

Guiding Web Proxy and Server Placement for High Performance Internet Content Delivery¹

Peter Triantafyllou

(contact author)

Department of Computer Engineering and Informatics,

University of Patras

Rio Patras, 26500, Greece

peter@ceid.upatras.gr

Telephone: +302610996913

FAX:+302610969011

Ioannis Aekaterinidis

Department of Computer Engineering and Informatics,

University of Patras, Greece

aikater@ceid.upatras.gr

¹ This work was supported by the European Commission under Project IST/FET DBGLOBE, no. IST-2001-32645.

Guiding Web Proxy and Server Placement for High Performance Internet Content Delivery

Peter Triantafyllou and Ioannis Aekaterinidis

Department of Computer Engineering and Informatics, University of Patras
{peter,aikater}@ceid.upatras.gr

Abstract

The performance of a network designed for efficient web content delivery largely depends on the strategic placement of (forward or reverse) web proxies and/or full-server replicas of web origin servers. This paper presents a tool, coined ProxyTeller, which we developed specifically for guiding such placement decisions. ProxyTeller decides on the position and the number of proxies and/or full server replicas required in order to achieve given performance constraints, which are expressed as percentage improvements in terms of network bandwidth, origin server load, and user-latency. ProxyTeller would be very helpful to ISPs/CDNs, content providers, and end-users and is, thus, very much lacking. We overview the design and implementation of ProxyTeller and present scenarios of its use and related results. ProxyTeller is available through the internet and our hope is that it will prove to be of real use to the community.

Keywords: [C.4.d] Modeling techniques, [H.3.5.e] Web-based services, [H.3.m] Miscellaneous, [J.m] Miscellaneous, Web Proxies, Web Replicas, Content Delivery Networks, Web Caching, Performance

1. Introduction

Web proxy servers are categorized into forward or reverse proxies (the latter are also called *surrogate servers*). The former can be found in ISP network backbone nodes, intercepting all web traffic. The latter are found in CDN servers being contacted only for content requests belonging to origin servers, which have specific business agreements with the CDN. Cached content in reverse proxies is under the explicit control of the content provider. Despite these differences, both proxy types perform fundamentally the same efficiency-boosting tasks and can prove very beneficial for the systems' efficiency.

ProxyTeller is a software tool guiding the placement of proxies (and full server replicas) [1]. The problem *ProxyTeller* solves accepts as inputs the network topology over which proxies will be placed, the desired sizes of their caches, the characteristics of the workload

of the community served by a proxy (i.e., representative workload traces), the performance metric of the proxy and the performance improvement that is desired to be achieved. The latter can involve one of the metrics of network bandwidth and network latency and is specified in terms of the percentage improvement that is desired to be achieved, compared to the case where no proxies are deployed. The output of *ProxyTeller* is the position and the number of proxies of web origin server(s) required in order to achieve the stated performance goals as well as the comparative results of full-server replica placements under the same performance constraints.

It should be clear that to meet the challenges inherent in the development of *ProxyTeller* we must be able to:

- (a) define the performance improvement that is the result of introducing a proxy at a specific position within the network, and
- (b) employ efficient proxy placement algorithms and measure their performance impact.

During the past several years a great wealth of knowledge on efficient web proxy caching ([2],[3],[4],[5],[6]) has been accumulated. In order to be able to build powerful web proxies and understand their performance, one must be able to appreciate the true impact and significance of various components and how they can be integrated. We undertook a study for this task [7]. With our results on proxy cache performance we were able to chart the problem's solution space identifying which algorithm performs better under different circumstances and by how much. The findings of this study play a fundamental role within *ProxyTeller*, since on the one hand it 'advises' the tool with respect to the best algorithm to use and, on the other, it quantifies the performance improvement with respect to the performance metrics of interest.

Another key issue is that of the suitable placement of server replicas in nodal points of the network. The server replica placement problem can also be viewed as the *facility location problem (FLP)*[8].

We concentrate on the efficient placement of forward or reverse proxy caches. Our approach is different than the ones in [9],[10]: instead of studying content placement, we study the placement of proxy caches, with given sizes. The functionality of each proxy cache is independent; it can use its own replacement algorithm

and mechanisms, utilizing the results of our first study [7], effectively harnessing the wealth of knowledge accumulated to decide which objects should be locally cached/replicated.

For obvious reasons, studying proxy (in essence, partial replicas) placement as opposed to full-server replica placement, is of vital importance for both ISPs/CDNs, content providers, and consumers, since the capital investment need for proxy caches is far smaller than that needed for full-server replicas. Using the results from our study on proxy-cache performance [7], we undertook a second study to define the performance impact of adapting the existing server-replica placement algorithms for proxy cache placement.

A demo of ProxyTeller is available through the internet [1]. Our aim is to make the source and executable code, as well as detailed results from the two constituent studies, freely available to the community.

2. Proxy placement

2.1 Study setup

We model the network under which the proxy placement takes place as a weighted graph. The nodes are connected with weighted edges that represent the time (in msec) that is required in order to transmit a packet of information from one node to another. Therefore, the edge weights can show how loaded are the links.

Nodes also have weight. The node weight represents the quantity of information that all users connected to this node request from the web origin server. Node weights can be derived by appropriately processing a given trace file which represents the expected workload of the web origin server.

The proxy placement algorithm will use edge and node weights, the network topology information and the performance tables of proxy caches (that shows which replacement algorithm performs better under which circumstances), in order to decide where and how many proxy servers to place.

2.1.1 Proxy server performance estimation. Apart from node weights, edge weights, and the network topology information, the performance impact of proxy servers plays an important role. The Byte Hit Ratio achieved by the selected cache algorithm, as it depends on the characteristics of the request stream and the available cache size will be used to measure the impact/benefits of placing a proxy at a specific location.

The estimation of the BHR (Byte Hit Ratio = 1 - Byte Miss Ratio (BMR)) value of a proxy server that is going to be placed in the network is based on the performance tables, which were derived through our study on proxy

servers [7]. We then use the characteristics of the request stream as index to the performance table.

2.1.2 Modeling of the total proxy placement cost.

We assume the existence of a collaborative environment under which proxies are placed. Thus, the objects that are not served by a proxy can be served by a nearby proxy with a probability which is based on the BMR of the proxy that generates a miss.

The placement algorithm tries to place the proxy servers in those nodes that will lead to the minimum total proxy placement cost. The later is computed taking into account the node weight, the edge weight, the minimum distance between two nodes and the network topology.

We also observe that the distance between two nodes $d(i,j)$ can model the transmission delay of a data packet from one node to the other or the number of hops in the path from i to j . In the first case, the total placement cost reflects the latency cost (i.e., the time needed) to serve all user requests as they appear in the workload. In the second case, the graph cost reflects the amount of data passing through the network links and can be expressed as bandwidth cost/requirements for the given workload.

Thus, if the latency perceived by users is the primary concern, one should choose to compute the total placement cost with latency constraints. If one's main concern is to improve the bandwidth cost one should choose the cost calculation reflecting bandwidth constraints.

2.2 Experimentation

We examined the performance of the following well-known algorithms: Greedy (which greedily places proxies in the network that yield the minimum total placement cost [13]), HotSpot (which tries to place proxies near communities of users that generate the greatest load [13]), and Random. For the performance study of the three algorithms (HotSpot, Random and Greedy) the backbone topologies of two major ISPs (AT&T and C&W) in North America were used. The performance results were similar for both topologies, AT&T and C&W. It appears that the Greedy algorithm is more efficient, while Random, as it is expected, is the worst. We observed for example by placing 2 proxies on the AT&T topology we achieve 26% improvement with the Greedy algorithm while the corresponding improvement for HotSpot is around 12% and for Random around 8% (The performance improvement is expressed as the decrease of the total placement cost which is the result of the proxy placement, compared to the total cost with no placement at all). Comparing our results with the performance results of other researchers [10],[13], who studied the same algorithms under the full-server replica placement problem, we note that despite the quantitative differences,

our results identifying the best algorithms are the same.

3. ProxyTeller

Based on our study on the placement of proxy servers [11] and on the performance results of cache replacement algorithms [7], we developed ProxyTeller that provides a solution to the placement problem. Currently, the tool is using the Greedy placement algorithm, but it can easily be updated to incorporate other algorithms as they emerge.

A user is initially asked to choose the scenario under which the ProxyTeller is called to give an answer (more on this in section 4). Then in the main input page (Figure 1), the user must give values to the key parameters that are essential for the execution of the placement algorithm. An analytical description of these follows.

Network topology

We use a specific format that describes the number of nodes, the number of edges, the edge weights, and the connectivity of each node. The user can select a

predefined network topology, such as AT&T's and C&W's, or create his own topology using a specific tool developed with Java.

Expected workload

A trace file, which is representative of the expected request stream, is essential in order to compute the node weights. Since the random assignment of node weights to nodes may lead to erroneous placement decisions, the user can alternatively provide N sub-traces for a N -node network topology, so as to associate each sub-trace to each node that generates the given workload. Also, the characteristics of workload are measured so as to determine the expected proxy server performance (in case of proxy placement). In the figures shown below, the "expected workload" parameter refers to the theta-value for the assumed Zipf distribution for the access requests. [14] Thus, the user can either import his representative trace file(s), or use already existing trace files.

The screenshot shows the ProxyTeller web application interface. At the top, there's a navigation bar with 'ProxyTeller' on the left, 'Choose Scenario' in the center, and 'Logout' on the right. Below this is a header for 'New Placement' and 'Scenario 1: Single Web Origin Server with Proxies'. A secondary navigation bar contains icons for 'Save Results', 'Load Results', 'Your Workloads', and 'Your Topologies'. The main content area is titled 'Input Parameters' and contains several form fields:

- Topology:** A dropdown menu labeled 'Select Topology' with a red error icon.
- Theta value for zipf distribution of requests:** A dropdown menu labeled 'Select Workload' with a red error icon.
- Cache Size:** A text input field followed by 'MBytes'. Below it, 'Average Requested Data per Node' is followed by another text input field and 'MBytes'.
- Expected Performance Improvement:** A text input field followed by '%'. It has a red error icon.
- Proxy Server Performance Metric:** A dropdown menu labeled 'Select Performance Metric' with a red error icon.
- Placement Performance Metric:** A dropdown menu labeled 'Select Placement Performance Metric' with a red error icon.
- Define a Single Origin Server (only 1):** A text input field with a red error icon. Below it, a note says 'Node numbers should be smaller than' followed by a text input field and 'Example: 1 4 6'.

On the right side of the 'Input Parameters' area, there are two links: 'Create your own Topology' and 'Upload your Expected Workload', both with red error icons. At the bottom center, there is a large blue button labeled 'Simulate Placement'.

Figure 1. ProxyTeller's input page

Available disk cache size at Proxies

The user is asked to define the size of the proxy caches to be placed in the network. The cache size is not necessary equal among all proxy caches. Thus, the user can choose to place N proxy caches with different cache sizes.

Desired performance improvement

In this part the user is required to define the percentage of the desired performance improvement that she wishes to achieve, by placing proxies (or full-server replicas) in the network. This percentage describes the reduction of the total network cost that is achieved due to the suitable placement, against the cost without any proxies at all in the network. As mentioned in section 2.1.2 the total network cost can be computed with either bandwidth or latency constraints. Thus, the user is also asked to define the appropriate performance metric. Additionally, the user can define the number of proxies he wishes to place.

Performance metrics

The user is also asked to define the performance metric of interest for individual proxy servers. He has three choices: bandwidth requirements, latency, and hit ratio. Depending on the user's selection, the proxy will be associated with a certain value of Byte Hit Ratio and the suitable local cache replacement algorithm for each proxy will be determined (selecting one of LRU, GDS(1) [4], GDS(lat) [4] and CSP [7], based on our results on the performance of proxy cache replacements algorithms).

How it works

After specifying these essential parameters the algorithm responsible for proxy placement, is iteratively called, placing more and more proxies, until the desired performance improvement is met, or until the prescribed number of proxies have been placed.

The output to the user consists of the results of Greedy placement: the network is shown with the proxies, the percentage of improvement achieved, the number of proxies placed, the replacement algorithm that the tool proposes for each proxy server, and the storage requirements for each proxy of the resulting placement.

The user can also run three comparison cases in order to compare results of proxy (forward or reverse) against full server replica placement (Figure 2). The tool gives a clear picture to the user about which placement to choose based on latency, bandwidth, and storage requirements.

By comparing the proxy against replica placement one can easily discover the most profitable (with respect to investments on storage devices) based on his expressed needs; i.e., to improve latency or bandwidth.

Apart from this basic procedure (specifying the input parameters and waiting for the tool's results), we added extra functionality in order to help users organize their

runs. Thus, users can store and load simulation results, descriptions of network topologies, and expected workloads. All this information is stored in files in a directory designated for each user, while extra information for managing these files is stored in a database.

4. ProxyTeller's utilization scenarios

In this section we show indicative scenarios for which ProxyTeller can be called to provide an answer. The scenarios are:

1. Single web origin server with proxies or full replicas

This usage scenario is relevant to a small organization, with a single web origin server, and its need to decide where to place proxy servers within the network to improve the efficiency of content delivery to clients. Similarly, ProxyTeller can be called to place full replicas, instead of proxies, able to store all objects of the web origin server.

2. Several web origin servers with proxies or full replicas

This scenario could emerge when there are many web origin servers storing different content. For example, a large CDN may wish to decide where to place reverse proxies and/or full replicas within its network to improve efficiency.

3. Incremental improvement: adding surrogate servers and/or full replicas in CDNs

Within a CDN the goal can be to improve further the overall performance by placing additional surrogate and/or full replica servers.

The results of ProxyTeller can also be used to decide if it is preferable to place surrogate servers vs full replicas. Thus the user may choose to run the comparison cases in order to compare placement results on scenarios 1, 2 and 3 (Figure 2).

5. Results for representative test cases

In this section we present some test case results for the above scenarios. We used the AT&T network backbone topology to test these scenarios, since we were able to obtain real load statistics concerning the edge weights. We asked ProxyTeller to improve the overall performance by 50% and we used as expected workloads the one of the synthetic workloads used in the study we made on proxy replacement algorithms performance [7]. The proxy server's performance metric in the following results was chosen to be latency reduction (have in mind that this choice affects the BHR value that characterizes each proxy).

Placement Results

Scenario 1
Comparison on Placement Results
Proxy Against Replica Placement

- Save Results
- Load Results
- Your Workloads
- Your Topologies

Input Parameters	
Topology	AT&T
Theta value	0.8
Cache Size	50 MByte
Expected Performance Improvement	50 %
Proxy Performance Metric	Latency
Placement Performance Metric	Latency

Results for Proxy Placement				
#	Overall Performance Improvement	Node Label	Cache Size	Proxy Cache Replacement Algorithm
[0]	0.0 %	1 (Origin Server)		
[1]	24.59 %	13	26.74 %	LRU
[2]	39.85 %	10	32.33 %	GDS(1)
[3]	49.26 %	5	47.07 %	GDS(1)
[4]	56.02 %	4	53.34 %	GDS(1)
Given desired performance improvement reached or exceeded				
4 proxies should be placed.				
Total number of Nodes in network 18				
Your performance constraint is latency. 56.02 % improvement has been achieved				
The performance improvement based on bandwidth constraints is 45.60 %				

Results for Full Server Replica Placement		
#	Overall Performance Improvement	Node Label
[0]	0.0 %	1 (Origin Server)
[1]	38.94 %	9
[2]	61.96 %	10
Given desired performance improvement reached or exceeded		
2 full replica servers should be placed.		
Total number of Nodes in network 18		
Your performance constraint is latency. 61.96 % improvement has been achieved		
The performance improvement based on bandwidth constraints is 51.59 %		

Comparison on Placement Results	
<p>Placing Proxies or Surrogate Servers and Storage Cost</p> <p>If you choose to place proxies or surrogate servers, you need 4 of them. In this case the storage requirements are expected to be 200 MBytes. ((Number of Proxies) x [Cache Size]) The performance improvement based on bandwidth constraints is 56.02 % The performance improvement based on latency given the above proxy placement is 45.60 %</p>	<p>Placing Full Server Replicas and Storage Cost</p> <p>If you choose to place full server replicas, you need 2 of them. This choice comes with a storage cost of 1625.29 MBytes. ((Number of Replicas) x [Sum of Server Object Sizes]) The performance improvement based on bandwidth constraints is 61.96 % The performance improvement based on latency given the above replica placement is 51.59 %</p>
<p>If your main concern is latency improvement then you should choose replica placement. With replica placement you increase latency improvement by 10.6 % compared to proxy placement.</p> <p>In that case the bandwidth improvement is increased by 13.14 % compared to proxy placement.</p> <p>If your main concern is the investment on storage devices then you should choose proxy placement as you need 87.69 % less space than replica placement.</p>	

Figure 2. ProxyTeller results: comparing proxy against replica placement for scenario 1

5.1 Initial placement and performance improvement results

5.1.1 Placement with latency constraints

The performance constraint was defined to be the minimization of latency (we used the latency reduction option for the total network cost calculation). The disk size (in the case of proxy placement) was defined to be 30% of the maximum required space to store all requested objects.

For scenario 1, we found that by placing 4 proxies in the network we can achieve a performance improvement up to 56.02%. These four proxies should be placed at nodes 13, 10, 5, and 4. The tool suggested the LRU replacement algorithm for proxy at node 13 while for the rest of the proxies the preferred replacement algorithm is GDS(1).

For the same scenario, we need to place only 2 replicas at nodes 9 and 10, achieving 61.96% improvement. It is clear that in this case we achieve better improvement but we have to pay the cost to equip the servers with enough space to accommodate the entire web origin server's content.

For scenario 2, assuming 3 web origin servers placed at nodes 1, 7 and 17, the tool decides to place 3 proxies at nodes 5, 9, and 10, achieving an overall performance improvement of 51.24%. These three proxies should be run the GDS(1) cache replacement algorithm. Deciding to place full server replicas, we need to place two full replicas in order to achieve 65.17% improvement.

5.1.2 Placement with bandwidth constraints

Next, we looked at the ProxyTeller results for minimizing bandwidth requirements. We present the results when placing forward proxies or surrogate servers.

For scenario 1, we found that by placing 5 forward proxies in the network we can achieve a performance improvement up to 51.87%. These four should be placed at nodes 4, 14, 5, 6, and 13. The tool suggested the LRU replacement algorithm for proxy at node 4 and 5 and GDS(lat) for the rest.

For full server replica placement, we need to place two full replicas in order to achieve 60.16% improvement.

For scenario 2, we have again 3 web origin servers placed at nodes 1, 7 and 17. The ProxyTeller decides to place four proxies at nodes 5, 4, 13, and 14, achieving an overall performance improvement of 51.42%. Proxies 5, 4 and 13 employ the GDS(lat) replacement algorithm while proxy 14 uses LRU. The corresponding results for full server replica placement yields the placement of two replicas at nodes 4 and 5 achieving an improvement of about 63.59%.

5.2 Incremental placement and performance improvement results

In this section we present our results of proxy against replica placement based on scenario 3 where previously a number of replicas in the network have already been placed. Our objective is to place more proxies or replicas in order to improve performance.

5.2.1 Placement with latency constraints

For latency minimization, ProxyTeller suggests the placement of 4 surrogate servers at nodes 9, 10, 5, and 6, under latency constraints (Figure 3). In this case we achieve an overall improvement of 53.88%. The corresponding results for full replica placement show that we should place 2 replicas at nodes 14 and 10, in order to achieve 56.41% improvement.

5.2.2 Placement with bandwidth constraints

For the minimization of bandwidth requirements, ProxyTeller suggests the placement of 7 surrogate servers at nodes 5, 4, 14, 13, 12, 10 and 11, under bandwidth constraints. In this case we achieve an overall improvement of 52.57%. The corresponding improvement achieved by placing 3 full server replicas (at nodes 14, 4, and 8) is 56.80%.

5.3 Storage requirements

The total storage cost associated with a decision to place different numbers of proxies or full-server replicas could be a key reason for choosing one or the other approach. This cost can be expressed as the storage investment needed to equip each full replica. Consider, for example, the case of Scenario 2 with latency constraints (section 5.1.1). Suppose for simplicity that each origin server stores an equal amount of data, S . Then each full replica server should be able to store $3*S$ quantity of data which yields a total capital investment of $2*(3*S)$. The corresponding investment for the proxy placement is $3*0.3*(3*S) = 0.9*(3*S)^2$.

Considering large-scale systems with hundreds of origins servers, the associated benefits will be much greater.

In our experiments we observed that the storage

² Apart from the storage cost, we should also notice the bandwidth required for the propagation of the replica updates in the case of full-replica placement. It is clear that proxy placement should be preferred over full-replica placement in such cases.

requirements in the case of replica placement are always greater compared to proxy placement, by a percent that

varies from 84% to 91%.

ProxyTeller

[New Placement](#)
[Logout](#)

Placement Results

Scenario 3 Comparison on Placement Results Proxy Against Replica Placement

[Save Results](#)
[Load Results](#)
[Your Workloads](#)
[Your Topologies](#)

Input Parameters	
Topology	AT&T
Theta value	0.8
Cache Size	50 MByte
Expected Performance Improvement	50 %
Proxy Performance Metric	Latency
Placement Performance Metric	Latency

Results for Proxy Placement					Results for Full Server Replica Placement		
#	Overall Performance Improvement	Node Label	Cache Size	Proxy Cache Replacement Algorithm	#	Overall Performance Improvement	Node Label
[0]	0.0 %	1 (Origin Server or Full Replica)			[0]	0.0 %	1 (Origin Server or Full Replica)
[0]	0.0 %	7 (Origin Server or Full Replica)			[0]	0.0 %	7 (Origin Server or Full Replica)
[0]	0.0 %	17 (Origin Server or Full Replica)			[0]	0.0 %	17 (Origin Server or Full Replica)
[1]	20.16 %	9	35.11 %	GDS(lat)	[1]	33.06 %	14
[2]	35.52 %	10	32.33 %	GDS(lat)	[2]	56.41 %	10
[3]	48.88 %	5	47.07 %	GDS(lat)	Given desired performance improvement reached or exceeded		
[4]	53.88 %	6	41.46 %	GDS(lat)	2 additional full replica servers should be placed.		
Given desired performance improvement reached or exceeded					Total number of Nodes in network 18		
4 surrogate servers should be placed.					Your performance constraint is latency. 56.41 % improvement has been achieved		
Total number of Nodes in network 18					The performance improvement based on bandwidth constraints is 42.74 %		
Your performance constraint is latency. 53.88 % improvement has been achieved					Your performance constraint is latency. 56.41 % improvement has been achieved		
The performance improvement based on bandwidth constraints is 33.00 %					The performance improvement based on bandwidth constraints is 42.74 %		

Comparison on Placement Results

Placing Proxies or Surrogate Servers and Storage Cost	Placing Full Server Replicas and Storage Cost
If you choose to place proxies or surrogate servers, you need 4 of them. In this case the storage requirements are expected to be 200 MBytes. (([Number of Proxies]x[Cache Size]) The performance improvement based on bandwidth constraints is 53.88 % The performance improvement based on latency given the above proxy placement is 33.00 %	If you choose to place full server replicas, you need 2 of them. This choice comes with a storage cost of 1625.29 MBytes. (([Number of Replicas]x[Sum of Server Object Sizes]) The performance improvement based on bandwidth constraints is 56.41 % The performance improvement based on latency given the above replica placement is 42.74 %
If your main concern is latency improvement then you should choose replica placement . With replica placement you increase latency improvement by 4.7 % compared to proxy placement.	
In that case the bandwidth improvement is increased by 29.52 % compared to proxy placement.	
If your main concern is the investment on storage devices then you should choose proxy placement as you need 87.69 % less space than replica placement.	

Figure 3. ProxyTeller results: comparing proxy against replica placement for scenario 3

6. Concluding remarks

We have presented ProxyTeller, a tool we have designed and implemented to guide the placement of proxies and/or server replicas of origin servers in a content delivery network. ProxyTeller accepts as inputs the network topology, the desired sizes of proxy caches, the characteristics of the workload of the community served by a proxy, and the desired performance improvement. The output of ProxyTeller is the position and the number of proxies required in order to achieve the stated performance goals in terms of access latency, network bandwidth, server load, and storage requirements.

ProxyTeller can be used for both forward and reverse proxies and, we hope, it will be of real use to the organizations maintaining them, and in particular:

- to ISPs/CDNs, to optimally decide on required capital investments, weighing them against expected performance improvements, and
- to content providers, which will be able to estimate their origin server off-load possible (as this is depicted in cache hit ratios) from a potential contract with a CDN and the expected performance improvements seen by the consumers of their content.

As mentioned, we have implemented the tool, making it available through the internet, for use by the community [1]. Our ultimate goal is, through ProxyTeller, to accumulate and combine related research results, which form its basis (i.e., new results on the performance of various mechanisms improving the performance of single proxies, improving/extending the cache replacement problem solution chart, and new results for server/proxy placement), make it available to all, and provide the infrastructure to utilize it in order answer key questions faced in the context of efficient content delivery.

7. References

- [1] ProxyTeller: A Proxy Cache Placement Tool. Demo available at <http://proxyteller.ceid.upatras.gr>
- [2] M. Abrams, C. Standridge, G. Abdulla, S. Williams, and E. Fox, "Caching Proxies: Limitations and Potentials", in Proceedings of the 1995 World Wide Web Conference, December 1995.
- [3] C. Aggarwal, J. L. Wolf and P. S. Yu, "Caching on the World Wide Web", IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, January/February 1999.
- [4] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", in Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, December 1997.
- [5] J. Shim, P. Scheuermann and R. Vingralek, "Proxy Cache Algorithms: Design, Implementation and Performance", in IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 4, July/August 1999.
- [6] M. Rabinovich and O. Spatscheck, "Web Caching and Replication", Addison-Wesley, ISBN: 0-201-61570-3, 2002
- [7] P. Triantafillou and I. Aekaterinidis, "Web Proxy Cache Replacement: Do's, Don'ts, and Expectations", In Proceedings of The 2nd IEEE International Symposium on Network Computing and Applications (NCA-03), April 2003.
- [8] M. Charikar, and S. Guha, "Improved Combinatorial Algorithms for the Facility Location and K-Median Problems", in Proceedings of the 40th Annual IEEE Conference on Foundations of Computer Science, 1999.
- [9] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks", in Proceedings of the ACM SIGCOMM Internet Measurement Workshop, November 2001.
- [10] A. Venkataramani, P. Weidmann, and M. Dahlin, "Bandwidth constrained placement in a WAN", Twentieth ACM Symposium on Principles of Distributed Computing (PODC 2001), August 2001.
- [11] P. Triantafillou and I. Aekaterinidis, "ProxyTeller: A Proxy Placement Tool for Content Delivery under Performance Constraints", In Proceedings of the 4th IEEE International Conference on Web Information Systems Engineering (WISE '03), December 2003.
- [12] B. Li, M. J. Golin, G. F. Italiano, and X. Deng, "On the placement of Web proxies in the Internet", in Proceedings of the INFOCOM 2000, March 2000.
- [13] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of Web server replicas", in Proceedings of the IEEE INFOCOM 2001, April 2001.
- [14] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", in Proceedings of the IEEE INFOCOM, March 1999.