# A New Scheduling Scheme for Multicast True VoD Service *

Huadong Ma[1] and Kang G. Shin[2]

[1] College of Computer Science & Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
mhd@bupt.edu.cn
[2] Real-Time Computing Lab., EECS Depart.,
The University of Michigan, Ann Arbor, MI 48019-2122, USA
kgshin@eecs.umich.edu

**Abstract.** Multicast Video-on-Demand (VoD) has excellent performance, but it is very difficult to equip such a system with full support for interactive VCR functions. In this paper, we propose a new scheme, called the *Best-Effort Patching* (BEP), that offers a TVoD service in terms of both request admission and VCR interaction for multicast VoD system. Moreover, by using a novel dynamic merging algorithm, BEP significantly improves the efficiency of TVoD interactivity, especially for popular videos. Our extensive simulation results show that BEP outperforms the conventional TVoD interaction protocols.

## 1 Introduction

A VoD service allows remote clients to play back any video from a large collection of videos stored at one or more video servers 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. TVoD service supports all of the control functions such as Play/Resume, Stop/Pause/Abort, Fast Forward/Rewind, Fast Search/Reverse Search and Slow Motion, and is an ideal service for consumers. 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. One efficient solution to these problems is to use multicast. Multicast VoD has excellent scalability and cost/performance efficiency.

There are several approaches for multicasting VoD service. One approach is to multicast each popular video at fixed intervals. In order to eliminate service latency, patching [8] was proposed to enable an existing multicast to serve new additional clients, but the conventional patching is suitable only for TVoD admission control. The other approach is to use a fixed number of multicast

* The work reported in this paper is partly supported by the US NSF under Grant EIA-9806280 and the NNSF of China under Grant 69873006. The work was done when Huadong Ma visited EECS Depart., the University of Michigan.

channels to periodically broadcast video objects to a group of subscribers [2, 9, 12]. This periodic broadcast is efficient in transmitting popular videos from one server to many clients, but it is difficult to support VCR interactivity.

In order to provide TVoD service, we propose a new patching scheme, called the *Best-Effort Patching* (BEP). Like the conventional patching, BEP multicasts a popular video via regular channels at fixed intervals. Requests between two consecutive regular channels will share the latest regular stream by patching the missed leading segment. Moreover, patching is used to support the user interaction in BEP. Once a client interaction exceeds the capability of his CPE buffer, BEP dispatches a patching channel to support the client's interaction and its merging into the nearest multicast channel. Further, BEP uses a novel dynamic merging scheme, thus offering TVoD service efficiently.

## 2 Background

To eliminate the service latency, patching was proposed in [8] by enabling each multicast session to dynamically add new requests. An important objective of patching is to increase the number of requests each channel can serve per time unit, thereby reducing the per-customer system cost. A new service request can exploit an existing multicast by buffering the video stream from the multicast while playing a new catch-up stream (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 can join the original multicast. Allowing clients to dynamically join an existing multicast 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. The time period during which patching must be used, is referred to as the *patching window* [4]. Two simple approaches to setting the patching window are discussed in [8]: greedy patching and grace patching. An improved patching technique, called as the *transition patching* [5], has better performance without requiring any extra download bandwidth at the client site. Other optimal patching schemes were presented in [6, 11].

In order to implement the interactivity of multicast VoD service, some efficient schemes have been proposed. For example, the SAM protocol [10] offers an efficient way for TVoD interactions, but it requires many I-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 [3], focused on interactions without picture. In this paper, we present an efficient approach to the implementation of the continuous TVoD interactions.

## 3 Best-Effort Patching for TVoD Interactivity

### 3.1 Basic Idea

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 (see Fig. 1). 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 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 continuously be supported if $t \leq \frac{d_r}{SP-1}$ (e.g. A is the desired play point). A similar observation can be made for a backward interaction.

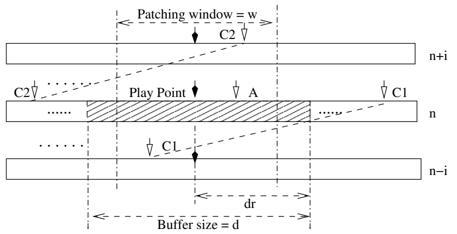

**Fig. 1.** The CPE buffer and VCR actions

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 take place. The interaction can't always join an existing channel immediately because there isn't always a channel with the desired playback time.

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 interaction time exceeds the capacity of the CPE buffer, 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, such as Fast Forward, Rewind, Pause, Stop, 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 closest multicast channel simultaneously. The video from the patching channel is played back immediately, and the video from the closest multicast channel is buffered in the CPE buffer. For merging, the patching channel is required only for the displacement between the play point of the closest channel and the desired play point, and then the patching channel is released. This way, the customer seamlessly joins an existing multicast group. It doesn't incur any additional CPE cost because it just makes use of the CPE buffer required by the conventional patching.

We can further illustrate BEP by Fig. 2. Once the interaction time 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 [10], while it is used as a patching channel in the merging phase. For the forward interaction shown in Fig. 2(a), such as Fast Search, the interaction takes $t$ units of time, and the customer's desired play point is A. P0 is the original play point. 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 length of the patching channel for the merging is $|P(n-i-1) - A|$. For the backward interaction shown in Fig. 2(b), such as Reverse Search, one can give a similar illustration.

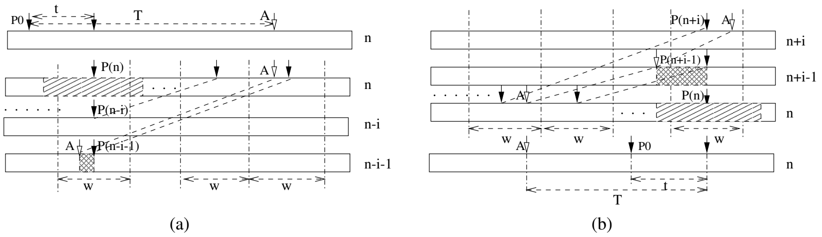

Fig. 2. Forward/Backward interactions

### 3.2 The Dynamic Merging Algorithm

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 Fig. 3. 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 downloads the video segment from Stream A and B simultaneously, and begins downloading of the video from the stream of multicast group $n$ after using up Stream B. Such merging of patching channels can go on, and the merged clients can be recorded in a *merging queue*.

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 initiating this queue, and the lifetime of $q$ is initially set to the offset of its head and will possibly 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,lefetime*; /*the offset and lifetime of queue*/
*int latime*; /*joining time of the latest client*/
}

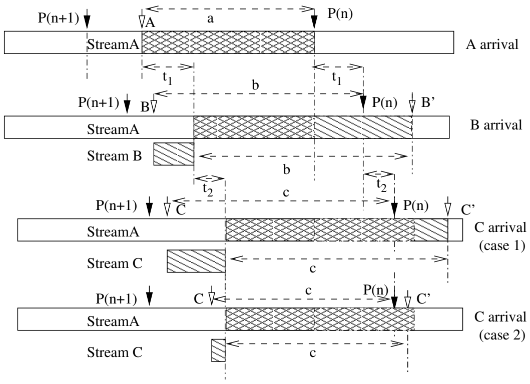

**Fig. 3.** Dynamic merging

When a new client C wants to join this 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 Fig. 3.

**Case 1:** $c > q.lifetime - t_2$, meaning that the lifetime of the queue $q$ is not 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 Stream 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 *q.lifetime* is enough to merge C, so the lifetime of 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* is unchanged. The client downloads the video segments from Stream 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 will be merged into the queue which saves the maximum usage time of channel.

The dynamic merging algorithm 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*/
q0 = max_{q∈Q}{min{q.offset + q.lifetime − c − (t − q.latime), q.offset}
        |q.offset ≤ c < q.offset + q.lifetime − (t − q.latime)}
if (q0 = null) { /* generate a new merging queue*/
  Genqueue(q0); /*generate a new queue*/
  q0.head is set to C; q0.offset = c;
  q0.lifetime = c; q0.latime = t; }
else { /* merging C into q0*/
  delta = t − q0.latime; q0.latime = t; lifetime(C) = c − q0.offset;
  /* the lifetime of patching stream C is changed*/
  if (c > q0.lifetime − delta) q0.lifetime = c;
  }
```

### 3.3 Discussion

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. Thus, patching channels are used to patch all of the segments that can't be provided by regular multicast channels for TVoD service, whereas 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. 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.

## 4 Performance Evaluation

### 4.1 Client's Interaction Model

We use the interaction model proposed in [1] to evaluate our approach. 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 system randomly transits to other interactive states or remains at Play state according to the behavior distribution. As shown in Fig. 4, transition probabilities $P_i$ $(i = 0, \ldots, 9)$ are assigned to a set of states corresponding to different VCR actions. For tractability, we divide customers into two types: *Very Interactive (VI)* or *Not Very Interactive (NVI)*. Our simulation assumes $P_8 = 0$, and $P_9 = 0.5$. The other transition possibilities are summarized as Table 1.

Assume that BEP serves each state for an exponentially-distributed period of time, and $d_i$ $(i = 0, 1, 2, \ldots, 8)$ 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 $K_2 = 2$.

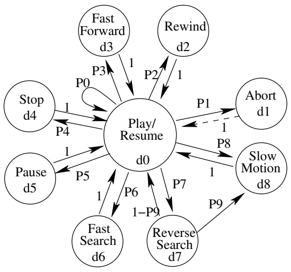


**Fig. 4.** VCR interactive model

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 |

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 |

### 4.2 The Simulation Results

We compare BEP with the SAM protocol [10], and the SAM protocol improved by the CPE buffer (abbreviated as 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, especially for the merging performance. For a video of 90 minutes, requests arrive according to a Poisson process with rate $\lambda$ ranging from 0 to 10 per minute. The patching window size $w$ is varied from 1 to 25 minutes, and the CPE buffer size $d$ is ranging from 0 to 30 minutes. Two types of interactive behaviors, VI and NVI, are simulated. The results are collected from 10-hour simulations.

Fig. 5(a) 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 a big difference between the merging channel requirements for BSM and BEP. The higher the request rate is, the bigger their difference is. Fig. 5(b) indicates that the patching window will greatly affect the merging channel requirement. When the patching window size is small, there is not a big difference between the merging channel requirements for BSM and BEP. When the patching window size is large, BEP significantly outperforms BSM. 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

Multicast is shown to be a good remedy for improving the performance of VoD system. In this paper, we proposed a new TVoD approach called the *Best-Effort*

*Patching*. This scheme supports both continuous VCR interactions and zero-delay requests admission. Moreover, a novel dynamic merging algorithm improves the efficiency of merging interaction and regular streams. Our simulation results indicate that BEP can achieve significantly better performance than the conventional TVoD protocols, especially for popular videos. BEP supports TVoD service with less bandwidth requirement.

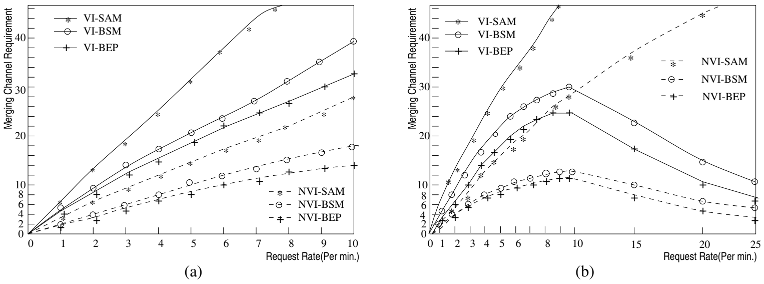


**Fig. 5.** (a) Effect of request rate $\lambda$ ($w = d = 5$ min.); (b) Effect of patching window $w$ ($\lambda = 5$, If $w \leq 5$ min. then $d = 5$ min., if $w > 5$ min. then $d = w$)

## References


1. E. L. Abram-Profeta and Kang G. Shin, Providing unrestricted VCR capability in multicast video-on-demand systems, Proc. of IEEE ICMCS'98, June-July 1998.
2. C.C. Aggarwal, J.L. Wolf, and P.S. Yu, A permutation-based pyramid broadcasting scheme for video-on-demand systems, Proc. of IEEE ICMCS'96, Hiroshima, Japan, June 1996.
3. K.C. Almeroth and M.H. Ammar, The use of multicast delivery to provide a scalable and interactive video-on-demand service, IEEE JSAC, Vol.14, No.6, pp. 1110–1122, Aug.1996.
4. Ying Cai and K.A. Hua, Optimizing patching performance, Proc. of SPIE MMCN'99, San Jose, January 1999.
5. Ying Cai and K.A. Hua, An efficient bandwidth-sharing technique for true video on demand systems, Proc. of ACM Multimedia'99, Orlando, Nov.1999.
6. D.L. Eager, M.K. Vernon, and J. Zahorjan, Optimal and efficient merging schedules for video-on-demand servers, Proc. of ACM multimedia'99, Orlando, Nov.1999.
7. L. Gao and D. Towsley, Supplying instantaneous video-on-demand services using controlled multicast, Proc. of IEEE ICMCS'99, Florence, Italy, June 1999.
8. 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, U.K., Sept. 1998.
9. K.A. Hua and S. Sheu, Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems, Proc. of ACM SIGCOMM'97, Sept. 1997.
10. W. Liao and V.O.K. Li, The split and merge protocol forinteractive video-on-demand, IEEE Multimedia, Oct.-Dec. 1997, Pages 51-62.
11. S. Sen, L. Gao, J. Rexford, D. Towsley, Optimal patching schemes for efficient multimedia streaming, Proc. of IEEE NOSSDAV'99, Basking Ridge, NJ, June 1999.
12. S. Viswanathan and T. Imielinski, Pyramid broadcasting for video on demand service, Proc. of SPIE MMCN'95, Volume 2417, Pages 66–77, San Jose, 1995.