Prefix Caching assisted Periodic Broadcast: Framework and Techniques to Support Streaming for Popular Videos[1]

Yang Guo, Subhabrata Sen, and Don Towsley
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{yguo,sen,towsley}@cs.umass.edu

Abstract

The bandwidth-intensive and long-lived nature of high quality digital video make it a challenging problem to transmit such video over the Internet. In this paper, we consider the problem of streaming a set of popular videos from a remote server to a large number of asynchronous clients, while making efficient use of network bandwidth, and still allowing clients to start instantaneous playback. We propose a scalable and flexible framework which combines proxy-based prefix caching in conjunction with periodic broadcast of the suffix of a video from the server. We develop a methodology for (i) optimally allocating the proxy buffer space among a set of popular videos and (ii) choosing appropriate prefix and suffix transmission schemes based on the principle of decoupling the suffix and prefix transmissions from each other. A greedy proxy buffer allocation algorithm is presented that minimizes the aggregate bandwidth usage on the server-proxy path. Our studies show that this approach yields a buffer allocation close to global optimal for practical settings where proxy-client path bandwidth is much cheaper than long-haul path bandwidth. When the proxy buffer is allocated to a set of videos using our allocation scheme, a total buffer space of just $5 - 20\%$ of the video repository is adequate to realize substantial reductions in the aggregate bandwidth usage on the server-proxy path. Finally, we present an integrated prefix and suffix transmission scheme such that the client only needs to listen to at most two channels simultaneously.

I. Introduction

A broad range of applications (entertainment and information services, distance learning, corporate telecasts, narrowcasts, etc.) will be enabled by the ability to stream continuous media data from servers to clients across a high-speed network. However, due to the bandwidth intensive nature (usually larger than 1 Mbps) of high quality digital video, and the long-lived nature (tens of minutes to a couple of hours) of video content, server and network bandwidths are major limiting factors in the widespread streaming of such videos over the Internet. The problem is further complicated by the fact that clients are plentiful and heterogeneous, and that clients asynchronously issue requests for the same media stream. Particularly for popular clips, a large number of client requests may arrive close together in time relative to the duration of the stream. Consequently there has been tremendous interest in developing techniques for the bandwidth-efficient distribution of video to such a client population. In this paper we explore the use of proxy-based prefix caching in conjunction with efficient transmission schemes, such as periodic broadcast and patching, for reducing the bandwidth cost of delivering popular videos to a large

[1]This research was supported in part by the National Science Foundation under NSF grants EIA-0080119, NSF ANI-9973092, ANI9977635, and CDA-9502639. Any opinions, findings, and conclusions or recomendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. The Appendix is included in the submission for the reviewers' perusal and will be excluded from the final version of the paper.

number of asynchronous clients while still allowing instantaneous playback startup. In particular, we develop techniques for allocating proxy buffer space across a set of videos, and determining the appropriate proxy-based and server-based transmission scheme, respectively, for each video such that the above goal is reached.

A. Background

Service providers can reduce response time, server load, and network traffic by deploying proxy caches. However, video files can be very large, particularly for high quality long duration videos, and techniques that cache entire objects [1, 2] are not appropriate for such videos. A number of caching strategies have been proposed in recent years [3–5] that cache a portion of a video file at the proxy. In particular, storing an initial prefix of the video [6] at the proxy has numerous advantages, e.g., shielding clients from delays and jitter on the server-proxy path, performing online smoothing to reduce the burstiness of Variable-Bit-Rate video without introducing additional client playback delays, and reducing traffic on the server-proxy path.

Recent research has focused on using multicast and broadcast connections to reduce server and network loads. Periodic broadcast and patching [7–13] described in more detail in Section II-A, are two approaches that have received considerable attention recently. These techniques exploit the use of multiple multicast channels to reduce network and server resource usage over the case of multiple unicast transmissions, while at the same time satisfying the asynchronous requests of individual clients and providing a guaranteed bound on playback startup latency. In periodic broadcast, the server divides a video object into a number of segments, and continuously broadcasts the segments on a set of transmission channels. Using a constant number of channels or segments, periodic broadcast can deliver video to an arbitrary number of asynchronous clients. In patching, the server only transmits video data on demand, when new clients arrive. Patching allows a client to begin playback immediately, while under periodic broadcast, a client has to wait until the beginning of the next broadcast period of the first segment to begin playback.

B. Contributions

In this paper, we explore the use of proxy-based prefix caching in conjunction with periodic broadcast for reducing the transmission bandwidth cost, while ensuring instantaneous playback startup for clients. We use periodic broadcast to transmit the suffix from the server, and either patching or a combination of patching and periodic broadcast for streaming the prefix from the proxy.

- We develop a methodology for optimally allocating the proxy buffer space among a set of popular videos and choosing appropriate prefix and suffix transmission schemes based on the principle of decoupling the suffix and prefix transmission from each other. Briefly, this methodology consists of
  - Choosing an appropriate periodic broadcast scheme for the suffix, and in the context of that scheme, determining the proxy buffer allocation and corresponding segmentation for each video, such that the aggregate bandwidth usage on the server-proxy path is minimized. *In particular, we choose the first segment size of suffix to be the same as the allocated prefix for computing the lengths of the segments in the suffix*. We shall see later that this can result in significant reductions in transmission cost.
  - Each proxy determines, based on the relative popularity of a video among clients of that proxy, an appropriate prefix transmission scheme that makes efficient use of bandwidth on the proxy-client path.
- We present a greedy algorithm to determine the allocation of the proxy buffer among the different videos such that the aggregate bandwidth usage on the server-proxy network path is minimized.

- We also present an integrated prefix and suffix transmission scheme such that the client only needs to listen to at most two channels simultaneously.

Decoupling the suffix and prefix transmissions enables different proxies to use different prefix transmission schemes, while sharing the same common suffix transmission from the server. This also allows a proxy to dynamically adapt its prefix transmission scheme to small variations in the local demand for any video without requiring changes to the global suffix transmission scheme which is shared among multiple proxies. This flexibility is especially desirable in dynamic, heterogeneous environments like the Internet, where the request rate for a particular video at a given proxy may vary with time, and the relative popularities of the videos as well as available client resources (e.g., reception bandwidth, buffer space) may vary across different proxies. We find that for practical settings where bandwidth on the local proxy-client path is significantly cheaper than the bandwidth on the long-haul server-proxy path, this approach of first determining the proxy buffer allocation that minimizes the bandwidth cost on the server-proxy path and then tailoring the prefix transmission to reduce the local bandwidth cost results in buffer allocation and end-end bandwidth cost which are close to what would be achieved under a global optimal approach that attempts to minimize the server-proxy and proxy-client bandwidth costs together.

Our studies show that for a single video, making the first segment of the suffix equal in size to the prefix can substantially reduce bandwidth usage on the long-haul path. Under our approach, caching a small to moderate prefix ($5-20\%$ of the video) can realize most of the bandwidth savings. When the proxy buffer is allocated to a set of videos using our allocation scheme, a total buffer space of just $5-20\%$ of the video repository is adequate to realize substantial reductions in the aggregate bandwidth usage on the server-proxy path. Furthermore, the choice of a particular periodic broadcast scheme does not significantly impact the number of server channels required provided that the proxy can cache around $10-20\%$ of the video repository. However, for smaller proxy buffers, more aggressive schemes, such as GDB(n) [8] with $n \geq 3$, exhibit superior performance.

This paper complements several recent works [14, 15] that combine caching with scalable transmission of continuous media. [15] combines patching with prefix caching to reduce transmission bandwidth overheads for a single video. In the context of periodic broadcast, [14] employs numerical techniques to determine the optimal proxy buffer allocation across a set of equal length videos, each with a predetermined fixed prefix size, that are delivered using periodic broadcast with a predetermined fixed segmentation of the video. Evaluations show that with the same available proxy buffer space, our scheme results in a much lower bandwidth usage on the server-proxy path than the scheme proposed in [10]. We discuss other differences in Section V-B.

The remainder of the paper is organized as follows. Section II presents the problem setting, and introduces key concepts and terminology used in the remainder of the paper. Section III presents the prefix caching assisted periodic broadcast framework. In Section IV, we give a greedy algorithm that optimally allocates the proxy buffer to different videos, and evaluate the algorithm. In Section V, we considers the impact that the choice of periodic broadcast scheme has on the bandwidth savings, and compare the prefix-caching assisted periodic broadcast with the scheme proposed in [14]. We also present a delivery scheme for bandwidth constrained clients. Finally, Section VI concludes the paper and describes ongoing work.

II. Problem setting

In its simplest form, a Streaming Content Distribution Network (Fig. 1) is a two level hierarchy, typically including a geographically distributed core of a few very large data servers connected via very high capacity links, and an an overlay network consisting of a much larger set of smaller caching servers, called *edge proxies*, distributed across the Internet, and located closer to end-clients. The data servers are the principle content

repositories and are expensive high-end servers or server farms configured with substantial storage (Terabytes), processing and IO bandwidth capacity for handling high request rates. By caching popular content at the edge proxies, and serving them locally to clients, service providers can significantly reduce response time, server load, and network traffic.

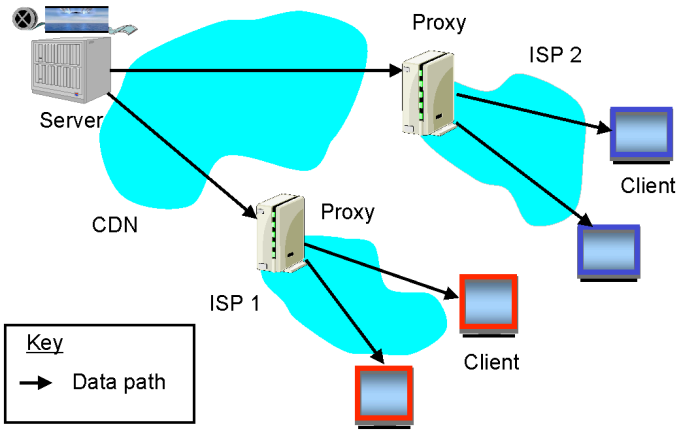


Fig. 1. An Internet Content Distribution Network

In this paper, we focus on prefix-caching based techniques for transmitting popular videos to clients over such a video distribution system. Consider a single proxy and the client population it serves. We assume that both the server-proxy network paths and proxy-client network paths are multicast capable, and that clients always request playback from the beginning of a video. The prefix is streamed to the client using a *prefix transmission* scheme. The remainder of the video (suffix) is delivered using a corresponding *suffix transmission* scheme. The prefix and suffix transmission schemes together constitute the transmission scheme for a video for the clients of a particular proxy. The following are some desirable properties of such a distribution scheme :

- *Low startup delays*: Clients should be able to start playback with low delays.
- *Bandwidth Scalability*: The distribution scheme should make efficient use of the end-end network bandwidth.
- *Flexibility*: The distribution scheme should be sufficiently flexible to accommodate inherent heterogeneities in the system. For example, the relative popularities of the videos may vary across proxies, and the actual request rate for a video may vary with time at a given proxy, and clients in different ISPs may have different resource (reception bandwidth, buffer space) constraints. A flexible distribution scheme would be able to to addresses the particular needs of different local client population, while still making efficient use of the server-proxy and proxy-client network bandwidths.

In Section III we present a methodology and techniques for determining the appropriate prefix transmission scheme, suffix transmission scheme, and corresponding prefix buffer allocation for each video that exhibits the above properties.

We next provide a formal model of the system, and introduce notation and key concepts that will be used in the remainder of the paper. Table I presents the key parameters in the model. We consider a multimedia server with a repository that includes K popular videos. We assume the videos are Constant-Bit-Rate (CBR) and have

Parameter	Definition
K	Number of popular videos
l_i	Length of video i (sec.)
B	Proxy cache size (sec.)
x_i	prefix of video i cached at the proxy (as a fraction of the video length)
c_i	Number of server channels for video i

TABLE I
PARAMETERS IN THE MODEL.

the same playback rate. Assume the length of video i is l_i seconds, and that the proxy can store up to B seconds worth of video. Assume the first $x_i l_i$ seconds of video i are cached at the proxy, $0 < x_i \leq 1$. Henceforth x_i will be referred to as the prefix size of video i. Note that the videos cached at the proxy cannot exceed the storage constraint of the proxy, that is, $\sum_{i=1}^{K} x_i l_i \leq B$.

An imporant goal in our scheme is to minimize the transmission bandwidth requirement on the server-proxy path, aggregated over all the videos in the repository, i.e., the metric $C = \sum_{i=1}^{K} c_i$, where c_i is the number of server channels for video i when a prefix of x_i is cached at the proxy. All channels are of equal bandwidth. In the rest of the paper, unless otherwise stated, we shall use the term *transmission cost* to refer to this metric. Once the prefix allocation (specified by the storage vector $\bar{x}$) is known, the individual video prefixes are then stored at the proxy, in advance of client requests, based on the proxy buffer allocation. In Section IV we present an algorithm to determine the optimal proxy cache allocation that minimizes the transmission cost.

For ease of exposition, we assume that the propagation delay is zero in the rest of the paper. However, our results are easily extended to account for a bounded propagation delay. We also assume the client has sufficient buffer space and network bandwidth to accommodate an entire video clip. Finally, note that we focus on a single server and a single proxy for simplicity of exposition. However our results apply directly to Content Distribution Networks with multiple proxies with identical storage capabilities, where each proxy serves a different set (no overlapping) of clients with potentially different local population sizes and different local demands for a set of videos. We next provide a brief background on periodic broadcast and patching schemes.

A. Periodic broadcast

We use periodic broadcast for suffix delivery. In periodic broadcast, the initial segments are repeated more frequently than later ones. This has the benefit of reducing both the client startup delay and the transmission bandwidth requirements. These schemes ([16] has a nice literature overview) can be broadly classified into three categories (1) Schemes that divide a video into segments of increasing length that are transmitted over channels of the same bandwidth [7, 8, 17]. (2) Schemes that divide the video into equal sized segments, and transmit earlier segments over higher bandwidth channels than later ones [18]. (3) Hybrid schemes that the above two approaches [19]. In this paper we focus on schemes belonging to the first category. However, our approach should be applicable to schemes in the other two categories.

We associate with each segment an integer $f(n)$, s.t. the length of the i-th segment is $f(i)l/\sum_j f(j)$, and $f(1)$ is taken to be one. We shall consider some representative schemes from the first category which use the following segment size progressions: Skyscraper: [1,2,2,5,5,12,12, $\cdots$], Dynamic skyscraper: [1,2,2,4,4,8,8,

$\cdots$], GDB(3): [1,2,4,6,8,12,16,$\cdots$], GDB(4): [1,2,4,8,14,24,40,$\cdots$], GDB(5): [1,2,4,8,16,30,56, $\cdots$], GDB(6): [1,2,4,8,16,32,62, $\cdots$].

B. Patching

Patching promotes sharing by allowing a later arriving client to join an on-going multicast transmission started by an earlier request, and the server transmits the initial missing part directly to the client. Several authors [11, 20, 21] have proposed to use a threshold to control the frequency at which new multicast sessions are started. The threshold is determined by minimizing the average bandwidth requirement when the clients arrive according to a Poisson process. Catching [22] combines periodic broadcast with patching in order to realize both zero latency client playback, as well as the resource efficiency of periodic broadcast. Catching use patching to deliver the first segment of the periodic broadcast. Since patching outperforms catching when the client request rate is low while catching is more suitable for high request rates, [22] uses a policy called selective catching to determine, based on the request rate of a video, whether to use patching or catching.

III. Prefix caching assisted periodic broadcast

In this section we describe the prefix-caching assisted periodic broadcast framework, and present some key intuitions and design principles guiding our approach. Our goal is to design a scalable, flexible framework for delivering video to clients with little or no startup delay. We first focus on the caching and transmission schemes for a single video and then consider the multiple-video resource allocation problem.

A. Distributing a single video

In our framework, a proxy is responsible for serving a prefix of the video, while the source server is responsible for serving the suffix. A salient feature of any periodic broadcast scheme is that the total number of server channels is constant for a given playback delay, independent of the number of clients requesting the video. This makes periodic broadcast very bandwidth efficient when arrival rates are high. We therefore choose Periodic Broadcast for transmitting the suffix of the video. In this work, we focus on the class of periodic broadcast schemes that use segments increasing in length and are transmitted over equal bandwidth channels.

The proxy uses either catching or patching to deliver the prefix - both schemes can provide instantaneous playback, and the scheme which incurs the lowest transmission cost is selected. Note that in the absence of prefix caching, the length of the first segment would determine the worst case startup delay for any client. In our scheme, because a separate prefix transmission scheme is used for the initial part of the video, we can afford to have a large first segment for the suffix, while providing instantaneous playback. Hence it becomes possible to reduce the number of channels by making the first segment of the suffix as large as possible. For example, with a small 30 sec. initial segment, a 100 min. video needs roughly 12 channels under Skyscraper broadcast (Fig. 2). With a larger 10 minute first segment, the number of channels reduces to 4. In addition, the reduction in number of channels is not linear in the size of the first segment. Instead, the reduction is large for small to moderate initial segment sizes and gradually approaches zero. Other periodic broadcast schemes, such as GDB(i), dynamic skyscraper, etc., show similar behavior. This trend argues persuasively for using a large first segment.

The length of the first segment, however, cannot be arbitrarily large and an upper constraint is imposed by the need to guarantee starvation-free playback to the client. To avoid starvation when switching from playing back the prefix to playing the suffix, a client should start to receive the suffix before the end of prefix is reached. To ensure this, the first segment of the suffix cannot be larger than the prefix. We therefore select the length of

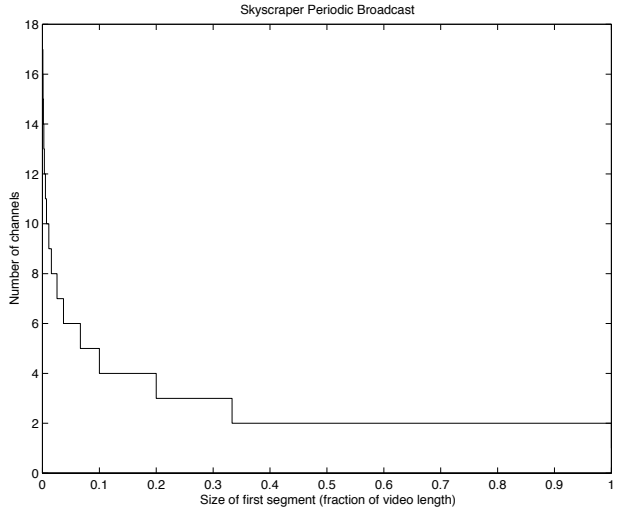


Fig. 2. Number of channels required given the size of first segment (skyscraper scheme)

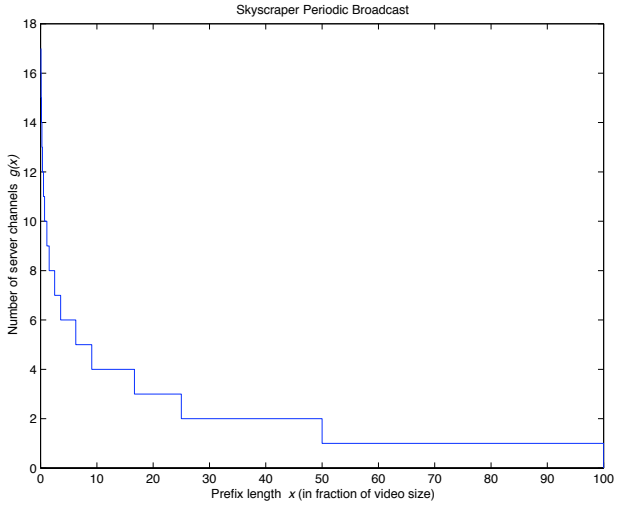


Fig. 3. Number of server channels vs. prefix size

the first segment to equal the prefix length. Fig. 3 depicts the corresponding tradeoff between the prefix buffer allocation and associated long-haul transmission cost, when skyscraper broadcast is used to deliver the suffix.

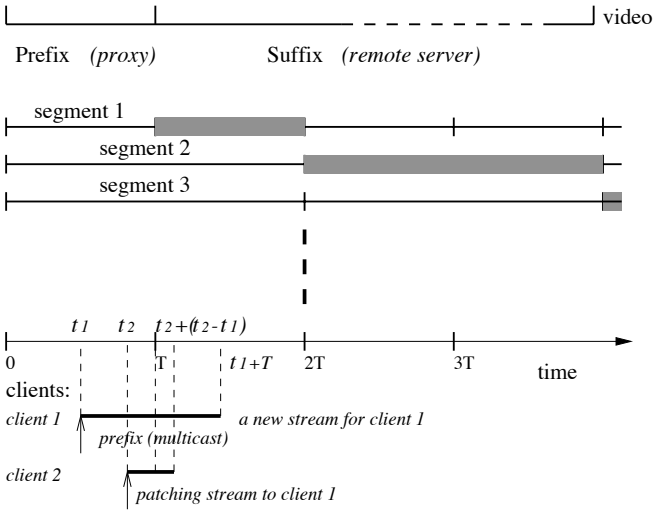


Fig. 4. Prefix-caching assisted periodic broadcast

Fig. 4 illustrates an example of prefix caching assisted periodic broadcast. The suffix is partitioned into several segments (we show the first three segments) which are periodically broadcast. In this example, the proxy uses patching to deliver the prefix. Client 1 arrives at time t_1, the proxy initiates a new stream to transmit the prefix to it. At time T, client 1 begin to receive the suffix starting from segment 1 (the shaded segments). Client 2 arrives at time t_2 while the proxy is still multicasting the prefix. Suppose that $t_2 - t_1$ is less than the patching threshold, client 2 immediately taps into the ongoing multicast. Simultaneously the proxy sends the patch, $[0, \quad t_2 - t_1]$, to client 2. Notice that client 2 will also begin to receive suffix at time T and receive the shaded segments at the same time as client 1. There is a period of time during which the client simultaneously receives both the prefix

and the suffix. Thus the number of channels the client needs to listen to, in the worst case, is the sum of the channels required to obtain the prefix and suffix.

B. Distributing a set of videos

We now turn our attention to the case where there are several popular videos in the repository. If we adopt the scheme described above for each video, *the next question is how to allocate a limited proxy buffer to all of the videos such that aggregate transmission costs are minimized*. From Fig. 3 we can see that for a given video, a large prefix reduces long-haul bandwidth requirements. We propose the following approach to allocate the proxy buffer and determine the prefix transmission scheme.

Step 1. Allocate the proxy buffer so as to minimize the aggregate server channel (long-haul bandwidth) requirement to support all videos.

Step 2. The proxy uses patching, or catching, to deliver the prefix based on the local client request rate.

Our design principle is to decouple the local (prefix) transmission from the long-haul path (suffix) transmission. First, we determine a buffer allocation that minimize the bandwidth consumed on the long haul path from the server to the proxy, without considering the delivery cost on the local path. Then each proxy tries to use an appropriate transmission scheme for the prefix that minimizes the average bandwidth on the proxy-client path. The rationale behind this approach is as follows:

- Typically it is more expensive to transmit a unit of data from the remote server to the proxy, than from the proxy to the client. So reducing the long-haul bandwidth usage is important.
- The local transmission cost is closely related to the local client request rate, which is not easy to obtain and difficult to predict, making the accurate estimation of local delivery cost practically infeasible. Furthermore, there can be tremendous heterogeneity among proxies. Even for a hot video, it may not be popular at every local area. It is better to let the proxy take the responsibility to decide how to deliver the prefix locally.
- In section IV-C, we shall show that the total bandwidth used to deliver a single video, including both long-haul and proxy-client path network bandwidth, is close to the long haul bandwidth requirement, if the cost of transmitting a unit of data locally is relatively small (compared to the long haul cost). Thus our decoupling approach yields a solution that is close to the globally optimal solution in many practical settings, e.g., proxy is much closer to user than remote server, while still enjoying the benefits of simplicity, flexibility, and robustness against the presence of incomplete request rate information.

In summary, prefix caching assisted periodic broadcast is a scalable and flexible framework with which to support a hot video streaming service with zero playback delay. The scalability derives from the combination of periodic broadcast with the proxies, as well as from the minimization of the long-haul server-proxy path bandwidth. The flexibility comes from decoupling the prefix and suffix transmission from each other, and the use of selective catching for the delivery of prefix. In the following section, we develop and evaluate a buffer allocation scheme that minimizes the long-haul transmission bandwidth.

IV. Proxy buffer allocation

As described in the previous section, we try to place as large a prefix of the video as possible at the proxy so as to minimize the number of server channels/long-haul path bandwidth. In the following, we present a greedy buffer allocation algorithm that minimizes the aggregate server-proxy path network bandwidth usage. We then evaluate the algorithm using a set of videos of varying lengths.

A. Optimal proxy buffer allocation algorithm

Denote by $s(i)$ the minimum prefix size (as a fraction of video length) for broadcasting a suffix using i server channels, and l_k the length of video k. We use the subscript to distinguish the videos. The segment size progression $\{f(n)\}$ is determined by the choice of periodic broadcasting scheme. The video consists of two parts: prefix and suffix. Recall that the first segment of the suffix is chosen to be equal in length to the prefix. If we assume i server channels are used to deliver the suffix, then we have:

$$\begin{aligned} s(i)l_k &+ \sum_{j=1}^{i} f(j)s(i)l_k = l_k, \\ s(i) &= 1/(1+\sum_{j=1}^{i} f(j)) \qquad \text{for } i \geq 1. \end{aligned} \tag{1}$$

We let $s(0) = 1$. Because $f(n)$ is a non-decreasing function, $s(i)$ is a convex function in i. Let $g(x)$ be number of server channels needed with prefix size of x. By definition, we have

$$g(x) = i \qquad \text{if } s(i+1) \leq x < s(i)$$

The proxy buffer allocation problem is to allocate the proxy buffer to a set of videos, given that the video lengths are known in advance, so as to minimize the total number of server channels (long-haul path bandwidth). The buffer allocation problem can be formulated as follows:

$$\begin{aligned} \min_{\{x_k\}} \quad & \sum_k g(x_k) \\ \text{subject to} \quad & \sum_k x_k l_k \leq B \\ & 0 < x_k \leq 1 \end{aligned} \tag{2}$$

where $k \in \{1, 2, \cdots, K\}$. Prefix of video k, x_k, cannot be equal to 0. Otherwise there is no prefix for video k to do patching or catching, and instantaneous playback cannot be achieved.

A greedy algorithm to solve the buffer allocation problem (2) is illustrated in Fig.s 5 and 6 (see Appendix for the proof). Here A is the set of videos, $L = \{l_k \mid k \in A\}$, N is the number of server channels used, and $C = \{c_k\}$. Denote by $\delta^+(i)$ the increase of prefix size to multicast video in $(i-1)$ server channels rather than i channels, and $\delta^-(i)$ the decrease of prefix size to multicast video in $(i+1)$ channels rather than i channels. We have:

$$\begin{aligned} \delta^+(i) &= s(i-1) - s(i), \\ \delta^-(i) &= s(i) - s(i+1). \end{aligned}$$

Both $\delta^+(i)$ and $\delta^-(i)$ are non-negative because $s(i)$ is non-increasing function. As illustrated in Fig. 5, in step 1 an initial point where all videos use the same amount of buffer is generated. Step 2 minimizes the proxy buffer usage without changing the aggregate number of server channels. In step 3, the leftover buffer is allocated to the videos so that the long-haul bandwidth is minimized. Specifically, at step 2 (Fig. 6), the algorithm determines the following two quantities at each round: 1) the video k, that needs the least extra amount of buffer, $\delta_k^+(c_k)$, in

Proxy_Buffer_Allocation(A, L, B)
Step 1. Generate a feasible initial point.
$c_k = g(B/(Kl_k))$
$N = \sum_k c_k$

Step 2. Minimize buffer usage without changing the number of channels used
Minimize_Buffer_Usage(A, L, C, N)

Step 3. Allocate the leftover buffer while minimizing the number of server channels
$B \leftarrow B - \sum_{k \in A} s(c_k) l_k$
$\delta_k^+(c_k) \leftarrow \min_{i \in A} \delta_i^+(c_i)$
while $(\delta_k^+(c_k) < B)$
    $B \leftarrow B - \delta_k^+(c_k)$
    $c_k \leftarrow c_k - 1$
    $\delta_k^+(c_k) \leftarrow \min_{i \in A} \delta_i^+(c_i)$

Fig. 5. **PBA Algorithm:** Pseudocode for allocating the proxy buffer to minimize the aggregate number of server channels

Minimize_Buffer_Usage(A, L, C, N)
initialize $\delta_i^+(\cdot)$ and $\delta_i^-(\cdot)$
$\delta_k^+(c_k) \leftarrow \min_{i \in A} \delta_i^+(c_i)$
$\delta_j^-(c_j) \leftarrow \max_{i \in A} \delta_i^-(c_i)$
while $(\delta_k^+(c_k) < \delta_j^-(c_j))$
    $c_k \leftarrow c_k + 1$
    $c_j \leftarrow c_k - 1$
    $\delta_k^+(c_k) \leftarrow \min_{i \in A} \delta_i^+(c_i)$
    $\delta_j^-(c_j) \leftarrow \max_{i \in A} \delta_i^-(c_i)$

Fig. 6. **MBU Algorithm:** Pseudo code for minimizing the proxy buffer usage while keep the total number of server channels unchanged.

order to use one less server channel; and 2) the video j, that can gives up maximum amount of buffer, $\delta_j^-(c_j)$, by using one more channel. It then compare $\delta_k^+(c_k)$ with $\delta_j^-(c_j)$. If $\delta_k^+(c_k)$ is less than $\delta_j^-(c_j)$, it will swap one channel from video j to video k, reducing the proxy buffer size while maintaining the aggregate number of server channels unchanged. The algorithm stops until no channel swapping can be performed. At this point, the total amount of buffer is minimized given N server channels used. In step 3, the algoirthm iteratively allocates the left-over buffer space to the video which requires least amount of buffer to use one less server channel. The minimization of long-haul bandwidth usage is achieved at the end.

B. Numerical examples

It has been observed [23] that most of the requests (about 80%) is for a small set of (10 to 20) popular videos. Thus we use a set of 20 videos in the following examples. We expect similar conclusions to hold for different number of videos.

We first consider a set of equal length videos. Suppose we want to support 20 videos of 100 mins each. Every

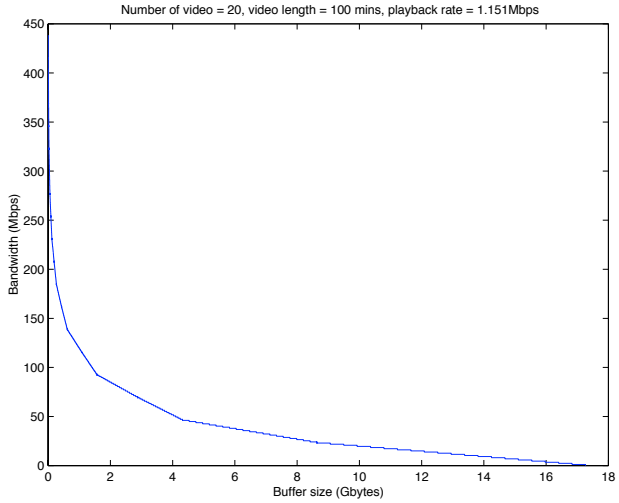


Fig. 7. Bandwidth vs. buffer size (a set of 20 videos)

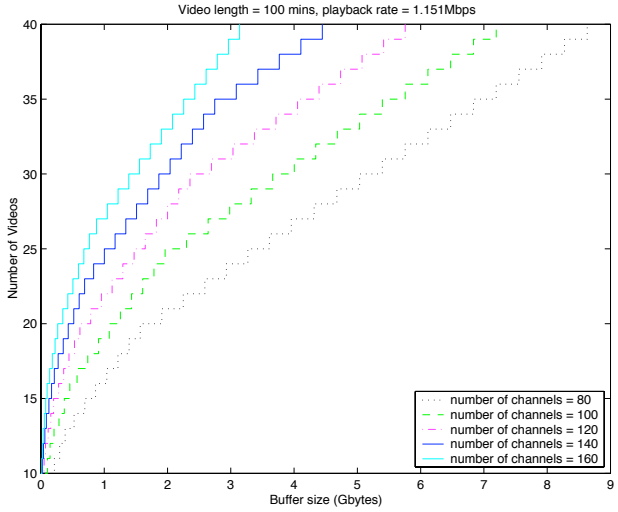


Fig. 8. Number of videos vs. buffer size

video has a playback rate of 1.15Mbps (VCD NTSC standard). Fig. 7 shows the long-haul path bandwidth required as a function of the proxy buffer size. Since all videos are of the same length, the buffer is distributed among videos so that the number of server channels for different videos do not differ by more than one channel. The curve decreases initially rapidly, and then more gradually with increasing proxy buffer size. The behavior suggests that most of the bandwidth saving can be achieved in the buffer region of 1 Gbytes - 4 Gbytes, which is about $6-23\%$ of the total video repository.

Fig. 8 depicts the number of videos that can be supported as a function of the buffer size, given a number of long-haul path channels (each channel has bandwidth equal to the playback rate 1.15Mbps). For instance, a proxy buffer of 2Gbytes is needed to support 20 videos with 80 remote server channels, and a proxy buffer of 10Gbytes is needed to support 40 videos with 80 remote server channels. Furthermore, if there are 160 channels for 40 videos, only 3.6Gbytes of buffer is needed.

We further investigate the proxy buffer allocation with a set of videos of different sizes. We choose a set of 20 videos with lengths 20min, 30min, ..., 210min. The videos are indexed in the order of their length.

Fig. 9 presents how the proxy buffer is allocated among these videos. We represent the prefix size as a fraction of its video length. One observation is that a larger fraction of the shorter videos are stored at the proxy. Furthermore, when we increase the proxy size from 5%, to 10%, 20%, and 30% of the total video repository, this trend becomes more significant. This phenomenon is understandable since at each iteration, the proxy buffer allocation algorithm always allocates buffer to the video that reduces the objective function the most. We observe from Fig. 3 that the number of server channels is determined by the prefix in terms of the fraction of its video size. Thus the same amount of buffer means more to a short video than a long one.

We also compare the optimal buffer allocation scheme with a naive buffer allocation scheme, where the buffer is evenly divided among the videos without considering the video size. The optimal buffer allocation always outperform the naive scheme; and the difference becomes larger as the prefix increases. In the region where buffer is about 10% to 20% of the video size, the optimal allocation scheme reduces the remote server channels by about 18% over the naive scheme.

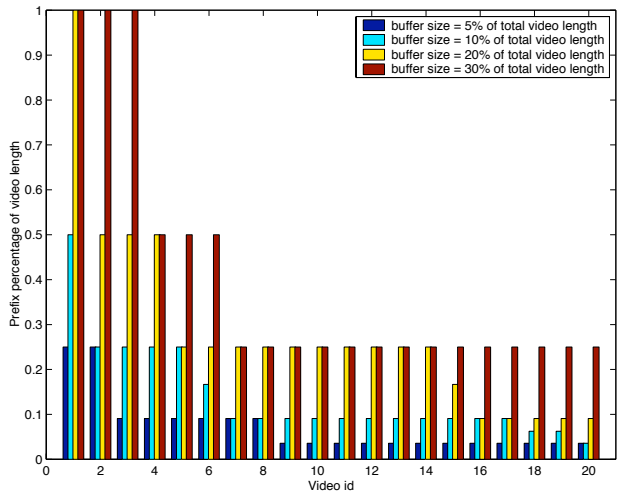


Fig. 9. Prefix percentage of video length (a set of 20 video with different length)

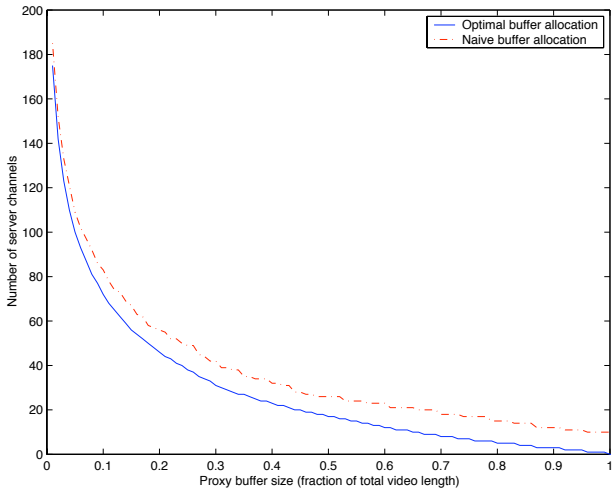


Fig. 10. Comparison of buffer allocation algorithm: optimal buffer allocation vs. even buffer allocation

C. Global optimization vs. decoupling approach

In Section III, we proposed a hierarchical two-step approach for determining the optimal proxy buffer allocation and appropriate prefix and suffix transmission schemes for each video. In the first step, we optimally allocate the proxy buffer to a set of videos ignoring the local delivery cost of prefix, i.e., we choose $\{x_k\}$ so as to minimize $\sum_k g(x_k)$. Ideally we may want to perform a global optimization in order to lower the total cost of both the average long-haul path bandwidth and average local bandwidth, that is:

$$\min_{\{x_k\}} \sum_k \{g(x_k) + \beta C_{local}(x_k, \lambda_k)\}$$

where β is a weight placed on the local bandwidth and $C_{local}(x_k, \lambda_k)$ is the average bandwidth for both prefix and suffix transmission over the local network path, given prefix size x_k and request rate λ_k for video k. Since the proxy is typically much closer to the clients than the remote server, β is expected to be much smaller than one. Solving this global optimization problem results in the minimum global cost, but requires client request rate information for each video at every proxy. Moreover, it is computationally intractable. Let us see how our approach compares with the globally optimized solution in a practical setting.

We consider a single video first. Fig. 11 depicts the average bandwidth requirement vs. prefix size for a 100 min. video for different values of β. We use the skyscraper scheme to deliver the suffix, and selective catching for prefix transmission. We choose a request rate of 60/min. We only plot the points at which the number of server channels decreases (the cost for the prefix size in between two adjacent jump points will increase. The reason is that the number of long-haul path channels remains the same, while the prefix delivery cost increases.). The optimal buffer allocation must occurs at these discrete points.

One observation is that the cost function is monotonically decreasing so long as $\beta < 1$, i.e., the larger the fraction of the video cached at the proxy, the less bandwidth is needed to deliver the video. This supports our intuition that we should make full use of the proxy buffer in most practical settings.

Another observation is that the total cost increases quickly once β becomes larger than 0.1, which suggests

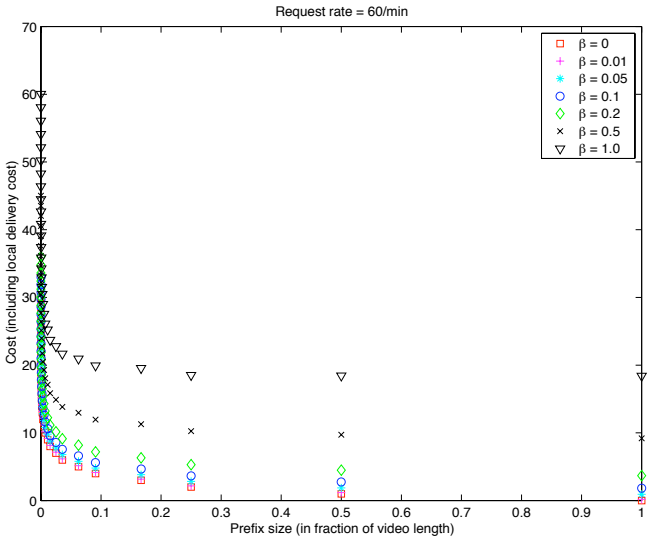


Fig. 11. Average bandwidth requirement vs. prefix size for a video of 100 mins

that, if possible, we should place the proxy as close to the end clients as possible.

If $\beta = 0$, only long-haul path bandwidth is considered, corresponding to the objective function used in our approach. The costs with $\beta \in [0.01,\ 0.05]$ are close to the cost for $\beta = 0$. Hence if β is in the the range $[0,\ 0.05]$, the cost of the buffer allocation obtained by our greedy algorithm will be close to the global optimal solution, while being simple, flexible, and robust to the uncertainties about the request rate.

We also computed the costs for request rates varying from 1/min to 500/min. For the sake of the brevity, these are not presented here. We also observe that the smaller the request rate, the larger range of β have the cost function close to that of $\beta = 0$. But even for the request rate of 500/min, the cost function with β equal to 0 and 0.01 are close.

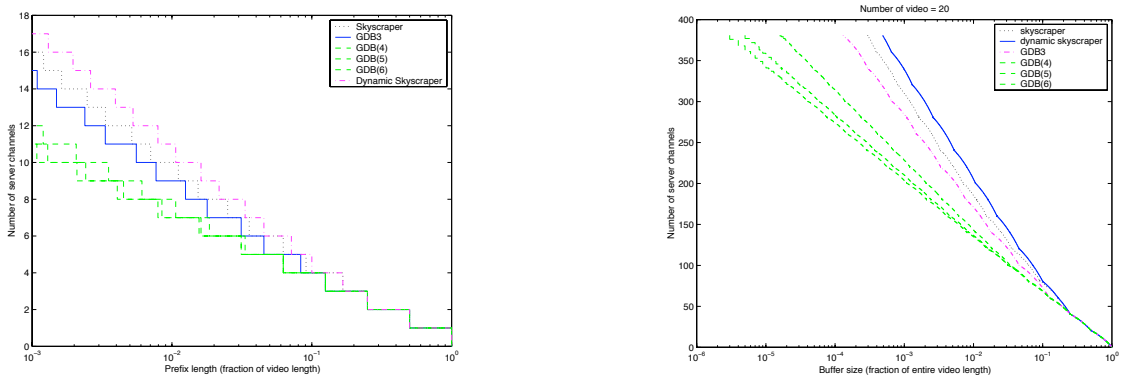
V. Discussions

In this section, we first investigate the impact that the choice of periodic broadcast scheme has on the bandwidth savings on the server-proxy path. We then compare the prefix caching assisted periodic broadcast to the scheme proposed in [14]. Last, we present an integrated prefix and suffix transmission scheme such that client only needs to listen to at most two channels simultaneously.

A. The choice of periodic broadcast scheme

As illustrated in Section II, the periodic broadcasting schemes differ in their segmentation series. The segment progression of more aggressive schemes grow faster. However more aggressive scheme usually require the client to be able to simultaneously listen to more channels and sometimes require larger client buffers. In the following we investigate the impact that the choice of periodic broadcasting scheme has on the server-proxy path bandwidth saving in our framework.

Fig. 12 depicts the number of server channels vs. the prefix size for several periodic broadcasting schemes, such as skyscraper, GDB3, GDB(4), GDB(5), GDB(6), and dynamic skyscraper. Note that if the prefix size is around 20% of the video size, then all schemes need the same number of server channels, more specifically, three server channels. The reason for this is that the first three segment sizes are very similar across all the the periodic



(a) single video case

(b) a set of 20 videos

Fig. 12. Comparing periodic broadcast schemes

broadcast schemes, and when the prefix size is around 20%, under our approach of having the first segment of the suffix equal to the prefix, the suffix has three segments. The graphs show that a proxy buffer size of around $10-20\%$ of the video repository, the difference between the different schemes is small. Only for much small buffer sizes, more aggressive broadcast schemes tend to perform better. We thus recommend choosing the most aggressive scheme that accommodates the client's buffer and bandwidth constraints.

B. Comparison with optimized regional caching

Optimized regional caching (ORC) [14] uses dynamic skyscraper broadcast to initially segment the video, and categorizes some segments as leading segments. The proxy is allowed to cache the entire video, the entire leading segments, or nothing. An analytical model is then used to determine the cache allocation that minimizes the end-end delivery cost, assuming that the client request rate information at all the proxies is known beforehand.

We next compare our approach with ORC from two main perspectives: (1) proxy buffer and network bandwidth usage, and (2) the objective function used in the optimization model.

Fig. 13 depicts the number of server channels required given a fixed amount of proxy buffer allocated to a 100 min long video. For fairness, we also use dynamic skyscraper broadcast for suffix delivery in our approach. For optimized caching, we choose the first segment size to be 0.1min, 1 min, and 1.5 mins respectively. Recall that in [14], the entire leading segments have to be cached in proxy. Here, for the sake of comparison, as many of the segments as possible are placed in the proxy buffer. The number of server channels is equal to the number of channels used to deliver the part that cannot be cached in the proxy.

Prefix-caching assisted periodic broadcast outperforms ORC for almost all proxy buffer sizes. For instance, when the proxy buffer is around 20% of the video size, our approach requires only $50-60\%$ of the long-haul path bandwidth required for ORC. From another perspective, in order to require no more than 5 server channels, the proxy would require to store just 7% of the video using our approach, and 17% of the video at the proxy under ORC. We do find that when the first segment for ORC is large (more than 5 min. long), the bandwidth usage is closer to our scheme. However, under ORC, the client delay is determined by the length of the first segment, and hence such a large first segment would result in long client delays.

We next compare the objective function used in these two works. In prefix caching assisted periodic broadcast, our goal is to minimize the number of server channels (or long-haul path bandwidth), where no client request rate information is needed. The heterogeneity of the local requests is handled by the proxy and the local prefix

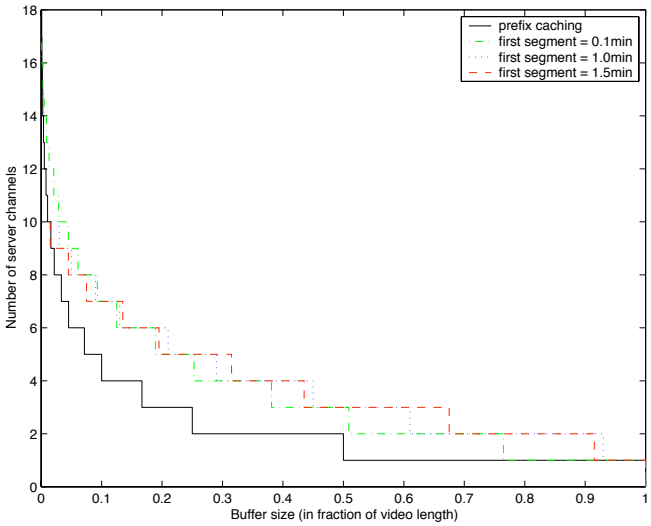


Fig. 13. Comparison of prefix caching assisted periodic broadcast vs. dynamic skyscraper scheme

delivery scheme. In contrast, the ORC approach assumes prior knowledge of client request rates at the different proxies, and incorporates this information in the objective function whose goal is to minimize the aggregated end-end bandwidth cost. In contrast, the optimization problem formulated in this paper is more from a CDN's perspective - trying to deliver the video as efficient as possible through long-haul path, and allowing the proxy to account for the heterogeneity locally. In addition, our solution is close to the global optimal for many practical settings, and more robust to the uncertainty of the local information, as discussed before.

C. Bandwidth constrained client

It is often the case that the client has limited reception bandwidth. For the end user connected to the network via DSL or cable modem, the access speed is limited - e.g., DSL offers around 760 Kbps. Despite the advances in coding techniques, the playback rate of a reasonably good quality video is at least 300 Kbps. Therefore the home user may only be able to receive from at most two channels simultaneously. In the following we present an integrated patching and periodic broadcast scheme that requires a client to listen to no more than two channels simultaneously.

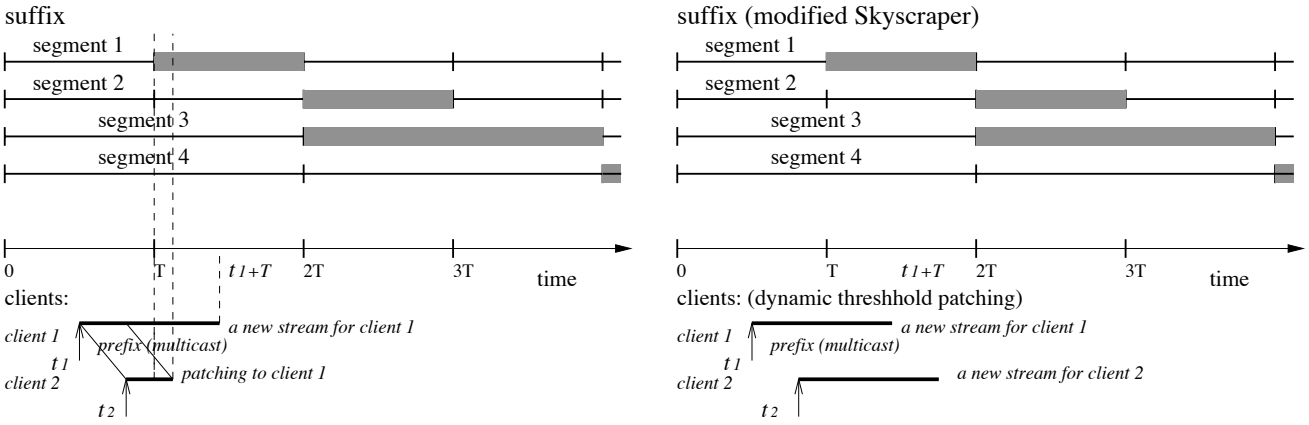


Fig. 14. Left: skyscraper for suffix and patching for prefix. Right: modified skyscraper and dynamic threshold patching.

One observation of prefix-caching assisted periodic broadcasting is that the receptions of prefix and suffix overlap. For example, for client 1 in Fig. 14 the receptions overlap during $[T, \ t_1 + T]$. During this period, a client with a two-channel constraint must be able to use only one channel to receive the prefix and only one channel to receive the suffix. We next modify the suffix broadcast scheme, and propose a new patching scheme for the prefix, called *dynamic threshold patching* to meet the constraints on the number of client channels.

First we choose a broadcast scheme which requires a client to listen to at most two channels simultaneously, e.g., Skyscraper broadcasting. We then add a new first segment, equal in size to the original first segment. For skyscraper, the progression changes from $1, 2, 2, 5, 5, \cdots$ to $1, 1, 2, 2, 5, 5, \cdots$. The benefit of having first two segments be of the same size is that a client can always receive these segments sequentially.

In dynamic threshold patching, the client is patched to an earlier stream only if (1) the arrival is within the normal threshold, and (2) either the patching stream itself or the earlier stream patched to does not overlap the first segment of the suffix. For example, client 2 cannot patch to client 1's stream since the patching stream will overlap the first segment (see right part of Fig.14). A new stream will be initiated for client 2.

VI. Conclusions and future work

In this paper, we explore the use of proxy-based prefix caching in conjunction with periodic broadcast for reducing transmission bandwidth costs, while ensuring instantaneous playback startup for clients. We proposed a prefix caching assisted periodic broadcast scheme and developed a methodology for (i) optimally allocating the proxy buffer space among a set of popular videos and (ii) choosing appropriate prefix and suffix transmission schemes based on the principle of decoupling the suffix and prefix transmission from each other. We present a greedy algorithm to determine the allocation of the proxy buffer among the different videos such that the aggregate bandwidth usage on the server-proxy network path is minimized. The framework is scalable, flexible, and supports popular video streaming service with instantaneous playback. The scalability derives from combining periodic broadcast with the prefix caching, as well as from the minimization of the long-haul path bandwidth. The flexibility is the result of our decoupling design principle, and the approach of allowing each proxy to determine the appropriate local prefix transmission scheme for each video. To handle bandwidth constrained clients, we present an integrated prefix and suffix transmission scheme such that the client only needs to listen to at most two channels simultaneously.

We are further exploring this research space along a number of directions. In this paper, we only consider one group of periodic broadcast scheme using increasing size of segments and logical channels of the same bandwidth. We are extending the current framework to accommodate other types of periodic broadcast schemes. Another interesting path involves extending our work to support VBR video delivery. Last, we are developing a server-proxy-client testbed [24] which we shall use to explore our framework in a realistic network setting.

References

[1] S. Williams, M. Abrams, C. R. Standbridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World Wide Web documents," in *Proc. ACM SIGCOMM*, pp. 293–305, August 1996.

[2] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in *Proc. USENIX Symp. on Internet Technologies and Systems*, pp. 193–206, December 1997.

[3] M. Kamath, K. Ramamritham, and D. Towsley, "Continuous media sharing in multimedia database systems," in *Proc. of 4th International Conference on Database Systems for Advanced Applications (DASFAA'95)*, April 1995.

[4] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based caching for Web servers," in *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[5] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A network conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. IEEE INFOCOM*, April 1998.

[6] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM*, April 1999.
[7] K. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM*, September 1997.
[8] L. Gao, D. Towsley, and J. Kurose, "Efficient schemes for broadcasting popular videos," in *Proc. Inter. Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.
[9] D. Eager and M. Vernon, "Dynamic skyscraper broadcasts for video-on-demand," in *Proc. 4^{th} Inter. Workshop on Multimedia Information Systems*, September 1998.
[10] D. Eager, M. Ferris, and M. Vernon, "Optimized regional caching for on-demand data delivery," in *Proc. Multimedia Computing and Networking (MMCN '99)*, January 1999.
[11] 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.
[12] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. International Conference on Computer Communications and Networks*, 1997.
[13] K. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM Multimedia*, September 1998.
[14] D. Eager, M. Ferris, and M. Vernon, "Optimized caching in systems with heterogeneous client populations," in *Performance Evaluation, Special Issue on Internet Performance Modeling*, September 2000.
[15] S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (mcache): An adaptive zero-delay video-on-demand service," in *Proc. IEEE INFOCOM*, April 2001.
[16] A. Hu, "Video-on-demand broadcasting protocols: A comprehensive study," in *Proc. IEEE INFOCOM*, April 2001.
[17] S. Viswanathan and T. Imielinski, "Pyramid broadcasting for video on demand service," in *IEEE Multimedia Computing and Networking Conference*, vol. 2417, pp. 66–77, 1995.
[18] L. Juhn and L. Tseng, "Harmonic broadcasting for video-on-demand service," in *IEEE Transactions on Broadcasting*, vol. 43, pp. 268–271, September 1997.
[19] J.-F. Paris, S. Carter, and D. Long, "A hybrid broadcasting protocol for video on demand," in *Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99)*, pp. 317–326, January 1999.
[20] 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.
[21] Y. Cai, K. Hua, and K. Vu, "Optimizing patching performance," in *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, January 1999.
[22] L. Gao, Z. Zhang, and D. Towsley, "Catching and selective catching: Efficient latency reduction techniques for delivering continuous multimedia streams," in *Proc. ACM Multimedia*, 1999.
[23] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, pp. 112–121, June 1996.
[24] M. K. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley, "Periodic broadcast and patching services - implementation, measurement, and analysis in an internet streaming video testbed," Tech. Rep. TR00-56, Department of Computer Science, University of Massachusetts Amherst, 2000.
[25] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*. The MIT Press, 1988.

APPENDIX

Note: The Appendix is included in the current submission for the reviewers' perusal and will be excluded from the final version of the paper.

Observe that $g(x)$ is a step function (see Fig. 3). If $\{x_k^{'}\}$ is an optimal solution of the above problem, then $\{x_k^*\}$, where $x_k^* = \inf\{x_k \mid g(x_k) = g(x_k^{'})\}$ (left jump point closest to $x_k^{'}$ in $g(\cdot)$), is also an optimal solution. Therefore the proxy buffer allocation problem can be reduced to the following problem.

$$\begin{aligned} \min_{\{x_k\}} \quad & \sum_k g(x_k) && (3)\\ \text{subject to} \quad & \sum_k x_k l_k \le B \\ & x_k \in \{s(i) \mid i = 0,1,2,\cdots\}, && k = 1,2,\cdots,K. \end{aligned}$$

where $\{x_k\}$ only takes discrete values. In the following we will describe a greedy algorithm to solve the above problem. Define the buffer minimization problem given the total number of server channels is N, $B(N)$, as follows:

$$\begin{aligned} B(N): \quad & \min \sum_k x_k l_k && (4)\\ \text{subject to} \quad & \sum_k g(x_k) = N \\ & x_k \in \{s(i) \mid i = 0,1,2,\cdots,N\}, && k = 1,2,\cdots,K. \end{aligned}$$

Denote by $\{x_k^*(N)\}$ the optimal solution of problem $B(N)$. Thus the total buffer size using N server channels is minimized at $\{x_k^*(N)\}$. We have the following theorem.

Theorem 1: If N^* is the minimum number of server channels that satisfies $\sum x_k^*(N^*)l_k \le B$, i.e.

$$N^* = \min_N\{\sum x_k^*(N)l_k \le B\},$$

then $\{x_k^*(N^*)\}$ is the optimal solution of proxy buffer allocation problem (3), and N^* server channels are required to provide streaming media service in prefix caching assisted periodic broadcast framework.

The proof of this Theorem relies on the following Lemma.

Lemma 1: $\{x_k^*(N)\}$ is the optimal solution of the following optimization problem:

$$\begin{aligned} \min_{\{x_k\}} \quad & \sum_k g(x_k) && (5)\\ \text{subject to} \quad & \sum_k x_k l_k = \sum_k x_k^*(N) l_k \\ & x_k \in \{s(i) | i = 0,1,2,\cdots\}, && k = 1,2,\cdots,K. \end{aligned}$$

Proof: We prove the lemma by contradiction. $\{x_k^*(N)\}$ is obviously feasible in (5). Suppose $\{x_k^*(N)\}$ is not the optimal solution of (5) and there exists the optimal solution $\{x_k^{'}\}$, where $\sum x_k^{'} l_k = \sum x_k^*(N) l_k$, and $\sum g(x_k^{'}) = M$, $M < N$. Construct $\{x_k^{''}\}$ as follows:

$$x_k^{''} = \begin{cases} s(g(x_1^{'}) + N - M) & \text{if } k = 1 \\ x_k^{'} & \text{otherwise} \end{cases} \quad (6)$$

Then $\sum g(x_k^{''}) = N$. Since $\{x_k^*(N)\}$ is the optimal solution of $B(N)$,

$$\sum x_k^*(N) l_k \le \sum x_k^{''} l_k. \quad (7)$$

However $s(i)$ is a non-increasing function of i, so $x_1^{''} l_k < x_1^{'} l_k$. Therefore $\sum x_k^{''} l_k < \sum x_k^{'} l_k = \sum x_k^*(N) l_k$. which contradicts the formula (7). ∎

Proof of Theorem 1: For all N s.t. $\sum x_k^*(N)l_k \leq B$, from Lemma 1, $\{x_k^*(N)\}$ is a feasible solution of problem (3). Furthermore, since N^* is minimum among all feasible solutions, $\{x_k^*(N^*\}$ is thus the optimal solution of proxy buffer allocation problem (3). In addition, because $\sum g(x_k^*(N^*)) = N^*$, N^* server channels are required to provide streaming video service. ∎

Now we turn to study how to find N^*. In the following Lemma we show that $\sum x_k^*(N)l_k$ monotonically decreases as N increases. Therefore if we can increase N until $\sum x_k^*(N)l_k \leq B < \sum x_k^*(N-1)l_k$, $\{x_k^*(N)\}$ is the optimal solution of problem (3).

Lemma 2: $\sum x_k^*(N)l_k$ monotonically decreases as N increases.

Proof: For arbitrary $M < N$, let $\{x_k^*(M)\}$ be the optimal solution of problem $B(M)$. Construct $\{x_k^{''}\}$ as follows:

$$x_k^{''} = \begin{cases} s(g(x_1^*(M)) + N - M) & \text{if } k = 1 \\ x_k^*(M) & \text{otherwise} \end{cases} \tag{8}$$

Then $\sum g(x_k^{''}) = N$.

Because $s(\cdot)$ is a non-increasing function, $x_1^{''} < x_1^*(M)$. So $\sum x_k^{''} l_k < \sum x_k^*(M)l_k$. Furthermore, $\{x_k^*(N)\}$ is the optimal solution of problem $B(N)$, so $\sum x_k^*(N)l_k \leq \sum x_k^{''} l_k$. Therefore $\sum x_k^*(N)l_k \leq \sum x_k^*(M)l_k$, which establish $\sum x_k^*(N)l_k$ as a decreasing function of N. ∎

The key to solve proxy buffer allocation problem (3) is to solve problem $B(N)$. The problem $B(N)$ can be rewritten as follows:

$$\begin{aligned} B(N): \quad & \min \sum_k s(c_k) \cdot l_k \\ \text{subject to} \quad & \sum_k c_k = N \\ & c_k \in \{0, 1, 2, \cdots, N\}, \qquad k = 1, 2, \cdots, K. \end{aligned} \tag{9}$$

This is a typical discrete resource allocation problem with a separable convex objective function. In [25], this type of problem is studied (Theorem 4.1.1) and "greedy" algorithm (or "marginal allocation") is given (Theorem 4.2.1). Starting with an initial solution $(0, 0, \cdots, 0)$ (which is not feasible), one unit of resource is assigned at each iteration to the most favorable activity (in the sense of minimizing the objective function) until $\sum_k c_k = N$ is attained. The greedy algorithm illustrated in Fig. 5 starts from a feasible point and leads to the optimal solution.