Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution

Bing Wang, Subhabrata Sen, *Member, IEEE*, Micah Adler, and Don Towsley, *Fellow, IEEE*

***Abstract*—In this paper, we address the problem of efficiently streaming a set of heterogeneous videos from a remote server through a proxy to multiple asynchronous clients so that they can experience playback with low startup delays. We determine the *optimal* proxy prefix cache allocation to the videos that minimizes the aggregate network bandwidth cost. We integrate proxy caching with traditional server-based reactive transmission schemes such as batching, patching and stream merging to develop a set of proxy-assisted delivery schemes. We quantitatively explore the impact of the choice of transmission scheme, cache allocation policy, proxy cache size, and availability of unicast versus multicast capability, on the resulting transmission cost. Our evaluations show that even a relatively small prefix cache (10%–20% of the video repository) is sufficient to realize substantial savings in transmission cost. We find that carefully designed proxy-assisted reactive transmission schemes can produce significant cost savings even in a predominantly unicast environment such as the Internet.**

***Index Terms*—Computer networks, multimedia communication, multimedia streaming, prefix caching, proxy caching, streaming media distribution.**

I. Introduction

THE EMERGENCE of the Internet as a pervasive communication medium, and a mature digital video technology have led to the rise of several networked streaming media applications such as live video broadcasts, distance education, corporate telecasts, etc. However, due to the high bandwidth requirements and the long-lived nature (tens of minutes to a couple of hours) of digital video, server and network bandwidths are proving to be major limiting factors in the widespread use of video streaming over the Internet. This is further complicated by the fact that the client population is likely to be large, with different clients asynchronously issuing requests to receive their chosen media streams. Also, different video clips can have very different sizes (playback bandwidths and durations) and popularities. In this paper, we address the problem of efficiently streaming a set of heterogeneous videos from a remote server through a proxy to multiple asynchronous clients so that they can experience playback with low startup delays. Before presenting the main contributions of the paper, we discuss some key challenges and limitations of existing techniques in achieving this goal.

Existing research has focused on developing *reactive* transmission schemes that use multicast or broadcast connections in innovative ways to reduce server and network loads, for serving a popular video to multiple asynchronous clients. The techniques are *reactive* in that the server only transmits video on-demand, in response to arriving client requests. *Batching, patching* and *stream merging* belong to this category. In batching, the server batches requests that arrive close together in time [1], and multicasts the stream to the set of clients. In patching or stream tapping [2]–[4], the server streams the entire video sequentially to the very first client. A later client receives (part of) its future playback data by listening to an existing ongoing multicast of the same video, with the server transmitting afresh only the missing prefix. Stream merging [5] is a related technique where all streams (complete and prefix) are transmitted using multicast, and clients can patch onto any earlier multicast stream.

An underlying requirement for the above schemes is the existence of multicast or broadcast connectivity between the server and the clients. However, IP multicast deployment in the Internet has been slow and, even today, remains severely limited in scope and reach. Therefore, transmission schemes that can support efficient delivery in predominantly unicast settings need to be developed. In addition, with the existing schemes, data still has to traverse the entire end-to-end path from the server to the clients, and network delays can cause substantial playback startup delays at the clients.

An orthogonal technique for reducing server loads, network traffic and access latencies is the use of proxy caches. This technique has proven to be quite effective for delivering Web objects. However, video files can be very large, and traditional techniques for caching entire objects are not appropriate for such media. Caching strategies that have been proposed in recent years [6]–[9] cache a portion of a video file at the proxy. In particular, caching an initial prefix of the video [7] has a number of advantages including shielding clients from delays and jitter on the server-proxy path, while reducing traffic along that path. However, existing research has, for the most part, been in the context of unicast delivery of a separate stream to each client. Recent work [10]–[13] combines caching with scalable video transmission. However, the focus has mostly been on transmitting a single video or using nonreactive schemes such as periodic broadcast [12], [14] and on networks with end-to-end multicast/broadcast capability.

Manuscript received December 30, 2002; revised August 1, 2003. This work was supported in part by the National Science Foundation under Grants EIA-0080119, ANI-9973092, ANI9977635, and CDA-9502639. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pascal Frossard.

B. Wang, M. Adler, and D. Towsley are with the Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA.

S. Sen is with AT&T Labs—Research, Florham Park, NJ 07932-0971 USA (e-mail: sen@research.att.com).

Digital Object Identifier 10.1109/TMM.2003.822788

In this paper, we explore the combination of proxy prefix caching and proxy-assisted reactive transmission schemes for reducing the transmission cost of multiple heterogeneous videos. Integrating the two techniques has the potential to realize the bandwidth efficiencies of both approaches, while also masking network delays from clients. In patching, for instance, the initial parts of the video are transmitted more frequently than the later parts, suggesting that prefix caching would be particularly effective for bandwidth reduction. Ideally, a proxy-assisted transmission scheme should be incrementally deployable and be able to work with existing unicast-based servers. We address the following questions in this paper.

1) What are suitable proxy-assisted reactive transmission schemes?
2) For a given transmission scheme, what is the optimal proxy prefix caching scheme that minimizes the transmission cost?
3) What are the resource (proxy cache space and transmission bandwidth) tradeoffs for the different transmission schemes?

A. Contributions

The following are the main contributions of this work.

- We develop a generalized allocation technique for minimizing the transmission cost. It is general in that it applies to any reactive transmission scheme. It is transmission-scheme aware in that the allocation is based on the transmission cost of a given scheme.
- Starting with traditional reactive transmission schemes, we develop corresponding schemes that use proxy prefix caching as an integral part of bandwidth-efficient delivery in Internet-like environments, where the end-to-end network connections provide unicast-only service, or at best offer multicast capability only on the last mile proxy-client path.
- We quantitatively explore the impact of the developed transmission schemes coupled with the optimal cache allocation, the proxy cache size and availability of unicast versus multicast capability, on the resultant transmission cost. We develop guidelines for aggregate proxy cache sizing, and identify the combination of transmission and caching schemes that provides the best performance under different scenarios.

To the best of our knowledge, this is the first systematic evaluation of resource (proxy cache space and transmission bandwidth) issues that arise when combining proxy prefix caching with reactive transmission for delivering multiple heterogeneous videos across networks. Complementary work [15] focuses on a particular reactive transmission scheme called bandwidth skimming, and numerically explores the proxy cache allocation problem in that context. Subsequent work [16] considers the scenario in which different transmission schemes (batching, patching, etc.) are used simultaneously for different videos and proposes a heuristic proxy caching algorithm.

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 our optimal proxy prefix caching technique. Section IV presents a set of proxy-assisted reactive transmission schemes. Our evaluations are presented in Section V. Section VI discusses implementation issues associated with deploying proxy-assisted transmission schemes. Finally, Section VII concludes the paper and presents future work.

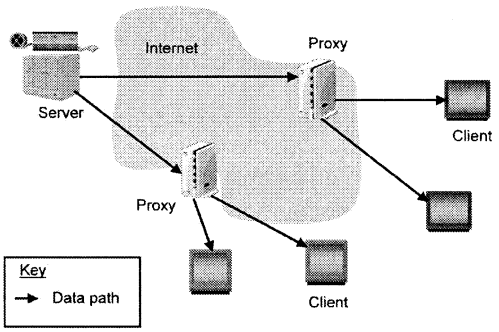

Fig. 1. Streaming video in the Internet: The video stream originates from a remote server and travels through the network to the end client. The proxies performing prefix caching are located close to the clients, e.g., at the head-end of the local access network.

II. Problem Setting and Model

Consider a group of clients receiving videos streamed across the Internet from a server via a single proxy (Fig. 1). We assume that clients always request playback from the beginning of a video. The proxy intercepts the client request and, if a prefix of the video is present locally, streams the prefix directly to the client. If the video is not stored in its entirety at the proxy, the latter contacts the server for the suffix of the stream, and relays the incoming data to the client.

In today's Internet, the network route from the server to the client often traverses multiple ISP domains, and predominantly uses unicast delivery, since IP Multicast is not widely deployed. We note that, while many-to-many inter-domain multicast has been slow to be deployed, one-to-many intra-domain multicast (as would be used in an enterprise or cable/DSL-based last-hop network environment) is much simpler to deploy and manage [17]. We therefore assume that the server-proxy network path is unicast-enabled, while the network paths from the proxy to the clients are either unicast or multicast/broadcast enabled. Since the proxy is located close to the clients, we assume the bandwidth required to send one bit from the proxy to multiple clients using multicast/broadcast is independent of the number of clients. Finally, for simplicity of exposition, we focus on a single server and a single proxy. Our results apply directly to multiple-proxy Content Distribution Networks where the server has unicast connections to the proxies, each proxy serves a different set of clients (no overlapping), and the proxies do not interact.

A. Model

We next provide a formal model of the system, and introduce notation and key concepts that will be used in the rest of the paper. Table I presents the key parameters in the model.

We consider a server with a repository of N constant-bit-rate (CBR) videos. We assume the access probabilities of all the

TABLE I
PARAMETERS IN THE MODEL

Para.	Definition
N	Number of videos
L_i	Length of video i (sec.)
b_i	Mean bandwidth of video i (bits per sec.)
u	Caching grain
n_i	Size of video i (units)
f_i	Access probability of video i
λ_i	Request rate for video i
λ	Aggregate request arrival rate
S	Proxy cache size (units)
v_i	Length (sec) of cached prefix for video i
v	Storage vector, $v = (v_1, v_2, \ldots, v_N)$
c_s	Transmission cost on server-proxy path (per bit)
c_p	Transmission cost on proxy-client path (per bit)
$C_i(v_i)$	Transmission cost per unit time for video i when a prefix of length v_i for video i is cached

videos and the aggregate access rate to the video repository are known a priori. In a real system, these parameters can be obtained by monitoring the system. Without loss of generality, we index the videos in nonincreasing order of their access probabilities. Let f_i be the access probability of video i, $\sum_{i=1}^{N} f_i = 1$. f_i measures the relative popularity of a video: every access to the video repository has a probability f_i of requesting video i. Let λ_i be the access rate of video i and λ be the aggregate access rate to the video repository. Then $\lambda_i = \lambda f_i, 1 \leq i \leq N$.

We introduce a *caching grain* of size u to be the smallest unit of cache allocation and all allocations are in multiples of this unit. It can be one bit or one minute's worth of data, etc. We express the size of video i and the proxy cache size as a multiple of a caching grain. Video i has playback bandwidth b_i bps, length L_i seconds, and size n_i units, $n_i u = b_i L_i$. We assume that the proxy can store S units where $S \leq \sum_{i=1}^{N} n_i$. The *storage vector* $v = (v_1, v_2, \ldots, v_N)$ specifies that a prefix of length v_i s for each video i is cached at the proxy, $i = 1, 2, \ldots, N$. Note that the videos cached at the proxy cannot exceed the storage constraint of the proxy, that is, $\sum_{i=1}^{N} b_i v_i \leq uS$. Let c_s and c_p respectively represent the costs associated with transmitting one bit of video data on the server-proxy path and on the proxy-client path. Our goal is to develop appropriate transmission and caching schemes that minimize the mean transmission cost per unit time aggregated over all the videos in the repository, i.e., $\sum_{i=1}^{N} C_i(v_i)$, where $C_i(v_i)$ is the transmission cost per unit time for video i when a prefix of length v_i of the video is cached at the proxy. In the rest of the paper, unless otherwise stated, we shall use the term *transmission cost* to refer to this metric.

For simplicity of exposition, we ignore network propagation latencies. All the results can be extended in a straightforward manner when network propagation latencies are considered [18]. On receiving a client request for a video, the proxy calculates a *transmission schedule* based on the predetermined transmission scheme. This transmission schedule specifies, for each frame in the video, when and on what *transmission channel* (unicast or multicast connection) it will be transmitted by the proxy. The proxy also determines and requests the suffix from the server. A *reception schedule* is transmitted from the proxy to the client specifying, for each frame in the video, when and from which transmission channel the client should receive that frame. Note that a client may need to receive data from multiple transmission channels simultaneously. Frames received ahead of their playback times are stored in a client-side workahead buffer. For simplicity, we shall assume the client has sufficient buffer space to accommodate an entire video clip. Finally, note that in our approach, the server only needs to transmit via unicast a suffix of the video requested by the proxy. Our delivery techniques are therefore incrementally deployable as these can work with existing predominantly unicast-based media servers and require no additional server-side functionality.

III. Optimal Proxy Cache Allocation

We next propose a general technique to determine the optimal proxy prefix cache allocation for any given proxy-assisted transmission scheme. For a given transmission scheme, the average transmission cost per unit of time for video i, $C_i(v_i)$, is a function of the prefix v_i cached at the proxy, $0 \leq v_i \leq L_i$. We make no assumption regarding $C_i(v_i)$; it may not exhibit properties such as monotonicity or convexity. For some transmission schemes, there may not even exist a closed-form expression for $C_i(v_i)$. In this case we assume that this value can be obtained by monitoring a running system.

Recall that the caching grain is the smallest unit of cache allocation (see Section II). The size of video i is n_i units and the size of the proxy is S units. Let $A_i = \{m_i | 0 \leq m_i \leq n_i\}$ denote the set of possible prefixes for video i, where m_i units is the size and $m_i u / b_i$ seconds is the length of a possible prefix of video i. Let $saving(m_i)$ denote the saving in transmission cost when caching an m_i-unit prefix of video i over caching no prefix of the video at the proxy, i.e., $saving(m_i) = C_i(0) - C_i(m_i u / b_i)$. Our goal is to maximize the aggregate savings and, hence, minimize the aggregate transmission cost over all the videos. The optimization problem can therefore be formulated as

$$\text{maximize} : \sum_{i=1}^{N} \text{saving}(m_i)$$

$$\text{s.t.} \sum_{i=1}^{N} m_i \leq S, m_i \in A_i.$$

Note that this formulation is a variant of the 0-1 knapsack problem, where the items to be placed into the knapsack are partitioned into sets and at most one item from each set can be chosen. We next use the following dynamic programming algorithm to determine the optimal allocation.

Let B be a two-dimensional matrix, where entry $B(i, j)$ represents the maximum saving in the transmission cost for the first i videos in a proxy cache of size j. When $i = 0$, $B(i, j) = 0$. When $i > 0$

$$B(i, j) = \max_{\forall m_i \in A_i} \{B(i-1, j - m_i) + \text{saving}(m_i)\}.$$

This matrix is filled in row order starting from $B(0, j), j = 0, \ldots, S$. The value $B(N, S)$ is the maximum saving in trans-

mission cost when all N videos have been used. The minimum transmission cost is $\sum_{i=1}^{N} C_i(0) - B(N,S)$, since the saving is relative to storing nothing at the proxy. The optimal cache allocation can now be computed as follows. For each entry, we store a pointer to an entry from which this current entry is computed. By tracing back the pointers from the entry $B(N,S)$, the optimal allocation is obtained. The execution time of the algorithm is $O(NSK)$, where $K = \max_{1\le i\le N} |A_i|$.

IV. Proxy-Assisted Transmission Schemes

In this section, we develop a set of reactive transmission schemes that use proxy prefix caching as an integral part for bandwidth-efficient delivery in Internet-like environments, where the end-to-end network connections only provide unicast service, or at best offer multicast capability on the proxy-client path. For each scheme, we develop a closed-form expression for the transmission cost $C_i(v_i)$ associated with video i, $1 \le i \le N$. In the interest of space, detailed derivations are omitted and can be found in [18]. The transmission cost $C_i(v_i)$ is used in Section III to determine the proxy cache allocation for each video that minimizes the aggregate transmission cost. The transmission schemes we propose are completely general and apply to any sequence of client arrivals. However, we shall assume a Poisson arrival process for analyzing the transmission costs. Our ongoing work shows that Poisson arrival is a conservative assumption for reactive schemes. A similar conjecture is presented in [19].

A. Unicast Suffix Batching (SBatch)

SBatch is a simple batching scheme that takes advantage of the video prefix cached at the proxy to provide instantaneous playback to clients. This scheme is designed for environments where the proxy-client path is only unicast-capable.

Suppose the first request for video i arrives at time 0. The proxy immediately begins transmitting the video prefix to the client. SBatch schedules the transmission of the suffix from the server to the proxy *as late as possible*, just in time to guarantee discontinuity-free playback at the client. That is, the first frame of the suffix is scheduled to reach the proxy at time v_i, the length of the prefix. For any request arriving in time $(0, v_i]$, the proxy just forward the single incoming suffix (of length $L_i - v_i$) to the new client, and no new suffix transmission is needed from the server. In effect, multiple demands for the suffix of the video are batched together. Note that in contrast to traditional batching, SBatch does not incur any playback startup delay. Assuming a Poisson arrival process, the average transmission cost for delivering video i is

$$C_i(v_i) = \left(c_s \frac{L_i - v_i}{1 + v_i\lambda_i} + c_p L_i \right) \lambda_i b_i.$$

When $v_i = 0$ ($v_i = L_i$), video i is transmitted from the server (proxy) using unicast, since it is impossible to batch multiple requests.

B. Unicast Patching With Prefix Caching (UPatch)

SBatch can be further improved by using patching for the suffix. Note that here we use patching in the context of unicast.

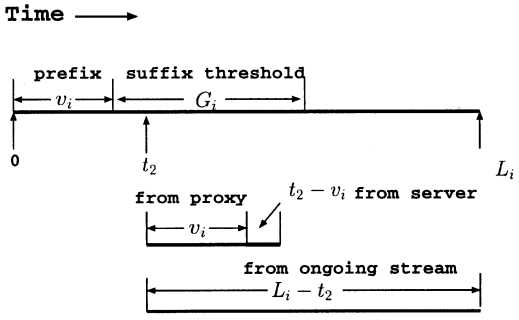

Fig. 2. Unicast patching with prefix caching (UPatch).

This is possible because the proxy can forward one copy of the data from the server to multiple clients.

Suppose that the first request for video i arrives at time 0 and the suffix reaches the proxy from the server at time v_i, as shown in Fig. 2. Suppose another client's request for video i comes at time $t_2, v_i < t_2 < L_i$. The proxy can schedule a transmission of the complete suffix at time $t_2 + v_i$ from the server. Another option is to schedule a patch of $[v_i, t_2)$ of the suffix from the server since segment $[t_2, L_i]$ has already been scheduled to be transmitted. Note that this patch can be scheduled at time $t_2 + v_i$ so that the client is still required to receive from at most two channels at the same time. The decision to transmit a complete suffix or a patch depends on a *suffix threshold* G_i, measured from the beginning of the suffix. If one request arrives within G_i units from when the nearest complete transmission of the suffix was started, the proxy schedules a patch from the server for it. Otherwise, it starts a new complete transmission of the suffix. Assuming a Poisson arrival process, the average transmission cost for video i is

$$C_i(v_i) = c_s\lambda_i b_i \frac{\frac{\lambda_i G_i^2}{2} + L_i - v_i}{1 + \lambda_i(v_i + G_i)} + c_p\lambda_i b_i L_i.$$

The suffix threshold G_i is chosen to minimize the transmission cost for video i for a given prefix v_i. Finally, when $v_i = L_i$, video i is transmitted from the proxy to clients using unicast.

C. Multicast Patching With Prefix Caching (MPatch)

If the proxy-client path is multicast capable, the proxy can use a multicast transmission scheme. We describe MPatch, a patching scheme that exploits prefix caching at the proxy.

Suppose the first request for video i arrives at time 0 (Fig. 3). The proxy then starts to transmit the prefix of the video via multicast at time 0. The server starts to transmit the suffix of the video to the proxy at time v_i and the proxy transmits the received data via multicast to the clients. Later requests can start a new complete multicast stream or join the ongoing multicast of the stream and use a separate unicast channels to obtain the missing data. Let T_i be a threshold to regulate the frequency at which the complete stream is transmitted. Suppose a request arrives $t_2 (0 < t_2 \le T_i)$ time units after the beginning of the nearest ongoing complete stream. Video delivery for this client can be classified into the following two cases according to the relationship of v_i and T_i.

- *Case 1:* $T_i \le v_i \le L_i$. This is shown in Fig. 3(a). The client receives segment [0, t_2] from a separate channel

via unicast from the proxy and segment $(t_2, L_i]$ via the ongoing multicast stream. Assuming a Poisson arrival process, the transmission cost function in this case $g_1(v_i, T_i)$ is

$$g_1(v_i, T_i) = \frac{\lambda_i b_i}{1+\lambda_i T_i}\left[(L_i - v_i)c_s + L_i c_p + \frac{\lambda_i {T_i}^2}{2} c_p\right].$$

- *Case 2:* $0 \le v_i < T_i$. This is shown in Fig. 3(b). If $0 < t_2 \le v_i$, then the transmission mechanism is the same as in Case 1. If $v_i < t_2 \le T_i$, the client receives segment [0,v_i] from a separate channel via unicast from the proxy and receives segment $(t_2, L_i]$ via the ongoing multicast stream. Segment $(v_i, t_2]$ is transmitted from the server to the client via the proxy using unicast. Assuming a Poisson arrival process, the transmission cost function in this case $g_2(v_i, T_i)$ is

$$g_2(v_i, T_i) = \frac{\lambda_i b_i}{1+\lambda_i T_i}\left[(L_i - v_i)c_s + L_i c_p + \frac{\lambda_i {v_i}^2}{2} c_p + \frac{\lambda_i (T_i - v_i)^2}{2}(c_s + c_p)\right].$$

Let $h_k(v_i)$ be the minimum transmission cost in Case k, $k = 1, 2$. That is

$$h_k(v_i) = \min_{T_i} \{g_k(v_i, T_i), 0 \le T_i \le L_i\},\ k = 1, 2.$$

For a given prefix v_i, the average transmission cost is

$$C_i(v_i) = \min\{h_1(v_i), h_2(v_i)\}.$$

Finally, note that if video i is streamed entirely from a single location (either the server or the proxy), the MPatch transmission scheme reduces to Controlled Multicast (CM) patching [4].

D. Multicast Merging With Prefix Caching (MMerge)

The key issue in stream merging is deciding how to merge a later stream into an earlier stream. Closest Target [5] is one online heuristic merging policy whose performance is close to that of optimal offline stream merging. This policy chooses the closest earlier stream still in the system as the next merge target.

Our MMerge scheme integrates proxy caching and stream merging. It uses the Closest Target policy to decide how to merge a later stream into an earlier stream. For a video segment required by the client, if a prefix of the segment is at the proxy, it is transmitted directly from the proxy to the client; the suffix not cached at the proxy is transmitted from the server *as late as possible* while still ensuring continuous playback at the client. Let p_j be the probability of requiring a j-second prefix per unit of time for video i, $0 \le j \le L_i$. Then the average transmission cost for video i is

$$C_i(v_i) = \sum_{j=1}^{v_i} j p_j b_i c_p + \sum_{j=v_i+1}^{L_i} (j(c_p + c_s) - v_i c_s) p_j b_i.$$

Finally, note that if video i is streamed entirely from a single location (either the server or the proxy), MMerge reduces to Closest Target stream merging.

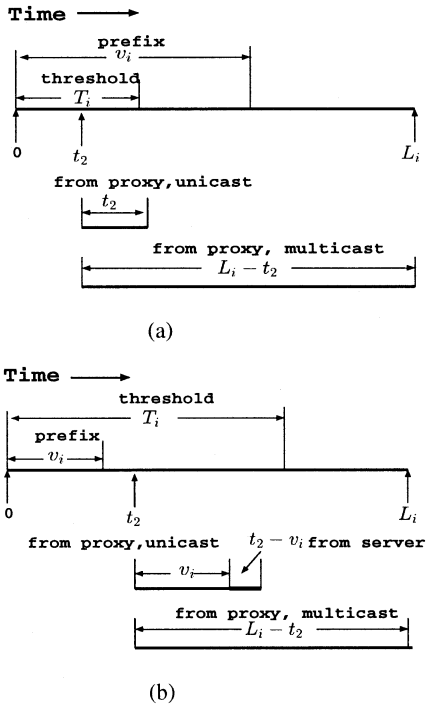

Fig. 3. Multicast patching with prefix caching (MPatch). (a) Case 1: $T_i \le v_i \le L_i$. (b) Case 2: $0 \le v_i < T_i$.

V. Performance Evaluation

In this section, we examine the resource tradeoffs under the previously described caching and transmission schemes. We consider a repository of 100 CBR video clips with access probabilities drawn from a Zipf distribution with parameter $\theta = 0.271$ [1]. In this paper, we focus mainly on the setting in which all the videos are two hours long, and of the same bandwidth. At the end of this section, we present some initial results for videos that differ in their bandwidth requirements. The transmission cost is normalized by both the value of c_s and the maximum video bandwidth. That is, the normalized transmission cost is $\sum_{i=1}^{N} C_i(v_i)/(c_s b)$, where $b = \max_{1 \le i \le N} b_i$. Let $\hat{c}_p = c_p/c_s$. In this section, we assume $\hat{c}_p \in [0, 1]$. Observe that $\hat{c}_p = 0$ corresponds to $c_p = 0$ and $\hat{c}_p = 1$ corresponds $c_p = c_s$. We represent the proxy cache size as a percentage, r, of the size of the video repository. We use one minute's worth of data as the caching grain. For MMerge, the probability of requiring a j-second prefix per unit of time for video i is obtained from a 150-h simulation run (the confidence intervals from 30 runs are very narrow).

We first compare the transmission costs using optimal prefix caching and optimal 0-1 caching. In optimal 0-1 caching, a video is allowed to be cached in its entirety or not at all. We then investigate differences in transmission cost under optimal prefix caching and a heuristic, proportional priority (PP) caching. In PP caching, the size of the proxy cache allocated to a video is proportional to the product of the size of the video and its access probability, under the constraint that the allocated space is no larger than the size of the video. PP caching takes account of both the popularity and the size of the video. A similar heuristic is suggested in [13].

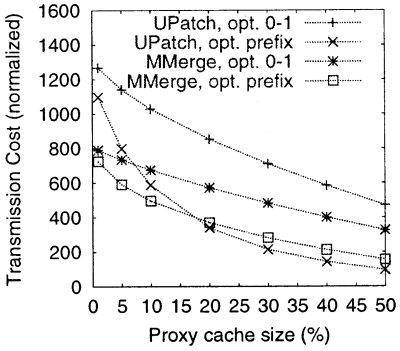

Fig. 4. Normalized transmission cost versus proxy cache size, $\lambda = 100/\text{min}$, $\hat{c}_p = 0$.

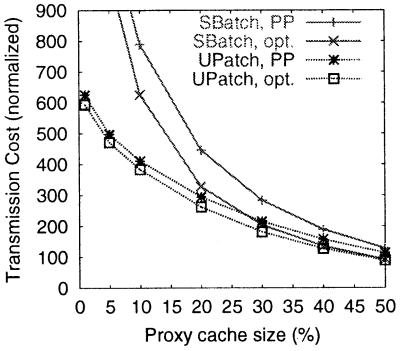

Fig. 5. Normalized transmission cost versus proxy cache size, $\lambda = 30/\text{min}$, $\hat{c}_p = 0$.

A. Optimal Prefix Caching Versus Optimal 0-1 Caching

The allocation under optimal 0-1 caching can be modeled as a 0-1 knapsack problem [18]. When the length and bandwidth of the videos are the same, the optimal 0-1 scheme caches videos in the order of their popularities. We find that optimal prefix caching significantly outperforms optimal 0-1 caching for all the schemes we examine. Fig. 4 plots the transmission costs under the two caching schemes for UPatch and MMerge when $\hat{c}_p$ is 0 and the arrival rate λ is 100 requests/min. UPatch and MMerge under optimal prefix caching result in substantially lower costs than under optimal 0-1 caching across the range of proxy cache sizes. For instance, when the proxy cache is 20% of the size of the video repository, optimal prefix caching reduces the costs over optimal 0-1 caching by 60% and 35% for UPatch and MMerge, respectively. We therefore focus on prefix caching for the rest of the paper.

B. Transmission and Caching Schemes Under Unicast

We first investigate the transmission cost when the proxy-client path is only unicast capable. Fig. 5 depicts the transmission cost as a function of r, when $\hat{c}_p$ is 0 and the aggregate arrival rate λ is 30 requests/min. The performance of SBatch and UPatch under both PP and optimal prefix caching are plotted on the graph. The reductions in the transmission costs by using optimal prefix caching over PP caching are similar for various proxy cache sizes. As the aggregate arrival rate increases, the cost reduction using optimal prefix caching over PP caching also increases (figures not shown). The reason will become apparent at the end of Section V-B.

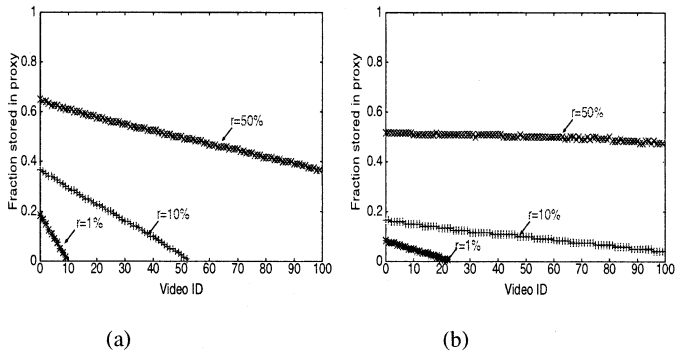

Fig. 6. Proxy cache allocation for UPatch under optimal prefix caching, $\hat{c}_p = 0$. (a) $\lambda = 10$/min. (b) $\lambda = 100$/min.

We observe from Fig. 5 that a small amount of cache at the proxy results in substantial cost savings for both transmission schemes under optimal prefix caching. For instance, with a proxy cache that is 10% of the size of the video repository, the transmission costs reduce to 17% and 88% of the corresponding costs without a proxy cache for SBatch and UPatch respectively.

We find that UPatch substantially reduces cost over SBatch under optimal prefix caching, particularly for small and moderate proxy sizes (see Fig. 5). For instance, when $r = 1\%$, the reduction under UPatch over SBatch is 69%. However, this is under the assumption that the optimal threshold for UPatch can be obtained. The choice of the threshold critically impacts the cost savings for UPatch—an arbitrary threshold value can result in performance degradation. Hence for situations where the appropriate threshold cannot be properly determined, SBatch may be preferred. SBatch, being simpler to implement, is also preferred for larger proxy cache sizes, where its performance is very close to that of UPatch.

The above discussion focused on the case of $\hat{c}_p = 0$. When $\hat{c}_p > 0$, we observe similar performance trends for the different transmission and caching schemes. This is because when the proxy-client path is only unicast-capable, the proxy has to transmit a copy of each data unit separately to each client. Hence, for a fixed $\hat{c}_p$, the transmission costs on the proxy-client path are identical for all transmission (unicast-based) and caching schemes.

Proxy cache allocation across the videos: We next examine the proxy cache allocation for SBatch and UPatch under optimal prefix caching. When the proxy-client path is only unicast-capable, the optimal prefix cache allocation is identical for all values of $\hat{c}_p$ for a given transmission scheme. This is because, as mentioned earlier, the transmission cost on the proxy-client path for a fixed $\hat{c}_p$ does not depend on cache allocation. Therefore allocating the proxy cache to minimize the total transmission cost is the same as that required to minimize the transmission cost on the server-proxy path, which is independent of the value of $\hat{c}_p$. In the following, $\hat{c}_p$ is chosen to be 0.

Fig. 6 depicts the proxy cache allocations under UPatch, for arrival rates of ten and 100 requests/min. The proxy cache allocation under SBatch is similar. We see that, when the proxy

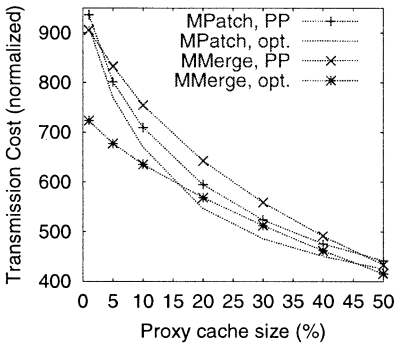

Fig. 7. Normalized transmission cost versus proxy cache size when $\lambda = 30/\text{min}$ and $\hat{c}_p = 0.5$.

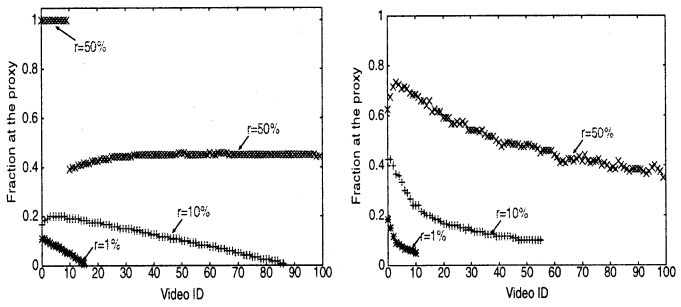

Fig. 8. Proxy cache allocation for (a) MPatch and (b) MMerge under optimal prefix caching when $\hat{c}_p = 0.1$ and $\lambda = 30/\text{min}$.

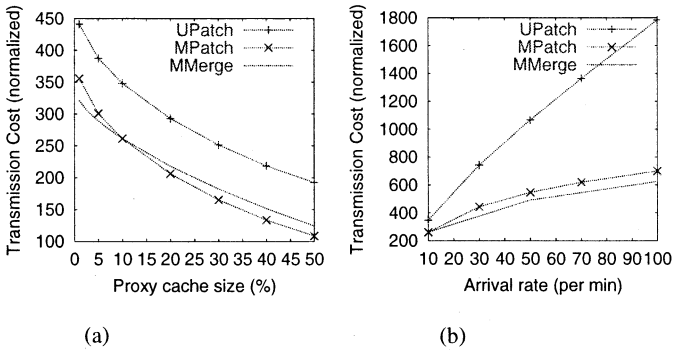

Fig. 9. Comparison between unicast and multicast schemes when $\hat{c}_p = 0.1$. (a) Normalized transmission cost versus proxy cache size $\lambda = 10/\text{min}$. (b) Normalized transmission cost versus arrival rate $r = 10\%$.

cache size is small, only the most popular videos are cached. As the proxy cache size increases, more videos are cached. For low aggregate arrival rates, the size of the proxy storage allocated to a video increases as a function of its access probability. At high arrival rates, the proxy storage tends to be more evenly distributed among all the videos; this differs substantially from the proportional allocation under PP caching and helps to explain the difference in transmission cost under the two caching schemes.

C. Transmission and Caching Under Multicast

We next investigate the transmission cost when the proxy-client path is multicast capable. Fig. 7 shows the normalized transmission cost as a function of r, when $\hat{c}_p$ is 0.5 and the aggregate arrival rate λ is 30 requests/min. The transmission costs for MPatch and MMerge under optimal prefix caching and PP caching are plotted on the graph. In the case of MPatch, the transmission costs under optimal prefix caching and PP caching are close for very small and large proxy sizes. In the case of MMerge, the difference in transmission costs under optimal prefix caching and PP caching is large for small proxy cache sizes. For instance, when $r = 1\%$, the transmission cost under optimal prefix caching is 20% lower than that under PP caching. The cost reduction achieved by optimal prefix caching over PP caching increases with the aggregate arrival rate.

Fig. 7 also demonstrates that a small amount of proxy buffer results in substantial transmission cost savings under optimal prefix caching. With a proxy cache that can hold 10% of the video repository, the transmission costs reduce to 65% and 85% of the corresponding costs in the absence of proxy cache for MPatch and MMerge respectively. It is interesting to notice from Fig. 7 that proxy-assisted MMerge does not always outperform MPatch. This is different from traditional server-based patching and stream merging, where stream merging always outperforms patching.

Proxy cache allocation across the videos: We next examine the proxy cache allocation for MPatch and MMerge under optimal prefix caching. When $\hat{c}_p = 0$, since the transmission from the proxy to clients does not incur any cost, using multicast or unicast along the proxy-client path does not make any difference to the allocation. Therefore, the allocation for MPatch is identical to UPatch as shown in Fig. 6.

Fig. 8(a) displays proxy cache allocations for MPatch when $\hat{c}_p = 0.1$ and $\lambda = 30/\text{min}$. We find that the size of the proxy cache allocated to a video is not a monotonically increasing function of the access probability. This is because the threshold tends to increase as the access probability decreases. Therefore some less popular videos may require larger prefixes than more popular videos to realize the optimal threshold.

Fig. 8(b) depicts the proxy cache allocations for MMerge when $\hat{c}_p = 0.1$ and $\lambda = 30/\text{min}$. In general, the proxy cache space allocated to a video decreases as its popularity decreases. However, when the proxy caches are large and the arrival rates are high, the size of the proxy cache allocated to a video can increase as the popularity decreases. This is because the average length of a prefix stream increases as the popularity (hence the arrival rate) decreases. We also observe that when $\hat{c}_p = 0.1$, only several of the most popular videos are cached for small and moderate proxy caches. When $\hat{c}_p = 0$ (not shown in the figure), proxy cache is more evenly distributed among the videos for small and moderate proxy caches.

D. Comparison Between Unicast and Multicast

When $\hat{c}_p > 0$, using multicast instead of unicast along the proxy-client path results in substantial savings. We set $\hat{c}_p$ to 0.1 in the following. Fig. 9(a) depicts the normalized transmission costs of UPatch, MPatch and MMerge under optimal prefix caching when $\lambda = 10/\text{min}$. We observe, in this case, that the transmission costs of MPatch and MMerge are significantly lower than those of UPatch across the range of proxy cache sizes. Fig. 9(b) shows the transmission costs as the arrival rate increases from ten to 100 requests/min when $r = 10\%$. The

savings under MPatch and MMerge over UPatch increase as the arrival rate increases. When the arrival rate is 10 requests/min, transmission costs under MPatch is 25% lower than under UPatch. When the arrival rate is 100 requests/min, the reduction becomes 61%. This clearly illustrates the benefits of using multicast locally, over the proxy-client path.

E. Videos With Heterogeneous Bandwidths

The above considers 100 videos of the same length and bandwidth. We next investigate two settings in which videos have different bandwidth requirements. In the first setting, the bandwidth of a video is randomly chosen from {50, 100, 200, 1000} kbps. In the second setting, a video with a higher access rate is assigned a higher bandwidth. That is, the top 25 popular videos have the highest bandwidth of 1000 kbps; the second 25 popular videos have the bandwidth of 200 kbps, etc. In both settings, we observe overall similar performance trends as when the videos are of the same bandwidth. For a specific transmission scheme and proxy cache size, the cost reduction percentage achieved by prefix caching over PP caching is slightly higher than the homogeneous bandwidth case in the first setting and more significant in the second setting (figures not shown), especially for moderate and large proxy cache sizes. Figures are omitted in the interest of space. More detailed exploration on heterogeneous videos is ongoing work.

F. Summary of Results

We summarize the key results from our evaluation.

- For the same proxy size, using prefix caching for a set of videos results in significantly lower transmission costs compared to entire-object caching policies. Under optimal prefix caching, even a relatively small proxy cache (10%–20% of the video repository) is sufficient to realize substantial savings in transmission cost.
- The allocation under optimal prefix caching is sensitive to the transmission scheme, the aggregate arrival rate and the value of $\hat{c}_p$. Optimal prefix caching can substantially outperform transmission cost agnostic PP caching, particularly for high arrival rates.
- Carefully designed reactive transmission schemes coupled with optimal proxy prefix caching can produce significant cost savings over using unicast delivery, even when the underlying network offers only unicast service.

VI. Experimental Evaluation

Previous sections presented an algorithmic and analytical treatment of the transmission cost reduction achieved by using proxy-assisted transmission schemes in combination with optimal proxy cache allocation. We are currently investigating implementation and performance issues associated with actually deploying proxy-assisted transmission schemes over the Internet. We expect such schemes to provide higher reception quality at the client side than one-to-one unicast from the server to the clients, as a result of lower end-to-end bandwidth requirements imposed by such schemes. On the other hand, quality of service (loss, delay and delay jitters) on the server-proxy path can have significant impact on viewing performance. For instance, losses in a stream transmitted over the server-proxy path will translate to losses at multiple clients when the stream is shared by these clients.

To explore these issues, we are running end-to-end streaming experiments between multiple Internet locations in the U.S. and abroad using an experimental client-server-proxy testbed that we have developed. We implemented two proxy-assisted transmission schemes: SBatch and UPatch. In our experiments, one site hosts the server and a remote site hosts both the proxy and client population. Initial experiments show that, under the same client arrival process, clients experience less bursty losses when using SBatch than when using one-to-one unicast from the server to the clients. This is because some client requests are batched together in SBatch and, hence, generate less bursty traffic along the long-haul server-proxy path. However, the loss rate along server-proxy paths still ranges from 4% to 12% even when using SBatch on some paths. We are in the process of developing efficient loss recovery schemes for use on the server-proxy path.

VII. Conclusions and Future Work

In this paper, we presented a technique to determine, for a given proxy-assisted transmission scheme, the *optimal* proxy prefix caching for a set of videos that minimizes the aggregate transmission cost. We presented and explored a set of proxy-assisted reactive transmission schemes that exploit proxy prefix caching to provide bandwidth efficient delivery. Our evaluations demonstrate that, even with a relatively small proxy cache, carefully designed transmission schemes under optimal prefix caching can lead to significant cost reductions. Among ongoing work, we explore how interactivity (e.g., fast forward, rewind, etc.) affects the allocation of videos in the proxy cache. We also intend to develop more complex models that considers not only the aggregate bandwidth but also the peak bandwidth usage for variable bit rate videos.

Acknowledgment

The first version of this paper appeared in Infocom 2002. The authors would like to thank Y. Guo, J. Rexford, P. Shenoy, and the anonymous reviewers for their insightful comments.

References

[1] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, June 1996, pp. 253–258.

[2] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. Int. Conf. Computer Communications and Networks*, Las Vegas, NV, 1997.

[3] K. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. ACM Multimedia*, Sept. 1998, pp. 191–200.

[4] L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," *IEEE Trans. Multimedia*, vol. 3, pp. 405–414, Dec. 2001.

[5] D. Eager, M. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM Multimedia*, Nov. 1999, pp. 199–202.

[6] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based caching for web servers," in *Proc. SPIE/ACM Conf. Multimedia Computing and Networking*, San Jose, CA, Jan. 1998.

[7] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM*, vol. 3, Apr. 1999, pp. 1310–1319.
[8] J. Almeida, D. Eager, and M. Vernon, "A hybrid caching strategy for streaming media files," in *Proc. SPIE/ACM Conf. Multimedia Computing and Networking*, San Jose, CA, Jan. 2001.
[9] 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*, San Francisco, CA, Apr. 1998.
[10] L. Gao, Z. Zhang, and D. Towsley, "Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams," *IEEE/ACM Trans. Networking*, vol. 11, pp. 884–894, Dec. 2003.
[11] S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (mcache): An adaptive zero-delay video-on-demand service," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.
[12] D. Eager, M. Ferris, and M. Vernon, "Optimized regional caching for on-demand data delivery," in *Proc. Multimedia Computing and Networking (MMCN '99)*, San Jose, CA, Jan. 1999.
[13] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini, "Joint server scheduling and proxy caching for video delivery," in *Proc. 6th Int. Workshop on Web Caching and Content Distribution*, Boston, MA, June 2001.
[14] S. Sen, L. Gao, and D. Towsley, "Frame-based periodic broadcast and fundamental resource tradeoffs," in *Proc. IEEE Int. Performance Computing and Communications Conf.*, Phoenix, AZ, Apr. 2001.
[15] J. Almeida, D. L. Eager, M. C. Ferris, and M. K. Vernon, "Provisioning content distribution networks for streaming media," in *Proc. IEEE INFOCOM*, New York, 2002.
[16] C. Venkatramani, O. Verscheure, P. Frossard, and K.-W. Lee *et al.*, "Optimal proxy management for multimedia streaming in content distribution networks," in *Proc. Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Miami Beach, FL, May 2002, pp. 147–154.
[17] C. Diot, B. Levine, B. Lyles, H. Kassan, and D. Balsiefien *et al.*, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 31, pp. 78–88, Jan. 2000.
[18] B. Wang, S. Sen, M. Adler, and D. Towsley, "Proxy-Based Distribution of Streaming Video Over Unicast/Multicast Connections," Dept. Computer Science, Univ. Massachusetts, Amherst, Tech. Rep. 01-05, 2001.
[19] D. L. Eager, M.K. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," *IEEE Trans. Knowl. Data Eng.*, vol. 13, pp. 742–757, Sept./Oct. 2001.

Bing Wang received the B.S. and M.S. degrees in computer science from Nanjing University of Science and Technology, Nanjing, China, in 1994 and the Chinese Academy of Sciences, Beijing, in 1997. She is currently working toward the Ph.D. degree in the Department of Computer Science, University of Massachusetts, Amherst.

Her research interests include Internet technologies and applications, network measurements, and modeling.

Subhabrata Sen (M'01) received the B.Eng. degree in computer science in 1992 from Jadavpur University, Calcutta, India, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1997 and 2001, respectively.

He is currently a Member of the Internet and Networking Systems Research Center at AT&T Labs—Research, Florham Park, NJ. His research interests include network measurement, peer-to-peer systems and overlay networks, multimedia proxy services, network security and network anomaly detection.

Dr. Sen is a member of the ACM.

Micah Adler received the B.S. degree in mathematics with computer science from the Massachusetts Institute of Technology, Cambridge, in 1990 and the Ph.D. degree from the Division of Computer Science, University of California, Berkeley, in 1996.

He joined the faulty of the Department of Computer Science, University of Massachusetts, Amherst, as an Assistant Professor in 1999, and is currently the Co-Director of the Theoretical Aspects of Parallel and Distributed Systems (TAPADS) Group. His primary research focus lies in theoretical issues concerning the design and use of communication networks and multiprocessor computational systems. His current interests include algorithmic aspects of network security, algorithmic aspects of the World Wide Web, asymmetric communication channels, mobile and wireless communication, and bandwidth efficient multiprocessor computation.

Dr. Adler is currently the Program Committee Chair for SPAA 2004, and a Managing Editor of *Internet Mathematics*. He was the recipient of post-doctoral research fellowships from both the Heinz Nixdorf Institute, Paderborn, Germany, and the University of Toronto, ON, Canada, and was recipient of a National Science Foundation CAREER Award in 2002.

Don Towsley (M'78–SM'93–F'95) received the B.A. degree in physics in 1971 and the Ph.D. degree in computer science in 1975, both from the University of Texas, Austin.

From 1976 to 1985, he was a faculty member with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is currently a Distinguished Professor in the Department of Computer Science, University of Massachusetts. He has held visiting positions at IBM T. J. Watson Research Center, Yorktown Heights, NY (1982–1983); Laboratories MASI, Paris, France (1989–1990); INRIA, Sophia-Antipolis, France (1996); and AT&T Labs-Research, Florham Park, NJ (1997). His research interests include networks and performance evaluation.

Dr. Towsley currently serves on the Editorial board of *Journal of the ACM*, and has previously served on several editorial boards including those of the IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE/ACM TRANSACTIONS ON NETWORKING. He was a Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE'92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has received the 1998 IEEE Communications Society William Bennett Paper Award and three best conference paper awards from ACM SIGMETRICS. He is a Fellow of the ACM.