Distributed Service Paradigm for Remote Video Retrieval Request

Youjip Won and Jaideep Srivastava*

Department of Computer Science, University of Minnesota
Minneapolis, MN 55455

{yjwon|srivasta}@cs.umn.edu

Abstract

The per service cost have been serious impediment to wide spread usage of on-line digital continuous media service, especially in the entertainment arena. Although handling the continuous media may be achievable due to the technology advances in past few years, its competitiveness in the market with the existing service type such as video rental is still in question. In this paper, we propose a service paradigm for continuous media delivery in a distributed infrastructure in an effort to reduce the resource requirement to support a set of service requests. The storage resource and network resource to support a set of requests should be properly quantified to a uniform metric to measure the efficiency of the service schedule. We developed a cost model which maps the given service schedule to a quantity. The proposed cost model is used to capture the amortized resource requirement of the schedule and thus to measure the efficiency of the schedule. We develop a scheduling algorithm which strategically replicates the requested continuous media files at the various intermediate storages.

Keywords: *video caching, storage overflow, video scheduling, continous media delivery, distributed service, cost model*

1 Introduction

1.1 Motivation

Entertainment, education, teleconferencing, telemarketing, telemedicine, etc. are a number of promising applications of the *Video-On-Demand* technology. In some applications such as business and medicine, the cost of the technology maybe justifiable, but in other applications such as home entertainment, this technology will have to compete with $2 - $3 video rental cost. Given the fact that there is sufficient consumer dissatisfaction with fast rising cable rates, any expensive technology is not likely to be very successful in the mass entertainment arena. On the other hand, if we examine existing video rental patterns, it is not unreasonable to assume that customers would be satisfied with a *Video-On-Reservation* service, where the customer makes the service request some time in advance to the actual presentation time, i.e. an hour, a day, etc . We believe it is possible to provide *Vide-On-Reservation* service in a cost-effective manner. Since the a set of user requests is available to entertainment provider in *VOR* service, the server can perform global optimizations based on the user request information as well as the the availibility of various resources.

Given the limits on I/O subsystems of the server, there are many difficulties in supporting multiple service requests directly from a single server, while simultaneously achieving low cost. Furthermore, as the length of the communication route between server and client increases, the cost of maintaining smooth media flow across the network increases dramatically. In this paper we present the service paradigm of the remote retrieval request in a distributed environment. The distributed environment consists of a number of *intermediate storages* and *video warehouses*, all connected by a high speed network.

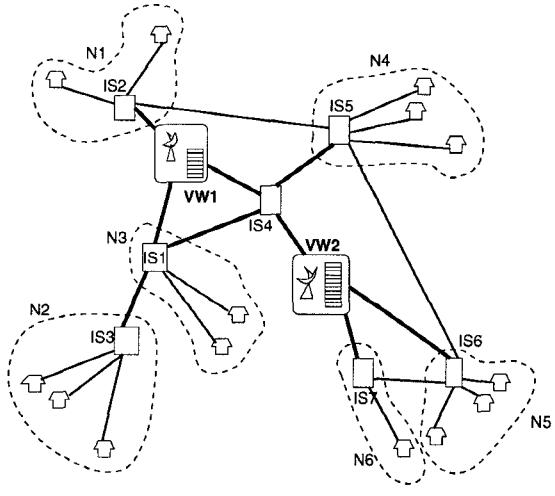


Figure 1. Topological Layout

*The work of this author is supported in part by Air Force contract number F30602-96-C-0B to Honeywell Inc., via subcontract number B0903054/AF to the University of Minnesota.

Fig. 1 depicts the environment, which consists of a video warehouse, intermediate storages IS_i's, and the users in each neighborhood N_i's. A video warehouse is an archive that may have several thousand video files in its storage system.

1.2 Related Works

In this section, we briefly introduce the works done in each component of our distributed service environment. Papadimitriou et al[9] proposed the service model for on-line home entertainment service and also devised an algorithm to arrange the service schedule. Hierarchical storage structure in video warehouse is the promising solution to achieve the cost-effectiveness of the data storage[8, 6, 14, 15]. Distributed hierarchical storage architecture[13, 2, 3] is also one of the emerging technology to maintain excessively large amount of data. In their architectural models, there is a centralized video server which consists of a two-stage hierarchy and a number of relatively small storage systems which are geographically placed in various locations within metropolitan area. The existing network infrastructure is not originally designed to support time critical application. There have been a number of proposals to guarantee the continuity over the high speed communication network[12, 17]. The other community of academia and industry focuses on utilizing the existing communication medium such as twisted pair telephone line[11, 1] or coaxial cable line[7, 1] which is already available to most of the residential unit. Development of optimal pricing model, *how much user has to pay for the service?*, suddenly draws wide attention from various communities. Shenker et al[10] and Cocchi et al[4] investigate the pricing model of the multiple service class network.

1.3 Contributions

A great amount of efforts from various community is put in supporting the continuity requirement of the on-line multimedia information delivery. However, economic aspect of servicing a set of users considering storage and network resources is paid little attention. In this paper, we propose the usage of intermediate storage to relieve the burden of the central server - *information provider* and to reduce the network traffic. We develop a mechanism, *cost model*, which maps the overall resource requirement for supporting the set of requests to the domain of comparable values, *cost*. The proposed cost model enables the information provider to examine the cost-effectiveness of a certain service delivery schedule. The cost of using intermediate storage and the cost of using network determines how to deliver the service to the end-user. We develop an algorithm to replicate the files to intermediate storages to obtain the efficient service schedule for the request of remote information retrieval.

The rest of the paper is organized as follows: In section 2, the mathematical formulation of the problem and the cost model for servicing delivery is provided. In section 3, we present our approach to solving to the video delivery problem, and in section 4, we discuss how to handle storage overflow. Performance evaluation is presented in section 5 and conclusions in section 6.

$srate(IS_i)$	charging rate for IS_i (in \$/$(byte \cdot sec)$
$nrate(i,j)$	charging rate for network (i,j) in \$/*byte*
$\mathcal{S}_i$	service schedule of file i
$\mathcal{S}_i^{new}(\Delta t, IS_j)$	service schedule with the constraints $(\Delta t, IS_j)$
$\mathcal{S}$	service schedule for all files
$\Psi(\mathcal{S})$	cost of schedule $\mathcal{S}$
$OF_{\Delta t, IS_j}$	Overflow Situation $\Delta t, IS_j$.
$\mathcal{H}_{\Delta t, IS_j, c_i}$	heat of rescheduling c_i w.r.t. $OF_{\Delta t, IS_j}$
ρ_i	playback length for file i
$size_i$	size of file i (byte)
d_i	file transfer information
c_i	file residency information

Table 1. Notations

2 Problem Formulation

2.1 Service Schedule

A user request arriving at the server consists of three attributes, *user_id, video_id* and *starting_time*. The *service schedule* $\mathcal{S}$ is a set of information about how to arrange the delivery of the continuous media streams to end users. There are two types of information in a service schedule - *network transfer information* $\mathcal{D} = \{d_1, d_2, \ldots, d_{n_d}\}$ and *usage of the intermediate storage* $\mathcal{C} = \{c_1, \ldots, c_{n_c}\}$. Servicing a video file to end user requires network transfer of the stream and possible caching of the file at the intermediate storage. The network transfer information d_i is $(route_i, t_i^d, id_i)$. $Route_i$ is a sequence of network nodes, i.e. $n_i^{src}, \ldots, n_i^{dst}$, where n_i^{src} and n_i^{dst} correspond to intermediate storages. Delivery information d_i informs the server and intermediate storages that flow of file i from n_i^{src} to n_i^{dst} is to begin at t_i^d. Routing information between n_i^{dst} and end user is not specified since we assume the path between the user and its local intermediate storage is uniquely defined. The intermediate storage is said to be *local* to a user if it is located in the same neighborhood with the respective user.

In servicing a set of requests, a video file has to be stored temporarily in various intermediate storages by copying data blocks from the on-going continuous media stream. The information c_i about temporary storage of a video file is the vector of five elements: $([t_i^s, t_i^f], loc_i, id_i, n_{src}, service\ \ list)$. $[t_i^s, t_i^f]$ denotes the interval of caching. t_i^s corresponds to the time when the file starts being loaded at the intermediate storage. t_i^f denotes the start time of the last service. The data blocks which are consumed by the last request in chronological order are no longer required. Caching interval $[t_i^s, t_i^f]$ is followed by the playback duration of the last service. loc_i and id_i represent the intermediate storage and the respective file. n_i^{src} is the

source of the on-going stream from which the data block is copied. n_i^{src} can be either another intermediate storage or *VW*.

2.2 Cost Modeling

To measure the overall cost-effectiveness of the service schedule $\mathcal{S}$, there needs to be a generic mechanism which maps the given schedule $\mathcal{S}$ into a *cost*. Schedule $\mathcal{S}$ is a collection of network transfer information d_i's and file residency information c_i's. The mapping mechanism should capture $\mathcal{S}$'s resource consumption in storage devices and the communication network. We define Ψ as a mapping function that transforms service schedule $\mathcal{S}$ into a *quantity*. The *cost of* $\mathcal{S}$ thus corresponds to $\Psi(\mathcal{S})$. We use the monetary metric to represent the cost for a schedule $\mathcal{S}$. The *cost* of a schedule $\mathcal{S}$ is sum of the cost of using storage devices and cost of using communication medium. The network transfer information d_i's and file residency information c_i's has the different formation and thus different function is used to obtain its respective cost. *Cost* is amount of the resource consumed multiplied by *per unit cost*[1] of each resource. In a heterogeneous environment, it is not possible to impose a uniform *per unit cost* to all resources which have different performance and characteristics. *Per unit cost* is inherent to an individual resource entity, e.g. each intermediate storage or each network hop. We develop the function $\Psi_{\mathcal{C}}(c_i)$ and $\Psi_{\mathcal{D}}(d_i)$ for file residency information c_i, and network transfer information d_i, respectively which maps given information into the domain of *cost*. The cost of schedule $\mathcal{S}$ is hence represented as in Eq. (1).

$$\Psi(\mathcal{S}) = \sum_{i=1}^{n_d} \Psi_{\mathcal{D}}(d_i) + \sum_{i=1}^{n_c} \Psi_{\mathcal{C}}(c_i) \quad (1)$$

2.2.1 Cost Model for Storage

The amortized resource requirement at an intermediate storage is a function of *file size*, *duration of residency*, and *playback length*. $\Psi_{\mathcal{C}}(c_i)$ is the amortized resource requirement multiplied by the charging rate at the respective intermediate storage. To capture the amortized storage space requirement of the service properly, $\Psi_{\mathcal{C}}(c_i)$ has to reflect all the three attributes. In the charging mechanism for storage, we propose a cost model that considers both the duration and the size of the service. Unit of the storage cost model is \$/(byte· sec).

Files are loaded on an intermediate storage by copying data blocks from streams during its transmission, and thus the disk space is filled incrementally until the entire file is cached. We assume that the storage space of $size_{id_i}$ needs to be reserved from the start of the caching. We categorize the file residency information c_i into two types: namely, *short residency* and *long residency*, depending on the length of interval $[t_i^s, t_i^f]$. Let ρ_{id_i} be a playback length of a file id_i. c_i is categorized as *short residency* type if $(t_i^f - t_i^s) < \rho_{id_i}$, and as *long residency* type otherwise.

[1] termed as *charging rate* of the resource in this paper.

For long residency, amortized storage cost for c_i is formulated as in Eq. (2).

$$\Psi_{\mathcal{C}}(c_i) = srate(IS_{loc_i}) \cdot size_{id_i} \left((t_i^f - t_i^s) + \frac{\rho_{id_i}}{2} \right) \quad (2)$$

For short residency, the amortized storage cost for c_i for *short residency* can be formulated as in Eq. (3).

$$\Psi_{\mathcal{C}}(c_i) = srate(IS_{loc_i}) size_{id_i} \left((t_i^f - t_i^s) + \frac{(t_i^f - t_i^s)^2}{2\rho_{id_i}} \right) \quad (3)$$

2.2.2 Cost Model for Network

The objective of the cost model for the communication network is to properly capture the amortized bandwidth cost to service a set of requests. A certain amount of network bandwidth needs to be guaranteed to deliver the service to end users with the given *QoS*. We assume that the bandwidth requirement for a request is determined by the requested file, which is provided by the service provider. Let B_{id_i} be the bandwidth requirement for file id_i's playback. The amortized bandwidth requirement for d_i corresponds to $\rho_{id_i} B_{id_i}$ *bytes*. As in the storage cost model, we use a monetary metric as the basic unit of cost, i.e. \$/byte. Let $route_i$ be $(n_i^1, \ldots, n_i^{n_{nw}})$ of d_i, where n_i^1 and $n_i^{d_i}$ correspond to n_i^{src} and n_i^{dst}. $nrate(n_i^j, n_i^k)$ is the charging rate of the route between n_i^j and n_i^k. In general, nrate(n_i^j, n_i^k) will depend on various factors, e.g. network topology, link capacities, routing, etc. However, we assume the underlying communication infrastructure maps all of these factors into a cost (i.e. \$ amount) per proposed unit, i.e. *byte*. Depending on the underlying network structure, charging rate can be defined on *per hop basis* or *end-to-end basis*. The function $\Psi_{\mathcal{D}}(d_i)$ is formulated as in Eq. (4) for *per hop basis* or *end-to-end basis*, respectively.

$$\Psi_{\mathcal{D}}(d_i) = \begin{cases} nrate(n_i^{src}, n_i^{dst}) \cdot \rho_{id} \cdot B_{id} \\ \sum_{j=1}^{n_{nw}-1} nrate(n_i^j, n_i^{j+1}) \cdot \rho_{id} \cdot B_{id} \end{cases} \quad (4)$$

2.3 Problem Description

We can view the overall service schedule $\mathcal{S}$ as the union of the service schedule for each file i, $\mathcal{S}_i$. The cost for servicing all requests is $\sum_{i=1}^{m} \Psi(\mathcal{S}_i)$, i.e. $\Psi(\mathcal{S}) = \sum_{i=1}^{m} \Psi(\mathcal{S})\mathcal{S}_i$. The goal of the video scheduler is to generate a set of service schedule with minimum cost. The service scheduling problem can now be stated as follows:

Definition 1 Video Scheduling Problem($\mathcal{VSP}$)*:Given a set of video requests $\mathcal{R}$, a set of network edges connecting a video warehouse VW, and a set of intermediate storages, the charging rates for intermediate storage* **srate** *and network* **nrate**, *and the storage capacity of each intermediate storage, find the set of file service schedules, $\mathcal{S}_{opt}$ such that*

$$\forall \mathcal{S}, \Psi(\mathcal{S}_{opt}) \leq \Psi(\mathcal{S})$$

Unfortunately, the problem $\mathcal{VSP}$ is NP-*complete*, and thus an efficient algorithm for the optimal solution is unlikely. The details of the proof are provided in [16]. Due to the intractable nature of $\mathcal{VSP}$, we focus our interest on developing an efficient heuristic algorithm.

3 Scheduling of Service Delivery

3.1 Design of Video Scheduler

Video Scheduler is a program in charge of arranging the service delivery The objective of the scheduler is to determine the *service schedule* $\mathcal{S}$ *with minimum* $\Psi(\mathcal{S})$, given the set of service requests. We approach the problem with a two phase algorithm.

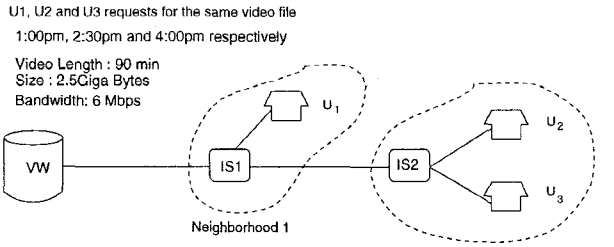
1. **Individual Video Scheduling**: *Find the schedules* $\mathcal{S}_i$ *for each file* i *individually with minimum* $\Psi(\mathcal{S}_i)$, *assuming that the capacity of intermediate storage is large enough to store any one video file.*

2. **Integrating Individual Video Schedules**: *Integrate the schedules* $\mathcal{S}_i$, *taking into account the storage capacity constraints, and resolving any resulting storage overflows.*

In our scheduling policy, *individual video scheduling* and *storage overflow detection* enables the scheduler to globally analyze the service cost for each file i, i.e. $\mathcal{S}_i, i = 1, \ldots, m$, and to determine the scheduling priorities among themselves. The file called *victim* which is determined to have lowest priority in scheduling is thus re-scheduled to resolve the *storage overflow*.

3.2 Individual Video Scheduling

In finding the service schedule $\mathcal{S}$, the scheduler collects the requests for the cycle and partitions them into sets $\mathcal{R}_i, i = 1, \ldots, m$ with each of the m distinct video files requested. $\mathcal{R}_i$ corresponds to the set of requests for video i. The schedule $\mathcal{S}_i$ for each $\mathcal{R}_i$ is computed individually, i.e. in computing $\mathcal{S}_i$ the scheduler does not consider the resources required to service the other sets $\mathcal{R}_j, j \neq i$. In the individual video scheduling phase, the scheduler focuses on minimizing the cost of each $\mathcal{S}_i, i = 1, \ldots, m$. Fig. 2 illustrates a sample topological layout, with respective $nrate$, $srate$, and user requests. We enumerate some of the possible video schedules for Fig. 2 and the cost of each schedule. Let id, ρ_{id} and B_{id} be the file id, playback length, and bandwidth requirement of the file, respectively. The cost is computed using the mapping function Ψ described in section 2.3. Since all video files reside at *VW* permanently, the cost of storing video files at the VW is assumed to be 0, i.e. $sratet(VW) = 0$. IS_1 is *local* to U_1 and IS_2 is *local* to $\{U_2, U_3\}$. We assume that ρ_{id} is 90 *min*, and $size_{id}$ is 2.5G Byte. We assume 6 Mbps bandwidth needs to be reserved for the playback of the given file.



	nrate(VW,IS1)	nrate(IS1,IS2)	
Network Cost	¢ 0.2	¢ 0.1	cents/(Mbps*Sec)

	VW	IS1	IS2	
Storage Cost	$0	$1	$2	$/(GBxHour)

Figure 2. Service Requests with Network Cost and Storage Cost

Schedule $\mathcal{S}1$:*All the requests for videos are delivered directly from the video warehouse.* $\Psi(\mathcal{S}1)$ corresponds to $259.2.

Schedule $\mathcal{S}2$:*While* U_1 *gets service directly from the* VW, IS_1 *caches the file.* U_2, *and* U_3 *are serviced from the video file cached on* IS_1. $\Psi(\mathcal{S}2)$ is $138.975.

Based on the proposed cost model, $\mathcal{S}2$ turns out to be the less expensive way of servicing the three users in Fig. 2.

Table 2 is a skeleton algorithm for the first phase. Function *find_video_schedule()* in table 2 computes the service schedule for each file. Papadimitriou et al[9] proposed a greedy heuristic of rectilinear algorithm, which can be used as a *find_video_schedule()*. Let us assume that there are

Algorithm 1 IVSP_solve

```
1   IVSP_solve(M, R, nrate, srate){
2       for each files i ∈ M
3           S_i^good ← find_video_schedule(R_i, nrate, srate)
            ;
4       Ψ(S^good) ← Σ_{i∈M} Ψ(S_i^good);
5       return(S^good);
    }
```

Table 2. pseudo code for Individual Video Scheduling Problem

l users $u_1, \ldots, u_l$ requesting video i, M available video files, and N intermediate storages. The users are numbered chronologically with respect to service start time, i.e. $t_i \leq t_{i+1}$. The steps below describe the function $find\_video\_schedule()$ in Table 2.

1. The algorithm iterates l times, i.e. once for each user $U_i, i = 1, \cdots, l$.

2. For u_i, given the schedule and respective cost found for $u_1, \cdots, u_{i-1}$, the scheduler computes the incremental network cost and storage cost for servicing u_i via each $IS_1, \cdots IS_N$.

3. To service u_i in addition to the users $u_1, \cdots, u_{i-1}$, the existing schedule has to be updated in one of the following ways. (1) The resident period of the file at a certain intermediate storage has to be extended, or (2) another intermediate storage which has not been used in servicing $u_1, \cdots u_{i-1}$ is introduced to cache the file. Either situation creates extra storage cost and network cost. or (3) The request is serviced from *VW*.

4. All the intermediate storage $IS_1, \cdots, IS_N$ are considered in a new schedule for servicing $u_1, \cdots, u_i$. The scheduler computes the incremental cost for each. An updated schedule with the minimum incremental cost would be chosen as the schedule for servicing $u_1, \cdots u_i$.

5. Considering the graph structure of the network, there can be more than one path between any pair of nodes, each of which can be VW or intermediate storage. If a new intermediate storage is introduced to cache the file, the scheduler has to compute the network transmission cost of transferring a file to a new cache.

3.3 Integration of Individual Video Schedules into a Global Schedule

It is possible that one or more intermediate storages are over-committed during certain time intervals when the individual schedules are integrated. We call this situation *Storage Overflow*. In the *Storage Overflow* situation, the scheduler has to re-schedule some of the files involved in the overflow so that the file does not reside at the intermediate storage during the overflow period. Specifically, the following decisions should be made for each *storage overflow* situation: (1) *Which video(s) are selected as victims?* and (2) *For each victim to be rescheduled, how to compute the new schedule for victim?*.

Victim is a file that is selected to be re-scheduled. In *most* cases rescheduling entails the additional cost. However, we cannot exclude the possibility that less expensive schedule is found as a result of rescheduling. This is because the schedule for victim $\mathcal{S}_{victim}$ is computed via a *greedy* heuristic in the first phase, there can exist a less expensive schedule for $\mathcal{S}_{victim}$. An important goal of handling *storage overflow* is to resolve the storage overflow with least possible cost increase in the scheduling which is incurred as a result of re-scheduling.

4 Handling Storage Overflow

4.1 Detection of Storage Overflow

Storage overflow $OF_{\Delta t, IS_j}$ is identified by its location - IS_j and the time interval Δt during which the overflow occurs. The scheduler maintains information about the available space at the intermediate storages. Analyzing this information, namely the *storage requirement* and the *storage availability*, the video scheduler detects all *storage overflow situations*. We define *Overflow_Set*$(IS_j, \Delta t)$ as *the set of file residency information* c_i*'s* which are involved in $OF_{\Delta t, IS_j}$. Fig. 3 illustrates the storage space requirement of the integrated video schedules at the intermediate storage. For the sake of simplicity, Fig. 3 does not show the gradual decrease in the storage requirement at the end of each stay. In Fig. 3, there are two distinct storage overflow situations, namely in intervals Δt_1 and Δt_2.

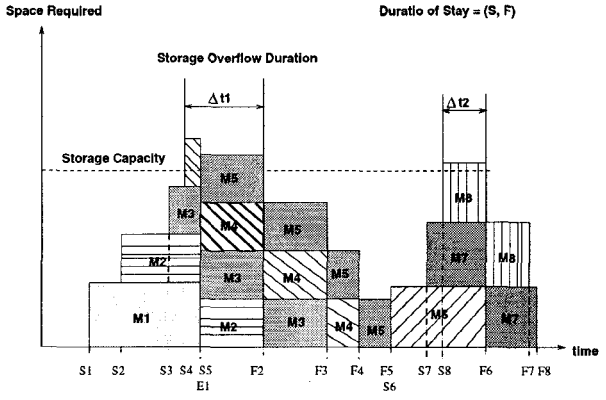


Figure 3. Integrated schedule and respective storage requirement

4.2 Victim Selection

$OF_{\Delta t, IS_j}$ is resolved by re-computing the service schedule for some of the files in $Overflow\_Set(\ \Delta t, IS_j)$. In re-computing the service schedule of id_i, server takes into account additional constraints in obtaining the new schedule $\mathcal{S}_{id_i}^{new}$. The additional constraints are that id_i *is not allowed to be cached at* IS_j *during* Δt and also the *server considers the currently available space at the other intermediate storages*. In Individual Video Scheduling phase, the server does not consider the space limitation of the intermediate storage. As a result of imposing additional constraints, i.e. *space availability*, possible caching site of file i is restricted to the intermediate storage with sufficient space. Let $\mathcal{S}_i^{new}(\Delta t, IS_j)$ be the schedule obtained by rescheduling file i with respect to the constraints $(\Delta t, IS_j)$. In selecting a victim, the scheduler needs to consider the overhead cost of rescheduling and respective improvement over overflow situation. In resolving $OF_{\Delta t, IS_j}$, there can be more than one choice for victim. Thus, selecting a victim with minimum cost increase is an important factor to obtain an efficient schedule. The cost increase which is entailed as a result of rescheduling file id_i with respect to Δt and IS_j is called *overhead cost*, and computed as $\Psi(\mathcal{S}_{id_i}^{new}(\Delta t, IS_j)) - \Psi(\mathcal{S}_{id_i})$.

Overhead cost alone is not sufficient information to select a victim. The purpose of rescheduling is to improve the overflow situation, i.e. reduction in the overflow interval or reduction in the worst case space requirement during the overflow interval. *Heat* is used as the selection criterion for *victim*. A file with a larger *heat* means that its rescheduling accomplishes *better* improvement with the same cost in-

crease and thus is preferred for rescheduling in resolving an overflow situations. To determine *better* improvement, metrics for improvement has to be defined beforehand. There are a number of metrics for *heat*, i.e. *time*, *space*, or *time-space product*. Each of the metrics have different interpretation of improvement and hence different file can be selected as victim under different metrics. We describe four metrics for *heat* in greater detail in section 4.3. and compare the performance through extensive simulation.

4.3 Storage Overflow Resolution Algorithm

Algorithm in Table 3 depicts the skeleton algorithm for storage overflow resolution problem. $\mathcal{SORP}$_solve() in Table 3 takes service schedule $\mathcal{S}$, information about intermediate storages, which includes the overflow sets, storage usage information, and storage capacity as input. $\mathcal{S}$ is a service schedule obtained from the individual video scheduling phase. Let $\mathcal{S}_{id_i}^{new}(\Delta t, IS_j)$ be the schedule for file id_i obtained with the constraint that file id is not allowed to stay at IS_j during Δt. In most cases, changes to $\mathcal{S}_{id_i}$ by rescheduling results in an increase in *cost* for file id_i i.e. $\Psi(\mathcal{S}_{id_i}(\Delta t, IS_j)) > \Psi(\mathcal{S}_{id_i})$. However, since the schedule $\mathcal{S}_{id_i}$ has been obtained via heuristic, less expensive schedule can be found as a result of resolving overflow under a certain circumstance. The ultimate objective in the Storage Overflow Resolution phase is to minimize the cost increase which is caused by overflow resolution.

```
1:  IS: set of intermediate storages ;
2:  Overflow_Set : set of storage overflows ;
3:  Overflow_Set(IS_j) : set of storage overflows at IS_j ;
4:  S_min, S_tmp : Service Schedule ;
5:  SORP_solve(S, IS) {
6:    Resolved = False ;
7:    While ( Resolved != TRUE){
8:      For all IS_j ∈ IS {
9:        For all Overflow_Set(Δt, IS_j)
            ∈ Overflow_Set(IS_j) {
10:         For all c_i ∈ Overflow_Set(Δt, IS_j) {
11:           S_tmp = ReflectiveGreedy(Δt, IS_j, c_i) ;
12:           heat = ComputeHeat(S_id_i, S_tmp) ;
13:           If (heat ≤ minheat){
14:             heat = minheat ;
15:             S_min = S_tmp ;
16:             victim = id_i ;
17:           }
18:         }
19:       }
20:     }
21:     S_victim = S_min ;
22:       Resolved = True ;
23:     For all IS_j ∈ IS {
24:       UpdateStorageUsage(S_min) ;
25:       Resolved = Resolved and
                     DetectStorageOverflow(IS_j)
26:     }
27:   }
28: }
```

Table 3. pseudo code for $\mathcal{SORP}$_solve(*SvcSchedule, C, Overflow_Set*)

For each iteration of the outermost while loop (line 7 - 28) of *SORP_solve()* in Table 3, *SORP_solve* examines all residency information c_i which it is involved in one of the overflows. *SORP_solve()* computes the new service schedule for the respective video file $\mathcal{S}_{id_i}^{new}(\Delta t, IS_j)$ for file id_i in c_i and computes the *effective* improvement of the rescheduling. The file with largest *effective* improvement is chosen as victim in each iteration of 7 - 28 while loop. The actual improvement accomplished over the $OF_{\Delta t, IS_j}$ as a result of rescheduling file id_i may not be directly relevant to *overhead cost*.

Let Δt of $OF_{\Delta t, IS_j}$ be $[t_s, t_f]$ and interval of c_i be $[t_i^s, t_i^f]$. Rescheduling of id_i with respect to $OF_{\Delta t, IS_j}$ will reduce the storage requirement over the interval of $[\max(t_s, t_i^s), \min(t_f, t_i^f + \rho_{id_i})]$. ρ_{id_i} is the playback length of file id_i. As explained in section 2.2, every caching interval is followed by the playack duration which also requires the storage space. Let ΔS be the amortized time-space product which can be improved by rescheduling a file id_i with respect to $OF_{\Delta t, IS_j}$. Under the continuous time domain, ΔS can be formulated as in Eq. 5.

$$\Delta S_{\Delta t, IS_j, c_i} = \int_{\max(t_s, t_i^s)}^{\min(t_f, t_i^f + \rho_{id_i})} f_{c_i}(t)dt \quad (5)$$

$$f_{c_i}(t) = \begin{cases} \gamma \cdot size_{id_i} & \text{if } t < t_i^f \\ \gamma \cdot size_{id_i}(1 - \frac{t - t_i^f}{\rho_{id_i}}) & \text{Otherwise} \end{cases} \quad (6)$$

$$\gamma = \begin{cases} 1 & if\ t_i^f - t_i^s \geq \rho_{id_i} \\ \frac{t_i^f - t_i^s}{\rho_{id_i}} & \text{Otherwise} \end{cases} \quad (7)$$

$f_{c_i}(t)$ in Eq. (6) computes the storage space requirement at each time t based on the storage cost model in section 2.2. The maximum amount of disk space required for c_i depends on whether c_i is either *long residency* or *short residency*. γ in Eq. (7) is a coefficient adjusting the maximum space requirement according to Eq. (2) and Eq. (3). We compare four different metrics for *heat* of rescheduling id_i with respect to $OF_{\Delta t, IS_j}$.

$$\mathcal{H}_{\Delta t, IS_j, c_i} = min(t_f, t_i^f + \rho_{id_i}) - max(t_s, t_i^s) \quad (8)$$

$$\mathcal{H}_{\Delta t, IS_j, c_i} = \frac{min(t_f, t_i^f + \rho_{id_i}) - max(t_s, t_i^s)}{\Psi(\mathcal{S}_{id_i}^{new}(\Delta t, IS_j)) - \Psi(\mathcal{S}_{id_i})} \quad (9)$$

$$\mathcal{H}_{\Delta t, IS_j, c_i} = \Delta S_{\Delta t, IS_j, c_i} \quad (10)$$

$$\mathcal{H}_{\Delta t, IS_j, c_i} = \frac{\Delta S_{\Delta t, IS_j, c_i}}{\Psi(\mathcal{S}_{id_i}^{new}(\Delta t, IS_j)) - \Psi(\mathcal{S}_{id_i})} \quad (11)$$

In Eq. (8) the effective improvement is measured in terms of the length of the improved period. In Eq. (9), the length of

improved period per overhead cost is considered as *heat*. In Eq. (10) and Eq. (11), the improvement on amortized time-space product, and the improvement on amortized time-space product *per overhead cost* is considered as *heat* of rescheduling. In the iteration of line 7 - 18 of *SORP_solve()* algorithm, the file with the largest *heat* is selected as victim. According to our experiments, Eq. (11) performs best, i.e. generates the least expensive schedule, on the average. Details of the experimental results are provided in Sec. 5.

4.4 *Reflective Greedy* Heuristic

The rescheduling algorithm *reflective_greedy()* in line 18 of Table 3 receives a tuple $(c_i, \Delta t, IS_j)$ as an input. It reschedules the file id_i of c_i with the constraint that there is no space available during Δt at IS_j. Rescheduling a file means re-arrange the service delivery of all requests for the file. Different from the *greedy heuristic* in individual video scheduling, the *Reflective Greedy* maintains the space usage information for the intermediate storages, and does not schedule a video file to the intermediate storage if there is not sufficient storage capacity available. This is to avoid generating a subsequent overflow situation as a result of rescheduling. The *reflective greedy* algorithm with input $(c_i, \Delta t, IS_j)$ generates a service schedule for file id_i with the constraint that it is not allowed to be cached at IS_j during Δt.

5 Experiment and Results

5.1 General Experiment Information

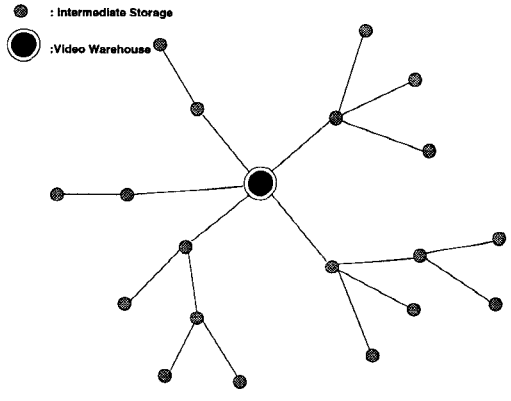


Figure 4. Graph Representation of Network and Storage Topology

Fig. 4 illustrates the topology and connection layout of the video warehouse and intermediate storages from our experiments. There are 20 nodes in total - one video warehouse and 19 intermediate storages. Each intermediate storage supports the users in its respective neighborhood. For notational simplicity, the users are omitted in the graph representation in Fig. 4. In our experiment, the number of users in each neighborhood is 10. Various factors can determine the characteristics of the video service environment. Among them, we take into account four attributes which affect the service scheduling process and its cost - the *Storage Charging Rate*, the *Intermediate Storage Size*, *Network Charging Rate*, and the *user access pattern*. Table 4 shows

Attributes	Values
Number of files	500
Average video file size	3.3 GB
Storage Charging Rate	3, 4, 5, 6, 7, 8 (*/Gbyte · sec*)
Intermediate Storage Size	5, 8, 11, 14 (Giga Bytes)
Network Charging Rate	300, 400, 500, 600, 700, 800, 900, 1000 (*/Gbyte*)
Access Pattern in Zipf	$\alpha = 0.1, 0.271, 0.5, 0.7$

Table 4. System Parameters

the actual values of each attribute used in our experiment. The values of the *storage charging rate* and the *network charging rate* represent values in the arbitrary charging system. In a practical situation, we can substitute those values for the actual charging rate of the monetary metric. Depending on the method of computing the *heat*, which is formalized in Eq.'s (8), (9), (10), and (11), the scheduler generates different service schedules. We compare the efficiency of these four different policies from the aspect of total service cost.

5.2 Experiment 1: Effect of Network Charging Rate

In this section, we mainly focus on visualizing the effect of the network charging rate. Fig. 5 shows the relationship between network charging rate and total service cost in four different storage charging rates. It also plots total service cost in the environment *without* intermediate storage. The advantage of using intermediate storage becomes more significant as the network charging rate increases. *Srate* in Fig. 5 means the storage charging rate. In Fig. 6, the variable is the user access pattern. In the *zipf* distribution, α determines the *skewness* of the user access pattern. Larger α implies a less biased distribution. Under the same environmental parameters, total service cost increases when the requests are more evenly distributed. The advantage of intermediate storage is to share a file between users and avoid repeated delivery of the same video file. When most of the requests are concentrated within a small set of video files, the schedule can maximize the usage of an intermediate storage. With the less expensive storage charging rate, the total service cost increases more slowly.

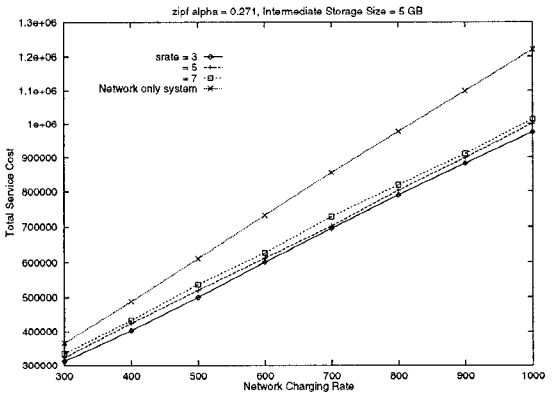


Figure 5. Under Different Storage Charging Rates

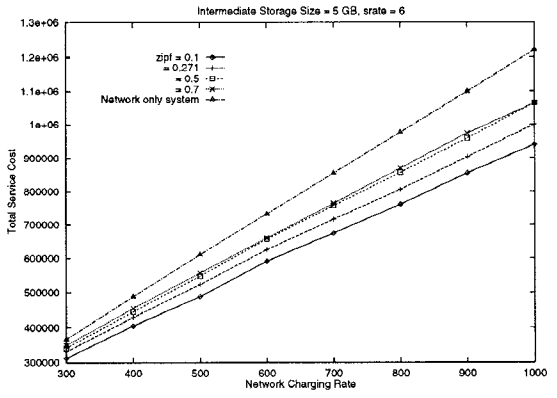


Figure 6. Under Different Access Patterns

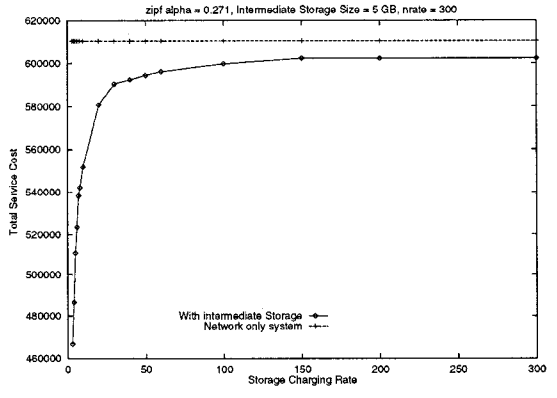


Figure 7. Storage Charging Rate vs. Total Service Cost

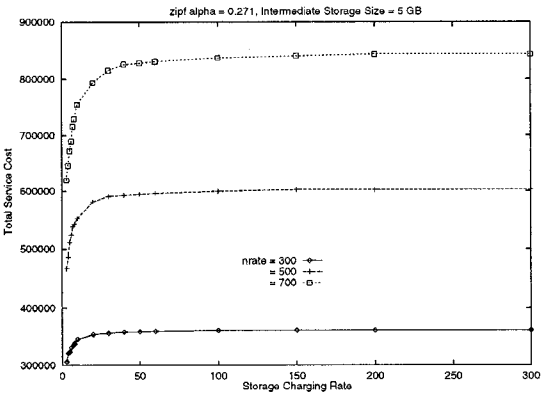


Figure 8. Storage Charging Rate vs. Total Service Cost

5.3 Experiment 2: Effect of the Storage Charging Rate

In this experiment, we examine the effect of the storage charging rate. Fig. 7 illustrates the effect of the storage charging rate on the total service cost. When the storage charging rate is relatively low, the scheduler tries to avoid repeated delivery and prefers using intermediate storage to delivering the video directly from the video warehouse. The cost of using intermediate storage dominates the cost of using network in total service cost in the situation with a low storage charging rate. Hence, the change in the storage charging rate has significant effect on total service cost. As the storage charging rate increases, the scheduler prefers repeated network deliveries to making the video residency in the intermediate storage for a longer period of time. Consequently, the total service cost becomes less sensitive to the increase in the storage charging rate as the storage charging rate increases. The total service cost curve in Fig. 7 approaches the total service cost of the *network only system* as the storage charging rate increases.

Fig. 8 shows the effect of storage charging rate under different network charging rates. It is worth noting that total service cost increases linearly with the increase of the network charging rate. Meanwhile, the effect of the increase in the storage charging rate is substantial only when the storage charging rate is low. This phenomenon arises due to the fact that there are substantial amount of unavoidable network delivery in the service schedule, e.g. servicing the earliest request for each neighborhood.

Let us examine the vertical distances between each line and curve in Fig. 5. Change in *srate* results in shifting the straight line up along the *y-axis*, with an increase in the slope of the curve. The vertical distance between each straight line, i.e. the amount of total cost increase due to the increase in the storage charging rate is small. This implies that the cost for using intermediate storage in the service schedule is relatively small, compared to the total network cost. If the number of users in each neighborhood and/or the

size of intermediate storage increases, the vertical distance between each line will become larger.

5.4 Experiment 3: Effect of Data Access Pattern

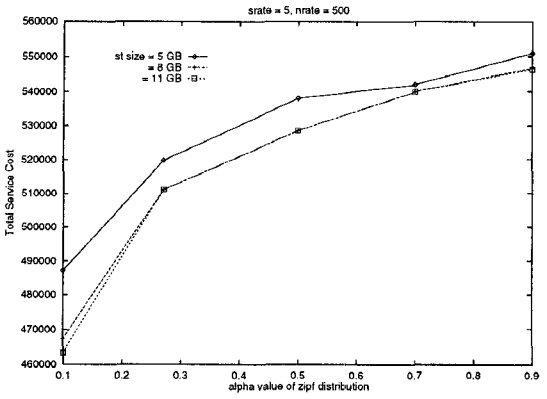


Figure 9. User access pattern vs. Intermediate Storage Size

In this experiment, we visualize the effect of data access pattern and the total service cost. We anticipate that it is more advantageous to build the distributed service environment as the user requests are concentrated on the small set of video files. Dan et al[5] showed that *zipf* distribution with $\alpha = 0.271$ approximates the commercial video rental pattern to a reasonable degree. Fig. 9 visualizes the effect of the data access pattern with various sizes of intermediate storage. As can be observed in Fig. 9, the total service cost increases as the access pattern becomes less biased. The merit of using intermediate storage is to avoid repeated file delivery to the same location. When the access pattern gets evenly distributed, the effect of using intermediate storage becomes less significant and thus total service cost increases. Let us look at the vertical distance between the three graphs in Fig. 9 with storage size is 5GB, 8GB, and 11GB, respectively. The environment with the smaller size intermediate storage results in more expensive total service cost. With the access pattern becomes more biased, the vertical distance between the graphs becomes larger. It implies that advantage of using larger size intermediate storage becomes more significant as user access pattern is more skewed.

5.5 Experiment 4: Computing *heat* and Cost Increase

Rescheduling should be viewed from two aspects - *cost* and *benefit*. The *cost* is the cost of rescheduling a victim and the *benefit* determines the improvement in the overflow situation. By combining these two factors - *cost* and *benefit*, we can obtain the *effective cost* of rescheduling. We provided four different ways of computing *heat* in Eq.'s (8), (9), (10), and (11). We compare the efficiency of schedules obtained under four different *heat* metrics. In 98% of 622 different circumstances, Eq. (9) or Eq. (11) generated the best results, and thus we provide the comparison between the two. Table 5 shows the performance of each metric. The experiments are performed under 785 different combinations of the *network charging rate, storage charging rate, size of the intermediate storage and user access pattern.* Under some situations, e.g. large intermediate storage capacity, or an expensive network charging rate, the scheduler generates an *overflow free* schedule at the individual scheduling phase. There are also the situations where method 2 in Eq. (9) and method 4 in Eq. (11) generate the same output result with the same cost. Overflow

Total Number of Cases	785
Δ Cost by overflow resolution	622
Method 2 in Eq.(9)	395 out of 622 (63 %)
Method 4 in Eq.(11)	437 out of 622 (70 %)
Method 2 or Method 4	614 out of 622 (98%)

Table 5. Performance of the each method

resolution makes the service schedule less efficient, which leads to an increase in the service cost. In our experiments, $\frac{\Psi(\mathcal{S}^{SORP})-\Psi(\mathcal{S})}{\Psi(\mathcal{S})}$ is 12% on the average, and 34% in the worst case. Also, *find_video_schedule*$(\mathcal{R}_i, nrate, srate)$ in Table 2 is known to generate the schedule for file i within the performance bound of 15%[9]. Empirically, the resulting schedule $\mathcal{S}^{SORP}$ is hence within the bound of 30% from the optimal solution on the average.

6 Concluding Remarks

On-line digital delivery of multimedia services makes demands upon the current state of the art of computer technology. Though its application area is expanding rapidly, the high cost of service provisioning has been serious impediment to its widespread usage. In an effort to analyze the costs of service provisioning, we have developed a service paradigm for servicing the requests in a distributed infrastructure. In this work, we developed a comprehensive cost model for storage and networks which elaborately captures the resource requirement of supporting a given set of continous media service. With the combination of a *VOR* service model and a distributed environment with a video warehouse and intermediate storages, system resources such as disk space and network bandwidth can be used in an efficient way via off-line computing and thus can accommodate more video streams and consume less storage or network resources. The algorithm is focused on making the best use of *VOR* property. With the algorithm proposed in this work, the cost of service schedule $\mathcal{S}^{SORP}$ is within 30% performance bound on the average from the optimal solution. Through the extensive simulation, we visualize the effect of the charging rate of the resources and/or user access

pattern on the total service cost. These relationships should be carefully examined in building or prototyping practical information service infrastructure where the parameters are tailored to fit the particular needs. As future extension of the work, we plan to extend our approach to resolve the bandwidth constraints of the intermediate storages and communication network. We hope our design and its heuristic algorithm can serve as useful guideline for the design of future multimedia service provisioning.

References

[1] T. E. Bell and M. J. Riezenman. Technology 1997 analysis and forecast commucations: Communications. *IEEE Spectrum*, pages 27–37, January 1997.

[2] G. Bianchi, R. Melen, and A. Rainoni. Performance of video storage hierarchy in interactive video services networks. In *Proceedings of the 1995 IEEE Global Telecommunications Conference. Part 2 (of 3).*, pages 805–810, Singapore, Singapore, 1995.

[3] D. W. Brubeck and L. A. Rowe. Hierarchical storage management in a distributed VOD system. *IEEE Multimedia Magazine*, 3(3):37–47, Fall 1996.

[4] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in Computer Networks: Motivation, Formulation and Example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, December 1993.

[5] A. Dan and D. Sitaram. Buffer Management Policy for an On-Demand Video Server. Technical Report IBM Research Report RC 19347, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, 1993.

[6] Y. Doganata and A. Tantawi. Making a cost-effective video server. *IEEE Multimedia*, pages 22–30, Winter 1994.

[7] T. Halfhill. Break the bandwidth barrier. *BYTE*, pages 68 – 80, September 1996.

[8] W. T. M. Kienzle, D. Sitaram. Using a storage hierarchy in movie-on-demand servers. Technical report, IBM T.J. Watson Research Center, Hawthorne, NY, 1994.

[9] C. Papadimitriou, S. Ramanathan, and P. Rangan. Information caching for delivery of personalized video programs on home entertainment channels. In *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, pages 213–223, Piscataway, N.J., 1994. IEEE Press.

[10] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. *Computer Communication Review*, 26(2):19–43, April 1996. ACM SIGCOMM.

[11] J. Sutherland and L. Litteral. Residential Video Services. *IEEE Communications Magazine*, 30(7):36–41, July 1992.

[12] D. Towsley. Providing quality of service in packet switched networks. Technical report, Dept. of Computer Science, University of Massachesetts, Amherst, 1993.

[13] Y. Won and J. Srivastava. Scheduling video delivery in a distributed environment. In I. Miloucheva, editor, *Proceedings of Second Conference on Protocols for Multimedia Systems*, pages 119–135, Salzberg, Austria, Oct. 1995.

[14] Y. Won and J. Srivastava. Minimizing blocking probability in hierarchical storage based video-on-demand server. In *Proc. of IEEE International Workshop for Multimedia Database Management Systems*, pages 12 – 19, Blue Mountain Lake, NY, Aug 1996.

[15] Y. Won and J. Srivastava. Parametric Evaluation of Performance Behavior in Hierarchical Storage Architecture. *Submitted to International Journal of Performance Evaluation*, 1997.

[16] Y. Won and J. Srivastava. Strategic Replication of Video Files in a Distributed Environment. *International Journal of Multimedia Tools and Application*, To Appear.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zapala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 5:8–18, September 1993.