A Packet-Level Simulation Study of Optimal Web Proxy Cache Placement

Gwen Houtzager
Department of Computer Science
University of Calgary
Email: gwen@cpsc.ucalgary.ca

Carey Williamson
Department of Computer Science
University of Calgary
Email: carey@cpsc.ucalgary.ca

ABSTRACT

The Web proxy cache placement problem is often formulated as a classical optimization problem: place N proxies within an internetwork so as to minimize the average user response time for retrieving Web objects. Approaches to this problem in the literature include graph theoretic approaches, combinatorial approaches, dynamic programming approaches, and vector quantization approaches.

In this paper, we tackle the cache placement problem using packet-level ns2 network simulations. There are three main conclusions from our study. First, network-level effects (e.g., TCP dynamics, network congestion) can have a significant impact on user-level Web performance, and must not be overlooked when optimizing Web proxy cache placement. Second, cache filter effects can have a pronounced impact on the overall structure of an optimal caching solution. Third, small perturbations to the Web workload can produce quite different solutions for the optimal cache placement problem. This implies that robust solutions are more desirable than "optimal" solutions. The paper provides several heuristics for cache placement based on our packet-level simulations.

Keywords: Web Performance, Modeling and Simulation, Network Traffic Studies

1.0 Introduction

Given the explosion of Internet use in the last decade, much effort has been directed towards improving user perceived experience with the World Wide Web. One popular approach is the installation and implementation of Web caching appliances or Web proxy caches. Web caching can reduce and balance Internet traffic across a network, reduce user perceived latency when accessing Web documents, and improve server responsiveness by reducing server load.

Web proxy caches provide a shared cache to a set of clients [16]. When a user requests a Web document from a particular origin server, the request goes through the proxy. If the document is cached at the proxy and the document is not stale, then the user's request is served from the proxy in the same way as it would have been served had it been handled by the origin server itself. Alternatively, if the document is not in the proxy cache, or if the document is not up to date, then the proxy forwards the request to the origin server. The origin server responds to the proxy, and the proxy forwards the response to the client. The proxy typically stores the document in its cache to serve future requests for the same document from other clients. To the origin server, the proxy appears and acts as a client making a request. To the client, the proxy appears and acts as the origin server responding to a request.

The strategic placement of Web proxies in a network can yield a number of performance gains. When a user's request for a Web document is served by a Web caching appliance in close proximity to the client, the request need not travel to the origin server over numerous wide area network links. Specifically, the goal is to keep request/response traffic off of slower inter-continental links that can dramatically inflate client response times. Since round trip delay is reduced, the download of the Web document is faster for the user. Equally important, unnecessary network traffic is eliminated on high traffic Internet backbones.

The Web proxy cache placement problem is often formulated as a classical optimization problem: place *m* proxies within an internetwork so as to minimize the average user response time for retrieving Web objects. Approaches to this problem in the literature include graph theoretic approaches, combinatorial approaches, dynamic programming approaches, and vector quantization approaches [6][9][10][11][12][13].

One drawback of most theoretical approaches to the Web proxy cache placement problem is the limiting assumptions that are needed to make the analysis tractable. For example, some approaches assume fixed-size documents, identical hit ratios at each proxy cache, and homogeneous clients in terms of the number of Web requests generated. These assumptions are in stark contrast to empirical observations of Web workload characteristics: heavy-tailed transfer size distributions [1][5], diminishing hit ratios at each successive level of cache due to filter effects [4][14][19], and Zipf-like distributions for client activity and Web object popularity [2][5]. Furthermore, theoretical approaches often abstract away details about network protocol effects, such as bursty packet traffic, packet losses, and the dynamics of TCP flow control.

The purpose of this paper is to study the Web proxy cache placement problem from a packet-level network-layer perspective. The motivation for the packet-based approach is three-fold. First, the aforementioned discrepancies between empirical observations about Web proxy caching and the modeling assumptions often used motivate a detailed simulation approach with fewer unrealistic assumptions. Second, some of our own earlier simulation work [20] has shown the significant impacts of network-level effects (e.g., round trip times, link speeds, network congestion, packet losses, TCP dynamics) on user-level Web performance. We seek to build upon and extend these observations. Finally, first-hand experiences with two different commercial Web proxy caching

appliances have shown that performance is surprisingly sensitive to TCP dynamics and network configurations.

In this paper, we tackle the optimal Web proxy cache placement problem using simulation, with the ns2 network simulator [3]. In essence, we take a "brute force" approach to the problem, conducting a large number of simulations for a simple Web proxy caching environment, but with fairly realistic network and workload assumptions. The experiments focus on the mean Web response time for the simulated Web clients.

The primary research questions addressed in the paper are:

- What are the differences, if any, between the solutions for optimal Web proxy cache placement determined by theoretical (e.g., dynamic programming) approaches and simulation (e.g., packet-level) approaches?
- How sensitive is the "optimal" Web proxy cache solution to the assumptions made about client workload, network structure, and Web caching effectiveness?

The results from our simulation experiments illustrate three main points. First, network-level protocol effects can have a significant impact on user-level Web performance, and thus on the cache placement decision. Second, assumptions about Web cache filter effects can completely change the structure of the optimal solution determined. Third, surprisingly small perturbations to the Web workload can produce quite different solutions for the optimal cache placement problem. Together, these three observations imply that robust “good” solutions are more desirable than "optimal" solutions, especially with incomplete knowledge of Web workloads.

The rest of the paper is organized as follows. Section 2 provides some background on the cache placement problem, and briefly discusses prior work on this problem. Section 3 describes the experimental methodology for our packet-level simulation experiments. Section 4 provides a synopsis of results from a dynamic programming approach to the Web cache placement problem. Section 5 presents the main results from our packet-level simulation study. Section 6 extends the study by varying selected assumptions regarding caching, workload, and network-level structure. Finally, Section 7 concludes the paper.

2.0 Related Work

The placement of Web proxy caches (proxies) in a given network poses an interesting problem. Specifically, there is a theoretical, optimal solution to the placement problem where the number, size, and cost of Web caching appliances is minimized while the benefit of Web caching is maximized in terms of reducing user latency and bandwidth usage. In graph theory, this problem is referred to as the *k*-median problem, and has been proven to be NP-hard. As a result, current solutions to the problem rely on heuristic approaches and approximate models [9][10][11][12].

Li *et al* have studied the problem of optimal placement of multiple Web proxies among potential sites, given a certain traffic pattern. The reduction in overall network traffic and the reduction in access latency are used as measures of optimal placement in this case. Using a dynamic programming approach, they propose an optimal solution for linear network topologies, and a sub-optimal approximation for tree topologies [13]. Later, they determined an optimal solution to the distributed caching problem in an active network with a tree topology [12]. The latter paper provides the basis for the dynamic programming approach used as an example in our work.

Krishnan *et al*’s research also focuses on the problem of where to place caches within a network of a defined topology [11]. They consider the problem as it relates to general caches, transparent

en-route caches, and mirror placement for a single Web server. The goal of their experiments was to formulate an algorithm that minimized overall network traffic and reduced average client delay through the strategic placement of caches. Using a dynamic programming approach, they propose optimal algorithms for ring and line topologies, and a sub-optimal approximation for a tree network topology.

Krishnan *et al* then ran simulation experiments to measure network performance based on their findings. Results from the experiments demonstrated that determining the placement of Web caches is non-trivial, and did in fact influence the overall benefit. In addition, their proposed algorithms outperformed greedy algorithms in either network traffic reduction or fewer caches placed in the network. From a practical point of view, their research showed an "advantage over the common use of caches at the edges of the networks" [11].

What makes the proxy placement problem more complicated than other instances of the *k*-median problem is the existence of upstream and downstream dependencies when evaluating potential proxy site locations. A hierarchical caching architecture is based on multiple levels of caching within a network. If a document is not cached at one level, there is a chance it is cached at the level above. Typically, it is assumed that there are four levels: client browser caches, institutional caches, regional caches, and national-level caches. The latter refer to cache locations close to the Internet backbone or international links [18].

Empirical measurements show that the cache hit ratio tends to decrease at each successive level of cache [14]. This phenomenon is called the cache filter effect [19]. However, hierarchical schemes such as Harvest, adaptive Web caching, and access-driven caching can still be used to effectively "diffuse popular Web content, especially if cooperating cache servers do not have high-speed connectivity"[12]. The simulation experiments in this paper consider hierarchical caching and cache filter effects.

3.0 Experimental Methodology

3.1 Network Model and Assumptions

The network model assumed for our study is shown in Figure 1. The chosen topology represents a compromise: large enough so that the placement of proxies is not trivial, yet simple enough so that it can be easily simulated and the results understood. The network model has a single Web server at the root (top) of an unbalanced tree, with six clients at the leaf level. Each leaf node could represent the aggregation of numerous clients in a Local Area Network (LAN), however, for simplicity, only six abstract clients are considered in this study. The circular nodes, labeled P1 to P11, represent candidate proxy locations, at national, regional, and institutional levels. (Browser caches are ignored in our study.) The connecting lines in the figure represent routing paths from the server to the clients. Router nodes have buffers of size 100 packets with DropTail queueing. All network links have 10 Mbps transmission capacity. Link propagation delays are shown on the left margin of the diagram. The percentages beneath each client (square) indicate the proportion of the total Web request workload generated by each client.

By design, the network structure and the traffic workload are unbalanced and asymmetric. Given this specific network topology, the goal is to determine an optimal placement for a set of *m* Web proxies. The numerical values adjacent to each link in Figure 1 represent weights used by the dynamic programming approach discussed in Section 3.5.

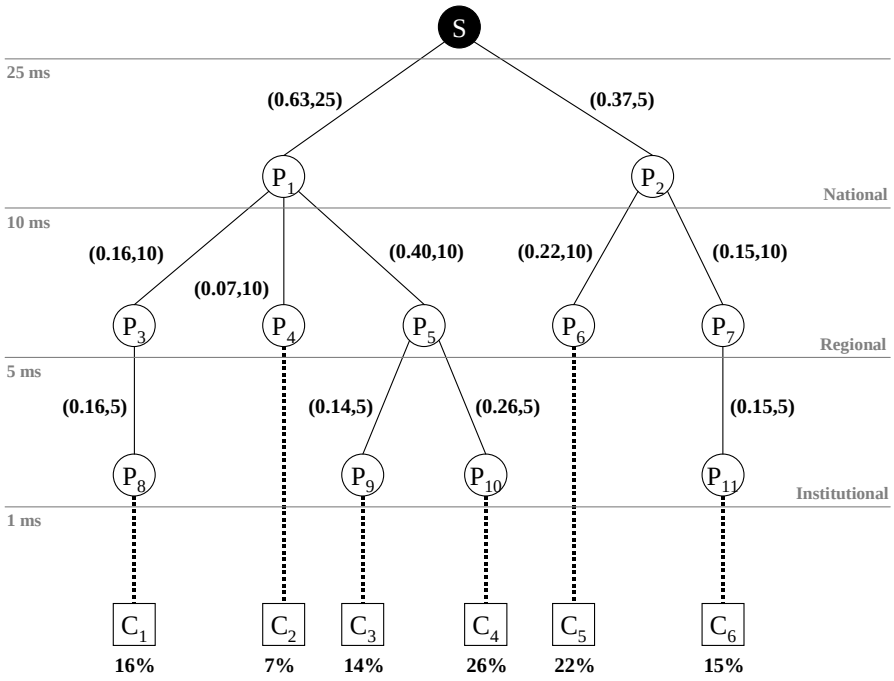


Figure 1 Network Topology and Workload Characteristics for Simulation Experiments

3.2 Web Workload Model

The Web workload in our study is synthetically generated using a version of WebTraff [15]. Each line in the workload file represents the download of a Web object by one of the simulated clients. The workload file format has four columns: a timestamp (request arrival time); a source node (provider of the Web object); a sink node (the requesting client); and a transfer size (in bytes). The request arrival process is Poisson, with a specified mean arrival rate (e.g., 30 requests per second). The transfer size is drawn from a hybrid distribution, with a log-normal body, and a Pareto tail. The median transfer size is 5 KB (11 TCP packets), while the mean is 11.5 KB (24 packets). The largest transfer is 453 KB, while the smallest is 88 bytes. Figure 2 shows a distribution of transfer sizes for the Web workload model. The source and sink for each Web transfer are chosen at random according to the desired request rate for each client and the cache hit ratio(s) being modeled. The default source for each Web transfer is the origin server.

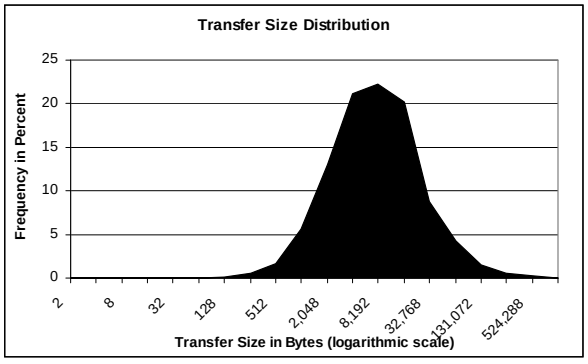


Figure 2: Web Workload Transfer Size Distribution for 2500 requests

The simulation experiments use a workload with 2500 requests. This relatively short trace length again represents a compromise: large enough to produce statistically useful results, yet short

enough to permit the large number of simulations (hundreds) needed in our study. With packet-level simulations in ns2, about 10 simulation runs can be completed per hour on departmental compute servers. We consider this trace length suitable for our purposes. Furthermore, the results with this trace length are consistent with those from test runs with 6000 requests.

3.3 Experimental Factors

There is one main *experimental factor* in our experiments: the number of proxies. This number is varied from 1 to 4. Its maximum possible value on our simulated network is 11. There are two main *workload factors* in our experiment: the request arrival rate, and the cache hit ratios at the proxies. The default request arrival rate is 30 requests per second, representing a moderate load of about 3 Mbps originating from the server. Given the stochastic arrival process, variable-size transfers, and TCP's bursty packet arrival process, the peak load can be much higher than this. The chosen cache hit ratios are shown in Table 1.

3.4 Performance Metrics

The primary performance metric is the overall mean response time for Web object downloads, which we call the *transfer time*. This value is computed across all clients. In some cases, we also mention the per-client mean transfer time, or the median transfer time. A secondary metric used in Section 6 is the level of packet loss in the network. By design, there are no packet losses in any of the experiments in Section 5.

3.5 Dynamic Programming Approach

Li *et al* present a proxy placement algorithm with time complexity $O(n^3m^3)$, where n is the number of nodes in the network and m is the number of proxies to be placed in the network. The authors observed a relationship among potential proxy sites with the surrounding links and paths in the network; namely that inserting a proxy at a downstream location affects the traffic pattern upstream of that location. Since the traffic pattern affects network latency, it is this observation on which their problem formulation is based. Their algorithm provides an optimal solution in terms of minimizing average latency. The authors provide justification of the algorithm's complexity and correctness [12].

The approach assumes that the network is a directed graph of vertices (or nodes) and edges. Each node has an associated weight that represents the volume of traffic expected to traverse the node in the no proxy scenario. In our network model, each client is assigned a weight relative to the percentage of overall traffic in the system. Each higher level node is assigned a weight representing the sum of the traffic from its children. Each node is also associated with an edge distance that can be "interpreted as either latency, link cost, hop count, or whatever" [12]. In our study, the edge distance represents the one-way link propagation delay. The cumulative cost from any downstream node u to an upstream node v is the sum of the edge distances. The weights and distances for each node are identified in Figure 1.

The dynamic programming approach in [12] then proceeds as follows. For each node v in the tree, there is a subtree T_v with v as its root. Note that for leaf nodes, the subtree contains only itself. Given any node u, it is possible to further partition the tree into a left node subset L_{uv}, a subtree T_u and the remainder set T_{uv} where:

$$L_{uv} = \{ x \in T_v : x \text{ is to the left of } u \}$$
$$T_{uv} = \{ x \in T_v : x \notin T_u \cup L_{uv} \}$$

Within the tree, a set of nodes must be chosen as proxy sites. Each proxy set solution has an associated cost and the set that minimizes the cost represents the optimal placement solution. The authors define $W(u, v)$ as the total contribution to the cost of a given proxy set in the set Luv, and $W(u, v, x)$ as the total contribution to the cost in the set Lxv, where:

$$W(u, v) = \sum_{x \in Lu,v} w(x)\, d(x, v)$$

$$W(u, v, x) = \sum_{y \in Lu,v,x} w(y)\, d(y, v)$$

Furthermore, $C(v,m)$ represents the optimal cost of placing m proxies in T_v and $C(u,v,m)$ represents the optimal cost of placing m proxies in T_{uv}. These are defined as follows[1]:

$$C(u, v, m) = \begin{cases} \sum_{x \in Tuv} w(x)\, d(x, v) & \text{if } m = 1 \\ \min_{x \in Tuv} \min_{x0<m'<m} & \\ (W(u, v, x) + C(x, m') + C(v, x, m - m') & \text{if } m > 1 \end{cases}$$

$$C(v, m) = \begin{cases} \sum_{x \in Tv} w(x)\, d(x, v) & \text{if } m = 1 \\ \min_{x \in Tv} \min_{x0<m'<m} & \\ (W(u, v) + C(x, m') + C(v, x, m - m') & \text{if } m > 1 \end{cases}$$

The static values for $W(u, v)$ and $W(u, v, x)$ can be pre-computed for all possible combinations of u, v and x, and stored in 2D and 3D arrays, respectively. Next, $C(v, m)$ and $C(u, v, m)$, along with their 'selected proxy sets' can be calculated according to the equations mentioned earlier. Further, if they are calculated in order of increasing m values and stored in another table, each equation processes entries that have been previously calculated. This means that the table is populated in a structured order. Each stage represents a solution for a given set of m proxies for all nodes n.

3.3 Simulation Approach

Our simulation study uses a "brute force" combinatorial approach to determine the optimal placement of proxies. It is simply an exhaustive search of all possible proxy set combinations. With n potential proxy sites, the number of ways to place m proxies is:

$$C(n,m) = \frac{n!}{m!\,(n-m)!}$$

For example, choosing a location for 1 proxy in the 11 node network of Figure 1, the expression is C(11,1) = 11!/1!10! = 11. For 2 proxies, the expression is C(11,2) = 11!/2!9! = 55. For more proxies, C(11,3) = 165, C(11,4) = 330, and so on. While this combinatorial search guarantees an optimal solution to the proxy placement problem (since all possibilities are tested), the effort required becomes prohibitive as the network and proxy set size grow.

All possible proxy set combinations up to size 4 were simulated using the ns2 network simulator. ns2 is a public-domain network simulator originating from Lawrence Berkeley Labs. It is widely used by networking researchers. It contains several TCP protocol models, and supports network animation for the visualization of network topology and network traffic dynamics [3].

[1] By default, the root (server) node is considered a proxy site, so the base case is $m = 1$.

In this study, each simulation consists of 2500 Web TCP transfers flowing uni-directionally from a source (server or proxy) node to a sink (one of the 6 clients). Network capacity and buffer size remained fixed, while Web workload characteristics and cache hit ratios are varied.

The desired cache hit ratios are achieved by carefully manipulating the input client workloads. For any client request, the request could be served from any one of the proxy nodes on the path to the server. If no proxies exist in the network, then every request involves the server. However, if a proxy on a client's path is part of the chosen proxy set, then with a probability equal to the cache hit ratio, a request from that client is served from the proxy instead of the server.

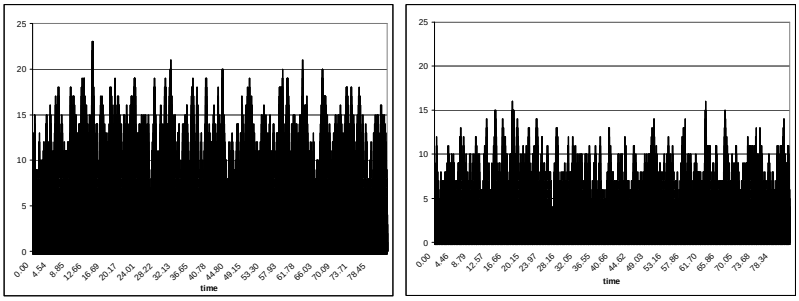
Cache hit ratios were chosen to represent the "diminishing returns" and cache filter effects reported in the literature [14][18][19]. Since higher level proxies serve more diverse client groups using a shared (finite) resource, the likelihood of an individual client finding a requested document at a higher level proxy drops. Cache hit ratios were assigned consistently across network caching levels for each client stream according to Table 1.

Table 1: Proxy Cache Hit Ratios

number of proxies	***Cache Hit Ratio***		
	level 1	*level 2*	*level 3*
1	30%	-	-
1	-	20%	-
1	-	-	15%
2	30%	15%	-
2	30%	-	10%
2	-	20%	10%
3	30%	15%	7.5%

Finally, we tested to ensure that the simulation results represent "steady state" performance. Specifically, it was important to choose a request arrival rate that produces a reasonable number of simultaneously active TCP connections within the simulated network. An arrival rate that is too low would produce no observable network congestion. An arrival rate that is too high would saturate the network and result in poor simulator performance.

Figure 3 provides two time series plots showing the number of simultaneous connections from two simulation runs. The "No Proxy" scenario is the worst case when the server handles all client requests. The "11 Proxies" represents the best case scenario (11 proxies in total, with the cache hit ratios prescribed earlier). These scenarios provide an upper bound and a lower bound on the mean transfer time in our simulation experiments. Both scenarios assume an arrival rate of 30 requests per second, with a mean transfer size of 11.5 KB and median transfer size of 5 KB.



(a) No Proxy (Worst Case)
Average:11.35 connections

(b) 11 Proxies (Best Case)
Average: 6.82 connections

Figure 3: Simulation "Steady-State" for Number of Simultaneous TCP Connections

4.0 Results: Dynamic Programming

4.1 Overview of Results

The results from the dynamic programming approach are shown graphically in Figure 4. The solution for m = 3 proxies is on the left, while the m = 4 proxy solution is on the right. The diagrams show the optimal proxy placement, while the tables show the relevant portion of the dynamic programming algorithm results.

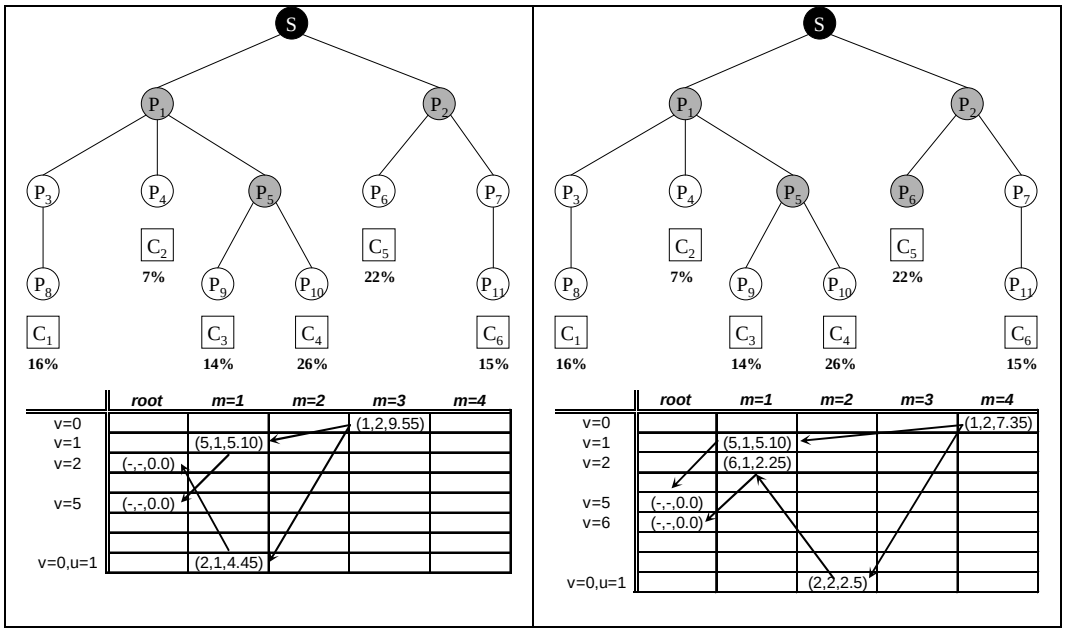


Figure 4: Dynamic Programming Proxy Sets and Tables for m=3 and m=4

The outcome of the dynamic programming algorithm is a table that can be read in reverse order to find an optimal proxy solution set for the subtree rooted at each cell in the table. Each cell in the table contains a *3*-tuple. The first number identifies the choice for the next optimal proxy location, the second number identifies the subtree from which the remaining proxy selections are chosen, and the third number represents an associated cost. For example, beginning at the top right corner for the m=4 table, the tuple reads (1,2,7.35). In this cell, we are concerned with choosing 4 proxies within the tree T_0 rooted at node 0 – which in this case happens to be the complete network since node 0 is the server. One of the 4 optimal proxy sites was evaluated to be node 1 as identified by the first number of the tuple. The second number of the tuple indicates that 2 proxies will be chosen from the subtree T_1 rooted at node 1, including node 1 itself. This means that we are interested in finding m=1 proxies under the tree at T_1. It follows that the other 2 proxies will be chosen from the subtree T_{01} (the nodes that are to the right of T_1). The tuple at T_1 for m=1 reads (5,1,5.10), indicating that node 5 has been selected as the single proxy site contained in this subtree. The tuple at T_{01} where m=2 reads (2,2,2.5). The first number of this tuple indicates that node 2 is chosen as a proxy site and the second number of the tuple indicates that another proxy will be chosen from the subtree rooted at node 2. Finally, the tuple at T_2 for m=1 reads (6,1,2.25), indicating that node 6 has been selected as the last proxy location.

The optimal placement for a single proxy in the network, according to the dynamic programming algorithm, is at site P1. The optimal placement for m=2 proxies is at site P1 and P2. For the latter result (as well as for m=3 and m=4 scenarios), all clients are directly affected by the

inclusion of a proxy in the network since at least one of these locations is contained in all the client paths to the server. A summary of relevant results and a comparison with the upcoming packet-level simulation results is provided in Table 2.

Table 2: Comparison of Dynamic Programming and Packet-Level Simulation Approaches

# of Proxies	Optimal Proxy Set		Reduction from Worst Case		Mean Transfer Time		*Difference*
	Dynamic	*Simulation*	*Dynamic*	*Simulation*	*Dynamic*	*Simulation*	
1	1	10	6.1%	7.5%	0.337	0.332	0.005
2	1,2	6,10	10.2%	13.4%	0.323	0.311	0.012
3	1,2,5	5,6,10	15.6%	19.2%	0.303	0.290	0.013
4	1,2,5,6	5,6,8,10	20.8%	24.4%	0.284	0.271	0.013

4.2 Discussion

The main observation is that with few proxies, the dynamic programming algorithm seems to favour proxy placement near the origin server. Here, the proxy location selection suggests a tendency to initially choose sites that serve a collection of clients and remove traffic from the busiest network links. A secondary influence appears to favour the placement of proxies in subtrees that reflect the highest traffic generating clients (in this network, clients 4 and 5). This observation is not surprising considering that these subtrees are responsible for generating most of the traffic on upstream channels. Interestingly, these tendencies differ from those in Section 5.

5.0 Results: Packet-Level Simulation

5.1 Overview of Results

The results from the packet-level simulation experiments are shown graphically in Figure 5. From top to bottom, these graphs represent the results for $m = 1$, $m = 2$, $m = 3$, and $m = 4$ proxies, respectively. In each row of the figure, the diagram on the left shows the optimal proxy placement determined from the simulation, while the histogram on the right shows the distribution of the mean transfer times from all proxy placements considered. In each histogram plot, the leftmost bar represents the optimal proxy placement found, while the rightmost bar shows the mean transfer time for the "No Proxy" case.

Table 3 provides a statistical summary of the transfer time performance results for the No Proxy case. Table 4 provides an example of the simulation results for $m = 1$ proxy, with results for locations P1, P5, and P10, respectively.

Table 3: Statistical Summary of Simulation Results for the "No Proxy" Case

Client	*# of Transfers*	*Transfer Size*		*Transfer Time*	
		mean	*median*	*mean*	*median*
1	400	14076	5819	0.397	0.337
2	175	11560	4769	0.327	0.294
3	350	10837	5176	0.366	0.336
4	650	10194	4939	0.358	0.336
5	550	10757	5520	0.322	0.295
6	375	13249	4745	0.382	0.336
ALL	2500	11583	5141	0.359	0.336

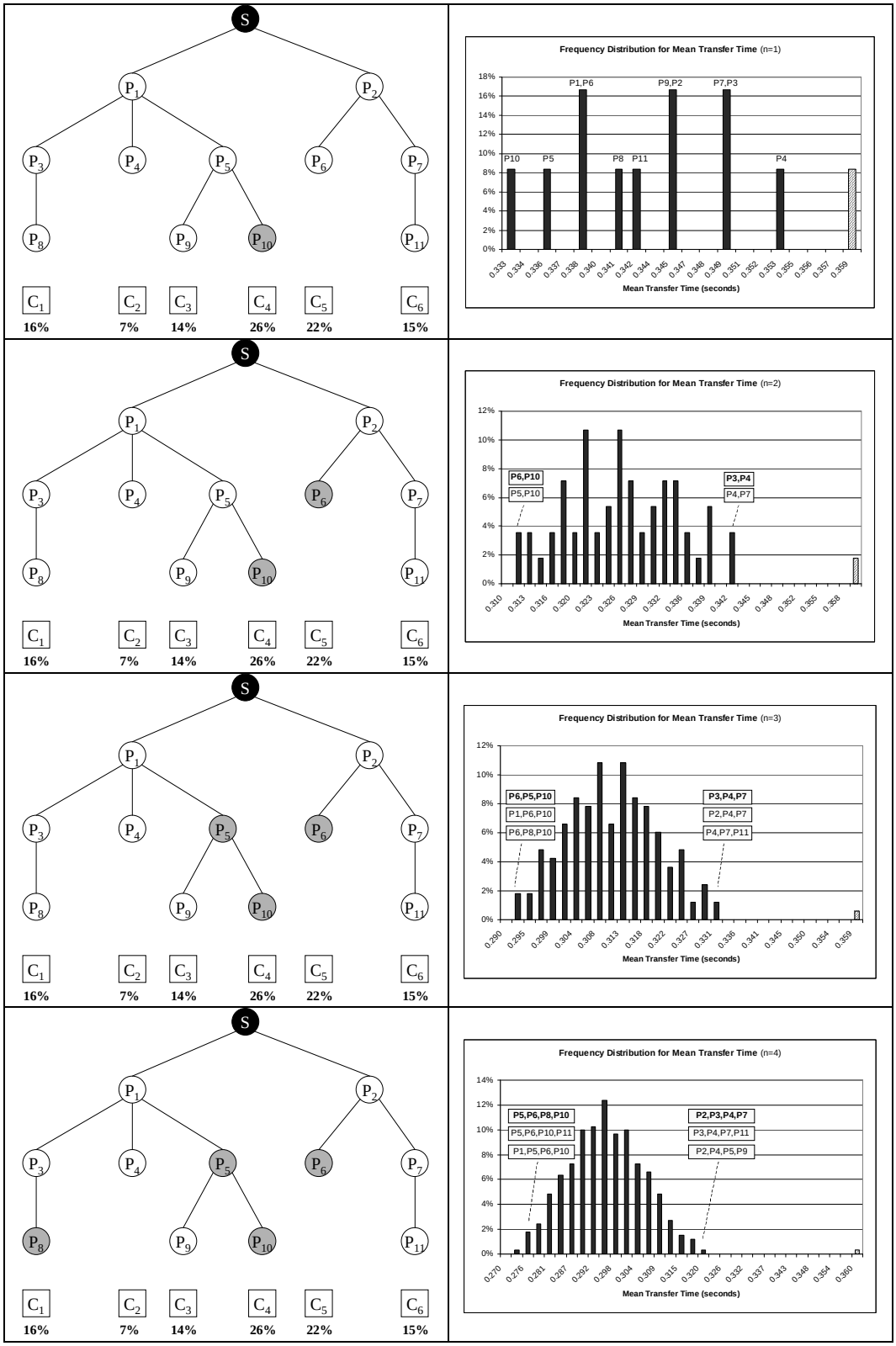


Figure 5: Packet-Level Simulation Results for m=1,2,3 and 4 Proxies

Table 4: Statistical Summary of Simulation Results for a Single Proxy at P1, P5, or P10

				Proxy P1		Proxy P5		Proxy P10	
	# of	Transfer Size		Transfer Time		Transfer Size		Transfer Time	
Client	Transfers	mean	median	mean	median	mean	median	mean	median
1	400	14196	5782	0.360	0.336	0.397	0.337	0.397	0.337
2	175	11199	4732	0.280	0.223	0.327	0.294	0.327	0.294
3	350	10905	5236	0.334	0.335	0.307	0.335	0.366	0.336
4	650	10196	4911	0.327	0.335	0.299	0.256	0.255	0.252
5	550	10903	5567	0.322	0.295	0.322	0.295	0.322	0.295
6	364	13210	4727	0.382	0.336	0.382	0.336	0.382	0.336
ALL	2489	11606	11606	0.337	0.335	0.336	0.335	0.332	0.335

5.2 Discussion

There are three main observations evident from our packet-level simulation results:

- The optimal placement of proxies tends to be close to the clients generating the most traffic. This placement makes sense intuitively, since it provides more "bang for the buck". The cache hit ratio is higher closer to clients; cache hits provide a dramatic improvement in mean transfer time for the busy client; and cache hits alleviate the traffic demand on higher-level links in the network, indirectly benefiting other clients in the same subtree. For example, placing a proxy at P10 reduces the mean transfer time for client 4 from 0.358 seconds to 0.255 seconds, an improvement of 29%, while the overall mean improves by 7.5%. These are exactly the benefits that motivated Web proxy caching in the first place.

- The optimal solutions in the packet-level simulations have a "monotonic" or "incremental" property. That is, when placing the k^{th} proxy, the first k-1 proxies are already in the "right" place. There is no reshuffling of proxy locations as the number of proxies is varied. This property may be an artifact of our workload assumptions and small-scale network model.

- As the size of the proxy set grows, the distribution of mean transfer times across the set of proxy configurations considered seems to converge to a Normal (Gaussian) distribution. This property is surprising (at least to us). We do not yet have an explanation for this behaviour.

The other obvious observation is that the proxy solution set determined by the dynamic programming (DP) algorithm does not correspond to the optimal solution set observed under our simulation experiments. One possible explanation for the discrepancy might be that the combination of propagation delay costs along with high network traffic weights assigned to higher layer nodes unfairly biases the DP algorithm towards off-loading traffic from the higher level network links. In this study, we consider a closed system in which all network traffic represents communication between a single Web server and six clients. The effect of a closed system may create unusual link costs with increasing network levels.

Another explanation for the contradiction in dynamic programming and simulation results is that the dynamic programming method takes into consideration only network traffic characteristics and link transmission latency. By contrast, it does not take into consideration network influences such as cache hit ratios, cache filtering effects, and TCP dynamics which are modeled in the packet-level simulations.

6.0 Results: Sensitivity Analysis

To broaden the scope of our packet-level simulation experiments, we individually varied selected workload factors in an attempt to understand sensitivities in the results. The factors studied are cache hit ratio, transfer size distribution, filter effect, request arrival rate, and network topology.

6.1 Effect of Cache Hit Ratio

The first experiment was originally designed to study the impact of cache consistency issues on Web proxy cache performance. In particular, when a Web object is modified, a requesting client must retrieve a fresh copy of that object from the origin server, rather than retrieving a stale copy from a proxy cache. We modeled this in our workload by choosing, uniformly at random, 10% of the (former) proxy cache hits and changing them into transfers from the origin server (though with the same object size). In retrospect, this "document modification" scenario is logically equivalent to a lower average cache hit ratio for all proxies.

These experiments were conducted only for $m = 1$ and $m = 2$ proxies cases. Although the mean transfer times for these experiments increased slightly, the simulation results were qualitatively similar to those stated in Section 5, and thus are not shown here. The results suggest that our observations are not highly sensitive to the cache hit ratio, at least over the range evaluated.

6.2 Transfer Size Distribution

A second experiment simplified our workload model to use fixed size Web objects. Every Web transfer was 11,583 bytes, so that the overall average network load was the same as in the previous experiments. These experiments were conducted only for $m = 1$ proxy.

The results from this experiment did not change from those in Section 5. The optimal placement for a single proxy was still P10, the nearest proxy location to the busiest Web client. The mean transfer time for the P10 proxy case was 0.376 seconds, a 7.8% improvement over the no proxy case (0.408 seconds) for this scenario.

6.3 Cache Filter Effect

The third experiment changed our assumption about the Web cache filter effect. Rather than modeling a diminishing cache hit ratio at progressively higher-layer Web proxy caches, we made the simpler assumption that the cache hit ratio is invariant. That is, a Web proxy cache is equally effective no matter where it is placed in the overall network. We simulated this scenario for $m = 1$ and $m = 2$ proxies.

Changing the filter effect assumption dramatically changed the simulation results (see Figure 6). Under the "equal effectiveness" assumption, the optimal proxy locations determined by the packet-level simulations are at or near the top of the modeled network. This placement makes sense intuitively, since (for example) a single proxy can serve the largest possible number of clients. The packet-level simulations show that Web cache filter effects can have a significant impact on the optimal cache placement.

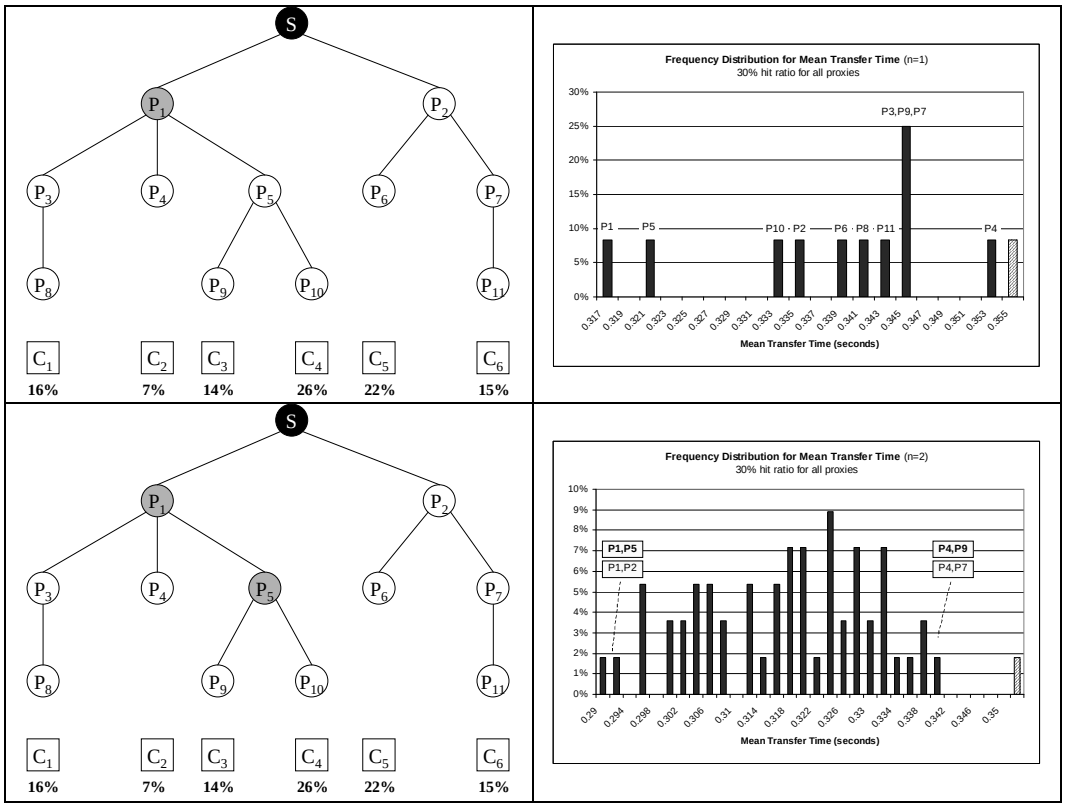


Figure 6: Simulation Results for m=1 and m = 2 Proxies with No Cache Filter Effect

6.4 Request Arrival Rate

The simulation results in Section 5 represent a moderate level of network load, with about 20% average utilization on the busiest link in the network. Under this workload, no packet losses occur in the simulation. The transfer times for Web objects are dominated by TCP slow start effects and network round trip times, with modest influence from queuing delays at the network routers.

The next experiment increases the request arrival rate to model a busier network, in which network congestion and packet loss occur. Increasing the request arrival rate to 120 requests per second produces an average of 80% utilization on the busiest link in the network, from the Web server to Proxy P1. An average of 44 TCP connections are simultaneously active in the network in steady state. We consider only $m = 1$ proxy.

With the high load workload assumption, the optimal proxy placement is at P1 (see Figure 7). The explanation for this placement requires an understanding of the effects of TCP packet loss. For this particular workload, a total of 65 packets are lost and retransmitted in the "No Proxy" case. This represents an average packet loss rate of 65/37026 = 0.18% for the busiest link in the network. This is the only link on which packet losses occur.

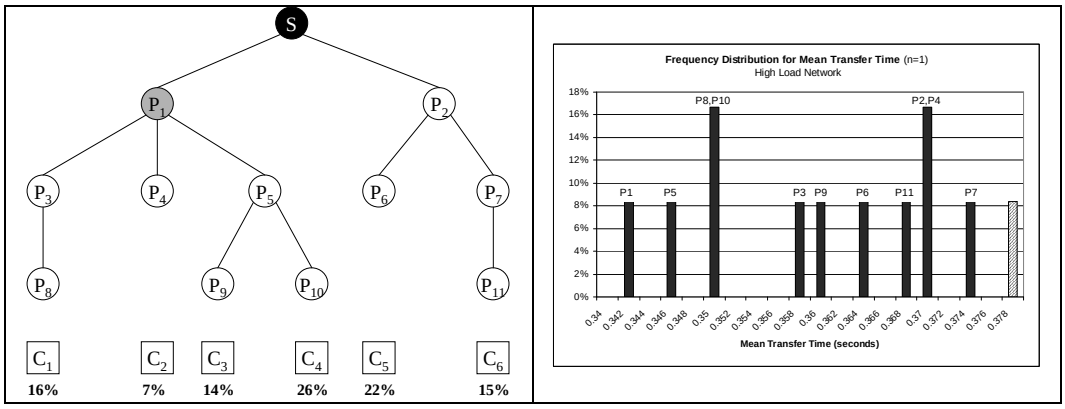


Figure 7: Simulation Results for m=1 in High Load Scenario

TCP packet losses can have two possible impacts on a Web object transfer. In a large transfer, a TCP packet loss is often detected using the TCP "fast retransmit" algorithm, wherein duplicate acknowledgements quickly trigger retransmission of the (single) missing packet. Recovery from these losses is quite "painless" to a Web user, since no TCP timeout occurs. The second possibility is a (coarse) TCP timeout and retransmission. This effect often adds significant latency to a TCP transfer, particularly for a short transfer [21].

The optimal placement of the proxy at P1 is related to the dynamics of TCP packet loss recovery. With a proxy at P1, approximately 15% of the workload is offloaded from the busy Server ←→ P1 link, reducing packet loss due to buffer overflow at the busy link. In our simulations, placing a single proxy at P1 reduces the packet loss to 13 packets. A single proxy at P5 results in 44 packet losses, while a single proxy at P10 results in 24 packet losses. The number of lost packets is highly dependent upon which objects are cache hits at the proxy, as well as the dynamics of TCP packet arrivals at the congested router. It is difficult to capture the detailed dynamics of these TCP effects in anything but a packet-level model of the Web cache placement problem.

6.5 Network Topology

The final experiment considers a small change to the network topology, wherein the Web server is relocated behind a router that in turn connects to P1 and P2. The side-effect of this small topology change is to combine the (former) Server ←→P1 traffic and the (former) Server←→P2 traffic onto a single bottleneck link (Server ←→ Router). In this topology, the impact of a network bottleneck is more acute, both in traffic volume and in the number of clients affected. For this experiment, we consider only the m = 1 proxy case.

At an arrival rate of 90 requests per second (average 90% utilization of the bottleneck link), the average packet loss rate was 690/60,085 = 1.15% on the bottleneck link. At this level of packet loss, many Web transfers were affected, for all clients. Some clients experienced losses both for an initial TCP packet as well as for its retransmission, which had a large impact on the transfer latency. While the median transfer times were similar, the mean transfer times varied significantly across proxy locations, and across clients.

The simulation results show that the optimal solution is to place a proxy at P5 (mean transfer time 0.401 seconds; 297 packet losses). This is a 20% improvement over the No Proxy case for this scenario (0.508 seconds). The next best solutions for a single proxy are at P8 (0.407 seconds; 284

packet losses) or P1 (0.411 seconds; 339 losses). Clearly, the dynamics of TCP behaviour can wreak havoc with determining "optimal" proxy cache placement.

7.0 Summary and Conclusions

This paper has presented a detailed packet-level simulation study of the optimal Web proxy cache placement problem. On a small and simple network model, we considered an exhaustive set of Web proxy cache configurations, using assumptions reflecting empirical Web workloads (e.g., unbalanced network load, variable size Web objects, heterogeneous client round trip times, and the presence of cache filter effects).

There are three main conclusions from our study. First, network-level effects (e.g., queueing delays, network congestion, packet losses, and the dynamics of TCP flow and congestion control) can have a significant impact on user-level Web performance, and thus on the cache placement decision. These effects are often ignored in classical approaches to the cache placement problem. Second, assumptions about Web cache filter effects can completely change the structure of the optimal solution determined. In fact, the presence or absence of the filtering assumption can completely reverse the overall structure of an optimal solution, for a small number of proxies. Third, relatively small perturbations to the Web workload can produce quite different solutions for the optimal cache placement problem.

Combined, these three observations suggest that robust “good” solutions are more desirable than perfectly "optimal" solutions. Robust heuristic solutions are certainly preferable, given the many uncertainties about Web workloads and TCP dynamics in a large internetwork.

Two cache placement heuristics seem to emerge from our packet-level simulation study (though these no doubt depend on our network and workload assumptions). First, for moderate network load, proxy caches should be placed close to clients generating the most traffic. This approach provides the largest possible benefit to a small number of clients, but also indirectly benefits other users of the network as well. Second, in a heavily loaded or congested network, proxy caches should be placed well up in the hierarchy to offload the congested links, reducing packet losses, and helping as many clients as possible.

Our ongoing work is considering larger Web workloads and more realistic network models, to add more statistical rigour to our observations, and to assess the generality of our conclusions. These additional results should be ready for the final version of the paper. Future work may consider radically different approaches to the Web proxy cache placement problem, such as evolutionary or genetic programming techniques, based on the heuristics and insights developed in this work.

8.0 References

[1] Arlitt, M., Williamson, C. *Internet Web Servers: Workload Characterization and Performance Implications*. IEEE/ACM Transactions on Networking, Vol. 5, No. 5, pp. 631-645 (October 1997).

[2] Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. *Web Caching and Zipf-like Distributions: Evidence and Implications*. Proceedings of IEEE INFOCOM, pp.126-134 (March 1999).

[3] Breslau, L., et al. *Advances in Network Simulation*. IEEE Computer, Vol. 33, No. 5, pp. 59-67 (May 2000).

[4] Busari, M., Williamson, C. *Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy*. Proceedings of IEEE MASCOTS, pp. 379-388 (2001).

[5] Busari, M., Williamson, C. *On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics.* Proceedings of IEEE INFOCOM 2001, Vol 3, pp. 1225-1234 (2001).

[6] Cameron, C., Low, S., Wei, D. High Density Model for Server Allocation and Placement. Proceedings of ACM SIGMETRICS 2002, pp. 152-159 (June 2002).

[7] Cormen, T., Leiserson, C., Rivest, R., Stein, C. *Introduction to Algorithms*. The MIT Press, Second Edition (2001).

[8] Grimaldi, R. *Discrete and Combinatorial Mathematics: An Applied Introduction*. The Addison-Wesley Publishing, Third Edition (1994).

[9] Jia, X., Li, D., Hu, X., Huang, H., Du, D. *Optimal Placement of Proxies of Replicated Web Servers in the Internet*. Proceedings of the First International Conference on Web Information Systems Engineering (WISE), Volume 1, pp. 55-59 (2000).

[10] Jia, X., Li, D., Hu, X. *Placement of Read-Write Web Proxies in the Internet*. IEEE, pp.687-690 (2001).

[11] Krishnan, P., Raz, D., Shavitt, Y. *The Cache Location Problem*. ACM Transactions on Networking, Vol. 8, No. 5, pp. 568-582 (October 2000).

[12] Li B., Golin, M., Italiano, G., Deng, X., Sohraby, K. *On the Optimal Placement of Web Proxies in the Internet*. Proceedings of IEEE Infocom, v3, pp. 1282-1290 (March 1999).

[13] Li B., Golin, G., Deng, X., Sohraby, K. *On the Optimal Placement of Web Proxies in the Internet: Linear Topology*. 8th IFIP Conference on High Performance Networking (September 1998).

[14] Mahanti, A., Williamson, C., Eager, D. *Traffic Analysis of a Web Proxy Caching Hierarchy*. IEEE Network Vol 14, Issue 3, pp. 16-23 (May/June 2000).

[15] Markatchev, N., Williamson, C. *WebTraff: A GUI for Web Proxy Cache Workload Modeling and Analysis*. Proceedings of IEEE/ACM MASCOTS, pp. 356-363 (October 2002).

[16] Rabinovich, M., Spatscheck, O. *Web Caching and Replication*. Addison-Wesley, (2001).

[17] Raunak, M., Shenoy, P., Goyalf, P., Ramamritham, K. *Implications of Proxy Caching for Provisioning Networks and Servers*. Proceedings of ACM SIGMETRICS (2000).

[18] Rodriguez, P., Spanner, C., Biersack, E. *Web Caching Architectures: Hierarchical and Distributed Caching*. Proceedings of the 4th International Web Caching Workshop , pp 37-48 (April 1999).

[19] Williamson, C. *On Filter Effects in Web Caching Hierarchies*. ACM Transactions on Internet Technology, Vol. 2, No. 1, pp. 47-77 (February 2002).

[20] Williamson, C., Markatchev, N. *Network-Level Impacts on User-Level Web Performance*. Proceedings of SCS SPECTS 2003, Montreal, Canada (July 2002). To appear.

[21] Williamson, C., Q. Wu. *A Case for Context-Aware TCP/IP*. ACM Performance Evaluation Review, Vol. 29, No. 4, pp. 11-23 (March 2002).