Analytic Comparison of Congestion Effects for Alternative Web Proxy Cooperation Models

EUGENIO DE LA ROSA[a] JOHN H. HARTMAN [b] and TERRIL HURST[c]

[a]Optical Sciences Ctr, Univ. of Arizona, Tucson, Arizona 85721
[b]Dept. of Computer Science, Univ. of Arizona, Tucson, Arizona 85721
[c]Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304
eugenio@u.arizona.edu

Abstract: – An analytic model was developed to evaluate network congestion effects on alternative Web proxy cooperation models (CMs). The model was applied to a system of identical, fully connected proxies, in order to compare performance of two common CMs: Broadcasting and URL–Hashing. The model has been validated against simulations performed in NS-2 with real proxy traces. Sensitivity was also determined for the future growth of request rates and number of cacheable Web objects. Our study provides estimates of limits within which cooperative Web caching can be expected to yield improvements over single–proxy operation.

Keywords: Web caching, analytic model, congestion avoidance

1. Cooperative Proxy Caching

The purpose of Web caching is to move content closer to the user, typically onto local storage of a proxy server that is part of the user's local area network. Requests that hit in the proxy cache are served by the proxy, avoiding expensive accesses to the origin Web server on the Internet. Web proxies have proven effective at reducing overall bandwidth requirements, server load, and latency. But the proxy can be a bottleneck, as it must process all requests by the clients. Multiple proxies can mitigate the problem, and if inter–proxy communication is not excessive, the proxies can coordinate their actions so as to optimize system performance. Forecasts of increased Web usage highlight the necessity of such cooperation among proxy servers as a means of reducing latency and network congestion,[1,2] .

Existing Web cache *cooperation models* (denoted by "CM," since the term "model" is used herein to refer to the analytic representation) fall into two broad categories: those that use a static mapping from Web pages to proxies, and those that use a dynamic mapping. The former is exemplified by URL–Hashing, in which the URL of the desired page is hashed to determine which proxy to access. Each URL is mapped to a single proxy, ensuring that there is only one copy per page in the proxy caches, thus maximizing the overall hit rate. URL–Hashing guarantees load–balancing among all proxies,[3] since any locality in the URL request streams is randomized by the hash. Balancing proxy loads is necessary for minimizing queueing delays at proxies and for improving access latency.

Other CMs use a dynamic mapping from URL to proxy. A good example of this is Broadcasting. Requests are broadcast to all the proxies to find the desired page.[4] The Internet Caching Protocol (ICP) was developed to aid in the query process.[5] Allowing a dynamic mapping from URL to proxy has the advantage that it allows proxies to operate independently; each can decide on its own what to cache and what not to cache. It also allows multiple cached copies of the same page. This is important for hot pages that are accessed frequently. The load induced by these pages may need to be spread across several servers. There are a few downsides of dynamic mapping. The first is that multiple copies of a page pollute the cache, reducing overall hit rate and mitigating some of the load-balancing benefits. Second, a method is needed for locating pages, which introduces overhead and complexity.

The particular instance of Broadcasting to find a page doesn't scale well to large numbers of proxies.

One thing that existing CMs have in common is that while they attempt to minimize access latency by balancing server loads and maximizing the overall cache hit rate, they do not take into consideration network congestion between proxies. Existing CMs assume that any client can access any proxy with the same network latency and bandwidth. This may not be the case due to network congestion, caused by requests from other clients, or simply cross-traffic on the same network. Minimizing access latency requires evaluating congestion, as well as balancing server loads and avoiding duplicate copies.

In this paper, after summarizing an analytic model of CMs that includes the effects of network congestion, we use it to forecast future performance of some CMs in the presence of different Web traffic growth scenarios. The model builds on work done by Wolman *et al*,[6] who expanded upon previous work done by Breslau *et al*.[7] Wolman's model is recognized as the best available tool for predicting the performance of large–scale Web caching.[8] This model has been already used in the analytic study of the Restricted Broadcast Query mechanism.[9] Section 2 describes previous work that has been done on Web proxy cooperation. The analytic model and test configuration of cooperating proxies are described in Section 3, including assumptions and parameter values. Section 4 presents results, Section 5 describes the validation of the analytic model through simulations, and Section 6 includes conclusions.

2. Related Work

Mature examples of proxy cache cooperation appear within commercial products,[1011] for LAN and WAN deployment and within government–funded research efforts,[12] .[13] The following examples describe other ideas found within the research literature.

The Internet Cache Protocol (ICP) was developed by the Harvest project.[5] ICP was first applied in a CM known as Broadcasting or Multicasting.[14] Upon a cache miss, ICP packets are transmitted to all cooperating proxies. Any cache member containing the requested object is identified, and the request will be forwarded to it. The individual proxies work autonomously, taking into account proximity to each other. Unfortunately, misses are discovered too late, since the querying proxy must wait for all caches to respond.[3] Therefore, in large configurations, hierarchical ICP caches have overhead–induced scalability issues.[10]

The CRISP project used directory schemes as a CM.[15] If a central directory is used, it becomes a bottleneck and a single point of system failure. If distributed directories are used, a high level of agreement is required among proxies. Directory exchanges lead to a heavy maintainance and query overhead.[3] Summary Cache[16] is an optimized version of directory schemes. The cache directory that each proxy keeps from the cached content at every other proxy is compressed using Bloom filters. A trade-off exists between scalability and the summaries' accuracy or update frequency.

Consistent Hashing[17] was implemented in the Cache Resolver[3]; it is also known in the literature as URL-Hashing. Browsers decide which proxies to contact using a hash function that maps URLs to a dynamic set of available caches. Inter–cache communication is largely eliminated, reducing overhead, but this also means that the scheme has no awareness of network proximity. The Cache Array Routing Protocol implements URL-Hashing for each proxy autonomously.[10]

In the Cache Routing approach,[18] there are no parent relationships among caches. Smart routing of cache requests is accomplished by gathering data from the Border Gateway Protocol, using the Whois service and modifying the tool traceroute. In Vicinity Caching, each proxy maintains directories of objects cached only in neighboring proxies.[19]

Congestion has been extensively studied in the last 20 years, but the bulk of the research has been done at levels lower than the Web application. To

our knowledge, the work reported within this paper is the first to analyze the effects of congestion on coooperative Web proxy caching. We chose to model congestion in two idealized CMs, Broadcasting and URL–Hashing, because they represent respectively the set of CMs using either dynamic or static mapping of URLs to proxies.

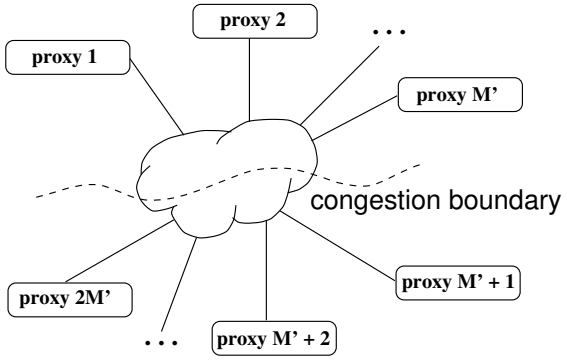

Figure 1. System of $M = 2M'$ identical, fully connected proxies. Each proxy also connects to N clients (not shown).

3. Web–Object Retrieval Process

We now describe the analytic model—including notation, assumptions, and equations—for comparing the performance of proxy CMs. Section 3.1 describes Wolman's basic model[6] as applied to single–proxy congestion; Section 3.2 includes the enhancements that we developed for comparing CM performance under proxy–link congestion.

3.1. Wolman's Single–Proxy Model

Consider a Web proxy servicing N identical clients. The total number of cacheable Web objects is n. The objects are rank–ordered according to their popularity, forming a Zipf–like distribution with exponent $\alpha = 0.80$,[4,7] The distribution is divided into two groups: n/k popular objects, and $n - n/k$ less–popular objects. From the literature, we set $k = 3000$.[6]

Each client is requesting objects at a mean rate λ. The set of popular objects is changing at rate P, and the set of unpopular objects is changing at rate U. Assuming that the size and popularity of the Web objects are uncorrelated, the following expressions give the predicted single–proxy cache hit rate, H_0, as a function of the above parameters:

$$H_0 = \int_1^{n/k} Y\left[\frac{1}{1+PZ}\right] dx + \int_{n/k}^{n} Y\left[\frac{1}{1+UZ}\right] dx \quad (1)$$

$$Y(x) \equiv \frac{1}{\Omega \cdot x^{\alpha}}; \quad (2)$$

$$Z(x) \equiv \frac{\Omega \cdot x^{\alpha}}{\lambda \cdot N}; \quad (3)$$

$$\Omega \equiv \int_1^{n} 1/x^{\alpha} dx; \quad (4)$$

3.2. Model Enhancements

In order to use Eqns. 1–4 for comparing different CMs, we augment the single–proxy model in the following ways:

- *Topology–neutral comparison.* One enhancement of Eqns. 1–4 arises from our desire to compare benefits of alternative CMs. We focus on the overhead and communication patterns between M cooperating proxies. Therefore, we use the same, fully–connected proxy topology for each CM—see Fig. 1.

- *Integration limits.* Since Eqns. 1–4 describe single–proxy operation, the hit rate is determined by integrating across the entire set of n objects. For specific CMs, the set of objects that each proxy can retrieve may be limited. Therefore, the integration limits will require modification.

- *Finite cache size.* Wolman's steady–state performance model assumes that cache size is infinite, and that cache contents change at rates P and U due to invalidations and updates from the origin server. But Wolman's values for P and U were obtained by measuring real systems having *finite* cache size. Therefore, cache size must also influence the time that items remain in cache. Thus, in applying Wolman's model to CMs, we make rates P and U functions of cache size.

The above enhancements have different implications for each CM; CM–specific equations are denoted with subscripts b for Broadcasting and h for URL–Hashing. In Broadcasting, clients query all available proxies. Clients see an effective cache size equal to the sum of M cooperating caches, which in turn see a request rate $\lambda \cdot N \cdot M$, the sum of arrivals from all clients. The larger cache reduces rates P and U:

$$P_b(M) = P/M; \quad U_b(M) = U/M \tag{5}$$

Furthermore, since each cooperating proxy may access any object, the integration limits within Eqns. 1–4 apply. The hit rate for Broadcasting is thus given by:

$$H_b(M) = \int_1^{n/k} Y\left[\frac{1}{1+P_b(M)Z}\right]dx + \int_{n/k}^{n} Y\left[\frac{1}{1+U_b(M)Z}\right]dx \tag{6}$$

For URL–Hashing, the set of objects that can arrive within a given proxy's cache is determined by the hashing function that maps URLs to proxies. Assuming perfect load–balancing across M proxies and no object replicas, each proxy's cache will see a request rate λN, and rates P and U will not depend on M. Rather, dependence on M is within the integration limits:

$$H_h(M) = \int_1^{n/kM} Y\left[\frac{1}{1+PZ}\right]dx + \int_{n/kM}^{n/M} Y\left[\frac{1}{1+UZ}\right]dx \tag{7}$$

$$\Omega_h = \int_1^{n/M} \frac{1}{x^\alpha}dx \tag{8}$$

Fig. 2 shows hit rates for each CM, inserting specific parameter values from the literature[6] into Eqns. 5–8. The object request rate $\lambda = 10,000$ objects per day; $P = 0.600$ objects per day; $U = 0.036$ objects per day. $N = 800$ clients per proxy. The total number of cacheable Web objects $n = 4$ billion. Hit rates for each CM are identical—within the range of numerical integration error. This reflects our assumptions that (1) the CMs have identical workloads and local cache replacement policies, and (2) although Broadcasting caches multiple copies of a small number of popular objects, their relatively small number means that differences in CM cache contents will be neglibly small.

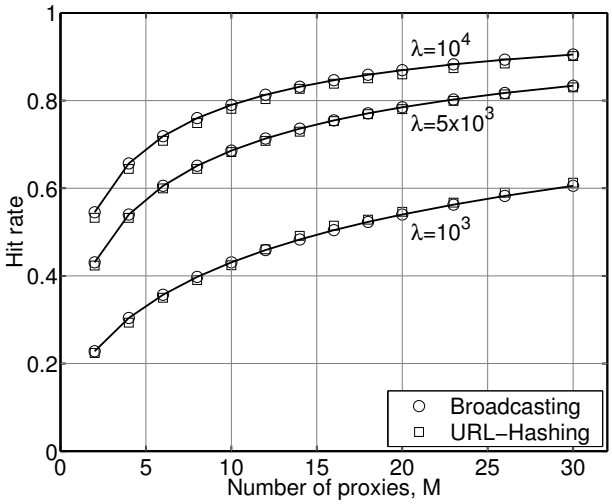

Figure 2. Hit rates vs. number of proxies for each CM.

The final elements required for CM analysis are processing times and hop latencies. The following values are typical of those found in the literature for LAN–connected systems,[4],[2021]; WAN–connected cases could be considered by appropriately modifying the values.

ICP packet processing: $\tau_{icp} = 1.4$ mS (9)

Broadcasting timeout: $\Delta_b = 45$ mS (10)

URL–Hashing lookup: $\tau_h = 0.25$ mS (11)

Proxy–origin server hop latency: $\Delta_{po} = 65$ mS (12)

Client–proxy hop latency: $\Delta_{cp} = 2$ mS (13)

Proxy–proxy hop latency: $\Delta_{pp} = \beta \cdot 5$ mS (14)

The ICP protocol affects Broadcasting performance through ICP packet processing time (Eqn. 9) and through the assigned timeout value (Eqn.

10). Similarly, URL–Hashing's table–lookup time (Eqn. 11) is counted as a processing time.

Hop latencies in Eqns. 12–14 are applied to both CMs, and are the mean times required for Web objects to travel between client and proxy, proxy and origin server, and between proxies. Each of these hop latencies includes the time for executing the HTTP protocol. Additionally, assignment of these hop latency values is equivalent to assigning link bandwidth capacities, since—within a given calculation—we fix the size of Web objects at the current estimated average, 10 kilobytes.[22] We introduce the effects of congestion between selected proxy links through the multiplier β in Eqn. 14.

3.3. Considering Congestion

A fair comparison of the effects of congestion on different CMs requires a combination of proxies and links that does not favor one CM over others. Therefore, we chose a system of identical, fully connected proxies—see Fig. 1. The dashed line denotes a *congestion boundary*, across which proxy links have congestion factor $\beta > 1$ (Eqn. 14). For example, a value of $\beta = 4$ means that available bandwidth for proxy links crossing the congestion boundary is reduced to 25% of its original value.

The results shown in Section 4 were calculated by summing the processing times and hop latencies while tracing a Web object's request path between two proxies and back again. The request path is determined by the particular CM; since the CM is unaware of congestion, some will cross the congestion boundary (i.e., $\beta > 1$), and others will not ($\beta = 1$). For the configuration in Fig. 1, on average, half of the requests will cross the congestion boundary, and half will not.

3.4. Retrieval Times

We can now write the expressions for each CM's end–to–end retrieval time in terms of hit rate, processing times, and hop latencies.

Since the following expression occurs within each retrieval time equation, we give it a shorthand notation:

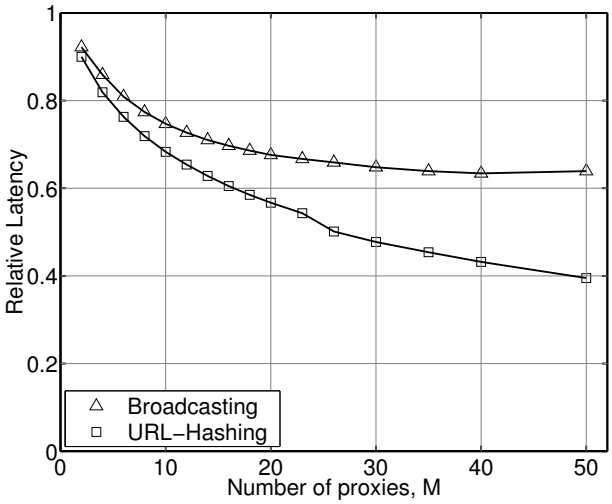

Figure 3. Relative Latency vs. number of cooperating proxies. $n = 4 \times 10^9$; $\lambda = 10^3$ (no congestion).

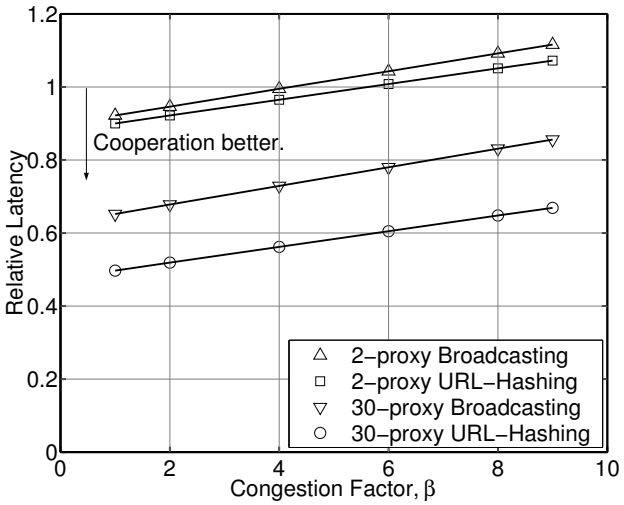

Figure 4. Comparing CMs' effectiveness under congestion conditions ($n = 4 \times 10^9$, $\lambda = 10^3$).

$$\gamma \equiv 2\,(\Delta_{pp} + \Delta_{po}) + \Delta_{cp} \tag{15}$$

CM retrieval times are given by:

$$R_b = \Delta_{cp} + H_b(2\Delta_{pp} + \Delta_{cp} + M\tau_{icp}/2) + (1 - H_b)\,(\gamma + \tau_{icp}) \tag{16}$$

$$R_h = \Delta_{cp} + H_h(2\Delta_{pp} + \Delta_{cp} + \tau_h) + (1 - H_h)\,(\gamma + \tau_h) \tag{17}$$

Our CM performance metric is *Relative Latency*, which is retrieval time for cooperation normalized by retrieval time for stand–alone proxy operation:

$$L_b \equiv \frac{R_b}{R_0}; \qquad L_h \equiv \frac{R_h}{R_0} \tag{18}$$

where stand–alone retrieval time is given by:

$$R_0 = \Delta_{cp}(1 + H_0) + (1 - H_0)(2\Delta_{po} + \Delta_{cp}) \tag{19}$$

Note that both numerator and denominator are non–constant functions of the parameters being explored, λ, n, M, and β. The highest congestion factor that was used was $\beta = 9$, since higher values would result in proxy–proxy hop latency exceeding the timeout value given for Broadcasting (Eqn. 10). Beyond this level, additional factors that are not included in our simple model would influence actual CM behavior.

4. Analysis Results

Fig. 3 shows that the Relative Latency curves for Broadcasting and URL–Hashing flatten at higher values of M. This is due to two reasons: (1) although total cache size is increasing, the number of cacheable objects n is constant, leading to diminishing returns from higher hit rates; (2) from Eqn. 16, we note that Broadcasting overhead increases directly with M, leading to more rapid flattening in Relative Latency than URL–Hashing. Based on these factors, we chose to use only values $M \leq 30$ for the remainder of results shown below.

Fig. 4 shows that Relative Latency worsens linearly with congestion for both CMs. For a given congestion factor, each CM's Relative Latency depends on hit rate—which grows with M (Fig. 2) and on cooperation overhead—which also grows with M. For a given number of proxies, there exists a threshold congestion factor, above which cooperation worsens performance (Relative Latency greater than 1). Since URL–Hashing shows better performance in the presence of congestion, and due to space limitations, we limit the remainder of results shown to URL–Hashing.

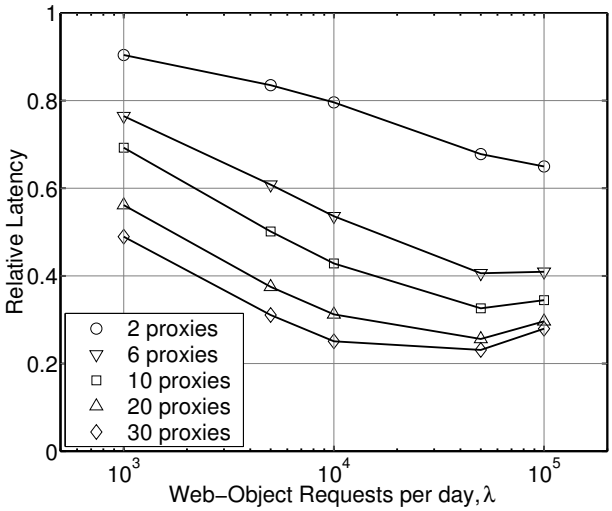

Figure 5. Impact of Web–object request rate, λ, on URL–Hashing (URL–Hashing; $n = 4 \times 10^9$, no congestion).

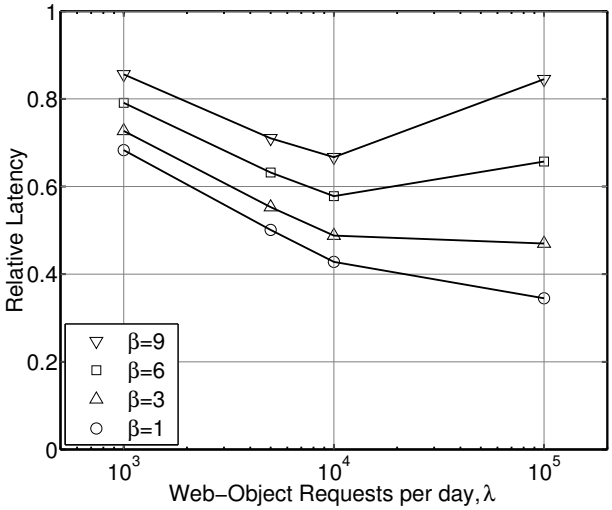

Figure 6. Impact of Web–Object request rate for various congestion factors (URL–Hashing; $n = 4 \times 10^9$, $M = 10$).

Fig. 5 shows the relationship between request rate λ and number of proxies, M, for URL–Hashing under no congestion ($\beta = 1$). Assuming that servers can handle the higher request rate, as λ increases, for $M < 6$, performance improves due to increased hit rate. But for $M > 6$ and $\lambda > 5 \times 10^4$, Relative Latency, L_h, begins to increase slightly. This is due to the fact that both numerator and denominator in L_h are functions of λ, and that single–proxy latency is increasing more slowly than URL–Hashing latency, thus increasing L_h.

Fig. 6 shows the effect of congestion on a system

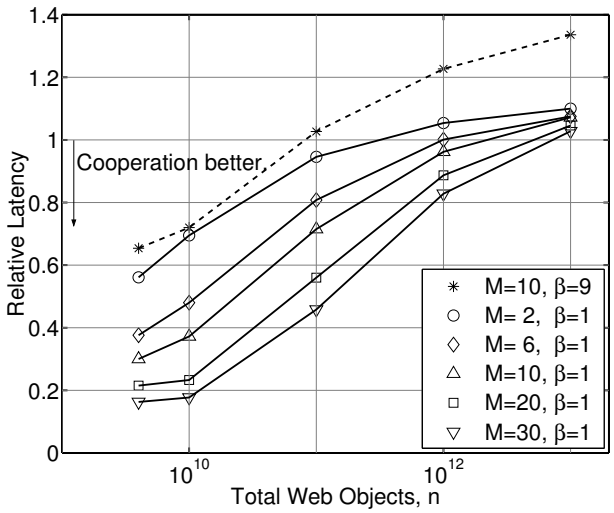

Figure 7. Effect of total number of Web objects for several numbers of cooperating proxies (URL–Hashing) under no congestion, and for 10 proxies under congestion factor $\beta = 9$—see dashed line.

of 10 proxies as the number of daily Web–object requests increases. For congestion factor $\beta \geq 6$, there exists a request rate at which Relative Latency reaches a minimum; for $\beta < 6$, Relative Latency monotonically decreases.

Fig. 7 shows the effect of increasing the number of cacheable objects, n, for different numbers of cooperating proxies having no congestion, and for 10 proxies under congestion level $\beta = 9$ (dashed line). For 10 proxies, the current value of $n = 4 \times 10^9$ results in a benefit for cooperation, but by the time n increases by a factor of 100, all benefits disappear with even minor congestion.

The total number of cacheable Web objects is currently estimated to be on the order of $n \approx 4 \times 10^9$,[6] but future Web growth will increase this number.[2] Benefits of cooperative Web caching will depend on the growth of this number vs. performance of future proxy servers. Using current proxy servers and values of $n \leq 10^{11}$ cooperation is beneficial ($M \leq 30$).

5. Validation of the Model

We used NS-2[23] to perform experiments on a network topology that mimics the one described by Fig. 1 and in Eqns. 9–14. We implemented Broadcasting and URL-Hashing in the NS-2 simulator. We used IRCache[12] tracefiles to drive the simulation. The caches were 6 GB with LRU (Least Recently Used) replacement and TTL (Time-To-Live) consistency policies. Hit rates and retrieval times obtained when the proxies are cooperating were respectively normalized by the hit rates and retrieval times of independent proxies, and the results were compared against the analytic model as shown in Figs. 8 and 9. The model parameters had to be adjusted to better fit the simulation results: λ and k were set to $4,000$ requests per day and $2,000$ due to the particular characteristics of the input tracefile, and P and U were set to 0.2 and 0.02 changes per day to account for the combined effects of finite cache size and the particular replacement and consistency policies.

In Fig. 8 The hit rates produced by the analytic model closely follow the data from the simulation experiments for a number of proxies higher than 10. Simulation data for URL and Broadcasting diverge with larger number of proxies as the effect of duplication of cached objects in Broadcasting becomes more important. Model and simulation produce similar retrieval times as shown in Fig. 9.

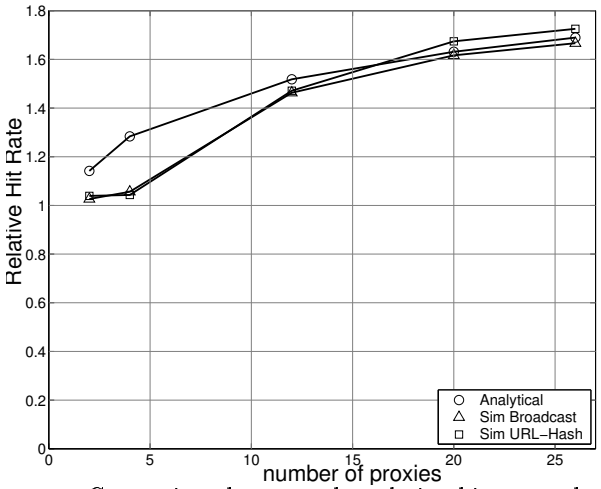

Figure 8. Comparison between the relative hit rates obtained from the analytic model, and the simulation (URL-Hashing and Broadcasting) .

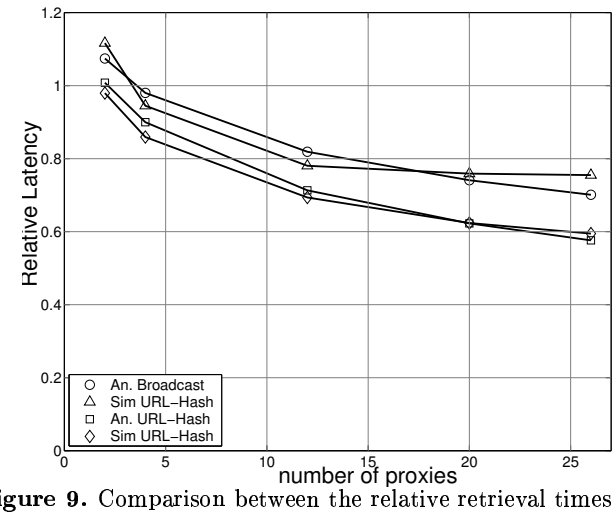

Figure 9. Comparison between the relative retrieval times obtained from the analytic model, and the simulation for URL-Hashing and Broadcasting .

6. Conclusions

We have presented an analytic model for comparing the benefits of Web proxy cooperation for different CMs as several parameters, including congestion, are varied. The model results lead to the following conclusions:

- Even though the model simplifies workload characteristics and network detailed behavior a good match is found between model results and detailed simulation experiments. When the model is used to describe a specific experiment it is required that the parameters of the model be adjusted to the particularities of the workload and cache sizes, replacement and consistency policies.

- Proxy cooperation overhead rises with increasing Web–object request rate and number of cooperating proxies. The challenge posed by increasing request rates can first be met by higher–performance proxy servers, but eventually, using a large number of proxies eliminates all advantages of cooperation. The reduction in cooperation benefits is exacerbated by proxy–link congestion.

- For the set of processing and latency parameter values used in this analysis, an upper bound of 30 cooperating proxies was found for Broadcasting, even with no congestion. For URL–Hashing, this number is higher due to its lower overhead.

- Moderate congestion levels ($\beta < 4$) are better managed with a large number of cooperative proxies ($20 \leq M \leq 30$). Higher congestion levels are better dealt with using a small number of proxies ($M < 6$) or with single–proxy operation.

- When no congestion is present, higher client–side demand rate increases cooperation benefits, due to the resulting improvement in cache hit rate.

The analytic model presented here will be useful for researchers who are exploring other CMs, or more generally, evaluating request re–direction methods for either supply–side (Web caching) or demand–side (content distribution networks). Our work highlights the implications of congestion for current and future Web environments. Actual CM designs focus on scalability. It would be desirable to simulate or prototype a CM whose request re–direction scheme is explicitly designed to adapt to congestion.

References:

1. C. Kenyon, "The evolution of Web–caching markets," Computer **34(11)**, November 2001.
2. L. G. Roberts, "Beyond Moore's Law: Internet growth trends," Computer **33(1)**, pp. 117–119, January 2000.
3. D. Karger et al., "Web caching with consistent hashing," Computer Networks **31(11–16)**, pp. 1203–13, May 1999.
4. M. Rabinovich and O. Spatscheck, Web Caching and Replication, Addison–Wesley, 2002.
5. D. Wessels and K. Claffy, Application of Internet Caching Protocol (ICP), September 1997. RFC2187 version 2.
6. A. Wolman et al., "On the scale and performance of cooperative Web proxy caching," Operating Systems Review **34(5)**, pp. 16–31, December 1999.

7. L. Breslau et al., "Web caching and Zipf-like distributions: Evidence and implications," in Proceedings, INFOCOM 1999, pp. 126–134, IEEE, March 1999.
8. D. S. Gadde, J. Chase, and M. Rabinovich, "Web caching and content distribution: a view from the interior," Computer Communications **24**, pp. 222–231, 2001.
9. E. de la Rosa, J. Hartman, and T. Hurst, "Analysis of RBQ: a new cooperative web caching mechanism that adapts to link congestion," Proceedings of SPIE ITCom: Performance and Control of Next-Generation Communications Networks , pp. 187–197, 2003.
10. M. W. Paper, "Scale-out caching with isa server." http://www.microsoft.com/isaserver/techinfo/deployment/scaleoutcachingwithisa.asp.
11. http://www.mirror–image.com.
12. IRCache, "Caching services project." http://www.ircache.net.
13. TERENA, "WWW cache coordination for Europe." http://www.terena.nl/tech/archive/tf–cache.
14. D. Wessels and K. Claffy, "ICP and the Squid Web cache," IEEE Journal on Selected Areas in Communications **16**, 1998.
15. S. Gadde, J. Chase, and M. Rabinovich, "A taste of crispy squid," in Workshop on Internet Server Performance, 1998.
16. L. Fan et al., "Summary Cache: A scalable wide-area Web cache sharing protocol," in Proceedings of ACM SIGCOMM, 1998.
17. D. Karger et al., "Consistent Hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 654–663, 1997.
18. G. Grimm, J. S. Voeckler, and H. Pralle, "Request routing in cache meshes," Computer Networks and ISDN Systems **30**, pp. 2269–2278, 1998.
19. M. Rabinovich, J. Chase, and S. Gadde, "Not all hits are created equal: Cooperative proxy caching over a wide–area newtork," Computer Networks and ISDN Systems **30**, pp. 2253–2259, 1998.
20. D. Wessels and K. Claffy, "ICP and the Squid Web cache," IEEE Journal on Selected Areas in Communications **16(3)**, April 1998.
21. S. G. Dykes and K. A. Robbins, "Limitations and benefits of cooperative proxy caching," IEEE Journal on Selected Areas in Communications **20(7)**, September 2002.
22. P. Rodriguez, C. Spanner, and E. Biersack, "Web caching architectures: Hierarchical and distributed caching," IEEE Transactions on Networking **9(4)**, August 2001.
23. NS, "Network simulator, available at." http://www.isi.edu/nsnam/ns/.