Provisioning Content Distribution Networks for Streaming Media•

Jussara M. Almeida[a] Derek L. Eager[b] Michael Ferris[a] Mary K. Vernon[a]

[a]Computer Sciences Department
University of Wisconsin Madison, USA
{jussara,ferris,vernon}@cs.wisc.edu

[b]Department of Computer Science
University of Saskatchewan, Canada
eager@cs.usask.ca

Abstract--**This paper develops simple cost models for provisioning content distribution networks that use the simple and highly scalable bandwidth skimming protocol for streaming. New insight is obtained into (1) how cost-effective proxy servers are in multicast streaming systems, (2) the most effective streaming protocol, and (3) the optimal proxy content, as a function of the system configuration and workload. A key result is that proxy servers are only cost effective if (a) the origin server does not have a multicast capability, or (b) the file request rate is low, and thus multicast is not highly effective, or (c) the cost of a proxy server stream is a very small fraction (i.e., approximately 1/*P*) of the cost of an origin server stream, where *P* is the number of proxy servers and the cost of either type of stream includes both the server and network resource costs. For cases where proxy servers are cost effective, results in the paper provide the optimal proxy content and the most effective streaming protocol, as a function of a wide range of system configuration and workload parameters. In contrast to previous work, full file caching outperforms prefix caching over a significant region of this system design space, due to more efficient multicast streaming protocols as well as a more complete exploration of the practical system configuration space.**

I. INTRODUCTION

A content distribution network (CDN) for data that is distributed via a wide area communication network generally consists of the origin server that contains the content and a set of "proxy" servers that each store a subset of the content closer to key client populations. Provisioning the CDN involves determining the network interface bandwidth, number of disks, and processing capacities that should be purchased or leased at the servers, as well as deciding which content should be stored at each proxy.

For conventional content, including stored media files that are delivered using simple unicast streams, each delivery from content stored at a proxy saves essentially an equal amount of work by the origin server, as well as the cost of transmitting the data from the origin to the proxy. Thus, in an abstract sense, a proxy should store the media that will be accessed most frequently, and should be provisioned with sufficient storage and bandwidth to off-load the origin server (and remote network) to the desired degree.

Provisioning a CDN for stored media content is significantly more complex if the system delivers the content using one of the recently proposed *scalable* streaming protocols (e.g., [3,6,9,11,13]), for at least two reasons. First, these protocols use (IP or application-level) multicast to deliver the content. Thus, for a given frequently accessed file, the origin server will perform less work per client request than a proxy must perform (because on average more clients will share each origin stream). Second, the protocols deliver the earlier portions of the stream more frequently (and to fewer clients on average per multicast) than the later portions of the stream. Thus, it may be more cost effective to store a prefix rather than a suffix of a given media file at the proxy. Together, these considerations imply that the subset of the content that should be stored at a given proxy depends not only on the content access rates, but also on the corresponding protocol-specific delivery rate for each portion of the file as well as the actual relative cost of a proxy (multicast) stream compared to an origin server (multicast) stream. Simple quantitative examples that illustrate these points are provided in Section III.A.

Previous work [4,5] has developed a relatively simple model for estimating delivery cost, and the proxy content that minimizes delivery cost, for variations in the scalable streaming protocol and for specified values of the content access rates and proxy bandwidth and storage capacity. Eager et al. developed the model for the case that the "partitioned dynamic skyscraper" (PDS) protocol is used by both the origin and the proxy servers. The PDS protocol only allows a pre-specified fraction of the file, or the full file, to be stored at the proxy. Ramesh et al. [14] modified the model server bandwidth calculations for several variations on the "patching" protocol [3,11] including "selective catching" [9], and applied the cost model to determine the optimal fraction of the file that should be stored at the proxy when the proxy has no bandwidth or storage constraints. Wang et al. [15] use a similar cost model but with the server bandwidth costs modified for the case that the origin uses unicast delivery to the proxy, the proxy caches an arbitrary fraction of the file, and the proxy uses one of several other variations on the patching protocol to deliver the content to the client. They develop a simpler solution method and apply the model for the case of high client arrival rate, *limited* proxy disk storage, but *unlimited* proxy disk bandwidth. Each of these previous studies has assumed that client bandwidth is at least two times the streaming rate. A key conclusion of each of these previous studies is that if client request rate is high and/or proxy storage is limited, storing file prefixes rather than (fewer) full files significantly reduces delivery cost.

This paper revises the cost model to provision CDNs that use more efficient scalable protocols, and obtains new insight into the cost-effectiveness of proxy servers, the most effective

•This work was partially supported by the National Science Foundation under Grants CCR-9975044, CCR-9972372, and ANI-0117810 and by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264. Jussara Almeida is partially supported by CNPq/Brazil and by a Lawrence H. Landweber NCR Graduate Fellowship in Distributed Systems.

To appear in *Proc. 21st Annual Joint Conf. IEEE Computer and Communications Societies (Infocom 2002)*, New York, NY, June 2002.

protocol, and the optimal proxy content, over a significantly wider region of the system design space than studied in previous work.

The CDN cost model is developed for the following protocols:

1. **BWSkim(*b*)**: The origin server as well as the proxy servers use the simple scalable *bandwidth skimming* protocol [6], with client bandwidth equal to *b*.[1]
2. **BWSkim/U(*b*)**: The origin server uses simple unicast streams to the proxy servers, but the proxy servers use bandwidth skimming to the clients.
3. **BWSkim[/U]+Batch(*b*)**: For each of the above systems, if the proxy stores a prefix, clients use extra bandwidth, or reduce their bandwidth used for listening to proxy streams, in order to batch together to share a new origin stream for the suffix.

We compare these protocols with respect to content delivery cost, and determine the optimal proxy content as a function of parameters that define a large system design space, including: (1) number of proxies varying from 1 – 100, (2) unlimited as well as realistically constrained proxy storage capacity and disk bandwidth, and (3) client arrival rate at each proxy varying from 1 – 10,000 requests, on average, per average playback duration. The optimal content is defined as the content that minimizes delivery cost, or a simpler rule of thumb that achieves near-minimum cost.

The key results are summarized in Section VI. One key result is that storing content at proxy servers is only cost effective if (a) the origin server does not have a multicast capability, or (b) the file request rate is low, and thus multicast is not highly effective, or (c) the cost of a proxy server stream is a very small fraction (i.e., approximately 1/*P*) of the cost of an origin server stream, where *P* is the number of proxy servers and the cost of either type of stream includes both the server and network bandwidth costs. Furthermore, over a large fraction of the system design space where proxy servers are cost effective, the optimal proxy content consists primarily of *full files* rather than file prefixes. Although seemingly in conflict, this result is shown to be consistent with previous results. The results also show that the regions of the design space where prefix caching significantly reduces delivery cost coincide with the regions where listening to an extra origin stream is cost-effective. In all other regions, the simpler BWSkim[/U] policies are cost-effective.

The remainder of this paper is organized as follows. Section II provides background information on the bandwidth skimming protocol and the delivery cost model. Section III defines the new CDN protocols more precisely and provides the cost model formulas for each protocol. Section IV and V provide results for unconstrained and constrained proxy storage and disk bandwidth, respectively. Section VI concludes the paper.

[1] The bandwidth skimming protocol is significantly more efficient than patching or PDS at high client arrival rates, and significantly more efficient than selective catching at low arrival rates, as will be shown in Section II.C.

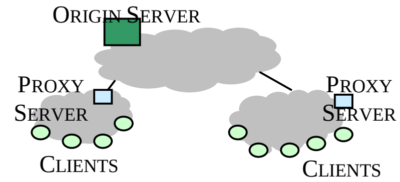


Figure 1: Content Distribution Network (CDN)

II. BACKGROUND

A. CDN Configuration

The CDNs evaluated in this paper, illustrated in Figure 1, consist of an origin server (or simply the *origin*) and *P* proxy servers. A client requests the content from a nearby proxy. Depending on the delivery protocol (defined in Sections III.C – III.E), if the requested data is not stored at the proxy, the origin either streams the data to the proxy which then delivers it to the client(s), or the origin delivers the stream directly to the clients. In the former, the proxy sends a request to the origin and delivers the response stream to the client. In the latter, the proxy might forward a request to the origin and then inform the client which origin multicast streams to listen to.

We assume that the proxy server uses the scalable (multicast) bandwidth skimming protocol to deliver the content to its clients. In most protocols, we assume that the origin server also employs the bandwidth skimming delivery protocol (using either IP multicast, which may become more widely available as new protocols supporting single-source multicast [10] are implemented, or using application-level multicast). We also consider the case that the origin can not multicast to the proxies, and thus the origin uses simple unicast streaming.

B. Bandwidth Skimming Protocol

The reader is referred to previous work [6,8] for a complete description of the bandwidth skimming protocol. The basic idea is that each client request initiates a new *multicast* stream. In one variant of the protocol, the client listens to the new stream as well as the closest (target) stream that is still active. When the new stream has delivered all of the data that the client missed in the target stream, the new stream is terminated, and all clients listening to the target stream (are now "merged" and) start listening to the next closest target stream that is still active. If a target stream terminates before the later stream is ready to terminate, the clients listening to the later stream simply start listening to the next closest target stream. Unlike previous scalable streaming protocols, if client bandwidth is less than twice the streaming rate, clients can listen to an increasing *fraction* of the target stream (and a corresponding decreasing fraction of their main stream) in order to merge with the target clients [6]. Thus, bandwidth skimming can be used to deliver *high quality* videos that require a significant fraction of the bandwidth to the client. Because clients are merged hierarchically, the server bandwidth required to provide immediate service to each client request grows only logarithmically (with a small constant factor) in the client request rate.

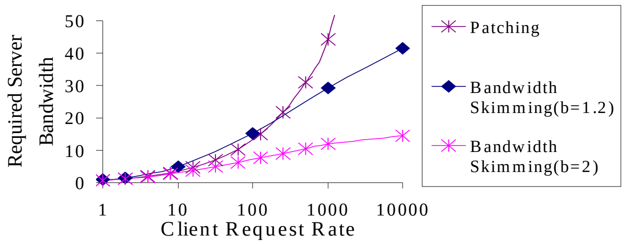


Figure 2: Server Bandwidth for Scalable Protocols

C. Required Server Bandwidth

Results in [7] show that the following formula is a good estimate of required server bandwidth[2], measured in units of the streaming rate, when b=2, bandwidth skimming is used to deliver the file, client arrivals are Poisson (as measured in [1]), each client requests the entire media file, and the entire file is stored at the server:

$$B(N) = \eta \ln (1 + N/\eta), \tag{1}$$

where η=1.62 for b=2, and N is the file request rate. We have also verified that the above formula with η=5.53 (*i.e.*, equation (6) in [7] with n=1.2), is an accurate estimate for bandwidth skimming if b=1.2 (i.e., results are within 12%).

Figure 2 plots the required server bandwidth for bandwidth skimming with two values of b, and for the patching protocol (which requires b=2) under the same assumptions [7]. These results show that bandwidth skimming is significantly more efficient than patching when client request rate is high.

An estimate of the required server bandwidth for the above assumptions but for deterministic inter-arrival times, obtained from a bound provided in [7], is given for b=2 by:

$$B(N) = 1.5\log_2 (1 + N).^3 \tag{2}$$

D. Previous CDN Cost Model

The CDN delivery cost, applied in [4, 14, 15], for a file that has a fraction f and client rate N/P at each proxy, is:

$$C(f, N,P,b) = B_{origin}(f, N,P,b) + P\times\beta \times B_{proxy}(f, N,P,b) \tag{3}$$

where B_{origin} and B_{proxy} depend on f, N, P, b, and the delivery protocol in use, and $\beta \le 1$ is the average cost per proxy stream relative to an origin server stream. Stream cost is based on estimates of the average resources used by the streams from the respective server. Origin stream cost will depend on whether the origin delivers content only to the proxy or directly to the clients. Note that multiplying both sides of equation (3) by the estimated origin stream cost yields the actual estimated CDN delivery cost. Equation (3) can also be modified in the obvious way for the case that each proxy has a different client rate and stores a different fraction of the file [5].

An optimization problem can be formulated in which the objective is to minimize, over the fraction of each file that is stored at each proxy, the sum of the delivery cost for each file (given by equation (3)), subject to bounds on the total proxy bandwidth available to deliver all files and on the proxy storage capacity. Solving the optimization problem, as in [4,5,15], yields both the minimum delivery cost and the proxy content that minimizes cost. Previous work used this approach for the CDN protocols reviewed in Section I.

[2] The "required server bandwidth" for a file is defined as the average server bandwidth used by the specified protocol to serve each client immediately.

[3] Using simulation, we have validated that this expression is a good estimate of required server bandwidth for the bandwidth skimming policy under the stated assumptions.

Table 1: Client Workload and Cost Model Parameters

Symbol	Definition
λ	Average client request arrival rate for a file
T	Media file duration (in minutes)
N	Client arrival rate, in arrivals per T ($N=\lambda T$)
b	Client bandwidth
f	Fraction of the file stored at the proxy
B_{proxy}	Server bandwidth required to deliver the content stored at the proxy, measured in units of the file streaming rate
B_{origin}	Server bandwidth required to deliver content from the origin, measured in units of the file streaming rate
β	Ratio of cost of one proxy server stream to cost of one origin server stream
P	Number of proxy servers
C	Total delivery cost for the file

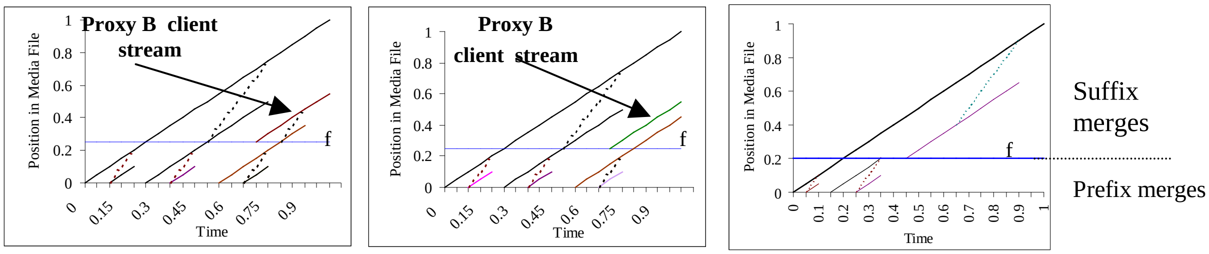
III. New CDN Protocols & Cost Models

Section III.A provides examples that illustrate the need for a simple delivery cost model, such as the model in Section II.D, to provision scalable streaming media CDNs. Section III.B discusses the assumptions that will be made in Sections III.C-E, which define the BWSkim(b), BWSkim/U(b), and BWSkim[/U]+Batch(b) protocols, and derive formulas for B_{origin} and B_{proxy} for the delivery cost model for each protocol.

A. Motivating Examples

To understand the need for a delivery cost model to provision a CDN when scalable streaming protocols are used, consider a CDN with ten proxy servers (i.e., P=10) and request rate per proxy (N/P) equal to 100 for a given media file. If the file is not stored at any of the proxies, the request rate to the origin is equal to 1000. Figure 2 shows that the required origin server bandwidth is approximately twelve (concurrent) streams for client request rate 1000 and client receive bandwidth b=2. On the other hand, if each proxy stores the entire file, Figure 2 shows that the required bandwidth at each proxy server (with request rate 100) is approximately seven streams. Whether ten proxy servers each transmitting seven streams is more cost effective than one origin server transmitting twelve streams depends on the average cost ratio of the origin server stream and the proxy server stream, which in turn depends on both the server and network resource costs.

Note that similar tradeoffs occur for other scalable streaming protocols. For example, periodic broadcast protocols (e.g., [12,13,15]) require a fixed server bandwidth (e.g., on the



a) BWSkim(2) b) BWSkim/U(2) c) BWSkim+Batch(3)

Figure 3: CDN Delivery Protocols (Streams Requested by Proxy A Clients)

order of 10 streams) for the origin or for each proxy that stores the file, independent of the client arrival rate.

The capability to store file *prefixes* at the proxy creates further complexity in provisioning the proxy storage. Consider a CDN with a single proxy with client rate equal to 1000 for a given file. As before, twelve origin streams are required to deliver the file if it is not stored at the proxy. If instead the proxy stores the first 10% of the file, the client request rate for the prefix, normalized to the length of the prefix, is 100. Thus, from Figure 2, the proxy transmits approximately seven concurrent streams to deliver the prefix to all clients. If the origin server transmits through the proxy, or informs the proxy of its streams for merging, the total required bandwidth is still twelve, so the origin transmits five streams to deliver the file suffix. Thus, a small amount of proxy storage (e.g., 10% of the file) offloads a large fraction of the origin server bandwidth (i.e., seven out of twelve of the origin server streams). Similar examples can be given for other scalable delivery protocols, since each delivers a prefix more frequently than later portions of a popular file. These factors are part of the basis for the previous conclusions [4,5,14,15] that prefix caching at the proxies minimizes delivery cost. However, modern disks have a relatively large amount of storage, so realistic client request rates and proxy bandwidth constraints for the assumed proxy storage capacity, together with using the more efficient bandwidth skimming protocols may lead to new conclusions.

Which data should be stored at the proxies depends in complex ways on the client workload, the number of proxies, the storage capacity and bandwidth of the proxies, and the relative costs of proxy and origin server streams. Simple models that capture main features of the system behavior are needed to obtain initial insights into the most effective delivery protocols as well as the proxy storage strategies that minimize delivery cost as a function of those parameters. Such models, which can be refined to include more details of the system behavior, are discussed next.

B. Model Assumptions

In this paper, we use the model of delivery cost, as a function of client arrival rate and fraction stored at the proxy, given in equation (3). The workload and cost model parameters are shown in Table 1. We assume that the proxy client workloads are statistically homogeneous (which implies that all proxies store the same fraction f of each file). Equation (3) also assumes that the average cost of an origin stream as compared with a proxy stream is expressed as a constant, β, which will be varied in the experiments in Sections IV and V to determine its impact on the optimal proxy cache content.

In Sections C-E below, we define, and derive formulas for the CDN server bandwidths, for three ways of employing the bandwidth skimming protocol: BWSkim, BWSkim/U, and BWSkim[/U]+Batch. The derived bandwidths were validated against simulation over a wide range of client arrival rates (i.e., N/P = 1–10,000), file fraction stored at the proxies (f = 0.01–0.99), and for 1–10 proxy servers. The formulas are within 15% of the simulation values for all of these parameter ranges.

In deriving the bandwidths, it is assumed that client arrivals are Poisson (as measured in [1]) and that each client retrieves the entire requested file. For the systems in Sections C and E, it is further assumed that the origin can stream the content directly to the client, by using, for example, the mechanism mentioned in Section II.A.

The bandwidth calculations are easily modified for various types of heterogeneities of the client workloads (as was done in [5]) or for various models of interactive client requests.

C. BWSkim(b) Protocols

In the BWSkim(b) protocol, the proxies can store an arbitrary fraction of a given file. Both the proxy and the origin use the simple closest target bandwidth skimming protocol (for client bandwidth b) to deliver their streams, as illustrated in Figure 3a for a file with fraction f stored at the proxy. A new client stream from the proxy is merged hierarchically with other streams from the same proxy. If the full stream or a suffix is delivered by the origin, origin streams are also merged. The origin may stream the media data directly to the clients rather than through the proxies, which only impacts the value of β.

With a small increase in implementation complexity (i.e., messages from the origin to the proxies informing them of each new origin stream for any file that has a prefix stored at the proxies), a proxy can direct the clients of the oldest of the active prefix streams for a file to listen to and begin merging with a closest active target origin stream. To simplify the calculation of required origin server bandwidth for the BWSkim protocol, we assume that the proxy implementation does not have this optimization. However, we have verified that the calculated origin server bandwidth is very close to the bandwidth for the optimized protocol (typically within 10%) over the parameter space of our validation experiments. Part of the explanation is that at moderate to high client arrival rate, new clients continually merge with the oldest proxy stream, which delays the time when it can start a successful merge with the closest target origin stream. At lower arrival rates, there is less overall opportunity for merging, and hence, variations in the merging protocol do not have a large impact.

The BWSkim proxy uses bandwidth skimming to deliver a file of length fT which has normalized client request rate equal to $\lambda/P \times fT = fN/P$. Letting $N_{proxy} = N/P$,

$$B_{proxy}\ (f, N,P,b) = \eta \ln (1 + f N_{proxy}/\eta), \tag{4}$$

where η depends on b, as given in Section II.C.

To derive B_{origin}, we first estimate the average arrival rate of (suffix) requests at the origin. The request rate from each proxy is equal to the rate at which proxy streams reach the end of the prefix without being merged into some earlier stream, which can be derived as:

$$\frac{\lambda_{origin}}{P} = \lim_{\delta T \to 0} \frac{B_{proxy}(f+\delta, N, P, b) - B_{proxy}(f, N, P, b)}{\delta T} = \frac{dB_{proxy}(f, N, P, b)}{d(fT)} = \eta \frac{\lambda / P}{fT\lambda / P + \eta}.$$

Given the normalized request rate to the origin server suffix, we can compute the required origin server bandwidth using either equation (1) or (2). Comparison against simulation revealed that the Poisson arrival assumption is accurate over the parameter space of our validations. Thus, noting that the normalized suffix request rate, $N_{origin} = \lambda_{origin} \times (1-f)T$,

$$B_{origin}(f,N,P,b) = \eta \ln(1+N_{origin}/\eta), \tag{5}$$

where:

$$N_{origin} = \frac{\eta(1-f)N}{fN/P + \eta}. \tag{6}$$

D. BWSkim/U(b) Protocols

In the BWSkim/U protocol, the origin uses simple unicast streams to deliver content that is not stored at the proxies. In this case, the origin streams the content to the proxy and the proxy multicasts it to the clients, so that clients listening to streams from the proxy can be directed to merge using the bandwidth skimming protocol. Note that each unicast stream will terminate when the clients listening to the multicast from the proxy merge with a target stream. Thus, the system operates as in Figure 3b, which is similar to Figure 3a except that each origin suffix stream (unicast to the proxy) can only merge with other suffix streams requested by the same proxy.

The proxy bandwidth needed to deliver the content stored at the proxy is computed as in the BWSkim protocol, using equation (4). The origin bandwidth needed to deliver the rest of the content requested by *one* of the proxy client populations can be computed by setting P=1 in equations (5) and (6). The total origin bandwidth is then P times this bandwidth:

$$B_{origin}(f,N,P,b) = P\ \eta \ln[1 + (1-f)\ N / (f\,N+\eta)\ ]. \tag{7}$$

We let the cost of the origin server stream include the cost of streaming through the proxy server (which requires network bandwidth but not disk I/O bandwidth at the proxy). Thus, for two CDNs that differ only in whether the BWSkim or the BWSkim/U protocol is used, the value of β will be smaller in the latter case. In Section V we quantify the network i/o bandwidth needed per proxy for origin server content in BWSkim/U CDNs.

E. BWSkim[/U]+Batch(b) Protocols

The BWSkim+Batch protocol operates the same as BWSkim(2), with the same CDN server bandwidths as derived in Section III.C, for each file that is fully stored or is not stored at the proxy. For a file that has a prefix stored at the proxies, BWSkim+Batch requires client bandwidth, b>2, and as illustrated in Figure 3c, each client that requests the file uses one unit of bandwidth to listen to the first origin suffix stream that starts after the client request. The proxy delivers the prefix to the clients using BWSkim(b-1). During the first fT of the *suffix* stream, no merging of suffix streams is performed (because client bandwidth is often entirely used for listening to the one suffix stream and the prefix stream(s)). If the suffix is longer than fT, the remainder of the suffix streams are merged by the origin using BWSkim(2).

The required proxy server bandwidth is computed using equation (4) but using η for b–1. The batching of clients in each origin suffix stream decreases the rate of suffix requests compared to the BWSkim protocol, yielding an average time between two suffix requests equal to fT+1/N. Note that this interval is approximately deterministic (unless f, T or N is small). Hence, the arrival rate at the origin server is given by

$$\lambda_{origin}(f) = \frac{1}{fT + 1/N}. \tag{8}$$

If $f \geq 0.25$, there is no merging at the origin server, and

$$f \geq 0.25: \quad B_{origin}(f, N,P,b) = \lambda_{origin} \times (1-f)T. \tag{9}$$

This is because merging only starts after 2fT (the prefix of length fT and fT of the suffix) and the minimum inter-arrival time at the origin is fT. Thus, if f=0.25, merging could start at T/2, but the closest target stream must be at or beyond position 3T/4, so merging cannot occur before the target stream ends.

If f < 0.25, merging occurs in the last 1–2f of the file. Since no merging of suffix streams occurs before this last segment, the "requests" for this last segment arrive approximately deterministically at rate given by equation (8). In total, the origin bandwidth needed to deliver that last portion of the file can be derived using an approach similar to the derivation of equation (2) in [7], as follows:

$$N^*_{origin}(f) = \frac{1}{fT + 1/N}(1-2f)T,$$

$$k(f) = \lfloor \log_2(N^*_{origin}(f) + 2 \rfloor - 1,$$

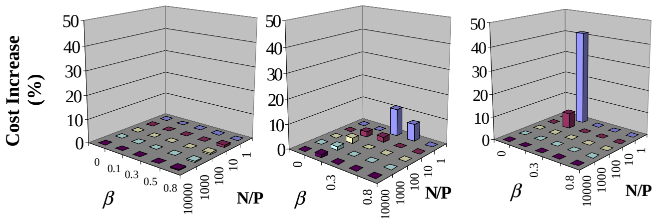
$$B_{origin,last-seg}(f) = \frac{3}{2}k(f) + \frac{N^*_{origin} + 2}{2^{k(f)}} - 2.$$

Thus,

$$f<0.25: \quad B_{origin}(f,N,P,b) = \lambda_{origin} \times fT + B_{origin,last-seg}(f). \tag{10}$$

The origin bandwidth calculations above can be modified for the BWSkim/U+Batch(b) protocol, as described in Section III.D for the BWSkim(b) protocol.

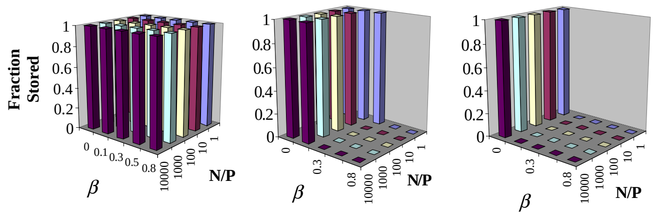
For the BWSkim+Batch protocol, we consider systems with client bandwidth (b) equal to 3 and 2.2, which implies



***P*=1** **_P_=10** **_P_=100**

a) Unicast Origin **b) Multicast Origin**

Figure 4: Cost Increase of BWSkim[/U](2) versus BWSkim[/U]+Batch(3)
(unconstrained proxy servers, one file)



a) _P_=1 **b) _P_=10** **c) _P_=100**

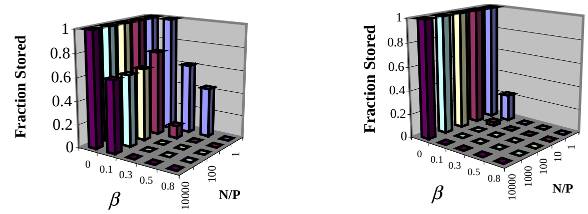
Figure 5: Optimal Proxy Content for BWSkim(2)
(unconstrained proxy servers, one file)

client bandwidth of 2 or 1.2, respectively, for merging proxy streams. The client bandwidth for receiving the extra origin suffix stream when the proxy stores a prefix might be provided by a system closer to the client than the proxy, which would buffer the data and send it to the client when needed, in the case that the "last mile" bandwidth to the client is limited. In fact, in the BWSkim or BWSkim+Batch protocols, the "client" might be a system very near the real client that implements all of the necessary buffering and then feeds a simple single stream to the actual client. In any case, the implementation of BWSkim+Batch is more complex than the BWSkim implementation, so it is preferable to use BWSkim unless the cost savings for BWSkim+Batch are significant. One interesting question to be explored is how BWSkim+Batch(*b*), with *b*=3 and *b*=2.2, compare to ordinary BWSkim(2), and in turn how much the cost increases for BWSkim(1.2), which may be important for streaming high quality content.

IV. Results for Unconstrained Proxy Servers

In this section the models are used to provision CDNs with a delivery protocol and storage policy in the case that the proxy can be configured with enough bandwidth and storage for the fraction of each file that minimizes the file delivery cost. For each policy and given values of *N/P*, *P*, β, and *b*, we numerically solve for the fraction *f* of the file stored at the proxy that minimizes the delivery cost given in equation (3), using the CDN server bandwidth costs derived in Section III.

Figure 4 provides the percent increase in cost for BWSkim(2) as compared with BWSkim+Batch(3) for *N/P* varying from 1 – 10,000, β varying from 0 – 0.8, and *P* equal to 1, 10, and 100. Note that the results for *P* = 1 also provide the cost increase for BWSkim/U(2) vs. BWSkim+Batch/U(3). The results show that the simple BWSkim(2) protocol has delivery cost within 5% of the BWSkim+Batch(3) protocol over nearly the entire design space.



a) _P_=10 **b) _P_=100**

Figure 6: Optimal Proxy Content for BWSkim+Batch(3)
(unconstrained proxy servers, one file)

Figure 5 shows, for the wide design space explored, that the optimal value of *f* for the BWSkim(2) protocol is 1 or 0, depending primarily on *P* and on β, and to a lesser degree on *N/P*. Note that the results in Figures 5b,c show that a file is only stored at the proxy servers if it is much less expensive to deliver it from the proxy (i.e., β is small), in which case the assumption the proxies have sufficient disk resources for the content may be reasonable.

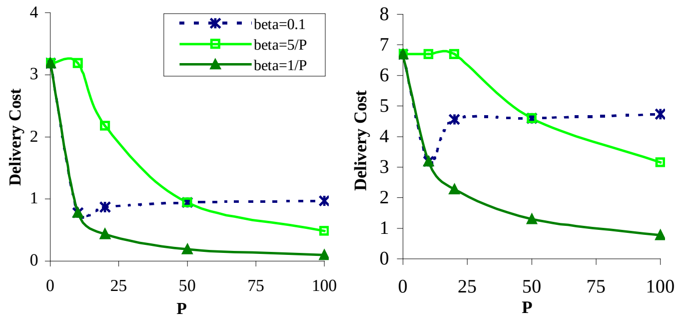
The optimal all-or-nothing storage results for CDNs that use BWSkim(2) contrast sharply with results in previous work on unconstrained proxy storage strategies [4,14] for CDNs that use the scalable PDS protocol, selective catching protocol, or variations on the patching protocol. For these less efficient streaming protocols, prefix caching and batching for origin suffix streams greatly reduce delivery cost. In the BWSkim(2) system, batching clients for origin streams is not more effective than the very efficient bandwidth skimming merges that occur in the streams delivered by the proxies or the origin.

According to Figures 4-6 and additional results omitted due to space constraints:

- For a multicast origin, large *P*, low *N/P*, and intermediate values of β (*i.e.*, *P* >10, *N/P*≈1, and $\beta = 0.1$ as in Figure 4c), BWSkim+Batch(3) outperforms BWSkim(2), and the optimal proxy content is a file prefix, as shown in Figure 6b.
- In all other cases, delivery cost is nearly minimized by using the simple BWSkim protocol and a simple all-or-nothing storage strategy.
- A bandwidth skimming CDN stores a given file (or prefix) at the proxies *only if* (1) $\beta = 0$, (2) $\beta \leq 0.1$ and $P \leq 10$, (3) $\beta \leq 0.3$, $P > 10$ and *N/P* is small, or (4) the origin uses unicast (or $P = 1$).

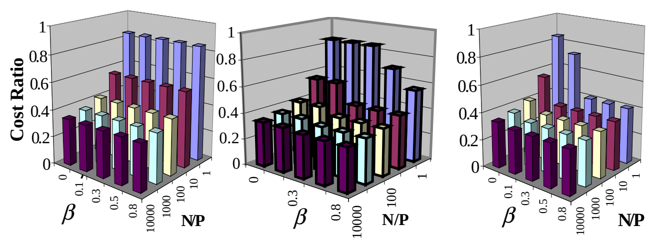
Figure 7 shows the delivery cost as a function of the number of proxies that store the file, illustrating again that when the origin server uses multicast, storing the content at multiple proxies is only cost effective if *N/P* is small or β is a very small fraction (i.e., approximately 1/*P*). For small *P*, *N/P*, and/or β, the limited opportunity for sharing of origin streams or the low relative cost of proxy streams leads to storing content at the proxies. For $P > 1$, as either β or *N/P* increases, it is more cost-effective to share the file delivery from the origin among all clients, and storing content at the proxies is not cost effective.[4] Thus, the experiments in Section V for proxies with storage

[4] Note however, that even if the content is not stored at the proxy, a very small prefix might be stored to reduce start-up latency or for smoothing VBR streams.



a) N=10 b) N=100

Figure 7: Delivery Cost vs P for Optimal Multicast Protocol
(unconstrained proxy servers, one file)



a) P=1 b) P=10 c) P=100

Figure 8: Cost Ratio of BWSkim(2) to BWSkim(1.2)
(unconstrained proxy servers, one file)

and bandwidth constraints will consider only $\beta \leq 0.1$ if the origin server is multicast enabled.

BWSkim+Batch is less cost effective if b=2.2 than if b=3 and thus for unconstrained proxy server storage and bandwidth, BWSkim(2) is preferred over BWSkim+Batch(2.2).

Figure 8 shows the cost ratio for BWSkim(2) to BWSkim(1.2). Both systems cache the same content at nearly every point in the design space, and thus the differences in cost are due to the decrease in merging efficiency as b decreases. As N/P increases for any given values of P and β, the cost ratio appears to converge to 0.3, which implies that delivery cost increases by at most about a factor of three if client bandwidth is limited to 1.2 streams (i.e., for higher quality content).

V. Cache Content for Constrained Proxies

In this section, the cost models are used to provision CDNs with a delivery protocol and proxy storage policy, assuming proxy storage and bandwidth is bounded. The goal is to obtain insight, for a given system and set of files for which storing the data at the proxy is cost-effective (e.g., origin server uses unicast streaming or $\beta \leq 0.1$, as identified in Section IV, Figures 5 - 6), into which (fractions of) files should be stored at the proxy and which delivery protocol has higher performance.

We consider a set of n origin server media files of equal duration T that have total client request rate M (arrivals per time T) and a skewed Zipf-like distribution[5] of access frequency. The proxy storage, P_s, is expressed as a fraction of the total amount of data at the origin. The proxy bandwidth, P_b, is expressed as the ratio of the available bandwidth to the total bandwidth that would be needed if the proxy fully stored all files, calculated by summing the required server bandwidth for delivering each file[6] (equation (1)) over all files.

Table 2 : Additional Parameters for Constrained Proxies

Symbol	Definition
n	Number of media files, each of duration T
M	Total client arrival rate, in arrivals per T (all files)
P_s	Proxy storage capacity as a fraction of n×file size
P_b	Ratio of available proxy bandwidth to bandwidth needed if the proxy fully stores all n files

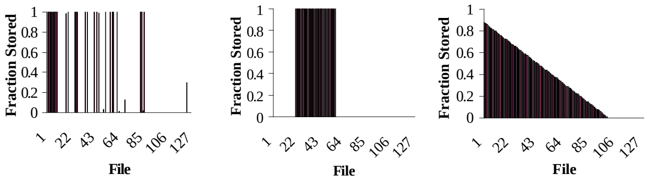
A CDN that uses a given delivery protocol is provisioned for a given set of system and workload configuration parameters (defined in Tables 1 and 2), by solving the constrained optimization outlined in Section II.D. This optimization involves minimizing the sum of the delivery cost for each file over all possible fractions of each file that might be stored at the proxies. To make the model tractable, we optimize over discrete values of f, ranging from 0 to 1 in steps of 0.01. The optimization problem is solved with constraints on proxy storage (P_s) and bandwidth (P_b), typically in under 1 minute, using the CPLEX solver within GAMS [2].

Simple storage strategies that achieve nearly the minimum delivery cost are of greater interest than complex strategies that achieve the true minimum. Thus, if the solution to the optimization model is a set of files and/or prefixes that must be enumerated rather than specified by simple ranges of file access rank and prefix size, we further constrain the model to obtain a simpler solution that has nearly the same delivery cost as the optimal solution, as discussed in Section V.A.

To obtain realistic relative values of P_b and P_s, for given values of M/P, T, and n, we compute P_b and P_s for an integral number of current generation Ultrastar 72zX disk and the MPEG-2 streaming rate of 4 Mb/s. A single such disk can store 44 hours of MPEG-2 content, and has i/o bandwidth (ideally) for 42 concurrent streams. Shorter duration files will need a smaller number of disks for a given value of P_s; thus, for a given M/P, n, and P_s, the value of P_b will be smaller for smaller file duration T. Similarly, a larger number of files (n) implies a larger number of disks for given values of T and P_s; thus, for a given M/P, T, and P_s, the value of P_b will be larger for larger n. Thus, the impact of varying n or T is to change the relative size of P_s and P_b, as would occur when changing the disk technology. By choosing n as either 128 or 1024, T as either 30 or 120 minutes, varying the number of disks so that P_s varies from 0.08 to 0.68, and varying M/P from 10 to 10,000, a wide range of absolute and relative values of P_s and P_b are included in the experiments below.

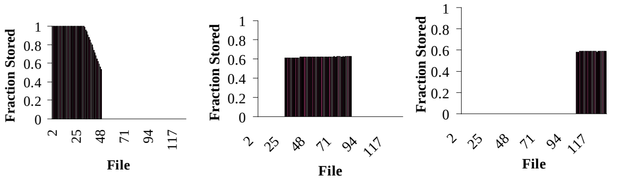
Sections V.A and V.B provide the results for the CDNs in which the origin server uses unicast or multicast, respectively, to deliver content. Based on the results for unconstrained proxy servers in Section IV, Section V.B only considers small values of β (i.e., $\beta \leq 0.3$). In most cases the optimal proxy content uses all of the proxy storage capacity or all of the proxy bandwidth, or both; these results are marked with the keywords "cap", "bw", or "both", respectively, in the figures.

[5] The access frequency for file i is equal to C/i, where C is a constant that ensures that the sum of the file access frequencies is equal to 1.
[6] Note that the average request rate at a given proxy server for a given file is given by the product of M/P and the file access frequency.



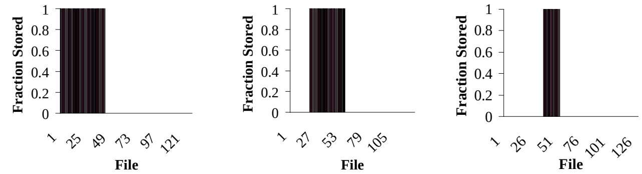
a) True Optimal P_b=0.32 **(bw)** **b) Near Optimal** P_b=0.32 **(bw)** **c) P_b=1** (unrealistic P_b)

Figure 9: Example Proxy Content for BWSkim/U(2) (P_s=34% , *M/P*=1000, *n*=128, *T*=2h)

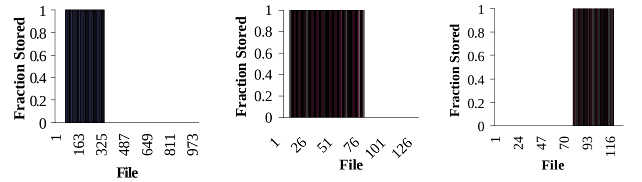


a) P_b=1.39 *M/P*=100 **(cap)** **b) P_b=0.32** *M/P*=1000 **(bw)** **c) P_b=0.13** *M/P*=10000 **(bw)**

Figure 10 : Optimal Proxy Content for BWSkim+Batch/U(3) (P_s=34%, *n*=128, *T*=2h)

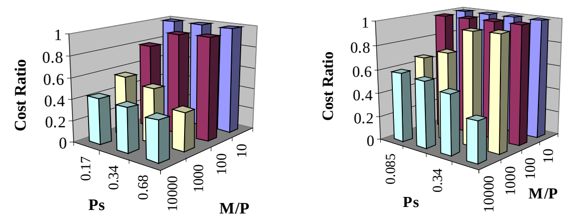


a) $P_b \geq 1$, P_s=34% (cap), *T*=2h *M/P*=100, *n*=128 *M/P*=1000, *n*=1024 **b) P_b=0.32, P_s=34% (bw),** *T*=2h *M/P*=1000, *n*=128 *M/P*=10000, *n*=1024 **c) P_b=0.13, P_s =34% (bw),** *T*=2h *M/P*=10000, *n*=128



d) P_b=0.35, P_s=34% (bw), *T*=30min *M/P*=1000, *n*=1024 **e) P_b=0.63, P_s=68% (bw),** *T*=2h *M/P*=1000, *n*=128 **f) P_b=0.26, P_s=68% (bw),** *T*=2h *M/P*=10000, *n*=128

Figure 11: Near-Optimal Proxy Content for BWSkim/U(2)



a) *n*=128, *T*=2h **b) *n*=1024, *T*=2h**

Figure 12 : Ratio of delivery cost for BWSkim/U(2) to cost of BWSkim/U(1.2)

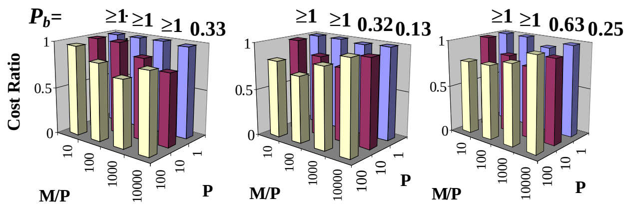
A. Origin Server uses Unicast Streams

When the origin uses unicast streaming, the proxy content that minimizes delivery cost, i.e., the content that maximally off-loads the origin, is the same for all *P* and β.

Over the wide range of system configuration parameters defined above, results illustrated in Figures 9-12 and further results omitted to conserve space show the following:

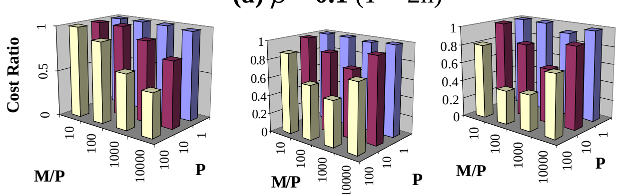
- The optimal proxy server content for BWSkim/U(*b*) consists primarily of full files, but for many configurations the optimal content is a chaotic set of files and prefixes, as illustrated in Figure 9a for a configuration with P_s = 0.34 and P_b = 0.32. In all such cases, the optimal solution when the proxy servers are constrained to store any contiguous (with respect to their access frequency rank) set of full files, as illustrated in Figure 9b, yielded delivery cost within 0.05% of the minimum. These near-optimal solutions are presented in the remainder of this section.
- The delivery cost savings for BWSkim+Batch/U(3) compared to BWSkim/U(2) are very small (i.e., no greater than 8%). Figure 10 shows that BWSkim+Batch/U(3) stores primarily file prefixes at the proxies (which increases the opportunities for batching clients for origin streams), but the benefit of batching is again not substantial enough to outperform the efficient bandwidth skimming merges in the simpler BWSkim policy.
- CDNs that use BWSkim+Batch/U(2.2) do not cache many prefixes and do not improve on BWSkim/U(2).
- The trends in Figures 11a–c, e–f were observed for all of the CDN configurations studied. In particular, for BWSkim/U(2) and values of P_b greater than 1, the proxy fully stores as many of the most popular files as its storage permits. For smaller P_b, the proxy (fully) stores fewer and less popular files. Figure 10, and Figure 11d compared with 11a and 11b, further illustrate that for a given value of P_s the optimal proxy content appears to depend heavily on the value of P_b.
- Figure 9c shows the optimal content (not constrained to be a contiguous set of full files) for the same BWSkim/U(2) system configuration as in Figure 9a, except that the proxy bandwidth (P_b) is arbitrarily increased by a factor of three. In this hypothetical configuration, the optimal proxy content contains only prefixes, which agrees with results in [15] for a similar CDN configuration that uses the patching protocol. However, this configuration is not feasible for expected disk technology trends. Furthermore, for feasible configurations with the same P_s and P_b as in Figure 9c (e.g., Figure 11a), prefix storage is not optimal.
- Figure 12 shows that, as in the case of unconstrained proxy storage and bandwidth, if client bandwidth is limited (as may be the case when streaming high quality videos), using BWSkim/U(1.2) increases the delivery cost by up to a factor of three. The trends in the optimal proxy content for BWSkim(1.2), as a function of P_b and P_s, are similar to the BWSkim(2) CDN.

In BWSkim/U CDNs, the origin uses unicast streaming to the proxies, which then multicast the content to their clients. Over all of the system configurations examined in this work in which the proxy is not severely under-provisioned (in terms of storage capacity and disk bandwidth) for the client load, the network i/o bandwidth needed per proxy for origin content, which is included in the relative cost of the origin streams, is less than the per-proxy disk i/o bandwidth.



P_s=34%, n=1024 P_s=34%, n=128 P_s=68%, n=128

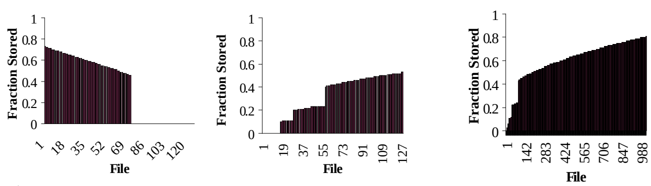
(a) β = 0.1 (T = 2h)



P_s=34%, n=1024 P_s=34%, n=128 P_s=68%, n=128

(b) β = 0 (T = 2h)

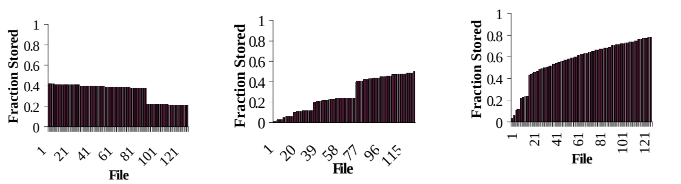
Figure 13: Cost ratio of BWSkim+Batch(3) to BWSkim(2)



a) P_b ≥1, P_s=34% P=10, (cap) M/P=100, n=128

b) P_b =0.32, P_s=34% P=10, (bw) M/P=1000, n=128

c) P_b =0.66, P_s=68% P=10, (bw) M/P=10000, n=1024

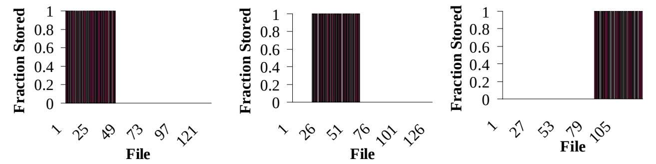


d) P_b ≥1, P_s=34% P=100, (cap) M/P=100, n=128

e) P_b =0.32, P_s=34% P=100, (bw) M/P=1000, n=128

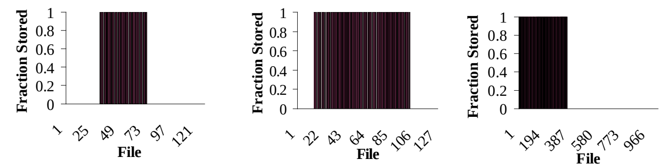
f) P_b =0.63, P_s=68% P=100, (bw) M/P=1000, n=128

Figure 14: Optimal Content for BWSkim+Batch(3) (β=0,T=2h)



a) P=10, β=0.1 b) P=10, β=0.3 c) P=100, β=0.1

P_b ≥1, P_s=34%, (cap), M/P=100, n=128, T=2h



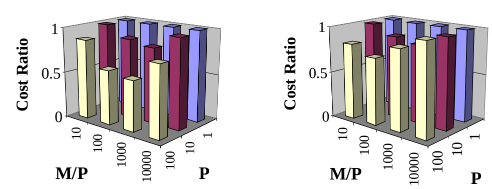
d) P_b=0.32, P_s=34% (both), n=128

e) P_b =0.64, P_s=68% (both), n=128

f) P_b ≥1, P_s=34% (cap), n=1024

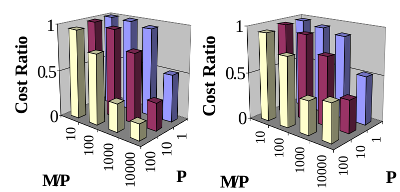
P=10, β=0.1, M/P=1000, T=2h

Figure 15: Optimal Cache Content for BWSkim(2)



a) β=0 b) β=0.1

Figure 16: Ratio of the delivery cost for BWSkim +Batch(2.2) to BWSkim(2) (P_s=34%, n=128, T=2h)



a) β=0 b) β=0.1

Figure 17: Cost Ratio for Best Policy when b≥2 to BWSkim(1.2) (P_s=34%, n=1024, T=2h)

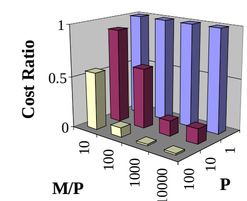


Figure 18: Cost Ratio for Origin Multicast to Origin Unicast (β=0, n=128, T=2h)

B. Origin Server Uses Multicast Streams

The results above show that the near-optimal BWSkim/U protocol and proxy content for CDNs with unicast streaming from the origin does not include prefixes for any realistic configurations considered. A key question for CDNs in which the origin delivers streams using multicast is whether (and in which regions of the design space) prefix caching and batching clients from different regions together for suffix streams, reduces the cost significantly. The results in Figures 13–17, and further results omitted due to space constraints show that:

- BWSkim+Batch(3) significantly reduces delivery cost (e.g., by more than 20%) *primarily* when $\beta \leq 1/P$, P > 10, $M/P \geq$ 100, and P_b > 0.1, as shown in Figure 13. If M/P is small, multicast is not highly effective and the optimal proxy content for both policies is similar (i.e., long prefixes or full file caching of the most popular files, as illustrated in Figures 14a and 15a). For very large M/P, P_b is small (as shown at the top of Figure 13), and both policies store less popular and fewer files.
- Figures 14 and 15 show how the optimal proxy content for BWSkim+Batch(3) and BWSkim(2) varies, respectively, over design regions in which each is the best policy. These figures illustrate the variations in the content as P_b, β, or P increases. If more disks are added to the proxies, BWSkim proxies store more files (e.g., Figures 15d,e) while BWSkim+Batch proxies store longer prefixes (e.g., Figures 14e,f or 14b,c). Less popular content is stored at the proxies as β increases (Figures 15a,b), P increases (Figures 14a,d or 15a,c), or P_b decreases (Figures 14a,b or 15f,e).
- As illustrated in Figure 16, BWSkim+Batch(2.2) reduces delivery cost over BWSkim(2) for the same configurations as BWSkim+Batch(3). Furthermore, perhaps surprisingly, the reduction in cost for BWSkim+Batch(b) compared to

BWSkim(2) is nearly as large for b=2.2 as for b=3, in spite of the reduced merging of prefix streams when b=2.2. In nearly all configurations, the optimal proxy content for BWSkim+Batch(b) is the same for b=2.2 and b=3.

- Figure 17 illustrates that, compared to the best policy for $b \geq 2$, using BWSkim(1.2) (e.g., for higher quality content) increases the delivery cost significantly (up to a factor of nine) for $P > 10$ and $M/P > 100$ or $M/P > 1000$.

Finally, we compute how much cost savings are achieved when the origin server uses multicast, as compared to unicast. Results in Figure 18 are representative of savings achieved in other regions of the design space in which the proxy is provisioned to handle a reasonable fraction of the origin load. The results show that, except when the proxy client request rate for all origin content (M/P) is small or P=1, the cost benefit of origin multicast is significant, with savings up to 98% for the configurations examined.

VI. CONCLUSIONS

This paper developed simple cost models for provisioning CDNs that use highly scalable streaming protocols. The main findings, over a large CDN configuration space, are:

- Storing content at the proxy servers is only cost effective if (a) the origin is not multicast enabled, or (b) the file request rate is low (thus multicast is not highly effective), or (c) the cost of a proxy stream is a very small fraction of the cost of an origin stream (*i.e.*, $\beta \leq 1/P$).
- For the configurations where proxies are cost effective, BWSkim+Batch[/U](3) outperforms BWSkim[/U](2) (*i.e.*, by more than 20%) *only if* the origin server uses multicast and (a) $\beta = 0.1$, proxy disk resources are unlimited, P > 10, and file request rate, $N/P \approx 1$, or (b) $\beta \leq 1/P$, proxy disk space and bandwidth are limited (in spite of the low relative cost of a proxy stream), $P > 10$, $P_b > 0.1$, and total client request rate per proxy (M/P) is greater than 100.
- The (near) optimal proxy content for the BWSkim(b) policy includes essentially no prefix storage, whereas the optimal cache content for BWSkim+Batch(b) includes primarily file prefixes when it significantly outperforms the simpler BWSkim policy.
- BWSkim+Batch(2.2) reduces cost compared to BWSkim(2) for the same CDN configurations as BWSkim+Batch(3), by nearly the same amount and with the same optimal content.
- BWSkim(1.2) increases delivery cost compared to the best policy with $b \geq 2$, by a up to a factor of three if proxy disk resources are sufficient, or up to a factor of nine when proxy disk storage and bandwidth are limited.
- The impact of varying the system configuration parameters on the optimal proxy content is as follows:
  1. If $\beta = 0$ and the arrival rate (M/P) is small, the optimal proxy content contains as many of the most popular full files as can be stored.
  2. As M/P increases, for constrained proxy disk resources, the proxies store less popular data.
  3. If the origin-proxy path is multicast-enabled, as either β or P increases, it becomes more cost-effective to stream more popular files from the origin. If the origin-proxy path is not multicast-enabled, neither β nor P has impact on the optimal cache content.
- If the proxies are not severely under-provisioned, unicast of origin data through the proxy can be performed with proxy network bandwidth less than twice the proxy disk bandwidth.
- Multicast instead of unicast delivery by the origin greatly reduces delivery cost unless M/P is small or P=1.

The models developed in this paper use a very simple approximation for the relative cost of proxy and origin server streams. Future research includes developing models for provisioning scalable streaming CDNs that include more precise network bandwidth costs.

REFERENCES

[1] J. M. Almeida, J. Krueger, D. L. Eager, M. K. Vernon, "Analysis of Educational Media Server Workloads", *Proc. NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[2] A. Brooke, D. Kendrick and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, CA, 1988.

[3] S. Carter and D. Long, "Improving Video-on-demand Server Efficiency Through Stream Tapping", *Proc. Int'l. Conf. on Computer Communications and Networks*, 1997.

[4] D. L. Eager, M. C. Ferris and M. K. Vernon, "Optimized Regional Caching for On-Demand Data Delivery", *Proc. 1999 Multimedia Computing and Networking*, San Jose, CA, Jan. 1999.

[5] D. L. Eager, M. C. Ferris and M. K. Vernon, "Optimized Caching in Systems with Heterogeneous Client Populations", *Performance Evaluation, Special Issue on Internet Performance Modeling*, Sep. 2000, pp. 163-185.

[6] D. L. Eager, M. K. Vernon and J. Zahorjan, "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand", *Proc.* 2000 *Multimedia Computing and Networking*, San Jose, CA, Jan. 2000.

[7] D. L. Eager, M. K. Vernon and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery", *IEEE Trans. On Knowledge and Data Engineering*, Special Section of invited papers from MIS'99, Vol. 13, No. 5, Sep./Oct. 2001, pp. 742-757.

[8] D. L. Eager, M. K. Vernon and J. Zahorjan, "Optimal and Efficient Merging Schedules for Video-on-Demand Servers", *Proc. ACM Multimedia'99*, Orlando, FL, Nov. 1999.

[9] L. Gao, J. Kurose and D. Towsley, "Catching and Selective Catching: Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams", *Proc. ACM Multimedia'99*, Orlando, FL, Nov. 1999.

[10] H. W. Holbrook and D. R. Cheriton, "IP Multicast Channels: Express Support for Large-Scale Single-Source Applications", *Proc. ACM SIGCOMM '99 Conf.*, Cambridge, MA, Aug./Sept. 1999.

[11] K. Hua, Y. Cai and S. Sheu, "Patching: A Multicast Technique for True Video-on-demand Services", *Proc. ACM Multimedia'98*, Bristol, U.K., Sep. 1998.

[12] K. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video on Demand Systems", *Proc. ACM SIGCOMM'97 Conf.*, Cannes, France, Sep. 1997.

[13] J-F. Pâris, S. W. Carter and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video on Demand", *Proc. 1999 Multimedia Computing and Networking*, San Jose, CA, Jan. 1999.

[14] S. Ramesh, I. Rhee and K. Guo, "Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service", *Proc. IEEE INFOCOM'01*, Anchorage, AL, April 2001.

[15] B. Wang, S. Sen, M. Adler and D. Towsley, "Proxy-based Distribution of Streaming Video over Unicast/Multicast Connections", TR 01-05, Dept. of Computer Science, Univ. of Massachusetts.