Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks

Chitra Venkatramani, Olivier Verscheure, Pascal Frossard and Kang-Won Lee
IBM T.J.Watson Research Center
P.O.Box 218
Yorktown Heights, NY 10598.
{chitrav,ov1,frossard,kangwon}@us.ibm.com

ABSTRACT

The widespread use of the Internet and the maturing of digital video technology have led to an increase in various streaming media applications. As broadband to the home becomes more prevalent, the bottleneck of delivering quality streaming media is shifting upstream to the backbone, peering links, and the best-effort Internet. In this paper, we address the problem of efficiently streaming video assets to the end clients over a distributed infrastructure consisting of origin servers and proxy caches. We build on earlier work and propose a unified mathematical framework under which various server scheduling and proxy cache management algorithms for video streaming can be analyzed. More precisely, we incorporate known server scheduling algorithms (batching/patching/batch-patching) and proxy caching algorithms (full/partial/no caching with or without caching patch bytes) in our framework and analyze the minimum backbone bandwidth consumption under the optimal joint scheduling and caching strategies. We start by studying the optimal policy for streaming a single video object and derive a simple gradient-descent-based cache allocation algorithm to enable management of multiple heterogeneous videos efficiently. We then show that the performance of our heuristic is close to that of the optimal scheme, under a wide range of parameters.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer Communication Networks—*Distributed Systems*

General Terms

Algorithms Management Design

Keywords

Multimedia streaming, Video Server Scheduling, Proxy Cache, partial video caching

NOSSDAV'02, May 12-14, 2002, Miami, Florida, USA.

1. INTRODUCTION

The widespread use of the Internet and the maturing digital video technology have led to an increase in various streaming media applications such as webcasts, distance-learning and corporate communications. Until the advent of broadband, the constrained last-mile bandwidth was the primary bottleneck in delivering quality streaming media over the Internet. As access providers roll out faster last-mile connections, the bottleneck is shifting upstream to the provider's backbone, peering links, and the best-effort Internet. This problem can be partially addressed by "edge delivery" of streaming objects from a nearby proxy or via content distribution networks. While the edge delivery of streaming media objects will increase scale and reach of streaming media, handling streaming objects brings additional complexities at the proxies due to the large object size, long-lived nature of the objects, and isochronous delivery requirements from the users.

A variety of techniques have been proposed in the literature to efficiently utilize the backbone network bandwidth for streaming. Some of them use multicast as a means to reduce the backbone bandwidth usage: periodic broadcasting [4, 1, 5, 13], simple batching, batching with patching [6, 12, 14] and optimized patching with classes of service [9]. While these multicast-based schemes afford very low backbone bandwidth usage, they are not in widespread use due to their dependency on network-level multicast, which is not widely available over the Internet except for limited instances such as on local area networks. Additional drawbacks of the above approaches include the batching delays, and the need for clients to be able to receive multiple simultaneous streams.

With the recent proliferation of caching proxies, some of these drawbacks can be masked by storing portions of the media object in the proxy to hide the startup latency, and by using application-level multicast when network-level multicast is not available. Related work in this area [7], [8], [11], [10], [15] combined scalable video delivery with proxy caching, where the focus was mostly on transmitting a single video. In [3], the authors studied batching and patching with prefix-caching at the proxy for multiple heterogeneous videos, but analyzed each scheme independently to determine the optimal caching unit. This restricts all assets in the system to be managed using a single scheme irrespective of the difference in their access patterns. In [7], we have investigated the video streaming problem in the context of joint server scheduling and caching strategy at proxy

to minimize the aggregate backbone bandwidth usage. However, this work studied three different schemes in a disjoint framework, for a single asset only.

In this paper, we build on earlier work and propose a unified mathematical framework under which various server scheduling and proxy cache management algorithms for video streaming can be analyzed. More precisely, we incorporate known server scheduling algorithms (*batching, patching, and batch patching*), and proxy management algorithms (*full caching, partial caching, and no caching*, with an option to cache patch bytes or not) in our framework and analyze the minimum backbone bandwidth consumption under the optimal joint scheduling/caching strategy.

To this end, we first study the case of streaming a single video object and analyze the minimum backbone bandwidth consumption to meet the user requests under the optimal scheduling/caching algorithm. In particular, we evaluate the impact of the following parameters: (i) user request rate, (ii) proxy-to-client bandwidth constraints, and (iii) patch caching at the proxy. We then study the case of streaming multiple, heterogeneous video objects under the optimal scheduling/caching algorithm. Finally, we derive a simple gradient-descent-based cache allocation algorithm that can be implemented at the proxy in practice. Using simulations, we validate our algorithm performs closely to the optimal algorithm under various resource constraints.

The following section presents the problem formulation and builds the mathematical framework under which various schemes are analyzed. Section 3 analyzes the effect of various parameters on the backbone rate for a single video. In Section 4, we present a simple one-dimensional gradient-descent based algorithm to perform cache allocation for multiple, heterogeneous videos. Experimental results are presented in Section 5 and finally, Section 6 presents the conclusions and future directions for the work.

2. PROBLEM FORMULATION

We consider a video streaming architecture composed of an origin server, a proxy cache, and a finite set of media assets. We assume reliable transmissions with bounded delay over the backbone network, and assume that the access network (i.e., proxy-to-client cloud) is lossless and multicast-enabled. We also assume that the origin server is a batch-patching server while the proxy serves clients with a batching interval not exceeding the acceptable playback delay. Finally, every stream from the origin server is constrained to go through the proxy for various reasons such as content adaptation purposes (e.g., adaptive FEC, rate control), unavailability of multicast on the backbone and accounting and billing in a CDN infrastructure.

2.1 Preliminaries

Let Ω denote the finite set of media assets. An asset $\omega \in \Omega$ is characterized by its CBR streaming rate r_ω, its duration T_ω, its average request rate (or popularity) λ_ω, and its admissible playback delay d_ω specified in the Service Level Agreement (SLA).

We consider the following problem: *Given* the set Ω, *find* the per-asset joint scheduling and caching strategy that *minimizes* the aggregate backbone rate $\mathcal{R}$ *under the constraints* imposed by the network service provider infrastructure. These constraints encompass the limited capacity of the proxy in terms of storage $\mathcal{S}$ and bottleneck bandwidth $\mathcal{B}$ which may be the disk or the network bandwidth.

Parameters related to video object $\omega \in \Omega$	
r_ω	streaming rate
T_ω	playback duration
λ_ω	average request rate (# of requests per second)
d_ω	admissible playback delay
Parameters for scheduling and caching for object ω	
Δ_ω	maximum network jitter ($\Delta_\omega = \Delta$)
P_ω	prefix duration
b_ω	virtual batching interval ($b_\omega = P_\omega + d_\omega - \Delta$)
W_ω	patching window ($W_\omega = N_\omega \times b_\omega$)
α_ω	binary indicator ($\alpha_\omega = 1$ if proxy caches patch)
I_ω	interval between two regular channels

Table 1: Parameters used in this paper to describe the unified mathematical framework.

We now propose a unified framework under which various joint scheduling and caching strategies may be analyzed.

2.2 Unified Framework

We consider the following scenario illustrated in Figure 1. The proxy cache views its time axis divided into intervals $[t_{k-1}, t_k]$ of duration d_ω units of time which is the maximum admissible playback delay at the clients. All requests arriving in $[t_{k-1}, t_k)$ are batched together and a single stream is sent by the proxy to the clients in this interval. The proxy also batches these requests over a possibly larger interval we call the *virtual batching interval* denoted by b_ω, and indicated in Figure 1 by $[\tilde{t}_{i-1}, \tilde{t}_i]$. At the end of virtual batching intervals, the proxy requests the origin server for the patch streams or the regular channel. The duration of b_ω depends on the cached prefix. If the proxy does not have any prefix cached ($P_\omega = 0$), it must make a request to the origin server every interval of d_ω and forward the patch as well as the regular channel to the client. In this case, $b_\omega = d_\omega$. If the proxy has a prefix of duration P_ω cached, then it can start streaming the prefix to the clients and ideally continue batching requests until the client has played the prefix and is ready for the suffix. Hence, in general, $b_\omega = P_\omega + d_\omega$. For simplicity, we assume b_ω holds an integral number of d_ω, when $d_\omega > 0$.

As an example, consider a case when the proxy has a prefix of $P_\omega > 0$ cached and is already serving some requests for the video. A new request arrives at time t_1 in $[t_{k-1}, t_k)$, as shown in Figure 1. At time $t = t_k$, the proxy starts streaming the prefix to the client. It continues to batch this client in the current virtual batching interval b_ω, until $\tilde{t}_i$, to either join the regular channel and request the origin server for the patch stream, which it then forwards to all the clients in the interval $[\tilde{t}_{i-1}, \tilde{t}_i]$.

Clearly, ensuring continuous playback (or lossless delivery over non-ideal backbone network) at the client requires sending requests to the origin server in advance to mask the effect of the network jitter. Let Δ_ω denote this network jitter for asset ω. For the simplicity of exposition, we set Δ_ω to the maximum network jitter Δ estimated from long-term measurements. Thus, the proxy must make the requests to the origin server Δ units of time ahead to mask the network jitter. That is, $b_\omega = P_\omega + d_\omega - \Delta$.

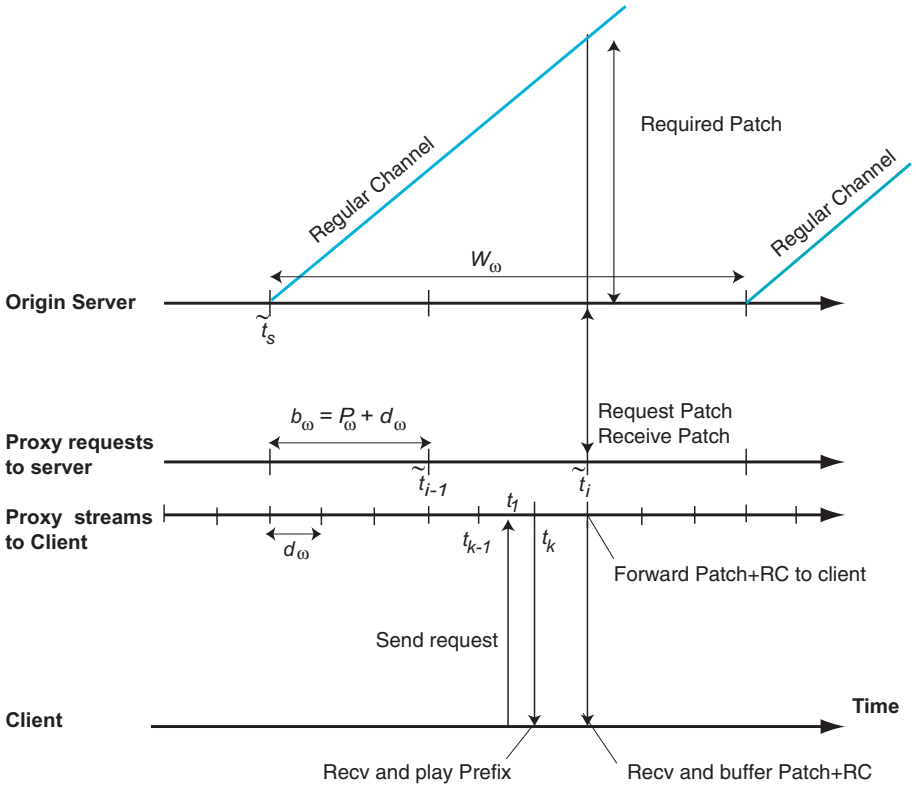

Figure 1: Unified framework for joint scheduling and caching strategies. The timing diagrams indicate the origin server, proxy and client request/transmission schedules (with $\Delta = 0$, for clarity). A client requests an asset at time $t_1 \in [t_{k-1}, t_k)$. The proxy sends the prefix at time t_k. At the end of the current b_ω interval, it requests a patch (of $2b_\omega$ in this example). It forwards the patch and the regular channel started at $\tilde{t}_s$ to all the clients in the current b_ω interval.

Now, assume the most recent regular channel (RC) for asset ω started at time $\tilde{t}_s$ (where $\tilde{t}_s < t_1$). Let W_ω denote a patching window of the server, whose duration also is an integral of b_ω, i.e., $W_\omega = N_\omega \times b_\omega$ for some integer N_ω. Then we have two cases:

- Case 1: When $P_\omega + t_k < \tilde{t}_s + W_\omega$ (i.e., patching can be applied), then the proxy cache starts forwarding the RC to the client at time $\tilde{t}_i$, which buffers the stream while playing back the prefix. Also at time $\tilde{t}_i$, the proxy requests a patch of interval $[\tilde{t}_s, \tilde{t}_i]$, forwards it to the client and optionally caches it for future requests within the same patching window. This case is illustrated in Figure 1.
- Case 2: When $P_\omega + t_k \geq \tilde{t}_s + W_\omega$ (i.e., when patching cannot be applied), then the proxy requests the origin for a new regular channel (RC) of duration $T_\omega - P_\omega$ (i.e., the suffix, since the prefix of duration P_ω is stored in the proxy), at time $\tilde{t}_i$, and forwards it to the clients.

Note that this framework encompasses various existing server scheduling algorithms and proxy cache management algorithms developed for streaming media applications. More precisely, it can model the following caching strategies: (i) Full caching ($P_\omega = T_\omega$). (ii) Partial caching ($0 < P_\omega < T_\omega$). (iii) No prefix caching ($P_\omega = 0$). At the same time, it models the following server scheduling schemes: (i) Batching ($b_\omega > 0$ and $N_\omega = 0$). (ii) Patching ($b_\omega = 0$ and $N_\omega > 0$). (iii) Batch-patching ($b_\omega > 0$ and $N_\omega > 0$). In addition, we can model the case when the proxy either temporarily caches the patch bytes ($\alpha_\omega = 1$) or not ($\alpha_\omega = 0$).

We now develop a set of equations for the aggregate backbone rate $\mathcal{R}_\omega$, the proxy storage $\mathcal{S}_\omega$ and network bandwidth $\mathcal{B}_\omega$ requirements for a media asset $\omega \in \Omega$, averaged over the interval I_ω. In the following we assume a Poisson request arrival distribution such that $q = e^{-\lambda_\omega b_\omega}$ is the probability to have an empty batch of duration b_ω units of time.

The *aggregate backbone rate* at stationary state, $\mathcal{R}_\omega$, includes the transmission of patches (μ_ω) during b_ω and the suffix of duration ($T_\omega - P_\omega$) from the origin server, and is given by:

$$\mathcal{R}_\omega = \frac{\mu_\omega b_\omega r_\omega + (T_\omega - P_\omega) r_\omega}{I_\omega}, \tag{1}$$

where μ_ω denotes the average number of transmitted patches of asset ω:

$$\begin{aligned} \mu_\omega &= \alpha_\omega \frac{q^{(N_\omega+1)} - (N_\omega+1)q + N_\omega}{1-q} \\ &\quad + (1-\alpha_\omega)(1-q)\frac{N_\omega(N_\omega+1)}{2}, \end{aligned} \tag{2}$$

and I_ω represents the interval duration between two regular

channels:

$$I_\omega = (N_\omega + 1)b_\omega + {\lambda_\omega}^{-1} \ . \quad (3)$$

The derivation of μ_ω is as follows: When the proxy caches the patch bytes ($\alpha_\omega = 1$), if there is a request in batch i and none in the later batches in I_ω, the proxy fetches exactly $ib_\omega r_\omega$ patch bytes from the server, for this interval. When the proxy does not cache the patch bytes ($\alpha_\omega = 0$), then the proxy fetches up to $\sum ib_\omega r_\omega$ patch bytes from the origin server.

$$\mu_\omega = \alpha_\omega \left(\sum_{i=1,N} (1-q)iq^{(N-i)} \right) + (1-\alpha_\omega) \left((1-q) \sum_{i=1,N} i \right) \quad (4)$$

This equation can be simplified to Equation 2.

The *proxy cache occupancy* $\mathcal{S}_\omega$, averaged over I_ω, is given by:

$$\begin{aligned} \mathcal{S}_\omega &= P_\omega r_\omega + \Delta\ r_\omega \frac{(T_\omega - P_\omega)}{I_\omega} \\ &\quad + \frac{\mu_\omega b_\omega r_\omega \left[\alpha_\omega N_\omega b_\omega + (1-\alpha_\omega)\ \Delta\right]}{I_\omega}. \end{aligned} \quad (5)$$

The first term indicates the storage for the prefix that stays for the entire duration, the second term represents the storage expended for the jitter-buffer associated with the suffix stream and the third term represents the storage for the patch bytes — if the patches are cached, then a storage of $\mu_\omega b_\omega r_\omega$ is used for a duration of $N_\omega b_\omega$ and if the patches are not cached, then a storage of Δr_ω is used for the period of $\mu_\omega b_\omega$. It can be noted that when the patches are not cached, the storage utilization simplifies to the prefix plus additional (Δr_ω) buffers for all the streams received from the origin and forwarded by the proxy, which is $\mathcal{R}_\omega$:

$$\mathcal{S}_\omega = P_\omega r_\omega + \Delta \mathcal{R}_\omega \qquad when\ \alpha_\omega = 0$$

The proxy batches client requests in intervals of d_ω which is the maximum playback delay. The proxy streams the prefix for every batch of d_ω, and forwards the patch bytes (every b_ω), and the regular channel (every I_ω) received from the origin server, to the client. Thus the *proxy network bandwidth*, assuming a multicast-enabled network from the proxy server to the clients, $\mathcal{B}_\omega$, averaged over I_ω, is expressed as:

$$\mathcal{B}_\omega = \begin{cases} \frac{(1-e^{-\lambda_\omega d_\omega})(N_\omega+1)b_\omega P_\omega r_\omega}{I_\omega d_\omega} + R_\omega & if \quad d_\omega > 0 \\ \frac{\lambda_\omega (N_\omega+1) b_\omega P_\omega r_\omega}{I_\omega} + R_\omega & if \quad d_\omega = 0 \end{cases} \quad (6)$$

When the playback delay is non-zero, the first term represents the total bytes of prefix (which is sent separately to each batch of clients in the interval d_ω), the second term represents the total bytes of the suffix (a single stream is sent to all the clients in the interval I_ω) including the patch bytes forwarded to the clients in the interval I_ω. When $d_\omega = 0$, then one prefix stream is sent to each client, besides the patches and the regular channel.

3. ANALYSIS

In this section we analyze the effect of various parameters on the backbone bandwidth usage in order to determine the optimal server scheduling and proxy caching strategies that will jointly minimize the backbone usage.

3.1 Backbone usage with cached prefix

Equation 1 shows that the backbone rate is determined by various parameters – the size of the cached prefix P_ω, the playback duration T_ω, the streaming rate r_ω and the popularity λ_ω. Intuitively, we expect the size of the cached prefix must affect the backbone rate the most. To simplify the analysis and understand this effect, we set the value of Δ, the network jitter, to zero and also set α_ω, the indicator to cache the patch bytes, to false.

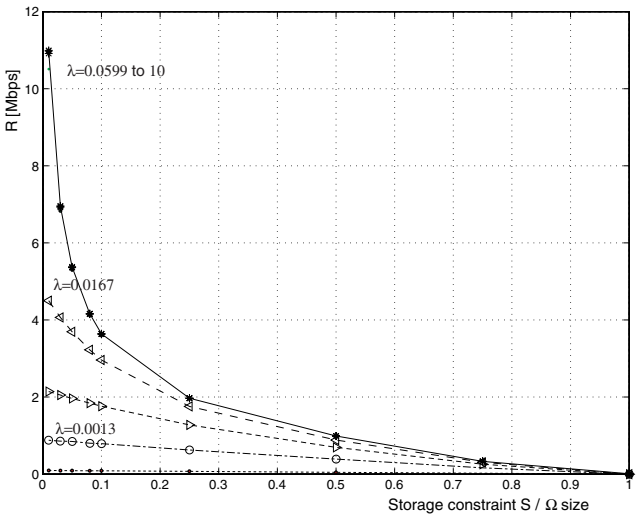

Figure 2: Normalized bandwidth usage vs. cached prefix size.

In Figure 2, we plot the value of the minimum backbone rate R_ω normalized over the streaming rate, against the size of the prefix cached at the proxy, for various values of λ_ω in the range of 10^{-4} and 10 requests per second.

Two significant observations can be made from this figure:

- The normalized backbone rate $\mathcal{R}_\omega$ decreases with the prefix size (P_ω). Here, N_ω is chosen such that it minimizes $\mathcal{R}_\omega$ for a chosen prefix size and is denoted by $N_\omega^\star$[1]. This decrease in $\mathcal{R}_\omega$ can be intuitively explained as follows. Setting $N_\omega = N_\omega^\star$ implies that the backbone rate is minimized for a given asset (i.e., T_ω, r_ω, λ_ω) and a prefix size P_ω. In other words, it equivalently minimizes the streaming rate of a virtual asset $\tilde{\omega}$ of duration $T_\omega - P_\omega$, the other parameters staying unchanged. Increasing P_ω to $P_\omega + \delta$ (with $\delta \geq 0$) is then equivalent to decreasing the size of the virtual asset $\tilde{\omega}$ to a duration of $T_\omega - P_\omega - \delta$. Since the minimal streaming rate of an asset of duration $T_\omega - P_\omega - \delta$ cannot be larger than the streaming rate of an asset of duration $T_\omega - P_\omega$, the backbone bandwidth R_ω is always decreasing with the prefix size P_ω. This can also be formally proved by verifying that the derivative $\frac{\partial \mathcal{R}_\omega}{\partial P_\omega}|_{N_\omega = N_\omega^\star}$ is indeed negative $\forall P_\omega \in [0, T_\omega]$.
- The popularity λ_ω has a significant effect on R. We observe that the plots are distinct for smaller values of λ_ω, but have near-complete overlap beyond $\lambda_\omega = 0.21$

[1] $N_\omega^\star$ minimizes $\mathcal{R}_\omega$ and is obtained by setting $\frac{\partial \mathcal{R}_\omega}{\partial P_\omega} = 0$ and solving for N_ω.

requests per second. At this value, the probability of seeing at least one request per batching interval b_ω becomes close to 1. Beyond this, requests get clumped into batches at the proxy and this does not affect the backbone rate any more. This threshold value depends on the batching interval b_ω of a video.

3.2 Proxy bandwidth constraint

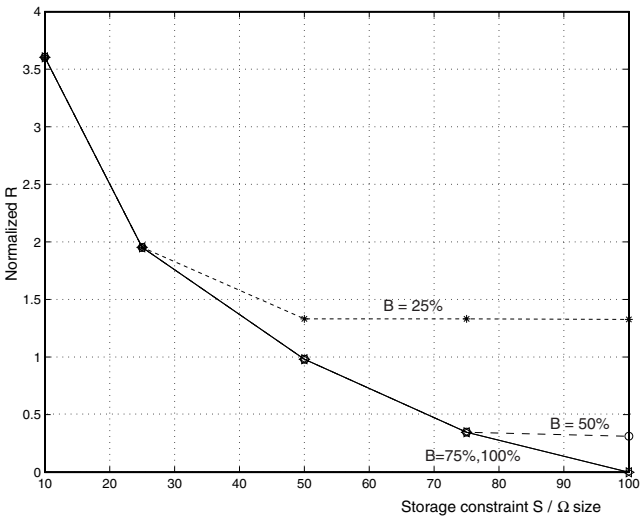

Figure 3: Normalized backbone rate vs. cached prefix size (when the bandwidth out of the proxy is constrained).

In the above analysis, the bandwidth out of the proxy was unconstrained, leading to the minimal backbone bandwidth usage. In Figure 3, we study the effect of constraining the network bandwidth out of the proxy. Once the proxy bandwidth is saturated the video cannot be streamed to the clients. The figure plots the region where the prefix is larger than 10% of the video. It shows that once the proxy bandwidth limit is reached, R_ω cannot be reduced further by increasing the prefix size. In Section 4, we analyze the effect of constrained bandwidth in the case of multiple, heterogeneous videos.

3.3 Caching patch bytes at the proxy

Next, we examine the effect of temporarily caching the patch bytes in the proxy. In Equation 1, this choice is indicated by the parameter α_ω. Figure 4 plots R_ω with respect to the prefix size for the following two cases: (i) patches are not cached ($\alpha_\omega = 0$) and (ii) patches are always cached ($\alpha_\omega = 1$), for all $\omega \in \Omega$. The figure clearly indicates that there is no savings in $\mathcal{R}$ by caching the patch bytes at the proxy. In other words, this implies that whenever space is available, it is more beneficial to use it to cache the prefix of the video rather than use it to cache patch bytes. This is because increasing the prefix increases the batching period and also reduces the suffix stream, both of which contribute to savings in $\mathcal{R}$.

However, the need to cache patch bytes may arise when the client, such as a mobile client, has limited bandwidth capacity. As described in [7], if only the prefix is cached at the proxy, the client could end up receiving up to three

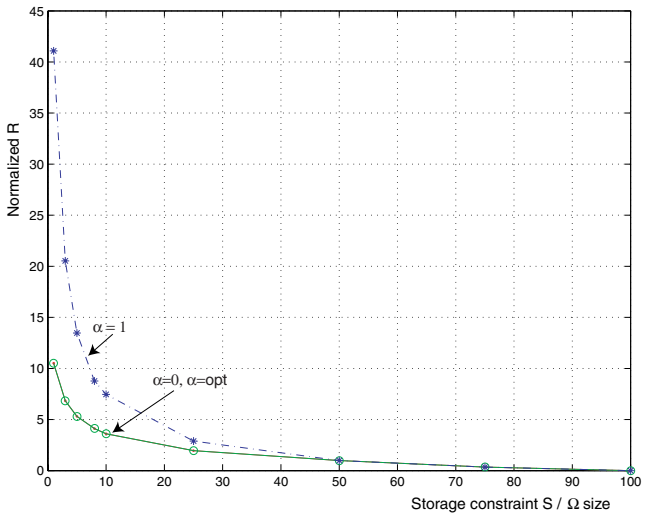

Figure 4: Normalized backbone rate $\mathcal{R}$ vs. cached prefix size with: $\alpha_\omega = 0$ and $\alpha_\omega = 1$.

simultaneous streams – prefix from the proxy, patch and regular channel from the origin server. In such cases, the proxy might be forced to cache the patch when the client cannot receive more than two simultaneous streams. In the remainder, we assume that the proxy does not cache the patch bytes (i.e., $\alpha = 0$).

Thus far, we have studied the effects of various scheduling and caching schemes on the usage of the backbone, in the context of a single asset. We found that the ideal scheduling policy at the origin server is batch-patching, with prefix caching performed at the proxy. The proxy can determine the patching window $N_\omega * b_\omega$ based on the prefix size and request new regular channels from the origin server when this window is crossed.

4. MULTIPLE HETEROGENEOUS VIDEOS

In this section, we first formulate and solve an optimization problem to determine the joint server-scheduling and proxy caching strategy that minimizes the backbone bandwidth usage for a given set of assets. We then use the findings in the previous section such as the non-increasing property of $\mathcal{R}$, to design a near-optimal gradient-descent-based cache allocation algorithm for multiple, heterogeneous videos.

4.1 Preliminaries

We aim to solve the following non-linear optimization problem (*P1*):

Problem P1 *Given* a set of media assets $\omega \in \Omega$ characterized by their streaming rates r_ω, durations T_ω, access rates λ_ω and admissible playback delays d_ω. *Given* also a proxy cache of storage capacity S and proxy network bandwidth B, and a backbone network with bounded jitter Δ. *Find* the tuples $(P_\omega, N_\omega, \alpha_\omega)$ that *minimize* the aggregate backbone rate $\mathcal{R}$ such that (i) $\sum_{\omega\in\Omega} \mathcal{S}_\omega \leq S$ (ii)$\sum_{\omega\in\Omega} \mathcal{B}_\omega \leq B$ and (iii) $\max(0, \Delta - d_\omega \leq P_\omega \leq T_\omega)$ (iv) $0 \leq N_\omega \leq \frac{T_\omega - P_\omega}{b_\omega}$ (v) $\alpha_\omega \in \{0, 1\}$.

Let $A = |\Omega|$ denote the cardinality of the set Ω. We are facing a non-linear optimization problem with $3A$ unknowns

($[N_\omega, P_\omega, \alpha_\omega]$ for each asset ω) and $2 + 6A$ inequality constraints (proxy space and bandwidth constraints + upper and lower bounds on the values of $[N_\omega, P_\omega, \alpha_\omega]$ for each asset ω). Note that the value of A may be as high as several thousands. Clearly, blindly applying a non-linear optimization technique to Problem *P1* may not lead to a solution (i.e., no convergence), or at least not in a reasonable amount of time (i.e., slow convergence). Choosing a tool which exploits some property or structure of the problem usually yields a solution more efficiently. We chose the *LANCELOT* optimization tool [2] to solve our problem, since it is designed for non-linear problems with a structure very similar to ours.

Note that Problem $P1$ may not have a solution. Recall that we made the assumption of a system where all streams to a client (i.e., prefix, patch and regular stream) are to pass through the proxy for various reasons such as the unavailability of multicast from the origin server to the clients, accounting/billing in a CDN infrastructure and content adaptation purposes (e.g., transcoding, fingerprinting). That is, setting all P_ω and N_ω to zero leads to $\sum_{\omega \in \Omega} \mathcal{B}_\omega > 0$, which may exceed the proxy bandwidth constraint. Clearly, the solvability of $P1$ depends on the constraint bounds. The selection of the most appropriate set Ω such that $P1$ has a solution, is beyond the scope of this work.

In the following, experiments are performed with $A = 50$ media assets whose λ_ω values follow the Zipf-distribution with $\xi = 0.8$, and uniformly distributed r_ω and T_ω in the discrete sets $r_\omega \in [800, 1600, 2400]$ kbps and $T_\omega \in [60, 90, 120]$ minutes. Using the earlier finding that caching the patch bytes is not beneficial in terms of backbone usage, we set $\alpha = 0$ in *P1*. Thus, Problem *P1* reduces to $2A$ unknows and $2 + 4A$ inequality constraints.

The remainder of this section is organized as follows: First, we propose a discrete 1-D gradient-descent technique provably optimal under the assumptions of null network jitter (i.e., $\Delta = 0$) and infinite proxy bandwidth (i.e., $B = \infty$). We then progressively relax these assumptions and demonstrate the appropriateness of the proposed algorithm. Each analytic step is verified by comparing the results with the optimal solution given by *LANCELOT*.

4.2 Gradient-descent based proxy allocation

Most of the research work in this area neglects the effect of the backbone jitter (i.e., $\Delta = 0$ is assumed). Also, proxy bandwidth is usually not limited (i.e., $B = \infty$). We show that, under these assumptions, one can find the optimal solution very efficiently with a gradient-descent based algorithm.

In this special case, the proxy cache size (Equation 5) reduces to $S_\omega = P_\omega r_\omega$. That is, the prefix size is the only parameter that may influence the required size of the cache. Moreover the proxy bandwidth is not a constraint here. Our objective is to minimize the aggregate backbone rate $\mathcal{R}$, which depends on both P_ω and N_ω. Therefore, among all possible values of N_ω, $N_\omega^\star$ such that $\frac{\partial R_\omega}{\partial N_\omega} = 0$ clearly leads to the minimum backbone rate for any given P_ω under the storage constraint. We thus replace the unknown N_ω in Equation 1 by its optimal value $N_\omega^\star$, which is a function of P_ω. The resulting problem consists of finding the prefix sizes P_ω that minimize the backbone rate $\mathcal{R}$, a function of P_ω only, such that $\sum_{\omega \in \Omega} P_\omega r_\omega \leq S$.

We propose a simple optimal discrete gradient descent-based algorithm. The pseudocode of Algorithm *A1* is presented in Figure 5. Let us assume c to be the smallest unit of cache allocation and all allocations are in multiples of this unit. The algorithm first computes the value R_ω for each increment of the prefix size in units of c. It then determines the incremental savings in R_ω on growing the prefix by increments of c. Once this is determined, the algorithm greedily fills up the cache by growing prefixes such that the gain in R_ω is maximized. Since R_ω decreases monotonically as the prefix size increases as shown in Figure 2, the above greedy algorithm results in the asymptotically-optimal allocation where the accuracy depends on the caching granularity c.

```
PrefixAlloc(lambda)
{
  Rgain[i][j] = getR(lambda[i],j*c) -
          getR(lambda[i],(j-1)*c);
  // walk through the lists, determine prefix size
  // block[i]:  index of the block to pick next
  // for asset i.
  for i=1 to NumVideos
      block[i] = 1; // initialize block[i]
  // while cache is not full.
  while (allocSpace <= cacheSize){
    // find the block with maximum benefit
    for i=1 to NumVideos
      for j=2 to numBlocks
        video = find max(Rgain[i][block[i]])
    RgainTotal += Rgain[video][block[video]];
    block[video]++;
    Prefix[video] += c;
    allocSpace += blockSize;
  }
}
```

Figure 5: Algorithm A1: Proxy Prefix Allocation.

5. EXPERIMENTAL RESULTS

In this section, we examine the impact of Δ and B on the quality of the solution provided by Algorithm *A1*. We compare the experimental results from Algorithm *A1* with the optimal solution obtained using LANCELOT.

5.1 Bounded Jitter and Unconstrained Proxy Bandwidth

We first consider the case where $\Delta = 0$ and $B = \infty$. Figure 6 first shows the solution from Algorithm *A1* with the optimal for a set of $A = 50$ media assets with various λ_ω, r_ω and T_ω as described earlier. The figure shows the evolution of the aggregate backbone rate $\mathcal{R}$ with the ratio between proxy storage capacity S and $\sum_{\omega \in \Omega} T_\omega r_\omega$. It shows that Algorithm *A1* offers the same performance as the optimal obtained using LANCELOT. As expected, the gradient-descent based method is near-optimal under the previous assumptions. We can also see that a fully optimized proxy management offers a significant backbone bandwidth gain compared to a simple prefix caching scheme (with batching over the prefix). The prefix caching scheme parameters have been optimized under a problem formulation similar to $P1$, by imposing $N_\omega = 0, \forall \omega \in \Omega$. This implies that the server does not perform batch-patching.

Moving from a null-jitter network to a real network (i.e., $\Delta_\omega > 0$) implies that the required cache space (Equation 5) now depends on N_ω since space needs to be expended for the jitter buffers. Recall that Equation 5 may be written as $S_\omega = P_\omega r_\omega + \Delta R_\omega$. The dependence of S_ω on N_ω is reflected by R_ω. Note that the value N_ω^* defined earlier

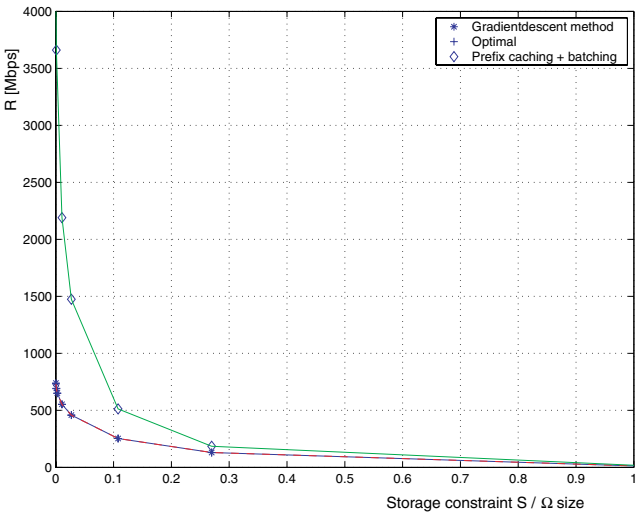

Figure 6: Aggregate backbone rate vs. proxy storage capacity for a heterogeneous set of $A = 50$ media assets, $\Delta = 0$ and $d = 3$ seconds, using Algorithm A1 and the optimal. The simpler algorithm where $N_\omega = 0$ for all $\omega \in \Omega$ (pure prefix batching) is also shown.

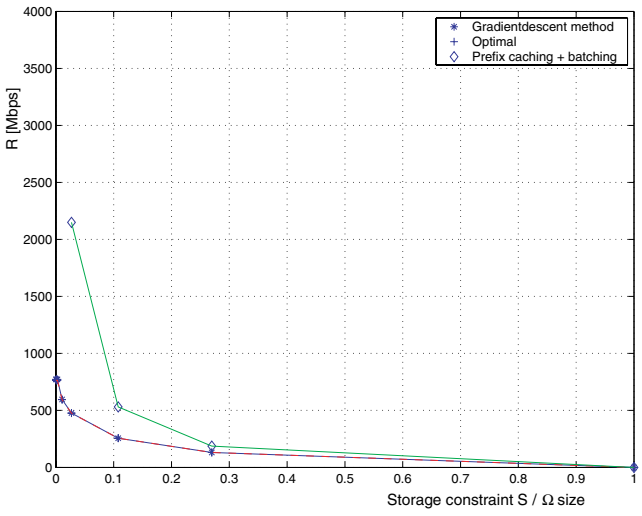

Figure 7: Aggregate backbone rate vs. proxy storage capacity for a heterogeneous set of $A = 50$ media assets, $\Delta = 2$ seconds and $d = 3$ seconds, using Algorithm A1 and the optimal. The simpler method where $N_\omega = 0$ for all $\omega \in \Omega$ (pure prefix batching) is also shown.

minimizes the backbone rate for asset ω, and thus minimizes the corresponding required storage as well (for any value of P_ω). The summation over all assets in Ω preserves the property such that Algorithm *A1* is again asymptotically optimal with respect to the cache allocation unit c when a real network is considered.

Figure 7 compares the solution from Algorithm *A1* with the optimal when $\Delta = 2$ and $d = 3$ seconds. As expected, $\Delta > 0$ does not impact the quality of the solution given by *A1*. We observe that optimal batch patching again greatly reduces the backbone rate as compared to the prefix caching scheme with plain batching; especially when the proxy can store much less than the entire set Ω. It is also interesting to note that the prefix caching scheme has no solution for small storage constraints (below 5%) because the jitter buffer needs to be accommodated.

5.2 Bounded Jitter and Constrained Proxy Bandwidth

Finally we add the constraint on the proxy bandwidth (i.e., $B_\omega < \infty$). The proxy bandwidth B_ω depends on N_ω in a more complicated way. Equation 6 shows that N_ω plays a role both in the first and the second term. In a first approximation, let $1/\lambda_\omega \ll b_\omega(N_\omega + 1)$. If this approximation holds, then I_ω (denominator) cancels out the $b_\omega(N_\omega + 1)$ factor (numerator) in Equation 6. Thus, the dependence on N_ω is reflected by the second term only (that is, R_ω). Again, the value N^*_ω minimizes the proxy bandwidth for any value of P_ω. Given the summation over all assets in Ω, Algorithm *A1* stays close to optimal.

From Equation 6, the approximation holds if either $\lambda_\omega \gg 1$ or P_ω is relatively small (which is the main multiplying factor of the error). In this case, it can be shown that optimal values of N with respect to both R and B are close [2]. In our experiments, we do not assume $\lambda \gg 1$. Instead we show that the results from Algorithm *A1* slightly diverge from the optimal solution when increasing the size of the proxy cache (i.e., increasing P_ω, in average). Figures 8 and 9 compare the results from Algorithm *A1* with the optimal for a highly and moderately constrained proxy bandwidth B. The figures show the evolution of the aggregate backbone rate $\mathcal{R}$ with the ratio between proxy storage capacity S and $\sum_{\omega\in\Omega} T_\omega r_\omega$, for $B = 2.67$ Gbps and $B = 1.335$ Gbps, respectively. The gradient-descent based algorithm stays close to optimal for low storage or bandwidth constraints. However, it then slightly diverges from the optimal. Nevertheless, the sub-optimal gradient-descent method stays a viable, and very simple solution to the proxy management problem.

6. CONCLUSIONS

In this paper, we address the problem of efficiently streaming heterogeneous videos to clients over a distributed infrastructure consisting of origin servers and proxy caches. We build on earlier work and propose a unified mathematical framework under which various server scheduling and proxy cache management algorithms such as batching, patching, batch-patching with partial caching at the proxy, can be analyzed. We formulate an optimization problem to determine the optimal scheduling and caching strategies such that the proxy space and bandwidth constraints are respected. We also propose a simple one-dimensional gradient-descent algorithm to solve the problem and show that the results closely match the optimal solution obtained using the well-known

[2]The assumption is true if the following condition is verified :

$$\frac{\lambda^2\, b^2\, r\, P\, (1 - e^{-\lambda\, d})}{d} \ll (1-e^{-\lambda\, b})\,(\lambda\, b+1)+2\, b\, \lambda^2\, (T-P)\,. \quad (7)$$

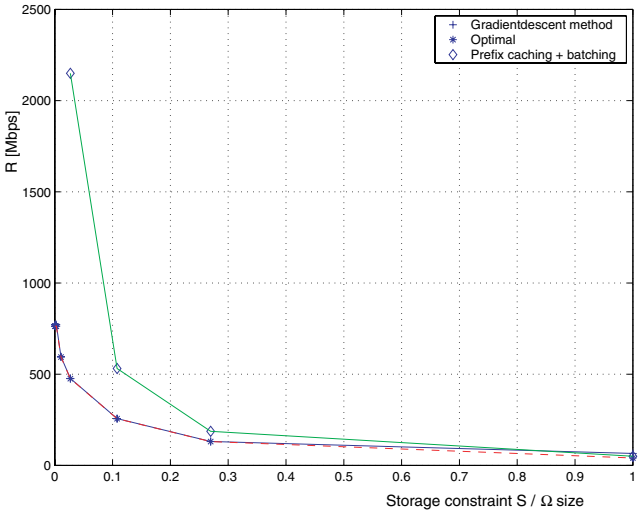

Figure 8: Aggregate backbone rate vs. proxy storage capacity for a heterogeneous set of $A = 50$ media assets, $\Delta = 2$ seconds, $d = 3$ seconds and $B = 2.67$ Gbps using Algorithm A1 and the optimal. The simpler method where $N_\omega = 0$ for all $\omega \in \Omega$ (pure prefix batching) is also shown.

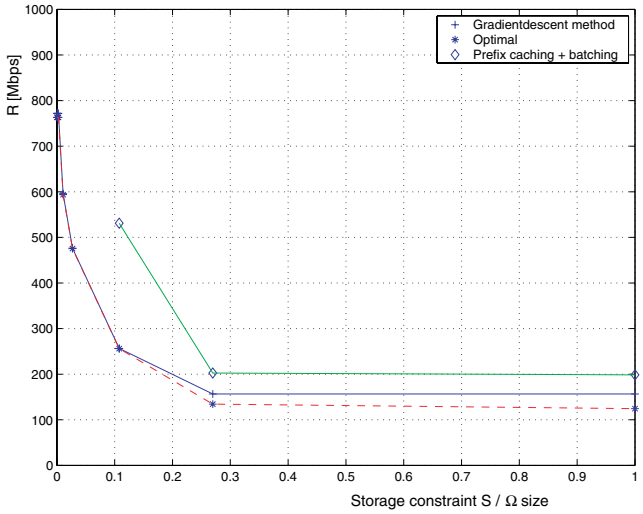

Figure 9: Aggregate backbone rate vs. proxy storage capacity for a heterogeneous set of $A = 50$ media assets, $\Delta = 2$ seconds, $d = 3$ seconds and $B = 1.33$ Gbps using Algorithm A1 and the optimal. The simpler method where $N_\omega = 0$ for all $\omega \in \Omega$ (pure prefix batching) is also shown.

LANCELOT optimization tool. From our experiments, we also determine that prefix caching at the proxy with batch-patching enabled at the origin server, is the most effective strategy.

Acknowledgements

The authors would like to thank Andrew Conn from the I.B.M. T.J.Watson Research Center for his help with understanding the applicability of the *LANCELOT* optimization tool to solve our problem.

7. REFERENCES

[1] A.Dan, D.Sitaram and P.Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. *Proceedings of ACM Multimedia*, Oct. 1994.

[2] A.R. Conn, N.I.M. Gould and Ph.L.Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Spinger-Verlag, 1992.

[3] B.Wang, S.Sen, M.Adler and D.Towsley. Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. *To appear in Proceedings of IEEE Infocom*, 2002.

[4] C.C. Aggarwal, J.L. Wolf and P.S. Yu. A permutation based pyramid broadcasting scheme for Metropolitan VOD systems. *Proc. of the IEEE International Conference on Multimedia Systems*, June 1996.

[5] K.A. Hua and S.Sheu. Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan VOD systems. *Proceedings of the ACM SIGCOMM*, 1997.

[6] K.A. Hua, Y.Cai and S.Sheu. Patching: A multicast technique for True On-Demand Services. *Proceedings of ACM Multimedia*, Sept. 1998.

[7] O.Verscheure, C.Venkatramani, P.Frossard and L.Amini. Joint Server Scheduling and Proxy Caching for Video Delivery. *Computer Communications, Special Issue*, 25, March 2002.

[8] P.Frossard and O.Verscheure. Batched Patch Caching for Streaming Media. *IEEE Communications Letters*, 6(4), April 2002.

[9] P.White and J.Crowcroft. Optimized Batch Patching with Classes of Service. *ACM Communications Review*, October 2000.

[10] S.-H. Gary Chan. Operation and Cost Optimization of a Distributed Servers Architecture for On-Demand Video Services. *IEEE Communications Letters*, 5(9), Sept. 2001.

[11] S.Ramesh, I.Rhee and K.Guo. Multicast with cache (mcache):An adaptive zero-delay video-on-demand service. *Proceedings of IEEE Infocom*, April 2001.

[12] S.Sen, L.Gao, J.Rexford and D.Towsley. Optimal patching scheme for efficient multimedia streaming. *Proc. of IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[13] S.Viswanathan and T.Imielinski. Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting. *Multimedia Systems*, 4, August 1996.

[14] Y.Cai, K.Hua and K.Vu. Optimizing Patching Performance. *Proceedings of ACM/SPIE Multimedia Computing and Networking*, Jan 1999.

[15] Y.Guo, S.Sen and D.Towsley. Prefix Caching assisted Periodic Broadcast: Framework and Techniques to Support Streaming for Popular Videos. Technical Report TR 01-22, UMass CMPSCI, 2001.