IEICE TRANS. COMMUN., VOL.E93–B, NO.12 DECEMBER 2010

LETTER

Patching with a Variable Segment VOD Scheduling**

Chan-Gun LEE†, **Yong-Jin JI**†, *Nonmembers*, **Ho-Hyun PARK**†*a), *Member*, **Jae-Hwa PARK**†, *and* **Sungrae CHO**†, *Nonmembers*

SUMMARY The patching technique has been used for reducing initial waiting time in VOD services. Traditionally the technique has been applied to fixed segment NVOD scheduling. However, variable segment NVOD scheduling is known to have a better server bandwidth and less initial waiting time. In this paper, we propose a new scheduling algorithm for a true VOD service by incorporating the patching technique into variable segment NVOD scheduling. Our algorithm provides jitter-free playback while minimizing the use of the patching bandwidth. We present the proof of the correctness of our algorithm.

key words: *VOD, broadcasting, segment, patching, scheduling*

1. Introduction

RVOD (Real VOD) and NVOD (Near VOD) are the two major types of VOD (Video On Demand) currently served. Almost every broadcast company supports RVOD on its website. This is because RVOD provides the most convenience with the users; it enables them to choose and watch their favorite videos with a single click at their convenient times.

In the presence of a large number of users, there can be congestion in servers and network, transmission delays, and dropping because every user occupies each transmission channel from the server to his/her terminal. To address this scalability problem of RVOD, NVOD can be applied. NVOD is currently served in many hotels, cable broadcast, and satellite broadcast. NVOD uses broadcasting or multicasting which dramatically reduces the cost of the servers by periodically transmitting the videos on multiple channels and a group of users connects to one of the channels.

There are two ways how to partition the whole video into server channels; one is to partition the video into fixed size segments and the other creates variable size segments. Staggered Broadcasting (SB) [1] is a typical method of the fixed segment NVOD algorithm. But this method inherently suffers from initial waiting time. In order to provide short waiting time, a considerable amount of network bandwidth is needed. The patching technique [2] is proposed to nullify the waiting time. The patching channels immediately serve the users' requests arriving at random times by unicast, thus enable the true VOD service meaning no initial waiting time.

On the other hand, several approaches based on variable segment NVOD algorithms, e.g. Pyramid Broadcasting [3], Skyscraper Broadcasting, Harmonic Broadcasting, and Fibonacci Broadcasting [4], [5] were proposed. There have been many observations showing that variable segment NVODs are better than fixed segment NVODs since the former requires less initial waiting time and has better channel efficiency [3]–[5].

In spite of the advantages of variable segment NVODs, little work was performed on applying patching to variable segment NVODs. While a recent work [6] attempted to apply the patching on a variable segment NVOD, it cannot provide the true VOD service. It was not for removing initial waiting time completely but for compensating regular channel shortages in the case of intensive user requests.

In this paper, we propose a new scheduling algorithm for a true VOD service by incorporating the patching technique into a variable segment NVOD scheduling. At first glance, applying the patching to a variable segment algorithm seems trivial. However, it turns out that a naive approach incurs jitters or significant patching bandwidth. The main challenge is that we need to partition the video into variable segments with appropriate sizes so that the algorithm still provides jitter-free playback of the video while consuming minimal bandwidth of the patching channel.

2. Problem Definition and Worst Case Analysis

In variable segment NVODs, a video is partitioned into multiple segments and the segments are delivered via the regular channels of the VOD server. We shall use S and $|S|$ to denote the entire video stream and its size in second. S is partitioned into the segments $S_0, S_1, \ldots,$ and S_{n-1}. n is the number of segments of the video. A VOD server periodically delivers these segments via the regular channels C_0, $C_1, \ldots,$ and C_{n-1}.

Among many variable segment NVODs, we will consider only the Generalized Fibonacci Broadcasting (GFB) [5] that is known to be a state-of-the-art periodic broadcasting algorithm. In GFB, a client can download maximum m channels concurrently. The segment partitioning scheme of GFB is as follows:

$$|S_k| = \begin{cases} 2^k|S_0| & : \text{ if } 1 \le k \le m-1 \\ \sum_{i=1}^{m} |S_{k-i}| & : \text{ if } k \ge m \end{cases} \quad (1)$$

Manuscript received June 3, 2010.
Manuscript revised August 12, 2010.

†The authors are with Chung-Ang University, Seoul, 156-756, Korea.
*The corresponding author.

**This research was supported by the Chung-Ang University Research Scholarship Grants in 2008.

a) E-mail: hohyun@cau.ac.kr

DOI: 10.1587/transcom.E93.B.3660

We assume that a client can download and buffer the first m segments without waiting for finding the boundaries of the segments as soon as the user requests the video. However, the play of a segment can start only after the client finds the boundary of the segment. In the original GFB [5], this is not true; in such a case, it starts downloading a segment as soon as it finds the boundary of the segment.

Optionally, the system can use patching channel C_p in order to eliminate the initial waiting time and enable jitter-free play. Without using the patching channel, variable segment NVOD algorithms may cause initial waiting-time to a user unless the request time perfectly matches with the boundary of the segment S_0.

As shown in Fig. 1, assume that a user requests a VOD service at time t and the nearest past boundary of S_0, which is not larger than t, locates at time t_0. The initial waiting time for traditional NVOD services without patching is $|S_0| - (t - t_0)$.

For each segment S_i, $D(S_i)$ represents the time when the boundary of the segment is downloaded via a regular channel. Therefore, the beginning part of the segment S_i is playable at $D(S_i)$. In Fig. 1, $D(S_0)$ is $t_0 + |S_0|$, $D(S_1)$ is $t_0 + |S_1|$, $D(S_2)$ is $t_0 + |S_2|$, $D(S_3)$ is $t_0 + |S_3|$, and so forth. The playtime of S_i, denoted by $P(S_i)$, is defined as the time when the segment should start playing in order to enable the jitter-free playback of the video and no-initial waiting time. For example, $P(S_0)$ is t, $P(S_1)$ is $t + |S_0|$, $P(S_2)$ is $t + |S_0| + |S_1|$, $P(S_3)$ is $t + |S_0| + |S_1| + |S_2|$, and so forth, as shown in Fig. 1.

2.1 Naive Approach with One Patching Channel

We argue that applying the patching technique to a variable segment NVOD algorithm is not trivial. The main challenge is that we need to partition the video into variable segments with appropriate sizes while achieving the minimum use of the patching channels and jitter-free play.

If we simply apply patching to a variable segment NVOD just with one patching channel, jitters can arise. Figure 1 shows an example of the jitter when the naive patching is applied to GFB where $m = 3$ and $n = 4$. The video segment sizes of GFB for $m = 3$ are $|S_0|$, $2|S_0|$, $4|S_0|$, $7|S_0|$, $13|S_0|$, and so forth.

When a user requests a video at time t, the slashed area of S_0 at C_0 is the missing portion and can be compensated thru C_p. The other portion of S_0 (shown as dark shaded area) will be downloaded thru C_0. At the same time, S_1 and S_2 will be downloaded thru C_1 and C_2 channels. As soon as the downloading of S_0 completes, S_3 starts to be downloaded thru C_3. For jitter-free playback, the beginning part ($D(S_1)$ and $D(S_2)$) of S_1 and S_2 should be downloaded before the playout time ($P(S_1)$ and $P(S_2)$) of the segments. However, in Fig. 1, $D(S_1)$ and $D(S_2)$ are later than $P(S_1)$ and $P(S_2)$, respectively.

2.2 Naive Approach with m Patching Channels

Since the video segments in the first m channels are downloaded concurrently, we need to patch all missing portions in the first m channels for jitter-free play. Figure 2 illustrates such a scenario. In this scenario, we can play jitter-free because $D(S_i) \leq P(S_i)$ for $i < m$, i.e., all the segments after the patched segments. However, this approach needs a lot of patching bandwidths (e.g., almost $3|S_0|$ in Fig. 2).

The following analysis proves that under naive scheduling, the maximum patching bandwidth may increase up to $m|S_0|$. Note that the patching channel is served by unicast for each user. The heavy use of the patching channel may hinder making the system scalable.

Lemma 1: In order to enable jitter-free play, $P(S_i)$ is computed as follows:
$P(S_i) = t + |S_i| - |S_0|$ if $1 \leq i \leq m - 1$
$P(S_i) = t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r| - |S_0|$ if $i \geq m$
where $r = i \bmod m$ and m is the number of concurrent channels in a client.
Proof:
Case 1: Consider S_i where $1 \leq i \leq m - 1$.

$$\begin{aligned} P(S_i) &= t + |S_{i-1}| + |S_{i-2}| + \ldots + |S_1| + |S_0| \\ &= t + (2^{i-1} + 2^{i-2} + \ldots + 2^1 + 2^0)|S_0| \\ &= t + (2^i - 1)|S_0| \end{aligned}$$

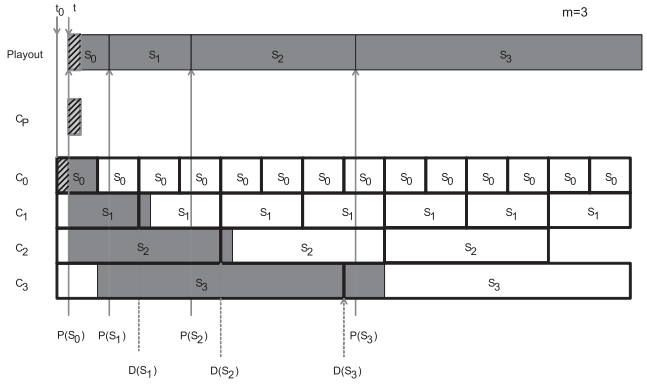

Fig. 1 Naive combining one patching channel into regular channels.

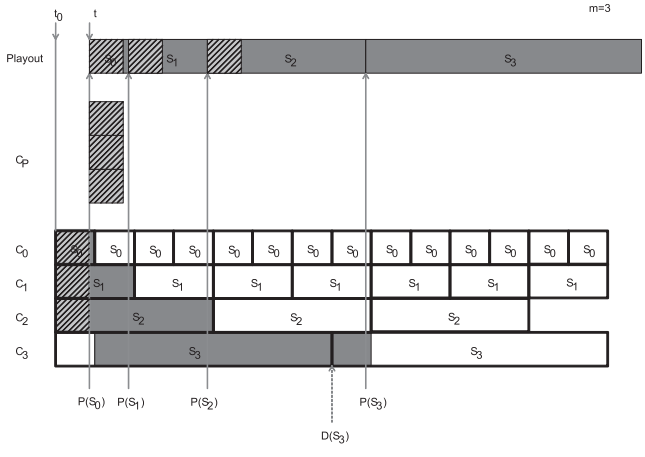

Fig. 2 Naive combining m patching channels into regular channels.

$$= t + |S_i| - |S_0|$$

Case 2: Consider S_i where $i \geq m$

$$\begin{aligned} P(S_i) &= t + (|S_{i-1}| + |S_{i-2}| + \ldots + |S_{i-m}|) \\ &\quad + (|S_{i-(m+1)}| + \ldots + |S_{i-2m}|) + \ldots \\ &\quad + (|S_{r-1}| + \ldots + |S_1|) + |S_0| \\ &= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots \\ &\quad + (2^{r-1} + 2^{r-2} + \ldots + 2^1 + 2^0)|S_0| \\ &= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + (2^r - 1)|S_0| \\ &= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r| - |S_0| \end{aligned}$$

□

Theorem 1: Suppose a configuration where all the periods of m segments, $S_0, S_1, \ldots, S_{m-1}$ start at the same time t_0 as shown in Fig. 2. In addition, $\delta = t - t_0 \approx |S_0|$, i.e., the request is made almost at the end of the period of S_0. With respect to the patching channel usage, this configuration is the worst case. The maximum use of the patching channel for enabling jitter-free play is $m\delta \approx m|S_0|$.

Proof:

Case 1: Consider S_i where $0 \leq i \leq m-1$. In this case, $D(S_i) = t_0 + |S_i|$, i.e., the boundaries of the segments are reached at $t_0 + |S_i|$. By Lemma 1, $P(S_i) = t + |S_i| - |S_0| = t_0 + \delta + |S_i| - |S_0|$. Hence, for $i < m$, $D(S_i) > P(S_i)$ holds since $0 < \delta < |S_0|$. This means that jitters can arise. Therefore, we need patching for all $i < m$ segments and the maximum usage of the patching channels is $m\delta \approx m|S_0|$.

Case 2: Consider S_i where $i \geq m$. Let us derive the download completion time for each segment. Let $r = i \bmod m$. Note that S_i, S_{i-m}, S_{i-2m}, $\ldots$, and S_r are served by the same client channel. Hence, the download of S_i completes at $t_0 + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r|$. In this case, the boundaries of all segments aligned with the boundaries of S_0; i.e., the boundaries of all segments start at multiples of $|S_0|$. This means that the boundary of S_i is always reachable by the download completion time minus $|S_0|$ even in the worst case. Hence, $D(S_i) \leq t_0 + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r| - |S_0|$. By Lemma 1, $P(S_i) = t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r| - |S_0|$. Since $t_0 \leq t$, $D(S_i) \leq P(S_i)$ holds, which means that the patching is not needed for this case.

By combining the cases, we derive that the maximum patching bandwidth may increase up to $m\delta \approx m|S_0|$. □

3. Our Approach

In this section, we design a new scheduling algorithm by applying the patching to a variable segment NVOD algorithm. The design goals of the algorithm are the followings: (1) It should require no initial waiting time and jitter-free playback for the users requesting the VOD service at random times. (2) The use of the patching bandwidth should be minimized.

Clearly, the lower bound of the maximum usage of patch channel is $|S_0|$. We propose that we can reduce the patch channel usage to $|S_0|$ by modifying the GFB segmentation slightly. The core idea is to half the size of the segments $S_1, S_2, \ldots, S_{m-1}$ where m is the number of client channels. For the remaining segments, we retain the same policy as the previous algorithm. More precisely, the following assignment scheme is used.

$$|S_k| = \begin{cases} |S_0| & : \text{if } k = 0 \text{ or } k = 1 \\ 2^{k-1}|S_0| & : \text{if } 2 \leq k \leq m-1 \\ \sum_{i=1}^{m} |S_{k-i}| & : \text{if } k \geq m \end{cases} \quad (2)$$

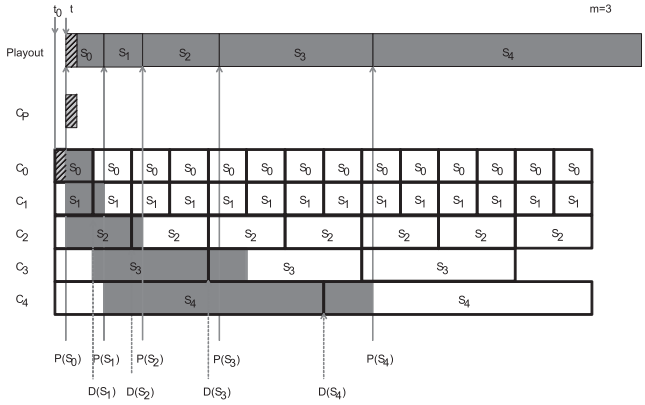

Fig. 3 Temporal-bandwidth map of our approach.

Figure 3 shows a temporal-bandwidth map of our scheme.

In the following, we present proofs that our algorithm provides jitter-free playback for entire stream S and the user experiences no initial waiting time.

Lemma 2: In order to enable jitter-free play in our algorithm, $P(S_i)$ is computed as follows:

$P(S_i) = t + |S_i|$ if $1 \leq i \leq m-1$

$P(S_i) = t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r|$ if $i \geq m$

where t is the request time, $r = i \bmod m$, and m is the number of concurrent channels in a client.

Proof:

Case 1: Consider S_i where $1 \leq i \leq m-1$.

$$\begin{aligned} P(Si) &= t + |S_{i-1}| + |S_{i-2}| + \ldots + |S_1| + |S_0| \\ &= t + (2^{i-2} + 2^{i-3} + \ldots + 2^1 + 2^0)|S_0| + |S_0| \\ &= t + (2^{i-1} - 1)|S_0| + |S_0| \\ &= t + |S_i| \end{aligned}$$

Case 2: Consider S_i where $i \geq m$.

$$\begin{aligned} P(S_i) &= t + (|S_{i-1}| + |S_{i-2}| + \ldots + |S_{i-m}|) \\ &\quad + (|S_{i-(m+1)}| + \ldots + |S_{i-2m}|) + \ldots \\ &\quad + (|S_{r-1}| + \ldots + |S_1|) + |S_0| \\ &= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots \\ &\quad + (2^{r-2} + 2^{r-3} + \ldots + 2^1 + 2^0)|S_0| + |S_0| \\ &= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots \end{aligned}$$

$$+ (2^{r-1} - 1)|S_0| + |S_0|$$
$$= t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r|$$

□

Theorem 2: The playback of S_1 to S_{n-1} is jitter-free.
Proof:
Case 1: Consider S_i where $1 \leq i \leq m-1$.
$P(S_i) = t + |S_i|$ by Lemma 2. The completion time of downloading S_i is $t+|S_i|$ for $i \leq m-1$ since they start downloading at t via independent concurrent channels. $D(S_i) \leq$ the completion time of downloading $S_i = t + |S_i| \leq P(S_i) = t + |S_i|$.
Case 2: Consider S_i where $i \geq m$.
$P(S_i) = t+|S_i|+|S_{i-m}|+|S_{i-2m}|+\ldots+|S_{r+m}|+|S_r|$ by Lemma 2. Let $r = i \bmod m$. The download of S_i is completed by $t+|S_i|+|S_{i-m}|+|S_{i-2m}|+\ldots+|S_r|$. Hence, $D(S_i) \leq$ the completion time of downloading $S_i = t + |S_i| + |S_{i-m}| + |S_{i-2m}| + \ldots + |S_r| = P(S_i)$. Therefore, $D(S_i) \leq P(S_i)$.
In both cases, $D(S_i) \leq P(S_i)$ holds. Therefore, the playback of S_1 to S_{n-1} is jitter-free. □

Theorem 3: The proposed algorithm provides jitter-free playback and no initial waiting time. It requires only one patching channel and the maximum usage of the patch channel is $|S_0|$.
Proof:
The playback of S_0 is jitter-free with no-waiting time due to the patching technique. The use amount of the first regular channel S_0 and the patch channel is $|S_0| - \delta$ and δ respectively. The playback of $S_1, S_2, \ldots, S_{n-1}$ is also jitter-free as shown in Theorem 2. Note that the proof in Theorem 2 was done by showing that the boundaries of the segments S_i are downloaded until $P(S_i)$ by using only regular channels; i.e., we did not use the patching channel at all to enable the jitter-free play for S_1 to S_{n-1}. Therefore, the algorithm requires only one patching channel and the maximum patching channel usage is $|S_0|$ where $\delta \approx |S_0|$. □

Our method consumes the patching channel bandwidths much less than the traditional patching with fixed segment NVODs because originally variable segment NVODs have much less initial waiting time and the patching channel disappears after the first segment patching whereas the patching channel in fixed segment NVODs lasts until the end of the video. In our method, however, the total number of segments may increase by 1 at most by comparing Eqs. (1) and (2).

4. Conclusion

In this paper, we studied how to apply patching to variable segment NVOD scheduling. We proved that a naive solution may incur jitter or significant amount of patching bandwidth. We derived the maximum patching bandwidth in the worst case. We modified a variable segment NVOD algorithm by adjusting the sizes of variable segments so that the algorithm provides jitter-free playback of the video but consumes minimal bandwidth of the patching channel. We proved that the proposed algorithm enables jitter-free playback and its patching channel usage is minimal.

References

[1] K.C. Almeroth and M.H. Ammar, "On the use of multicast delivery to provide a scalable and interactive video-on-demand service," IEEE J. Sel. Areas Commun., vol.14, no.5, pp.1110–1122, 1996.
[2] K. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," Proc. ACM Multimedia, pp.191–200, 1998.
[3] S. Viswanathan and T. Imielinski, "Pyramid broadcasting for video on demand service," Proc. IEEE Multimedia Computing and Networking Conference, pp.66–77, 1995.
[4] A. Hui, "Video-on-demand broadcasting protocols: A comprehensive study," Proc. IEEE INFOCOM, pp.508–517, 2001.
[5] E.M. Yan and T. Kameda, "Generalized fibonacci broadcasting: An efficient VOD scheme with user bandwidth limit," Discrete Appl. Math., vol.154, no.16, pp.2418–2429, 2006.
[6] S. Chand, B. Kumar, and H. Om, "Segmented patching broadcasting protocol for video data," Comput. Commun., vol.32, no.4, pp.679–684, 2009.