Supporting interactivity in Patching - Techniques for Reducing Bandwidth Overheads [1]

Bing Wang, Subhabrata Sen, Lixin Gao†, and Don Towsley

Tech. Report 01-24
{bing,sen,towsley}@cs.umass.edu †lgao@ecs.umass.edu

Abstract

This paper explores a number of techniques for supporting interactive operations such as pause and bi-directional index jumps in patching. We present and examine three schemes, which differ in whether the client buffer contents is used for rescheduling and whether the patches after an interactive action are transmitted by unicast or multicast. Performance evaluations, based on a combination of analysis and simulations, suggests that (i) proper choice of the threshold for controlling the frequency at which new multicasts of the complete video are started, (ii) careful use of client buffer content and (iii) using multicast for the patches after an interactive operation can result in significant savings in transmission bandwidth overheads.

I. Introduction

A pervasive high-speed networking infrastructure and a mature digital video technology has led to the emergence of several networked streaming media applications. Examples include live video broadcasts, distance learning, corporate telecasts, narrowcasts, and streaming of Web video clips. However, due to the high bandwidth requirements ($4-6$ Mbps for full motion MPEG-2) and the long-lived nature ($10-120$ minutes) of digital video, server and network bandwidths are proving to be major factors in limiting the widespread usage of video streaming over the Internet. This is further complicated by the fact that the client population is likely to be both large and asynchronous. There has, therefore, been tremendous interest in developing techniques for bandwidth-efficient distribution of network video to such a client population.

For popular videos, the server and network resources can be significantly reduced by allowing multiple clients to receive all, or part of, a single transmission [1–6]. For example, the server could *batch* requests that arrive close together in time [1], and multicast the stream to the set of clients. Another technique, *patching* or *stream tapping* [4,5], exploits the client's buffer space and the existence of sufficient client network bandwidth to listen to multiple simultaneous transmissions (either by passively listening on a shared medium or by joining/leaving multiple multicast groups) to reduce server and network transmission bandwidth requirements, while guaranteeing zero-delay playback startup. The server multicasts the entire video sequentially to the very first client. Client-side *workahead buffering*

[1]This research was supported in part by the National Science Foundation under Grants NSF ANI-9973092, NSF grants EIA-0080119, ANI9977635, ANI-9977555, NSF ANI-9977555, and NSF CAREER Award grant ANI-9875513. Any opinions, findings, and conclusions or recomendations expressed in this material are those of the authors and so not necessarily reflect the views of the National Science Foundation.

is used to allow a later client to receive (part of) its future playback data requirement by listening to an *existing* ongoing multicast transmission of the same video, while the server transmits (via unicast) afresh the remaining required frames. A threshold [2,3,5] is used to control the frequency at which new multicasts of the complete video are started.

This paper explores a number of techniques for supporting interactive operations such as pause and bi-directional index jumps in patching, while still making efficient use of server and network bandwidths. We consider a *baseline* patching scheme in which, after an interactive action, the server locates the ongoing complete transmission which is nearest to the client's resume point, patches the client into this stream, and transmits any missing data via unicast. This simple approach treats a resume request just like a new request. The bandwidth requirements of interactive operations can be further reduced if retransmitting data already present in the client's workahead buffer can be avoided. The Buffer-Unicast (BU) patching scheme accounts for the client buffer contents for rescheduling after an interactive action. In addition to exploiting the client buffer contents, the BM (Buffer-Multicast) patching scheme multicasts the patch after an interactive operation so that this transmission can be shared by a later client. In many streaming applications, clients can tolerate some imprecision in the position from which playback resumes and a few seconds latency in the resumption of playback after an interactive operation. These tolerances are utilized in the above three schemes to maintain low transmission overheads.

Our evaluations, based on a combination of analysis and simulations, suggest that (i) proper choice of the threshold for controlling the frequency at which new multicasts of the complete video are started, (ii) careful use of client buffer content and (iii) using multicast for the patches after an interactive operation can result in significant savings in transmission bandwidth overheads. For instance, we find that careful use of client buffer content can lead to as much as 50% savings in bandwidth usage over the case of not utilizing the buffer contents, and using multicast for the patches after a forward jump can result in as much as 47% additional bandwidth savings in some settings. However, we find that exploiting the client tolerances for approximate resume position and resume latency does not result in significant savings in the server load, for the scenarios we consider. For example, for 30 seconds of approximate jump and resume latency, the server load is only reduced by 10% under BM patching.

This research complements other works on interactive video streaming. Substantial literature exists on supporting VCR-like operations in the context of one-one video streaming [7–9]. [10] deals with interactivity for staggered broadcasts, [11,12] provide VCR-functions in the context of batching, and [13] considers interactivity in the context of periodic broadcast. [14] extends stream tapping [4] by transmitting the patches via unicast to handle interactivity. Our work complements existing research by (i) investigating the importance of choosing the appropriate threshold for minimizing bandwidth overheads, (ii) evaluating the usage of client buffer to minimize transmission overheads, (iii) investigating the substantial bandwidth savings from using multicast to send the patches after an interactive action, and (iv) exploring the potential of approximate jump and playback restart latency on reducing the server load.

The rest of the paper is organized as follows. Section II describes the problem setting. Section III presents the techniques to support interactivity in patching. Section IV evaluates the bandwidth requirements of the techniques in two scenarios. Finally, Section V concludes the paper and describes ongoing work.

II. Patching Model

We next provide a formal model of the patching system, and introduce notation and key concepts that will be used in the rest of the paper. Consider a video server and a group of clients connected by a multicast (or broadcast) capable network. We focus on patching for a single video, since requests for different streams do not interact. When multiple clients arrive simultaneously, the requests are served as a single batch.

We assume that the initial request from any client is for playback from the beginning of the video. In response to a client request, the server computes and delivers a *reception schedule* to the client. A *transmission channel* denotes a logical entity that may intermittently transmit different segments of the video. A channel can be either a unicast or multicast channel. The reception schedule specifies, for each frame in the video, when and from which channel a client should receive that frame. The server then transmits the video in accordance with the computed reception schedule. In patching, a client may receive data from multiple channels simultaneously and may need to switch between different channels across time. Frames received ahead of their playback times are stored in a *workahead* buffer.

We focus on three interactivity operations: Pause, Backward index Jump (BJ) and Forward index Jump (FJ). In many applications, clients may be willing to tolerate a certain amount of delay before playback resumes after an interactive operation. Define *Tolerable Resume Latency* D to be the maximum duration that a client is willing to wait. In some applications, it may not be important to begin playback at the precise destination of the interactive operation, and a nearby position may be an acceptable resume point. Define *Tolerable Jump Imprecision* δ to be the maximum acceptable distance between the destination of a jump and the actual playback resume point.

For simplicity, we assume that the client has sufficient buffer space to accommodate an entire video file. This is a reasonable assumption for a range of client devices - e.g., commodity PCs today contain a few tens of GB disk space. In this scenario, *pause* action can be supported transparent to the server in a straightforward manner: for the duration of the pause action, the client continues receiving the video according to the original reception schedule, and buffers the data locally. In the remainder of the paper, we focus on how bi-direction jump can be supported in patching.

III. Techniques for reducing bandwidth overhead

In patching, a complete stream is scheduled for the first request using multicast. A later request shares the complete stream and obtains the missing data via unicast. A client issuing a resume request after a jump action can be treated in a similar way. That is, the server finds an ongoing complete stream for the client to patch into and transmits the missing data by either unicast or multicast. If no ongoing complete stream is found, a multicast stream is scheduled for the client. At the time a client issues a resume request, all the multicast streams that are transmitting data later (closer to the end of the video) than the resume point are *candidate* streams for the client to patch into.

We consider a *baseline* patching scheme in which, after an interactive action, the server locates a candidate stream nearest to the client's resume point, patches the client into this stream, and transmits any missing data via unicast. The bandwidth requirements of interactive operations can be further reduced

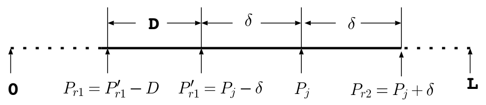


Fig. 1. Tolerable jump imprecision and resume delay.

if retransmitting data already present in the clients workahead buffer can be avoided. The Buffer-Unicast (BU) patching scheme accounts for the client buffer contents when computing the clients reception schedule after an interactivity action. In addition to exploiting the client buffer contents, the BM (Buffer-Multicast) patching scheme multicasts the patch after an interactive operation so that this transmission can be shared by a later client.

We next describe the three schemes in detail. Suppose the jump is to position P_j within the video. As shown in Fig. 1, with a tolerable jump imprecision of δ, the resume point can be any point between $P'_{r1} = \max(0, P_j - \delta)$ and $P_{r2} = \min(L, P_j + \delta)$, where L is the length of the video. With a tolerable resume delay of D, any stream that is in the region from $P_{r1} = P'_{r1} - D$ to $P_{r2} - D$ will be in the valid resume region after time D. In summary, with the tolerable jump imprecision and resume delay, if the client patches into any candidate stream which is at the position from P_{r1} to P_{r2}, no additional patch is needed.

A. Baseline patching

Baseline patching is a simple scheme where a resume request is treated the same as a new request. The client buffer contents are not used for rescheduling after a jump action. It is described as:

- Case 1: if a candidate stream is transmitting data between P_{r1} and P_{r2}, then the client patches into this stream and no unicast is required.
- Case 2: If no candidate stream is transmitting data in the range of $[P_{r1}, P_{r2}]$, the client patches into the one that is at a point nearest to P_j since it requires the least additional bandwidth. The server transmits the missing data via unicast.
- Case 3: If there is no candidate stream, the server schedules a multicast stream transmitting data from the resume point to the end of the video for the client.

The advantage of this scheme is that it is easy to implement: the server does not need to keep track of what contents are in the client buffer and the client does not need to inform the server its buffer information.

B. BU patching

BU patching takes advantage of the data stored in the client buffer to reduce the additional bandwidth required to accommodate interactivity. It differs from Baseline patching only in Case 2. Suppose in

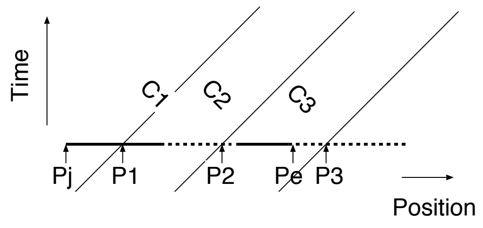


Fig. 2. Case 2 in BU patching: an example.

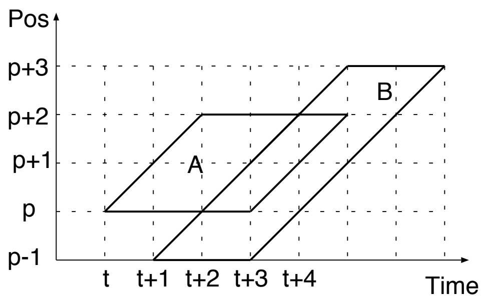
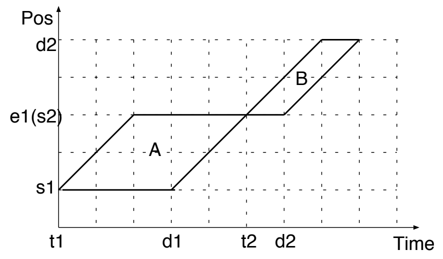


Fig. 3. Merge patch streams in BM patching: an example.

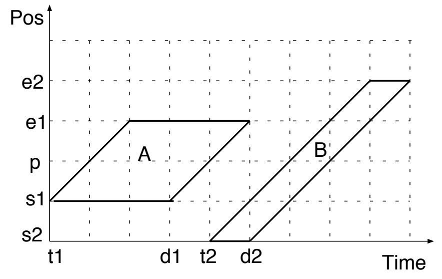
Case 2, it patches into a stream which is at position P_m. The contents after P_m not in the client buffer are received from this stream and the missing segments between P_j and P_m are received via unicast in a lazy manner, just in time for playback. Receiving in a lazy manner is important since it maximizes the amount of workahead data received from an ongoing multicast stream thereby reducing the additional bandwidth.

We now illustrate Case 2 in BU patching by an example shown in Fig. 2. The solid lines and the dashed lines represent respectively that the corresponding segments are stored or not stored in the client local buffer. The jump is to position P_j and there is no content in the buffer after position $P_e(P_e \geq P_j)$. At the time of the jump, an ongoing candidate stream C_i is at position $P_i(i = 1, 2, 3)$ and $P_j < P_1 < P_2 < P_e < P_3$. The client selects to patch into stream C_1 since P_1 is nearest to P_j. In this case, no additional bandwidth is needed as all the contents between P_j and P_1 are already in the client buffer. If C_1 does not exist, then the client selects to patch into C_2 since P_2 is nearest to P_j. All of the data after P_2 not residing in the client buffer is obtained from C_2. All the segments between P_j and P_2 not in the client buffer are obtained by unicast in a lazy manner. Similarly, if both C_1 and C_2 do not exist, then the client patches into C_3 and the two missing segments between P_j and P_3 are obtained via unicast in a lazy manner.

In BU patching, in order to simplify buffer management, we can specify that only segments larger than some length are stored in the buffer.



(a) Case 1: the requested segments of the two clients do not overlap.



(b) Case 2: there can be overlap in the requested segments of the two clients.

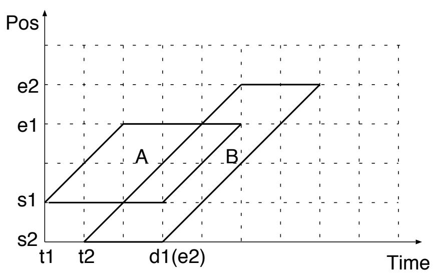
Fig. 4. The earliest start time of the patch for client B is later than the latest start time of client A ($t_2 \geq d_1$).

C. BM patching

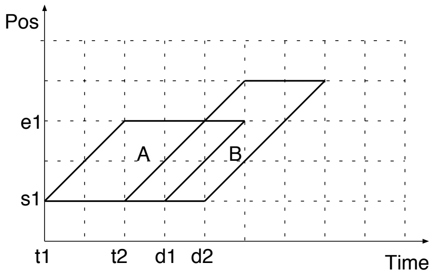
In BU patching, when a client patches into a candidate stream which is at position P_m, the missing segments between P_j and P_m are transmitted using unicast in a lazy manner. In BM patching, these missing patches are transmitted using multicast so that later clients can share them in order to further reduce bandwidth. To simplify the scheme and also to ensure that the client needs to receive from at most three channels (the reason will be made clear later), only the patch closest to P_m is transmitted using multicast. For example, in Fig. 2, when the client patches into stream C_3, only the segment of $[P_e, P_3]$ is sent from the server via multicast.

A critical component in the BM patching deals with merging multicast patches to save bandwidth. When the buffer contains data that will be played out prior to the patch or when resume delay is allowed, the transmission of the patch is not required to be started immediately. There is a range of times that the transmission of the patch can be started to ensure smooth playback. Fig. 3 shows an example. Client A needs a patch of $[p, p+2]$ to be transmitted between t and $t+3$. In Fig. 3, the range of start times and the required segment for client A is represented by a parallelogram with the left-bottom point of (t, p) and the right-top point of $(t+3, p+2)$. A later client B needs a patch of $[p-1, p+3]$ to be transmitted between time $t+1$ and $t+3$. If the patch required by A (i.e. $[p, p+2]$) is transmitted at time t, it reaches position $p+1$ when B comes (at time $t+1$). This patch can then be extended to position $p+3$ for B. But B still needs the segment of $[p-1, p+1]$, which leads to another bandwidth usage of 2 units. A total bandwidth of 5 is required for client A and B. If a patch of $[p-1, p+3]$ is transmitted at time $t+1$, it can be shared by both client A and B. Therefore a total bandwidth of 4 units is required. It is clear from this example that, to minimize bandwidth, the patch streams for client A and B should be merged by considering the range of starts times and requested segments of both clients together.

We now describe how to merge the patch stream for a later client into the patch stream for an earlier client. The algorithm is shown in Fig. 6. Suppose a patch stream M_1 of the segment $[s_1, e_1]$ is scheduled for client A. The earliest and the latest start time for the patch are time t_1 and d_1 respectively. A later client B comes at time t_2 for segment $[s_2, e_2]$. Its latest start time is time d_2. This algorithm is divided into 4 cases based on the range of start times and the requested segments of the two clients. In both



(a) Case 3: start position of client B is smaller than that of client A.



(b) Case 4: start position of client B is no less than that of client A.

Fig. 5. The earliest start times of the patch for client B is no later than the latest start time of client A ($t_2 \leq d_1$).

Case 1 and Case 2 (see Fig. 4), the range of the start times of the two clients do not overlap. In Case 1, the requested segment of the two clients do not overlap either. The server schedules a multicast patch of the segment $[s_2, e_2]$ with the earliest and latest start time of t_2 and d_2 respectively for client B since the streams for client A and B can not be merged. In Case 2, there can be some overlap between the two clients. As shown in Fig. 4 (b), at time t_2, the transmission of patch M_1 for client A reaches position p. This patch can be extended to position e_2 for B. And the server transmits the segment of $[s_2, p]$ to B by unicast. In Case 3 and Case 4 (see Fig. 5), the ranges of the start times of the two clients overlap. In Case 3 and Case 4, the start points of client B are respectively smaller and no less than that of client A. In both cases, the patches for the two clients can be merged depending on the relative range of start times and requested segments of both clients, similar to the example in Fig. 3.

In the algorithm, the patch for a later client can be accommodated by extending the patch for an earlier client. The segment that can not be obtained from an earlier patch is transmitted from the server by unicast in a lazy manner. Therefore clients need to receive from at most three streams simultaneously: at most two multicast streams (a complete stream and a patch stream) and one unicast stream (the missing segments transmitted in a lazy manner). In both Baseline and BU patching, clients need to receive from at most two streams simultaneously.

In BM patching, using the contents in the client buffer not only decreases the bandwidth required from the server but also increases the distance between the earliest and the latest start time of the patch so that more patches can be merged together. The tolerable resume latency also increases the distance between the earliest and the latest start time of the patch.

D. Threshold Selection

Thresholds are incorporated into several patching schemes [2, 3, 5] to determine when to initiate a new complete transmission of the entire video. The server does not start a new complete transmission unless the client request arrives beyond a threshold time from when the latest complete stream is started. For a Possion arrival process, it is possible to derive a closed-form expression for the optimal value of the threshold [5].

In the presence of interactivity, a natural question to ask is whether a threshold still helps and so

*[t]

```
Case 1 (Fig. 4 (a)): $((t_2 > d_1)\&\&(t_2 > d_1 + e_1 - s_1))$
  Schedule a multicast patch of $[s_2, e_2]$ between time $t_2$ and $d_2$ for client B.
Case 2 (Fig. 4 (b)): $((t_2 > d_1)\&\&(t_2 < d_1 + e_1 - s_1))$
  $p = s_1 + t_2 - d_1$
  if $(p > s_2)$
    Extend the end point of the patch $M_1$ to $\max(e_1, e_2)$.
    Client B obtains segment $[p, e_2]$ from the patch $M_1$.
    Server transmits segment $[s_2, p]$ for client B via unicast.
  else
    Schedule a multicast patch of $[s_2, e_2]$ between time $t_2$ and $d_2$ for client B.
Case 3 (Fig. 5(a)): $((t_2 \leq d_1)\&\&(s_2 < s_1)\&\&(e_2 > s_1))$
  $time = d_1 - (s_1 - s_2)$
  If $(time < t_2)$
    $d_1 = \min(d_1, d_2)$
    Client B obtains segment $[s_1, e_2]$ from the patch $M_1$
    Server transmits segment $[s_2, s_1]$ for client B via unicast.
  else
    $d_1 = time$
    Schedule a multicast patch of $[s_2, \max(e_1, e_2)]$ with the latest start time of $d_1$.
Case 4 (Fig. 5(b)): $((t_2 \leq d_1)\&\&(s_2 \geq s_1)\&\&(s_2 < e_1))$
  $time = t_2 + s_2 - s_1$
  If $(time > d_2)$
    Schedule a multicast patch of $[s_2, e_2]$ between time $t_2$ and $d_2$ for client B.
  else
    $d_1 = \min(d_1, d_2 - (s_2 - s_1))$
    Schedule a multicast patch of $[s_1, \max(e_1, e_2)]$ with the latest start time of $d_1$.
```

Fig. 6. Algorithm to merge the patch for a later client (client B) into the patch M_1 for an earlier client in BM patching.

how to choose it. In our model, when the interactive operations are only BJ and Pause (without FJ), the reception schedule does not need to be changed after an interactivity action. Thus the optimal threshold will be the same as the one without interactivity.

When interactivity results in changes to the client's reception schedule and, hence, the sharing of the streams, the threshold with the presence of interactivity may differ from the one without interactivity. The threshold can be affected by several factors. The jump actions may reduce the sharing of the complete stream, which tends to increase the length of optimal threshold. On the other hand, the threshold are forced to be decreased since longer threshold leads to longer patch for a jump action, which can increase the bandwidth required to support interactivity. Furthermore, whether multicast or unicast is used for the patches after the interactivity affects the selection of the threshold. As we show in the evaluation, the optimal threshold makes a balance of the several factors.

IV. Performance Study

In this section, we evaluate the performance of the techniques proposed in Section III in two scenarios. We assume a constant-bit-rate (CBR) 90-minute long video and there are sufficient channels in the server. The size of the client buffer is assumed to be as least 90 minutes. Requests for the video arrive according to Poisson process with an arrival rate of λ. A client starts in the normal playback mode. The duration of the playback is assumed to be exponentially distributed with mean of $1/\mu$. When the client issues an interactivity request, it enters an interactivity mode, where it has the probability of executing a Pause, BJ or FJ action. After the interactivity action, it returns to the normal playback mode.

We consider two scenarios in which FJ is the only form of interactivity. The scenarios differ in that the length of the jump is assumed to be deterministic of in Scenario 1 and is assumed to be uniformly distributed in a range in Scenario 2. The focus of the evaluation is to investigate the benefits of using the client buffer contents and using multicast for the patches after an interactive action. Pause actions accumulate contents in the client buffer. The possibility of utilizing contents in the client buffer is large after a BJ action. Hence, for the same number of interactivity actions, the number of patches required to support interactivity with Pause and BJ actions is less than that with only FJ actions. Therefore, we expect, with the presence of Pause and BJ, the advantage of BU patching over Baseline patching becomes more dramatic and the advantage of BM patching over BU patching becomes less dramatic than in the two scenarios.

The evaluation is based on both simulations and analysis. Each simulation lasts for 9000 minutes. The goal of the analysis is to provide some insight into the problem. We use the average number of busy channels per unit of time as a measure of transmission bandwidth. We use the total number of patches used for interactivity throughout the simulation time to measure the load on the server. The reason is that data for the patches may have to be extracted from the disk if it is not in the memory; thus places a load on the server. In Scenario 1, jump imprecision and resume latency are not allowed. The focus is to investigate the bandwidth requirement and determine how FJ affects the selection of threshold in the three schemes. In Scenario 2, jump imprecision and resume latency are allowed. Another focus in this scenario is the effect of tolerable jump imprecision and resume latency on the server load in BM patching.

A. *Evaluation in Scenario 1*

We investigate the bandwidth usage in Baseline and BU patching by analysis. For BM patching, an exact analysis is hard to achieve. Therefore the results are by simulation. We set the length of the threshold to be in the range from 0 to the length of the video. The one requires the minimum bandwidth is determined to be the optimal threshold.

In the analysis, we assume a discrete time system. Suppose the threshold is T. Let $\{k_i\}_{i=0}^{\infty}(k_0 = 0)$ denote the times at which the server schedules the ith transmission of the complete stream. Let $\{Z_i\}_{i=0}^{\infty}$ denote the inter-start times of consecutive complete streams, $Z_i = k_{i+1} - k_i$. Denote the mean of Z_i as Z. Then $Z = T + 1/\lambda$. In the following, we assume two complete streams are Z units apart for simplicity. The analysis focuses on requests in one complete transmission period. Let W denote the server bandwidth for requests arriving in this period. Let $E[W_o]$ be the average server bandwidth required without interactivity. Let $E[W_a]$ be the average additional bandwidth required because of interactivity. Then,

$$E[W] = E[W_o] + E[W_a]$$

The optimal threshold minimizes the average bandwidth per unit of time: $\frac{\lambda E[W]}{\lambda T+1}$. We know $E[W_o] = L + \frac{\lambda}{2}T^2$ from [5]. It remains to derive an expression for $E[W_a]$.

In the analysis, we use the length of the deterministic jump as one unit. Denote the length of the video as L units. Suppose FJ actions can only happen at the beginning of a time unit with probability p. When the play duration is exponential with the mean of $1/\mu$, we have $p = 1 - e^{-\mu}$. The number of jumps in length L is according to Binomial distribution.

Let $\bar{F}(i)$ be the probability of having no less than i jumps before reaching the end of the video (in the distance of L units). Then:

$$\bar{F}(i) = 1 - \sum_{k=0}^{i-1} \left(\begin{array}{c} L \\ k \end{array} \right) (1-p)^k p^{l-k}$$

A.1 Baseline patching

Let $P(i)$ be the probability of doing the first FJ at time $i (i \geq 1)$ relative to when playout starts. Then:

$$P(i) = (1-p)^{i-1}p$$

Claim: If a request is s time units behind the complete stream that it patches into, the additional bandwidth needed for the client before the next jump is $B(s)$ when client buffer contents are not used for rescheduling, where

$$B(s) = \sum_{i=1}^{s-1} iP(i) + s(1 - \sum_{i=1}^{s-1} P(i)).$$

Proof: When the client buffer contents is not used for rescheduling, a patch is needed for the client after each FJ. If after rescheduling, the client is s units behind the complete stream it patches onto, it needs a patch of s units. If the first FJ happens at time $i (i < s)$, the rest of the unicast patch can be released which leads to an actual bandwidth of i units. If the first FJ happens at time $i (i \geq s)$, a bandwidth of s is needed. Therefore the actual usage of bandwidth is $B(s)$. ∎

A client arriving at $i-1$ ($1 \leq i \leq T$) after the complete stream is started is $i-1$ time units behind the complete stream. The jth ($j < i$) jump places it behind the complete stream by $i-1-j$ time units. Hence, after the jth jump, it can still patch into the original complete stream and needs a patch of length $i-1-j$, which leads to a bandwidth usage of $B(i-1-j)$. The additional bandwidth thus required is:

$$E[W_a^{i1}] = \sum_{j=1}^{i-1} \bar{F}(j)B(i-1-j)$$

After the ith jump, it patches into a complete stream which is Z units ahead of the original stream. Thus it is behind this complete stream by $Z-1$ unit and needs a patch of length $Z-1$. Let $Y = Z-1$.

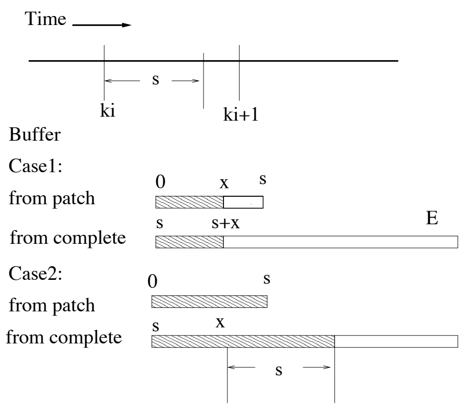


Fig. 7. Client buffer contents at the time of a FJ action.

The jth ($i < j < i + Y$) jump places it behind the complete stream by $Y - (j - i)$ time units, which leads to an actual usage of bandwidth of $B(Y - j + i)$. This can happen m times, where $m = \lfloor \frac{L-i}{Z-1} \rfloor - 1$. The additional bandwidth thus required is:

$$E[W_a^{i2}] = \sum_{k=0}^{m} \sum_{j=i+kY}^{i+(k+1)Y-1} \bar{F}(j)B(Y - j + (i + kY))$$

Therefore, the total additional bandwidth required is:

$$E[W_a] = \lambda \sum_{i=1}^{T} (E[W_a^{i1}] + E[W_a^{i2}])$$

A.2 BU scheme

Claim: If a request is s time units behind the complete stream that it patches into, it is only necessary to reschedule a complete stream for it when its jump distance exceeds s when client buffer contents are used for rescheduling.

Proof: Consider the arrivals in the interval $[k_i, k_{i+1}]$. Suppose that an arbitrary request arrives at $k_i + s$ as shown in Fig. 7. It is behind the latest complete stream by s time units. Assume the position of the first jump is x and the jump distance is y.
(i) Case 1 ($0 < x \leq s$): FJ occurs when receiving from the partial stream. If $y \in [0, s - x]$, the client jumps to a new point in the partial stream. Then the schedule for the partial stream is changed and the schedule for the complete stream can be maintained. If $y \in (s - x, s]$, the client jumps to a point in the complete stream. Because data $[s, s + x]$ resides in the client buffer, there is no need to change the schedule for the complete stream. Only when $y > s$, it is necessary to reschedule.
(ii) Case 2 ($x > s$): The FJ occurs after complete receiving the partial stream. At the time of the FJ, the client has s unit of data in the buffer. The only time the client has to reschedule is when $y > s$.
The situation where the client is s units behind the complete stream it patches into because of rescheduling after an interactivity can be treated in the same way. ∎

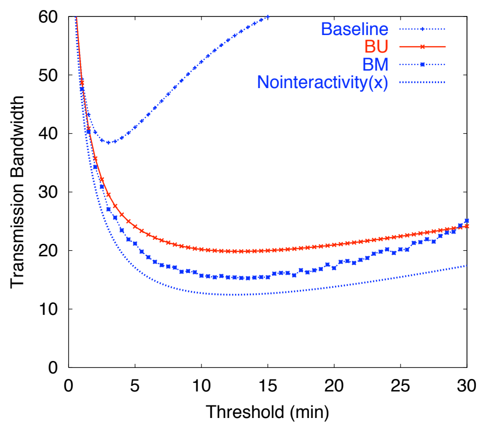


Fig. 8. Transmission bandwidth as a function of threshold in Scenario 1, λ=1.0/min, $1/\mu$ = 10 minutes.

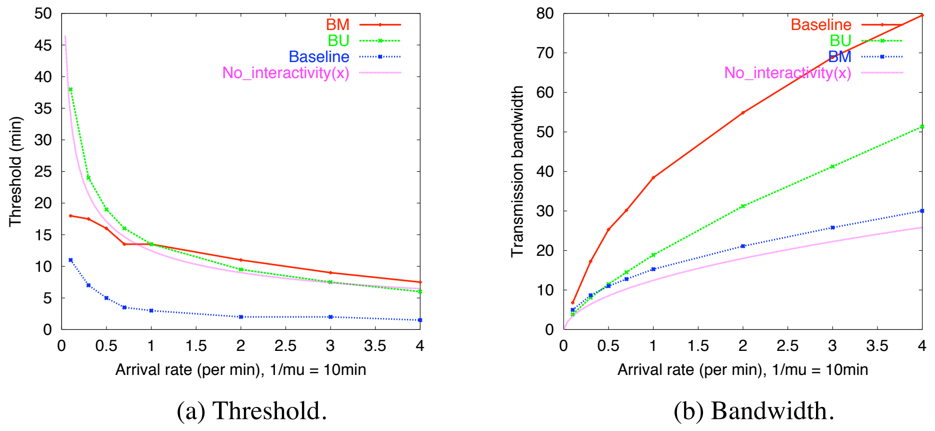


Fig. 9. Threshold and bandwidth as a function of the arrival rate in Scenario 1.

From the above claim, a client arriving $i-1$ $(1 \leq i \leq T)$ units after the complete stream is started needs to be rescheduled to a new complete stream after its ith jump. It is then behind the new complete stream by $Z-1$ units and needs a patch stream of length $Z-1$ via unicast. Afterwards, its $(Z+i)$th jump leads to another patch stream of length $Z-1$. A jump while receiving from unicast stream saves one unit of bandwidth. While receiving from a patch stream of l units, the average bandwidth saving is lp. Therefore the additional bandwidth actually required is $l(1-p)$. In summary, the additional bandwidth required is:

$$E[W_a^i] = \sum_{k=1}^{m} \bar{F}(i+kZ)(Z-1)(1-p) - ip$$

Therefore, the total additional bandwidth required is:

$$E[W_a] = \lambda \sum_{i=1}^{T} E[W_a^i]$$

A.3 Numerical results

We now use numerical results from analysis and simulations to evaluate the performance of the three schemes in Scenario 1. The length of the deterministic jump is assume to be 30 seconds. It is small relative to the length of the video and can be used as one unit. The results for Baseline and BU patching

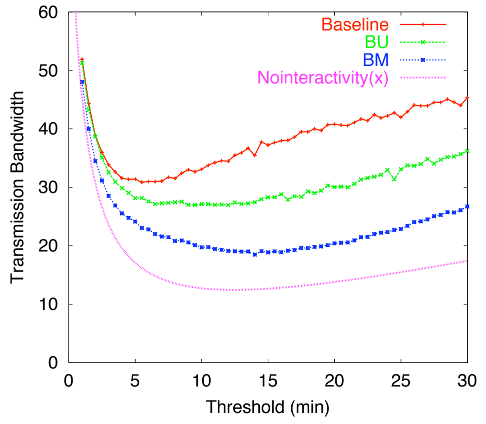


Fig. 10. Transmission bandwidth as a function of threshold in Scenario 2, λ=1.0/min, $1/\mu$ = 10 minutes.

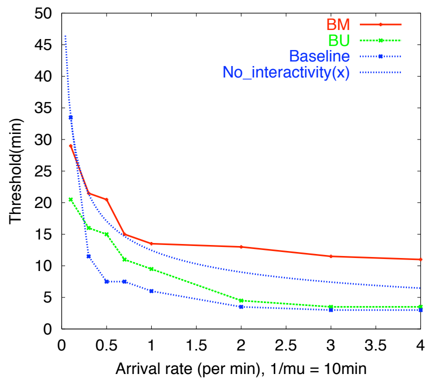
are obtained from both analysis and simulations. The results from the analysis match well with those obtained from simulation. Due to the limitation of the space, we only show the results from the analysis.

Fig. 8 shows how transmission bandwidth varies as a function of the threshold with average arrival rate λ=1.0/min and average play duration $1/\mu$ = 10 minutes. The transmission bandwidth in the absence of interactivity for the same arrival rate is also plotted. From the figure we observe the following. First, the choice of threshold has a significant impact on the performance of all schemes. Second, the optimal thresholds in BU and BM patching are close to that without interactivity, while the optimal threshold in Baseline patching is much shorter. This can be explained as follows. In Baseline patching, each FJ action leads to a unicast patch. The bandwidth requirement because of the patches forces the threshold to be short to keep the bandwidth low. In BU and BM, the patches for the interactivity force the threshold to be decreased. However the FJ action reduces the sharing of the complete stream which requires the threshold to be increased. The balance of the two factors makes the threshold to be still close to that without interactivity.

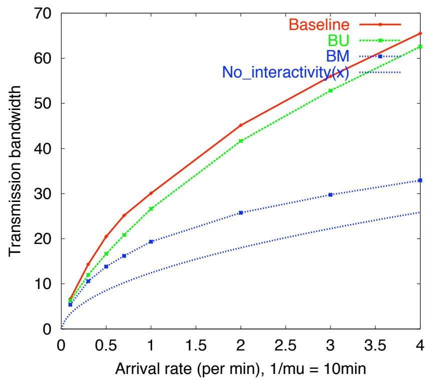
The optimal threshold and the corresponding bandwidth for Scenario 1 are shown in Fig. 9. The arrival rate λ ranges from 0.1 to 4 per minute and $1/\mu = 10$ minutes. In Fig. 9 (a), the optimal thresholds for Baseline patching are shorter than those without interactivity throughout the arrival rate. In both BU and BM, the optimal thresholds are close to those without interactivity throughout the arrival rate. In Fig. 9 (b), we observe that BU needs only half of the bandwidth required by Baseline patching, demonstrating the importance of using the contents in the client buffer. The bandwidth required for BM is close to that required for BU when the arrival rate is low. When the arrival rates is higher than 0.7 request per minute, BM uses less bandwidth than BU. The saving of BM over BU increases as the arrival rate increases. When the arrival rate is 4 requests per minute, the bandwidth required by BM is 28% less than that by BU.

B. Evaluation in Scenario 2

In Scenario 2, an exact analysis is hard to achieve because the length of the jump is a random variable. The following results are from simulations. We first set the tolerable resume latency and jump imprecision to be 0 and examine the effect of the threshold on the bandwidth requirement in the three schemes. We then examine the effect of tolerable jump imprecision and resume latency in BM patch-



(a) Threshold.



(b) Bandwidth.

Fig. 11. Threshold and transmission bandwidth as a function of the arrival rate in Scenario 2.

ing. Throughout the experiments, the length of FJ is assumed to be uniformly distributed in the range of 0 to 10 minutes.

B.1 Numerical results

Fig. 10 shows how transmission bandwidth varies as a function of the threshold with λ=1.0/min and $1/\mu$ = 10 minutes. The function without interactivity for the same arrival rate is also plotted. We observe that threshold also plays an important role here.

The optimal threshold and the required bandwidth for this scenario are shown in Fig. 11. The arrival rate ranges from 0.1 to 4 per minute and $1/\mu = 10$ minutes. In Fig. 11 (a), we observe more significant changes in the optimal threshold with the presence of FJ actions. The optimal thresholds in Baseline patching are much shorter than those without interactivity, for the same reasons as in Scenario 1. Unlike Scenario 1, when the arrival rate is high, the optimal thresholds in BU become much shorter than those without interactivity, while the optimal thresholds in BM are much larger than than those without interactivity. This is because in BU, when the arrival rate is high, the bandwidth required for the patches increases more dramatically with the range of jump distance, which forces the threshold to be shortened. In BM, since the patch is by multicast, longer patches increases the sharing of patches. Hence the optimal thresholds tends to be increased.

In Fig. 11 (b), similar to Scenario 1, the saving of bandwidth requirement for BM scheme over BU scheme increases as the arrival rate increases. The saving is 47% when the arrival rate is 4 requests per minute, much higher than in Scenario 1. This implies that using multicast leads to more savings when the average distance of FJ is long. However the savings from BU over Baseline patching is only 5% to 10% even for arrival rates higher than 1 request per minute. The reason is that optimal thresholds in BU are as short as those in Baseline patching for higher arrival rates as shown in Fig. 11(a). Therefore the benefit of using client buffer contents becomes less dramatic.

B.2 Effect of jump imprecision and resume latency in BM patching

Tolerable resume latency and jump imprecision increases the chance of finding an ongoing multicast stream. In BM patching, tolerable resume latency also increases the number of patches being merged together so that the server load can be decreased. We set the jump imprecision and resume latency to be maximumly 30 seconds. Compared with exact jump and immediate resume, when the tolerable jump imprecision δ and resume latency D are both set to be 10 seconds, the number of patches for interactivity is only reduced by 5% when the minimum bandwidth usage is achieved throughout the arrival rates. When δ and D are both set to be 30 seconds, the saving is 10%. This is because the effect of merging patches is not obvious for low arrival rates. However, when the arrival rate is high, the relative effect of jump imprecision and resume latency is not dramatic since the optimal threshold is long.

V. Conclusions and Future Work

In this paper, we explore a number of techniques for supporting interactive operations such as pause and bi-directional index jumps in patching. We present and examine three schemes: Baseline, BU and BM patching. Our evaluations, based on a combination of analysis and simulations, suggests that (i) proper choice of the threshold for controlling the frequency at which new multicasts of the complete video are started, (ii) careful use of client buffer content and (iii) using multicast for the patches after an interactive operation can result in significant savings in transmission bandwidth overheads.

In this paper, we investigate supporting interactivity in a linear media, such as a movie, where users basically go through the content sequentially. In this context, tolerable jump imprecision and resume latency should be constrained into values that are unnoticable to the users. As ongoing work, we are pursuing in supporting interactivity in orchestrated media, such as an interactive on-line course, where the content has a structure and users do not usually go through the content sequentially. In this context, we conjecture that using multicast for the patches after an interactivity gains more benefits since resume requests can be grouped together more easily. For example, in an on-line course, where the contents are separated into chapters, users usually only jumps to the beginning of a chapter.

Also in this context, users can tolerate larger resume delay. We conjecture that exploiting tolerable resume latency in this context can lead to more significant gains.

References

[1] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[2] K. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM*, September 1997.

[3] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," in *Proc. Inter. Workshop on Network and Operating System Support for Digital Audio and Video*, 1999.

[4] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. International Conference on Computer Communications and Networks*, 1997.

[5] L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, 1999.

[6] D. Eager, M. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM Multimedia '99*, October 1999.

[7] J. Dey-Sircar, J. Salehi, J. Kurose, and D. Towsley, "Providing VCR capabilities in large-scale video servers," in *Proc. ACM Multimedia*, October 1994.

[8] J. Dey, S. Sen, J. Kurose, D. Towsley, and J. Salehi, "Playback restart in interactive streaming video applications," in *Proc. IEEE Conference on Multimedia Computing and Systems*, pp. 458–465, June 1997.

[9] W. Feng, F. Jahanian, and S. Sechrest, "Providing VCR functionality in a constant quality video-on-demand transportation service," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[10] K. Almeroth and M. Ammar, "On the use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communication*, vol. 14, August 1996.

[11] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel allocation under batching and VCR control in video-on-demand systems," *Parallel and Distributed Computing*, vol. 30, pp. 168–179, November 1995.

[12] W. Liao and V. O. Li, "The split and merge protocol for interactive video-on-demand," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, October 1997.

[13] Z. Fei, M. H. Ammar, I. Kamel, and S.Mukherjee, "Providing interactive functions through active client buffer management in partitioned video broadcast," in *First International Workshop on Networked Group Communications (NGC)*, November 1999.

[14] S. W. Carter, D. D. E. Long, and Jehan-Fran cois Pâris, "An efficient implementation of interactive video-on-demand," in *Proc. 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2000.