Presentation Planning for Distributed Video Systems *

Eenjun Hwang, **B. Prabhakaran** and **V.S. Subrahmanian**

Computer Science Department
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20770
{hwang, prabha, vs}@cs.umd.edu

Abstract

A distributed video-on-demand system is one where a collection of video data is located at dispersed sites across a computer network. In a single-site environment, a local video server retrieves video data from its local storage device (or devices). However, in the setting of a distributed VoD system, when a customer requests a movie from his/her local server, the server may need to interact with other servers located across the network. In this paper, we present three types of *presentation plans*, that a local server must construct in order to satisfy the customer's request. Informally speaking, a presentation plan is a detailed (temporally synchronized) sequence of steps that the host server must perform at given points in time. This involves obtaining committments from other video servers, obtaining committments from the network service provider, as well as making committments of local resources, within the limitations of available bandwidth, available buffer, and customer/client data consumption rates. The three types of plans described in this paper all work at different "levels of abstraction" in this planning process. Furthermore, we introduce *two* measures of how good a plan is: minimizing wait time for the customer, and minimizing a quantity called access bandwidth (which informally speaking, specifies how much network/disk bandwidth is used). We develop algorithms to compute optimal (w.r.t. the above measures) plans for all three types, and show experimentally that in all three cases, one of the three types of plans (called a *hybrid presentation plan*) systematically outperforms the other two. In addition to these new concepts, our framework has the advantage that many results that had previously been *verified experimentally* in the literature can now be conclusively *proved* mathematically.

*This work was supported by the Army Research Office under Grant Nr. DAAH-04-95-10174, by the Air Force Office of Scientific Research under Grant Nr. F49620-93-1-0065, by ARPA/Rome Labs contract F30602-93-C-0241 (ARPA Order Nr. A716), by Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium and by an NSF Young Investigator award IRI-93-57756. Proof of all theorem are in Appendix II.

1 Introduction

With the rapid advent of sophisticated, yet cheap digitization technology, accompanied by concomitant advances in networking technology, and increased consumer demand, there is now a tremendous amount of interest in distributed video on demand (VoD) systems. Such distributed VoD systems are characterized by the following salient features:

1. The video data is typically located at multiple sites on the network (which may be a global "open" network, or a proprietary corporate network). Each site has a local video server, and its own local resources (e.g. buffer space, disk bandwidth, etc.). Of course, the resources available at different servers may vary.

2. The customers (human) who wish to access this data are, likewise, located at geographically dispersed sites, and use some kind of local display device to view the video. Just as different video servers exhibit different characteristics above, so does the hardware available to each customer. One customer may use an outdated display device with a small buffer, and may have a very low consumption rate, while other higher-end users may have machines that have large buffers, and may also have high consumption rates.

3. When the human customer contacts his/her local server, and requests a movie, that local server must attempt to deliver the movie to the customer, taking into account, his hardware configuration, as well as the bandwidth of the communications channel between the local server and the customer. *The situation gets further complicated if the local server does not have the entire movie.* In this case, it must request appropriate "parts" of the movie from one or more remote servers. This, in turn, requires precise and carefully planning, and in particular, requires answers to the following questions:

   (a) *who* to ask?

   (b) *which* movie blocks to ask for ?

   (c) *when* to ask for a specific block ?

   (d) *how much resources* to ask for (e.g. bandwidth) from the network service provider?

   (e) *what resources to commit* (e.g. local buffer) and when should such committments be made?

   Answering all these questions is not enough – in order to ensure a jitter free presentation, the answers to the above questions, for different blocks of the movie, must be synchronized as well.

In this paper, we make the following contributions:

1. First and foremost, we present a VoD server architecture in which each server maintains some set of movie blocks. Unlike many previous works, we do not require that entire movies be stored at a site. Movies may be stored in part, or as a whole, at one or more sites.

2. We then formally define three types of *presentation plans* that a server could generate, when a customer makes a request for a movie. These presentation plans either describe how the movie will be delivered to the customer at a *block-oriented* level, or at a *segment oriented level*, or at a *hybrid* of the two.

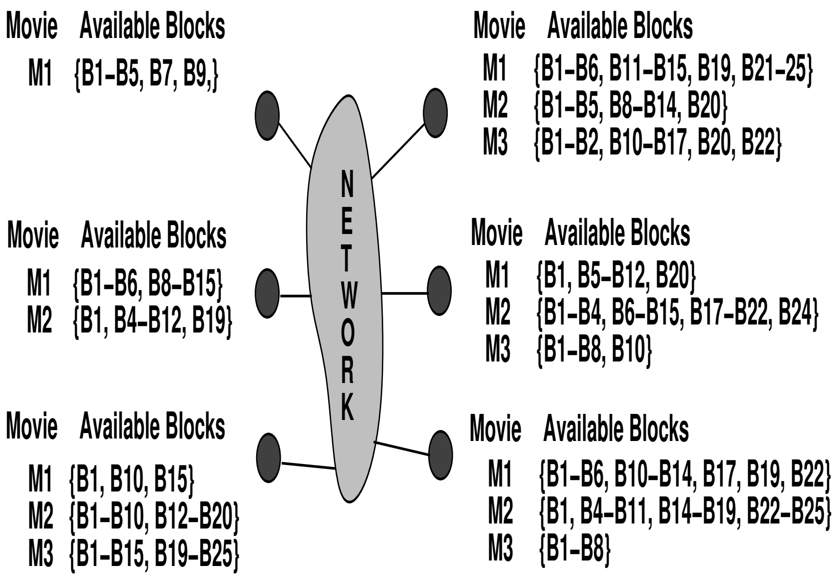

Figure 1: Movie Placement on VoD Servers

3. Some presentation plans may be better than others. Two measures of "how good" a presentation plan is, are introduced. In the first, the goodness of a presentation plan is defined merely in terms of how little time the customer has to wait, before viewing a jitter free presentation. However, serving one (current) customer may cause a future customer to be denied service (or to be delayed), and hence, an alternative measure, called *minimizing access bandwidth* is also proposed.

4. The three types of plans – *block oriented*, *segment oriented* and *hybrid* – are compared and various theoretical results are established.

5. We then provide a novel algorithm (outlined in the text, but in full gory detail in Appendix I) to compute optimal presentation plans w.r.t. these two criteria.

6. We then describe the results of simulations of VoD servers that we ran, to compare *block oriented*, *segment oriented* and *hybrid* presentation plans under both a *static(naive)* buffer allocation policy, as well as a *dynamic* buffer allocation policy. The data we used was derived from actual rental data at a video store [25] and has also been used by other authors ([2]).

   The results show, more or less conclusively, that in VoD servers, we are almost always better off computing hybrid presentation plans.

2 VoD Server Architecture

The architecture of the VoD server considered in our work is as shown in Figure 1. Movies may or may not be stored entirely on a particular server. Blocks of the same movie may be replicated and stored on many VoD servers on the network. Hence, if $\mathcal{MOVIE}$ is the set of movies that are stored in the set V of VoD servers, we may define a *placement mapping* as follows :

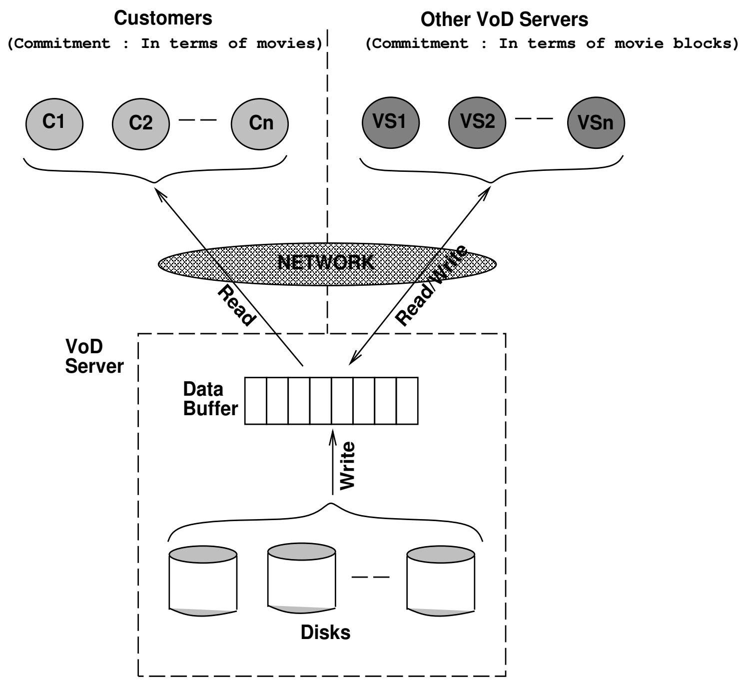

Figure 2: Functional View of a VoD Server

Definition 2.1 (Placement Mapping) A *placement mapping* is a mapping, $\wp$ that takes as input, (1) a movie $m_i \in \mathcal{MOVIE}$ and (2) a block number, b, and returns as output, a subset of V.

Typically, a VoD server serves a set of customers at any given point in time. A movie requested by a server may or may not be available entirely on the server. If the requested movie is not available on the server, the server must try to obtain the relevant blocks of the movie from other VoD servers. Hence, in addition to customers, a VoD server may be accessed by other VoD servers as well.

Figure 2 shows the functional view of a VoD server. Each server has a set of buffers where the movie blocks are loaded. The movie blocks may be written onto the buffers either by reading from the local disk or by obtaining the data from other VoD servers. In the same way, the movie blocks located at one server may be read by both customers as well as other VoD servers. We introduce three types of frameworks that a VoD server may use in order to read and write movie blocks to its buffer.

- *Access (read or write) movie blocks individually :* Here, customers download a movie block by block. In a similar manner, other VoD servers also download a movie, block by block, as shown in Figure 3(a). We call this a *block-oriented presentation.*

- *Access the movies in a specified segments of contiguous blocks :* Here, customers and other VoD servers download movies in a specified sets of blocks, as shown in Figure 3(b). We call this a *segment-oriented presentation.* The download operation is assumed to be complete only when the entire set of blocks is available on the requesting system (customer or a VoD server).

- *Access the movies in a flexible set of blocks :* Here, downloading is done in terms of a set of blocks, as in the case of a segmented-oriented presentation. However, as shown in Figure

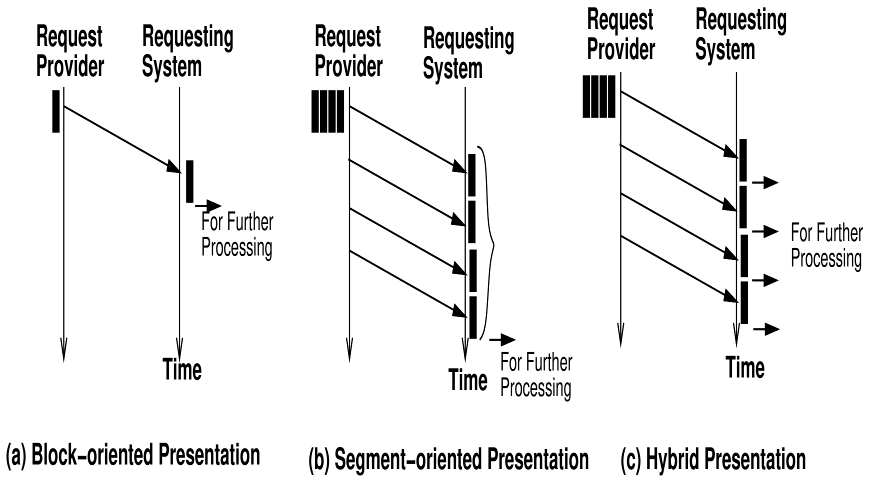

Figure 3: Types of Movie Presentation

3(c), the requesting system can start using a block (in the set of blocks being downloaded) immediately, without having to wait for the downloading of the entire set of blocks to be completed. We call this a *hybrid presentation.*

3 VoD Server Interaction

As discussed above, a VoD server may interact with :

- A customer for satisfying a movie request.
- Another (remote) VoD server for transferring movie block(s). The transfer of movie block(s) may be made either individually or in terms of segments, as discussed in Section 2.

Each VoD server has a fixed amount of buffer space that it can use to store the data downloaded from other servers. If v is a node, we use $\texttt{buf}(v)$ to denote the total amount of buffer space it has. The actual amount available may vary from time to time, depending upon how much of this buffer space is currently in use. In a similar manner, the network bandwidth available for the transfer of movie blocks (to customers or other VoD servers) is denoted by $\texttt{bw}(v, x, t)$. This bandwidth $\texttt{bw}$ specifies the maximum possible data rate of communication at any time t between the VoD server v and another system (customer or VOD server) x, as agreed to by the network service provider.

3.1 Server-Customer Interaction

A VoD server must construct a delivery schedule for a requesting system (customer or another VoD server) based on certain criteria of optimality. This schedule must contain a description not only of which blocks it delivers to the customer/client at which point in time, but also includes information about when it requests data from remote servers, what data rate that remote data will arrive at, how much local buffer the local server will commit to buffering data from each such remote server, and the rate at which this data will be shipped to the customer. In addition, there are numerous similar constraints that must be satisfied at the customer's end (e.g. there is no point in shipping data to the customer at a high rate, if the customer has a small buffer and a very low consumption rate). Many criteria for optimality of a presentation plan can be used. We use the following two criteria in the paper :

- *Minimizing the customer wait time :* The presentation plan is generated in such a way that the wait time for the customer to start watching an uninterrupted movie, is minimized.
- *Minimizing the access bandwidth :* Here, the plan is generated in such a way that the accesses (local disk or network) required for buffering the movie blocks in the VoD server is minimized. (We minimize the amount of bandwidth used, both at the disk level and the network level, rather than minimize the total number of accesses).

A presentation plan contains a detailed schedule specifying what a server s must do in order to satisfy the request for a movie from a customer. Presentation plans can be generated for each of the presentation types : block-oriented, segment-oriented and hybrid. For a server to generate a presentation plan, we will assume that the following capabilities of the customer are made known :

1. **Customer Consumption Rate:** The value, $\mathtt{ccr}(C)$, specifies the rate at which customer C consumes media data. In particular, we assume (without loss of generality) that the units used here are the same as for specifying the bandwidth of edges in the network/bandwidth of network servers.
2. **Customer Buffer Size:** The value, $\mathtt{buf}(C)$, specifies the total amount of buffer space available at the customer's end.
3. **Customer-Server Bandwidth:** This value, denoted $\mathtt{bw}(C, v) = \mathtt{bw}(v, C)$ specifies the bandwidth of the link between the customer C and the server v assigned to him/her.

3.1.1 Server-Server Interactions

A VoD server interacts with another VoD server when one or more movie blocks required for a presentation are not available locally. Suppose a server v has obtained a request from a customer C for movie m. Suppose movie m has $\mathsf{bnum}(m)$ blocks altogether. Server s attempts to obtain these blocks from different servers so that it can present these to the customer.

4 Presentation Record : Data Structure

When processing a customer's request for a movie, the VoD server has to identify where the desired movie blocks are stored. It is assumed that the movie placement mapping is known to each VoD server. In case some movie blocks are not available locally, the VoD server has to download the blocks from other VoD servers. A *presentation record* r is the data structure used by a VoD server for interacting with a customer as well as with another VoD server. If the VoD server is constructing block oriented plans, then it associates one presentation record with each block of the requested movie. In the case of segment-oriented and hybrid presentation plans, a presentation record is associated with each set of movie blocks. In the case of a block oriented presentation plan, a record is defined for each movie block.

1	**Orig**	Specifies the server that originated the request.
2	**Target**	Specifies the server that will satisfy the request.
3	**Movie**	Specifies the movie-id associated with the request.
4	**Start**	Specifies the first movie block being requested.
5	**End**	Specifies the last block being requested.
6	**Reqtime**	This is the value at which block request is initiated.
7	**ConOK**	This is the time at which the connection is successfully made.
8	**BWAssign**	This is the bandwidth assigned to the request by the target server.
9	**DelivSt**	This is the time at which delivery starts.
10	**DelivEnd**	This is the time at which target server completes delivering blocks.

Table 1: Presentation Record : For Interaction With Target VoD Servers

11	**CustShipSt**	This is the time at which the originating server starts shipping the blocks to the customer.
12	**CustShipEnd**	This is the time at which the originating server finishes shipping the blocks to the customer.
13	**CustConsStart**	This is the time at which the customer starts consuming the blocks.
14	**CustConsEnd**	This is the time at which the customer finishes consuming the blocks.

Table 2: Presentation Record : For Interaction With Customers

Start and **End**	$End > Start$, a segment consisting of more than one block.
ReqTime	$t_{Req(r.Orig,r.Target,r.Movie,[r.Start,r.End])}$
ConOk	$r.conOK = t_{Req(r.Orig,r.Target,r.Movie,[r.Start,r.End])}$ $+ ct(r.Orig, r.Target)$
BWAssign	$r.BWAssign \leq \texttt{bw}(r.Target, r.Orig, t)$
DelivSt	$r.DelivSt = r.conOK$
DelivEnd	$r.DelivEnd = r.DelivSt + \frac{(r.Start - r.End + 1) \times \texttt{bsize}}{r.BWAssign}$
CustShipSt	$r.CustShipSt \geq r.DelivEnd$
CustShipEnd	$r.CustShipEnd = r.CustShipSt + \frac{(r.End - r.Start + 1) \times \texttt{bsize}}{\texttt{bw}(r.Orig, C, t_{r.CustShipSt})}$, where C is the customer.
CustConsStart	$r.CustConsStart \geq r.CustShipEnd$
CustConsEnd	$r.CustConsEnd = r.CustConsStart + \frac{(r.End - r.Start + 1) \times \texttt{bsize}}{ccr(C)}$

Table 3: Segment-Oriented Presentation Record

The presentation record has two sets of fields that describe : (i) the interaction with another VoD server, termed *target server* (ii) the interaction with the customer. Basically, the fields in the presentation record describe some of the *actions* carried out by a VoD server and the time instant associated with these actions. These actions deal with :

- Establishing a connection with another VoD server for downloading movie blocks.
- Downloading (start and end) of the blocks from the VoD server
- Downloading the blocks to the customer site.

Tables 1 and 2 describe the fields associated with a presentation record for interacting with VoD servers and customers.

In the above presentation record data structure, fields (1) - (10) describe the parameters used for interacting with other VoD servers. Here, the term *originating server* denotes the server to which a customer is attached for downloading the requested movie. The term *target server* denotes a server from which the originating VoD server downloads missing movie blocks. In a similar manner, the fields (11) - (14) describe the interactions with a customer.

4.1 Presentation Records for Different Plans

Different presentation plans such as block-oriented, segment-oriented and hybrid, assign different values and structures to the fields in a presentation record. Table 3 describes the expressions used for the fields in a segment-oriented presentation record. Table 4 describes the fields in a hybrid presentation record. The first 9 fields in the hybrid presentation record are the same as those in a segment-oriented presentation record.

The fields of a block-oriented presentation record are the same as in the case of a segment-oriented presentation plan, except that the number of movie blocks requested at any point in time is only one, i.e., $Start = End$. Hence, we can say the following :

DelivEnd	For each block b_w where $r.Start \leq w \leq r.End$, $r.DelivEnd[w] = r.DelivSt + \frac{(w - r.Start + 1) \times \texttt{bsize}}{r.BWAssign}$
CustShipSt	For each block b_w where $r.Start \leq w \leq r.End$, $r.CustShipSt[w] \geq r.DelivEnd[w]$
CustShipEnd	For each block b_w where $r.Start \leq w \leq r.End$, $r.CustShipEnd[w] = r.CustShipSt[w] + \frac{\texttt{bsize}}{\texttt{bw}(r.Orig, C)}$
CustConsStart	For each block b_w where $r.Start \leq w \leq r.End$, $r.CustConsStart[w] \geq r.CustShipEnd[w]$
CustConsEnd	For each block b_w where $r.Start \leq w \leq r.End$, $r.CustConsEnd[w] = r.CustConsStart[w] + \frac{\texttt{bsize}}{ccr(C)}$

Table 4: Hybrid Presentation Record

Definition 4.1 A segment-oriented presentation record r is said to be a *block oriented presentation record* iff r.**End** = r.**Start**.

It is important for the reader to notice that in the case of hybrid presentation records, we consider each and every block of the segment of video being shipped. However, unlike block oriented presentation records, we do not need multiple records to store them. Furthermore, in hybrid presentation records, once a connection has been opened to the target server, the connection stays open for all blocks in the segment being requested; in contrast, in block-oriented presentation records, connections need to be requested and opened for each record, thus leading (possibly) to higher wait times for the customer.

4.2 Feasible Presentation Plans

A VoD server must create and maintain a *presentation plan* for each customer arriving with a request for a movie. As discussed earlier, this presentation plan can be one of the following three types : a segment-oriented presentation plan, a block-oriented presentation plan or hybrid presentation plan. Any presentation plan must ensure the following conditions:

- A commitment must have been obtained from the originating VoD server to ship movie blocks to the customer so that the movie can be watched without any interruptions.
- Commitments must have been obtained from target VoD server(s) for providing movie blocks to the originating VoD server when all the movie blocks are not available local to the originating server.
- Committments must have been obtained from the network service provider to ensure that bandwidth is available to ship the blocks at the desired transfer rate.

The above commitments are maintained as *Commitment Records* by the (originating and target) VoD servers. The following information is maintained as part of the commitment record list :

Informally speaking, a presentation plan is said to be feasible if the following conditions are satisfied.

BegCom	This specifies the start time of a commitment.
FinCom	This specifies the finish time of a commitment.
Client	This could either be a customer, or another server to whom a commitment is being made.
Movie	This specifies what movie forms part of the commitment.
BlockSt	This specifies the starting block of the movie.
BlockEnd	This specifies the ending block of the movie associated with this commitment.
BWCom	This specifies the amount of bandwidth committed to this commitment.

Table 5: Commitment Record

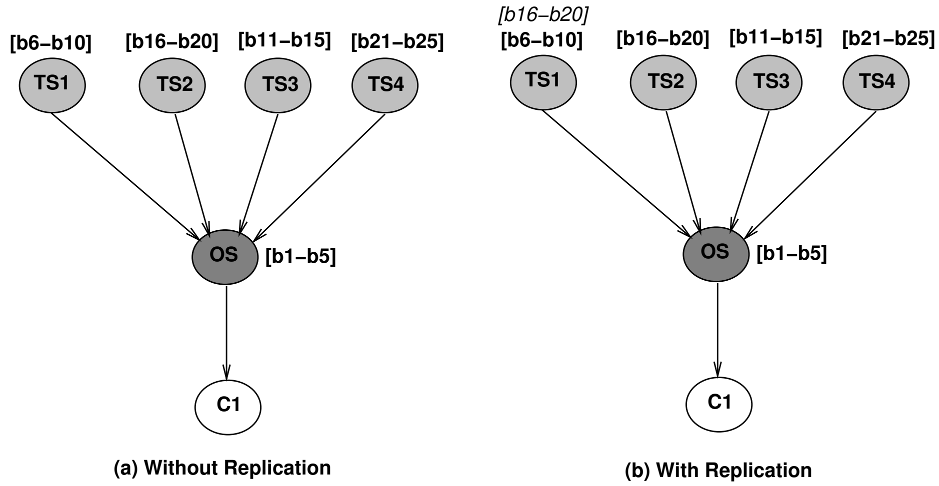

Figure 4: Serving A Typical Customer

- The load on the originating and the target servers are such that the customer's request can be handled by them.
- Buffer space is available in the customer site for downloading the movie.
- Buffer space is available in the originating server site for holding the blocks downloaded from a target server (till the blocks are shipped to the customer).
- Bandwidth is available (from the network service provider) to accomplish shipping the data at the desired rate.

4.2.1 Feasible Presentation Plan : An Example

Figure 4 shows an originating server OS serving a customer $c1$. It is assumed that the requested movie has 25 blocks distributed over the originating server and the target servers ($TS1$ to $TS4$). Figure 4 (b) shows the scenario where some of the movie blocks are replicated. The server OS has to

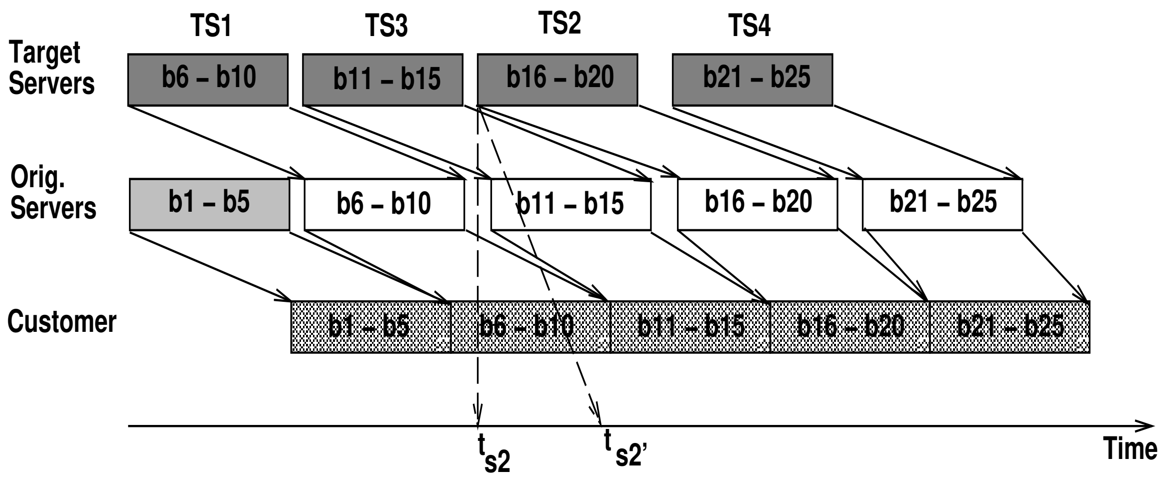

Figure 5: A Feasible Presentation Plan

download blocks $b6$ to $b25$ from the target servers, in order to satisfy the customer's request. Before delivering the requested movie to the customer $C1$, the server OS has to first create a presentation plan. In this example, let us assume that the server OS creates a segment-oriented presentation plan.

Figure 5 describes a feasible segment-oriented presentation plan for serving the customer. Blocks $b1 - b6$ are available local to the server OS and hence can be shipped to the customer directly. The server OS has to get a commitment from the target servers for downloading the missing blocks as follows : $b6 - b10$ from $TS1$, $b11 - b15$ from $TS3$, $b16 - 20$ from $TS2$ and $b21 - b25$ from $TS4$. In order to download the blocks from the target servers, the server OS has to specify the time at which the blocks are needed by OS. Based on the request time of the blocks, the target servers have to make an entry in their commitment record for downloading the blocks to the server OS. In case, a target server is not able to commit for the download at the requested time, the server OS can either try another target server or request the same target server for another commitment time. In Figure 5, let us assume that the target server $TS2$ is not able to commit at the requested time t_{s2}. Instead, it is able to commit for the blocks $b16 - b20$ at time $t_{s2'}$. In the case of Figure 4 (a), there is no replication of movie blocks. Hence, the server OS has to shift the *entire* presentation plan by $t_{s2'} - t_{s2}$ in order to ensure a jitter free presentation. In the case of Figure 4 (b), the server OS can possibly try to download the blocks from the server $TS1$.

We are now ready to *formally* define what it means for a segment/block oriented presentation plan to be feasible.

Definition 4.2 (Feasible Segment/Block Oriented Presentation Plan) Suppose $PP = r_1, \ldots, r_k$ is a segment oriented (resp. block oriented) presentation plan for delivering movie m to customer C via originating server v. PP is said to be *feasible* iff the following conditions hold:

1. Let $CRL(s)$ denote the commitment record list associated with server s, $s \in V$. For each $1 \leq i \leq k$, insert the tuple
$(r_i.DelivSt, r_i.DelivEnd, v, r_i.Movie, r_i.Start, r_i.End, (r_i.End - r_i.Start + 1)$
$\times$`bsize`) into $CRL(r_i.Target)$.

Constraint 1: For each point t in time, $r_1.Start \leq t \leq r_k.End$, and for each server s, the load on server s at time t must be less than or equal to 1 (100%).

2. Let v be the primary server associated with customer C. Let t be any time point such that $r_1.DelivSt \leq t \leq r_k.DelivEnd$. The set of *deliver-but-unconsumed blocks* $DBUB(C,t)$ to customer C at time t is the set $\{b_j \mid$ for some $1 \leq i \leq k$, $r_i.Start \leq b_j \leq r_i.End$ and $t \geq r_i.CustShipEnd$ and $t < r_i.CustConsEnd\}$.

   Constraint 2: For each point t in time such that $r_1.DelivSt \leq t \leq r_k.DelivEnd$

$$\texttt{buf}(C) \geq \texttt{bsize} \times card(DBUB(C,t)).$$

3. Let v' be any server. Let t be any time point such that $r_1.DelivSt \leq t \leq r_k.DelivEnd$. The set of *delivered-but-unconsumed blocks* $DBUB(v',t)$ to server v' at time t is the set $\{b_j \mid$ for some $1 \leq i \leq k$, $r_i.Start \leq b_j \leq r_i.End$ and $r_i.Target = v'$ and $r_i.DelivSt \leq t$ and $r_i.CustShipEnd \geq t\}$.

   Constraint 3: For each point t in time such that $r_1.DelivSt \leq t \leq r_k.CustShip$ End

$$\texttt{buf}(v') \geq \texttt{bsize} \times card(DBUB(v',t)).$$

Definition 4.3 (Wait Optimal Segment/Block Presentation Plan) A *Segment/Block Oriented Presentation Plan* for delivering movie m to customer C via originating server v is any feasible presentation plan.

The *customer wait* associated with a segment/block oriented presentation time table $r_1, \ldots, r_k$ is defined to be the value of the field $r_1.CustConsStart$.

A segment (resp. block) oriented presentation time table $PPT = r_1, \ldots, r_k$ for delivering movie m to customer C via originating server v is said to be *Wait Optimal* iff for all other segment (resp. block oriented) presentation time tables $PPT' = r'_1, \ldots, r'_m$ for delivering movie m to customer C via originating server v,

$$r_1.CustConsStart \leq r'_1.CustConsStart.$$

In other words, PPT is optimal iff there is no other presentation time table with a "smaller" customer wait.

An alternative criterion for optimality is *access bandwidth*. Every time the originating server satisfying a customer's request reads data into its buffers, it does so because either the data was shipped to it by another server, or because it read it from its local disk. The *access bandwidth* of a presentation plan $PP = r_1, \ldots, r_k$ is defined to be the *total amount (in blocks) of data* that is either shipped across the network or that is read from disk.

Of course, when a server S is satisfying a customer's request for a movie M, the reader may feel that the access bandwidth of the movie is equal to the number of blocks of the movie. However, there is some subtlety here: the number of blocks in the movie is only an *upper bound* on the access bandwidth of the presentation plan. The *actual* access bandwidth depends upon the presentation plan, because the presentation plan may take into account *other committments* that the server has made to *other customers*. For instance, the originating server for (new) customer C_{new} may take into account, the fact that it had just constructed a partial presentation plan for another (older) customer C_{old}, and it may be able to retrieve data from a remote server once, and satisfy both the old and the new customer by a single retrieval.

Definition 4.4 (AB Optimal Segment/Block Presentation Plan) A *Segment/ Block Oriented Presentation Plan* for delivering movie m to customer C via originating server v is *AB-Optimal* (AB stands for Access Bandwidth) iff there is no other presentation plan that has a smaller access bandwidth.

4.2.2 Hybrid Presentation Plans

Suppose $\mathsf{HPP} = r_1, \ldots, r_k$ is a hybrid presentation plan. The structure of the constraints that must be satisfied by HPP are somewhat different from those satisfied by segment (resp. block) oriented time tables because of the different structure of hybrid presentation records.

Definition 4.5 (Feasible Hybrid Presentation Plan) Let C be a customer and let v be the customer's originating server. A hybrid presentation plan $\mathsf{HPP} = r_1, \ldots, r_k$ is said to be *feasible* iff it satisfies the constraints listed below:

1. For each $1 \leq i \leq k$ and for each $r_i.Start \leq w \leq r_i.End$, insert the tuple

   $$(r_i.DelivSt + \frac{(w - r_i.Start) \times \mathtt{bsize}}{r.BWAssign}, r_i.DelivEnd[i], v, r_i.Movie, w, w, \mathtt{bsize})$$

   into $CRL(r_i.Target)$. Notice the difference between the tuple inserted here and the tuple inserted in the definition of segment/block oriented feasible presentation plans.

   Constraint 1: For each point t in time, $r_1.Start \leq t \leq r_k.End$, and for each server s, the load on server s at time t must be less than or equal to 1 (100%).

2. Let t be any time point such that $r_1.DelivSt \leq t \leq r_k.DelivEnd[r_k.End]$. The set of *delivered-but-unconsumed blocks* $DBUB(C,t)$ to customer C at time t is the set $\{b_j \mid \text{ for some } 1 \leq i \leq k, r_i.Start \leq b_j \leq r_i.End \text{ and } t \geq r_i.CustShipEnd[j] \text{ and } t < r_i.CustConsEnd[j]\}$.

   Constraint 2: For each point t in time such that $r_1.DelivSt \leq t \leq r_k.DelivEnd\,[r_k.End]$

   $$\mathtt{buf}(C) \geq \mathtt{bsize} \times card(DBUB(C,t)).$$

   Again, note the subtle differences between this definition and definition 4.2.

3. Let v' be any server. Let t be any time point such that $r_1.DelivSt \leq t \leq r_k.DelivEnd[r_k.End]$. The set of *delivered-but-unconsumed blocks* $DBUB(v',t)$ to server v' at time t is the set $\{b_j \mid$ for some $1 \leq i \leq k$, $r_i.Start \leq b_j \leq r_i.End$
   and $r_i.Target = v'$ and

   $$r_i.DelivSt + \frac{(j - r_i.Start + 1) \times \mathtt{bsize}}{\mathtt{bw}(v', v)} \leq t$$

and $r_i.CustShipEnd[j] \geq t\}$.

Constraint 3: For each point t in time such that $r_1.DelivSt \leq t \leq r_k.CustShip End[r_k.End]$

$$\texttt{buf}(v') \geq \texttt{bsize} \times card(DBUB(v', t)).$$

Again, note the subtle differences between this definition and definition 4.2.

Definition 4.6 (Wait Optimal Hybrid Presentation Plan) A *Hybrid Presentation Plan* for delivering movie m to customer C via originating server v is any feasible hybrid presentation plan.

The *customer wait* associated with a hybrid presentation time table $r_1, \ldots, r_k$ is defined to be the value of the field $r_1.CustConsStart[1]$.

A hybrid presentation time table $PPTp = r_1, \ldots, r_k$ for delivering movie m to customer C via originating server v is said to be *Wait Optimal* iff for all other hybrid presentation time tables $PPT' = r'_1, \ldots, r'_m$ for delivering movie m to customer C via originating server v,

$$r_1.CustConsStart[1] \leq r'_1.CustConsStart[1].$$

In other words, PPT is optimal iff there is no other hybrid presentation time table with a "smaller" customer wait.

AB-optimality of hybrid presentation plans is defined in the same way as for segment/block oriented plans.

5 Computing Presentation Plans

In this section, we shall describe how to compute presentation plans to retrieve a movie requested by a customer. First, we shall describe how we can compute a segment-oriented presentation plan. Block-oriented and hybrid presentation plans are variations of segment-oriented presentation plans. We assume that the originating server OS for a customer has access to the following information.

- Movie placement function.
- Network bandwidth, bw, to the target servers that have the blocks for the requested movie.

In order to minimize the access bandwidth, the originating server OS does the following.

- In case multiple requests for the same movie arrive simultaneously (or shortly after one another), the server OS minimizes the access bandwidth by doing the following :
  - Keeping downloaded segment(s) (from other target servers) for a maximum time period τ. If another request for the same segment(s) can be satisfied within that time, then this obviously decreases the access bandwidth, by avoiding shipping the same object twice. Holding the downloaded segment is done only if sufficient space is available.

- The above methodology can also be applied for segments retrieved from disks. The segments retrieved from disks may be held in the main memory, and if another request for the same segment arrives within time τ, then the locally stored segment does not need to be retrieved again.

When a new request for a movie is made by a customer, the originating server OS creates a presentation plan based on the following steps. (The detailed algorithm is contained in Appendix I).

1. The following variables are set initially. Starting movie block number NSB (*Next Start Block*) is set to 0. Earliest movie start time *sttime* is set depending on the request arrival time and a minimal processing time (δt). NPD, the next presentation deadline for the blocks starting from NSB, is set to *sttime*.

2. OS checks whether any consecutive blocks starting from NSB are available locally (stored on the disk or downloaded and cached from other servers). If so, NSB is incremented depending on the number of consecutive available blocks.

3. If the blocks starting from NSB are not available locally, OS has to first identify the target servers from which the segment of movie blocks can be downloaded in such a way that the blocks will be available by NPD.

4. In order to identify the target servers, OS has to determine the time taken to download the required segments of blocks from the different target servers. The time at which the required segment might be available at each target server may be determined as follows. Suppose the server OS asks for a segment comprising of blocks $b1$ to $b2$ from a target server $TS1$. Then, this request is handled as follows.

   - **Request Time:** Let $t_{Req(v_1,v_2,m,[b1,b2])}$ denote the time at which the request is *issued* by server v_1.
   - **Estimated Connection Time:** Let $ct(v_1, v_2)$ reflect the *estimated connection time* between v_1 and v_2. If a connection already exists between v_1 and v_2, then $ct(v_1, v_2) = 0$. Thus, server v_2 receives the request $Req(v_1, v_2,$ $m, [b1, b2])$ from server v_1 only at time

     $$ct(v_1, v_2) + t_{Req(v_1,v_2,m,[b1,b2])}.$$

   - **Download Complete:** Finally, suppose each *block* has size `bsize`. Then the total download time to download all blocks in the interval $[b_1, b_2]$ is given by

     $$\frac{(b_1 - b_2 + 1) \times \texttt{bsize}}{\texttt{bw}(v_2, v_1, t_{Req(v_1,v_2,m,[b1,b2])})}.$$

     Thus, the actual time that server v_1 receives the required data is

     $$ct(v_1, v_2) + t_{Req(v_1,v_2,m,[b1,b2])} + \frac{(b_2 - b_1 + 1) \times \texttt{bsize}}{\texttt{bw}(v_2, v_1, t_{Req(v_1,v_2,m,[b1,b2])})}.$$

5. After determining the request time for segment of blocks, the server OS has to issue a request to the target server giving the following information :

- Segment(s) of the requested movie.
- The time at which the segment(s) are required ($t_{Req(v_1,v_2,m,[b1,b2])}$) .

6. The target server, depending on its load, can do one of two things: *accept* or *postpone* the request by a time Δt.

7. If the set of target servers that agree to process the request is non-empty, determine the target server with minimum waiting time. NSB, next starting block, *This is computed as a function not just of when the target server can start shipping the data, but also taking into account, the bandwidth that the network service provider can provide.* is incremented appropriately. NPD, next presentation deadline, is incremented by the time required for playing the set of blocks whose presentation plan has been fixed above. The above sequence of steps is repeated from Step 2. The algorithm terminates when the plans have been fixed for all the movie blocks.

8. If the set of target servers that agree to process the request is empty, OS has no other option but to delay the movie start time. Hence, OS selects the minimum delay Δt (at which the requested blocks starting from NSB will be available). The movie start time, *sttime*, is incremented by Δt, NSB (next starting block) is set to 0, and the whole sequence is repeated from Step 2. If the movie start time exceeds the maximum waiting time specified by the customer, then the request is rejected (and the algorithm terminates).

The above sequence of steps are fully described as an algorithm in Appendix I. The above discussion applies to all the three presentation plans. In the case of a block-oriented presentation plan, the segment size is always 1 block. However, *after* the above plan is created, the three presentation plans differ in the following manner.

- For segment and block oriented presentation plans, the delivery of the segments to the customer starts *after* the download of the *entire* segment is complete. (For block-oriented plan, the segment size is always one).

- For hybrid presentation plan, the delivery can start immediately after *any one* of the blocks in the requested segment is downloaded.

6 Simulation Experiments

Simulation experiments of the suggested VoD architecture were carried out. A total of 600 customers were assumed to make requests for movies. Table 6 summarizes the parameters used for simulation. The access patterns of the movie follow a Zipf distribution and use raw data obtained from a video rental store[25], and that has previously been used by several other authors [1, 3]. (It is worth noting that it does not necessarily follow that requests to a VoD system will exhibit the same access patterns as rentals from a video store of the sort currently found in shopping malls. However, in the absence of other data, the assumption that the requests follow a Zipf distribution is reasonable). Furthermore, the access patterns were derived from actual data obtained from a video rental store

For the movie placement mapping, we use the concept of *replication factor* defined originally in [15]. *Replication factor* is defined as the ratio of the sum of the number of movie blocks stored in the VoD servers to the sum of the number of blocks required for the movies stored, i.e.,

1	Number of Movies	300
2	Number of Segments	20-30 per movie (avg 25)
3	Size of Segment	10-30 blocks (avg 20)
4	Size of Block	6 seconds' compressed video data
5	Number of Requests	600
6	Req Arrival Time (for overall system)	5-25 sec (avg 12)
7	Request Pattern	Based on actual data referenced in [1] (it's almost matched with Zipf distribution)
8	Number of Servers	5
9	Disk Buffer size	30 MB
10	Disk Bandwidth	Avg. 1.9 MB

Table 6: Parameters Used For Simulation

$$Replicationfactor = \frac{\sum_{server\ s} numblocks(s)}{\sum_{movie\ m} numblocks(m)}$$

where $numblocks(s)$ denotes the total number of movie blocks stored at serve s and $numblocks(m)$ denotes the total number of blocks in movie m.

The replication factor is 1 when there is no replication of movie blocks in the set of VoD servers. The simulation experiments were carried out with replication ratios 1, 1.30 and 1.50.

The *Window size* (earlier denoted by the variable τ) for keeping downloaded blocks in memory was varied from 0 to 90 seconds, in steps of 30 seconds. Also, the buffer allocations on the VoD servers were done with two different strategies :

- **Naive Strategy :** Here, buffer allocation was done for the *entire* set of blocks in a segment for the required time interval.
- **Dynamic Strategy :** Here, buffer allocation was done for *each* block at the required time.

Figure 6 shows the results of constructing and executing block-oriented, segment-oriented, and hybrid presentation plans obtained using the naive buffer allocation strategy. Figure 7 show the results under the dynamic buffer allocation strategy. The performance of the different presentation plans were as follows.

1. **Block-oriented Presentation Plan :** Performance is more or less the same for both the naive and dynamic buffer allocation strategies. The number of accepted customers increased and the average customer wait time decreased as the movie replication ratio increases. As the window size (for maintaining the buffers in memory) increases, the number of accepted customers show a marginal increase. However, the average time for computing the presentation plan was significantly higher (three to four times) than that for hybrid presentation plans.

2. **Segment-oriented Presentation Plan :** Performance is the worst because of poor utilization of buffer resources. The number of accepted customers increased and the average customer wait time decreased with the dynamic buffer allocation strategy (as compared to the static case).

Experiment Result Under Naive Buffer allocation Strategy

ratio	schedule / buffer interval	Block-Oriented 0	Block-Oriented 30	Block-Oriented 60	Block-Oriented 90	Segment-Oriented 0	Segment-Oriented 30	Segment-Oriented 60	Segment-Oriented 90	Hybrid 0	Hybrid 30	Hybrid 60	Hybrid 90
R = 1	Accepted requests	498	496	503	504	34	33	33	33	473	474	477	479
	Waiting time(sec)	32	30	34	32	189	184	184	184	41	43	45	44
	Buffer allocated	1568	1567	1594	1580	22685	23509	23427	23427	1626	1644	1639	1660
	Access bandwidth	71	71	71	71	44	45	45	45	70	70	69	69
	Execution time(ms)	55	58	63	61	27	27	27	27	20	21	20	20
R = 1.30	Accepted requests	595	595	594	589	34	34	34	36	571	571	570	574
	Waiting time(sec)	22	22	21	17	128	128	128	126	19	19	18	18
	Buffer allocated	1199	1207	1218	1234	22658	22632	22658	21400	1273	1281	1297	1293
	Access bandwidth	70	70	69	69	44	44	44	41	70	70	72	70
	Execution time(ms)	51	58	58	57	12	12	12	11	17	17	17	16
R = 1.50	Accepted requests	598	598	599	600	33	33	33	33	596	596	598	598
	Waiting time(sec)	19	19	18	15	118	118	118	112	16	16	17	14
	Buffer allocated	1032	1038	1054	1086	25309	25309	25309	23481	1100	1108	1125	1137
	Access bandwidth	70	70	69	69	54	54	54	45	71	71	70	69
	Execution time(ms)	50	53	54	55	12	12	13	10	16	16	16	15

Replication ratio R = total number of movie copies / total number of movies
Accepted requests = total number of requests out of 600 whose presentation schedules can be made
Waiting time = start of presentation - customer request arrival time
Buffer allocated = total buffer space allocated during all presentations / number of accepted customers
Disk bandwidth = total diskbandwidth allocated during all presentations / number of accepted customers
Execution time = time spent for making a presentation schedule (measured for accepted customers)

Figure 6: Simulation Results : Static Buffer Allocation

Experiment Result Under Dynamic Buffer Allocation Strategy

ratio	schedule / buffer interval	Block-Oriented 0	Block-Oriented 30	Block-Oriented 60	Block-Oriented 90	Segment-Oriented 0	Segment-Oriented 30	Segment-Oriented 60	Segment-Oriented 90	Hybrid 0	Hybrid 30	Hybrid 60	Hybrid 90
R = 1	Accepted requests	498	500	500	506	84	82	82	78	490	490	489	493
	Waiting time(sec)	30	32	33	36	155	154	172	156	39	38	36	36
	Buffer allocated	1556	1566	1603	1581	10382	10251	10569	10938	1585	1590	1604	1611
	Access bandwidth	71	72	72	69	60	62	62	61	70	71	70	70
	Execution time(ms)	56	62	63	62	41	43	44	46	21	20	20	20
R = 1.30	Accepted requests	595	595	594	589	88	88	86	85	593	593	592	589
	Waiting time(sec)	22	22	21	17	131	131	127	130	22	22	21	17
	Buffer allocated	1199	1207	1218	1234	9623	9644	10193	10069	1221	1227	1246	1256
	Access bandwidth	70	70	69	69	57	57	62	63	70	70	69	69
	Execution time(ms)	54	61	62	62	37	37	38	39	19	19	18	18
R = 1.50	Accepted requests	598	598	599	600	82	82	87	86	598	598	600	599
	Waiting time(sec)	19	19	18	15	132	132	126	128	16	16	17	15
	Buffer allocated	1032	1038	1054	1086	10371	10371	9962	10151	1056	1064	1080	1099
	Access bandwidth	70	70	69	69	65	65	68	66	71	71	71	69
	Execution time(ms)	49	55	56	57	40	40	38	39	17	16	16	16

Replicatio ratio R = total number of movie copies / total number of movies
Accepted requests = total number of requests out of 600 whose presentaion schedules can be made
Waiting time = start of presentation - customer request arrival time
Buffer allocated = total buffer space allocated during all presentations / number of accepted customers
Disk bandwidth = total disk bandwidth allocated during all presentations / number of accepted customers
Execution time = time spent for making a presentation schedule (measured for accepted customers)

Figure 7: Simulation Results : Dynamic Buffer Allocation

3. **Hybrid Presentation Plan :** performance is more or less similar to that of block-oriented presentation plans. However, hybrid presentation plan performance improves with the usage of dynamic buffer allocation strategy (as compared to the static one). Also, the time required for computing hybrid presentation plans are much lower than that for block-oriented plans.

In summary, block-oriented and hybrid presentation plans perform more or less equally. However, hybrid presentation plans took less time to compute. Additionally, the number of commitment records to be maintained is significantly smaller than that for block-oriented plans. Hence, hybrid presentation plans seem to be the best option for distributed video presentations.

7 Properties of Presentation Plans

In the preceding section, we have presented *three* types of plans – *segment-oriented* presentation plans, *block-oriented* presentation plans, and *hybrid* presentation plans. These plans all enjoy some structural variations, but all of them are executable. In this section, we study the properties and inter-relationships between these different plans.

7.1 Relationship between Different Plans

In this section, we study the relationships between the different types of plans introduced earlier in this paper. While some of these results are generally expected and not surprising, they had previously been justified on *empirical* grounds. In contrast, here we are able to *formally prove* them, thus providing formal mathematical backing to some results that had hitherto been experimentally validated.

Proposition 7.1 For a network of VoD servers, it is the case that :

1. If PP is a block oriented presentation plan for delivering movie m to customer C via originating server s, then PP is also a segment oriented presentation plan for this task.
2. If PP is a segment oriented presentation plan for delivering movie m to customer C via originating server s, then there exists a hybrid presentation plan $PP^{\star}$ for this task which was the same wait time as PP. □

Theorem 7.1 Let BPP, SPP, HPP be *optimal* block oriented, segmented oriented, and hybrid presentation plans for delivering movie m to customer C via originating server s. Let $\mathsf{WAIT}_B, \mathsf{WAIT}_S, \mathsf{WAIT}_H$ be the customer wait times associated with BPP, SPP, HPP respectively. Then:

$$\mathsf{WAIT}_H = \mathsf{WAIT}_S = \mathsf{WAIT}_B.$$

□

The above results indicate that in order to minimize the waiting time for a customer, all three presentation plans yield plans that are equivalent in terms of their optimality properties. Thus, given our previous experimental results it is best to develop VoD servers that use the notion of *hybrid* presentation plans.

7.2 Properties of Presentation Plans with Changes in the Logical Network Layout and/or Resources

In this section, we study how the notion of a plan is affected when changes are made to the logical network.

Definition 7.1 Suppose $(V, E, \mathtt{bw}_1)$ and $(V, E, \mathtt{bw}_2)$ are logical networks. We say that $\mathtt{bw}_1 \preceq \mathtt{bw}_2$ iff

$$(\forall t, e_1, e_2)\ \mathtt{bw}_1((e_1, e_2), t) \leq \mathtt{bw}_1((e_1, e_2), t).$$

Intuitively, $\mathtt{bw}_1 \preceq \mathtt{bw}_2$ iff at all times t, the available bandwidth in any network link according to $\mathtt{bw}_2$ is at least as much as that according to $\mathtt{bw}_2$.

Theorem 7.2 (Effect of Increased Bandwidth) Suppose

$$\mathcal{NL}_1 = (V, E, \mathtt{bw}_1, \mathcal{MOVIE}, \wp) \text{ and } \mathcal{NL}_2 = (V, E, \mathtt{bw}_2, \mathcal{MOVIE}, \wp)$$

are two logical networks and suppose $\mathtt{bw}_1 \preceq \mathtt{bw}_2$. Suppose t is the task of delivering movie m to customer C via originating server s. Let BPP_i, SPP_i, HPP_i be *optimal* block-oriented, segment-oriented, hybrid presentation plans for task t w.r.t. $\mathcal{NL}_i$ $(i = 1, 2)$. Let $\mathsf{WAIT}_{B,i}, \mathsf{WAIT}_{S,i}$ and $\mathsf{WAIT}_{H,i}$ denote the customer wait w.r.t. BPP_i, SPP_i, HPP_i for $i = 1, 2$. Then:

$$\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}_{B,1}; \mathsf{WAIT}_{S,2} \leq \mathsf{WAIT}_{S,1}; \mathsf{WAIT}_{H,2} \leq \mathsf{WAIT}_{H,1}.$$

□

The theorem conclusively establishes that if a low-bandwidth line in the network is replaced by a higher bandwidth line, then our notion of a plan will pass the benefits on to the customer by diminishing the time s/he has to wait.

Definition 7.2 Suppose

$$\mathcal{NL}_1 = (V, E, \mathcal{MOVIE}, \wp_1) \text{ and } \mathcal{NL}_2 = (V, E, \mathtt{bw}, \mathcal{MOVIE}, \wp_2)$$

are two logical networks. We say that $\wp_1 \preceq \wp_2$ iff

$$(\forall m \in \mathcal{MOVIE})(\forall b)\wp_1(m, b) \subseteq \wp_2(m, b).$$

This definition basically says that $\wp_1 \preceq \wp_2$ iff whenever site s contains block b of movie m according to placement mapping $\wp_1$, then site s also is considered to have block b of movie m according to placement mapping $\wp_2$. However, $\wp_2$ may place extra blocks of one or more movies at site s.

Theorem 7.3 (Effect of Increased Replication) Suppose

$$\mathcal{NL}_1 = (V, E, \mathtt{bw}_1, \mathcal{MOVIE}, \wp) \text{ and } \mathcal{NL}_2 = (V, E, \mathtt{bw}_2, \mathcal{MOVIE}, \wp)$$

are two logical networks and suppose $\wp_1 \preceq \wp_2$. Let BPP_i, SPP_i, HPP_i, $\mathsf{WAIT}_{B,i}$, $\mathsf{WAIT}_{S,i}$ and $\mathsf{WAIT}_{H,i}$ be defined as in Theorem 12.2. Then:

$$\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}_{B,1}; \mathsf{WAIT}_{S,2} \leq \mathsf{WAIT}_{S,1}; \mathsf{WAIT}_{H,2} \leq \mathsf{WAIT}_{H,1}.$$

The above theorem shows that if we "increase" the placement function, then we are guaranteeing the customer a smaller wait time. The paper [7] provides an experimental claim that caching initial segments of movies at servers leads to improved performance as compared to not doing so. Theorem 7.3 is a significant improvement on that result for the following reasons:

1. First, our result is a *theorem* rather than an experimental observation.
2. Second, our result does not apply *just to initial segments* of movies. In fact, it is entirely possible that one or more interim blocks are added to a server by $\wp_2$ and this may still lead to a lower wait time. The example shown in Figure 5 described how this may happen.
3. Third, our result applies to all three notions of plans, not just the one studied in [7].

Suppose $\mathcal{NL} = (V, E, \mathtt{bw}_1, \mathcal{MOVIE}, \wp)$ is a logical network layout. We say that $\mathtt{buf}_1 \preceq \mathtt{buf}_2$ iff for all $v \in V$, $\mathtt{buf}_1(v) \leq \mathtt{buf}_2(v)$. The following theorem says that increases in buffer space lead to diminished wait times for the customer.

Theorem 7.4 (Effect of Increased Buffer Space) Suppose $\mathcal{NL} = (V, E, \mathtt{bw}_1, \mathcal{MOVIE}, \wp)$ is a logical network layout. Suppose that $\mathtt{buf}_1 \preceq \mathtt{buf}_2$. Let BPP_i, SPP_i, HPP_i, $\mathsf{WAIT}_{B,i}, \mathsf{WAIT}_{S,i}$ and $\mathsf{WAIT}_{H,i}$ be defined as in Theorem 12.2. Then:

$$\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}_{B,1}; \mathsf{WAIT}_{S,2} \leq \mathsf{WAIT}_{S,1}; \mathsf{WAIT}_{H,2} \leq \mathsf{WAIT}_{H,1}.$$

8 Related Work

Issues in the design of a video on-demand server have been dealt with in [4]. The emphasis has been on scheduling mechanisms for disk accesses that significantly lower the buffer-size requirements in the case of disk arrays. Issues in the design of multi-user HDTV storage server have been discussed in [11]. In contrast, we deal with the construction of presentation plans to deliver videos across distributed networked sites. Our framework may, for instance, use characteristics of the HDTV storage servers of [11] in creating distributed presentation plans. We do not restrict ourselves to the type of movies stored (HDTV or normal).

Data access strategies in a high performance Multimedia-on-Demand server have been discussed in [10, 24, 18, 7]. Here, algorithms for a multimedia server operation for retrieval of remote media objects are presented. The algorithms also exploit knowledge of data access patterns to improve system throughput. Experimental results have been provided to establish the performance of the algorithms. In our work, we deal with algorithms for computing different presentation plans in the case where movie blocks are distributed over a set of servers. The three types of presentation plans we have proposed are novel, and the algorithms to construct them are new, as is the experimental; result establishing the superiority of hybrid presentation plans. In addition to our experimental results, we have also proved *mathematically*, a number of results that had only had experimental validation previously[7].

[12, 13, 23] discuss the network requirements for multimedia-on demand. [17] presents resource reservation schemes for guaranteeing network throughput. [14] describes how retrieval schedules can be determined by a client based on flexible temporal specifications of multimedia document presentation. In our work, we deal with creating presentation plans for distributed video data. We assume the network to provide guaranteed throughput for VoD presentation.

Caching of movie blocks has been described in [1]. They also provide valuable user access patterns of movies derived from a real-life video rental store data. In our work, we use the same access pattern for our experiments.

9 Conclusions

There are a vast number of applications of video-on-demand systems, ranging from sophisticated home entertainment systems, to educational on demand programs where users at remote locations (e.g. on ships, on in the isolated areas of Montana) wish to access videos of lectures, at their leisure. Furthermore, in the rapidly emerging area of multimedia databases [22] and video databases [20, 9, 19, 21, 8], users may query a large *distributed* multimedia archive and retrieve the desired videos (in part, or in whole) across the network.

Commercial vendors who support such applications will, in all likelihood, use a distributed set of servers for the simple reason that distributed systems are less likely to experience system wide failures than a centralized system. In effect, what this means is that video data will be distributed across a network (proprietary, or open) that will be accessed by customers.

In this paper, we have provided a distributed VoD architecture that supports customer-server and server-serve interactions. When a customer requests his/her server to deliver a movie to him/her, the server constructs a *presentation plan*. Informally put, a presentation plan specifies what the server must retrieve, when it must retrieve it, the rate at which it will retrieve it, and where (local or remote) it will retrieve it from. Presentation plans cannot be constructed completely autonomously by the customer's local server: rather the local server must interact with remote serves and the network service provider to ensure that they all agree to commit the required resources. In this paper, we provide a formal foundation for creating presentation plans – specifically, we formally define three types of presentation plans and define how these plans may be measure/evaluated using customer-wait times, and access bandwidths associated with the plan. We develop an algorithm to compute optimal presentation plans of all three types, and implement the algorithms in a simulation of a distributed VoD system. Using data obtained from a video store to characterize access patterns for video rentals, we derive experimental results showing that the notion of a hybrid presentation plan seems to be the best of the three types of plans.

Acknowledgements. We are grateful to Asit Dan for providing us with the data in [25].

References

[1] A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads", *Multimedia Computing and Networking*, San Jose, January 1996.

[2] A. Dan, M. Kienzle, and D. Sitaram, "A Dynamic Policy of Segment Replication for Load-balancing in Video-On-Demand Servers", *ACM/Springer-Verlag Multimedia Systems*, 1995.

[3] A. Drapeau, D. Patterson, R. Katz, "Toward Workload Characterization of Video Server and Digital Library Applications", *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Nashville, May 1994.

[4] A.N. Mourad, "Issues in the Design of a Storage Server For Video-on-Demand", *ACM/Springer-Verlag Multimedia Systems*, (4), 1996, pp. 70-86.

[5] C.H. Papadimitriou, S. Ramanathan, and P. Venkat Rangan, "Information Caching for Delivery of Personalized Video Programs on Home Entertainment Channels", *Proceedings of IEEE Multimedia*, 1995.

[6] D. Ferrari, A. Gupta, G. Ventre, "Distributed advance reservation of real-time connection", *Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, April, 1995.

[7] D. Jaday, C. Srinilta, A. Choudhary, P.B. Berra, "Design and Evaluation of Data Access Strategies in a High Performance Multimedia-on-Demand Server", *Proceedings of IEEE Multimedia*, 1995.

[8] D. Rotem and J.L. Zhao, "Buffer Management for Video Database Systems", *Proceedings of IEEE Intl. Conference on Data Engineering*, 1995, pps 439-448.

[9] E. Oomoto and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System", *IEEE Transactions on Knowledge and Data Engineering*, Aug. 1993, 5, 4, pps 629–643.

[10] G. Miller, G. Baber, and M. Gilliland, "News On-Demand Multimedia Networks", *Proceedings of ACM Multimedia*, Anaheim, CAA, August 1993, pp. 383-392.

[11] H.M. Vin and P.V. Rangan, "Design of a Multi-user HDTV Storage Server", *IEEE Journal on Selected Areas in Communication*, Special Issue on High Definition Television and Digital Video Communication, Vol. 11, No. 1, January 1993.

[12] J. Nussbaumer, B. Patel, F. Schaffa, and J.P.G. Sterbenz, "Networking Requirements for Interactive Video on Demand", *IEEE Journal on Selected Areas in Communication*, January 1995.

[13] K. Nahrstedt and Ralf Steinmetz, "Resource Management in Networked Multimedia Systems", *IEEE Computer*, Vol. 28, No. 4, April 1995.

[14] K. S. Candan, B. Prabhakaran, and V. S. Subrahmanian, "CHIMP : A Framework for Supporting Multimedia Document Authoring and Presentation", to appear in *Proceedings of ACM Multimedia*, Boston, November 1996.

[15] K.S. Candan, E. Hwang and V.S. Subrahmanian, "An Event-Based Model for Continuous Media Data on Heterogeneous Disk Servers", *ACM Multimedia Systems Journal*, accepted, to appear.

[16] L.C. Wolf, L. Delgossi, R. Steinmetz, S. Schaller, H. Wittig, "Issues of Reserving Resources in Advance", *Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, April, 1995.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol", *IEEE Network*, September 1993, pp.8-18.

[18] M. Budhikot, G. Parulkar, and J.R. Cox Jr, "Design of a Large Scale Video Server", *Journal of Computer Networks and ISDN Systems*, December 1994, pp. 504-517.

[19] R. Hjelsvold and R. Midtstraum. "Modeling and Querying Video Data", *Proceedings Intl. Conference on Very Large Databases*, Santigo, Chile, Sep. 1994, pps 686–694.

[20] S. Adali, K.S. Candan, S.-S. Chen, K. Erol, and V.S.Subrahmanian, "Advanced Video Information Systems", *ACM Multimedia Systems Journal*, 4, pps 172–186, 1996.

[21] S. Gibbs, C. Breiteneder and D. Tsichritzis, "Data Modeling of Time-Based Media", *Proceedings of ACM SIGMOD Conference on Management of Data*, Minneapolis, Minnesota, June 1994, pps 91–102.

[22] S. Marcus and V.S. Subrahmanian, "Foundations of Multimedia Database Systems", *Journal of the ACM*, Vol. 43, 3, pps 474–523, 1996.

[23] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi "Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation", *IEEE Journal on Selected Areas in Communication*, Special issue on Multimedia Synchronization, Vol. 14, No. 1, January 1996.

[24] T.D.C. Little, *et al*, "A Digital On-demand Video Service Supporting Content-based Queries", *Proceedings of ACM Multimedia*, Anaheim, CA, August 1993, pp. 427-436.

[25] *Video Store Magazine*, Dec. 13, 1992.

10 Appendix I : Algorithm For Creating Presentation Plan

SSR : **Set of Schedule Requests** ;
NSB : **Next Start Block number which should be scheduled** ;
MPD : **Movie Presentation Deadline** ;
NPD : **Next Presentation Deadline** ;
WS : **Window Size during which caching is done** ;
δt : **initial overhead time** ;
OptSchedule : **optimal movie presentation schedule** ;

```
GenOptSchedule ( WS, request )
{
        sttime = request.arr_time + δt ;
        Finished = FALSE ;
        While ( sttime ≤ MPD and Finished != TRUE )
        {
                delay = 0 ;
                NSB = 0 ;
                NPD = sttime ;
                OptSchedule = ∅ ;
                Again = FALSE ;
                While ( NSB < Totbnum ( request.mov ) and Again != TRUE )
                {
                        CommRec =check_LocalCache ( NSB, NPD, request ) ;
                        If ( CommRec.flag == SUCCESS )
                        {
                                OptSchedule = OptSchedule ∪ CommRec ;
                                NSB = NSB + CommRec.blknum ;
                                NPD = NPD + CommRec.playtime ;
                        }
                        Else
                        {
                                SSR = identify_TargetServer ( request, NSB);
                                        /* identify all the servers and segments containing NSB */
                                sort_ScheduleRequest  (SSR) ;
                                        /* sort schedule requests by the number of blocks */
                                Scheduled = FALSE ;
                                While (SSR != ∅ and Scheduled != TRUE)
                                {
                                        sr = next schedule request in SSR
                                        CommRec = request_ScheduleToTarget ( sr, NPD ) ;
                                                /* request scheduling of data blocks to the target */
                                                /* server within the deadline. */
                                        If ( CommRec.flag == SUCCESS and check_LocalBuf ( CommRec ) == OK )
                                                Scheduled = TRUE ;
                                }
                                If ( Scheduled == TRUE )
                                {
                                        OptSchedule = OptSchedule ∪ CommRec ;
```

```
                    NSB = NSB + CommRec.blknum ;
                    NPD = NPD + CommRec.playtime ;
                }
                Else
                {
                    cancel_CommRec ( OptSchedule ) ;
                    delay = comp_delay(SSR) ;
                    sttime = sttime + delay ;
                    Again = TRUE ;
                }
            }
        }
        If ( NSB ≥ Totbnum( request.mov) )
            Finished = TRUE ;
    }
    If ( Finished == TRUE )
        return (OptSchedule) ;
    Else
        Reject Request ;
}
```

11 Appendix II : Algorithm For Handling Schedule Request

---

btime : **block transfer time** ;
interval : **time duration** ;
connId : **established network connection identifier** ;
bandwidth : **bandwidth available in network or disk** ;
net_resv : **specifies interval and network throughput** ;
disk_resv : **specifies interval and disk throughput** ;
cache_resv : **specifies blocks in a segment available in cache** ;
CommRec : **commitment record for the request** ;

```
handle_ScheduleRequest ( sr, connId, NPD )
/* this routine is called within request_ScheduleToTarget() to handle */
/* the schedule request. connId is given as a result of connection establishment */
{
    net_resv = retrieve_netConnectionStatus ( connId ) ;
        /* retrieve information about the network connection establishment */
        /* requested with required bandwidth and connection duration */
    bandwidth = net_resv.bandwidth ;
    btime = block_size / bandwidth ;
    blocks = comp_netCapacity ( net_resv.interval, btime ) ;
    sr.endblk = sr.stblk + blocks - 1 ;
    tmp_schedule = ∅
    For ( b_id = sr.stblk ; b_id ≤ sr.endblk ; b_id++ )
    {
        blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime ) ;
        cache_resv = lookup_cacheTaskTable ( sr.mov, b_id, blk_deadline ) ;
            /* checks the cache table for the movie and segment and record */
```

```
            /* blocks hit in the cache during interval */
        If ( cache_resv.status == hit )
        {
            tmp_schedule = append_BlkSchedule ( cache_info ) ;
            blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime ) ;
            continue ;
        }
        disk_resv = lookup_diskScheduleTable ( sr.mov, b_id, blk_deadline ) ;
            /* look up the disk schedule table to see if the requested block */
            /* can be scheduled within the block deadline */
        If ( disk_resv.status == rejected )
            If ( tmp_schedules == ∅ )
            {
                CommRec.flag = FAIL ;
                return ( CommRec ) ;
            }
            Else
            {
                CommRec = creat_CommRec ( sr, b_id ) ;
                CommRec.flag = SUCCESS ;
                CommRec.schedule = generate_Schedules ( tmp_schedules ) ;
                return ( CommRec ) ;
            }
        tmp_schedule = append_BlkSchedule ( disk_info ) ;
        blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime ) ;
    }
    CommRec = creat_CommRec ( sr, b_id ) ;
    CommRec.flag = SUCCESS ;
    CommRec.schedule = generate_Schedules ( tmp_schedules ) ;
    return ( CommRec ) ;
}
```

12 Appendix III : Proofs of Results

Proposition 12.1 For a network of VoD servers, the following claims can be made :

1. If PP is a block oriented presentation plan for delivering movie m to customer C via originating server s, then PP is also a segment oriented presentation plan for this task.
2. If PP is a segment oriented presentation plan for delivering movie m to customer C via originating server s, then there exists a hybrid presentation plan $PP^{\star}$ for this task which was the same wait time as PP.

Proof. (1) If b is a block, then $[b, b]$ is a segment. As the structure used to represent BOPPs and SOPPs is the same, it follows that each record in PP represents a segment $[b, b]$ and hence, as PP satisfies Constraints (1)–(3) in the definition of a feasible SOPP, the result follows immediately.
(2) Suppose PP is a SOPP. Let r be a record in PP. We will show how we may construct a presentation record $r^{\star}$ from r that has the structure of a hybrid presentation record and that is feasible iff r is feasible and such that r and $r^{\star}$ have the same customer wait time. This establishes the result directly.

Given a segment based presentation record r, we construct $r^{\star}$ as follows: $r^{\star}$'s Orig, Target, Movie, Start, End, Reqtime, ConOK BWAssign and DelivSt fields are set to those of r. In addition:

1. For all $r.Start \leq i < r.End$, $r^{\star}.DelivEnd[i] = r^{\star}.Deliv + \frac{i - r.Start+1) \times \mathtt{bsize}}{r^{\star}.BWAssign}$.
   $r^{\star}.DelivEnd[r.End] = r.DelivEnd.$
2. $r^{\star}.CustShipSt[r.Start] = r.CustShipSt.$
   For all $r.Start < i \leq r.End$, $r^{\star}.CustShipSt[i] = r^{\star}.CustShipSt[i-1] + \frac{\mathtt{bsize}}{r^{\star}.BWAssign}$.
3. For all $r.Start \leq i < r.End$, $r^{\star}.CustShipEnd[i] = r^{\star}.CustShipSt[i] + \frac{\mathtt{bsize}}{\mathtt{bw}(r.Orig,C)}$.
   $r^{\star}.CustShipEnd[r.End] = r.CustShipEnd.$
4. For all $r.Start \leq i \leq r.End$, $r^{\star}.CustConsStart[i] = r.CustConsStart + \frac{(i - r.Start+1) \times \mathtt{bsize}}{\mathtt{ccr}(C)}$.
5. For all $r.Start \leq i < r.End$, $r^{\star}.CustConsEnd[i] = r.CustConsStart[i] + \frac{\mathtt{bsize}}{\mathtt{ccr}(C)}$.
   $r^{\star}.CustConsEnd[r.End] = r.CustConsEnd.$

If $PP = r_1, \ldots, r_k$ is a SOPP, then let $PP^{\star}$ be the HPP $r_1^{\star}, \ldots, r_k^{\star}$. It follows immediately by construction that PP is feasible iff $PP^{\star}$ is feasible[1] Furthermore, from item(4) above, it follows that the customer wait time associated with r and $r^{\star}$ is identical. □

Theorem 12.1 Let BPP, SPP, HPP be *optimal* block oriented, segmented oriented, and hybrid presentation plans for delivering movie m to customer C via originating server s. Let $\mathsf{WAIT}_B, \mathsf{WAIT}_S, \mathsf{WAIT}_H$ be the customer wait times associated with BPP, SPP, HPP respectively. Then:

$$\mathsf{WAIT}_H = \mathsf{WAIT}_S = \mathsf{WAIT}_B.$$

[1]It should be noted that in steps (4) and (5) of the above construction, equivalence of plans is not preserved, but feasibility of plans and the corresponding customer wait times is preserved.

Proof. By Proposition 12.1, it follows immediately that

$$\mathsf{WAIT}_H \leq \mathsf{WAIT}_S \leq \mathsf{WAIT}_B.$$

To show that $\mathsf{WAIT}_H = \mathsf{WAIT}_S = \mathsf{WAIT}_B$, it suffices therefore to show that $\mathsf{WAIT}_B \leq \mathsf{WAIT}_H =$.

Suppose this is not the case, i.e. $\mathsf{WAIT}_H < \mathsf{WAIT}_B$. In this case, we will show that we can construct from HPP, a block oriented plan, BPP_1 such that the wait time, $\mathsf{WAIT}_{B,1}$ associated with $BPP_1 = \mathsf{WAIT}_H < \mathsf{WAIT}_B$ thus contradicting the assumption (in the theorem statement) that BPP is an optimal block oriented plan.

The construction is as follows: suppose r is any presentation record in HPP and suppose $(r.End - r.Start + 1) = k$. BPP_1 will then contain k block oriented records $r_0, \ldots, r_{k-1}$ obtained from r as follows.

1. $r_i.Orig = r.Orig$ for all $0 \leq i \leq (k-1)$;
2. $r_i.Target = r.Target$ for all $0 \leq i \leq (k-1)$;
3. $r_i.Movie = r.Movie$ for all $0 \leq i \leq (k-1)$;
4. $r_i.Start = r_i.End = r.Start + i$ for all $0 \leq i \leq (k-1)$.
5. $r_i.Reqtime = r.Reqtime$;
6. $r_i.ConOK = r.Reqtime + i \times ct(r.Orig, r.Target)$.
7. $r_i.BWassign = r.BWAssign$;
8. $r_i.DelivSt = r.DelivSt + \frac{i \times \mathtt{bsize}}{r.BWAssign}$;
9. $r_i.DelivEnd = r.DelivEnd[i]$;
10. $r_i.CustShipSt = r.CustShipSt[i]$;
11. $r_i.CustShipEnd = r.CustShipEnd[i]$;
12. $r_i.CustConsStart = r.CustConsStart[i]$;
13. $r_i.CustConsEnd = r.CustConsEnd[i]$.

The above construction yields a valid block-oriented presentation plan whose wait time W is equal to WAIT_H. As WAIT_B is an *optimal* block oriented presentation plan,it follows that

$$\mathsf{WAIT}_B \leq W = \mathsf{WAIT}_H.$$

We have thus shown that $\mathsf{WAIT}_B \leq \mathsf{WAIT}_H$ and $\mathsf{WAIT}_H \leq \mathsf{WAIT}_B$, implying that $\mathsf{WAIT}_B = \mathsf{WAIT}_H$. As $\mathsf{WAIT}_H \leq \mathsf{WAIT}_S \leq \mathsf{WAIT}_B$, it follows that

$$\mathsf{WAIT}_B = \mathsf{WAIT}_S = \mathsf{WAIT}_H.$$

□

Theorem 12.2 (Effect of Increased Bandwidth) Suppose

$$\mathcal{NL}_1 = (V, E, \mathtt{bw}_1, \mathcal{MOVIE}, \wp) \text{ and } \mathcal{NL}_2 = (V, E, \mathtt{bw}_2, \mathcal{MOVIE}, \wp)$$

are two logical networks and suppose $\mathtt{bw}_1 \preceq \mathtt{bw}_2$. Suppose t is the task of delivering movie m to customer C via originating server s. Let BPP_i, SPP_i, HPP_i be *optimal* block-oriented, segment-oriented, hybrid presentation plans for task t w.r.t. $\mathcal{NL}_i$ $(i = 1, 2)$. Let $\mathsf{WAIT}_{B,i}, \mathsf{WAIT}_{S,i}$ and $\mathsf{WAIT}_{H,i}$ denote the customer wait w.r.t. BPP_i, SPP_i, HPP_i for $i = 1, 2$. Then:

$$\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}_{B,1}; \mathsf{WAIT}_{S,2} \leq \mathsf{WAIT}_{S,1}; \mathsf{WAIT}_{H,2} \leq \mathsf{WAIT}_{H,1}.$$

Proof. We will prove below that $\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}_{B,1}$. The proofs that $\mathsf{WAIT}_{S,2} \leq \mathsf{WAIT}_{S,1}$ and $\mathsf{WAIT}_{H,2} \leq \mathsf{WAIT}_{H,1}$ are exactly analogous.

Let $BPP_1 = r_1, \ldots, r_k$. We will show that when the bandwidth considered is increased from $\mathtt{bw}_1$ to $\mathtt{bw}_2$, then there *exists* a block oriented plan BOPP' such that the wait $\mathsf{WAIT}(\mathsf{BOPP}')$ associated with BOPP' is equal to $\mathsf{WAIT}_{B,1}$. As BPP_2 is an *optimal* such plan, it follows that $\mathsf{WAIT}_{B,2} \leq \mathsf{WAIT}(\mathsf{BOPP}') = \mathsf{WAIT}_{B,1}$ which proves the result.

We can construct $\mathsf{BOPP}' = r_1', \ldots, r_k'$ as follows by replacing each record r_i by a new record r_i'.

1. For all $1 \leq i \leq k$, $r_i'.Orig r_i'.Target, r_i'.Movie, r_i'.Start, r_i'.End, r_i'.Reqtime,$ $r_i'.ConOK$ are identical to the corresponding fields in record r_i.
2. For all $1 \leq i \leq k$, $r_i'.BWAssign = \mathtt{bw}_2(r_i.Target, r_i.Orig, t)$. (Recall that this value is *higher* than the bandwidth $\mathtt{bw}_1(r_i.Target, r_i.Orig, t)$.
3. $r_i'.DelivEnd = r_i.DelivEnd$.
4. $r_i'.DelivSt = r_i.DelivEnd - \frac{\mathtt{bsize}}{r_i'.BWAssign}$.
5. $r_i'.CustShipEnd = r_i.CustShipEnd$;
6. $r_i'.CustShipSt = r_i'.CustShipEnd - \frac{\mathtt{bsize}}{\mathtt{bw}(r.Orig, C, t_{r_i'.CustShipSt})}$.
7. $r_i'.CustConsSt = r_i'.CustShipEnd$;
8. $r_i'.CustConsEnd = r_i'.CustConsSt + \frac{\mathtt{bsize}}{\mathtt{ccr}(C)}$. $\square$