분산 VoD 시스템을 위한 프리젠테이션 플랜닝

황 인 준† · 변 광 준††

요 약

분산 Video-on-Demand(VoD) 시스템에서는 비디오 데이터가 컴퓨터 네트워크상에 흩어져 존재하게 된다. 단일 사이트 환경에서는 로컬 비디오 서버가 자신의 로컬 저장 장치로부터 비디오 데이터를 얻는다. 하지만 분산 VoD 시스템 환경에서는 사용자가 로컬 서버에게 영화와 같은 비디오 데이터를 요청할 때, 서버는 경우에 따라 네트워크상의 다른 서버들과 접촉하게 된다. 본 논문에서는 사용자의 요청을 만족시키기 위하여 로컬 서버가 구성할 수 있는 세가지 유형의 프리젠테이션 플랜을 제시한다. 프리젠테이션 플랜은 요청된 비디오 데이터를 사용자에게 보이주기 위해 로컬 서버가 수행하는 동기화된 일련의 상세한 스텝들을 말한다. 이것은 사용 가능한 대역폭 및 버퍼, 사용자의 데이터 소비 비율의 범위 내에서 로컬 자원들에 대한 획약뿐만 아니라 다른 비디오 서버들, 그리고 네트워크 서비스 제공자로부터의 획약을 얻는 작업을 포함한다. 또한, 서로 다른 프리젠테이션 플랜들의 성능을 비교하기 위해 두 가지 기준, 다시 말해 사용자의 대기 시간 최소화와 사용된 네트워크/디스크 대역폭을 나타내는 접근 대역폭의 축소화를 도입한다. 본 연구에서는 세 유형에 따른 최적 프리젠테이션 플랜을 계산하는 알고리즘을 제시하고 이들의 성능을 비교하기 위하여 다양한 조건하에서 여러가지 실험을 수행하였다. 그리고 제시된 방법을 바탕으로 그 동안 실험적인 환경하에서 입증되었던 프리젠테이션 플랜들에 대한 일부 결과들을 수학적으로 입증하였다.

Presentation Planning for Distributed VoD Systems

Eenjun Hwang† · Kwang-June Byeon††

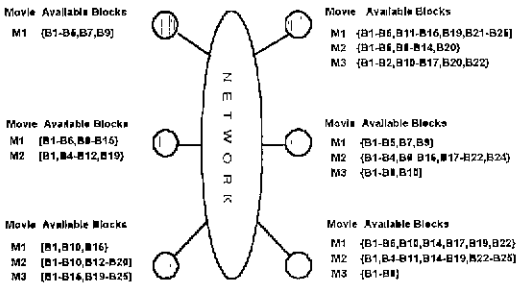
ABSTRACT

A distributed video-on-demand (VoD) system is one where 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. However, in the setting of a distributed VoD system, when a customer requests a movie from the 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 request. Informally speaking, a presentation plan is a temporally synchronized detailed sequence of steps that the local server must perform for presenting the requested movie to the customer. This involves obtaining commitments from other video servers, obtaining commitments from the network service provider, as well as making commitments of local resources, within the limitations of available bandwidth, available buffer, and customer data consumption rates. Furthermore, for evaluating the goodness of a presentation plan, we introduce two measures of optimality for presentation plans: minimizing wait time for a customer, and minimizing access bandwidth which informally speaking, specifies how much network/disk bandwidth is used. We develop algorithms to compute optimal presentation plans for all three types, and carry out extensive experiments to compare their performance. We have also mathematically proved certain results for the presentation plans that had previously been verified experimentally in the literature.

† 정 회 원 : 아주대학교 정보및컴퓨터공학부 교수
†† 종신회원 : 아주대학교 정보및컴퓨터공학부 교수
논문접수 : 1999년 12월 24일, 심사완료 : 2000년 2월 2일

1. Introduction

With the advent of sophisticated, yet cheap digitization technology and advances in networking technology, there is now a tremendous amount of interest in distributed VoD systems. In distributed VoD systems, the video data is typically located at multiple sites on the network. Each site has a local video server with possibly different amount of resources such as buffer space and disk bandwidth. Likewise, the customers who wish to access this data are located at geographically dispersed sites, and use some kind of local display device to view the video presentation. When the customer contacts the 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 communication channel between the local server and the customer. If the local server does not have the entire movie, it must request appropriate parts of the movie from one or more remote servers. This requires precise and careful planning on how to get those data from remote sites in a continuous way.

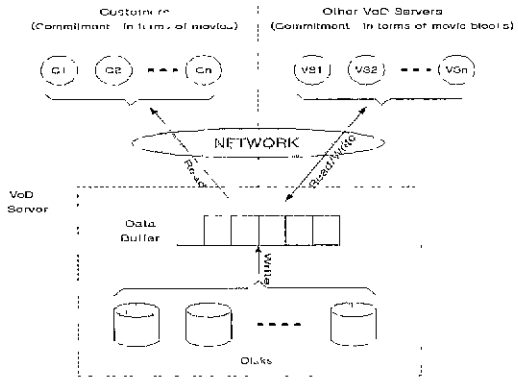


(Figure 1) Movie Placement on VoD Servers

In this paper, we first 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 one site. Movies may be stored in part, or as a whole, at one or more sites. We then formally define three types of presentation plans that a server could generate for customer requests. These presentation plans either describe how the movie will be delivered to the customer at a block level, or at a segment level, or at a hybrid of the two. We introduce two criteria in order to measure how good a presentation plan is. The first criteria is defined merely in terms of how long the customer has to wait, before viewing a jitter-free presentation. However, serving one customer may cause a future customer to be denied or delayed service, and hence, as alternative criteria, the amount of access bandwidth used for the presentation is proposed. We then provide a novel algorithm to compute optimal presentation plans with respect to these two criteria. Finally, we describe the results of simulations of VoD servers that we want to compare three types of plans under different buffering schemes.

2. VoD Server Architecture

The architecture of the VoD server considered in our work is shown in (Figure 1). Movie 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 $MOVIE$ 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

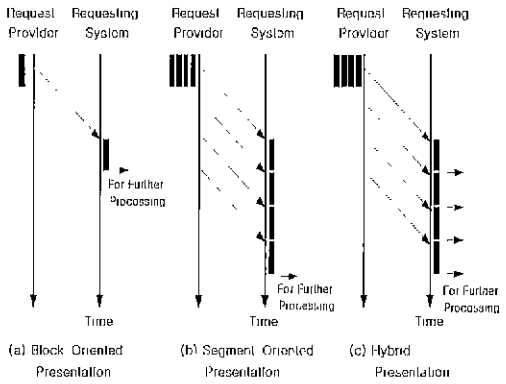
Definition 2.1 (Placement Mapping) A *placement mapping* is a mapping, δ that takes a movie $m_i \in MOVIE$ and a block number b as input, and returns a subset

of V as output

Typically, a VoD server serves a set of customers at any given point in time. A movie requested by a customer may or may not be available entirely on the server. If the requested movie is not available on the server, the server interacts with other VoD servers to obtain the missing blocks.

(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, both customers and other VoD servers may read the movie blocks located at one server. We introduce three types of frameworks that a VoD server may use in order to read and write movie blocks to its buffer.

- *Access movie blocks individually*: Here, customers and other VoD servers 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 segment of contiguous blocks*: Here, customers and other VoD servers download movies in a specified set 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.



(Figure 3) Types of Movie Presentation

- *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 3(c)), the requesting system can start using a block in the segment 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 Interactions

As discussed above, a VoD server may interact with remote VoD servers for downloading movie blocks. The transfer of movie blocks may be made either individually or in terms of segments. A presentation plan contains a detailed schedule specifying what a server must do in order to satisfy the customer request. When a VoD server constructs a presentation plan for a requesting system, it must be done based on certain criteria of optimality. 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*: Here, 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 presentation plan is generated in such a way that the amount of bandwidth used, both at the disk level and at the network level for buffering the movie blocks in the VoD server is minimized.

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 some of the customer capabilities such as customer consumption rate, buffer space, network bandwidth between the customer and the server are made known.

4. Presentation Record: Data Structure

When processing a customer request for a movie, the VoD server has to identify where the desired movie blocks are stored. 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. The presentation record has two sets of fields that describe: (i) the interaction with another VoD 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 video blocks
- Downloading the video 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, 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 in <Table 2> describe the parameters for interacting with a customer.

4.1 Presentation Records for Different Plans

Different presentation plans 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.

〈Table 1〉 Presentation Record - For Interaction With Target VoD Servers

Orig	specifies the server that originated the request
Target	specifies the server that will satisfy the request
Movie	specifies the movie associated with the request
Start	specifies the first movie block being requested
End	specifies the last movie block being requested
ReqTime	specifies the time at which block request is initiated
ConOK	specifies the time the connection is successfully made
BWAssign	specifies the bandwidth assigned to the request by the target
DelivSt	specifies the time block delivery starts
DelivEnd	specifies the time block delivery ends

〈Table 2〉 Presentation Record : For Interaction With Customers

CustShipSt	specifies when the originating server starts shipping the blocks to the customer
CustShipEnd	specifies when the originating server starts finishes shipping the blocks to the customer
CustConsSt	specifies when the customer starts consuming the blocks
CustConsEnd	specifies when the customer finishes consuming the blocks

It should be noted 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.

〈Table 3〉 Segment-Oriented Presentation Record

Orig, Target	
Movie	
Start, End	End > Start, which says a segment has more than one block
Reqtime	t_{Req}(Orig, Target, Movie, Start, End)
ConOK	r.conOK = r.Reqtime + ct (t.Orig, r.Target)
BWAssign	r.BWAssign ≤ bw (r.Target, r.Orig)
DelivSt	r.DelivSt = t.conOK
DelivEnd	r.DelivEnd = r.DelivSt + (t.Start-r.End+1) * bsize / r.BWAssign
CustShipSt	t.CustShipSt ≥ r.DelivEnd
CustShipEnd	r.CustShipEnd = r.CustShipSt + (r.End-r.Start+1) * bsize / bw(r.Orig, C)
CustConsSt	r.CustConsStart ≥ r.CustShipEnd
CustConsEnd	r.CustConsEnd = t.CustConsStart + (r.End-r.Start+1) * bsize / cer(C)

〈Table 4〉 Hybrid Presentation Record

DelivEnd	For each block b_w where r.Start ≤ w ≤ r.End, r.DelivEnd[w] = t.DelivSt (w-r.Start-1) * bsize / r.BWAssign
CustShipSt	For each block b_w where r.Start ≤ w ≤ t.End r.CustShipSt[w] ≥ r.DelivEnd[w]
CustShipEnd	For each block b_w where r.Start ≤ w ≤ r.End, t.CustShipEnd[w] = r.CustShipSt - bsize / bw(r.Orig, C)
CustConsStart	For each block b_w where r.Start ≤ w ≤ r.End, r.CustConsSt[w] ≥ r.CustShipEnd[w]
CustConsEnd	For each block b_w where r.Start ≤ w ≤ r.End, r.CustConsEnd[w] = r.CustConsSt [w]+ bsize / cer (C)

4.2 Feasible Presentation Plans

A VoD server must create and maintain a presentation plan for each customer request. This presentation plan can be one of the three types described earlier. Any presentation plan must ensure the following conditions

- A commitment must have been obtained from the originating server to ship movie blocks to the customer so that the movie can be watched without any interruptions
- Commitments must have been obtained from the target server for providing movie blocks to the originating server when all the movie blocks are not available locally to the originating server.
- Commitments 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. <Table 5> describes the fields associated with a commitment record.

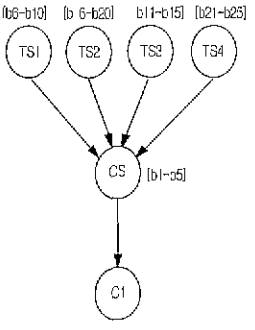
〈Table 5〉 Commitment Record

BegCom	specifies the start time of a commitment
FinCom	specifies the finish time of a commitment
Client	either a customer, or another server
Movie	specifies what movie forms part of the commitment
BlockSt	specifies the starting block of the movie
BlockEnd	specifies the ending block of the movie
BWCom	specifies the amount of bandwidth allocated to this commitment

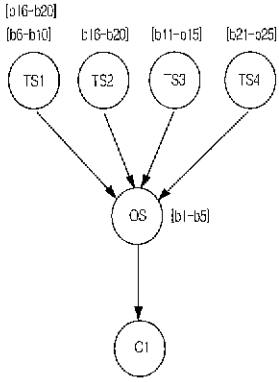
Informally speaking, a presentation plan is said to be feasible if the load on the originating and the target servers are such that the customer request can be handled by them and the buffer space at the customer site is enough for downloading the movie.

Example : Feasible Presentation Plan

(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 at more than one site.



(a) Without Replication



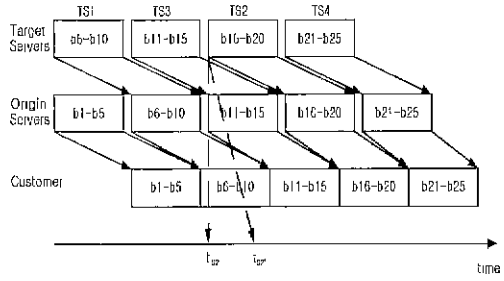
(b) With Replication

(Figure 4) Serving A Typical Customer

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-b5 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 *OS* needs the blocks. 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 now formally define what it means for a segment/block oriented presentation plan to be feasible. The feasibility for hybrid presentation plan can be defined in a similar way.

Definition 4.1 (Feasible Segment/Block Oriented Presentation Plan) Suppose $PP = r_i, .., r_k$ is a segment-oriented (resp. block-oriented) presentation plan for delivering movie m to customer C via originating server v. A presentation plan 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 $1 \le i \le k$, insert the tuple ($r_i.DelivSt$, $r_i.DelivEnd$, v, $r_i.Movie$, $r_i.Start$, $r_i.End$, $(r_i.End - r_i.Start + 1) * bsize$) into $CRL(r_i.Target)$.



(Figure 5) A Feasible Presentation Plan

Constraint 1: For each point t in time, $r_i.Start \le t \le r_k.End$, 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 server associated with customer C and t be any time point such that $r_i.DelivSt \le t \le r_k.DelivEnd$. The set of *delivered-but-unconsumed blocks* $DBUB(C, t)$ to customer C at time t is the set $\{ b_j \}$ for some $1 \le i \le k$, $r_i.Start \le b_j \le r_i.End$ and $t \ge r_i.CustShipEnd$ and $t < r_i.CustConsEnd$).

Constraint 2: For each point t in time such that $r_i.DelivSt \le t \le r_k.DelivEnd$,

$$buf(C) \geq bsize * card(DBUB(C, t))$$

3. Let v' be any server and 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 $r_1.DelivSt \leq t \leq r_k.CustShipEnd$, $buf(v') \geq$ (bsize * card($DBUB(v', t)$)

The customer wait associated with a segment/ block oriented presentation plan $r_1,..,r_k$ is defined to be the value of the field $r_1.CustConsStart$. A presentation plan $PP = r_1, ..r_k$ for delivering movie m to customer C via originating server v is said to be wait optimal iff for all other presentation plans $PP' = r'_1,...,r'_k$ for delivering movie m to customer C via originating server v,

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

An alternative criterion for optimality is access bandwidth. Every time the originating server satisfying a customer 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 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 request, 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 commitments that the server has made to other customers.

4.3 Hybrid Presentation Plans

Suppose $PP = r_1,...,r_k$ is a hybrid presentation plan. The structure of the constraints that must be satisfied by PP is somewhat different from that satisfied by segment (resp. block) oriented presentation plans because of the different structure of hybrid presentation records

Definition 4.2 (Feasible Hybrid Presentation Plan) Let C be a customer and let v be the customer's originating server. A hybrid presentation plan $PP = r_1,...,r_k$ 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 ($r_i.DelivSt$ + (w-$r_i.Start$) * bsize / $r_i.BWAssign$, $r_i.DelivEnd[i]$, v, $r_i.Movie$, w, w. 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 $\{ b_j \mid$ for some $1 \leq i \leq k$; $r_i.Start \leq b_j \leq r_i.End$ and $t \geq r_i.CustShipEnd[j]$ and $t < r_i.CustConsEnd[j]\}$.

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

   $$buf(C) \quad bsize * card(DBUB(C, t))$$

4. Let v' be any server and 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 $r_i.CustShipEnd[j] \geq t$ and $r_i.DelivSt + (j - r_i.Start + 1) * bsize / bw(v',v) \leq t \}$.

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

$$\text{buf}(v') \ge \text{bsize} * \text{card}(DBUB(v', t))$$

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

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

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 originating server *OS* does the followings to minimize the access bandwidth

- 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 initialized. *NSB(Next Start Block)* is set to 1. Earliest presentation start time, *sttime* is initialized depending on the request arrival time and a minimal overhead 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 according to the number of consecutive blocks that are available locally.

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 segment 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 $t_{Req(v1,v2,m,b1,b2)}$ denote the time at which the request is issued by server *v1*.
- **Estimated Connection Time** Let $ct(v_1,v_2)$ reflect the estimated connection time between *v1* and *v2*. If a connection already exists between *v1* and *v2*, then $ct(v1, v2) = 0$. Thus, server *v2* receives the request from server *v1* only at time

$$ct(v1,v2) + t_{Req(v1,v2,m,b1,b2)}$$

- **Download Complete:** Finally, suppose each block has size *bsize*. Then the total download time to download all blocks in the interval $[b1,b2]$ is given by

$$(b1 - b2 + 1) * bsize / \text{bw}(v1, v2, t_{Req(v1,v2,m,b1,b2)})$$

Thus, the actual time that server *v1* receives the

required data is

$$ct(v1, v2) + t_{Req(v1,v2,m,b1,b2)} + (b2 - b1 + 1) * bsize / bw(v1, v2, t_{Req(v1,v2,m,b1,b2)})$$

5. After determining the request time for segment of blocks, the server *OS* has to issue a request to the target server with the information on the requested movie segment and the time at which the segment is required.

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 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 presentation start time. Hence, *OS* selects the target server which returns the minimum delay δt. The movie start time, *sttime*, is incremented by δt, *NSB* is reset 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 discussion applies to all the three presentation plans. In the case of a block-oriented presentation plan, the segment size is always one block.

<Table 6> Parameters Used For Simulation

Number of Movies	300
Number of Segments	20-30 per movie (avg. 25)
Segment Size	10-30 blocks (avg. 20)
Block Size	6 seconds' compressed video data
Number of Requests	600
Request Arrival Time	5-25 sec (avg. 12)
Request Pattern	based on the data collected from video shop[1]
Number of Servers	5
Disk Buffer Size	30MB
Disk Bandwidth	Avg. 1.9MB

6. Simulation Experiments

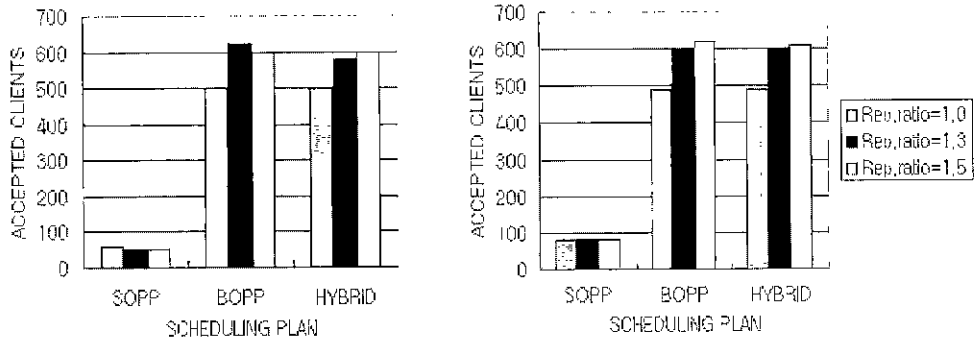
Simulation experiments of the suggested VoD architecture were carried out. A total of 600 customers were assumed to make requests for movies. The user requests arrive uniformly every 12 seconds on the average. The requests are assigned to the servers uniformly. The playback rate of the movies is 30 frames per second. <Table 6> summarizes the parameters used for simulation. For the request pattern of the movie, we use raw data obtained from a video rental store[25], and that has previously been used by several other authors [1, 3]. This pattern turns out to follow the Zipf distribution.

From the movie placement mapping, we calculate the replication ratio which was defined originally in [15]. *Replication ratio* is defined as the ratio of the sum of the number of movie blocks currently stored in the servers to the sum of the number of blocks of the movies, i.e.,

$$Replication\ ratio = \sum_{s \in Server} BlocksInServer(s) \ / \ \sum_{m \in Movie} BlocksInMovie(m)$$

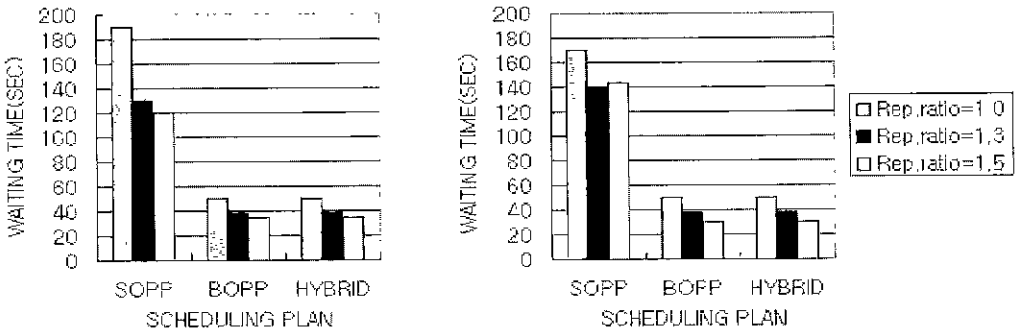
where *BlocksInServer(s)* denotes the total number of movie blocks stored at server s and *BlocksInMovie(m)* denotes the total number of blocks in movie m. The replication ratio is 1 when there is no replication of movie blocks in the set of VoD servers. The simulation experiments were carried out with replication ratios 1.0, 1.3 and 1.5. The window size denoted by the variable τ for holding downloaded blocks in the buffer at the server site 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 is allocated for the whole segment until the segment is completely consumed.
- **Dynamic Strategy**: Here, buffer is allocated for



(a) Using Naive Buffer Allocation (b) Using Dynamic Buffer Allocation

(Figure 6) Number of Accepted Clients



(a) Using Navc Buffer Allocation (b) Using Dynamic Buffer Allocation

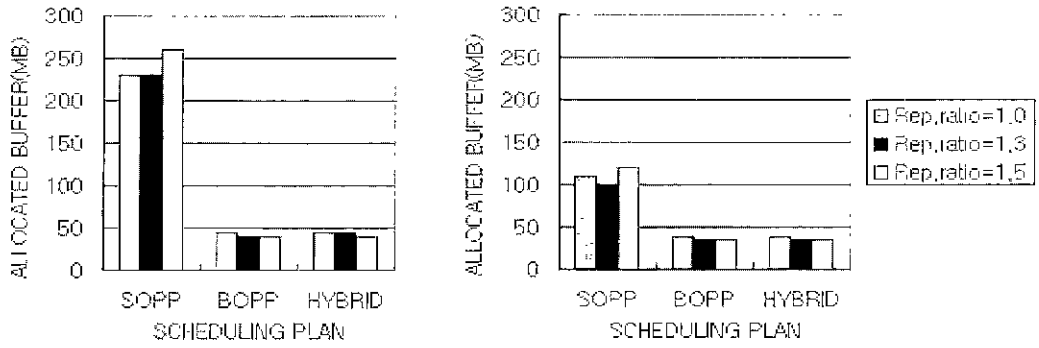
(Figure 7) Initial Waiting Time

each block of the segment until the block is consumed.

(Figure 6) shows how many clients were accepted within the maximum initial waiting time BOPP and HYBRID perform better in terms of the number of accepted clients. Both these plans also perform better (in terms of the increase in number of accepted clients) with higher replication ratio However, increasing the replication ratio beyond a certain point (1.3 in this experiment) did not help much As far as SOPP is concerned, the number of accepted clients is very low because of the large buffer requirements With dynamic buffer allocation strategy, this number doubles but is still very low compared to BOPP and HYBRID.

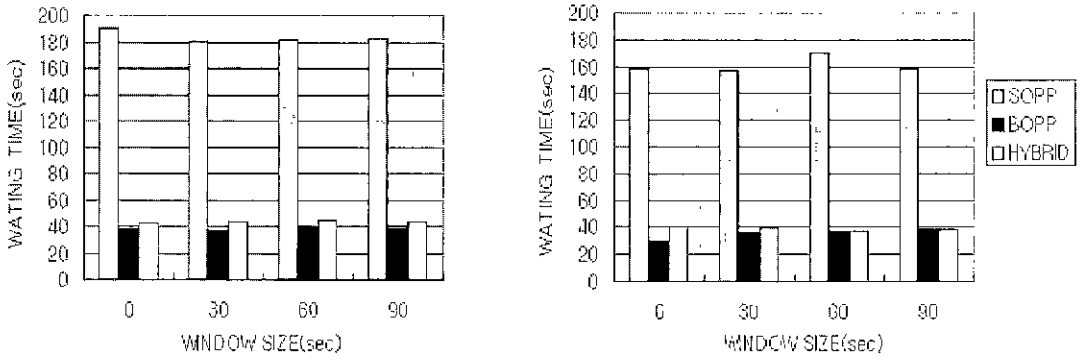
The performance of the presentation plans with respect to the initial customer wait times is show in (Figure 7). Again, BOPP and HYBRID perform better than SOPP with smaller customer wait times. Also, data replication helps in reducing the customer wait times. Dynamic buffer allocation strategy helps SOPP to reduce initial customer wait time. The benefit for BOPP and HYBRID is only marginal with dynamic buffer allocation.

In (Figure 8), the amount of buffer allocated for each presentation plan is shown. From the graphs, we can see that BOPP and HYBRID used much less buffer than SOPP. Data replication did not have much effect on buffer usage. Dynamic buffer allocation helps in reducing buffer usage for SOPP to a great extent.



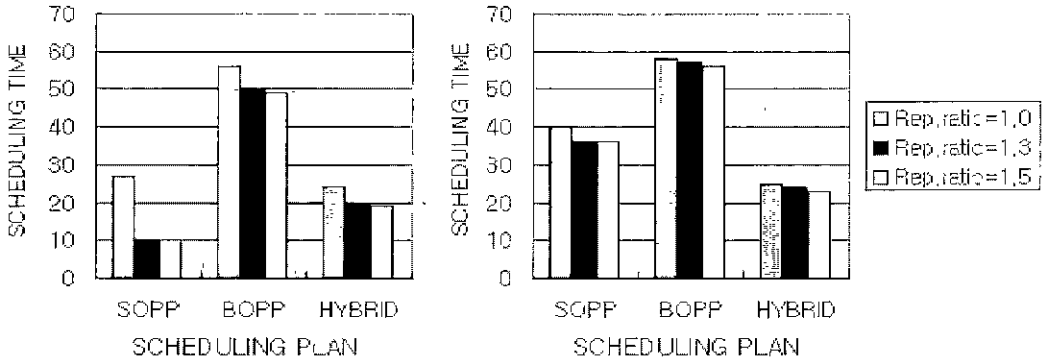
(a) Using Naive Buffer Allocation (b) Using Dynamic Buffer Allocation

(Figure 8) Buffer Size Allocated



(a) Using Naive Buffer Allocation (b) Using Dynamic Buffer Allocation

(Figure 9) Effect of Window Size



(a) Using Naive Buffer Allocation (b) Using Dynamic Buffer Allocation

(Figure 10) Time for Generating Schedule

To see the effect of window size, we measured the initial waiting time under different window sizes. As we can see in (Figure 9), the window size did not have much effect on the initial waiting time. This is because even though some customers can get prompt response from the buffered data, other customers could experience delays due to the lack of available memory space.

For most categories, *BOPP* and *HYBRID* performed almost alike and much better than SOPP. However, BOPP took much longer time and memory space than HYBRID to generate and maintain the presentation plan, as shown in (Figure 10). It should be noted that replication of data reduced the computation time of presentation plans. However, dynamic buffer allocation strategy lead to an increase in the plan computation time. The effect is more pronounced in the case of SOPP where the computation time nearly doubles.

Summarizing the experiment results, the performance of three presentation plans is 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 increases, the number of accepted customers shows 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 naive case.

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 naive one. Also, the time required for computing hybrid presentation plans is much lower than that for block-oriented presentation 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 presentation plans. These plans all enjoy some *structural variations, but all of them are executable*. In this section, we study some of the properties and relationships among them.

7.1 Relationship between Different Plans

In this section, we study the relationships among the different types of presentation plans. 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, 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 for this task which was the same wait time as PP.

Theorem 7.1 Let *BOPP*, *SOPP*, *HYBRID* be optimal block-oriented, segment-oriented, and hybrid presentation plans for delivering movie m to customer C via originating server s. Let $WAIT_B$, $WAIT_S$, $WAIT_H$ be the customer wait times associated with *BOPP*, *SOPP*, *HYBRID* respectively. Then

$$WAIT_H = WAIT_S = WAIT_B$$

The above results indicate that in order to minimize the customer waiting time, 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 Network Resources

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

Definition 7.1 Suppose $NL_1 = (V, E, bw_1, MOVIE, \delta_1)$ and $NL_2 = (V, E, bw_2, MOVIE, \delta_2)$ are two logical networks. We say $bw_1 \leq bw_2$ iff $(\forall t, e_1, e_2)$ $bw_1(e_1, e_2, t) \leq bw_2(e_1, e_2, t)$

Intuitively, $bw_1 \leq bw_2$ iff at all times t, the available bandwidth in any network link according to bw_2 is at least as much as that according to bw_1

Theorem 7.2 (Effect of Increased Bandwidth) Suppose NL_1 and NL_2 are two logical networks as defined above and suppose $bw_1 \leq bw_2$. Let $BOPP_i$, $SOPP_i$, $HYBRID_i$ be optimal block-oriented, segment-oriented, hybrid presentation plans for delivering movie m to customer C via originating server s w.r.t. $NL_i (i = 1,2)$. Let $WAIT_{Bi}$, $WAIT_{Si}$ and $WAIT_{Hi}$ denote the customer wait time w.r.t. $BOPP_i$, $SOPP_i$, $HYBRID_i$ for $i = 1, 2$. Then,

$$WAIT_{B2} \leq WAIT_{B1},\ WAIT_{S2} \leq WAIT_{S1},\ WAIT_{H2} \leq WAIT_{H1}$$

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 onto the customer by diminishing the time the customer has to wait.

Definition 7.2 Suppose NL_1 and NL_2 are two logical networks. We say that $\delta_1 \leq \delta_2$ iff

$$(\forall m \in MOVIE)\ (\forall b)\ \delta_1(m, b) \subseteq \delta_2(m, b)$$

This definition basically says that $\delta_1 \leq \delta_2$ iff whenever site s contains block b of movie m according to placement mapping δ_1, then site s also is considered to have block b of movie m according to placement mapping δ_2. However, δ_2 may place extra blocks of one or more movies at site s

Theorem 7.3 (Effect of Increased Replication) Suppose NL_1 and NL_2 are two logical networks defined above and suppose $\delta_1 \leq \delta_2$. Let $BOPP_i$, $SOPP_i$, $HYBRID_i$, $WAIT_{Bi}$, $WAIT_{Si}$ and $WAIT_{Hi}$ be defined as in Theorem 12.2 Then

$$WAIT_{B2} \leq WAIT_{B1},\ WAIT_{S2} \leq WAIT_{S1},\ WAIT_{H2} \leq WAIT_{H1}$$

The above theorem shows that if we increase the placement mapping, then we guarantee 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 δ_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 the work[7].

Suppose $NL = (V, E, bw, MOVIE, \delta)$ is a logical network layout. We say that $buf_1 \leq buf_2$ iff for all $v \in$

V, $buf_1(v) \leq buf_2(v)$. 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 $NL = (V, E, bw, MOVIE, \Psi)$ is a logical network layout. Suppose that $buf_1 \leq buf_2$. Let $BOPP_i$, $SOPP_i$, $HYBRID_i$, $WAIT_{B,i}$, $WAIT_{S,i}$ and $WAIT_{H,i}$ be defined as in Theorem 12.2. Then:

$$WAIT_{B,2} \leq WAIT_{B,1};\ WAIT_{S,2} \leq WAIT_{S,1};\ WAIT_{H,2} \leq WAIT_{H,1}$$

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 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 [7, 10, 18, 24]. 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 properties that had only had experimental validation previously.[12, 13, 23] discuss the network requirements for multimedia-on-demand applications.[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 that the network 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. 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 wish to access videos of lectures, at their leisure. Furthermore, in the rapidly emerging area of multimedia databases [22] and video databases [8, 9, 19, 20, 21], 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 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-server interactions. When a customer requests a movie from his/her local server, 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 it will retrieve it. Presentation plans cannot be constructed autonomously by the local server. Rather, the

local server must interact with remote servers 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 measured/evaluated using customer wait time and access bandwidth 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.

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
t : initial overhead time ,
OptSchedule : optimal movie presentation schedule ;

```
01:GenOptSchedule ( W S, request )
02:{
03:   stime = request.arr_time + fft ,
04:   Finished = FALSE ;
05:   While ( stime < MP D and Finished != TRUE )
06:   {
07:      delay = 0 ,
08:      NSB = 0 ;
09:      NP D = stime ,
10:      OptSchedule = , ,
11:      Again = FALSE ,
12:      While ( NSB < Totbnum ( request.mov ) and Again != TRUE )
13:      {
14:         CommRec =check_LocalCache ( NSB, NP D, request ) ;
15:         If ( CommRec.flag == SUCCESS )
16:         {
17:            OptSchedule = OptSchedule ∪ CommRec ;
18:            NSB = NSB + CommRec.blknum ,
19:            NPD = NP D + CommRec.playtime ,
20:         }
21:         Else
22:         {
23:            SSR = identify_TargetServer ( request NSB),
24:            /* identify all the servers and segments containing NSB */
25:            sort_ScheduleRequest (SSR) ,
26:            /* sort schedule requests by the number of blocks */
27:            Scheduled = FALSE ,
28:            While (SSR != NULL and Scheduled != TRUE)
29:            {
30:               sr = next schedule request in SSR
31:               CommRec = request_ScheduleToTarget ( sr NPD ) ,
32:               /* request scheduling of data blocks to the target */
33:               /* server within the deadline */
34:               If (CommRec.flag=SUC and check_LocalBuf ( CommRec ) = OK )
35:               Scheduled = TRUE ;
36:            }
37:            If ( Scheduled == TRUE )
38:            {
39:               OptSchedule = OptSchedule [ CommRec ,
40:               NSB = NSB + CommRec.blknum ,
41:               NP D = NP D - CommRec.playtime ;
42:            }
43:            Else
44:            {
45:               cancel_CommRec ( OptSchedule ) ;
46:               delay = comp_delay(SSR) ,
47:               stime = stime + delay ,
48:               Again = TRUE ,
49:            }
50:         }
51:      }
52:      If ( NSB ≥ Totbnumt request.mov) )
53:         Finished = TRUE ,
54:   }
55:   If ( Finished == TRUE )
56:      return (OptSchedule) ,
57:   Else
58:      Reject Request ;
59:}
```

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 ,

```
01  handle_ScheduleRequest ( sr, connId, NPD )
02  /* this routine is called within request_ScheduleToTaget() to handle */
03  /* the schedule request connId is given as a result of connection establishment */
04  {
05     net_resv = retrieve_netConnectionStatus ( connId ) ,
06     /* retrieve information about the network connection establishment */
07     /* requested with required bandwidth and connection duration */
08     bandwidth = net_resv.bandwidth ,
09     btime = block_size / bandwidth ,
10     blocks = comp_netCapacity ( net_resv.interval, btime ) ,
11     sr.endblk = sr.stblk + blocks - 1 ;
12     tmp_schedule = ;
13     For ( b_id = sr.stblk , b_id ≤ sr.endblk , b_id-- )
14     {
15        blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime ) ,
```

```
16          cache_resv = lookup_cacheTaskTable ( srmnov, b_id, blk_deadline ),
17          /* checks the cache table for the movie and segment and record */
18          /* blocks hit in the cache during interval */
19          If ( cache_resv.status == hit )
20          {
21              tmp_schedule = apcond_BlkSchedule ( cache_info ),
22              blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime ),
23              continue ,
24          }
25          disk_resv = lookup_diskScheduleTable ( srmnov, b_id, blk_deadline ),
26          /* look up the disk schedule table to see if the requested block */
27          /* can be scheduled within the block deadline */
28          If ( disk_resv.status == rejected )
29              If ( tmp_schedules == , )
30              {
31                  CommRec.flag = FAIL ,
32                  return ( CommRec ),
33              }
34              Else
35              {
36                  CommRec = creat_CommRec ( sr, b_id ),
37                  CommRec.flag = SUCCESS ,
38                  CommRec.schedule = generate_Schedules ( tmp_schedules ),
39                  return ( CommRec ) ;
40              }
41          tmp_schedule = append_BlkSchedule ( disk_info ) ;
42          blk_deadline = comp_nextBlkDeadline ( blk_deadline, NPD, btime );
43      }
44      CommRec = creat_CommRec ( sr, b_id )
45      CommRec.flag = SUCCESS ,
46      CommRec.schedule = generate_Schedules ( tmp_schedules ),
47      return ( CommRec ) ;
48  }
```

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, Jan. 1996

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

[3] A. Drapeau, D. Patterson, R. Katz, "Toward Work load 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 off 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, CA, Aug. 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, Jan. 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, Jan. 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," 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, Vol.6, No.4, pps 251-270, 1998.

[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, pp.8-18, September 1993.

[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 of Intl. Conference on Very Large Databases, Santiago, 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.

황 인 준

e-mail : chwang@madang.ajou.ac.kr
1988년 서울대학교 컴퓨터공학 학사
1990년 서울대학교 컴퓨터공학 석사
1998년 University of Maryland at College Park 전산학 박사
1998년~1999년 Bowie State University 조교수
1999년~1999년 HRL Research Lab Visiting Professor
1999년~현재 아주대 정보및컴퓨터공학부 조교수
관심분야 : 분산 멀티미디어 시스템, 멀티미디어 데이터베이스

변 광 준

e-mail : byeon@madang.ajou.ac.kr
1985년 서울대 컴퓨터공학 학사
1987년 Pennsylvania State University 전산학 석사
1993년 University of Southern California 전산학 박사
1994년~현재 아주대 정보및컴퓨터공학부 조교수, 부교수
관심분야 : 멀티미디어 데이터베이스, 분산 데이터베이스 시스템, 분산 객체 컴퓨팅