Accelerating Internet Streaming Media Delivery using Network-Aware Partial Caching[*]

Shudong Jin
Computer Science Department
Boston University, Boston, MA 02115
jins@cs.bu.edu

Azer Bestavros
Computer Science Department
Boston University, Boston, MA 02115
best@cs.bu.edu

Arun Iyengar
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
aruni@watson.ibm.com

Abstract

Internet streaming applications are affected by adverse network conditions such as high packet loss rates and long delays. This paper aims at mitigating such effects by leveraging the availability of client-side caching proxies. We present a novel caching architecture and associated cache management algorithms that turn edge caches into accelerators of streaming media delivery. A salient feature of our caching algorithms is that they allow partial caching of streaming media objects and joint delivery of content from caches and origin servers. The caching algorithms we propose are both network-aware and stream-aware; they take into account the popularity of streaming media objects, their bit-rate requirements, and the available bandwidth between clients and servers. Using realistic models of Internet bandwidth derived from proxy cache logs and measured over real Internet paths, we have conducted simulations to evaluate the performance of various cache management alternatives. Our experiments demonstrate that network-aware caching algorithms can significantly reduce service delay and improve overall stream quality. Our experiments also show that partial caching is particularly effective when bandwidth variability is not very high.

1 Introduction

The increasing popularity of multimedia content on the Internet has stimulated the emergence of streaming media applications. This trend is partly prompted by the wide distribution of software from industry such as RealNetworks and Microsoft Windows Media, and the adoption of application-level protocols such as RTSP [22] and RTP [21] for streaming media transmission. Access to streaming media requires a high and stable transmission rate. To meet such requirements, customers and ISPs typically upgrade their connections to the Internet (e.g., by going to higher bandwidth services). While necessary, this upgrade of the "last mile" bandwidth does not translate to improved quality of service (QoS) for streaming media access. Specifically, for such upgrades to yield the desired effects, Internet and streaming media servers must be able to handle the increased demand. To that end, several issues need to be addressed.

First, it is not clear whether the physical bandwidth of the Internet can match the pace of increasing demand. Internet resources are shared by a large number of connections; an individual customer can only expect to get a portion that reflects his/her "fair share" of the physical available bandwidth. This is the result of requirements that network transport protocols for streaming media be "TCP-friendly" [9, 18]. Thus, the bandwidth available to an individual connection is likely to be limited by the unpredictable sharing of Internet resources beyond the over-provisioned last mile. Second, the Internet exhibits extreme diversity in terms of end-to-end packet loss rates and round-trip delays. This diversity is made worse by the bursty nature of these metrics as evidenced by findings in a number of recent studies on Internet traffic characteristics [8, 16, 10]. Thus, Internet dynamics deprive streaming applications of the smoothness conditions necessary to meet customer expectations. Finally, even if the Internet transport would meet its end of the bargain, streaming media servers may themselves become a significant bottleneck, especially at times of high usage.

Combined, all these factors suggest the importance of efficient and robust streaming content delivery mechanisms that go beyond the point-to-point, server-to-client delivery of streaming media content. Caching of streaming media is a good example of such mechanisms. By placing streaming media objects closer to customers, network bandwidth requirements are reduced, user-perceived quality is improved, and demand on servers is decreased. Caching techniques have been widely explored for serving traditional Web objects such as HTML pages and image files [5, 6, 12, 25, 26]. For streaming media objects, caching becomes especially at-

[*]This research was supported in part by NSF (awards ANI-9986397 and ANI-0095988) and by IBM. Part of this work was done while the first author was at IBM Research in summer 2001.

tractive due to the static nature of content, long duration and predictable sequential nature of accesses, and high network resource requirements.

Paper Contributions and Overview: This paper proposes a novel technique that turns edge caches into accelerators of streaming media delivery. A salient feature of our proposed technique is the ability of a proxy to *partially* cache a streaming media object. Partial or whole streaming media objects are placed in caches closer to clients to accelerate access and improve stream quality. The cache management algorithms we propose are both *stream-aware* and *network-aware*; they not only account for the popularity of streaming media objects, but also they account for the bitrate requirements of these objects, and the network conditions, such as the available bandwidth between server, client, and caches. Using realistic synthetically-generated access workloads, our simulations show that our proposed acceleration algorithms can efficiently utilize cache space to reduce streaming media service Delay and improve stream quality. Our simulations are unique because they rely on realistic models of available bandwidth conditions, reflecting bandwidth characteristics we observed in proxy cache (NLANR proxy cache [15]) logs and from real Internet paths.

The remainder of the paper is organized as follows. We first formalize the cache problem and propose our service acceleration algorithms. Section 3 describes the methodology of our performance evaluation experiments. Section 4 presents results from our simulations. We revisit related work in Section 5, and end in Section 6 with conclusion and directions for future research.

2 Caches as Accelerators: Algorithms

In this section, we describe an architecture that uses caches to accelerate streaming media access. We formalize the cache management problem and propose algorithms based on this formalization.

2.1 Architecture of Streaming Media Delivery

We consider an Internet streaming media delivery architecture consisting of: (1) caches deployed at the edge of the Internet; (2) streaming media objects that are replicated either *entirely or partially* on these caches; and (3) clients whose requests are satisfied (possibly *jointly*) by servers and caches. Figure 1 illustrates such an architecture.

We first examine how streaming media accesses may proceed in the absence of caches. A client may request any objects available from an “origin” server. To do so, the client measures the bandwidth between the server and itself. If the bandwidth is abundant, the client can play the stream immediately (it may still need to buffer a few initial frames of the stream in order to tolerate network jitters). If the bandwidth is not high enough to support immediate and continuous play

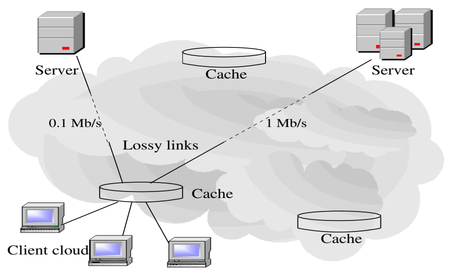


Figure 1. Using caches to accelerate Internet streaming access and improve quality. Streaming media accesses are served by both servers and caches.

of the stream at an acceptable QoS, e.g., the object playback rate is 400 Kb/s but the bandwidth is only 200 Kb/s, two choices are possible: (1) it introduces a delay, during which it prefetches a prefix of the stream, before continuously playing the stream, or (2) it negotiates with the server and degrades the stream quality. For the previous example, the client can retrieve a half of a layer-encoded object.

Now, we turn our attention to streaming media access in the presence of caches deployed closer to the client (e.g. within the “last mile”). Rather than relying solely on the origin server, a client could retrieve an entire or partial stream from a neighboring cache with higher bandwidth. It does so by measuring the bandwidth from the server and the bandwidth from the cache, and deciding if it is possible for *both* the server and the cache to jointly support immediate and continuous play. For the previous example, if half of the object has been cached, then the client can immediately and continuously play out the object. While the client is playing out the object from the cache, the other half is prefetched from the server. Generally, with caches, clients are less affected by the limited and variable bandwidth from the server.

2.2 Formalization of Caching Problem

As we hinted earlier, “network awareness” is an important aspect of the streaming media caching techniques we propose in this paper. To appreciate this, consider the paths from a cache to two origin servers (shown in Figure 1), whereby one path has a bandwidth of 1 Mb/s while the other can only support streaming at a 0.1 Mb/s rate. Intuitively, it is more important and valuable to cache objects available from the second origin server.

Before we formalize the cache management problem, we make several assumptions. First, we assume that the bandwidth of a path is constant over some appropriate time scale. In a real setting, and as we explained earlier, the bandwidth achievable over a given Internet path may vary significantly with time. In section 2.5, we relax this assumption and show how our algorithms can be modified to manage bandwidth variability. Second, we assume that the objective of the system is to minimize service delay. We define service delay to be the total delay perceived by the client before the playout of an object (at some acceptable QoS) can begin. Third, we assume that streaming media objects are encoded using a constant bit-rate (CBR) technique. For variable bit-rate (VBR) objects, we assume the use of the optimal smoothing technique [20] to reduce the burstiness of transmission rate. Finally, we assume abundant bandwidth at the client side. Also, we assume that clients (a client cloud) behind a caching proxy are homogeneous.

Let N be the number of streaming media objects available for access. For any such object i, we denote by T_i the object's duration in seconds, by r_i the object's CBR encoding in Mb/s, by λ_i the arrival rate of requests for that object, and by b_i the bandwidth between the cache and the original server storing that object. The notation y^+ means that $y^+ = y$ if $y > 0$ and 0 otherwise.

Let C denote the total capacity of the cache, and let x_i denote the size of the cached part of object i. Upon requesting object i, the playout of that object must be delayed by $[T_i r_i - T_i b_i - x_i]^+/b_i$. Notice that $T_i r_i$ reflects the total size of the requested object and that $T_i b_i$ reflects the size of the portion of the object that can be streamed during playout.

The optimization problem that the cache management algorithm must address is thus to find a set of values $\{x_i, 1 \le i \le N\}$, which would minimize the average service delay of all streaming media accesses. Namely, we need to maximize

$$\frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i [T_i r_i - T_i b_i - x_i]^+/b_i,$$

subject to the constraint $\sum_{i=1}^{N} x_i \le C, x_i \ge 0$.

2.3 An Optimal Solution for Populating Caches

We derive the optimal solution under static conditions. By static conditions, we mean that the cache content is static (i.e. no replacement). By optimal solution, we mean that caching decisions are made with prior knowledge of request arrival rates. We obtain the optimal solution by solving the above optimization problem.

First, for an object i, if $r_i \le b_i$ (i.e., the bandwidth is higher than the object's bit-rate), then there is no need to cache that object (i.e., $x_i = 0$).

Now we consider all other objects. Let I denote the set of objects whose bit-rate is higher than the bandwidth. The above optimization problem is equivalent to minimizing:

$$\frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i \in I} \lambda_i (T_i r_i - T_i b_i - x_i)/b_i,$$

subject to the constraint $\sum_{i \in I} x_i \le C$, $0 \le x_i \le (r_i - b_i)T_i$. Notice that we restrict x_i to be no larger than $(r_i - b_i)T_i$ since a larger x_i does not yield more delay reduction.

The above minimization is equivalent to maximizing $\sum_{i \in I} \lambda_i x_i / b_i$. This is a fractional Knapsack problem, which has the following optimal solution: the caching algorithm chooses those objects with the highest λ_i/b_i ratios, and caches them up to $(r_i - b_i)T_i$ until the cache is used up.

2.4 Dealing with Unknown Request Rates through Replacement

In practice, caching algorithms have no prior knowledge of request arrival rates. Thus, the optimal solution derived in the previous section (which assumed *a priori* knowledge of these rates) is not practical. To approximate the optimal solution, we propose a cache replacement algorithm. Our cache replacement algorithm estimates the request arrival rate λ_i of each object by recording the number (or frequency) of requests to each object, which we denote by F_i. Our cache replacement algorithm works as follows.

As before, if the bit-rate of an object is lower than the measured bandwidth to the server (i.e. if $r_i \le b_i$), then the object is not cached. Otherwise, we define the *utility* of object i as the ratio F_i/b_i. The cache replacement algorithm always caches those objects with the highest utility value. The size of the cached part of object i is up to $(r_i - b_i)T_i$.

Such a replacement algorithm can be implemented with a priority queue (heap) which uses the utility value as the key. Note, on each access to an object, the object's utility value is increased. Therefore, the replacement algorithm may evict other objects. The processing overhead for heap operations is $O(\log n)$, where n is the number of objects in the cache.

2.5 Dealing with Bandwidth Variability through Over-provisioning

In our exposition so far, we assumed that the bandwidth of a path is a constant. In realistic settings, the bandwidth of an path may change (possibly drastically and unpredictably) over time. Without *a priori* knowledge of bandwidth variability, it is impossible to derive an optimal solution for caching (as was done in the previous sections). Hence, we use a heuristic modification of our caching algorithm.

The basic idea behind our heuristic is to make partial caching decisions based on a more conservative estimation of bandwidth. This would result in caching more than the minimum $T_i(r_i - b_i)$ needed for object i. To understand why we need to do so, we observe that when bandwidth varies significantly, if we only cache $T_i(r_i - b_i)$, then it is very possible that this portion of object i will not be enough to hide the access delay. But, how much of the object would be "enough" to cache? Intuitively, such a determination should depend on the amount of bandwidth variability. If bandwidth does not vary (much), then caching $T_i(r_i - b_i)$ is (close to) optimal. If bandwidth varies, then more conservative caching decisions are warranted—the larger the variations, the larger the portion of the object to be cached.

In the extreme case, the most conservative heuristic would yield a caching algorithm that chooses those objects with the highest λ_i/b_i ratios, and would cache them up to r_iT_i (i.e., it would cache whole objects) until the cache is used up. We use the term *Integral* caching to refer to techniques that restrict cached content to be of complete objects. Thus, Integral caching disallows partial caching. With Integral caching, the cache can be used up quickly since it would accommodate fewer objects. As we show in Section 4, such an approach is only advantageous when bandwidth variability is extremely high.

3 Evaluation Methodology

This section describes the methodology used in our simulations. We first present our analysis of NLANR proxy cache [15] logs as well as results from measurement experiments we conducted on real Internet paths to get realistic bandwidth models. Next, we describe how to generate synthetic streaming media access workloads to drive our simulations. Finally, we describe the performance metrics used to compare various algorithms.

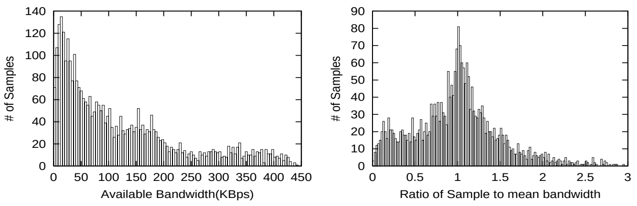
3.1 Network Bandwidth Modeling

For our performance evaluation, we needed to adopt a realistic model of the base bandwidth over various paths, and how such base bandwidth may vary over time. We derived such models using two methods: (1) analysis of proxy cache logs, and (2) measurement of bandwidth over a set of real Internet paths.

We obtained bandwidth statistics by analyzing the NLANR proxy cache logs. We used a nine-day log of site UC (April 12-20, 2001). This site has a popular client (a low level proxy). We observed those missed requests for objects larger than 200 KB. A bandwidth sample is obtained by dividing the size of an object by the connection duration. We used requests for large objects since long duration of HTTP connections results in more accurate measurement of bandwidth.[1] We used the missed requests so that the objects were served by the original servers.

Figure 2(a) shows a histogram of the bandwidth values observed from analysis of the NLANR proxy cache logs. It shows that the bandwidth of various paths varies drastically. This heterogeneity of bandwidth suggests that caching algorithms could benefit from differentiating between objects from origin servers with widely different bandwidth.

We also observed the bandwidth variability for requests made to the same server over different times. To do this, we first computed the average bandwidth of each path. Then we took the ratio of the bandwidth samples to the average value. Figure 2(b) shows the distribution of this sample-to-mean ratio. It indicates that the bandwidth of a single path may vary significantly.



(a) Bandwidth distribution (b) Bandwidth variability

Figure 2. Bandwidth values observed in NLANR cache logs. Here (a) shows the number of samples in each 4KB/s slot; (b) shows sample-to-mean ratio of the same paths.

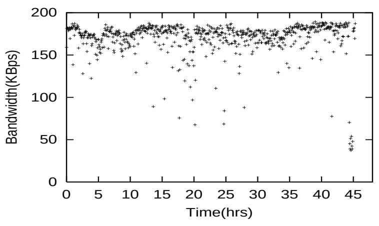
It is important to note that the analysis of NLANR logs cannot give us a realistic model for bandwidth variability. To obtain more realistic bandwidth variability models, we measured the bandwidth of real Internet paths over long periods. We repeatedly downloaded large files from Web servers around the world. Each download takes 5 to 30 seconds. We carefully scheduled the downloads to avoid overlapping them on the client machine (IP address: 128.197.12.3). Figure 3 shows the bandwidth evolution (time series) of three such paths spanning over 30 to 45 hours (starting from 2PM, Oct. 15, 2001). The following observations were made: (1) The magnitude of bandwidth variability depends largely on the paths. For instance, the INRIA server appears to have much lower variability than the other two servers. (2) All paths have much lower variability than those obtained through analysis of the NLANR cache logs. This comparison was done by computing the coefficient-of-variation of the samples in Figure 3, and contrasting it to the coefficient-of-variation obtained from Figure 2.

3.2 Synthetic Workload Generation

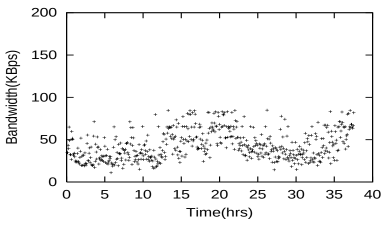
We used the GISMO toolset [13] to generate a synthetic workload. Table 1 lists the characteristics of this workload. The workloads used in our simulations consisted of requests to $N = 5000$ streaming media objects, whose popularity follows a Zipf-like distribution [27]. With Zipf-like distributions, the relative popularity of an object is proportional to $r^{-\alpha}$, where r is the rank of the object's popularity. The probability that the ith ranked object is accessed is $r^{-\alpha}/\sum_{j=1}^{N} j^{-\alpha}$. The default value for α is 0.73; we also present results with a range of values.

Each workload (for a single run) consisted of 100000 requests. Request arrivals were generated using a Poisson process; i.e., the requests arrive independently. The duration (in minutes) of the streaming media objects follow a Lognormal distribution. The average duration of the objects is

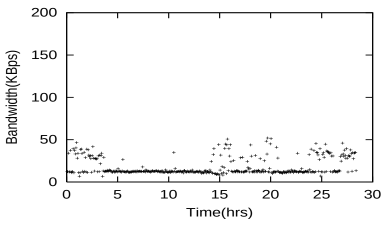
[1]Large file transfers enable TCP to "exit" slow-start phase and settle on a transfer rate that reflects available bandwidth. Our characterization of the correlation between NLANR transfer times and file sizes (not shown) suggests that 200 KB is a good threshold.



(a) INRIA,France (138.96.64.17)



(b) Taiwan (140.114.71.23)



(c) Hong Kong (143.89.40.4)

Figure 3. Bandwidth variation of real Internet paths from Boston University, MA (IP address:128.197.12.3) to several Web servers. One sample was taken every four minutes.

Table 1. Characteristics of the Synthetic Workload

Number of Objects	5,000
Object Popularity	Zipf-like
Number of Requests	100,000
Request Arrival Process	Poisson
Object Size	Lognormal,$\mu = 3.85$,$\sigma = 0.56$
Object Bit-rate	2KB/frame, 24 frames/sec.
Total Storage	790 GB
Bandwidth Distribution	NLANR logs
Bandwidth Variation	NLANR logs and measurement

about 79 K frames, or about 55 minutes since we assume 24 frames per second. The bit-rate of the objects is 48 KB/s. The total unique object size is 790 GB. Although we have changed the base parameters and generated workloads of different characteristics, we found that the relative performance of the different algorithms is fairly similar.

In our simulation experiments, we varied the cache size from 4 GB, about 0.5% of the total unique object size, to 128 GB, about 16.9% of the total unique object size. The bandwidth between the cache and the servers follows the sample distribution from the NLANR logs, c.f. Figure 2. When we study the impact of bandwidth variability, we generate bandwidth instances varying according to the models depicted in Figures 2 and 3.

3.3 Performance Metrics

Different caching algorithms have different objectives. A particular algorithm may optimize one performance metric but sacrifice others. For example, an algorithm (such as LRU and LFU) caches objects based on their access frequency only, not on the network bandwidth. It aims at improving hit ratios and reducing traffic, but not the average service delay or stream quality. Therefore, we need to compare several performance metrics so that we can understand the tradeoffs of different caching algorithms.

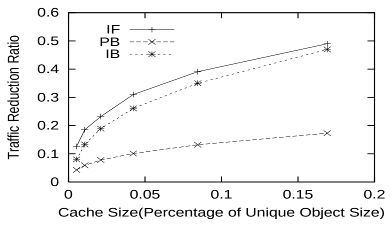
In our experiments, we considered a number of performance metrics, each reflecting a different objective of caching. We discuss these metrics below. (1) Caching algorithms may aim at reducing backbone traffic. We define the *traffic reduction ratio* as the fraction of the total bytes that are served by a cache. (2) When bandwidth (assisted by cache or not) is not enough to support the immediate playout of a stream, the client may choose to wait for service. During this waiting period, the client may prefetch a prefix before continuously playing the stream. In this case, we are interested in the *average service delay*. (3) When bandwidth (assisted by cache or not) is not enough to support a full stream, the client may chose to downgrade the quality of the stream in order to play it out immediately. In this case, we are interested in the *average stream quality*, which we define to be the percentage of the full stream that yields an immediate playout.

4 Simulation Results

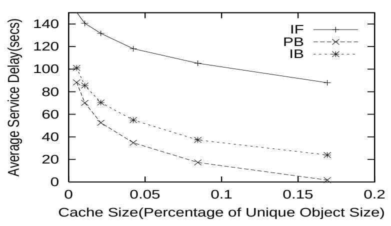
This section presents results from our simulation experiments in which we compared the performance of various caching algorithms and heuristics. Specifically, we study how bandwidth variability affects the performance of caching algorithms. Each result is obtained by averaging ten runs of the simulated system.

4.1 Performance of Replacement Algorithms

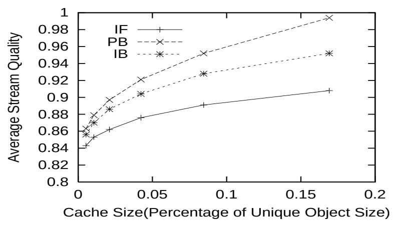
We conducted the first set of simulations to compare the performance of three caching algorithms. The first algorithm caches those objects with the highest request arrival rates and only allows whole objects to be cached. We call this *Integral Frequency-based caching*—or IF caching for short. The second algorithm is the one described in section 2.3. It caches those objects from origin servers without abundant bandwidth for streaming (i.e., preference is given to those with higher λ_i/b_i ratio). It allows partial caching. We call this *Partial Bandwidth-based caching*—or PB caching for short. The third algorithm is the one described in section 2.5. It caches those objects with the highest λ_i/b_i ratio, but does not allow partial caching. We call this *Integral Bandwidth-based caching*—or IB caching for short. All of them estimate object access frequency and network bandwidth progressively, and make eviction decisions when the cache is used up.



(a) Traffic Reduction Ratio



(b) Average Delay



(c) Average Stream Quality

Figure 4. Comparison of IF, PB, and IB cache replacement algorithms under constant bandwidth assumption.

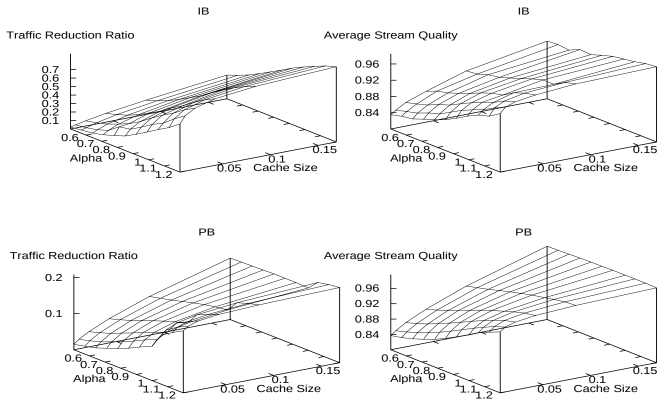
Figure 4 shows the results we obtained from our simulation experiments. For each run of the simulation program, we first warm up the cache using the first half of the workload, and then compute the performance metrics from the second half. For this set of simulations, we assumed that the bandwidth of a path does not vary over time (i.e., no bandwidth variability).

As depicted in Figure 4(a), IF caching achieves the highest backbone traffic reduction while PB caching achieved the least such reduction. This is expected since PB caching does not cache whole objects even if the objects are very hot. Alternately, Figure 4(b-c) shows that PB caching achieves the lowest average service delay and the highest average quality, whereas IF caching achieves the worst results for these metrics. Even when cache size is relatively high, the inferiority of IF caching is still obvious. The reason is that it results in caching hot objects even when there is abundant bandwidth for streaming such objects from origin servers—thus limiting its ability to effectively use the cache.

Figure 4(a) shows that IB caching yields performance metrics that lie in between those of the other two algorithms. IB caching achieves high traffic reduction ratios, and is reasonably close to PB caching in terms of average service delay and stream quality. An additional advantage of IB caching is simplicity: it caches whole objects making it unnecessary to coordinate joint service by origin servers and caches (whereas PB caching requires a client to download a stream from the cache and the origin server in parallel).

4.2 Impact of Temporal Locality of Reference

We conducted a second set of simulations to study the effect of the skewness (i.e., the parameter α) of the Zipf-like distribution governing streaming media object popularity. This skew is a measure of temporal locality of reference. When α increases, temporal locality in the workload is intensified. In simulations, we varied α from 0.5 to 1.2. Figure 5 shows the results of these simulations with respect to the various performance metrics. Only traffic reduction ratio and stream quality are considered here, and only the results of IB caching and PB caching are presented. In general, intensifying temporal locality results in performance gains for both algorithms. Moreover, the relative performance of



(a) Traffic Reduction Ratio (b) Average Stream Quality

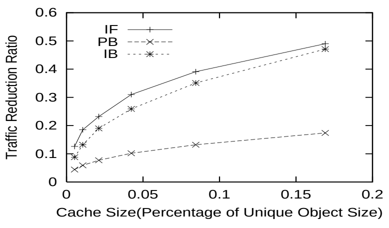
Figure 5. Effect of Zipf-like popularity distribution.

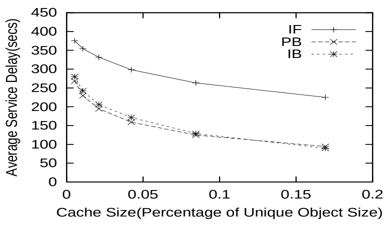
the algorithms does not seem to change: IB caching obtains much higher traffic reduction ratios, whereas PB caching achieves moderately better average stream quality.

4.3 Impact of Bandwidth Variability

We conducted a third set of simulations to study the impact of bandwidth variability on the performance of the three caching algorithms under consideration. In these simulations, we allowed the bandwidth of a path to change over time. We generated such variations as follows: each path has an average bandwidth that follows the distribution in Figure 2(a), but an instance of the bandwidth is obtained by multiplying that bandwidth by a random ratio that follows the distribution in Figure 2(b). The results from this set of simulations are shown in Figure 6.

Comparing Figure 6(a) with Figure 4(a) shows no noticeable difference in traffic reduction ratio for all three algorithms. However, the other two metrics (average delay and average stream quality) exhibit major differences. First, bandwidth variability results in increased service delay and degraded stream quality for all three algorithms. When bandwidth varies drastically over time, partial caching becomes less effective in accelerating access and improving stream quality. High bandwidth variability makes it diffi-


(a) Traffic Reduction Ratio


(b) Average Delay

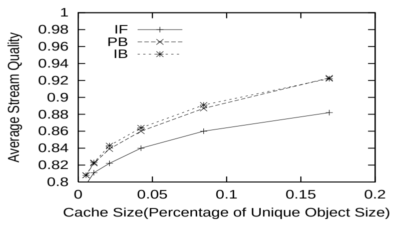
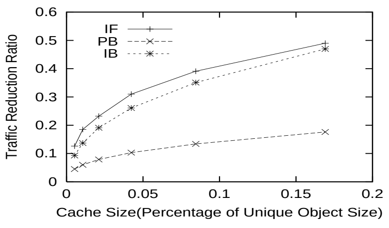
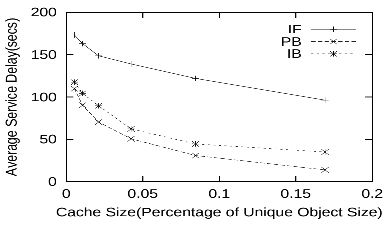

(c) Average Stream Quality

Figure 6. Comparison of caching algorithms under variable bandwidth assumption. The variation is obtained from cache logs.


(a) Traffic Reduction Ratio


(b) Average Delay

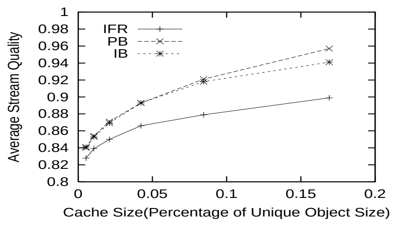

(c) Average Stream Quality

Figure 7. Comparison of caching algorithms under variable bandwidth assumption. The variation is measured from Internet paths.

cult to choose the right objects and the right fraction of each object to cache. Second, IB caching is no worse than PB caching. This is because the optimality of PB caching depends on the constant bandwidth assumption. When bandwidth is insufficient (due to variability), clients see higher delays or lowered quality. On the contrary, IB caching makes conservative decisions, and caches whole objects with the highest λ_i/b_i ratio. It caches those objects with high access frequency and low bandwidth for streaming.

As we discussed before, the bandwidth observed from NLANR cache logs appears to have higher variability than real Internet path measurements we performed. To that end, we conducted a fourth set of simulations using the lower variability modeled by the distribution in Figure 3. The results of these simulations are shown in Figure 7. We observe that, with this more realistic setting, PB caching outperforms the other integral algorithms (IF and IB) in reducing service delay and improving stream quality. These results suggest that the choice of partial versus integral caching should indeed depend on the level of bandwidth variability.

5 Related work

Several studies focused on streaming media workload characterization [1, 3, 4, 7]. In particular, temporal locality is key to the effectiveness of caching techniques. Acharya $et\ al.$ characterized streaming objects [1] and user access patterns [3]. Their work revealed the skewed popularity of objects, and the existence of temporal locality. Almeida $el\ al.$ [4] analyzed workloads from two educational media servers. Chesire $et\ al.$ [7] analyzed a client-based streaming-media workload and found that a small percentage of all requests are responsible for almost half of the total bytes served. They found that requests during periods of peak loads exhibit a high degree of temporal locality.

Several studies proposed caching techniques for streaming media. Wang $et\ al.$ [24] proposed a video staging strategy to reduce network bandwidth requirements. Sen $et\ al.$ [23] proposed that proxies cache prefixes and use work-ahead smoothing. Miao $et\ al.$ [14] proposed selective caching to maximize the robustness of video streams against network congestion. Rejaie $et\ al.$ [19] considered layered-encoded multimedia streams and proposed a proxy caching mechanism to increase the delivered quality of popular streams. Acharya $et\ al.$ [2] proposed the MiddleMan cooperative caching techniques to utilize the aggregate storage of clients. Reisslein $et\ al.$ [17] developed and evaluated a caching strategy which explicitly tracks client request patterns to achieve higher hit ratios. To the best of our knowledge, none of them considered measuring network bandwidth, and none used bandwidth models derived from real Internet measurements in performance evaluation.

6 Conclusion

We proposed a caching architecture and associated cache management algorithms that turn Internet edge caches into accelerators of streaming media delivery. A salient feature of our caching algorithms is that they allow *partial* caching of streaming media objects and *joint* delivery of content from caches and origin servers. The caching algorithms we proposed are both *network-aware* and *stream-aware* in that they optimize cache occupancy decisions based on knowledge of network conditions as well as streaming media object properties. Performance evaluation experiments using realistic streaming media access workloads and a realistic model of Internet path bandwidth demonstrated the effectiveness of caching mechanisms that take into consideration network bandwidth information. Our simulation experiments have shown that bandwidth variability may affect the effectiveness of partial caching in reducing service delay and stream quality. However, simple over-provisioning heuristics work reasonably well, even in the presence of high bandwidth variability.

Our ongoing and future work will proceed on a number of fronts. First, as evident from our findings, accurate measurement of network bandwidth and jitter is key to efficient streaming media delivery techniques. We are in the process of augmenting GISMO [13] with realistic models of Internet path bandwidth and bandwidth variability distributions. Second, we are investigating the possibility of combining our partial caching mechanisms with other streaming content delivery techniques, such as patching techniques at caching proxies. Finally, given the importance of real-time bandwidth measurement techniques, we are considering approaches that integrate active bandwidth measurement techniques [11] into proxy caches. This would allow us to prototype the acceleration architecture proposed in this paper using off-the-shelf caching proxies.

Acknowledgments: The authors have benefitted from discussions with Lisa Amini and Olivier Verscheure, and would like to acknowledge them.

References

[1] S. Acharya and B. Smith. An experiment to characterize videos stored on the Web. In *Proceedings of MMCN*, 1998.

[2] S. Acharya and B. Smith. MiddleMan: A video caching proxy server. In *Proceedings of NOSSDAV*, June 2000.

[3] S. Acharya, B. Smith, and P. Parns. Characterizing user access to videos on the World Wide Web. In *Proceedings of MMCN*, January 2000.

[4] J. Almeida, J. Krueger, D. Eager, and M. Vernon. Analysis of educational media server workloads. In *Proceedings of NOSSDAV*, June 2001.

[5] A. Bestavros, R. Carter, M. Crovella, C. Cunha, A. Heddaya, and S. Mirdad. Application level document caching in the Internet. In *Proceedings of SDNE: The Second International Workshop on Services in Distributed and Networked Environments*, June 1995.

[6] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proceedings of USITS*, December 1997.

[7] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming workload. In *Proceedings of USITS*, March 2001.

[8] M. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of SIGMETRICS*, May 1996.

[9] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[10] M. Glossglauser and J. Bolot. On the relevance of long-range dependence in network traffic. In *Proceedings of SIGCOMM*, 1996.

[11] K. Harfoush, A. Bestavros, and J. Byers. Measuring Bottleneck Bandwidth of Targeted Path Segments. Technical Report BUCS-TR-2001-016, Boston University, Computer Science Department, July 2001.

[12] S. Jin and A. Bestavros. Popularity-aware GreedyDual-size Web proxy caching algorithm. In *Proceedings of ICDCS*, April 2000.

[13] S. Jin and A. Bestavros. GISMO: Generator of Streaming Media Objects and Workloads. *Performance Evaluation Review*, 29(3), 2001.

[14] Z. Miao and A. Ortega. Proxy caching for efficient video services over the Internet. In *Proceedings of PVW*, April 1999.

[15] National Laboratory for Applied Network Research. http://ircache.nlanr.net/.

[16] V. Paxson. Wide-area traffic: The failure of poisson modeling. In *Proceedings of SIGCOMM*, August 1994.

[17] M. Reisslein, F. Hartanto, and K. W. Ross. Interactive video streaming with proxy servers. In *Proceedings of First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN)*, February 2000.

[18] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Proceedings of INFOCOM*, April 1999.

[19] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia proxy caching mechanism for quality adaptive streaming applications in the Internet. In *Proceedings of INFOCOM*, March 2000.

[20] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley. Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. In *Proceedings of SIGMETRICS*, May 1996.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, Jan 1996.

[22] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol(RTSP), April 1998.

[23] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of INFOCOM*, April 1999.

[24] Y. Wang, Z.-L. Zhang, D. H. Du, and D. Su. A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers. In *Proceedings of INFOCOM*, 1998.

[25] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox. Removal policies in network caches for World-Wide Web documents. In *Proceedings of SIGCOMM*, August 1996.

[26] R. Wooster and M. Abrams. Proxy caching that estimates page load delays. In *Proceedings of WWW Conference*, 1997.

[27] G. K. Zipf. *Relative Frequency as a Determinant of Phonetic Change*. Reprinted from Harvard Studies in Classical Philiology, XL, 1929.