Scheduled Video Delivery—A Scalable On-Demand Video Delivery Scheme

Min-You Wu, *Senior Member, IEEE*, Sujun Ma, and Wei Shu, *Senior Member, IEEE*

***Abstract*—Continuous media, such as digital movies, video clips, and music, are becoming an increasingly common way to convey information, entertain and educate people. However, limited system and network resources have delayed the widespread usage of continuous media. Most existing on-demand services are not scalable for a large content repository. In this paper, we propose a scalable and inexpensive video delivery scheme, named *Scheduled Video Delivery (SVD)*. In the SVD scheme, users submit requests with a specified *start time*. Incentives are provided so that users will specify the start times that reflect their real needs. The SVD system combines requests to form the multicasting groups and schedules these groups to meet the deadline. SVD scheduling has a different objective from many existing scheduling schemes. Its focus has been shifted from minimizing the waiting time toward meeting deadlines and at the same time combining requests to form multicasting groups. SVD compliments most existing video delivery schemes as it can be combined with them. It requires much less resources than other schemes. Simulation study for the SVD performance and the comparison to other schemes are presented.**

***Index Terms*—Content delivery, continuous media, incentives, multicast, Video-on-Demand.**

I. Introduction

In the last decade, we have witnessed a rapid growth of continuous media traffic over the networked world. The characteristics of continuous media are different from traditional text-based or image-based files. Typically, continuous media imposes high bandwidth and real-time requirements. Although audio and video can be used to convey information, entertain and educate people, the media applications are not widespread yet. This is largely due to limited system and network resources. Most current media delivery systems are best-effort. Servers and networks are designed for the peak time and the resources are underutilized during the nonpeak time. The server and network capacities become the major concern of continuous media services. Since media is delivered upon requests, rigid playback deadlines coupled with resource constraints make continuous media delivery a challenging task.

With existing video services, such as Video-on-Demand (VoD), batching and patching, the number of channels required increases with the number of clients when serving a large number of videos. Here, a channel represents the bandwidth requirement to deliver one video stream. In VoD, the number of channels is the same as the number of requests. The batching scheme delays the delivery of a request by a batching time, so that all requests that arrived during the batching time can be merged. The patching scheme is scalable to popular videos as its required number of channels is proportional to the logarithm of the number of requests for the video [1]. However, batching and patching are not scalable to a large video repository [2]. When the number of videos is very limited, the Near VoD (NVoD) can service any number of clients, but also with a certain period of waiting time. With the limited resources, only dozens of videos can be serviced. Searching for a scheme that is scalable to a large number of video objects is indeed a challenging task. This objective can be made possible if the fact that not all contents are needed at the request time is considered. In many situations, people can plan ahead to request some content before it is actually consumed, which provides opportunities for the *Scheduled Video Delivery (SVD)* scheme. The SVD approach was first introduced in [3]. With SVD, the traffic can be smoothed by shifting the peak-time traffic to the nonpeak time. Furthermore, requests can be combined to reduce the server load and network traffic. A similar work with a different approach was presented in [4], where a reservation system is proposed to reduce the stream requirement.

The SVD scheme uses the system resources efficiently by fully utilizing the time interval from when the request is made to when the user actually views the requested video. The large-scale applications of continuous media call for the development in two directions: increasing resource provision and reducing resource consumption. With the SVD scheme, many peak-time requests can be placed and executed ahead of time to utilize the nonpeak-time server access and network bandwidth. It amounts to adding more underlying infrastructure to meet a higher demand at peak-time; therefore, increasing provision of the resource without incurring hardware costs. In the SVD scheme, requests that are submitted ahead of time can provide a great opportunity for the efficient scheduling and implicit combining, thereby reducing the overall resource consumption. In addition to multicasting or broadcasting, which reduces the resource requirement by combining requests across space, SVD combines requests across time as well.

This paper will describe the SVD architecture. Related literatures are briefly discussed in Section II. The SVD architecture is proposed in Section III. In Section IV, we discuss incentives and pricing for SVD. The system and scheduling are described in Sections V and VI, respectively. The methodology and results of the simulation experiments to test the effectiveness of

Manuscript received April 7, 2003; revised January 26, 2005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ton Kalker.

M.-Y. Wu is with the Department of Computer Science and Engineering, Shanghai JiaoTong University, Shanghai, China (e-mail: wu-my@cs.sjtu.edu.cn).

S. Ma and W. Shu are with the Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM 87131-0001 USA (e-mail: sjma@ece.unm.edu; shu@ece.unm.edu).

Digital Object Identifier 10.1109/TMM.2005.861280

this SVD scheme are described in Section VII. Finally, discussion is provided together with the conclusion in Section VIII.

II. Related Research

The VoD service is attractive since it is selective and responsive [5]–[7]. With SVD, the selectivity is retained, but the responsiveness has been changed into a plan-ahead and guaranteed arrival. This alternative can be justified in several situations. Users may be very happy to exercise their planning discipline. Pricing can be differentiated to provide a further incentive. While the VoD service can still be effective for short clips, SVD will be suitable for high-quality, full-length movies and education/training courseware applications.

It is critical to investigate new methods for the scalable video delivery. Two approaches for the scalable video delivery have been developed. The open-loop approach [8]–[10] requires no return path so it can be used for an one-way cable system, whereas the closed-loop approach [11]–[13] requires a two-way cable system. The closed-loop system only delivers the requested video to users whereas the open-loop system continually broadcasts a video even if nobody watches it. Normally, only dozens of videos can be provided simultaneously in an open-loop system. In the closed-loop approach, a number of methods, such as batching [11], [14]–[17], patching [13], [18], catching [19], stream tapping [20], and stream merging [21], are used to combine more requests to minimize the number of broadcast or multicast streams. Many research works on the closed-loop approach emphasize merging more requests while minimizing the waiting time. However, a common fact has not been noticed, that is, not all users want to view the requested video immediately. By utilizing this characteristic, a video delivery scheme can become much more efficient. The major difference between SVD and most batching schemes is that we changed the "waiting" model to the "plan-ahead" model. Since the users may plan ahead, they watch the video at their desired time instead of simply "waiting" for the video.

Digital fountain [22] is a scalable approach to disseminating large files to many users. It is not suitable for video delivery with short deadlines. However, it can be used together with SVD to deliver videos. Other approaches to improving scalability are caching [23], mirroring [24], or CDN [25]. These methods can be used with SVD to maximize the scalability. Two commercial works, TIVO and Replay, can record the pre-scheduled broadcast streams. It facilitates the personal multimedia system. The SVD shares this idea with availability of storage and plan-ahead, but changes this passive scheme into an active one.

Interactive Multimedia Jukebox (IMJ) reduces the channel requirement by scheduling requests to a certain time period, in the hope that other users will watch the already-scheduled videos [26]. Improved from IMJ, SVD presumes some user storage space, so the time requirement is further relaxed.

III. Scheduled Video Delivery

The SVD scheme has a number of advantages compared to the traditional VoD. First, SVD extends a user's option from click-wait-see patterns to plan-ahead alternatives, being able to submit requests with timing specification. Second, it enriches servers with the capability of efficiently combining and scheduling requests. Third, it alleviates the peak-time overloading problem and improves utilization of the limited bandwidth resources. We assume a broadcast or a multicast scheme so that a number of requests can be combined and delivered together. A cable (Hybrid Fiber/Coax) system can be used for this purpose. The Internet with multicast capacity will also satisfy this requirement. Different from patching, where the decoding capacity of at least two channels is required, SVD assumes that the client is able to decode only one channel.

In the SVD scheme, a user makes a request of video with a *start time*. The server schedules a time slot for delivery of the video to the user's storage space on or before the start time. The start time can be as short as a few seconds and as long as 24 h or more. A pricing scheme controls the overall cost for the video delivery. This scheme also can be applied to content delivery from the original server to proxy servers. Some contents such as news and sports need to be delivered as soon as they become available.

IV. Incentives and Pricing

The optimal efficacy can be achieved only if many requests are planned ahead. What will motivate users to do so? Why won't they always wait until the last minute to make a request? How to create incentives is crucial to the success of the SVD scheme. Without an incentive mechanism, every user might request the shortest possible start time that does not reflect their real needs and the system resources could be underutilized. Usage-disciplines and pricing policies could be combined to present users such an incentive system.

The usage-disciplines can be justified in several situations. Planning is a common discipline in our daily life, users could be very happy to practice planning if a proper infrastructure is available. The community can be educated to change their usage-patterns to conserve resources in a global environment. More importantly, if the peak-time on-demand requests most likely experience high rejection rate or low-quality video, users tend to plan-ahead for a guaranteed on-time, high-quality delivery.

The plan-ahead requests can also be encouraged by price differentiation. Various pricing schemes have been investigated by many researchers. In studies of [27], the performance penalty received for requesting a less-than-optimal service class is offset by the reduced price of the service. In the SVD scheme, the plan-ahead requests still receive the high-quality video in time. The works in [28] built a usage-sensitive pricing model. The SVD scheme shares this similarity to penalize the instant requests during congestion time, but also provides alternatives to place a request earlier and still to consume the media at the peak time. While the scheme presented by [29], [30] triggers increasing of price when congestion is dynamically detected, SVD has placed an emphasis on shaping the bandwidth consumption in a larger time domain to provide a variety of price choices.

In the SVD scheme, a pricing scheme is integrated to control the overall cost for delivering videos. Considering a broadcast/multicast based system as an example. The cost of the video delivery consists of what is to be paid to the content producer, c_c, and the cost of video delivery, c_d. Normally, c_c is a constant for a given video, and c_d depends on the number of requests that can be combined for delivery. The number of requests to be combined is proportional to the plan-ahead time $t_{plan} = t_{start} - t_{arrival}$. The cost of delivery is no more than $c_d = c_1/\lceil \lambda \cdot t_{plan} \rceil$, and the total cost

$$\mathcal{C}_t = c_c + \frac{c_1}{\lceil \lambda \cdot t_{plan} \rceil}$$

where c_1 is the cost to deliver a single video object. Since λ is the request arrival rate for the video, $\lambda \cdot t_{plan}$ is the number of requests for the video during the plan-ahead time. When this number is less than 1, no request can be combined, and the delivery cost is c_1. A popular video will cost less to deliver even for a short plan-ahead time since its λ is larger than a nonpopular video.

A sample price for a video is shown in Table I, assuming $c_c = \$0.5$ and $c_1 = \$6$, For a video with $\lambda = 0.1$, if you would like to watch it at 7 pm and make an instant request, you will pay \$6.50. However, if you make a request before going home at 5 pm, you will pay only \$1. Note that the prices for a 1-min request and for a 10-min request are the same since both have a little chance to combine other requests. Furthermore. the price of a video depends not only on the plan-ahead time, but also on the popularity of the video.

When the arrival rate increases, the price will be lower. As an example, for a video of $\lambda = 1$, a request on the video with a 6-min plan-ahead time costs only \$1.50. On the other hand, people have to wait for a long time to watch a nonpopular video with a low price. Providers have to confront issues of pricing and cost recovery. This price scheme will be published in advance or made known to the users during QoS negotiation. SVD provides flexibility to both providers and users. Providers are able to deploy the service incrementally. The high price discourages short-time requests or generates revenue for service expansion otherwise.

V. System Overview

There are three basic entities in the SVD scheme: a Servicing Unit (SU), a Planning Unit (PU), and a Consuming Unit (CU). An SU consists of several modules: a scheduling module, a transmission module, and a content management module. A request for video delivery is initiated from the PU to the SU. It goes through QoS negotiation, and is admitted and scheduled at the SU. Based on the agreed delivery schedule, the content will be transmitted from the SU to the CU in time.

Content Management Module of SU: This module maintains the record of N video objects, $\mathcal{O}_1, \ldots, \mathcal{O}_N$. For each object $\mathcal{O}_j$, $l(\mathcal{O}_j)$ is its playback length in minutes. Though objects may have different bit rates and a single object may have a variable-bit-rate (VBR), in this paper we assume all objects have the same constant bit rate r. The popularity of an object, $p(\mathcal{O}_j)$, varies from time to time, 1 is the highest and N is the lowest.

Planning Unit: A request $\mathcal{R}_i$, arrived at $a(\mathcal{R}_i)$, specifies its start time of consumption as $s(\mathcal{R}_i)$. The plan-ahead time $q(\mathcal{R}_i)$

TABLE I
Sample Price for $\lambda = 0.01, 0.1,$ and 1

t_{plan}	$\leq$ 100 min.	200 min.	5 hr.	10 hr	20 hr
$\lambda = 0.01$	\$6.50	\$3.50	\$2.50	\$1.50	\$1.00
t_{plan}	$\leq$ 10 min.	20 min.	30 min.	1 hr	2 hr
$\lambda = 0.1$	\$6.50	\$3.50	\$2.50	\$1.50	\$1.00
t_{plan}	$\leq$ 1 min.	2 min.	3 min.	6 min.	12 min.
$\lambda = 1$	\$6.50	\$3.50	\$2.50	\$1.50	\$1.00

is equal to $s(\mathcal{R}_i) - a(\mathcal{R}_i)$ The planning unit performs QoS negotiation with the scheduling module based on the available resources from SU.

Transmission Module of SU: Considering the outgoing network bandwidth as well as the server's output capacity, the total available bandwidth W can be equally partitioned into M channels if all the streams have the same bit rate r, thus $M = W/r$. Consequently, the requests can be mapped onto the time domain of M communication channels. When the requests for the same object can be combined, an upper-bound of the bandwidth requirement for the object can be established based on the object length $l(\mathcal{O}_j)$, bit rate r, and plan-ahead time $q(\mathcal{R}_i)$. Assume all requests have the same $q(\mathcal{R}_i)$, denoted as t_{plan}:

$$B = r \times \frac{l(\mathcal{O}_j)}{t_{plan}} = \frac{r}{\beta}$$

where $\beta = (t_{plan})/l(\mathcal{O}_j)$ represents a ratio of the plan-ahead time to the length of an object. The higher the ratio, the less bandwidth is required. Note that this upper-bound of bandwidth requirement is independent of the total number of requests. This relationship indicates that longer plan-ahead time implies less bandwidth consumption.

Scheduling Module of SU: The functionality of this module is to apply a scheduling algorithm to realize mapping

$$\mathcal{F}_{alg}(\mathcal{R}_k, \mathcal{C}^k, \mathcal{P}^k) \Longrightarrow \begin{cases} \mathcal{C}^{k+1}, & \text{a new schedule} \\ \mathcal{P}^{k+1}, & \text{a new waiting pool} \end{cases}$$

where P^k is the current waiting pool and C^k the current schedule for each individual channel. Upon the arrival of request $\mathcal{R}_k$, the scheduling module is invoked to change the schedule from C^k to C^{k+1}, as well as the waiting pool from P^k to P^{k+1}.

Requests for the same object could be combined into a group as long as they all meet deadlines. A group is the basic unit for scheduling. To maintain these groups, N pools, $P_1, P_2, \ldots, P_N$, are formed as $P_j = \{\mathcal{G}_{j,0}, \ldots, \mathcal{G}_{j,n_j-1}\}$, where, n_j is the number of groups for object $\mathcal{O}_j$. For each group

$$\mathcal{G}_{j,v} = \langle U_{j,v}, e_{j,v}, c_{j,v}, t_{j,v}, d_{j,v} \rangle$$

where

- $U_{j,v} = \{\mathcal{R}_i | o(\mathcal{R}_i) \equiv j, a(\mathcal{R}_i) \leq t_{j,v} \leq s(\mathcal{R}_i)\}$ all requests in group $\mathcal{G}_{j,v}$;
- $e_{j,v} = \min_{\mathcal{R}_i \in U_{j,v}} (s(\mathcal{R}_i) + l(\mathcal{O}_j))$ the deadline of $\mathcal{G}_{j,v}$;
- $c_{j,v}$ the channel that $\mathcal{G}_{j,v}$ is scheduled to;
- $t_{j,v}$ the time that $\mathcal{G}_{j,v}$ is scheduled to;
- $d_{j,v}$ the length of object $\mathcal{O}_j$ that $\mathcal{G}_{j,v}$ has delivered.

All requests for $\mathcal{O}_j$ that have not started their delivery should be in the same group. The deadline of $\mathcal{G}_{j,v}$ is defined as the earliest

```
For a newly arrived request $\mathcal{R}_k$, the requested video object is $O_j$
      init $need\_resch = false$
if there exists a group $\mathcal{G}_{j,i}$ with $d_{j,i} \equiv 0$
      join group $\mathcal{G}_{j,i}$ by adding $\mathcal{R}_k$ to $U_{j,i}$
      if $(s(\mathcal{R}_k) + l(O_j)) < e_{j,i}$, the deadline of $\mathcal{R}_k$ is earlier
            $e_{j,i} = s(\mathcal{R}_k) + l(O_k)$; update the deadline
            $need\_resch = true$; need to reschedule
else create a new group $\mathcal{G}_{j,n_j}$ in $\mathcal{P}_j$ for object $O_j$
      let $v = n_j$ and $n_j = n_j + 1$, add $\mathcal{R}_k$ to $U_{j,v}$
      $e_{j,v} = s(\mathcal{R}_k) + l(O_j)$; set the deadline
      $need\_resch - true$; need to reschedule
if $need\_resch \equiv true$
      for every channel $c = 1, 2, ..., M$, let $T_c^{(k+1)} = t_{now}$;
      sorting all groups in the ascending order of their deadlines $e_{j,i}$
      reschedule each group $\mathcal{G}_{j,i}$ from the sorted list
            find a channel $c$ with the earliest available time $T_c^{(k+1)}$
            if $(T_c^{(k+1)} + l(O_j) - d_{j,i}) \leq e_{j,i}$, the deadline can be met
                  $t_{j,i} = T_c^{(k+1)}$ and $c_{j,i} = c$; schedule $\mathcal{G}_{j,i}$ onto channel $c$
                  $T_c^{(k+1)} = T_c^{(k+1)} + l(O_j) - d_{j,i}$; update available time
            else fail to schedule the new request $\mathcal{R}_k$
                  restore $C^{(k+1)} = C^k$ and $\mathcal{P}^{(k+1)} = \mathcal{P}^k$
```

F_{MEDF}: Modified Earliest Deadline First (MEDF) Algorithm

Fig. 1. F_{MEDF}: Modified Earliest Deadline First (MEDF) algorithm.

```
For a newly arrived request $\mathcal{R}_k$, the requested object is $O_j$,
   apply $\mathcal{F}_{MEDF}$ to obtain an initial schedule
each channel $c$ has the earliest available time $T_c$.
For each channel $c$, reconstruct an as-late-as-possible schedule
   let $\mathcal{A}_c = \{(a_{c,1}, a'_{c,1}), ...\}$;  the list of available slots
   let $\mathcal{B}_c = \{(b_{c,1}, b'_{c,1}), ...\}$;  the list of booked slots
   initially, $\mathcal{A}_c = \{(T_c, \infty)\}$, $\mathcal{B}_c = \{(t_{now}, T_c)\}$ and $L_c = \infty$
   mark all groups with $d_{j,0} > 0$ as *affinity*; those cannot be moved
   mark the other groups as *nonaffinity*; those can be moved
   For all *nonaffinity* groups with $c_{j,i} \equiv c$ (those scheduled at channel $c$)
      sort them in descending order of $t_{j,i}$
      to shift group $\mathcal{G}_{j,i}$ to its latest available slot
         move slot $(t_{j,i}, t_{j,i} + l(O_j))$ from $\mathcal{B}_c$ into $\mathcal{A}_c$;
            modify $t_{j,i} = \min(e_{j,i}, L_c) - l(O_j)$
         move slot $(t_{j,i}, t_{j,i} + l(O_j))$ from $\mathcal{A}_c$ into $\mathcal{B}_c$;
            modify $L_c = t_{j,i}$, a new boundary
For all *nonaffinity* groups
   sort them in the ascending order of $p(O_j)$; less popular objects first
   to move $\mathcal{G}_{j,i}$ with the less popular stream forward
      move slot $(t_{j,i}, t_{j,i} + l(O_j))$ from $\mathcal{B}_c$ into $\mathcal{A}_c$;
      find a channel $b$ with an earliest $(a_x, a'_x)$ from $\mathcal{A}_b$,
         such that $a'_x \leq e_{j,i}$, and $a'_x - a_x \leq l(O_j)$
      move slot $(a_x, a_x + l(O_j))$ from $\mathcal{A}_b$ into $\mathcal{B}_b$;
```

F_{LPF}: Least Popularity First (LPF) Algorithm

Fig. 2. F_{LPF}: Least Popularity First (LPF) algorithm.

deadline among all requests in the group, and $d_{j,v}$ maintains the length of object $\mathcal{O}_j$ that group $\mathcal{G}_{j,v}$ has delivered.

Consuming Unit: This unit manages the user's resources and provides facility for a user to consume the media object requested. Examples of resources to be handled include storage space and network bandwidth. It may include the demodulation unit, decoding unit, and player. At the playback time, interactive functions such as fast-forward and jump can be performed as long as the media content resides in the local storage. In the case that the storage space is tight, buffer management should be taken care by the consuming unit too.

VI. SVD Scheduling

The main objective of scheduling is to minimize the rejection rate. This in turn, can be described as two objectives.

- Schedule deliveries to meet the requested start time.
- Combine as many as possible requests together to minimize the resource consumption.

The first objective is essential and the contribution of the other is twofold. First, combining requests reduces the resource requirement for a given load and minimizes provider's cost. Second, it leaves more space to help other requests meet their deadlines. When a request is submitted, a schedule will be computed. If a feasible schedule exists, the request will be admitted; otherwise, it is rejected or further negotiated.

Since SVD scheduling is a real-time processing problem, the well-known Earliest Deadline First (EDF) algorithm can be applied with some modification. In addition, EDF used in this case also combines as many as possible requests for the same object. This modified EDF algorithm with combining is named the MEDF algorithm as shown in Fig. 1.

Upon arrival of request $\mathcal{R}_k$ for object $\mathcal{O}_j$, we first check if there is any request group in P_j whose delivery has not started. If so, the newly arrived request can be combined into the existing one. In fact, if there is such a group, it must be the only group for the object. As a result of combining, rescheduling is required if and only if $s(\mathcal{R}_k)$ is earlier than the earliest start time of the existing requests. If there exists no such a group, a new group will be created and rescheduling is invoked.

The scheduling procedure is applied to groups instead of requests. The number of groups is usually much smaller than the number of individual requests, especially for those popular objects. This multiple-channel scheduling uses a simple heuristics based on the earliest-deadline-first criteria. That is, the group with the earliest deadline will be scheduled first. It is a preemptive scheduling algorithm. The newly arrived request with an earlier deadline can preempt a running process as long as the deadline of the process is met. When there exists no feasible schedule, the newly arrived request will be rejected.

Although this algorithm can combine requests into groups, it does not minimize the number of groups. To combine as much as possible requests, the request groups for more popular objects should postpone their delivery as long as the deadline is not missed. Such a group has a better chance to combine more upcoming requests. An algorithm that delays delivery of popular objects can maximize combining. It is called the Least Popularity First (LPF) algorithm as shown in Fig. 2. First, the MEDF algorithm is applied to generate an initial schedule and at the same time, the schedule-ability is tested. Second, the schedule is modified by pulling all delivery slots as late as possible without violating their deadlines. The popularity of an object is then utilized. Based on the schedule produced, the request groups for less popular objects are moved forward if feasible. The request groups for more popular objects tend to be left behind, increasing opportunities of combining.

VII. Experimental Results

In this section, we conducted various experiments to test the performance of SVD. Since it is difficult to model the user's behaviors properly, the challenge in evaluating pricing strategies is not easily solved with analysis. In this study, we model the user's behavior based on the user's *intention* and/or their *budget*. Without the concerns of budget, a user may have the freedom to choose a plan-ahead time that reflects his/her intention. This is defined as *intention-driven*. On the other hand, a user may behave as *budget-driven*, that dictates how much a user is willing to pay for a video. We propose two models for an intention-driven user. The base model that assumes the plan-ahead time is uniformly distributed in a time period. In this simulation, the plan-ahead time is uniformly distributed between 1 and 1440 min (24 h). The base model provides a simplified user behavior, which is used in most of the simulations to provide a quick view of SVD performance. In order to understand the potential impact on performance when the plan-ahead time of requests is not uniformly distributed, the other model assumes a variety of user intentions, which is named as the xyz model. A pattern of the plan-ahead time distribution is defined by xyz, where $x + y + z = 10$, and

- $10 \cdot x\%$ of the users expect to watch videos instantly;
- $10 \cdot y\%$ of the users expect to watch videos within 2 h, whose plan-ahead time is uniformly distributed between 1 and 120 min; and
- $10 \cdot z\%$ of the users expect to watch videos at a time uniformly distributed between the 121th and 1440th minute.

Three patterns used in this simulation are 136, 244, and 433.

We also propose two models for a budget-driven user. The *fixed budget* model assumes that each user has a maximum budget allocated for a video. The budget of each user follows the power law:

$$B_i = \frac{W}{i \cdot \sum_{j=1}^{V} \frac{1}{j}} + c_c$$

where V is the number of users and W is a cost coefficient. Here, each request is made by a different user. For each request, the plan-ahead time is computed from the available budget as follows:

$$t_{plan} = \begin{cases} 1, & if\ B \geq c_c + c_1 \\ \left\lceil \frac{c_1}{(B_i - c_c) \cdot \lambda} \right\rceil, & if\ B < c_c + c_1. \end{cases}$$

The request with a higher budget will have a shorter plan-ahead time. In particular, a request with a budget that is larger than or equal to $c_c + c_1$ is an instant request for any video. On the other hand, a request with a low budget may lead to a plan-ahead time that is longer than 24 h. In this case, the user will randomly reselect a video until the plan-ahead time is within 24 h. The second model, named the *thrift* model, does not fix the user budget. Instead, it assumes that selection of the plan-ahead time is influenced by its cost. In reality, when users are aware of price incentives, they tend to make a longer plan-ahead time.

Table II shows various scheduling policies to be used. The simulation environment is described here. The number of video objects, N, is 1000 by default. For each request generated, a video is selected using the Zipf distribution [31]. The video length is chosen from a uniformly distribution between 100 and 140 min with an average of 120 min. The request arrival rate a is set from 1 to 100/min. The request arrival pattern is the Poisson distribution.

TABLE II
DESCRIPTION OF SCHEDULING POLICIES

Policy	Description
$\mathcal{F}_{OnDemand}$	Reject a request if it cannot be satisfied in one minute
$\mathcal{F}_{Batching}$	Force delay of delivery up to 10 minutes to wait for more upcoming requests and reject the request if it cannot be satisfied in 12 minutes
$\mathcal{F}_{Patching}$	The on-demand plus patching policy, rejecting a request if it cannot be scheduled
$\mathcal{F}_{MEDF}$	The modified earliest-deadline-first policy, rejecting a request if it cannot be scheduled
$\mathcal{F}_{LPF}$	the less-popularity-first policy, rejecting a request if it cannot be scheduled

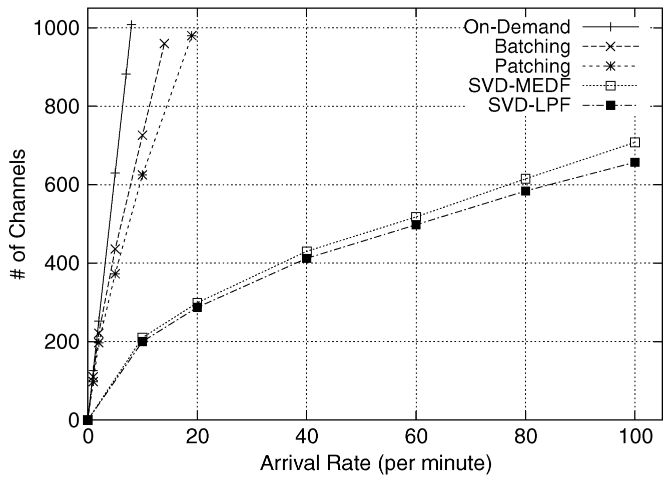

Fig. 3. Number of channels required.

Performance of various scheduling policies are compared against two metrics: i) the required number of channels if no request is rejected; and ii) the rejection rate for a fixed number of channels. Fig. 3 shows the number of channels required for the five policies. Here, the base model is used. The number of channels required by $\mathcal{F}_{OnDemand}$ is proportional to the arrival rate. $\mathcal{F}_{Batching}$ and $\mathcal{F}_{Patching}$ require less channels. It can be seen that $\mathcal{F}_{LPF}$ for SVD performs much better than other scheduling policies. The number of channels required by the batching policy is 51% to 56% of that of the on-demand policy. The number of channels required by the patching policy is 46% to 52% of that of the on-demand policy. $\mathcal{F}_{MEDF}$ for SVD requires 708 channels when the arrival rate is 100/min. $\mathcal{F}_{LPF}$ further improves the performance and requires only 657 channels. The advantage of $\mathcal{F}_{MEDF}$ is its simplicity and its performance is fairly good. The difference between $\mathcal{F}_{MEDF}$ and $\mathcal{F}_{LPF}$ is within a few percent. The comparison of theoretical and experimental results for SVD is shown in Table III. The LPF algorithm generates schedules that are close to perfect scheduling.

When the resources are limited, some requests will be rejected as the arrival rate increases. Fig. 4 shows the rejection rates on 200 channels for these policies. $\mathcal{F}_{Batching}$ and $\mathcal{F}_{Patching}$ improve the rejection rates compared to

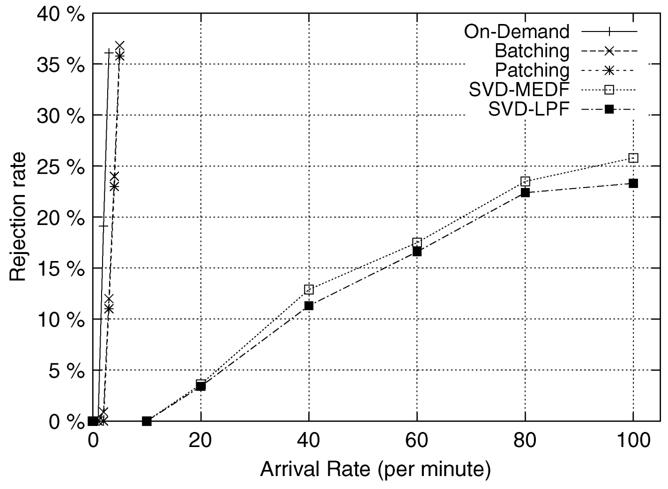

Fig. 4. Rejection rates on 200 channels.

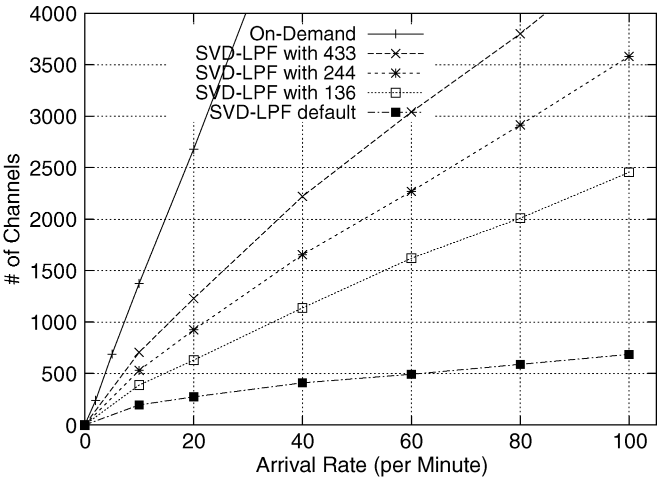

Fig. 5. Number of channels required with different request patterns.

TABLE III
THE CHANNEL REQUIREMENT FOR SVD

Arrival rate (per minute)	20	40	60	80	100
Theoretical result	276	380	461	529	589
Experimental result (LPF)	287	412	498	584	657

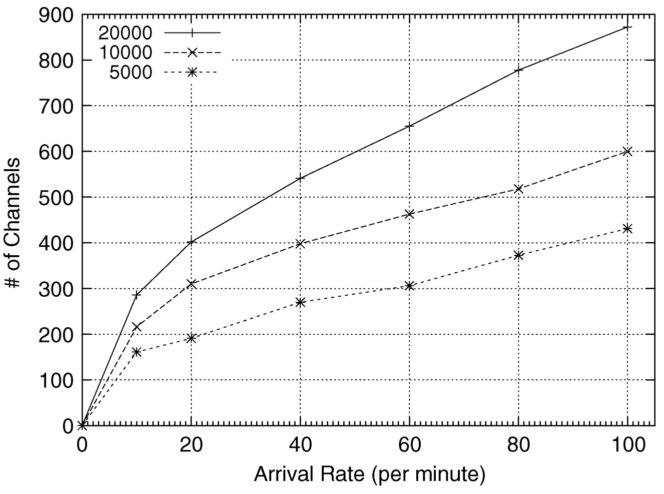

Fig. 6. Number of required channels with fixed budget.

$\mathcal{F}_{OnDemand}$. For the same rejection rate, both of them can serve about twice of that for the on-demand policy. SVD utilizes the user-provided deadlines and significantly reduces the rejection rate.

Consider a HFC system, assume the system capacity is 200 channels which is equivalent to 25 analog cable channels for 3 Mbps MPEG-2 videos. Assume 6 h of peak time per day [32] and the arrival rate in the peak time is five times higher than that in nonpeak time [33]. With rejection rate of less than 5%, the $\mathcal{F}_{OnDemand}$ service can serve up to 1000 requests, the $\mathcal{F}_{Batching}$ and the $\mathcal{F}_{Patching}$ policies serve about 2000 requests, and $\mathcal{F}_{LPF}$ about 35 000 requests in 24 h. The SVD can serve about 35 times more requests compared to the on-demand service.

Next, the performance using the xyz pattern is presented. Here, three patterns are studied, which are 136, 244, and 433. For example, with the 136-pattern, 10% of requests are on-demand, 30% of requests are willing to wait for 1 min to 2 h, and 60% of requests are willing to wait for 2 to 24 h. On the extreme of the spectrum, the pure on-demand pattern is equivalent to the case of $x = 10$, $y = 0$, $z = 0$. For the SVD LPF, Fig. 5 shows the number of channels required with respect to different plan-ahead time patterns. Resource requirements are rapidly increased with the number of on-demand requests. Thus the pricing incentives are crucial for this scheme.

The above performance shows the situation when the user's choice of plan-ahead time is not influenced by the price. Now, we present the performance of the fixed budget model. The value of W is set to be \$20 000, \$10 000, and \$5000. This value determines the users' budgets for a video. For example, when $W = \$10\,000$, the richest user has about \$900 and the poorest user has \$0.53 for a video. Fig. 6 shows the channel requirement. It can be seen that more channels are required when users have a bigger budget. In one extreme, when the users have very high budgets, the system becomes a true VoD system and more channels can be provided with the high budget. On the other hand, if the user's budget is very low, the system becomes a download system. In this case, videos can be delivered to multiple users through multicast with a low cost. Thus, this system is adaptive to various user budgets and requirements.

In Fig. 7, the thrift model is applied. Assuming $c_c = \$0.5$ and $c_1 = \$6$, the number of channels with and without the influence of pricing for the $\mathcal{F}_{LPF}$ policy are shown in Fig. 7, where "LPF-thrift" is for performance with the thrift model. The figure shows that the number of channels can be reduced when users under the influence of pricing.

With SVD-LPF and the default plan-ahead time pattern, we illustrate more characteristics to help further understanding of the SVD scheme. Fig. 8 shows the average group sizes corresponding to the top 50 popular video objects when the arrival rate is 20, 60, 100 requests/min, respectively. The curve shapes roughly approaching the Zipf distribution. The average group sizes decrease as expected when the video objects become less popular. Obviously, the higher the arrival rate, the larger the average group size. Consequently, it is interesting to check the correlation between the popularity of video objects and the number

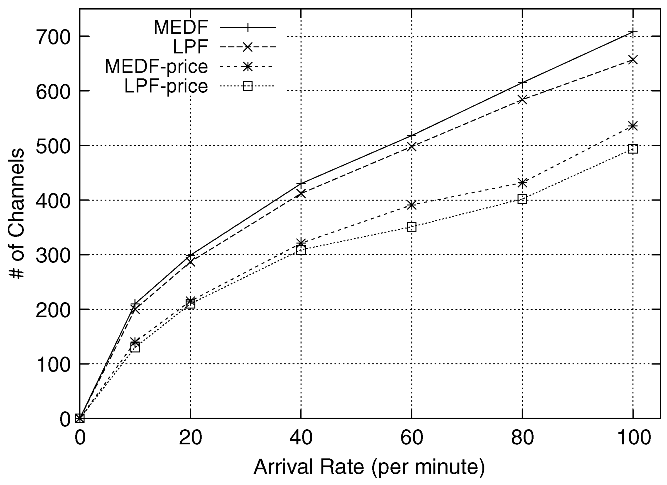

Fig. 7. Number of required channels influenced by pricing.

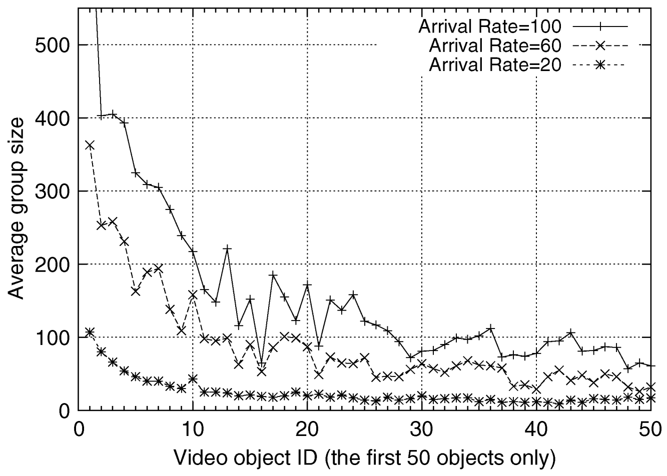

Fig. 8. Average group sizes with different video object IDs.

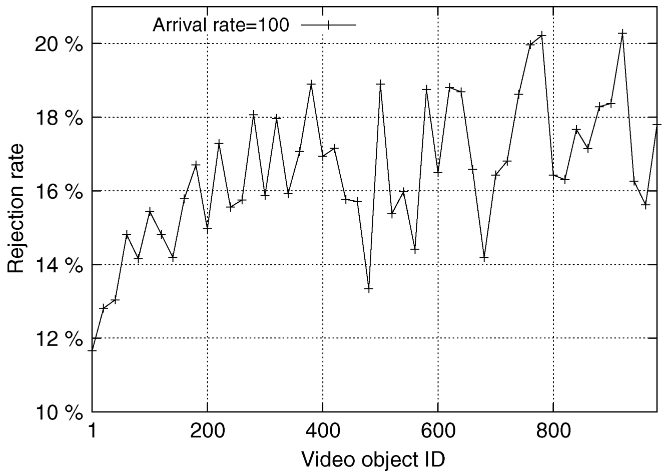

Fig. 9. Rejection rates for the different video object IDs.

of requests to be rejected. The percentage of the rejected requests for a particular video object is measured to illustrate the correlation in Fig. 9. Apparently, a popular video object has a relatively low rejection rate since a newly-arrived request has a better opportunity to be merged into an existing group. Though

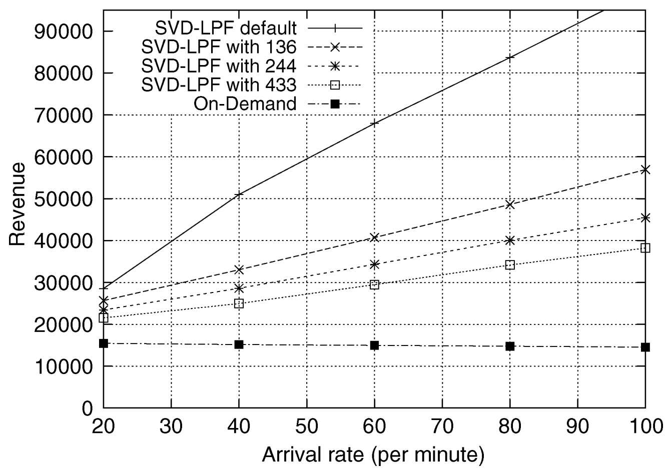

Fig. 10. Revenue comparison with different request arrival patterns.

the difference is not significant, this problem could be addressed by reserving some channels for the less popular videos.

The emphasis of our scheduling algorithms is maximizing the utilization of the video delivery system and minimizing the rejection rate. Also, a service provider will be more interested in revenue gained from all serviced requests. Shown in Fig. 10 is the revenue for different access patterns, given the same amount of resources, 200 channels. The on-demand service generates the constant revenue when the arrival rate is larger than 20/min since the system is saturated. For other access patterns, under the same pricing model presented in Section IV, the revenue consistently improves as the arrival rate increases. However, this improvement in revenue suffers the penalty of high rejection rate, resulting in dissatisfaction of the service provided. On the other hand, encouraging clients to plan ahead could be a positive approach to boosting the revenue. For example, in Fig. 10, the revenue generated by the default plan-ahead time pattern is significantly better than the 433 plan-ahead time pattern, though more clients in the 433 pattern are willing to pay a higher price to be qualified to an on-demand service. In general, maximizing the revenue integrated with provision of good quality service can be a challenge for further research. A strategy of selective rejection may provide a potential solution.

VIII. Discussion and Concluding Remarks

The work presented in this paper aims at a framework of scheduled video delivery. The preliminary results show that SVD is a promising approach for a truly scalable video delivery. SVD is suitable for the digital cable television network which is based on a broadcast scheme and has a fixed number of channels. SVD can also be applied to the Internet with a multicasting capability. Recently, the server-based multicast on the overlay networks becomes practical [34], [35], to be used as the multicast SVD service.

Another issue on the Internet is pricing. Internet service is traditionally free and people might not be willing to pay for the video service either. On the other hand, it is also a tradition that people pay for movie rental. As a substitution of movie rental, people would like to pay for the SVD service. The saving of on-line delivery should reduce the cost for the provider and the consumer should also pay less than that in a video rental store. In addition to convenience, the saving will turn consumers from the video rental store to the SVD service. Incentive and pricing is an important component of SVD. The user's choice of plan-ahead time can be influenced by the price. In fact, not only the user's choice of plan-ahead time, but also the choice of videos can be influenced by the price, which will be studied in future works.

As a low-cost video delivery system, SVD has a number of advantages over the existing schemes. First, requests that are submitted ahead of time can provide a great opportunity for efficient scheduling and combining, as a result, reducing the overall resource consumption significantly. Second, peak-time requests can be executed ahead of time to utilize nonpeak time server access and network bandwidth. It effectively increases the amount of usable system resource. Compared to VoD, SVD provides not only instant video access, but also allows users to plan-ahead to reduce the cost and price of video delivery. With SVD, on-demand video can be provided with much lower cost. Compared to the batching and patching approaches, SVD is able to combine more requests, reducing the system resource requirement further. Compared to Near VoD, SVD does not send the same video repeatedly. The same number of channels can provide a large selection of video titles.

The SVD scheduling is different from any existing scheduling scheme. It does not aim at minimizing the waiting time like in most video scheduling algorithms, instead, it focuses on meeting deadlines and combining requests to form multicasting groups. SVD defines a new class of scheduling algorithms. SVD is scalable to both the number of requests and the number of videos. The analysis of scalability of the SVD scheme is detailed in [2], [36]. SVD requires much less resources to provide an on-demand video service compared to the true VoD. Integrating SVD with patching, and considering peak time behaviors will further improve scalability of video delivery.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their thorough comments.

REFERENCES

[1] J. E. G. Coffman, P. Jelenkovic, and P. Momcilovic, "Provably efficient stream merging," in *Sixth Int. Workshop on Web Caching and Content Distribution (WCW'01)*, Mar. 2001.

[2] W. Shu and M. Wu, "Resource requirements of closed-loop video delivery services," *IEEE Multimedia*, pp. 24–37, Apr.–Jun. 2004.

[3] M. Wu, S. Ma, and W. Shu, "Scheduled video delivery for scalable on-demand service," in *Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[4] S. Chan and S. Yeung, "Broadcasting video with the knowledge of user delay preference," *IEEE Trans. Broadcasting*, vol. 49, no. 2, pp. 150–161, Jun. 2003.

[5] J. Y. B. Lee, "On a unified architecture for video-on-demand services," *IEEE Trans. Multimedia*, vol. 4, no. 1, pp. 38–47, Mar. 2002.

[6] M. Wu and W. Shu, "Optimal scheduling for parallel CBR video servers," *Multimedia Tools Applicat.*, vol. 14, no. 1, May 2001.

[7] ——, "Efficient support for interactive browsing operations in clustered CBR video servers," *IEEE Trans. Multimedia*, vol. 4, no. 1, Mar. 2002.

[8] H. C. D. Bey, Program Transmission Optimization, Mar. 1995.

[9] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *Int. Conf. Multimedia Computing and Systems*, 1996, pp. 118–126.

[10] K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," in *SIGCOMM*, 1997, pp. 89–100.

[11] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *IEEE Int. Conf. Multimedia Computing and Systems*, Jun. 1996.

[12] A. Dan, Y. Heights, and D. Sitaram, "Generalized interval caching policy for mixed interactive and long video workloads," in *Proc SPIE Conf. on Multimedia Computing and Networking*, Jan. 1996, pp. 344–351.

[13] L. Gao and D. F. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in *ICMCS*, vol. 2, 1999, pp. 117–121.

[14] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," *ACM Multimedia*, pp. 15–23, 1994.

[15] ——, "Dynamic batching policies for an on-demand video server," *ACM Multimedia Syst.*, no. 4, pp. 112–121, 1996.

[16] S. Sheu, K. A. Hua, and T. H. Hu, "Virtual batching: a new scheduling technique for video-on-demand servers," in *The 5th DASFAA*, Apr. 1997.

[17] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1110–1122, 1996.

[18] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," in *Proc. 9th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99)*, Basking Ridge, NJ, Jun. 1999.

[19] L. Gao, Z.-L. Zhang, and D. Towsley, "Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams," in *7th ACM Int. Multimedia Conf. (ACM Multimedia '99)*, 1999, pp. 203–206.

[20] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *ICCCN 97*, Las Vegas, NV, 1997, pp. 200–207.

[21] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *7th ACM Int. Multimedia Conf. (ACM Multimedia '99)*, Nov. 1999, pp. 199–202.

[22] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *SIGCOMM*, 1998, pp. 56–67.

[23] S. Chan and F. Tobagi, "Caching schemes for distributed video services," in *Proc. 1999 IEEE Int. Conf. Communications (ICC'99)*, Vancouver, BC, Canada, Jun. 1999.

[24] S. D. Stoller and J. DeTreville, "Storage replication and layout in video-on-demand servers," *Network and Operating System Support for Digital Audio and Video*, pp. 330–341, 1995.

[25] J. Akamai, "Internet Bottlenecks: The Case for Edge Delivery Services,", 1999.

[26] K. Almeroth and M. Ammar, "The interactive multimedia jukebox (IMJ): a new paradigm for the on-demand delivery of audio/video," in *The Seventh Int. World Wide Web Conf.*, Apr. 1998.

[27] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in computer networks: motivation, formulation, and example," *IEEE/ACM Trans. Networking*, vol. 1, pp. 614–627, 1993.

[28] J. K. MacKie-Mason and H. R. Varian, "Pricing congestible network resources," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1141–1149, Sep. 1995.

[29] X. Wang and H. Schulzrinne, "Performance study of congestion price based adaptive service," in *Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Jun. 2000, pp. 1–10.

[30] ——, "An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications," *IEEE J. Select. Areas Commun.*, vol. 18, no. 2002, pp. 2514–2529, Dec..

[31] G. Zipf, *Human Behavior and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.

[32] K. Almeroth, "Adaptive, workload-dependent scheduling for large-scale content delivery systems," *IEEE Trans. Circuits Systems for Video Tech.*, no. 3, pp. 426–439, Mar. 2001.

[33] T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–23, Fall 1994.

[34] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," *ACM Sigmetrics*, 2000.

[35] M. Wu, Y. Zhu, and W. Shu, "Placement of proxy-based multicast overlays," *Computer Networks*, 2005.

[36] W. Shu and M. Wu, "Capacity analysis of scheduled video delivery service," in *Int. Conf. Computers and Their Applications (CATA-2004)*, Mar. 2004.

Min-You Wu (M'86–SM'96) is an IBM Chair Professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He serves as the Chief Scientist at Grid Center of Shanghai Jiao Tong University.

His research interests include grid computing, wireless networks, sensor networks, overlay networks, multimedia networking, parallel and distributed systems, and compilers for parallel computers. He has published over 110 journal and conference papers in the above areas.

Dr. Wu is a member of ACM.

Sujun Ma was born in Hebei Province, China, in 1967. He received the B.S. degree in electrical engineering from Nanjing University of Science & Technology, Nanjing, China, in 1989, the M.S. degree in electrical engineering from Beijing Institute of Technology, Beijing, China, in 1999, and the Ph.D. degree in computer engineering from the University of New Mexico, Albuquerque, in 2003.

From 1989 to 1999, he was a System and Hardware Engineer at China Electronics Technology Group Corporation (CETGC) no. 54 Research Institute, Shijiazhuang, China. His research interests include networked multimedia and embedded system.

Wei Shu (M'90–SM'99) received the Ph.D. degree from the University of Illinois at Urbana-Champaign.

Since then, she worked at Yale University, the State University of New York at Buffalo, and University of Central Florida. She is currently an Associate Professor in the Department of Electrical and Computer Engineering, the University of New Mexico, Albuquerque. Her current interests include dynamic scheduling, resource management, multimedia networking, distributed systems, and sensor networks.

Dr. Shu is a member of the ACM.