Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 2231–2241
computer communications
www.elsevier.com/locate/comcom

A hybrid priority-based video-on-demand resource sharing scheme

Chenn-Jung Huang [a,*], Yi-Ta Chuang [b], Chih-Tai Guan [a], Yun-Cheng Luo [a], Kai-Wen Hu [a], Chun-Hua Chen [a]

[a] Department of Computer & Information Science, College of Science, National Hualien University of Education, 123 Huashi Road, Hualien, Taiwan 970, Taiwan
[b] Institute of Computer Science & Information Engineering, National Chiao Tung University, Taiwan

Received 19 March 2007; received in revised form 11 February 2008; accepted 11 February 2008
Available online 19 February 2008

Abstract

Video-on-demand (VoD) environments frequently batch video requests to decrease I/O demand and increase throughput. Since users may leave due to waiting too long, a good video scheduling policy has to consider not only the batch size, but also the user defection probabilities and waiting times. Moreover, a practical VoD resource sharing scheme should endeavor to provide some free streams to serve a high-priority clients requests immediately, since the high-priority clients might pay for the requested video. To tackle these problems, this work proposes a hybrid resource sharing model that integrates controlled multicasting and batching. The proposed hybrid model applies a bandwidth borrowing and reserving scheme to give high-priority clients a prompt service, while still providing low-priority clients with a reasonable service. Furthermore, a novel probability model-based scheduling policy is proposed to alleviate the user defection behavior and unfairness issue. Experimental results demonstrate that the proposed resource sharing scheme is effective and feasible in terms of blocking probability of high-priority clients, the defection probability, service delay time and fairness to low-priority users.

Keywords: Video-on-demand; Batching; Bandwidth borrowing and reserving; Scheduling; Controlled multicasting; Quality of service

1. Introduction

Recent advances in communication and computer technology have increased the speed of transmission across the Internet. Next-generation networks support transmission rates that are higher than current rates. Therefore, the explosive increase in commercial usage of the Internet has led to a rapid growth in demand for video deliver technologies. In such a system implemented with client–server architectures, viewers have the flexibility of specifying both the video they want and the time when they wish to watch the video.

A video-on-demand (VoD) system comprises a video server with a video archive and several client machines linked by a local area network. Users request their desired videos using client software. The server delivers the requested video to the user in an isochronous data stream in response to a service request. Bar-Noy [1] identified two major bottlenecks in current VoD system architecture:

- The limited number of broadcasting channels available on the access network.
- The number of movies that a server can send concurrently.

Many schemes, including resource sharing and scheduling [2], admission control [3] and switched digital video (SDV) architecture [4], have been developed to enhance the efficiency of the VoD system. This work concentrates on resource sharing and scheduling schemes as described below.

Resource sharing can significantly enhance the performance of VoD servers. Resource sharing strategies include

* Corresponding author. Tel.: +88638226738.
E-mail address: cjhuang@mail.nhlue.edu.tw (C.-J. Huang).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.02.007

batching [5], patching [6], piggy-backing [7] and broadcasting [8]. Batching [5] is the most common resource sharing scheme for VoD systems, since it groups users waiting for the same video data and serves them by a multicast channel. This batching process can occur passively while the users are waiting, or actively by delaying the service of early-arriving users to wait for late-arriving users to join the batch. In batching, requests to the same movies are accumulated and served simultaneously. A video server serves a batch of requests for the same video, all of which arrive within a short period using one server channel by using the multicast facility. Therefore, batching utilizes server bandwidth and network resources efficiently.

Patching [6] dynamically expands the multicast tree to include new requests, and thus decreases the request waiting time, but increases the bandwidth and buffer spaces required at the client's site. Patching exploits buffers videos using the client's disk space, decreasing the requested waiting time. Therefore, patching strongly depends on the ability of the client side's disk space to cache the video data.

Piggy-backing [7] request almost immediately by varying the play-back rate on the client side so that the request can catch up with a preceding stream, thus lowering the quality of the initial presentation. To preserve the display quality of the video, the adjustment of the delivery rate must be kept below 5%, which restricts the number of channels that piggy-backing can adaptively merge to save resources.

Broadcasting [8] splits a video into several segments, and periodically broadcasts them in order on a dedicated channel. Therefore, broadcasting requires a high bandwidth and buffer spaces at client sites. The server bandwidth requirement is independent of the number of users that the system is designed to support. Nevertheless, this policy is only feasible for very popular videos when utilizing periodic broadcasting.

Switched digital video (SDV) [4] technology fundamentally changes the way digital video is delivered over cable networks, enabling cable operators to offer a wider variety of programming while effectively managing valuable hybrid fiber coaxial (HFC) network bandwidth. SDV technology decouples bandwidth from content by employing several SDV servers, transmitting content to subscribers only in areas where programs are being requested in real-time. Although the SDV architecture employs several SDV servers to enhance the scalability, it only allocates a limited number of channels allocated for each SDV server. Moreover, the problems with the conventional broadcasting scheme are left unresolved, because the SDV architecture also delivers videos by broadcasting.

Controlled multicasting [10] is a reactive instantaneous VoD system, which assumes the bandwidth provided by the server is unlimited, and that the clients' buffer size is not a constraint. The benefit of controlled multicasting is that earlier requests can be served instantaneously without waiting for later requests, as is necessary in batching. However, controlled multicasting involves many channels, because it can serve requested videos immediately. Kong et al. [20] presented a new patching mechanism called turbo-slice-and-patch (TSP), based on dynamic channel assignment, to serve video stream services in a metropolitan-scale network. They employed network multicast to increase significantly the system's scalability to cope with the immense workload in a metropolitan scale streaming service. TSP currently seems to be the best patching scheme for providing high-quality video and economy of scale in metropolitan-scale video streaming services.

The performance of these resource-sharing strategies can be further enhanced if VoD servers can schedule the waiting requests in an appropriate order. The scheduling policies serve different batches first when a server channel becomes available. First-come-first-served (FCFS) serves the batch holding the oldest request with the longest waiting time as soon as bandwidth becomes available. Maximum-queue-length-first (MQL) [11] serves the batch with the largest number of pending requests (i.e., the longest queue). FCFS is fair, because it treats each user equally, irrespective of the popularity of the requested video. However, it has a low-system throughput, since it may elect to serve a batch with fewer requests first while causes another batch with more requests to wait. To address this issue, MQL maintains a separate waiting queue for each video, and delivers the video with the longest queue first. This policy maximizes server throughput, but is unfair to users requesting less popular videos. Maximum-factored-queued-length first (MFQL) [16] attempts to provide reasonable fairness and high-server throughput. When a server channel becomes free, MFQL delivers the video v_i with the longest queue weighted by a factor $1/\sqrt{f_i}$, However, experimental results demonstrate that MFQL is not fair in most situations, because it bases its choice only on the queue length of the video. As is well known, popular movies typically have longer queues than other videos. Moreover, the effect of the factor f_i is much smaller than the queue length.

Each of the above-mentioned resource sharing strategies has different benefits and flaws, and might therefore complement each other. Therefore, this work presents an adaptive hybrid resource sharing scheme that combines the functionality of both controlled multicasting and batching to tackle the above bottlenecks and gain the benefits of both schemes. This work also utilizes a bandwidth borrowing technique to manage the temporary bandwidth crisis in the original controlled multicasting scheme. An effective fair scheduling policy is also presented to enhance the performance of the proposed VoD resource sharing model.

The remainder of this paper is organized as follows. Section 2 presents the hybrid resource sharing scheme that embeds channel borrowing, and channel reserving mechanisms and a probability model-based scheduling policy. The simulation result is given in Section 3. Conclusion is made in Section 4.

2. The hybrid video-on-demand resource sharing scheme

This work proposes a hybrid resource sharing scheme that incorporates a probability model-based scheduling mechanism and adaptive channel reserving and borrowing mechanism. Fig. 1 illustrates the proposed scheme, which utilizes controlled multicasting and batching scheme for high- and low-priority clients, respectively, to alleviate the shortcomings of contemporary video architectures. For low-priority batching, a novel probability model-based scheduling policy based on the clients' waiting time is applied to ensure fairness among the requests. Conversely, controlled multicasting scheme is combined with channel reservation and borrowing, so that the available channels can be utilized fully by dynamically varying the channel allocation based on the priority of the request.

The incoming client's request is initially categorized as a high- or low-priority task. High-priority requests access the video data using the proposed channel borrowing and reserving scheme if necessary, while the low-priority requests access the video data through the proposed probability-based scheduler. When a client requests a video, the video server first specifies the priority of the request. If the request is low-priority, and the controlled multicasting channels are not overloaded, then the server services the request immediately. The server places the low-priority requests into the scheduler when the controlled multicasting channels are overloaded.

If the request belongs to the high-priority task group, and sufficient free controlled multicasting channels are available, then the server allocates a channel and serves the request immediately. A channel can also be allocated from any free reserved channels if no controlled multicasting channels are available.

2.1. Integration of controlled multicasting and channel borrowing

A controlled multicasting [10] scheme assumes that infinite server channels are available, and enables two clients that request the same video at different times to share a channel. However, the late-arriving client is permitted to use another free channel to download the portion of the video segment that was received by the early-arriving client. Unlike other batching schemes, controlled multicasting does not delay an earlier request that is still sharing a channel when it received a higher-priority request.

Clients are divided into two different priority groups according to their certification guarantees. The clients in different classes have different QoS guarantees, where only the high-priority clients are assumed to have to pay for the requested videos. Therefore, the controlled multicasting scheme is dedicated to clients in the high-priority group, while the batching scheme serves low-priority clients who access videos for free. Each video is divided into several parts according to the broadcasting scheme. That is, each channel is divided into fixed time slots for transmitting a specific part of a video. The videos in the two priority

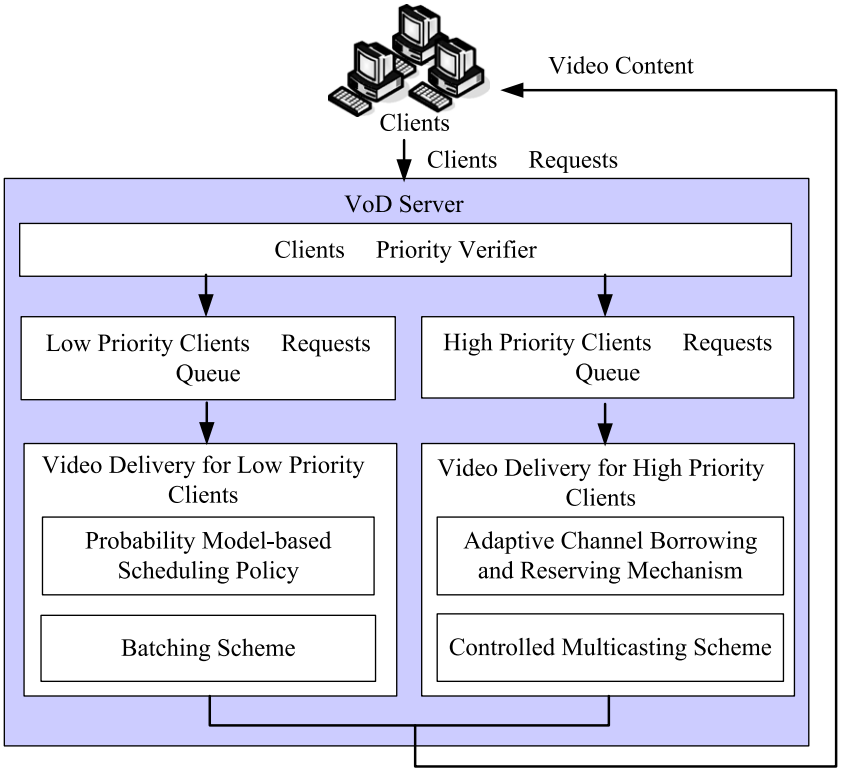

Fig. 1. The architecture of the hybrid priority-based VoD resource sharing scheme.

groups are broadcast in turn. The controlled multicasting and batching scheme are integrated by altering the mode in each channel after fix time slot.

High-priority clients are preferred for accessing channels in the controlled multicasting mode, while low-priority class clients are preferred for accessing channels in batching mode. The low-priority clients can access channels in the controlled multicasting mode if the current network loading is not heavy. Channel borrowing is utilized to enable high-priority clients to share a channel that is currently serving low-class clients by eliminating the topmost layer of the ongoing streams that serve low-class clients in the controlled multicasting mode. The shared channel is employed to catch up the missing video segment received by the early-arriving high-priority clients.. The broadcasting scheme broadcasts each video in turn. This approach is appropriate only for popular videos, and consumes more server resources than the proposed scheme, in which the channel access mode can be changed. In contrast, the proposed scheme is thus more flexible than broadcasting, and can utilize both controlled multicasting and batching modes.

2.2. Adaptive channel reserving mechanism

Channel borrowing [9] has been presented to handle temporary bandwidth crises in controlled multicasting by borrowing bandwidth from other ongoing streams. A scalable layer video coding technique is employed to code video streams into multiple layers, comprising the base layer and several enhanced layers, as defined in the MPEG-2 coding standard. A layer corresponds to a particular QoS. Some of the topmost ongoing video streams are eliminated during a temporary bandwidth crisis to accommodate new streams. These missing layers are restored when bandwidth becomes available after a regular or patch stream is dropped.

This work proposes an adaptive channel reserving mechanism to reserve free channels for the incoming high-class clients to increase their priority and reduce their blocking probability. Since the proposed hybrid resource sharing scheme reserves an appropriate number of free channels for the expected incoming high-priority clients during the next time period, determines the number of reserved channels for the high-class clients during the next time period from the current network traffic load using the following equation:

$$\mathrm{Rr}(t+1) = K \cdot \mathrm{Str} \cdot \frac{b(t) \cdot \hat{n}_{\mathrm{H}}(t+1)}{\hat{n}_{\mathrm{L}}(t+1)}, \tag{1}$$

where Str denotes the total number of the server streams; b is the blocking probability for class 1 video during the current time period; $\hat{n}_{\mathrm{H}}(\cdot)$ and $\hat{n}_{\mathrm{L}}(\cdot)$ represent the predicted numbers of high- and low-priority clients during the next time period, respectively, and K is a constant less than 1. Notably, the values of $\hat{n}_{\mathrm{H}}(\cdot)$ and $\hat{n}_{\mathrm{L}}(\cdot)$ are predicted by the weighted moving average method.

2.2.1. Weighted moving average method

Since $\hat{n}_{\mathrm{H}}(\cdot)$ and $\hat{n}_{\mathrm{L}}(\cdot)$ can both be considered as time series, this work predicts the value of the next time interval using a well-known time series predictor, the weighted moving average method [19], which has been reported to perform well on time series prediction [19]. A time series is a sequence of numerical values indexed by increasing time unit. The conventionally adopted time series prediction techniques in the literature include the 'average' method, which determines the mean of the past m measurement periods:

$$\bar{\lambda} = \frac{\sum_{j=0}^{m-1} \lambda(t-j)}{m}, \tag{2}$$

and the weighted moving average, which increases the weight of the last measurement period:

$$\hat{\lambda}(t+1) = (1-\rho) \cdot \bar{\lambda} + \rho \cdot \lambda(t), \tag{3}$$

where $\rho = 1$ and $\bar{\lambda}$ represents the average calculated in Eq. (2).

2.3. The probability model-based scheduling policy

Since the batching scheme leads to different videos in the same priority class waiting for service, another selecting mechanism is required to choose the next video to serve in the same priority group. Wu et al. [22] assumed that the user can wait for an infinite time. This assumption is impractical, because the client cannot reasonably wait for an unknown long time after having paid to watch a video on the Internet. This study proposes a practical probability model-based scheduler, called the largest effective waiting time first scheduling policy, to minimize user defection and unfairness.

If the probability that a client leaves the system at t seconds is $p(t)$, then the probability for the client leaving the system within t seconds is given by

$$D(t) = \int^{t} p(x)\,\mathrm{d}x. \tag{4}$$

Assume that a client requests a video and does not cancel it within T seconds, and that the conditional probability for this client leaving the system after T seconds is given by

$$q(t) = \frac{1}{c(T)} p(t), \tag{5}$$

where $c(T)=1 - D(T)$ denotes the probability that the client will not cancel the request after T seconds.

The selection criteria for each video queue can then be expressed as:

$$v_i = \sum (t_j \cdot q(t_j)), \tag{6}$$

where t_j denotes the current cumulative waiting time for client j.

This work assumes the user defection behavior is normally distributed as in [16]. However, Eq. (6) must still

be solved quickly to make this approach feasible in practice, particularly when the calculation involves an integration, as in Eq. (4). This study proposes a simplified computation strategy that adopts an approximate cubic polynomial to replace the integration, as presented in Eq. (4) below.

The probability function of a variable x, normally distributed with mean μ and standard deviation σ, can be expressed as:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}. \tag{7}$$

The cumulative distribution function is

$$D(x) = \int_{-\infty}^{x} p(x')\,dx'. \tag{8}$$

As mentioned earlier, the cumulative distribution function is in an integral form and is computationally expensive. Hence, this study proposes a cubic polynomial, $g(x)=a+bx+cx^2+ex^3$ to approximate the original cumulative distribution function, as presented in Eq. (8).

The derivation of the coefficients in the cubic polynomial, $g(x)=a+bx+cx^2+ex^3$, can be completed by the following error function:

$$E(x) = \int_{-\infty}^{\infty} (a+bx+cx^2+ex^3 - D(x))^2 p(x)\,dx. \tag{9}$$

Let

$$I_n = \int_{-\infty}^{\infty} x^n p(x)\,dx = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x^n e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{10}$$

and

$$J_n = \int_{-\infty}^{\infty} x^n p(x) D(x)\,dx = \frac{1}{2\pi\sigma^2} \int_{-\infty}^{\infty} x^n e^{-\frac{(x-\mu)^2}{2\sigma^2}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\,dt\,dx \tag{11}$$

The computation of Eq. (9) can then be further transformed into a linear matrix:

$$\begin{bmatrix} a \\ b \\ c \\ e \end{bmatrix} = \left[\begin{array}{cccc|c} I_0 & I_1 & I_2 & I_3 & J_0 \\ I_1 & I_2 & I_3 & I_4 & J_1 \\ I_2 & I_3 & I_4 & I_5 & J_2 \\ I_3 & I_4 & I_5 & I_6 & J_3 \end{array}\right], \tag{12}$$

where the values of I_0, I_1, I_2, I_3, I_4, I_5 and I_6 can be derived by Eq. (10), and J_0, J_1, J_2 and J_3 can be obtained by Eq. (11). Since the coefficients of the cubic polynomial are tedious to derive from Eq. (12), we set $z = \frac{x-\mu}{\sigma}$ and rewrite Eqs. (7) and (8) as follows to simplify the calculation:

$$p(z) = \frac{1}{\sqrt{2\pi}} e^{z^2/2}, \tag{13}$$

$$D(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{z'^2/2}\,dz'. \tag{14}$$

The cubic polynomial that approximates the cumulative distribution function thence becomes

$$g(z) = a + bz + cz^2 + ez^3. \tag{15}$$

After substituting for z in Eq. (15) using $\frac{x-\mu}{\sigma}$, we obtain

$$g(x) = a + b\frac{x-\mu}{\sigma} + c\left(\frac{x-\mu}{\sigma}\right)^2 + e\left(\frac{x-\mu}{\sigma}\right)^3, \tag{16}$$

and the error function becomes

$$E = \int_{-\infty}^{\infty} (a+bz+cz^2+ez^3 - D(z))^2 p(z)\,dz, \tag{17}$$

Since $D(z)$ is symmetric at $(0,\frac{1}{2})$, it can be shown that

$$a = \frac{1}{2}, \tag{18}$$

and

$$c = 0. \tag{19}$$

Meanwhile, the error function as given in Eq. (9) is expressed as:

$$E = \int_{-\infty}^{\infty} \left(\frac{1}{2} + bz + ez^3 - D(z)\right)^2 p(z)\,dz. \tag{20}$$

Notably, the difference between the cubic polynomial and the cumulative distribution function becomes zero when appropriate values are chosen for b and e in Eq. (20). Restated

$$\frac{\partial E}{\partial b} = 0 = \frac{\partial E}{\partial e}. \tag{21}$$

Eq. (23) indicates that:

$$\int_{-\infty}^{\infty} \left(\frac{1}{2} + bz + ez^3 - D(z)\right) z p(z)\,dz = 0, \tag{22}$$

and

$$\int_{-\infty}^{\infty} \left(\frac{1}{2} + bz + ez^3 - D(z)\right) z^3 p(z)\,dz = 0. \tag{23}$$

Eq. (12) can now be recast as,

$$\begin{bmatrix} b \\ e \end{bmatrix} = \left[\begin{array}{cc|c} I_2 & I_4 & J_1 - \frac{1}{2}I_1 \\ I_4 & I_6 & J_3 - \frac{1}{2}I_3 \end{array}\right], \tag{24}$$

where

$$I_n = \int_{-\infty}^{\infty} z^n p(z)\,dz, \tag{25}$$

and

$$J_n = \int_{-\infty}^{\infty} z^n p(z) D(z)\,dz. \tag{26}$$

The values of I_n and J_n are now determined. Since

$$(-z)^n p(-z) = -z^n p(z), \tag{27}$$

when n is odd, thus

$$I_{2n-1} = 0, \ \forall\ n \in N. \tag{28}$$

When n is even, I_n becomes a recurrence relation as follows:

$$I_{n+2} = (n+1) I_n. \tag{29}$$

Thus, we obtain

$$I_2 = 1; \quad I_4 = 3; \quad I_6 = 15. \tag{30}$$

Next we let

$$K_n = \int_{-\infty}^{\infty} z^n p^2(z)\,\mathrm{d}z. \tag{31}$$

It can be shown that

$$J_{n+1} = nJ_{n-1} + K_n, \tag{32}$$

and

$$J_1 = \frac{1}{2\sqrt{\pi}}; \quad J_3 = \frac{5}{4\sqrt{\pi}}. \tag{33}$$

Accordingly, Eq. (24) can be expressed as:

$$\begin{bmatrix} b \\ e \end{bmatrix} = \left[\begin{array}{cc|c} 1 & 3 & \frac{1}{2\sqrt{\pi}} \\ 3 & 15 & \frac{1}{4\sqrt{\pi}} \end{array}\right] = \frac{1}{24\sqrt{\pi}} \begin{bmatrix} 15 \\ -1 \end{bmatrix}, \tag{34}$$

and the cubic polynomial as given by Eq. (16) becomes

$$g(x) = \frac{1}{2} + \frac{5}{8\sqrt{\pi}}\left(\frac{x-\mu}{\sigma}\right) - \frac{1}{24\sqrt{\pi}}\left(\frac{x-\mu}{\sigma}\right)^3. \tag{35}$$

Accordingly, the tedious computation of Eq. (4) can be replaced by the simplified cubic polynomial form, as given in Eq. (35).

The proposed VoD server selects the video to serve based on its v_i value as given in Eq. (6). The video with the largest v_i value is served first.

2.4. Time complexity analysis

2.4.1. Time complexity of the probability model-based scheduling policy

Based on Eqs. (4)–(7), and 35, the selection criteria for each video queue, as given by Eq. (6), can be recast as

$$v_i = \sum_{j=1}^{n} \frac{t_j \times \mathrm{e}^{-\frac{(t_j-\mu)^2}{2\sigma^2}}}{\frac{1}{2} - \frac{5}{8\sigma\sqrt{\pi}}(t_j-\mu) + \frac{1}{24\sigma\sqrt{\pi}}(t_j-\mu)^3}. \tag{36}$$

The total counts of instructions required for the probability model-based scheduling policy adopted in the low-priority batching scheme can be obtained by solving Eq. (36). The required instructions consist of $3n$ additions, $\left(9 + \frac{\mu^2}{2\sigma^2}\right)n$ multiplications, $5n$ divisions and n square root operations. Here, n indicates the number of incoming clients, and μ and σ represent the predefined average client waiting time and the standard deviation of the client waiting time, respectively.

2.4.2. Time complexity of the adaptive channel reserving mechanism

Based on Eqs. (1)–(3), the number of reserved channels for the high-class clients during the next time period, as given by Eq. (1), can be written as

$$\mathrm{Rr}(t+1) = K \cdot \mathrm{Str} \cdot b(t)\frac{m \cdot \lambda(t+1) - \sum_{j=0}^{m-1}\lambda(t-j)}{m \cdot \lambda(t) - \sum_{j=0}^{m-1}\lambda(t-j)}. \tag{37}$$

Eq. (37) demonstrates that the total counts of instructions required for the adaptive channel reserving mechanism employed in the controlled multicasting method are $3 + 2m$ additions, $6 + 2m$ multiplications and 1 division. Here, m denotes the number of the past cumulative measurement periods.

We thus conclude that the additional computational load on the server required for the probability model-based scheduling policy and the adaptive channel reserving mechanism, as shown in Fig. 1, is the greater of O(n) or O(m). The added computation overhead compared with human response is insignificant.

3. Simulation

3.1. Simulation scenarios

This study considered a scenario based on the operation in YouTube, in which the video server contains the general videos and updates videos within a fixed period of time. Yu et al. [12] and Veloso et al. [13] analyzed a wide variety of media workloads on the Internet. They accumulated the workloads from both the client and the server sides in the Web, VoD, P2P, and live streaming environments between 1998 and 2006. The media content was delivered via Web/P2P downloading or unicast/multicast streaming. Both Veloso et al. [13] and Sripanidkulchai et al. [14] reported that live streaming media systems follow stretched exponential (SE) distribution. The SE distribution lays out an analytical foundation to establish scheduling methods for delivering the rapidly increasing quantity of Internet media content. Therefore, this study assumed that the VoD system frequently updates contents in our simulation scenarios, matching the characteristics of a SE distribution. Similar to the scenarios in [16–18], the proposed VoD system was assumed to contain 100 videos, each 120 min long. The server capacity was in the range 50–500 [16,17], and the client arrival rate was exponentially distributed in the range 20–60 per minute [16,19,20]. The compared schemes are the primitive hybrid scheme that integrates controlled multicasting and batching (AHVoD1), the hybrid scheme embedded with channel borrowing mechanism (AHVoD2), the proposed hybrid system embedded with channel borrowing and reservation mechanisms (AHVoD3), controlled multicast (CM) [10], and turbo-slice-and-patch (TSP) [20]. Notably, TSP is chosen here because it is currently the state-of-the art patching scheme for providing high-quality video and economy of scale in metropolitan-scale video streaming services.

3.2. Performance metrics

The resource sharing and scheduling policies were analyzed by the performance metrics adopted in [16], as described below.

- Blocking probability: the probability that an arriving high-priority client leaves the system without being serviced due to the lack of a server stream.
- Defection probability: the probability that an arriving low-priority client leaves the system without being served because the waiting time exceeds that acceptable to the viewer. The defection probabilities obviously vary according to the video. Let r_i represent the defection probability for video i. The mean defection probability can thus be expressed as:

$$\bar{r} = \sum_{i=1}^{N} r_i / N. \tag{38}$$

  Meanwhile, the user defection behavior is modeled with normal distribution with a mean value of 5 min and a standard deviation of one.
- Average latency time: the latency of a client is the period that elapses between the arrival of the video request and the time when the service to the display device is initiated. The latency time measure considers only non-defecting clients.
- Unfairness: the unfairness is defined as follows. [16]

$$\text{unfairness} = \sqrt{\sum_{i=1}^{n} (r_i - \bar{r})^2 / (n-1)}. \tag{39}$$

In many batching polices, the clients wait longer for the less popular, cold videos than for the more popular, hot videos. Higher unfairness values correspond to less fair policies. If all videos are treated equally, then the unfairness as given by Eq. (39) should be small.

3.3. Simulation result

Experiments were performed with arrival rates in the range 20–60 requests per minute, and the server capacity fixed at 200 streams. The server capacity was then varied as 100–500 streams with the arrival rate fixed at 50 requests per minute.

Figs. 2 and 3 compare the blocking probability of high-priority clients in the three proposed methods and the controlled multicasting model. The proposed hybrid system embedded with channel borrowing and reservation mechanisms (AHVoD3) performed better than the hybrid scheme embedded with channel borrowing mechanism (AHVoD2), the primitive hybrid scheme (AHVoD1), controlled multicast (CM) and the above-mentioned state-of-the art patching mechanism (TSP). As revealed in Figs. 2 and 3, TSP and CM performed poorest because they always allocate a channel to a client who requests a video, regardless whether the client has high or low priority. Assigning a channel to each low-priority client obviously significantly reduces of number of available channels for incoming high-priority clients.

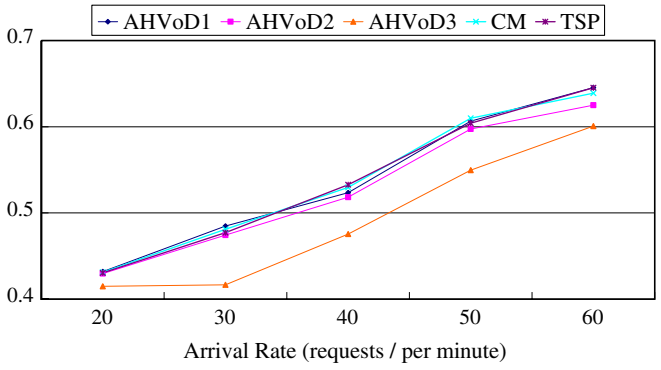

Fig. 2. Blocking probability of high-priority clients for the three proposed schemes and controlled multicasting scheme at varied arrival rates.

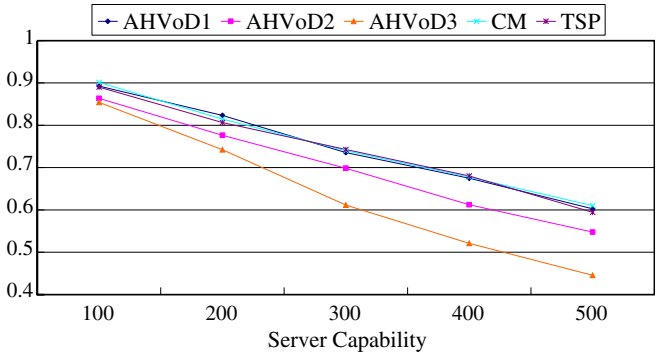

Fig. 3. Blocking probability of high-priority clients for the three proposed schemes and controlled multicasting scheme under varied server capacity.

In contrast, the three proposed schemes ensure that the controlled multicasting channels are allocated to the high-priority clients first, and permit access by the low-priority clients only when they are not overloaded. Furthermore, the hybrid embedded with channel borrowing mechanism alone also significantly outperformed the primitive hybrid scheme. Thus, the channel borrowing and adaptive channel reservation mechanism adopted in this work indeed boosted the performance of the proposed resource sharing system. Fig. 3 shows that the channel borrowing and reserving mechanisms rectify the deterioration problem on blocking probability of high-priority clients when the server capacity is equal to or larger than 300, because they have more channels that can be reserved or borrowed by high-priority clients.

Figs. 4 and 5 show the defection probabilities of low-priority clients for the three proposed schemes and the representative scheduling schemes MFQL and FIFO. The primitive hybrid scheme (AHVoD1) performed worse than MFQL, because the prompt service of high-priority clients in controlled multicast operating mode reduces the defection probability of low-priority clients operating in batch mode. The hybrid scheme embedded with channel borrowing mechanism (AHVoD2) performed slightly better than the hybrid system embedded with channel borrowing and reservation mechanisms (AHVoD3), because AHVoD3 reserves some channels for high-priority clients, moderately reducing the defection probability for low-priority clients.

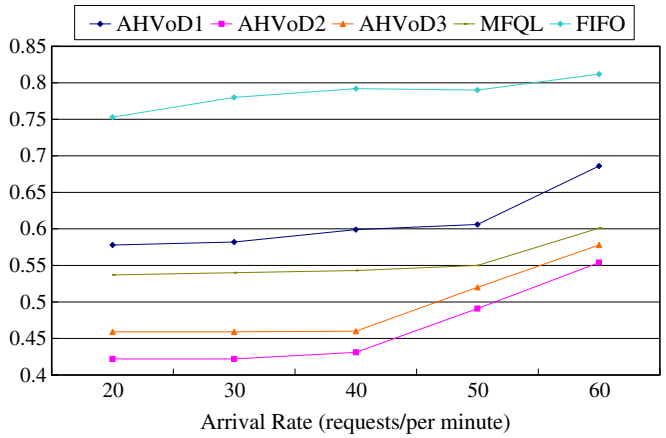

Fig. 4. Defection probability of low-priority clients at varied arrival rates.

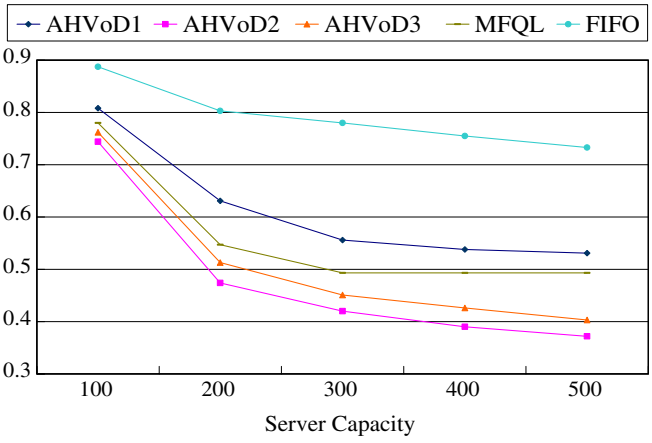

Fig. 5. Defection probability of low-priority clients under varied server capacity.

Figs. 6 and 7 illustrate the service delay times of low-priority clients for the three proposed schemes, MFQL and FIFO. These figures indicate that the three proposed schemes had shorter service delay times than MFQL and FIFO, since they allow low-priority clients to access the video immediately when the server is not overloaded.

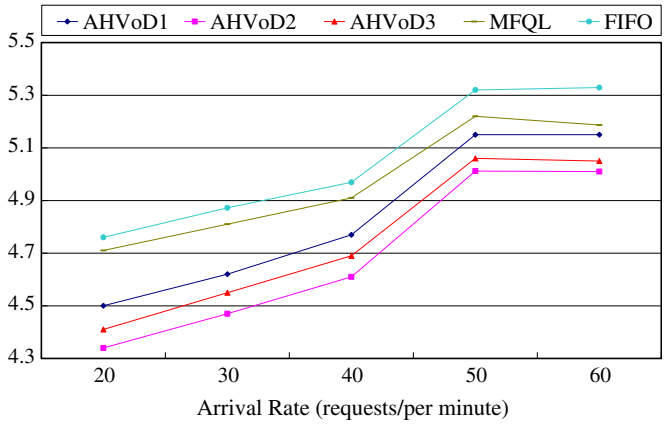

Fig. 6. Service delay time of low-priority clients at varied arrival rates.

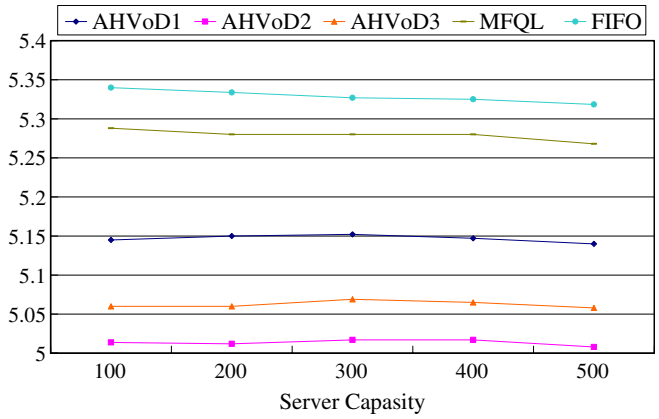

Fig. 7. Service delay time of low-priority clients under varied server capacity.

Notably, the hybrid system embedded with channel borrowing and reservation mechanisms (AHVoD3) performed better than the hybrid scheme embedded with channel borrowing mechanism (AHVoD2), because its channel borrowing and reservation mechanisms allocate free channels more effectively, and assigns more channels operated in controlled multicasting mode assigned to low-priority clients.

Figs. 8 and 9 illustrate the comparison of unfairness for the low-priority clients in the five resource sharing schemes. The three proposed schemes were found to be fairer than MFQL, because the proposed scheduling policy schedules each video by the waiting time and leaving probability of each client in that video queue. Therefore, the proposed scheduling policy selects the next video to be served based on the maximum waiting time and the maximum leaving probability. Interestingly, the hybrid system embedded with channel borrowing and reservation mechanisms (AHVoD3) performed better than AHVoD2 and AHVOD1, because its channel borrowing and reservation mechanisms effectively allocate the free channels operated in controlled multicasting mode, increasing the possibility of low-priority tasks accessing

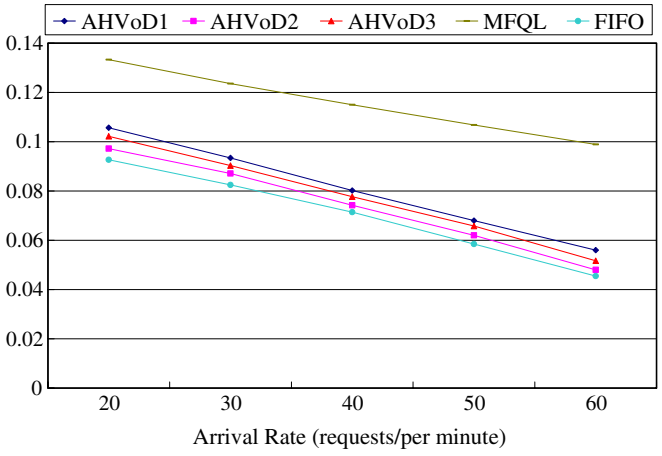

Fig. 8. Unfairness of low-priority clients at varied arrival rates.

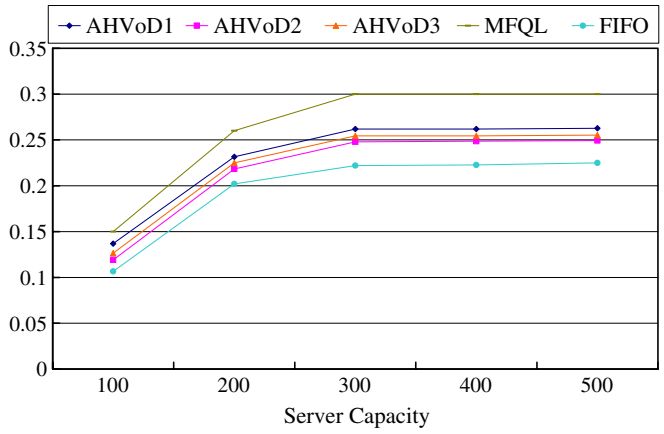

Fig. 9. Unfairness of low-priority clients under varied server capacity.

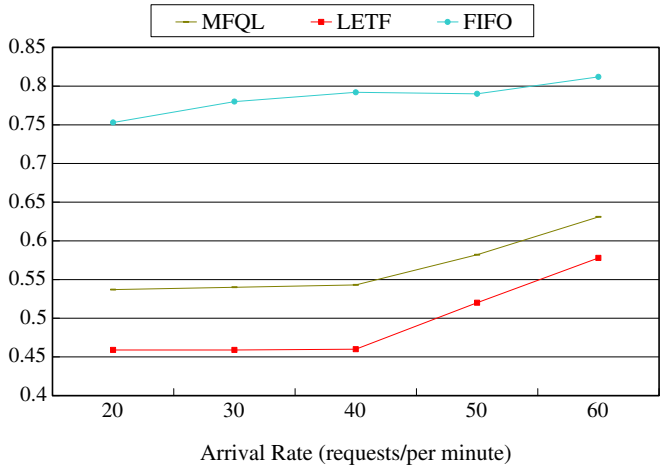

Fig. 10. Defection probability of the clients at varied arrival rates.

the video data, thereby serving more low-priority videos. Although FIFO is the fairest scheme among the five schemes, the proposed three hybrid system can achieve lower defection probability and short service delay time, as indicated in Figs. 3–6.

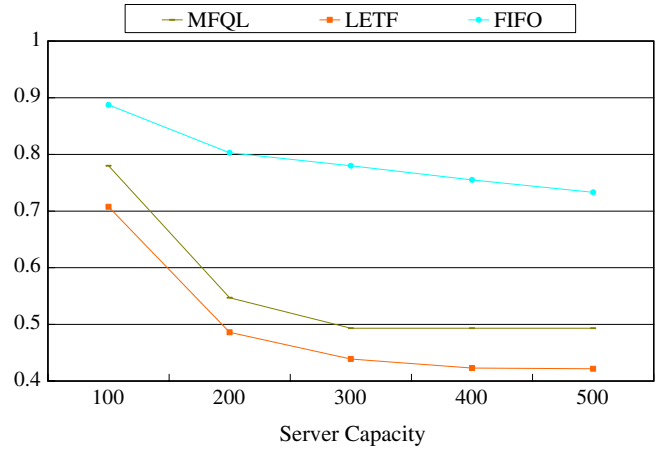

Fig. 11. Defection probability of the clients under varied server capacity.

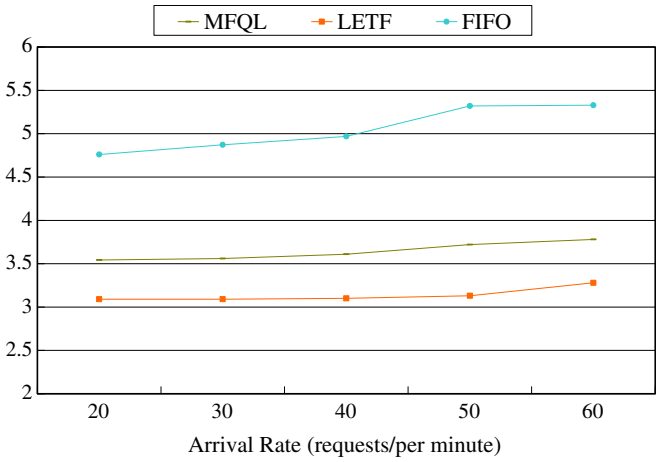

Fig. 12. Service delay time of the clients at varied arrival rates.

To measure the effectiveness of the proposed scheduling policy, a further series of test was run under batching mode wherein all the clients were treated identically. The proposed scheduling policy (LETF) was compared with MFQL and FIFO. Figs. 10 and 11 compare the defection probability of the clients under the three scheduling policies. As expected, the proposed scheduling policy effectively reduced the defection probability. Since the proposed scheduling policy based on the probability model builds the video queue according to the waiting time and leaving probability of each client, it selects the next video to serve before the client leaving the system by waiting for a long time.

The proposed scheduling policy has a shorter service delay time than MFQL and FIFO, as indicated in Figs. 12 and 13. This is because the scheduler can effectively schedule the video queue with the maximum waiting time multiplied by the maximum leaving probability. A longest queue often has the largest value for this factor.

Figs. 14 and 15 compare the unfairness of the proposed scheduling policy with MFQL and FIFO. Clearly, the leaving probability of the clients in the queue and the waiting

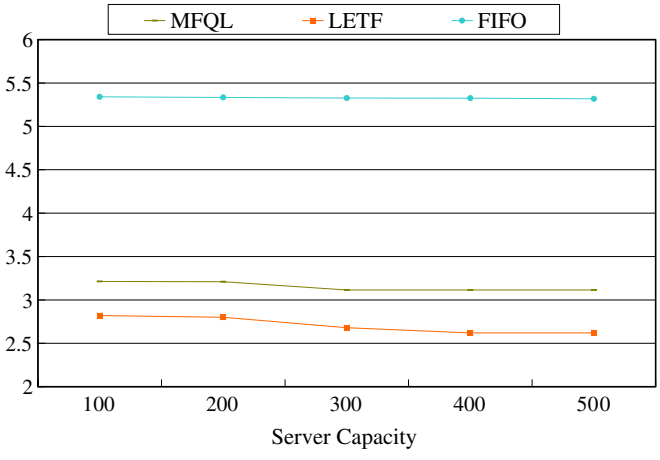

Fig. 13. Service delay time of the clients under varied server capacity.

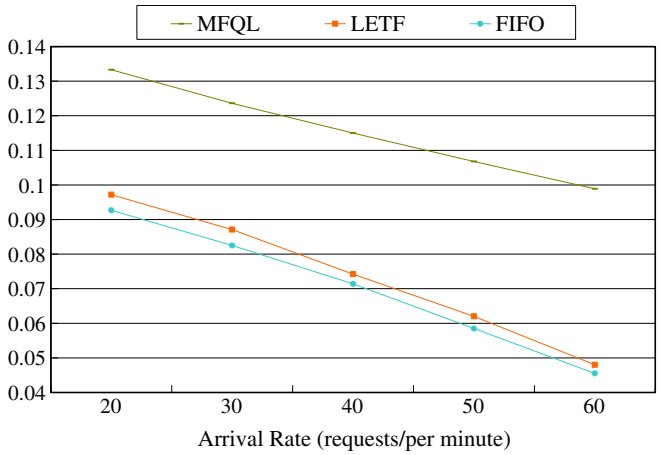

Fig. 14. Unfairness of the clients at varied arrival rates.

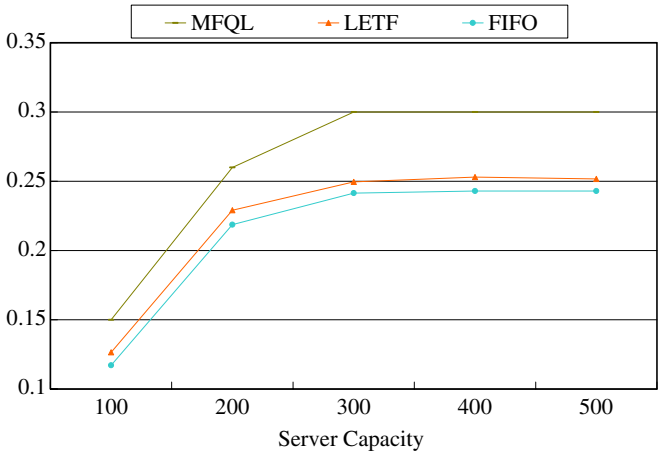

Fig. 15. Unfairness of the clients under varied server capacity.

time of the incoming clients makes the proposed scheduling policy fairer than MFQL, but slightly less fair than FIFO. The selection criteria for the next video to serve in Eq. (6) reveals that the queue with longest waiting time and highest leaving probability value is served next. A client who waited for a long time, and has high probability of being defected by the system, should ideally be served first.

4. Conclusions

This study proposes a scheduling model that allows high-priority clients who pay for the requested videos to receive them quickly from the VoD server, while also giving service to users who access the videos for free. A hybrid VoD resource sharing system along with a bandwidth borrowing technique is proposed to reduce the probability of defection and the waiting times of clients in different priority groups. Furthermore, an adaptive bandwidth reserving mechanism is also presented to improve the efficiency of the VoD server in using free streams. Additionally, a probability-based scheduling policy, called largest effective waiting time first, is proposed to improve the fairness of the batch scheduler, and decrease the probability of defection. A series of simulations was run to compare the proposed hybrid VoD resource-sharing model with the controlled multicasting and MFQL scheduling model. The proposed hybrid model was found to lower the blocking probability of high-priority clients. The defection probability and delay latency of low-priority clients were found to be lower in the proposed scheme than in MFQL. Experimental results show that the proposed bandwidth borrowing and adaptive channel reservation model can decrease the blocking probability of high-priority clients and the defection probability of low-priority clients. Meanwhile, the proposed probability-based scheduling model not only has the lowest defection probability, but also the best fairness, for low-priority clients. The superiority and the feasibility of the proposed hybrid resource-sharing scheme are thus verified. Notably, Lehtonen et al. [23] reported that controlled multicast framework does not enable Internet-wide control for multicast sources and receivers, and is not completely secure. Future work will focus on resolving these two issues.

Acknowledgments

The authors thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contract No. NSC 95-2221-E-026-001. Ted Knoy is appreciated for his editorial assistance.

References

[1] A. Bar-Noy, J.A. Garay, A. Herzberg, Sharing video on demand, Discrete Applied Mathematics 129 (1) (2003) 3–30.
[2] L. Juhn, L. Tseng, Harmonic broadcasting for video-on-demand service, IEEE Transactions on Broadcasting 43 (3) (1997) 268–271.
[3] B. Qazzaz, J. Moreno, J. Xiao, P. Hernandez, R. Suppi, E. Luque, Admission control policies for video on demand brokers, International Conference on Multimedia and Expo 2 (2003) 529–532.
[4] L. Rovira, L. Bombelli, Scientific-Atlanta, and Paul Brooks, Time Warner Cable, The elements of an architecture, switched digital video, Communications Technology magazine, 23(02), 2006.
[5] W.K.S. Tang, E.W.M. Wong, S. Chan, K.-T. Ko, Optimal video placement scheme for batching VOD services, IEEE Transactions on Broadcasting 50 (1) (2004) 16–25.
[6] W.-F. Poon, K.-T. Lo, J. Feng, Determination of efficient transmission scheme for video-on-demand (VoD) services, IEEE Transactions on Circuits and Systems for Video Technology 13 (2) (2003) 188–192.
[7] W.-F. Poon, K.-T. Lo, J. Feng, Provision of continuous VCR functions in interactive broadcast VoD systems, IEEE Transactions on Broadcasting 51 (4) (2005) 460–472.
[8] J. B Kwon, H.Y. Yeom, VCR-oriented video broadcasting for near video-on-demand services, IEEE Transactions on Consumer Electronics 49 (4) (2003) 1106–1113.
[9] S.A. Azad, M. Murshed, L.S. Dooley, Bandwidth borrowing schemes for instantaneous video-on-demand systems, IEEE International Conference on Multimedia and Expo 3 (2004) 2011–2014.

[10] L. Gao, D. Twosly, Supplying instantaneous video-on-demand services using controlled multicast, in: Proceedings of IEEE International Conference on Multimedia Computing and Systems, 1999, pp. 117–121.
[11] A. Dan, D. Sitaram, P. Shahabuddin, Scheduling policies for an on-demand video server with batching, in: Proceedings of the ACM Conference on Multimedia, 1994, pp. 391–398.
[12] H. Yu et al., Understanding user behavior in large scale video-on-demand systems, in: Proceedings of EuroSys, April, 2006.
[13] E. Veloso et al., A hierarchical characterization of a live streaming media workload, in: Proceedings of the ACM SIGCOMM IMW, November, 2002.
[14] K. Sripanidkulchai et al., An analysis of live streaming workloads on the Internet, in: Proceedings of the ACM SIGCOMM IMC, October, 2004.
[16] C.C. Aggarwal, J.L. Wolf, P.S. Yu, The maximum factor queue length batching scheme for video-on-demand systems, IEEE Transactions on Computers 50 (2) (2001) 97–110.
[17] A. Salahuddin Azad, M. Manzur, An efficient transmission scheme for minimizing user waiting time in video-on-demand systems, IEEE Communications Letters 11 (3) (2007) 285–287.
[18] N.L.S. Fonseca, H.K.S. Rubinsztejn, Dimensioning the capacity of true video-on-demand servers, IEEE Transactions on Multimedia 7 (5) (2005) 932–941.
[19] J.Y.B. Lee, Channel folding – an algorithm to improve efficiency of multicast video-on-demand systems, IEEE Transactions on Multimedia 7 (2) (2005) 366–378.
[20] C.-W. Kong, J.Y.B. Lee, M. Hamdi, V.O.K. Li, Turbo-slice-and-patch: an algorithm for metropolitan scale VBR video streaming, IEEE Transactions on Circuits and Systems for Video Technology 16 (3) (2006) 338–353.
[22] M.-Y. Wu, S. Ma, W. Shu, Scheduled video delivery-a scalable on-demand video delivery scheme, IEEE Transactions on Multimedia 8 (1) (2006) 179–187.
[23] R. Lehtonen, J. Harju, Controlled multicast framework, in: Proceedings, LCN 2002, 27th Annual IEEE Conference on Local Computer Networks, 2002, pp. 565–571.