & [f(n+1) - 1 - (f(n) - x)]^+
\end{aligned}
$$

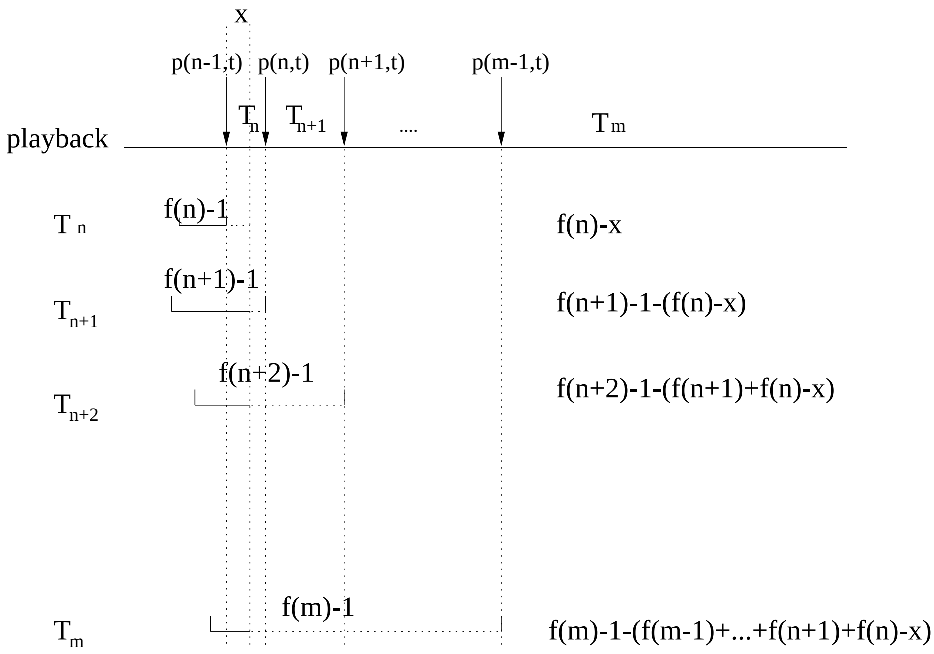

Figure 3: *The maximum time units of data saved at* $p(n-1,t)+x$

$$
\begin{aligned}
&+ \quad [f(n+2)-1-f(n+1)-(f(n)-x)]^+ \\
&+ \quad \ldots \\
&+ \quad [f(K)-1-\sum_{j=n+1}^{K-1} f(j)-(f(n)-x)]^+ \\
&\leq \quad f(n)-1-[x-1]^+-(f(n)-x) \\
&+ \quad \max\{\max_{n+1\leq j\leq K} f(j)-1,(f(n)-x)\} \qquad (9) \\
&\leq \quad \max_{n\leq j\leq K} f(j)-1 \\
&\leq \quad \max_{1\leq j\leq K} f(j)-1
\end{aligned}
$$

time units of data. Inequality (9) is derived from Lemma 3.2 by having $x_1 = f(n)-x \geq 0$ and $x_i = f(n+i)-1 \geq 0$ for $i > 1$. ∎

4 Greedy Disk-Conserving Broadcasting Scheme GDB(i)

In this section we introduce a greedy disk-conserving broadcast scheme, GDB(i), for a given client I/O bandwidth of ib where i is an integer and $i > 3$. GDB(i) selects its partition function "greedily" so as to minimize the number of server channels needed to guarantee a given service latency. Corollary 1 tells us to select the partition function of GDB(i) to "grow" as fast as possible starting at $f(1)=1$. To constrain the client bandwidth to i channels, we greedily choose $f(n)$ to be the largest number that satisfies the sufficient condition given in Theorem 3.2 and the non-workahead requirement given in Equation (5). Specifically, for $n \leq i$, $f(n)$ is the largest number that satisfies $f(n) \leq \sum_{j=1}^{n-1} f(j)+1$. For the case that $n > i$, $f(n)$ is selected to be the maximum number that satisfies $f(n) \leq \sum_{m=n-i+2}^{n-1} f(m)+\gcd(f(n), f(n-i+1))$. Therefore, we have the following partition function, $f_{GDB(i)}(n)$ that guarantees a client I/O bandwidth of ib,

$$
f_{GDB(i)}(n) = \begin{cases} 2^{n-1}, & \text{if } n \leq i \\ \lfloor (\sum_{j=n-i+1}^{n-1} f_{GDB(i)}(j)) \\ \quad / f_{GDB(i)}(n-i+1) \rfloor f_{GDB(i)}(n-i+1), & \\ & \text{if } n > i \end{cases}
$$

The resulting partition series is

$$1, 2, 4, 8, 16, \ldots, 2^{i-1}, 2^i-2, 2^{i+1}-8, \ldots$$

For example, when $i=4$, the partition series is

$$1, 2, 4, 8, 14, 24, 40, 70, \ldots$$

The partition function is further constrained by the disk space requirement. We see from Theorem 3.3 that the storage requirement of the GDB(i) scheme is determined by the largest segment size. Therefore, if C is the number of time units of data in the largest segment, we have the following partition function $f^C_{GDB(i)}(n)$ determined by C,

$$f^C_{GDB(i)}(n) = \begin{cases} f_{GDB(i)}(n), & \text{if } f_{GDB(i)}(n) < C \\ L, & \text{if } f_{GDB(i)}(n) \geq C \\ & \quad > f_{GDB(i)}(n-1) \\ C, & \text{if } f_{GDB(i)}(n-1) \geq C \end{cases}$$

where L is the largest number that satisfies $L \leq C$ and $L \leq \sum_{m=n-i+2}^{n-1} f_{GDB(i)}(m) + gcd(L, f_{GDB(i)}(n-i+1))$. Note that we limit $f^C_{GDB(i)}(n)$ by L for the case that $f_{GDB(i)}(n) \geq C > f_{GDB(i)}(n-1)$ in order to satisfy the sufficient condition (in Theorem 3.2) for the client bandwidth being at most ib for any C. For example, when $C = 65$, the partition series for GDB(4) is

$$1, 2, 4, 8, 14, 24, 40, 65, 65, 65, ...$$

When $C = 67$, the partition series for GDB(4) is

$$1, 2, 4, 8, 14, 24, 40, 66, 67, 67, 67, ...$$

4.1 The Client Resource Requirements

We can establish that the partition function $f^C_{GDB(i)}(n)$ satisfies the sufficient condition that a client's I/O bandwidth is at most ib. Furthermore, the sizes of the segments are monotonically increasing. Now, from Theorem 3.2 and 3.3, we have the client resource requirement of GDB(i).

Theorem 4.1 *GDB(i) requires a client disk bandwidth of ib and a client storage space of* $(f^C_{GDB(i)}(K)-1)Db/\sum_{n=1}^{K} f^C_{GDB(i)}(n)$.

4.2 Comparing the Resource Requirements of GDB(i) and SB

We compare the resource requirement of GDB(i) with the recently proposed broadcasting scheme, SB. The SB scheme is a disk-conserving broadcasting scheme with the partition function

$$f_{SB}(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2, & \text{if } n = 2, 3 \\ 2f_{SB}(n-1)+1, & \text{if } n > 3, n \bmod 4 = 0 \\ 2f_{SB}(n-1)+2, & \text{if } n > 3, n \bmod 4 = 2 \\ f_{SB}(n-1), & \text{if } n > 3, n \text{ odd} \end{cases}$$

We can generalize the partition function for any client storage space.

$$f^C_{SB}(n) = \min\{f_{SB}(n), C\}$$

We show that GDB(i) requires no more resources than SB does. It can be easily verified that $f^C_{GDB(i)} \succ f^C_{SB}$. From Corollary 1 and Theorem 3.3, we conclude that GDB(i) requires no more server channels and no more storage space than SB for guaranteeing the same service latency.

Figure 4 illustrates the client storage and the server channel requirement for SB, GDB(4), GDB(5), GDB(6). We assume that the video is displayed at a rate of 1.5 Mb/s, the video length D is 100 minutes and the guaranteed service latency is 6 seconds. We select the range 100–750 MBytes for the client storage space and 10–20 for the number of server dedicated channels. From the figure, we can see that when the server dedicates 16 channels, GDB(5) needs only 110MByte storage space while SB needs 220 MBytes, which is a 50% savings on the storage space. Even GDB(4) requires only 117 MByte storage space, which is close to a 50% savings. In addition, if the client storage space is 220 MBytes, GDB(5) needs only 11 dedicated broadcast channels while SB requires 16, which is a 31% savings on the server bandwidth. Furthermore, SB requires at least 16 dedicated broadcast channels regardless of the amount of client storage, while GDB(5) needs only 11 dedicated broadcast channels when the client has only 330MByte of storage space.

The curve labeled GDB(4) in Figure 4 plots the number of channels required verus client storage space when the client I/O bandwidth is $4b$. The GDB(5) curve shows how the channel requirements are decreased when the I/O bandwidth is increased to $5b$. However, we can see that the curves for GDB(5) and GDB(6) are very close to each other. In fact, the curves for GDB(5) and GDB(6) overlap when the client storage space is greater than 330 MBytes. Therefore, we see that most of the performance gains that can be realized by increasing the client I/O bandwidth are achieved by the time the I/O bandwidth is equal to 4 or 5 times the video channel bandwidth.

5 Broadcasting Scheme GDB3

In this section, we present a broadcasting scheme (referred to as GDB3) for the case that the client I/O bandwidth is $3b$. GDB3 is a disk-conserving scheme that is different from GDB(3). Recall that GDB(3) scheme uses a linear growing partition function

$$f_{GDB(3)}(n) = \begin{cases} 2^{n-1}, & \text{if } n \leq 3 \\ f_{GDB(3)}(n-1)+2, & \text{if } n > 3 \end{cases}$$

Since it can be easily verified that $f_{SB} \succ f_{GDB(3)}$, the disk-conserving broadcasting scheme with $f(n)$ as

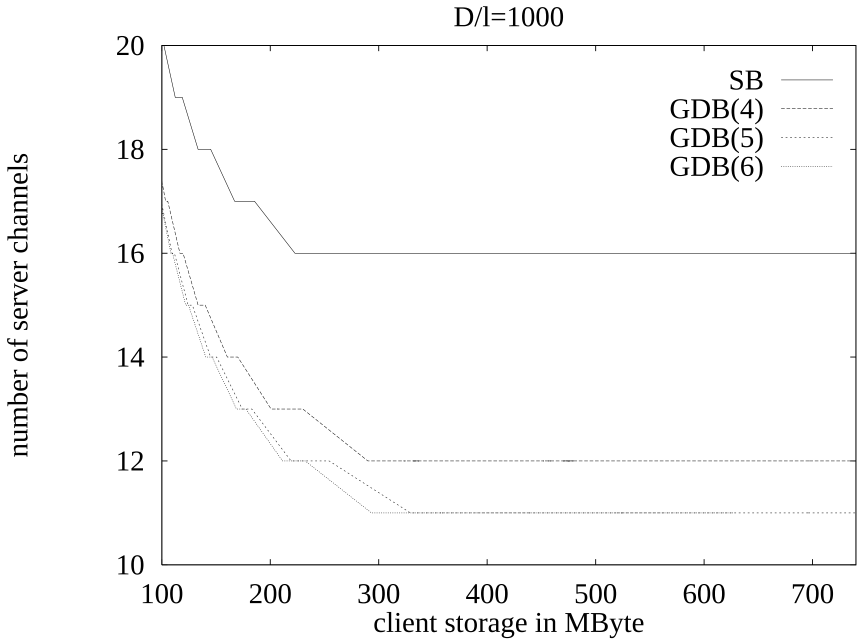

Figure 4: *The tradeoff exhibited by SB, GDB(4), GDB(5) and GDB(6)*

the partition function requires more resources than SB does. Therefore, we need to select an alternative partition function $f_{GDB3}(n)$ for the case that the client I/O bandwidth is $3b$. The intuition behind our choice of the partition function $f_{GDB3}(n)$ is to conserve the client I/O bandwidth by having two consecutively equal-length segments since the two equal-length segments can be received consecutively. Therefore, the segment size can grow exponentially without additional client I/O bandwidth. We have the following partition function for GDB3

$$f_{GDB3}(n) = \begin{cases} 2^{n-1}, & \text{if } n \leq 3 \\ 4, & \text{if } n = 4 \\ 10, & \text{if } n = 5, 6 \\ 24, & \text{if } n = 7, 8 \\ 5f_{GDB3}(n-4), & \text{if } n > 8 \end{cases}$$

The partition series generated by this function is

$$1, 2, 4, 4, 10, 10, 24, 24, 50, 50, 120, 120, ...$$

This partition function $f_{GDB3}(n)$ ensures that no more than two segments are simultaneously received. Therefore, the GDB3 scheme requires a client I/O bandwidth of $3b$.

As with the GDB(i) scheme, we can limit the size of the largest segment by parameter C, expressed in time units of data. More precisely, we have a disk-conserving broadcasting schemes that uses the partition function $f^C_{GDB3}(n)$ as follows.

$$f^C_{GDB3}(n) = \begin{cases} f_{GDB3}(n), \\ \quad \text{if } f_{GDB3}(n) < C \text{ and } n \text{ is odd} \\ L, \\ \quad \text{if } f_{GDB3}(n) \geq C > f_{GDB3}(n-2) \\ \quad \text{ and } n \text{ is odd} \\ C, \\ \quad \text{if } C \leq f_{GDB3}(n-2) \text{ and } n \text{ is odd} \\ f^C_{GDB3}(n-1), \\ \quad \text{if } n \text{ is even} \end{cases}$$

where L is the largest number that satisfies $L \leq C$ and $L \leq 2f_{GDB3}(n-2) + \gcd(f_{GDB3}(n-3), L)$. Note that we limit $f^C_{GDB3}(n)$ by L for the case that $f_{GDB3}(n) \geq C > f_{GDB3}(n-2)$ in order to satisfy the sufficient condition that the client bandwidth is at most $3b$ for any C.

5.1 The Client Resource Requirement

In this section, we analyze the client disk bandwidth and storage space required by GDB3. First, we see that the partition function $f^C_{GDB3}(n)$ satisfies the condition for ensuring that the client disk bandwidth required is at most $3b$. Therefore, the theorem follows.

Theorem 5.1 *GDB3 requires a client I/O bandwidth of $3b$ and a client storage space of $(f^C_{GDB3}(K) - 1)Db / \sum_{n=1}^{K} f^C_{GDB3}(n)$.*

5.2 Comparing the Resource Requirements of GDB3 and SB

We prove that GDB3 requires no more resources than SB does. It can be easily verified that $f^C_{GDB3} \succ f^C_{SB}$. From Corollary 1 and Theorem 3.3, we thus conclude that GDB3 requires no more server channels and no more storage space than SB does for guaranteeing the same service latency.

Figure 5 compares the resource requirements of SB and GDB3. We assume that the video is displayed at a rate of 1.5 Mb/s, the video length D is 100 minutes, and the guaranteed service latency is 6 seconds. The figure shows that GDB3 decreases the client storage space requirement and the number of dedicated server channels without additional client I/O bandwidth. For example, when the server dedicates 16 channels, GDB3 requires only 160 MBytes storage space while SB requires 230 MBytes, which is 30% reduction. Furthermore, when the client has 230 MBytes of storage space, GDB3 needs to dedicate only 15 server channels while SB requires 16 server channels, which results one channel saving per video.

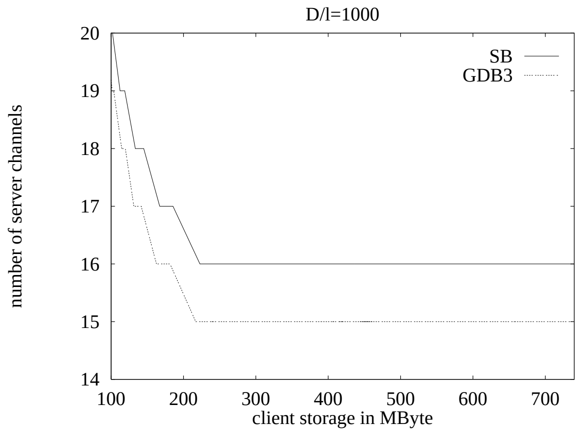

Figure 5: *The tradeoff exhibited by GDB3 and SB.*

6 Greedy Disk-Conserving Broadcasting Scheme GDB(K)

In Section 4, we saw that as the client I/O bandwidth grows, the client storage space and the number of server channels required are reduced. However, the client I/O bandwidth requirement never exceeds Kb in a disk-conserving scheme, since the client does not receive data from more than $K-1$ channels. Now, we present a disk-conserving broadcast scheme (referred to as GDB(K)) that assumes the client I/O bandwidth is at least Kb. Since the only constraint on the partition function is the non-workahead condition, the partition function for GDB(K) is

$$f_{GDB(K)}(n) = 2^{n-1}.$$

Limiting the size of the largest segment, we have a family of disk-conserving broadcasting schemes each of which depends on parameter C.

$$f^C_{GDB(K)}(n) = \max\{f_{GDB(K)}(n), C\}$$

In fact, GDB(K) is an optimal disk-conserving scheme. That is, given the client storage size, GDB(K) requires the smallest number of server channels among all disk-conserving schemes. This is because its partition function is the fastest growing partition function that satisfies the non-workahead condition and the storage size constraint. It follows from Theorem 3.2 that GDB(K) requires a client I/O bandwidth of at most $b(\lceil \log_2 C \rceil + 1)$. Therefore, GDB(K) requires a client I/O bandwidth of $b(\min\{K, \lceil \log_2 C \rceil + 1\})$, and storage space of $(f^C_{GDB(K)}(K) - 1)Db / \sum_{n=1}^{K} f^C_{GDB(K)}(n)$.

Figure 6 illustrates the client storage requirement and the number of server channels for schemes SB, GDB3, GDB(4), GDB(5), GDB(6), GDB(K). From the figure, we can see that by increasing the client I/O bandwidth by b, GDB(4) reduces both the client storage space and the number of dedicated channels significantly compared to GDB(3). For example, when the number of dedicated channels is 15, GDB(4) needs only 60% of the client storage space required by GDB(3). When the client storage space is 290 MBytes, GDB(4) needs to dedicate 3 fewer server channels than GDB(3) does. Furthermore, GDB(5)'s resource requirement is close to GDB(K)'s, which means that we can achieve close to optimal performance with a client I/O bandwidth of only $5b$.

6.1 The Minimum Number of Server Channels Required by GDB(K)

We see from the previous section that GDB(K) requires the minimum number of server channels when the client storage is only 550 MBytes. One natural question to ask is whether the minimum number of server channels required by GDB(K) is close to the optimal achievable by any broadcasting scheme. In this section, we show that the minimum number of server channels required by GDB(K) is only 1.44 times the minimum number of server channels required by *any* broadcasting scheme. We first determine the minimum server bandwidth required to achieve a given

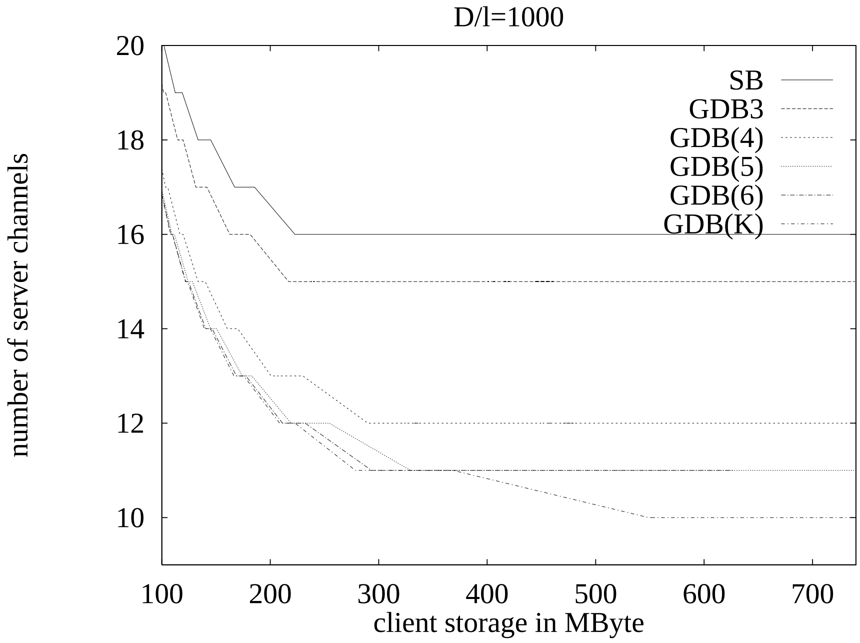

Figure 6: *The tradeoff exhibited by GDB(i) and GDB(K)*

latency for any broadcasting scheme. Using an argument on the broadcast frequency of a video data for any non-workahead broadcasting scheme, we show that the server needs to dedicate at least $\ln((D+l)/l)$ channels, for a D-second video.

Theorem 6.1 *Any broadcasting scheme needs to dedicate at least* $\ln((D+l)/l)$ *server channels in order to guarantee a service latency of* l *for a D-second video.*

PROOF : Consider the video data between the xth and $(x+dx)$th seconds of the video. A client arriving at time t has to receive the data between time t and $t+l+x$ in order to ensure the continuous playback, since the client has to begin to play out this data at time $t+l+x$. Therefore, the server needs to broadcast the data at least once between time t and $t+l+x$. Moreover, since a client can arrive at any time, the server needs to broadcast the data at least once between time t and $t+l+x$, for any t. We conclude that the data has to be broadcast every $(l+x)$ seconds.

To broadcast the data of length dx every $(l+x)$ seconds, the server needs a bandwidth of at least $bdx/(x+l)$. Since the server has to broadcast data between the xth and $x+dx$th second for all $x < D$, the total bandwidth required to broadcast the whole video is

$$\int_0^D bdx/(x+l) = b\ln((D+l)/l).$$

∎

The minimum number of server channels required by GDB(K) is $\log_2(D/l+1)$ for achieving a service latency l. This is because in GDB(K), to achieve a service latency l, the first segment is of length at least l, the second segment is of length at least $2l$, ... Therefore, we need at least $\log_2(D/l+1)$ segments or server channels for a D-second video. From the above theorem, the minimum number of channel required by GDB(K) is only 1.44 times of the minimum number of server channels required by *any* broadcasting scheme. From the previous section, we see that GDB(5)'s resource requirement is close to GDB(K). Therefore, with a reasonable client I/O bandwidth of only $5b$ or 0.94MBytes/s, the minimum number of server channels required by GDB is close to 1.44 times the minimum server resource requirement of *any* broadcasting scheme.

7 Summary

In this paper, we have considered the problem of broadcasting popular videos from a video-on-demand server to a large number of clients across a high-speed network. We proposed a formal framework for studying broadcasting schemes and designing a family of schemes, Greedy Disk-conserving Broadcasting (GDB). GDB gives a partition strategy, broadcasting schedule, and reception schedule according to the size of the client resources (including client I/O bandwidth and client storage space). We show analytically that GDB requires less resources than the most recently proposed scheme, *Skyscraper Broadcasting* (SB), while guaranteeing the same service latency. Furthermore, our performance study illustrates that GDB significantly reduces the resource requirements. Finally, we show that with reasonable client I/O bandwidth, both the server resource and the client storage space required by GDB is close to the minimum achievable by *any* disk-conserving broadcasting scheme.

Acknowledgments. The first author would like to thank Sneha Kumar Kasera and Zhi-Li Zhang for their helpful comments on the paper.

References

[1] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, "A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems", *Proc. of the IEEE Int'l Conf. on Multimedia Systems*, June 1996.

[2] C.C. Aggarwal, J.L. Wolf, and P.S. Yu. "On Optimal Batching Policies for Video-on-Demand Storage Server", *Proc. of the IEEE Int'l Conf. on Multimedia Systems*, June 1996.

[3] P.W. Agnew, and A.S. Kellerman, *Distributed Multimedia*, Addison Wesley, ACM Press.

[4] K.C. Almeroth, and M.H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service", *IEEE Journal on Selected Areas in Communications*, 14(6):1110-1122, August 1996.

[5] D. Comer, *Internetworking with TCP/IP*, Prentice Hall, 1995, pages 289-300.

[6] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", *Proc. of ACM Multimedia,* Oct. 1994, pages 15-23.

[7] A. Dan, D. Sitaram, and P. Shahabuddin. "Dynamic Batching Policies for an On-Demand Video Server", *Multimedia Systems,* 4(3):112-121, Jun. 1996.

[8] A. Dan, P. Shahabuddin, Dinkar Sitaram, and D. Towsley, "Channel allocation under batching and VCR control in movie-on-demand servers", *IBM Research Report*, 1994.

[9] S. Deering, RFC 1112.

[10] K.A. Hua, and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *ACM SIGCOMM.* Sept. 1997.

[11] S. Viswanathan, and T. Imielinski, "Metropolitan area video-on-demand service using Pyramid Broadcasting", *IEEE Multimedia Systems.* 4:197-208, 1996.