Best-Effort Patching for Multicast True VoD Service *

HUADONG MA
(mhd@bupt.edu.cn, School of Computer Science & Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China).
KANG G. SHIN
(kgshin@eecs.umich.edu, the Real-Time Computing Laboratory,
EECS Depart., The University of Michigan, Ann Arbor, MI 48109, USA).
WEIBIAO WU
(wbwu@galton.uchicago.edu,Department of Statistics,
The University of Chicago, IL 60637,USA)

Abstract

A multicast Video-on-Demand (VoD) system allows clients to share a server stream by batching their requests, and hence, improves channel utilization. However, it is very difficult to equip such a VoD system with full support for interactive VCR functions which are important to a growing number of Internet applications. In order to eliminate service (admission) latency, *patching* was proposed to enable an existing multicast session to dynamically add new clients, and requests can be served without delay if patching channels are available. A true VoD (TVoD) service should support not only zero-delay client admission but also continuous VCR-like interactivity. However, the conventional patching is only suitable for admission control. We propose a new patching scheme, called *Best-Effort Patching* (BEP), that offers a TVoD service in terms of both request admission and VCR interactivity. Moreover, by using a novel dynamic merging algorithm, BEP significantly improves the efficiency of TVoD interactivity, especially for popular videos. We also model and evaluate the efficiency of the dynamic merging algorithm. It is shown that BEP outperforms the conventional TVoD interaction protocols.

Keywords: QoS, VCR interaction, Best-Effort Patching (BEP), Video-on-Demand (VoD), multicast.

*This work reported in this paper was partly done during Huadong Ma's visit to the University of Michigan, and supported in part by the NSF of USA under Grant EIA-9806280, the NSFC of China under Grant 69873006 and 60242002, the Chinese EYTP of MOE.

1 Introduction

Video-on-Demand (VoD) service allows remote clients to play back any video from a large collection of videos stored at one or more video servers in any mode at any time. VoD service is usually long-lived and real-time, and requires high storage-I/O & network bandwidths, and needs to support VCR-like interactivity. A VoD system is usually designed with a focus on system cost and client-perceived Quality-of-Service (QoS). The key cost components are the video server capacity, storage-I/O bandwidth, network bandwidth & throughput, and customer premise equipment (CPE). VoD clients' QoS is related to service latency, interactivity, and playback effects. Usually, there is a trade-off between clients' QoS and system costs. TVoD service supports all of the control functions [21], and is an ideal service for customers. The conventional TVoD system uses one dedicated channel for each service request, which offers the client the best QoS and interactive service. However, it incurs high system costs, especially in terms of storage-I/O and network bandwidth. Moreover, the conventional VoD service has poor scalability and low cost/performance efficiency. One efficient solution to improve VoD systems is to use multicast.

Multicast offers efficient one-to-many data transmission and thus provides the foundation for various applications that need to distribute data to many receivers in a scalable manner. It reduces both the server-side overhead and the overall network load. Thus, multicast VoD has good scalability and excellent cost/performance efficiency (See [23] for an excellent survey of multicast VoD services). However, it is difficult to support VCR-like interactivity with multicast VoD and, at the same time, improve service efficiency. There are several proposals [1, 4, 20] to solve this problem. Because different videos are requested at different rates and at different times, the popularities of videos are assumed to follow the Zipf distribution with a skew factor of 0.271. Videos are usually hot (popular) or cold, and requests for the top 10-20 of 100 videos are known to constitute over 60% of the total demand. So, it is crucial to improve the service efficiency of hot videos. We will, in this paper, focus on both service latency and interactivity for hot videos.

One approach is to multicast each popular video at equally-spaced intervals as in [8]. An alternative is to use a fixed number of multicast channels to periodically broadcast video objects to a group of subscribers [3, 10, 14, 17, 27, 24]. This periodic broadcast is efficient in transmitting popular videos from one server to many clients, but it is difficult to support VCR interactivity. For the equally-spaced interval multicast, the clients' requests between two consecutive regular channels must be delayed, which will degrade service latency and system throughput. In order to eliminate service/admission latency, such techniques as adaptive piggybacking [18], patching [16] and bandwidth skimming [12, 13], are proposed to enable an ongoing multicast channel to serve new additional clients. Patching offers a good method to improve QoS of multicast VoD services, but the conventional patching [6, 16] is only suitable for TVoD admission control. In order to support both admission control and interaction for TVoD service, we propose a new patching scheme, called *Best-Effort Patching* (BEP), which works in three phases as follows.

Admission: BEP multicasts a popular video via regular channels at fixed time intervals. Requests arriving between two consecutive regular channels will share the latest regular stream by patching the missed leading segment.

Interaction: During VCR-like interactions, a client won't need any other channel if the interaction duration is less than that supported by the CPE buffer. Once a client interaction exceeds the capacity of his CPE buffer, BEP dispatches a patching channel to support the client's interaction.

Merging: After finishing an interaction in case it exceeds the capacity of CPE buffer, the client needs to join a regular patching group, but he may not be able to join an existing channel right away because the channel doesn't have his desired playback time. In such a case, if there exists an interaction channel, it will be merged into a regular stream by using the interaction channel; otherwise, BEP dispatches a patching channel to the client so that he can receive continuous interaction service without any disruption.

Moreover, BEP uses a novel dynamic merging scheme and thus offers TVoD service efficiently. Our analysis and simulation show that the proposed new scheme outperforms the conventional multicast TVoD interaction protocols.

The remainder of this paper is organized as follows. Section 2 introduces the conventional patching for VoD service, and overviews the interactive functions for TVoD. BEP for TVoD service is presented in Section 3. Section 4 evaluates the performance of BEP and presents the simulation results. Finally, we summarize our contributions in Section 5.

2 Background

Patching offers excellent performance as a general efficient means of eliminating service latency for multicast VoD service. In this section, we first review the patching technique and then discuss VCR interactivity necessary in TVoD service.

2.1 Conventional Patching

The bottleneck of VoD service is the limited number of channels available to a video server. It is therefore important to have channels shared among as many clients as possible, i.e., early requests for a video are forced to wait for more requests to arrive, and the entire group of "batched" requests is then served via a single multicast channel. This is referred to as *batching*, and since most requests are forced to wait, only near-VoD can be achieved. It is desirable to keep the client's maximum waiting time as short as possible, because clients are likely to renege if they are kept waiting too long. To eliminate service latency, some dynamic techniques are proposed [16, 18] by enabling each multicast session to dynamically add new requests. Patching [16] is a typical dynamic multicast. An important objective of patching is to substantially increase the number of requests each channel can serve per time unit, thereby sufficiently reducing the per-customer system cost. The bandwidths of the server and the network are organized into a set of logical channels each of which is capable of transmitting a video at the clients' playback rate. A new service request can exploit an existing multicast channel by buffering the video stream from the multicast while playing a new catch-up stream (obtained via a patching channel) from the beginning. Once the new catch-up stream is played back to the skew point, it can be terminated and the new client starts to consume the multicast data from the buffer (the skew is absorbed by the new client's buffer). Allowing clients to dynamically join an existing multicast group improves the multicast efficiency. Moreover, requests can be honored immediately, achieving zero-delay VoD service.

In the patching scheme, channels are often used to patch the missing portion of a video, or deliver a patching stream, rather than multicasting the entire video. Given that there is an existing multicast of a video, when to schedule another multicast for the same video is crucial. The time period during which patching must be used, is referred to as the *patching window* [6]. If the patching window is set too small, multicasts are scheduled too frequently. Two simple approaches to setting the patching window are discussed in [16]. The first uses the length of the video as the patching window. That is, no multicast is initiated as long as there is an ongoing multicast for the video. This approach is called the *greedy patching* because it tries to exploit an existing multicast as much as possible. However, an over-greed can actually result in less data sharing [16]. The second approach, called the *grace patching*, uses a patching stream for the new client only if it has enough buffer space to absorb the skew. Hence, under grace patching, the patching window is determined by the client buffer size. An improved patching technique, called as the *transition patching* [7], improves performance without requiring any extra download bandwidth at the client site. Considering such factors as video length, client buffer size, request rate, some optimal patching schemes were presented in [6, 11, 15, 25]. [5] discussed the implementation,measurement, and analysis of a working video server testbed implementing

patching algorithms. [26] developed a simple analytic model to evaluate the performance of multicast streaming protocols including patching.

In patching, a client might have to download data from both regular multicast and patching channels at the same time. To implement patching, a client station needs to have two data loaders from the two channels, and a video player to play back the video.

2.2 VCR Interactivity in TVoD

After their admission, customers can have the following types of interactions, as identified in [19].

Play/Resume **:** regular play-out from the beginning or any other location of a video.

Stop/Pause/Abort **:** stopping the presentation, without picture or sound (Stop, Abort), or with picture and without sound (Pause). An Abort action terminates the connection.

Fast Forward/Rewind **:** fast jump to a particular video location in the forward (backward) direction.

Fast Search/Reverse Search **:** quickly moving the presentation forward (backward) with picture and possibly sound.

Slow Motion **:** moving the presentation forward slowly, with picture and, possibly, sound.

TVoD service may also provide support for other types of interactions, such as *Reverse* and *Slow Reverse*, which correspond to a presentation in the reverse direction, at normal or slow speed. In this paper, we will not consider them as part of the usual interactive behavior of a customer.

We classify interactive operations into two types: (1) *forward interactions*, such as Fast Forward and Fast Search; (2) *backward interactions*, such as Rewind, Reverse Search, Slow motion, and Stop/Pause. This classification depends on whether the playback rate after interactive operations is faster than the normal playback or not. Multicast VoD systems can offer either *continuous* or *discontinuous* interactive functions. Continuous interactive functions allow a customer to fully control the duration of all actions to support TVoD service, whereas in case of discontinuous interactive functions, an action can only be specified for a duration that is an integer multiple of a predetermined time (thus supporting NVoD service). Note that the size of discontinuity is a measure for the QoS experienced by the customers under NVoD service.

From an implementation viewpoint, we also categorize interactions as *interactions with picture* or *interaction without picture*. Fast/Reverse Search and Slow Motion are typical interactions with picture, whereas Fast Forward and Rewind are typical interactions without picture. In general, it is easier to implement interactions without picture because it requires less system resources.

In order to implement the interactivity of multicast VoD service, some efficient schemes have been proposed. For example, the SAM protocol [20] offers an efficient way for TVoD interactions, but it requires many I(interaction)-channels, thus resulting in a high blocking rate. The authors of [1] improved the SAM protocol by using the CPE buffer. Other researchers, such as those of [4], focused on interactions without picture. In this paper, we present an efficient approach to the implementation of continuous TVoD interactions.

3 Best-Effort Patching for TVoD Interactivity

3.1 Basic Idea

The conventional patching, such as grace patching, supports NVoD VCR interactive functions because there are only $\lceil \frac{L}{w} \rceil$ regular multicast channels for a video, where L is the average length of videos, and w is the size

of patching window, which is closely related to the CPE buffer size, d. In general, we may take $w = d$, but may sometimes take $w \neq d$ for optimal patching efficiency [6].

Continuous service of VCR actions can be supported in multicast VoD systems by employing the CPE buffer, but this support is limited by the size of CPE buffer. The CPE buffer and VCR actions are depicted in Figure 1. The CPE buffer can be thought of as a *sliding window* over the largest usable sequential portion of the video frame, which is composed of the video between the most recent frame prefetched from the multicast group and the oldest frame. The play point lies between these two frames, usually at the middle of buffer. Initially, the play point corresponds to the most recent frame and the CPE buffer is progressively filled in with past frames as the playback continues. The *Play* operation doesn't change the relative position of the play point with respect to the most recent frame. When Forward/Backward interactions are performed, the play point will eventually be near the most recent frame/the oldest frame. The displacement of the play point depends on the speedup factor *SP* or the slow motion factor *SM*. For example, the Fast Forward interaction spanning over time t will cause an actual displacement of $(SP-1)t$. Suppose d_r (usually equals $d/2$) is the displacement between the play point and the most recent frame, then Fast Forward will be supported continuously if $t \leq \frac{d_r}{SP-1}$. A similar observation can be made for a backward interaction. For instance, if A is the desired play point after an interaction, the displacement between A and the normal play point of its multicast group is less than d_r, the continuous interaction can be achieved with the CPE buffer.

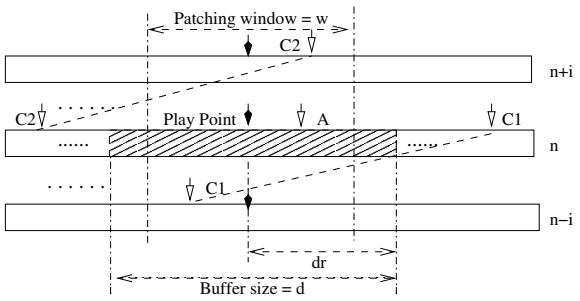

Figure 1: The CPE buffer and VCR actions

Now, let's consider another situation when a Fast Forward interaction was performed over time $t \geq \frac{d_r}{SP-1}$, or Rewind interaction took place over the time allowed by the CPE buffer. The CPE buffer can't guarantee a smooth transition between adjacent multicast groups, i.e., a discontinuous VCR interaction may occur. The interaction cannot always join an existing channel immediately because there may not always be a channel with the desired playback time. So, it may have to wait to join the nearest channel.

BEP eliminates discontinuity so that customers may enjoy zero-delay VCR interactions. The main idea behind BEP is as follows. In executing VCR interactive operations, once the customer leaves his CPE buffer (i.e., the interaction time exceeds the capacity of the CPE buffer), he needs a channel to execute the interaction and join a new regular multicast group after finishing the interaction. In this case, BEP dispatches a patching channel to transmit the video from the desired time point, and supports the interaction and its merging into the latest multicast channel. For an interaction without picture, BEP only needs to dispatch a patching channel for its merge into an existing channel after the interaction, where the client downloads the video from both the patching channel and the latest multicast channel simultaneously. The video from the patching channel is played back immediately, and the video from the latest multicast channel is buffered in the CPE buffer. Note that the latest multicast channel is the closest one whose play point is earlier than, or equal to, the desired play point. For merging, the patching channel is required only for the displacement between the play point of the latest channel and the desired play point in order to catch up with the latest multicast channel, and then the patching channel is released. This way, the customer seamlessly joins an existing multicast group. Neither

does it incur any additional CPE cost because it just makes use of the CPE buffer required by the conventional patching.

BEP offers customers a TVoD service in both request admission and VCR interactivity. BEP also uses an efficient dynamic merging technique to reduce the channel requirement. This scheme is a *general* multicast VoD approach.

As mentioned earlier, BEP works in three phases: admission, interaction, merging. New requests are admitted by transition patching or grace patching. Next, we will discuss the interaction and merging phases.

3.2 Support for VCR Interactivity

Discussed below is how BEP supports interactions without and with picture.

A. Fast Forward/Rewind/Pause/Stop

These four interactions are very simple for BEP to support because they don't need picture during interactions. BEP supports them in the following two cases.

Case 1: When the resume point of an interaction is located within the CPE buffer (i.e., the CPE buffer can completely guarantee continuous interaction), no additional patching channel is required for the interaction.

Case 2: When the resume point of an interaction is outside the CPE buffer (i.e., the CPE buffer cannot completely guarantee continuous interaction), no channel is required during the interaction, because no video data is transmitted. However, after completing the interaction, the customer's video stream has to be merged into the latest existing multicast group, so that the customer may obtain the video data from the desired point via a patching channel. In Figure 2, for a user in the multicast group n, the original play point is located at P_0. After a forward interaction, such as Fast Forward, the normal play point of the group n becomes $P(n)$. But the desired play point is A, which is located between the play points of group $n-i$ and group $n-i-1$ which are denoted as $P(n-i)$ and $P(n-i-1)$, respectively.[1] Note that the time displacement between groups n and $n-i$ is $i*w$. After the interaction is completed, the user's video stream should be merged into group $n-i-1$. So, a patching channel is required to enable the user to catch up with the multicast stream of group $n-i-1$, and the length of the patching channel for such merging is $|P(n-i-1)-A|$ [2] which maps the grid part in Figure 2. Similarly, one can explain the backward interaction shown in Figure 3.

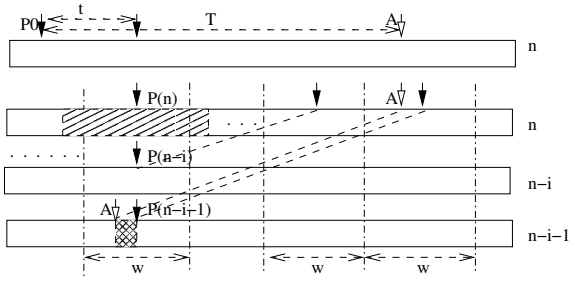

Figure 2: Forward interactions

[1] The group number of a multicast stream scheduled later is greater than that of a stream scheduled earlier.

[2] $|P-A|$ denotes the distance between P and A.

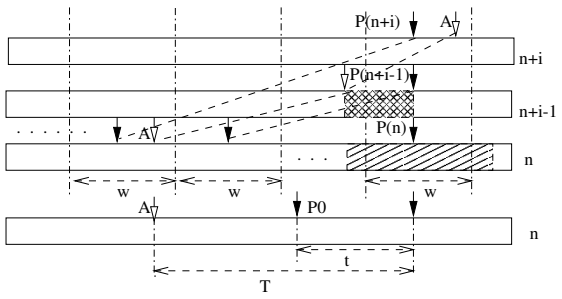

Figure 3: Backward interactions

B. Fast Search/Reverse Search/Slow Motion

These interactions require BEP to assign I-channels once their storage requirement (to hold pictures during an interaction) exceeds the CPE buffer capacity. After completing the interaction, the customer can obtain the video data from the desired point through a patching channel. Interaction and patching are usually done by using the same channel. The channel occupancy time is the sum of interaction and patching times. As before, we illustrate the BEP-supported interactions with picture in the following two cases.

Case 1: When the resume point of an interaction is located within the CPE buffer (i.e., the CPE buffer can guarantee continuous interaction with picture), no additional channel is required for the interaction.

Case 2: Once the interaction's storage requirement exceeds the CPE buffer capacity, BEP will assign a channel to the customer. The channel is used during both the interaction and merging phases. In the interaction phase, the channel acts like an I-Channel in the SAM protocol [17], while it is used as a patching channel in the merging phase. For the forward interaction shown in Figure 2, such as Fast Search, the interaction takes t units of time, and the customer's desired play point is A. Upon completion of the interaction, the normal play point of the multicast group n that the customer shares, becomes $P(n)$. The interaction phase uses the channel for a period of $t - t_0$ $(t > t_0)$, where t_0 is the duration the CPE buffer can support Fast Search without an additional channel, and $t_0 = \frac{d_r}{SP-1}$. The merging phase uses the channel for the same amount of time as the interaction phase without picture. That is, the length of a patching channel for merging is $|P(n-i-1) - A|$ in Figure 2. For the backward interaction shown in Figure 3, such as Reverse Search or Slow Motion, one can illustrate similarly.

In BEP, the CPE has three threads: 2 Stream Loaders which download video streams from the patching (interaction) channel and the regular multicast channel, and a Video Player which plays the video from the two channels.

3.3 The Dynamic Merging Algorithm

Like the SAM protocol, BEP requires many channels for merging clients' interaction streams into regular multicast streams when interactions took place over the time allowed by the CPE buffer. Our simulation shows the channel requirement to be high for the merging phase. Thus, the efficiency of merging is critical to interaction QoS. In order to improve the merging of interaction and multicast streams, BEP uses a novel dynamic merging approach as described below.

Let $P(n)$ be the play point of multicast group n. After completing an interaction, A is the desired play point in Figure 4. Thus, the client needs to use a patching channel to catch up with the multicast stream of group n. The patching stream for A is denoted as *Stream A*, and its *lifetime* is the same as its group offset $a = |P(n) - A|$.

When the merging is going on, say t_1 minutes later, the other interaction completes and its stream also needs to be merged into that of group n, and its desired play point is B behind A in the relative offset of the stream. The group offset of B is $b = |P(n) - B|$. If t_1 is less than the lifetime of Stream A and $a - t_1 > b - a$, we can make the patching stream for the second interaction (denoted as *Stream B*) share the grid part of Stream A so that the need/use of patching channels can be reduced. In this situation, Stream A will be extended so that its lifetime is b, and the lifetime of Stream B is set to $b - a$. The new client to download the video segment from Stream A and B simultaneously, and begins to download the video from the stream of multicast group n after using up Stream B. Note that the latter client can't download the video from the stream of group n when it downloads video segments from Stream A and B simultaneously because there are only two stream loaders. Such merging of patching channels can go on, and the merged clients can be recorded in a *merging queue*.

We now consider the merging of a new interaction. Assume that q is a merging queue for multicast group n, the offset of q is the group offset of its first client, and the head record of q is the client patching stream initiated this queue, and the lifetime of q is initially set to the offset of its head and may be changed when a new client joins it. A merged client patching stream is called as an *element* of the queue. If t is the time when the latest client joins, the queue will be released when the time is t plus its lifetime. A merging queue has the following data structure:

struct MQueue {
*element *head*; /*the head record*/
int offset; /*the offset of queue*/
int lifetime; /*the lifetime of queue*/
int latime; /*joining time of the latest client*/
}

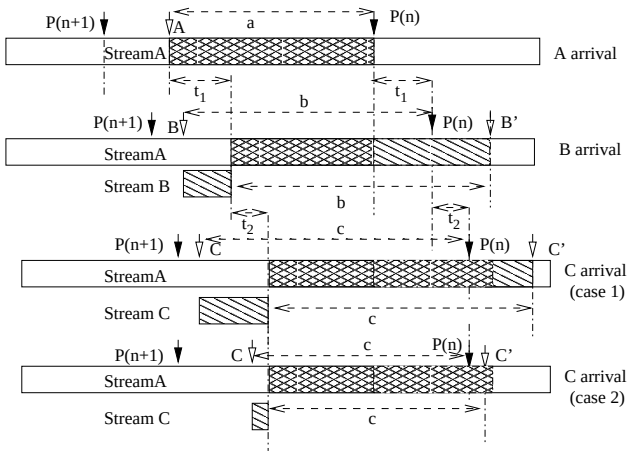

Figure 4: Dynamic merging

When a new client C wants to join this merging queue that holds a merging stream A, given that *c* is C's group offset and the arrival time of C is $t+t_2$. We also assume that $c \geq$ *q.offset* and $c <$ *q.offset*+ *q.lifetime*$-t_2$; otherwise, C shouldn't join q. We can manage this queue for C to join in the two cases shown in Figure 4.

Case 1: $c > q.lifetime - t_2$, meaning that the lifetime of the queue q is not long enough to merge C, so *q.lifetime*, also equal to the lifetime of Stream A, must be extended. That is, after C is merged, the lifetime of Stream C, the patching stream for the missing leading segment of C, is set to *c-q.offset, q.lifetime=c, q.latime*$= t + t_2$. The client downloads the video segments from Streams A and C simultaneously. After

c-q.offset time units, the client begins to download the video from the stream of group n. In this case, the usage time of a patching channel that our algorithm can save is *q.offset*+*q.lifetime*$-t_2 - c$ time units.

Case 2: $c <=$*q.lifetime*$-t_2$, meaning that the lifetime of the queue q is long enough to merge C, so the lifetime of q (or Stream A) need not be extended. After C is merged, the lifetime of Stream C, the patching stream for the missing segment of C, is set to *c-q.offset*, *q.latime*$= t + t_2$, *q.lifetime* remains unchanged. The client downloads the video segments from Streams A and C, and the stream of group n in the same way as in Case 1. In this case, the usage time of a patching channel that our algorithm can save is *q.offset* time units.

If C can't be merged into the existing queues, a new queue will be initiated. Because there are probably many merging queues, C is likely to be merged into the queue which saves the maximum usage time of the channel.

The dynamic merging algorithm for the server-end is described below.

Algorithm DMA(Q, C, n, t)
/*Q is a set of merging queues*/
/*C is an interaction client*/
/*n is the group number*/
/*t is the arrival time of C*/
$c = offset(C, n)$; /* get the offset of C in group n*/
$q_0 = \max_{q \in Q}\{min\{q.offset + q.lifetime - c - (t - q.latime), q.offset\} | q.offset \leq c < q.offset + q.lifetime - (t - q.latime)\}$
if ($q_0 = null$) { /* generate a new merging queue*/
$Genqueue(q_0)$; /*generate a new queue*/
$q_0.head$ is set to C; $q_0.offset = c$;
$q_0.lifetime = c$; $q_0.latime = t$;
}
else { /* merging C into q_0 */
$delta = t - q_0.latime$;
$q_0.latime = t$;
$lifetime(C) = c - q_0.offset$; /* the lifetime of patching stream C is changed */
if ($c > q_0.lifetime - delta$) $q_0.lifetime = c$;
}

In the client-end, when merged into a queue, the client downloads video segments from the stream of the merged queue (that of the queue head element) and the patching stream for merging the client interaction simultaneously. After the patching stream for merging the client interaction is downloaded, the client begins to download the video from the multicast stream she/he joins.

3.4 Discussion

In the above illustration, we assumed that video delivery is scheduled as multicast streams at a *fixed* interval. Actually, BEP can support multicast batching at varying intervals. Even at a fixed interval, BEP allows the interval to be adjusted. That is, the patching window, limited by the CPE buffer, can be adjusted according to the request rate. On the other hand, when the request rate is high, the patching (interaction) channel requirement will increase. Because server storage-I/O and network bandwidths are limited, it may be difficult to guarantee QoS in such a case. To handle this difficulty, we support graceful degradation of TVoD service (e.g., only interactions without picture), and exercise admission control for a fixed worst-case period in case of limited channel resource.

BEP differs from the SAM protocol in at least three aspects. First, BEP aims to offer a zero-delay (or continuous) service for both request admission and VCR interactions, whereas the SAM protocol just supports continuous VCR interactions without considering service admission. Thus, patching channels are used to patch all of the segments that can't be provided by regular multicast channels for TVoD service, whereas the I-Channels in the SAM protocol are used only for VCR interaction service. Second, the SAM protocol uses *synchronization buffer* to merge I-Channels and regular multicast channels, whereas BEP uses a patching channel to patch the missing segments of multicast VoD and merge it with regular multicast channels using the client's CPE buffer. The CPE buffer can be used to save patching channels for VCR interactions and merging efficiency. Of course, use of the CPE buffer can also improve the efficiency of the SAM protocol [1]. Third, BEP uses a dynamic technique to merge interaction streams with regular multicast streams, significantly improving the efficiency of multicast TVoD service.

In summary, the SAM protocol only focuses on continuous VCR interactions, and the conventional patching just aims to eliminate the admission latency of clients' requests. Thus, neither of them can support TVoD service. By dynamically admitting requests and merging streams, BEP offers TVoD service efficiently with both zero-delay admission and continuous interactions.

4 Performance Evaluation

4.1 Client's interaction model

A client's behavior can be characterized by his admission and interaction behavior. The authors of [2] discussed the admission behavior and derived the optimal throughput related to service latency. In this section, we focus on the client's interaction models.

Several interaction models have been proposed [1, 2, 9, 19]. A model should capture three specific parameters that may significantly impact the performance of a VoD server: (1) the frequency of requests for VCR actions, (2) the duration of VCR actions, (3) the bias of interaction behavior. The model proposed in [1] contains all these parameters and hence is adopted here. In this model, a set of states corresponding to different VCR actions are designed durations and probabilities of transition to neighboring states. If the initial state is Play, then the interaction system randomly transits to other interactive states or remains at Play state according to the behavior distribution. A patching channel serves each interaction state for an exponentially-distributed period of time.

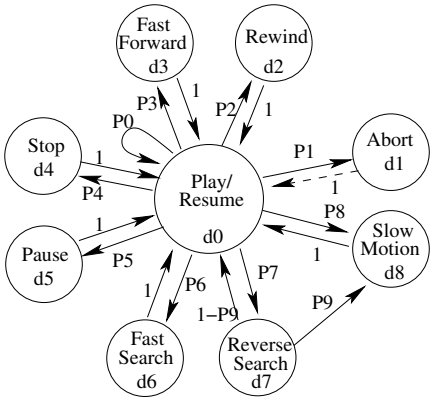

Figure 5: The VCR interactive model

4.1.1 Interaction Behavior Distribution

As shown in Figure 5, transition probabilities P_i $(i = 0, \ldots, 9)$ are assigned to a set of states corresponding to different VCR actions. It is important to notice that the above-mentioned parameters are captured by this representation of viewers' activities. Finding representative values for P_i $(i = 0, \ldots, 9)$ is still an open issue, since there are no published realistic models or data of customer interactions. For tractability, we divide customers into two types: *Very Interactive (VI)* or *Not Very Interactive (NVI)*. Our simulation assumes that Slow Motion is requested only after Reverse Search, i.e., $P_8 = 0$, and $P_9 = 0.5$. The other transition possibilities are summarized as Table 1.

Table 1. Transition probabilities from Play/Resume

Behavior	P_0	P_1	P_2, P_3	P_4	P_5	P_6, P_7
VI	0.50	0.04	0.08	0.06	0.08	0.08
NVI	0.75	0.02	0.04	0.03	0.04	0.04

4.1.2 Interaction Duration Distribution

Assume that t is the duration of a client's interaction. Unlike the assumption of fixed-duration interaction states in [19, 20], we assume that BEP serves each state for an exponentially-distributed duration, the distribution parameters for Play, Rewind, Fast Forward, Stop, Pause, Fast Search, Reverse Search, and Slow Motion are $\mu_0 = \frac{1}{d_0}$, $\mu_2 = \frac{1}{d_2}$, $\mu_3 = \frac{1}{d_3}$, $\mu_4 = \frac{1}{d_4}$, $\mu_5 = \frac{1}{d_5}$, $\mu_6 = \frac{1}{d_6}$, $\mu_7 = \frac{1}{d_7}$, $\mu_8 = \frac{1}{d_8}$, respectively, where d_i $(i = 0, 1, 2, \ldots, 8)$ are the mean durations for the corresponding interaction states ($d_1 = 0$), and their default values are given in Table 2. Meanwhile, the speedup factors of Fast Forward/Rewind and Fast Search/Reverse Search are defined as K_0, K_1, respectively, and the speeddown factor of Slow Motion is defined as K_2. Our simulation used $K_0 = 10$, $K_1 = 3$, and $K_2 = 2$.

Table 2. Mean interactive durations

Parameter	d_0	d_1	d_2, d_3	d_4, d_5	d_6, d_7	d_8
Default	10	0	0.5	5	2.5	2

The interaction duration of a client depends on his behavior. In order to reflect its variation, we introduce a *duration factor* f to describe the relationship between the client's interaction duration and the default duration. Assume that d_i $(i = 1, \ldots, 8)$ in Table 2 are mean default durations for various interactions, and the client with the duration factor f has the mean durations $f * d_i$ $(i = 1, \ldots, 8)$.

4.1.3 Channel Requirements

The channel requirement of multicast VoD service can be divided into two parts: one for regular multicast streams, denoted as C_r, and the other for TVoD services, denoted as C_t. C_r is related to the patching window size w in transition or grace patching, and is expressed as $C_r = \lceil \frac{L}{w} \rceil$.

On the other hand, C_t can be expressed as

$$C_t = C_a + C_i + C_m$$

where C_a is the number of channels requested for admission, C_i is the number of channels for interactions, C_m is the number of channels for merging interaction streams into regular streams.

Thus, the total channel requirements for TVoD service are

$$C = C_r + C_a + C_i + C_m.$$

Because BEP aims to improve the efficiency of TVoD interactions, we only focus on the channel requirements for continuous VCR interactions, i.e., $C_I = C_i + C_m$. In fact, C_i is related to the request rate λ, the interaction duration d_i ($i = 6, 7, 8$) and the CPE buffer size d. C_m is related to the request rate λ, the patching window size w and the CPE buffer size d.

Note that C_I can be reduced significantly by the use of the CPE buffer. We define a factor α ($0 \leq \alpha \leq 1$) as the percentage of interactions that require additional channels. This factor reflects the VCR interaction support with the CPE buffer.

In order to measure the ratio of merging channels to the number of channels for VCR interactions, we also define a merging channel ratio

$$\beta = \frac{C_m}{C_i + C_m}.$$

4.2 Requirement Analysis of Merging Channel

We first introduce the following definitions:

Interaction duration (ID) **:** is the time a user executes an interaction operation, or holds down the interaction button. It is denoted as t.

Interaction effect offset (IEO) **:** means the video offset an interaction makes. This parameter, denoted as *T*, reflects the real effect of an interaction.

Interaction group offset (IGO) **:** is the multicast group offset an interaction makes. This parameter determines the transformation of multicast group and the requirement of interaction channel. It is denoted as G and, in general, $G = T - t$.

After completing an interaction, the client needs to continue using the interaction (patching) channel to join the nearest multicast group. Assuming that the patching window size w is the grouping interval, the channel usage duration for merging is $a = w - mod(G + U, w)$, where $U \sim Uniform[0, w]$ is the group offset of the client before the beginning of interaction.

If $U \sim Uniform[0, 1]$, G is any random variable which is independent of U, then $\{G + U\}$ is also $Uniform[0, 1]$. Thus, without the dynamic merging algorithm (DMA), the mean channel requirement for merging once is:

$$E[a] = \frac{w}{2}.$$

We now evaluate the merging channel requirement of DMA used in the BEP scheme. Given the client n finish the interaction and need to be merged with multicast streams at time t. The desired merging offset is W_n.

Assume that there are r queues available for merging, i.e., the merging queue set $Q = \{q_1, q_2, \ldots, q_r\}$. Let

$$\delta_i = q_i.lifetime - (t - q_i.latime).$$

Let W_i be the offset of queue i and A_n be the event that client n is successfully merged into the existing queues

$$A_n = \bigcup_{i=1}^{r} \{W_i \leq W_n \leq W_i + \delta_i\}.$$

For computational convenience, we assume that $w = 1$ (unit length). We can then find the merging probability of the client n as:

$$E[\mathbf{1}_{A_n}] = P(A_n) = 1 - P(\bigcap_{i=1}^{r} \{W_n \notin (W_i, W_i + \delta_i)\})$$

$$= 1 - \int_0^1 P^r(x \notin [W_i, W_i + \delta_i])dx$$

Since $P(x \in [W_i, W_i + \delta_i]) = x - x^2/2$, we have

$$\begin{aligned} 1 - \int_0^1 P^r(x \notin [W_i, W_i + \delta_i])dx &= 1 - \int_0^1 (1 - x + x^2/2)dx \\ &= 1 - \int_0^1 \sum_{j=0}^{r} \binom{r}{j} (1-x)^j (x^2/2)^{r-j} dx \\ &= 1 - \sum_{j=0}^{r} \binom{r}{j} 2^{j-r} \frac{j!(2r-2j)!}{(2r-j+1)!} \\ &= 1 - \sum_{j=0}^{r} 2^{j-r} \frac{(r+j)!(2r-2j)!}{r!(2r-j+1)!} \end{aligned}$$

where $\mathbf{1}_{A_n}$ is the indicator function of the event A_n. So, $\lim_{r\to\infty} E[\mathbf{1}_{A_n}] = 1$. Thus, the probability of being merged increases as the number of queues increases in the queue.

Using the notation that $a \wedge b = min[a, b]$, we obtain the time (S) saved by merging client n as:

$$S = max_{1\le i\le r}\{(W_i + \delta_i - W_n) \wedge W_i | W_i \le W_n \le W_i + \delta_i\}.$$

We know

$$\begin{aligned} E[S] &\le E[max_{1\le i\le r}(\delta_i \wedge W_n)] \\ &= \int_0^1 P(max_{1\le i\le r}\delta_i \wedge W_n > y)dy \\ &= 1 - \int_0^1 P(max_{1\le i\le r}\delta_i \wedge W_n < y)dy \\ &= 1 - \int_0^1 \int_0^1 P(max_{1\le i\le r}\delta_i \wedge x < y)dxdy \\ &= 1 - \int_0^1 \int_0^1 P^r(\delta_i \wedge x < y)dxdy \\ &= 1 - \int_0^1 [\int_0^y + \int_y^1] P^r(\delta_i \wedge x < y)dxdy \\ &= 1 - \int_0^1 (y + y^r(1-y))dy = \frac{1}{2} - \frac{1}{(r+1)(r+2)}. \end{aligned}$$

Note that

$$P(\delta_i \wedge x < y) = \begin{cases} 1, & if x \le y \\ y, & if x > y \end{cases}$$

Compared to the approaches without DMA, the percentage of being saved channel time can be approximated by

$$\begin{aligned} \rho = \frac{E[S] * E[\mathbf{1}_{A_n}]}{E[a]} &= 2(\frac{1}{2} - \frac{1}{(r+1)(r+2)})E[\mathbf{1}_{A_n}] \\ &= (1 - \frac{2}{(r+1)(r+2)})E[\mathbf{1}_{A_n}]. \end{aligned}$$

Therefore, if λ is large, r is large and $\lim_{\lambda\to\infty} \rho = 1$. This is the expected efficency of channel time saving. If r is small, the saving efficiency is not high.

4.3 The Simulation Results

We want to find the relationships among the multicast strategy, the request rate, the bandwidth requirement, and the CPE buffer size and the patching window. We compare BEP with the SAM protocol [19], and with the SAM protocol improved with the CPE buffer (abbreviated as Buffer-SAM or BSM) [1]. Because the latter two schemes also achieve the same admission performance as that of BEP by transition or grace patching, we focus only on the comparison of their TVoD interactivities.

Table 3. Parameters setting

Parameter	Default	Variation
Client interactive behavior	VI	VI/NVI
Speedup factors K0,K1	10,3	N/A
Speeddown factor K2	2	N/A
Video length (minutes)	90	N/A
The CPE buffer size d (minutes)	5	0-30
Request rate λ (per minute)	5	0-10
Patching window size w (minutes)	5	0-25
Duration factor f	1	0-2.0

We studied the mean bandwidth requirement of a 90-minute video in our simulation. Requests arrive according to a Poisson process with rate ranging from 0 to 10 per minute. The patching window size is varied from 0 to 25 minutes, and the CPE buffer size is changed from 0 to 30 minutes. Two types of interactive behaviors, VI and NVI, are simulated. The results are collected from 10-hour simulations. The simulation parameters are summarized in Table 3.

4.3.1 The channel requirement ratio

We first consider the effect of the CPE buffer. Some interactions don't need any additional channels if they can be supported by the CPE buffer. Figure 6 shows that the greater the CPE buffer, the lower the ratio α becomes. In other words, the blocking rate of interaction requests will decrease with the increase of CPE buffer size even if no interaction channels are available.

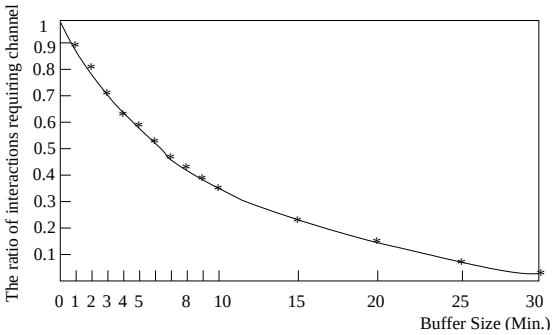

Figure 6: The effect of the CPE buffer ($f = 1, \lambda$=5)

We also examined the ratio of merging channels for VCR interactions. Figure 7 shows the relationship between the ratio β of merging channels and the duration factor f. It indicates that a high proportion of

channels are merging channels for VCR interactions. BEP improves the efficiency of merging so that its ratio β is smaller than that of SAM.

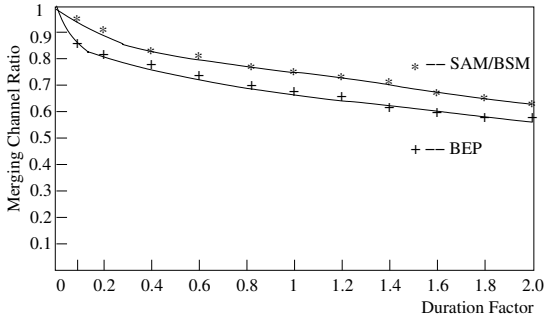

Figure 7: The duration factor and merging channel ratio (λ=5, w=d=5 min.)

4.3.2 The merging channel requirements

We also evaluated the requirement of merging channels for VCR interactions under three different schemes in NVI/VI mode. Figure 8 shows the merging channel requirement while varying the request rate. The merging channel requirement for SAM is significantly greater than that for both BSM and BEP. When the request rate is low, there is not much difference between the merging channel requirements for BSM and BEP. The higher the request rate, the bigger their difference. Figure 9 indicates that the patching window will greatly affect the merging channel requirement. When the patching window size is small, there is not much difference between the merging channel requirements for BSM and BEP. When the patching window size is large, BEP significantly outperforms BSM. The simulation results approximate the theoretical results closely. For example, when f=1, w=d=5 min., and the request rate λ is 2, if the client interactive mode is NVI, C_m are about 3.82 (BSM) and 3.6 (BEP), respectively; if the client interactive mode is VI, C_m are about 8.36 (BSM) and 7.72 (BEP), respectively. In addition, C_a is about 4.5, and C_i is about 1.64 (NVI) or 3.58 (VI) in this case. Totally, the number of patching channels reserved for true VoD service of a video in the assumed case, except the number of the multicast channels, are 9.74 (BEP, NVI), 9.96 (BSM, NVI), 15.8 (BEP, VI), 16.44 (BSM, VI), respectively. Note that BEP and BSM can work only if the patching window size is less than, or equal to, the CPE buffer size.

5 Conclusion

A TVoD system should offer clients full support for interactive VCR functions. However, server storage and network I/O throughput are known to be a serious bottleneck in realizing a TVoD system. Multicast is shown to be a good remedy for this problem. In this paper, we evaluated the existing multicast TVoD schemes, and then proposed a new TVoD approach called the *Best-Effort Patching* (BEP). This scheme supports both continuous VCR interactions and zero-delay admission of requests. Moreover, we proposed a novel dynamic merging algorithm to improve the efficiency of merging interactions and regular streams. Our simulation results and theoretical analysis have shown that BEP can achieve significantly better performance than the conventional TVoD protocols, especially for popular videos. BEP supports TVoD service with less bandwidth requirement.

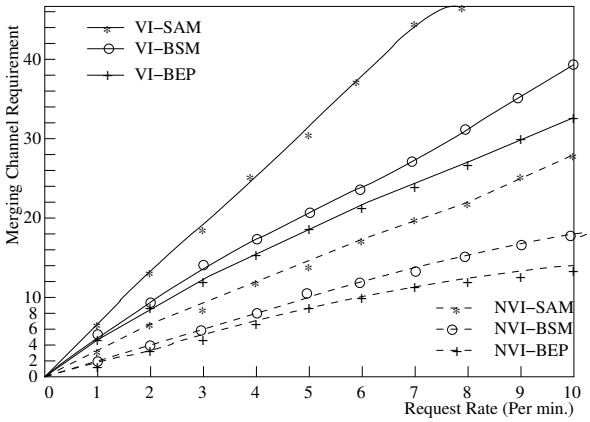

Figure 8: Effect of request rate λ (f=1,w=d=5 min.)

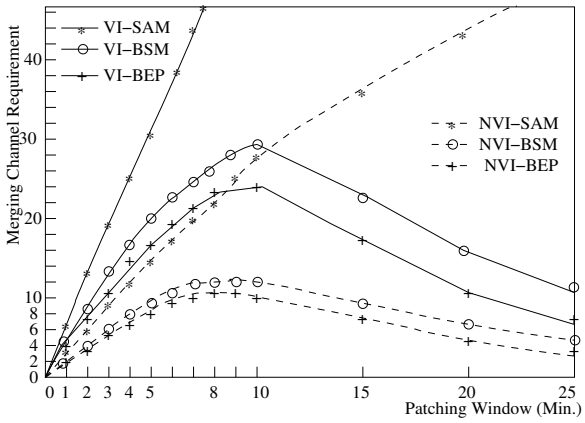

Figure 9: Effect of patching window w ($\lambda = 5, f = 1$, If $w \leq 5$ min., then $d = 5$ min.; if $w > 5$ min. then $d = w$)

References

[1] Emmanuel L. Abram-Profeta and Kang G. Shin, "Providing unrestricted VCR capability in multicast video-on-demand systems," *Proc. of IEEE Int'l. Conference on Multimedia Computing and Systems'98*, June-July 1998.

[2] Emmanuel L. Abram-Profeta, "Support for service scalability in video-on-demand end systems," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, MI, May 1998.

[3] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," *Proc. IEEE Int'l. Conf. on Multimedia Computing and Systems'96*, Hiroshima, Japan, June 1996.

[4] 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, Aug. 1996.

[5] Michael K. Bradshaw, Bing Wang, Subhabrata Sen, Lixin Gao, Jim Kurose, Prashant Shenoy, and Don Towsley "Periodic Broadcast and Patching Services - Implementation, Measurement, and Analysis in an Internet Streaming Video Testbed," *Proc. ACM Multimedia'2001*, Ottawa, Canada.

[6] Ying Cai and K.A. Hua, "Optimizing patching performance," *Proc. of SPIE's Conference on Multimedia Computing and Networking* (MMCN'99),pp.204-215, San Jose,Jan. 1999.

[7] Ying Cai and K.A. Hua, "An efficient bandwidth-sharing technique for true video on demand systems," *Proc. ACM Multimedia'99*, pp.211-214, Orlando, Nov. 1999.

[8] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," *Proc. of ACM Multimedia'94*, pp. 15–23, San Francisco, Oct. 1994.

[9] J.K. Dey-Sircar, J.D. Salehi, J.F. Kurose, and D. Towsley, "Providing VCR capabilities in large-scale video servers," *Proc. of ACM Multimedia'94*, pp. 25–32, San Francisco, Oct. 1994.

[10] D. L. Eager and M. K. Vernon, "Dynamic skyscraper broadcasts for Video-on-Demand," *Proc. Multimedia Information System (MIS'98)*, pp.18-32, Istanbul, Turkey, Sept. 1998.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," *Proc. ACM multimedia'99*,pp.199-202, Orlando, Nov. 1999.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Bandwidth skimming: A technique for cost-effective video-on-demand," *Proc. MMCN'2000*, San Jose, Jan. 2000.

[13] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery," *IEEE Trans. on Knowledge and Data Engineering*, 2001, 13(5), pp.742-757.

[14] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," *Proc. NOSSDAV*, Cambridge, UK, July 1998.

[15] L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," *Proc. IEEE Multimedia Computing and Systems*, pp.117-121, Florence, Italy, June 1999.

[16] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," *Proc. of ACM Multimedia'98*, pp. 191–200, Bristol, UK, Sept. 1998.

[17] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," *Proc. ACM SIGCOMM'97*, pp.89-100, Cannes, France, Sept. 1997

[18] L. Golubchik, J. Lui, and R. Muntz, "Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers," *Multimedia Systems*, vol. 4, no. 3, 1996.

[19] V.O.K. Li, W. Liao, X. Qiu, and E.W.M. Wong, "Performance model of interactive video-on-demand systems," *IEEE JSAC*, vol. 14, no. 6, pp. 1099–1109, Aug. 1996.

[20] W. Liao and V.O.K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia*, Oct.-Dec.1997, pp. 51–62.

[21] T.D.C. Little and Dinesh Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 1, no. 3,1994, pp. 14–24.

[22] Huadong Ma and Kang G. Shin, "A new scheduling scheme for multicast true VoD service," *Lecture Notes in Computer Science (Proc. PCM2001)*, Vol. 2195, pp.708-715, Springer, Oct. 2001.

[23] Huadong Ma and Kang G. Shin, "Multicast video-on-demand services," *ACM Computer Communication Review*, ACM Press, 2002,32(1):31-43.

[24] Huadong Ma and Kang G. Shin, "Hybrid broadcast for video-on-demand service," *Journal of Computer Science and Technology*, Science Press, 2002, 17(4):397-410.

[25] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," *Proc. NOSSDAV'99*, Basking Ridge, NJ, June 1999.

[26] H. Tan, D. L. Eager, M. K. Vernon, and H. Guo, "Quality of service evaluations of multicast streaming protocols", *Proc. ACM SIGMETRICS 2002*, Marina del Rey, CA, June 2002.

[27] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, 4(4):197–208, August 1996.

Huadong Ma received the B.S. degree in Mathematics from Henan Normal University in 1984, the M.S. degree in Computer Science from Shenyang Institute of Computing Technology, Chinese Academy of Science (CAS) in 1990 and the Ph.D. degree in Computer Science from Institute of Computing Technology, CAS, in 1995.

He is a Professor with the School of Computer Science & Technology, Beijing University of Posts and Telecommunications, China. He visited UNU/IIST as research fellow in 1998 and 1999, respectively. From 1999 to 2000, he held a visiting position in the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on multimedia, networking, e-commerce and computer graphics, and he has published over 70 papers and 3 books on these fields. He is member of IEEE and ACM.

Kang G. Shin received the B.S. degree in Electronics Engineering from Seoul National University, Korea, in 1970, and both the M.S. and Ph.D degrees in Electrical Engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively.

He is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 49 PhD theses, and authored/coauthored over 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. Dr. Shin is Fellow of IEEE and ACM, and member of the Korean Academy of Engineering.

Weibiao Wu received the Ph.D degree in statistics from the University of Michigan, Ann Arbor in 2001. He is currently an Assistant professor of statistics at the University of Chicago. His research interests include probabilistic network modelling and simulation, data-base compression, asymptotic theory and statistical inference of stochastic processes.