ONLINE BANDWIDTH-EFFICIENT SCHEDULING FOR VIDEO-ON-DEMAND WITH RECURSIVE PATCHING

Yinqing Zhao, Zhi Shi and C.-C. Jay Kuo

Integrated Media Systems Center and Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089-2564
E-mail: {yinqingz, zshi, cckuo}@sipi.usc.edu

ABSTRACT

The service bandwidth optimization problem for multicast video-on-demand systems with early service merging is investigated, and a series of on-line algorithms based on recursive patching are proposed in this research. We adopt a simple fixed-threshold starting rule and introduce a control window to regulate the degree of service merging. Our results indicate that the cost-aware recursive patching (CARP) with a carefully chosen control window can significantly reduce the service bandwidth consumption. We also study the promoting rule and the limited client buffer effect for practical recursive patching algorithms. Experimental result shows that the proposed recursive patching schemes outperform graceful patching with a wide margin even with a very small client buffer size.

1. INTRODUCTION

Recent advances in computing and networking technology make real time multimedia applications such as video-on-demand (VoD) a reality. VoD services usually consume a large amount of resources, primarily the storage space and the network bandwidth. Moreover, the maximum number of concurrent streams that a server can support is limited by the smaller one of the server I/O bandwidth and the interconnection bandwidth. It is therefore important to design efficient scheduling algorithms for video streaming to improve the bandwidth utilization and overall system performance.

Similar to the patching algorithm proposed in [1], recursive patching is a technology built on the multicast mechanism, by which multiple clients can share the data from the same stream. The difference is that recursive patching can achieve further bandwidth reduction by allowing the merge of partial streams.

Previous research on recursive patching focus on the mathematical analysis and the improvement of the competitive ratio of online algorithms to the optimal off-line algorithm [2, 3, 4]. In this research, we propose a series of practical recursive patching algorithms to greatly reduce the service bandwidth consumption yet support immediate on-demand services. We start with a simple fixed threshold starting rule. Then, we introduce the concept of the control window to regulate the merging schedule. Unlike controlled multicast or other window-based patching optimizations, the dynamically calculated control window is determined only by the initial time of the stream. Therefore, no parameter fine-tuning is involved.

The rest of the paper is organized as follows. The basic staring rule is proposed Section 2. The proposed recursive patching algorithms are detailed in Section 3. The promoting starting rule and limited buffer effect are studied in Section 4. Performance comparisons between proposed algorithms and graceful patching are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. STARTING RULES OF RECURSIVE PATCHING

Recursive patching adopts a *receive-two* model, where clients can receive data from at most two streams simultaneously. In such a system, the server bandwidth is multiplexed into a set of logical channels, each of which is capable of transmitting a video program at the playback rate. When a client request arrives and there is a preceding stream of the same content available, the client immediately plays the program from a new stream while caching the subsequent part from the preceding stream. After some time, the client starts consuming data from the buffer while receiving data from one or two preceding streams to put into the buffer. From that point, we say that the client request is successfully merged into an earlier stream. The process can be done recursively so that multiple streams can be merged to an existing earlier stream. Therefore it is named recursive patching. Fig.1 shows a simple example of the recursive patching algorithm. Note that stream A_2 is extended from time 6 to time 12 due to subsequent requests coming at time 5, 6 and 7.

In recursive patching, when a new client request arrives at the server, a new stream x is initiated. The scheduler has to decide whether stream x can be merged into an eligible

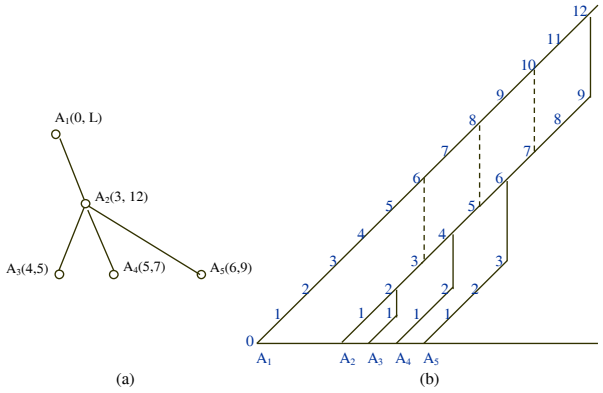


Fig. 1. Recursive patching: (a) Merge tree representation, and (b) time schedule.

preceding stream at some time in the future, or stream x is scheduled as a full stream. Furthermore, if x can be integrated into a service merge tree T (refer to Fig. 1 (a)), then the algorithm should decide to which stream on tree T that stream x should be connected. The first problem to resolve is when to start a full stream (*i.e.* the starting rule). Here we propose a simple starting rule as given below.

Fixed Threshold Starting Rule: If x is the very first arrival in the system, a full stream is initiated. Otherwise, let r denote the latest full stream, then a full stream is initiated if and only if $\tau_x^r - \tau_r^r > B$, where τ_x^r and τ_r^r are the initial times of x and r, B is the available client buffer space, and L is the length of the video program served.

The above starting rule can be briefly explained as follows. Here, for simplicity, we use the same symbol to represent both the client and the multicast stream initiated for that client. In order for stream x to finally merge with root r at certain time t, x must have received the same data as r at t. Since streams x and r have the same playback rate, when the gap between them is greater than buffer B, x can never accumulate enough data to "catch up" r before it ends. Even though the fixed threshold starting rule is not optimal, it does guarantee that any arrival $\tau_x^r - \tau_r^r \leq B$ can be successfully merged into the merge tree rooted at r.

With this starting rule, we are left with the problem of how to incorporate a new arrival into an existing service merge tree. Assume that we have an existing service merge tree T_n for arrivals $x_1, \cdots, x_n$ and would like to form a new tree T_{n+1} by incorporating a new arrival x_{n+1}. Based on the optimal merge tree property [5], we have the following basic merging rule.

Basic Merging Rule: If $x_1 = s_1, s_2, \cdots, s_k = x_n$ is the right most branches of T_n, then $T_{n+1} = T_{n+1}^i$ for some $1 \leq i \leq k$ such that $\tau_{x_{n+1}}^r < 2\tau_{x_n}^r - \tau_{s_{i-1}}^r$, where T_{n+1}^i is defined to be the tree T_{n+1} with s_i chosen as the parent of x_{n+1}.

3. RECURSIVE PATCHING ALGORITHMS

We propose several on-line recursive patching algorithms adopting the fixed threshold starting rule and the basic merging rule in this section. To simplify the choice of the eligible stream for a new stream to merge with, we introduce the notion of *control window*, which specifies the range of future arrivals that can merge with current streams. Let x represent a stream with initial time τ_x^r in the merge tree T. We associate x with a control window (τ_x^r, τ_x^w), which specifies that any future stream that can merge with x must be initiated before τ_x^w.

3.1. Greedy Recursive Patching

In the greedy recursive patching (GRP), the new arrival picks the closest stream as its merging target. Obviously, the goal of GRP is to make the length of the new stream as short as possible. We define the *cost* of stream x, $c(x)$, as the length of the stream. The control window of GRP can be calculated by the following proposition.

Proposition 1 Let x denote a stream in the merge tree created by GRP, τ_x^r denote the initial time of stream x. Then, the control window of node x can be expressed as $(\tau_x^r, \tau_x^r + c(x))$ using the greedy recursive patching (GRP).

3.2. Controlled Greedy Recursive Patching

GRP allows a new arrival to merge with its closest stream as long as the stream is active upon the arrival of the request. However, this merge may prolong the stream and increase the overall cost of the merge tree significantly. We need to tighten this condition a little bit. A stream is said *reachable* to a new arrival x, if x can be merged with this stream before the stream merges with one of its preceding streams. We add this restriction to GRP call the result algorithm controlled greedy recursive patching (CGRP) scheme.

Proposition 2 The control window of a non-root stream x can be expressed as $(\tau_x^r, \tau_x^r + \frac{c(x)}{2})$, where τ_x^r is the initial time of x and $c(x)$ is the cost of x. For the full streams, the control window is $(\tau_x^r, \tau_x^r + L)$, where L is the length of the program.

3.3. Cost-aware Recursive Patching

In GRP and CGRP, a new stream is allowed to merge with its closest preceding stream. This is based on one intuitive assumption, *i.e.* to merge with the closest preceding stream is the most desirable. However, this assumption may not always be true. Let x be the latest stream and $p(x)$ the immediate parent of x in the merge tree. For a new arrival y, the greedy scheme is preferable only when $\tau_y^r < \tau_x^r + \frac{\tau_x^r - \tau_{p(x)}^r}{2}$. This implies that it may be desirable to choose the control window of a non-root stream x to be $(\tau_x^r, \tau_x^r + \frac{\tau_x^r - \tau_{p(x)}^r}{2})$.

Although this choice does not guarantee the optimal performance, it outperforms GRP and CGRP greatly in our experiment. The result scheme is called cost-aware recursive patching (CARP).

4. DESIGN ISSUES OF RECURSIVE PATCHING

4.1. Promoting Starting Rule

In the previous section, we assume that all window-controlled algorithms follow the fixed threshold starting rule. However, this starting rule may cause the unwanted worst case performance for certain incoming arrival distribution, *i.e.* the uniform distribution with inter-arrival time equal to $L/2$. To make the starting rule adaptive, we consider the following modification.

Promoting rule: If x is the very first arrival in the system, a full stream is initiated. Otherwise, if the addition of a new arrival causes the cost of its ancestors to exceed that of a full stream, then the latest such ancestor y and its subtree are removed from the original tree to form a new tree T'. The node y is promoted to be the root of the new merge tree T', and the length of the stream is extended to L.

It is desired that the stream being promoted, denoted as y, is an immediate son of the original root r. Otherwise, the prolonged transmission of the ancestors of y will be wasted since now y can provide all necessary data segments for its descendants, which were provided by the ancestors of y in the original tree.

4.2. Effect of Limited Client Buffer

The client buffer is used to gap the distance between streams that will finally merge together. Since a client may only have a limited buffer, the recursive patching scheme should adapt to this constraint. Let B denote the client buffer size. For a stream x on a merge tree T rooted at r, define $\Delta\tau = \tau_x^r - \tau_r^r$ as the skew between the root and x. Then, we can extend the original stream merge pattern to three different patterns according to the relation between $\Delta\tau$ and B. (1) *Head merge* with $\Delta\tau \leq B$. The client buffer is large enough to hold the skew and the buffered content is the initial portion of the video data. (2) *Tail merge* with $\Delta\tau \geq L - B$. The buffered content is the last portion of the video data and the partial stream may finish later than the full stream. (3) *Bounded merge* with $B < \Delta\tau < L - B$. The client buffer is not large enough to hold the skew and there is a certain period during which the client only receives from one stream. The client buffer effect is studied in the next section.

5. EXPERIMENTAL RESULTS

We conducted experiments to compare the performance of three recursive patching algorithms (*i.e.* GRP, CGRP and CARP) with graceful patching (GP), which is one of the best patching schemes also built on the receive-two model. We use the mean service bandwidth required to support on-demand services and the normalized number of requests [1] as the performance metric. A better scheme should require less mean service bandwidth and support relative large normalized number of requests at the same time.

Fig. 2 shows the performance of patching schemes under different request inter-arrival time. The incoming requests follow a Poisson distribution. CGRP and CARP consistently outperform GP and GRP by a large margin. The performance gap decreases with the increase of the inter-arrival time. This is due to the increase in the non-overlapping portion of requested streams. When the inter-arrival time is large, each stream will deliver more data before it can be merged to its precedents, which is proved by the drop in the normalized number of requests in Fig. 2(b).

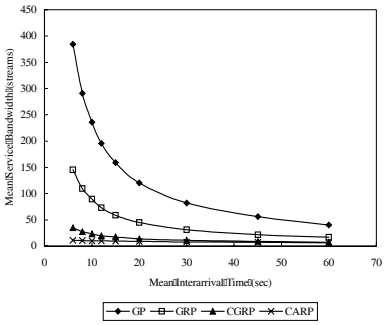
Fig. 3 illustrates the effect of the promoting rule on uniform arrivals with the inter-arrival time set to 1 second. The schemes adopting the fixed threshold starting rule are denoted by an "F" suffix, while the schemes adopting the promoting rule with a "P" suffix. We see that CARP-P consistently outperforms CARP-F in the heavy load case. As shown in Fig. 3, the curves of schemes adopting the promoting rule is smoother than those schemes adopting the fixed threshold starting rule, which indicates that the promoting rule is actually a more dynamic rule with less fluctuation under different system load.

Finally, we compare the proposed schemes with graceful patching under the limited buffer constraint. We adopted the fixed threshold starting rule for GRP and CGRP, and chose the promoting rule for CARP, denoted as PR-CARP. The incoming requests follow a Poisson distribution with the mean inter-arrival time set to 10 seconds. As shown in Fig. 4, the proposed recursive patching schemes outperform graceful patching with a wide margin under a small client buffer size (less than 60 seconds). Compared with other three schemes, PR-CARP can merge the service streams much more effectively with very limited client buffer space.

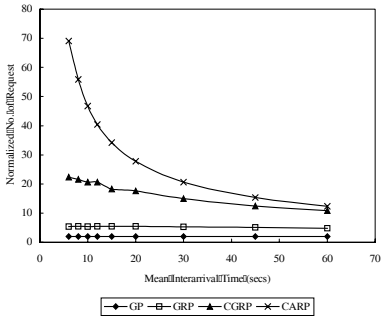
6. CONCLUSION

Three practical on-line recursive patching algorithms (*i.e.* GRP, CGRP and CARP) were proposed in this research. It was shown by simulation results that the proposed schemes improved the performance of the traditional graceful patching scheme by a factor of $50\% - 90\%$. Furthermore, CARP with the promoting rule can successfully adapt to various system load conditions without performance degradation.

[1] The normalized number of requests is the number of requests that can be served by the bandwidth used to serve a single request using unicast streaming method.

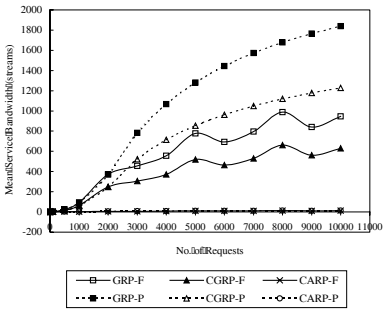


(a) Mean service bandwidth

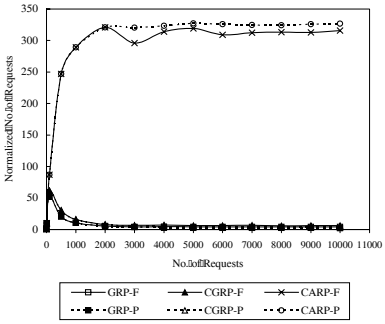


(b) Normalized number of requests

Fig. 2. The effect of the mean inter-arrival time with Poisson distributed incoming requests.

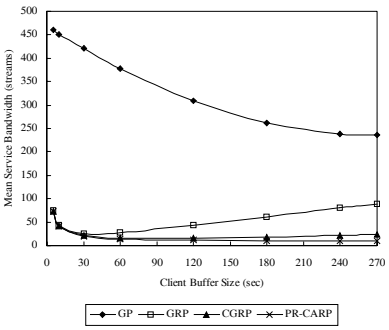


(a) Mean service bandwidth

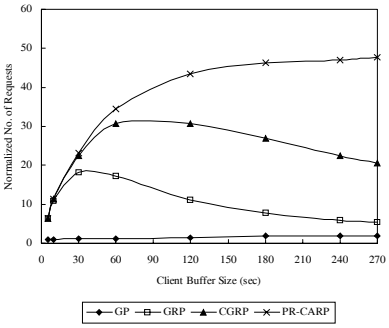


(b) Normalized number of requests

Fig. 3. The effect of the promoting rule on the uniform distributed arrivals.



(a) the mean service bandwidth



(b) the normalized number of requests.

Fig. 4. The effect of the limited buffer constraint.

7. REFERENCES

[1] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," *Proceedings of the sixth ACM international conference on Multimedia*, pp. 191–200, September 1998.

[2] Amotz Bar-Noy and Richard E. Ladner, "Competitive on-line stream merging algorithms for media-on-demand," *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 364–373, January 2001.

[3] E. Coffman, J. Jelenkovic, and P. Momcilovic, "Provably efficient stream merging," *Proceedings of Sixth lnternational Workshop on Web Caching and Content Distribution*, June 2001.

[4] W.T. Chan, T.W. Lam, H.F. Ting, and Prudence W.H. Wong, "Improved on-line stream merging: from a restricted to a general setting," *Proceedings of the 7th Annual International Computing and Combinatorics Conference (COCOON)*, pp. 432–442, 2001.

[5] Z. Shi, "Theory and applications of bandwidth-efficient scheduling for multicast on-demand services," *Ph.D thesis, Department of Electrical Engineering-System, USC*, May 2002.