Cache Satellite Distribution Systems: Modeling and Analysis

Aner Armon Hanoch Levy
School of Computer Science
Tel Aviv University
Tel Aviv, Israel

***Abstract*—Web caches have become an integral component contributing to the improvement of the performance observed by Web clients. Content Distribution Networks (CDN) and Cache Satellite Distribution Systems (CSDS) have emerged as technologies for feeding the caches with the information clients are expected to request, ahead of time. In a Cache Satellite Distribution System (CSDS), the proxies participating in the CSDS periodically report to a central station about the requests they are receiving from their clients. The central station processes this information and selects a collection of Web documents (or "Web pages"), which it then "pushes" via a satellite broadcast to all, or some, of the participating proxies, hoping most of them will request most documents in the near future. The result is that upon such request, the documents will reside in the local cache, and will not need to be fetch.**

In this paper[*] we aim at addressing the issues of how to operate the CSDS, how to design it, and how to estimate its effect. Questions of interest are 1) What classes of Web documents should be transmitted by the central station, and how they are characterized, and 2) What is the benefit of adding a particular proxy into a CSDS. We offer a model of this system that accounts for the request streams addressed to the proxies and which captures the intricate interaction between the proxy caches. Unlike models that are based only on the access frequency of the various documents, this model captures both their frequency and their locality of reference. We provide an analysis of this system that is based on the stochastic properties of the traffic streams that can be derived from HTTP logs. The model and analysis can serve as a basis for the design and efficient operation of the system.

Keywords—Web Cache; Content Distribution Network; Satellite Network; Performance analysis; Performance evaluation.

Methods keywords—System design; Simulations; Stochastic processes/Queueing theory.

I. INTRODUCTION

Web caches have become an integral component contributing to the improvement of the performance observed by Web clients. Content Distribution Networks (CDN) and Cache Distribution Satellite Systems (CSDS) have emerged as technologies for feeding the caches with the information clients are expected to request, ahead of time. A typical Cache Satellite Distribution System (CSDS), of which a schematic drawing is given in Fig. 1, consists of a set *P* of proxy caches, and one central station. The proxies participating in the CSDS periodically report to the central station about the requests they are receiving from their local clients. The central station processes this information and uses it to predict what documents will be desired by other proxy caches in the system in the near future. The central station then selects a collection of Web documents, which it retrieves, usually from the terrestrial network, and then "pushes" these documents via a satellite broadcast link to all, or some, of the participating proxies. The result is that upon such request, the documents will reside in the local cache, and will not need to be fetched using the local terrestrial network.

The advantage of broadcasting a document over CSDS is that once the document is requested at any participating proxy, the document is available in all caches. Thus, the user delay is reduced and the bandwidth cost is saved (assuming that the cost of broadcasting a document to *K* destinations is cheaper than to retrieve it on the terrestrial network *K* times). Nonetheless, such benefits do depend on whether the document will indeed be needed at the receiving proxy. In the case that the document is *not needed*, no benefit is gained. In fact, some damage may be caused, since the unwanted document "contaminates" the cache by pushing all current documents residing in the cache, and potentially causing another document (that may be needed by the proxy) to be removed from the cache. Thus, broadcasting a document over CSDS is not always beneficial.

In a similar manner, the benefit for a proxy from participating in a CSDS may vary, depending on the mutual properties of the participating proxies. To demonstrate this, consider two proxies, *A* and *B*, which consider sharing a CSDS. If the proxies have interest in similar documents then it is likely that they will mutually benefit from a CSDS. For example, such mutual benefit is expected from two proxies of commercial ISP's in the same country. On the other hand, consider proxies *A* and *B* located in two different countries whose audience is interested mainly in their native language documents. In this case the documents requested by the proxies are highly disjoint, and the use of a CSDS will only cause the two proxies to "contaminate " each other.

The aim of this paper is to devise an analytic tool that can be used in the design decisions involved in the operations of a

[*] This research was supported in part by MAGNET, Chief Scientist Office, Ministry of Trade and Commerce, Israel.

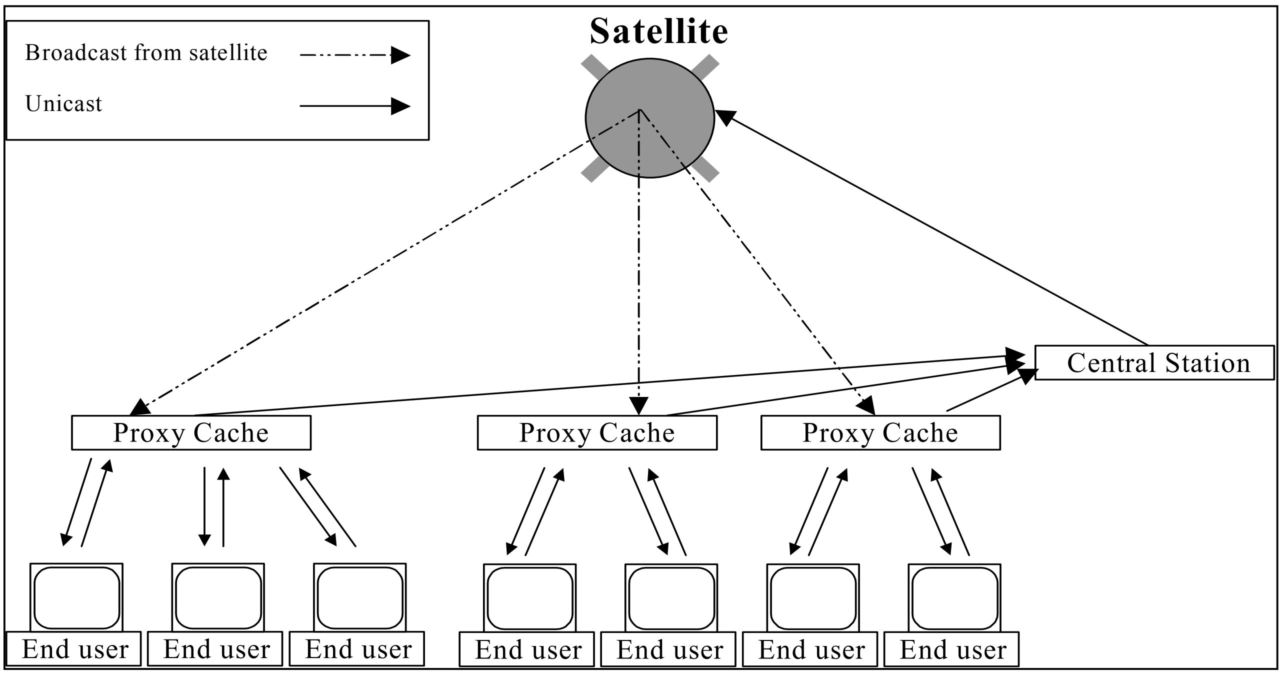


Figure 1: A schematic diagram of the system

CSDS. These include issues such as the selection of classes of Web documents for broadcast in the system and the selection of proxies to participate in the system. To this end, one must take into consideration the properties of the various proxies and the properties of the traffic streams they are exposed to and to use this data to analyze the system. Two properties are inherent to streams directed at Web caches (as to other caches as well). These are the relative request frequencies of the various documents (some documents are accessed more frequently and some are less) and the locality of reference (a document that was requested recently is more likely to be requested again). Unfortunately, accounting for both properties is difficult and most common models do not account for both.

We propose a stochastic model that accounts for both properties of the request streams directed to a proxy. The statistics of these properties can be derived from the HTTP log of the proxy. The model uses these properties to predict the hit ratio, which is the fraction of the requests made to the cache and which result with the requested document found at the cache. Thus, using this analysis one can evaluate the system performance under various conditions and operations.

The model accounts for the intricate interaction of the different streams at the different caches and makes use of two document request distributions: The stack depth distribution, and the request count distribution. The stack depth distribution has been used successfully to model the behavior of cache systems (see, e.g., [5]), and in [8] and [9] it was used to model and analyze the interaction of database caches. The request count distribution helps us to estimate the potential gain of sharing the first miss of every document, among all participating proxies.

Our analysis yields a set of recursive equations whose computation complexity is $O((K\alpha)^2)$, where K is the cache size, and $\alpha \le 1$ is a modeling parameter that can be chosen to be quite small as to make the computation very efficient. The results of the analysis provide expressions for the improvement in the cache hit-ratio in each of the participating proxies, as function of the document classes (or streams) broadcast to participating proxies. Thus, the analysis can be used for selecting the more effective streams to be broadcast, as well as for assessing the value of joining a proxy into a CSDS.

The analysis is supported by numerical results in which we examine the predictions of the analysis by comparing it to simulation. We observe good agreement between the simulation and the analysis especially in predicting the relative effect of different streams or different proxies on the performance. As such, the analysis fits well in assisting in system design and in operational rule design.

The structure of the rest of this paper is as follows: In Section II we provide the model and the analysis approach. In Section III we provide the analysis of the system. In Section IV we discuss how the results can be used for devising operation rules and designing the system. Numerical results are provided in Section V. Finally, Section VI summarizes the paper and its results.

A. Related work

Cache modeling and analysis received attention in the past in the context of operating systems and databases. For example, see Coffman and Denning [5]. In the context of databases, Levy and Morris [8][9] proposed a model and analysis for evaluating the interaction of various traffic streams in a cache system. More recently, cache modeling on the World Wide Web received attention as well. For example, Breslau et al. [3] tested the sufficiency of using Zipf distribution to describe behavior of web proxies. They provided a comparison to experimental observations, and proposed some cache replacement algorithms. Some attention was also given to properties of different request types on the web, e.g. Almeida et al. [1], that provided classification and interpretation of specific proxy logs, based on geographical characterization.

The work on the modeling and analysis of Cache Satellite Distribution Systems has been quite limited. Rodriguez and Biersack [10] provided an analysis of the performance of Cache Satellite Distribution Systems, but without accounting for modeling aspects of the cache capacity, the inter-relation between the document requests and their inter-reference, and the effects of the different streams on the cache performance. Hu et al. [7], provided an analytic model based on Poisson distributed request rate, and a suggestion of client filtering policy in a cache satellite distribution system based on web servers visited in previous days. Chang [4] formalized the Cache Satellite Distribution problem as an optimization problem, based on the assumption that documents are distributed according to the Zipf distribution; that paper does not account for the inter-reference of document requests as well.

Several references focused on the problem of the selection of documents to be broadcast, based on very detailed information dynamically sent to the central station from the proxies regarding their needs of specifically identified documents. These include Cohen et al. [6] and Askoy et al. [2]. These may require a significant amount of data sent from the proxies to the central server, and they do not deal with the system design issues.

II. General Model and Analysis Approach

A. System modeling and analysis approach

A Cache Satellite Distribution System can be modeled as follows: Each cache is subject to a stream of local document requests, originated by the end users of the proxy. Upon receiving a local request, the proxy will examine whether the requested document is in the cache, in which case it will return the requested document to the user. If the requested document is not in the cache, the proxy will retrieve it from the terrestrial network, and will announce the central station that the document was requested and was not found. The central station then will decide (typically by identifying the document as a part of some stream, defined herein) whether to retrieve the document from the terrestrial network and to broadcast it to all the participating proxies via the satellite link. Upon receiving a document from the satellite link, a proxy may select to either save it in its cache or to discard it.

Remark 1: The notion that not all participating proxies must accept all broadcast documents and put them in their cache, can actually split the set of recipients of the satellite broadcast to several subsets, by common proxy interests. We will model this behavior by defining subsets of proxies, each sharing interest in similar document classes, i.e. only a portion of each document class, originating at each subset, is being broadcast. Most of our analysis focuses on a single subset, but is constructed in a way that can handle multiple proxy subsets.

Our aim in the analysis is to examine how the broadcast of a document or a class of documents (e.g. all the documents of xyz.com) will affect the performance of the individual caches. This analysis is to be carried as function of the parameters of the document class, as to assist us in the decision of which documents (or classes of documents) should be broadcast and which not. We assume that the system designer is equipped with the HTTP logs of the proxies (these are commonly available), and that the statistical data available in the logs can be used by the analysis to predict the relative benefit (or damage) of broadcasting a class. Once this benefit is computed, it will be used to decide which classes of documents to broadcast, and which proxies to join to the system.

Two models that were used to analyze cache systems are the Inter-reference model (IRM) and the Stack Reference Model (SRM). Their advantage is in capturing both the locality of reference of the requests, and their relative frequencies, which are both important properties for cache modeling. In contrast, other models, such as the common used Zipf model (see, e.g. [3]), capture only one of these properties. SRM has been used successfully in [8] and [9] to model the interaction between various caches. More specifically, it was used to examine the performance resulting from merging several disjoint streams into one cache. Due to the interaction between the caches in CSDS, it is appealing to attempt modeling CSDS by the SRM model. That approach however, seems to be hard to apply since the interaction between the streams in CSDS is much more complicated than in the problem addressed in [8] and [9]. This is true since this problem includes interaction of streams that are not mutually exclusive, and it also includes cross-cache effects.

To overcome the complexity of the problem, we decompose the problem and its analysis to two parts, in each of which using suitable modeling tools to focus on capturing the major factors. The approach is as follows: Consider a document R that is broadcast by the central station to cache-proxy C_p, which stores it in the cache. The storing of R in C_p affects the performance of C_p in two aspects: a) It affects the hit ratio of future requests to R. This effect is non-negative (that is the hit ratio of these requests may only go up due to the storage of R). b) It affects the hit ratio of future requests to documents other than R. This effect is non-positive since the presence of R in C_p may push the other documents out of the cache and decrease their hit-ratio.

We will focus on the *change in performance* of C_p due to the broadcast, that is, in how the hit ratio changes due to the broadcast operation. As was previously hinted, we decompose our analysis to two cases:

1. In the first case we deal with the situation at which there is *at least one request* to R by C_p. This is either the request that triggered the broadcast of R, or a later request. Let E_B denote the broadcast event, and let E_p denote the event of the first request to R by C_p.

At this case the *change in performance* due to the broadcast of *R* (and its storage by C_p) is two-fold:

a. A non-negative effect on the hit ratio of requests to *R* (these are the requests to *R* happening after E_p).

b. A non-positive effect on requests to documents other than *R*. A more precise examination shows that this effect is limited to the period between the events E_B and E_p. This limitation applies, since after the epoch at which E_p occurs, *R* is present in C_p regardless of the broadcast operation E_B, and thus the *change in performance*, due to E_B, past the epoch of E_p, is zero.

2. In the second case we deal with the situation at which there is *no request* to *R* by C_p. In this case, the broadcast *does not affect* the hit ratio for requests to *R*. However, the broadcast *does affect* (non-positively) the hit ratio of documents other than *R*, since *R* now "contaminates" C_p.

In our analysis, we will neglect the effects on performance of Case 1.b above, due to its limited scope (which makes it negligible compared to the other effects). Our analysis will therefore focus on the above Case 1.a (section III.A below) and Case 2 (section III.B below). The reason the effect of Case 1.b is so minor, is because we assume that the probability of *R* to be accessed locally is much higher than the probability the document it replaced in the cache will be requested. The reason for this assumption to hold, is because of the inherent property of every stream (that a model such as SRM captures so naturally), that a page that was just requested has a much higher probability to be requested again, than a page that was not requested for a long time (so it is thrown out of the cache). The assumption of negligibility of Case 1.b tends to be less accurate, of course, if the cache of C_p is very small compared to its local stack depth distribution, or the rate of R's class is much smaller than the rate of the class of the page that was thrown out.

B. Document Request Modeling and Model Notation

We consider a set *P* of proxy caches $C_1 .. C_{|P|}$. Considering the entire incoming requests set for all proxies, we assume that we can classify this request set as a set *S* of disjoint streams, whereas each stream $s \in S$ represents all requests for a class of documents, i.e. a collection of documents that share some commonality. For example, a stream may represent all of the requests to any document of xyz.com, or alternatively, only to the home page of abc.com.

We further assume that each stream *s* is associated with a random variable N_s denoting the request count, which is the number of requests that are made for a specific document $R \in s$, from the *whole* set *P*, until document *R* expires. Such expiration may represent document refresh, document removal from its hosting Web server or expiration of the document at all proxy caches. In all cases the implication is that after the document is requested N_s times it may never be requested again or will be considered new and will have to be read again by at least one of the proxy caches. Let the *request count distribution*, which is the distribution of N_s, be denoted as $d_s^{cnt}(n) = \Pr[N_s = n]$. We further assume that the association of a request with a specific proxy $C_p \in P$ is given by a Bernoulli process, that is the probability that an arbitrary request of stream *s* originates at proxy C_p is given by $\lambda_{s,p}$, where $\sum_{p \in P} \lambda_{s,p} = 1$.

Since we want to construct our model in a way that can handle multiple proxy groups (see Remark 1 above), we will assume only a subset of proxies $P' \subseteq P$ needs to participate in the CSDS. Let $\Lambda_{s,P'} = \sum_{p \in P'} \lambda_{s,p}$ denote the probability that a request of stream *s* originates at the subset *P'*.

We assume that the association of a request with a specific stream $s \in S$, is also given by a Bernoulli process. For the clarity of presentation, we can define the combined ratio $\gamma_{s,p}$, which is the probability that an arbitrary request originates at stream *s and* at proxy C_p. We can further widen the definition to stand for any subset S^* of streams, and/or for any subset P^* of proxies, yielding

$$\gamma_{s,P^*} = \sum_{p \in P^*} \gamma_{s,p} , \tag{1}$$

$$\gamma_{S^*,p} = \sum_{s \in S^*} \gamma_{s,p} , \tag{2}$$

and

$$\gamma_{S^*,P^*} = \sum_{p \in P^*} \sum_{s \in S^*} \gamma_{s,p} . \tag{3}$$

Remark 2: Note that we will never use any γ as a stand alone value, but only as ratios of γ's – so that γ can be represented in any units – requests per time unit, requests per log length, etc. As an example, we can give an alternative definition of $\lambda_{s,p}$, as $\lambda_{s,p} = \frac{\gamma_{s,p}}{\gamma_{s,P}}$, and we can immediately see that the previous summation $\sum_{p \in P} \lambda_{s,p} = \sum_{p \in P} \frac{\gamma_{s,p}}{\gamma_{s,P}} = 1$ still holds.

Remark 3: More detailed modeling of the Stack Reference Model, that is specific for section III.B below, is given within that section.

III. Analysis

A. The effect of broadcasting document R on the performance of requests to R

In this section, we aim at evaluating the potential performance improvement resulting from broadcasting stream *s* to a subset of proxies $P' \subseteq P$. We will consider a tagged document $R \in s$, whose number of requests is given by the random variable N_s, as noted in section II.B. In addition to the modeling assumptions given in section II, we further assume that all proxies has large enough caches to hold the document from the first request until the last local request to the proxy is made. This assumption is reasonable for capturing the behavior of document *R*.

Consider first a system *without CSDS*. Under this system, for every document under every proxy, the first request for the document will result with a cache miss (since the document is not in the cache yet). All later requests from that proxy to that document will result with a hit since the document will be present in the cache. Focusing on cache C_p and conditioning on $N_s = n$, the probability that C_p will experience a (single) miss on *R* is given by the probability that out of the *n* requests made to *R*, it will be requested at least once at C_p:

$$\Pr[\textit{a document miss at } C_p \mid N_s = n, \textit{no CSDS}] = 1 - \overline{\lambda_{s,p}}^{\,n} \quad (4)$$

where $\overline{\lambda_{s,p}} := (1 - \lambda_{s,p})$.

Let $F_{s,p}$ be the total document fault ratio of stream *s* at C_p with no CSDS, defined to be the expected number of misses per document encountered at C_p. Using (4) we can derive:

$$F_{s,p} = \sum_{n=1}^{\infty} d_s^{cnt}(n)(1 - \overline{\lambda_{s,p}}^{\,n}) = 1 - \sum_{n=1}^{\infty} d_s^{cnt}(n)\overline{\lambda_{s,p}}^{\,n} \quad (5)$$

Let $M_{s,p}$ be the total *request miss ratio* of stream *s* at C_p with no CSDS, defined to be the expected number of misses per request, and which equals to the ratio between the number of misses and the expected number of requests made at C_p. Using (4) again, we can derive:

$$M_{s,p} = \frac{\sum_{n=1}^{\infty} d_s^{cnt}(n)(1 - \overline{\lambda_{s,p}}^{\,n})}{\sum_{n=1}^{\infty} d_s^{cnt}(n)\lambda_{s,p} n} = \frac{1 - \sum_{n=1}^{\infty} d_s^{cnt}(n)\overline{\lambda_{s,p}}^{\,n}}{E[N_s]\lambda_{s,p}} \quad (6)$$

Now consider the system with CSDS, in which when the document is requested by the first proxy to request it, the document is broadcast to all other proxies in its subset *P'*. In this case, the first proxy in *P'* will experience a single miss while none of the other proxies in *P'* will experience any miss.

Under this case, C_p experiences a miss on the tagged document, only if there is at least one request to the document that originates from a proxy in *P'*, and the first such request was issued by C_p. This event is given by:

$$\Pr[\textit{document miss at } C_p \mid N_s = n, \textit{with CSDS}] = \frac{\lambda_{s,p}}{\Lambda_{s,P'}}(1 - \overline{\Lambda_{s,P'}}^{\,n}) \quad (7)$$

Let $F_{s,p}^{CSDS(P')}$ and $M_{s,p}^{CSDS(P')}$ be, respectively, the document fault ratio and the request miss ratio of stream *s* at C_p using a CSDS system connecting the subset of proxies *P'*, defined similarly to $F_{s,p}$ and $M_{s,p}$. Their values are given by:

$$F_{s,p}^{CSDS(P')} = \sum_{n=1}^{\infty}\left(d_s^{cnt}(n)\frac{\lambda_{s,p}}{\Lambda_{s,P'}}(1 - \overline{\Lambda_{s,P'}}^{\,n})\right) = \frac{\lambda_{s,p}}{\Lambda_{s,P'}}\left(1 - \sum_{n=1}^{\infty} d_s^{cnt}(n)\overline{\Lambda_{s,P'}}^{\,n}\right), \quad (8)$$

and

$$M_{s,p}^{CSDS(P')} = \frac{\sum_{n=1}^{\infty}\left(d_s^{cnt}(n)\frac{\lambda_{s,p}}{\Lambda_{s,P'}}(1 - \overline{\Lambda_{s,P'}}^{\,n})\right)}{\sum_{n=1}^{\infty} d_s^{cnt}(n)\lambda_{s,p} n} = \frac{1 - \sum_{n=1}^{\infty}\left(d_s^{cnt}(n)\overline{\Lambda_{s,P'}}^{\,n}\right)}{E[N_s]\Lambda_{s,P'}} \quad (9)$$

Remark 4: Note the similarity in form of (9) to that of (6). The only difference is that while (6) is dependent on $\lambda_{s,p}$, (9) is dependent on $\Lambda_{s,P'}$. The explanation is that from a hit/miss point of view, under CSDS the collection of participating proxies behaves as one large proxy, since they all share the same single miss per document. Note also that under CSDS, the miss ratio is independent of $\lambda_{s,p}$ and of *p*, and is the same for every participating proxy.

Let us define $G_{s,p}^{CSDS(P')}$, the hit ratio gain of C_p due to using CSDS on stream *s*, which is simply

$$G_{s,p}^{CSDS(P')} = M_{s,p} - M_{s,p}^{CSDS(P')}. \quad (10)$$

Now we can calculate $G_p^{CSDS(P')}$, the total hit ratio gain of C_p from using CSDS on all streams in *S*. This is done using the terms $\frac{\gamma_{s,p}}{\gamma_{S,p}}$, which are the probabilities that a request of proxy C_p belongs to stream *s*:

$$G_p^{CSDS(P')} = \sum_{s \in S} \frac{\gamma_{s,p}}{\gamma_{S,p}} G_{s,p}^{CSDS(P')} = \frac{\sum_{s \in S} \gamma_{s,p} G_{s,p}^{CSDS(P')}}{\gamma_{S,p}} \quad (11)$$

Remark 5: Note that for every stream *s* that is not requested by C_p, $\gamma_{s,p} = 0$, and for all streams that are not broadcast, by definition $G_{s,p}^{CSDS(P')} = 0$. Practically, we can therefore define a subset of streams $S' \subseteq S$ of broadcast streams, and to sum $G_{s,p}^{CSDS(P')}$ only on *S'* (*S'* will be used in section IV below).

B. *The effect of broadcasting document R to a proxy that will not request R*

In the previous section, we assumed that every proxy cache is of sufficient size - so it can contain all local accessed documents, as well as additional CSDS broadcast documents, as many as we choose to broadcast. Under that assumption, and if broadcast costs or constraints are not accounted for, it is clearly optimal to broadcast *all streams* to *all proxies*, because there is no constraint on the proxy cache capacity, nor on the broadcast capacity.

To properly model the negative effects of document R on other documents, one needs to model a limited capacity proxy cache. In this more realistic situation the broadcast of an un-needed document may have adverse effects, since it may occupy space in the cache and preclude more needed documents from the cache. Thus, the modeling of a finite size cache should allow one to account for the negative effects of a broadcast document. These effects can be termed as "cache contamination", namely a document that is kept in the cache while it is not needed can be considered to contaminate the cache. Wrong broadcast policy (e.g., one that broadcasts "everything") may contaminate the cache to a high degree, thus preventing it from serving the locally wanted documents.

A broadcast document is called a *waste document* in cache C_p, if it will not have any request directed to it at C_p. For a given proxy $C_p \in P'$, and a document whose number of requests is N, the document is a waste at C_p if it has at least one request by P' and no request by C_p. The probability of this event is given by:

$$\Pr[\textit{document of stream s is waste at } C_p \mid N=n] = \overline{\lambda_{s,p}}^{\,n}\left[1-\left(\frac{\overline{\Lambda_{s,P'}}}{\overline{\lambda_{s,p}}}\right)^{n}\right] = \overline{\lambda_{s,p}}^{\,n} - \overline{\Lambda_{s,P'}}^{\,n} \quad . \tag{12}$$

Now we can calculate the fraction of documents in s which end up being a waste at C_p:

$$\Pr[\textit{document of stream s is waste at } C_p] = \sum_{n=1}^{\infty}\left[d_s^{cnt}(n)\left(\overline{\lambda_{s,p}}^{\,n} - \overline{\Lambda_{s,P'}}^{\,n}\right)\right] \quad . \tag{13}$$

In a similar manner to what we did in (4)-(6), let $W_{s,p}$ be the total *request waste ratio* of stream s at proxy C_p, defined to be the ratio between the expected number of waste documents and the expected number of requests for document of stream s. Using, (13), we can derive:

$$W_{s,p} = \frac{\sum_{n=1}^{\infty}\left[d_s^{cnt}(n)\left(\overline{\lambda_{s,p}}^{\,n} - \overline{\Lambda_{s,P'}}^{\,n}\right)\right]}{E[N]} \quad . \tag{14}$$

Since the different broadcast streams are disjoint, and each broadcast document is different, we can simply total the waste for all incoming streams, after normalizing it by the relative stream rate $\frac{\gamma_{s,P}}{\gamma_{S,P}}$. Thus, we get

$$W_{S,p} = \sum_{s\in S}\frac{\gamma_{s,P}}{\gamma_{S,P}} W_{s,p} \quad . \tag{15}$$

To end our definitions, we will further define the proxy's total request ratio $L_p = \frac{\gamma_{S,p}}{\gamma_{S,P}}$.

To model the interaction of the waste streams with the other streams on the cache assume that the streams are having *stack depth distributions*. When a stream is said to have a stack depth distribution it means that when the stream is applied to a LRU-managed cache, the probability that the current reference finds the element that it references at depth n ($n=1$ is the most recently used element in the cache) is $d(n)$. Given the above distribution we can define a matching cumulative distribution $D(k)=\sum_{n=1}^{k} d(n)$, which is actually the stream hit ratio for cache size k, i.e. the probability that the element will be found at depth smaller or equal to k. Note that $d(n)$ is a ``defective" distribution in that it may not sum to unity: elements that have never previously been accessed will be assumed to be found at an infinite stack depth.

We will assume that each of the streams (the waste stream and the stream consisting of the other requests) obey the *Stack Reference Model* (*SRM*), that is, they are stochastic processes which choose their next reference according to independent samplings of the stack depth distribution. Such processes are called *stack depth processes*, and their hit ratio curves as a function of cache size coincide with their cumulative stack depth distributions. This is one of several simple models for reference streams, and it has been reported, for example in [5], that this model tends to be quite successful in capturing temporal locality of references within a trace, and is superior to the so called *Independent Reference Model* (*IRM*).

Given the stack depth distribution of proxy C_p, $d_p(n)$, and the corresponding cumulative distribution $D_p(n)$, we will calculate the effect of the waste stream on the stack depth distribution. To this end, we will use a methodology similar to the one developed in [8], [9] and we will track a tagged local document R through its journey through the local cache at which both the local documents and the waste documents accumulate. This will allow us to derive the distribution of total depth (consisting of local and waste pages) at which R is requested, which will form the depth distribution of the combined cache.

Let n_L denote the number of local documents (non-waste documents) residing "above" (that is, at a lower depth) R in the cache, and let n_W denote the number of waste documents residing above R in the cache. Then, at any time in the journey, the position of R in the cache is described by the state (n_L, n_W). To track the behavior of this cache we will focus only on the events where there is access to this cache, either by the waste documents or by the local requests. The probabilities that an event is a waste document arrival or a local access are given by

$$x_p^W := W_{S,p}/(W_{S,p}+L_p) \tag{16}$$

and

$$x_p^L := L_p/(W_{S,p}+L_p) \tag{17}$$

respectively.

Since these probabilities are fixed and do not depend on the other events, the state (n_L, n_W) is sufficient to predict the future of R. To track the behavior of R in the cache, recall that the cache operates under the LRU policy. This means that when any document *R'* is requested, the document is placed at the top position (whose depth is 1) of the cache. This causes all the documents which have been prior to the operation above *R'* to be pushed one position downward (deeper). In the event that *R'* was not in the cache prior to the request, this should cause the deepest document in the cache to be pushed out.

Now, to track the journey of the tagged document *R* along the cache, assume that *R* is in state (n_L, n_W) just after the *i-th* request ("time *i*") and examine the transition or *R* due to the *i+1st* request. *R* will encounter one of the following events:

1. *R* remains at (n_L, n_W). This occurs if a request is made for a local document of depth smaller than or equal to n_L. Thus, the probability of this event is given by

$$x_p^L D_p(n_L). \tag{18}$$

2. *R* moves to (n_L+1, n_W). This occurs if a request is made for a local document of depth greater than n_L+1. Thus, the probability of this event is given by

$$x_p^L \overline{D_p(n_L+1)}. \tag{19}$$

3. *R* moves to (n_L, n_W+1). This occurs if a request is made for any waste document. Thus, the probability of this event is given by

$$x_p^W. \tag{20}$$

4. *R* finishes its journey. This occurs if a request is made for a local document of depth n_L+1. Thus, the probability of this event is given by

$$x_p^L d_p(n_L+1). \tag{21}$$

The list of the states to which *R* moves *given that it leaves* (n_L, n_W), and the corresponding transition probabilities, are (Note that the probabilities below are independent of n_W.):

1. *R* moves to (n_L+1, n_W). The probability of this event is

$$\frac{x_p^L \overline{D_p(n_L+1)}}{x_p^L \overline{D_p(n_L)} + x_p^W}. \tag{22}$$

2. *R* moves to (n_L, n_W+1). The probability of this event is

$$\frac{x_p^W}{x_p^L \overline{D_p(n_L)} + x_p^W}. \tag{23}$$

3. *R* finishes its journey. The probability of this event is

$$\frac{x_p^L d_p(n_L+1)}{x_p^L \overline{D_p(n_L)} + x_p^W}. \tag{24}$$

Now let $q(n_L, n_W)$ denote the probability that *R* will eventually reach the state (n_L, n_W) in its journey, assuming that the journey starts at the top of the stack, that is defined as state *(0,0)*. Thus, we clearly have *q(0,0)=1*, and $q(n_L, n_W)$ can be calculated recursively as follows:

$$q(n_L,n_W) = q(n_L-1,n_W)\frac{x_p^L \overline{D_p(n_L)}}{x_p^L \overline{D_p(n_L-1)} + x_p^W} + q(n_L,n_W-1)\frac{x_p^W}{x_p^L \overline{D_p(n_L)} + x_p^W}. \tag{25}$$

Now, let $e(n_L, n_W)$ denote the probability that *R* will finish its journey at (n_L, n_W). This value is given by the probability that *R* will reach (n_L, n_W), and then it will be called at C_p:

$$e(n_L,n_W) = q(n_L,n_W)\frac{x_p^L d_p(n_L+1)}{x_p^L \overline{D_p(n_L)} + x_p^W}. \tag{26}$$

Now we can compute the depth distribution of a local document in proxy C_p, $d_{p,L}(n)$:

$$d_{p,L}(n) = \sum_{n_L=0}^{n-1} e(n_L, n-1-n_L). \tag{27}$$

Since all requests originating at proxy C_p are for local documents, this depth distribution is the distribution of an arbitrary document in the merged cache in proxy C_p.

Finally, the hit rate in this cache, when its size is *K*, is given by:

$$D_{p,L}(K) = \sum_{k=1}^{K} d_{p,L}(k). \tag{28}$$

C. The Net Hit Ratio Gain: Accounting for Gain and Loss

Following the analysis in sections III.A and III.B, we can derive the *net* hit ratio gain on proxy C_p, with cache size K_p. This is done by subtracting the loss (Section III.B above) from the gain (Section III.A above), to get the net hit ratio gain $G_p^{CSDS\_NET(P')}$, using (11) and (28):

$$G_p^{CSDS\_NET(P')} = G_p^{CSDS(P')} - \left(D_p(K_p) - D_{p,L}(K_p)\right). \tag{29}$$

Finally, let us define $G^{CSDS(P')}$, as the total hit ratio gain over all proxies $C_p \in P'$. Just like in deriving (11), we need to take caution in selecting the normalizing factor, which is now $\frac{\gamma_{S,p}}{\gamma_{S,P'}}$, i.e. the probability that an arbitrary request originated at proxy C_p, given it originated at one of the proxies participating in CSDS. Thus, we get:

$$G^{CSDS(P')} = \sum_{p\in P'} \frac{\gamma_{S,p}}{\gamma_{S,P'}} G_p^{CSDS\_NET(P')}. \tag{30}$$

D. Computational complexity

Equations (25)-(28) can be computed in a recursive manner (starting from the low indices and going upwards). Thus, the computation complexity per proxy is $O(J^2)$ where J is the number of entries in the stack depth distribution. In a straightforward approach, one would take J to be the cache size (K), in which case the complexity is $O(K^2)$. However, if one is interested in reducing the complexity, one can represent the distribution by $J = \alpha K$ values, where $\alpha < 1$. In this case the complexity is $O(\alpha^2 K^2)$. Proper selection of α (small value) can lead to a drastic reduction in the complexity without significantly affecting the accuracy.

IV. OPERATIONAL RULES AND EFFICIENT DESIGN

The model and analysis provided in this paper form a tool that can be used in the design and operations of a CSDS. An important question that needs to be addressed by the operator of the CSDS is the stream assignment problem, i.e. which classes (or streams) of documents should be broadcast in the system. Intuitively, it is expected that the operator should decide to broadcast classes of documents of "common interest" to the various proxies. The results provided in this analysis, namely in (29), represent the net hit gain on C_p due to broadcasting a set *S'* of streams (as introduced in Remark 5 above). Thus, to answer the question of which streams to broadcast the operator can compute (29) for various sets of streams, and derive the relative benefits to the systems as the result of broadcasting alternative sets of streams. For example, suppose that set *S'* is broadcast, and the operator would like to extend the set to $S' \cup s$ where *s* could be any of *J* alternative streams $s_1,...,s_J$. The operator can then evaluate (29) for $S' \cup s_i, i = 1,...,J$, and use it to rank the streams s_i ,*i=1,…,J*, and to decide which of them will be broadcast. Having selected the stream s_i, a new set is now formed, $S' \cup s_i$. The process may now repeat on how to extend $S' \cup s_i$.

Another important design question is the proxy assignment problem, i.e. which proxies should participate in a system. A simpler question can be whether to add a proxy C_p to an existing system where the participating proxies are the set *P'*. The value of adding C_p to the system can again be evaluated by applying (29) to the set *P'* and to the set $P' \cup C_p$.

These questions and other relevant questions, and a more detailed analysis of them, are a topic of current ongoing research.

V. NUMERICAL RESULTS

In this section we aim to examine the quality of the model and analysis. We will first describe the system we generated, and then test our analytic models against simulations.

We consider a system composed of four proxy caches, receiving requests on five Zipf distributed SRM streams. Any other distribution might be used[1], but the Zipf distribution might emulate real life the best. The relative request rate for each proxy and stream, used to generate the $\gamma_{s,p}$ matrix, is given in Table 1. Stream 1 is requested mainly (90%) by Proxy 1, and the rest of the requests are divided uniformly across the rest of the proxies. Stream 2 is requested mainly (80%) by Proxy 2, and the rest of the requests are divided uniformly across the rest of the proxies. Streams 3 and 4 are divided uniformly across all proxies. Stream 5 is divided uniformly across all proxies except Proxy 4. All streams have the same total request share, except for Stream 4, whose request volume is 5 times lower than the volume of each of the other streams.

The results are generated as follows: We start by generating a simulative request log, composed of 5,000,000 (five million) requests, based on the relative request rate matrix and the global streams' SRM stack depth distributions, classifying each request to arrive from a stream and a proxy. Then we can simulate any broadcast scenario. In addition, we generate from the log the request count distribution $d_s^{cnt}(n)$ for every global stream, and the local stack depth distribution $d_p(n)$ for every proxy. The generated cumulative local stack depth distribution (across all streams) for each proxy, which is equal to the local hit ratio of every proxy without using CSDS, is given in Fig. 2. Now, for every broadcast scenario we need to run the analysis of section III.C above and the simulation, and compare the total net gain for the given scenario.

Table 1: The matrix of relative proxy & stream ratios

	Proxy1	Proxy2	Proxy3	Proxy4	Total/St.
Stream1	900	33	33	34	1000
Stream2	66	800	67	67	1000
Stream3	250	250	250	250	1000
Stream4	50	50	50	50	200
Stream5	333	333	333	1	1000
Total/Pr	1599	1466	733	402	4200

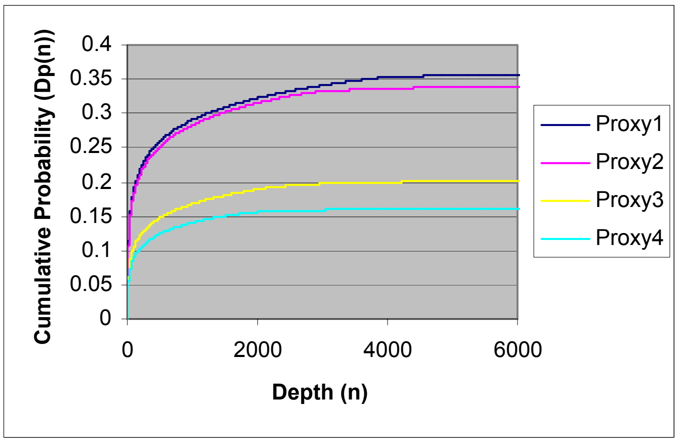


Figure 2: Calculated cumulative local stack depth distribution (local hit ratio)

[1] We tested the system on various distributions, including handpicked distributions and geometric distributions, and received similar accuracy.

First, we examine the effect of broadcasting each of the various streams on every proxy. We consider a cache size of 4,000 documents[2], which is on the order of the saturation point of our distribution (the point at which increasing the cache size does not increase the local hit rate), and run separate CSDS, each broadcasting one stream. The results are reported in Fig. 3. First, we notice the good correlation between the analytic results and the simulation results, for the entire test scenario. We observe that the most beneficial stream to broadcast is, as expected, the high-volume, evenly-distributed stream 3. We see a difference of its contribution for different proxies, since its relative presence in Proxy 3, is the highest. Observe that Stream 5 actually reduces the hit rate of proxy 4, because broadcasting it only contaminates proxy 4's cache, but does not contribute any hits. For the other proxies it contributes quite a lot, because, again, it is distributed evenly across those proxies. Stream 1 is not distributed uniformly: Proxy 1 receives most requests, and thus it will broadcast a high amount of documents to the other proxies. Now, if the receiving proxy has a low amount of other requests (like Proxy 4), receiving Stream 1 will not degrade the local hit rate by a substantial amount. For "loaded" proxies, Proxy 2, the benefit we observe is small, because the arriving documents throw out other important documents. Stream 3 is evenly distributed, but due to its low volume, it does not contribute much gain.

We then repeat the experiment for a much smaller sized cache, of 1000 documents. The results are reported in Fig. 4. We notice our analytic results are still very well correlated to the experimental results, although the differences are more noticeable. Notice also that now the gains are smaller (since the contamination has a higher impact), especially on proxies with low correlation with the broadcast stream (e.g. stream 2 on proxy 1, or stream 1 on proxy 2).

As a comparison, we give in Fig. 5 a partial broadcast set scenario, of streams 3, 4, and 5, for a 4,000 documents sized cache, and a full broadcast set scenario, of streams 1-5 for the same cache size. We notice, again, the good correlation between the analytic results and the simulation results. We can also see the high gain of Proxy 3 and Proxy 4, which are relatively small proxies.

We can also check our main source of inaccuracy – misses that are not "first access" misses. Such misses appear when the cache sizes are small in a considerable measure than their stack depth distribution, so they can not hold even sufficient local requests. As already mentioned, this is not a common case in real life, and in our model we assumed it could be neglected. In Fig. 6 the effect of decreasing cache sizes on a CSDS that broadcasts only the uniformly distributed stream 3 can be seen. Since we mostly care about the relative error in the gain calculation, we plot the difference of the error in calculating each proxy's gain, from the weighted average error of the entire system. In Fig. 7 we repeat the experiment for a CSDS that broadcasts only stream 5, which is not distributed uniformly. First, notice in both cases the error diverges for very small caches. Of course as the cache is smaller it diverges faster (so we notice Cache 4 diverges the fastest). For all caches larger than 700 documents, we get in both cases a reasonable error divergence (less than 1 percent apart). Caches smaller than 700 documents are not likely to be practical (see their very low hit ratio at Fig. 2).

Thus, we observe that for all reasonable cache sizes that estimation error of the analysis is limited and the model will yield good results, especially for optimization and operational rules.

[2] We repeated the numerical tests for cache sizes of 400 and 40,000 (with proper caches and distributions) as well, and received similar accuracy.

VI. SUMMARY

In this work we dealt with the Cache Satellite Distribution System and aimed at providing a framework for the analysis and efficient operation of this system. We proposed a model that accounts for the intrinsic behavior of caches and request streams and which captures both the locality of reference experienced in these streams and their relative frequencies. Using the model we provided an analysis that predicts the hit ratio gain for each stream and each proxy as function of the stream properties. We presented how these predictions, whose computational complexity is relatively low, can be used directly in solving the operational and design questions. Numerical examination versus simulation indicates that the analysis indeed captures the relative effect of various streams on various proxies and thus will predict well the relative merits of alternative operational rules.

REFERENCES

[1] V.F. Almeida, M.G. Cesario, R.C. Fonseca, W. Meira Jr., and C.D. Murta, "Analyzing the Behavior of a Proxy Server in Light of Regional and Cultural Issues", *3rd International WWW Caching Workshop*, June 1998

[2] D. Askoy, M. Franklin, and S. Zodnik, "Data Staging for on-demand broadcast," in *Proceedings of the 27'th VLDB Conference,* 2001.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenkar, "On the Implications of Zipf's Law for Web Caching", Proceedings of *the 3rd International WWW Caching Workshop*, June 1998

[4] S.-G. Chang, "Caching Strategy and Service Policy Optimization in a Cache-Satellite Distribution Service", *Fifth INFORMS Telecommunications Conference*, March 2000.

[5] E. G. Coffman and P. J. Denning, "Operating System Theory," Prentice-Hall, 1973.

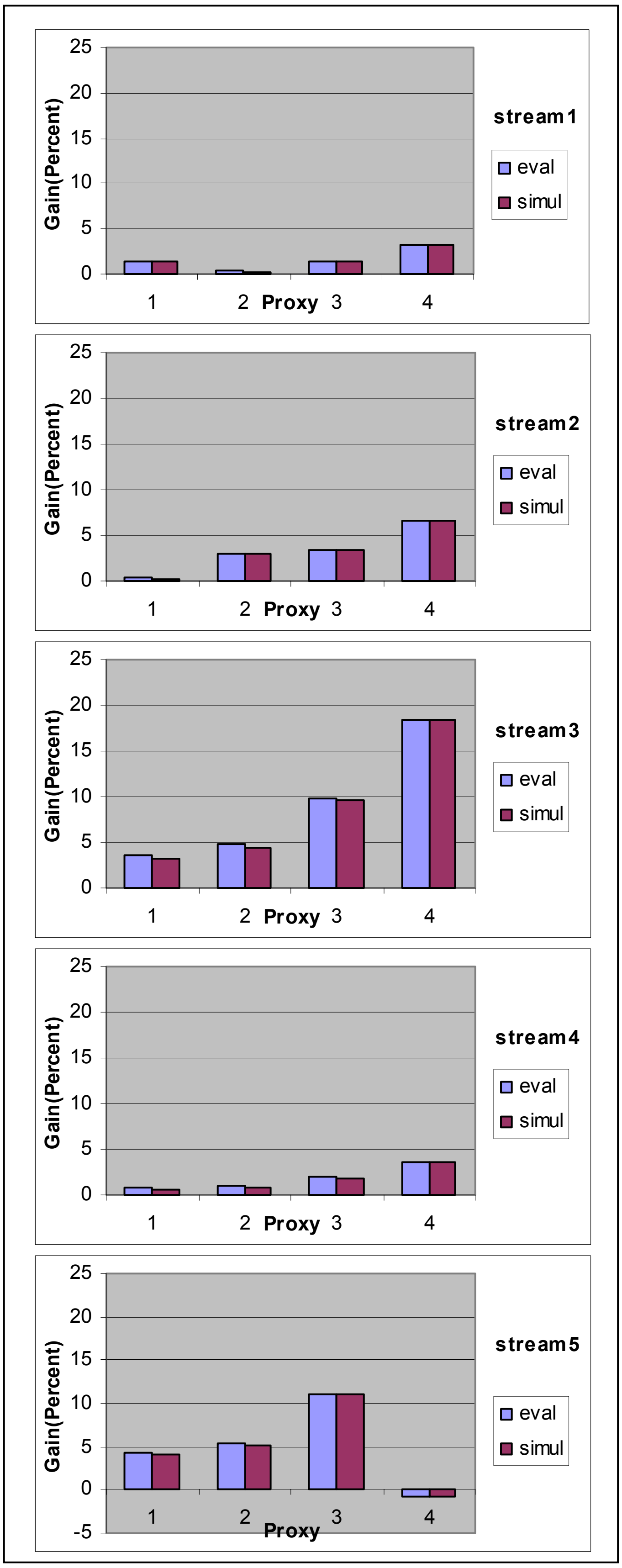
[6] R. Cohen, L. Kazir, and D. Raz, "Scheduling Algorithms for a Cache Pre-Filling Content Distribution Network", *Infocom 2000*, New York, June 2002.

[7] X.-Y. Hu, P. Rodriguez, and E.W. Biersack, "Performance Study of Satellite-linked Web Caches and Filtering Policies", *Networking 2000*, Paris, May 2000, 580-595.

[8] H. Levy and R.J.T. Morris, "Exact Analysis of Bernoulli Superposition of Streams into a Least Recently Used Cache", *IEEE Trans. On Software Eng*. 21:8, 1995, 682-688.

[9] H. Levy and R.J.T. Morris, "Should Caches be split or combined? Analysis using the superposition of bursty stack depth processes," *Performance Evaluation*, 27 & 28 (1996), pp. 175-188.

[10] P. Rodriguez and E.W. Biersack, "Bringing the Web to the Network Edge: Large Caches and Satellite Distribution", *Mobile Networks and Applications*, 7(1):67-78, January 2002.



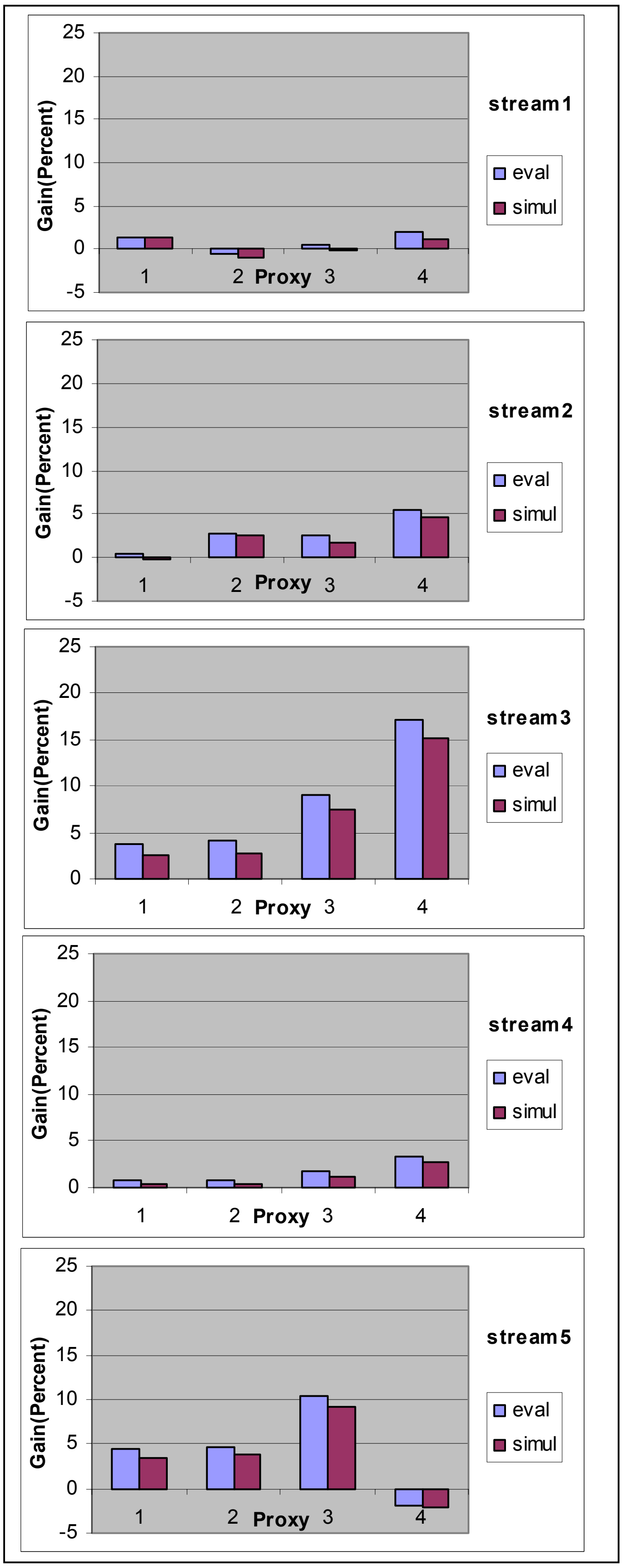


Figure 3: Simulative and analytic gain for a single stream broadcast, for cache size = 4,000 documents

Figure 4: Simulative and analytic gain for a single stream broadcast, for cache size = 1,000 documents

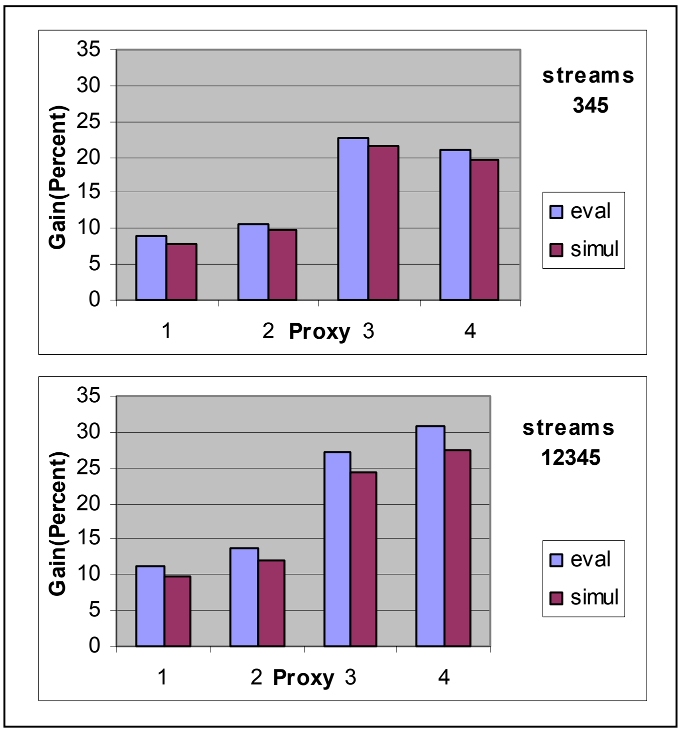


Figure 5: Simulative and analytic gain for multiple streams broadcast, for cache size = 4,000 documents.

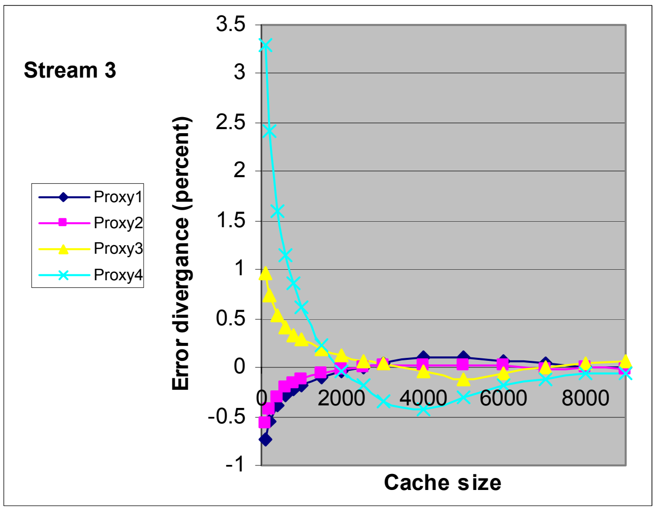


Figure 6: Difference of stream 3 per-proxy error ratio from the total error expected value.

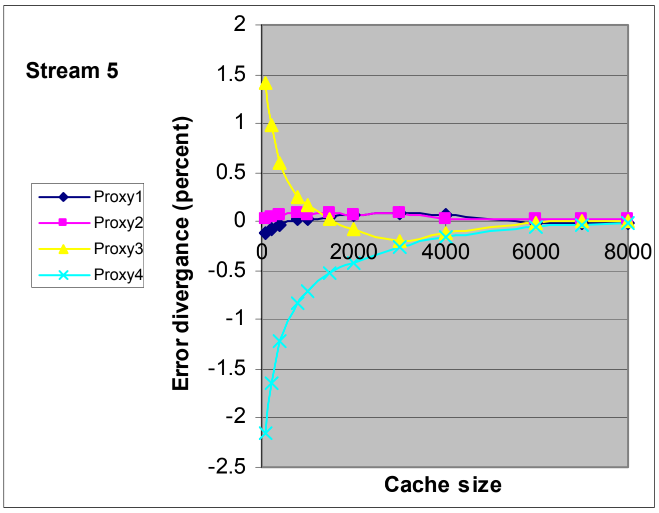


Figure 7: Difference of stream 5 per-proxy error ratio from the total error expected value.