Content Distribution Through Autonomic Content and Storage Management*

Nikolaos Laoutaris, Antonios Panagakis, and Ioannis Stavrakakis

Department of Informatics and Telecommunications,
University of Athens, 15784 Athens, Greece
{laoutaris, apan, ioannis}@di.uoa.gr

Abstract. Content Distribution has to date been addressed by a mix of centralized and uncoordinated distributed processes, such as server replication and traditional node caching mechanisms, respectively. It is an inherently distributed process that is also increasingly relying on entities that are not only increasingly distributed but also increasingly autonomous. Consequently, centralized – and typically targeting the "socially optimal" – decisions are rather unrealistic for a distributed environment of autonomic entities. Instead, a distributed management of the engaged autonomic entities, which take decisions dynamically, should be key to efficient content distribution. The latter is advocated in this paper in which two entities that are central to content distribution - specifically the content and the node storage – are considered and it is discussed how their autonomic behavior drives the operation of a content distribution network. In the first case, it is the content that manages itself by dynamically generating duplicate copies and pushing them to (seizing) the appropriate storage. In the second one, it is the node storage that is in charge, deciding on the content to be locally stored. The decisions taken by the distributed and autonomic entities may – in the extreme case – be driven by self-awareness and self-interest only, without any network state information and co-operativeness. Or, they may use (some) network information and take decisions in a more cooperative manner, despite their autonomic and self-interest-driven nature. An example is presented on the later case, showing the potential both social and individual benefits.

1 Introduction

Communication networks have up to very recently been designed, optimized, and built, based on a careful planning and allocation of the primary network resource, the bandwidth. The emergence of the Internet and the World Wide Web as the main information delivery vehicles of our society, have necessitated the deployment of large amounts of network storage (or memory). The addition of storage capacity in network nodes, for the caching or replication of popular information documents in

* This work and its dissemination efforts have been supported in part by the IST Program of the European Union under contract IST-6475 (ACCA) and by the project entitled "Always Best Connected Provision in Heterogeneous Mobile Networks" funded by the Greek Ministy of Education under the framework "Pythagoras".

close proximity to the end users, has appeared as a viable and efficient alternative to adding more bandwidth, or deploying complicated quality of service architectures. It is generally believed that the deployment of network storage has helped in reducing end-user delays, network traffic, and in improving the overall scalability in the Internet content distribution chain. Thus, network storage has emerged as another important network resource that can substantially enhance the network performance and/or reduce the requirement for bandwidth.

Following the usual Internet development track, storage capacity has been added progressively by a plethora of different authorities and applications that, in most cases, operate independently. Services and applications like CDNs, P2P, and others like end-system multicast, have formed logical overlay networks over the physical internet infrastructure. As a result, the Internet has been seeded by a large amount of storage capacity that now serves as a common substrate for the support of a diverse set of content delivery schemes, both contemporary and planned ones.

Despite the impressive reduction in the cost of storage brought by the latest generation of storage devices, storage remains a valuable resource (both in owning and also in managing), especially in view of the latest user trend to exchange voluminous information documents, e.g., multi-megabyte music and video files which, by latest reports, amount to well above of 75 % of all Internet traffic [19]. Contributing to this trend is also the automatic dissemination of software updates (operating system/application updates, virus fixes) that has become a standard feature of most operating systems and applications [16]. The combination of uncoordinated deployment of storage, and the conventional wisdom that storage is cheap, has resulted in a rather limited emphasis on exploiting the new resource up to its full potential, and has set the stage for what appears to be a new contention for resources - this time for storage capacity - fuelled by the desire to disseminate voluminous content.

If the provisioning of memory continues to materialize as it has in the recent past, then in the very near future (autonomous) memory pools (CDN nodes, or local proxy servers) will be in place in most systems that constitute the Internet [6]. Building adaptive overlay content distribution systems on top of the underlying memory pools can provide for a significant alternative to the static provisioning of memory as materialized with the current replication schemes that employ very large granules of memory (e.g., entire mirror site).

Should (autonomic) memory pools exist and be marketed, content creators (or intermediaries) can build distribution systems on them by leasing storage capacity dynamically. The main advantage of such a scenario is that memory will be utilized more efficiently, and at a finer granularity, as potential users or applications will be able to use it on-demand and release[1] it when no longer needed making it available to other users that may request and pay for it (protocols and e-currencies for such resource trade paradigms have been proposed recently

[1] Datagrams (on demand allocation of bandwidth to packets) has been the cornerstone of the Internet. The on demand allocation of storage to content seems to be as meaningful.

[21]). Re-organizing the memory is not possible with the current installment of dedicated mirrors and proxies in fixed locations and with fixed capacities. It is believed that the ability to reorganize the existing resources will be central to future intelligent information systems (see IBM's *autonomic computing* initiative [8]). The decisions concerning the management of the resources should be based both on the content requirements and the storage availability. In the sequel we take two different approaches at discussing the efficient utilization of the storage resource, one from the perspective of content itself, and the other from the perspective of the amorphous storage.

Our motivation for discussing these two approaches is to augment the current paradigm of placing the content only at fixed distribution points (e.g., the point of emergence (creation) and some mirror points), by allowing for the content to track adaptively the topology of the demand, and initiating a migration towards the areas of high demand without the intervention of a centralized authority. Such an approach will hopefully allow for a group of cooperating nodes to adaptively track and best serve the demand, without requiring centralized control; such a control is usually not present in Internet applications that are distributed and handled by multiple authorities.

2 Content Perspective

In this section we examine some requirements for autonomic content distribution, stemming from the perspective of content; the amount and the locations of the available storage are considered to be known here. This could materialize by first communicating with a Storage Broker entity (centralized or distributed one) from which storage is leased dynamically. The goal is to use the available storage to best serve the dynamically changing demand. For this purpose we conceptualize the tools of content movement and content duplication.

Content movement aims at pushing the appropriate content closer to the appropriate location. Content duplication spawns dynamically multiple copies of an information document in accordance with the request intensity; high demand leads to the increase of the number of copies, while a low demand leads to the decrease of the number of copies, trying to maintain an appropriate number of copies at various locations, to best suit the demand from the clients. We can make the following observations regarding the essence of employing each one of these concepts in isolation.

Sole application of content duplication without a limit on the number of replicas (extreme self-serving content behavior) for a given document allows multiple (or even all) clients to "own a local copy" of the desired document. A common problem with this strategy is that – due to the lack of coordination and the unrestricted number of allowed replicas – it leads to an excessive repetitious replication of the same (few) documents; the latter is clearly sub-optimal, considering that an off-line optimal replication policy forces multiple clients to "share" a single document replica, thereby increasing the number of distinct documents that may be hosted altogether.

Sole application of content movement with a small-fixed number of document replicas available, leads to a game of "tug-of-war", where the client (individual or group) that issues the most requests for a document succeeds in drawing it closer. Although it is justifiable to have the content closer to the location of highest demand, content movement alone falls short of best handling the demand, as it has to operate under a fixed number of document replicas, which may be a serious restriction. Under a fixed number of replicas that happens to be lower than the number of high-demand locations, the users will be served under a sub-optimal solution, as the freedom of allowing each location to have its own local copy (if that were the optimal solution under high enough request rates) would not exist.

The above discussion suggests that both concepts be employed and an adaptive trade-off between content duplication and movement be exercised by an efficient content distribution strategy. Thus practical and efficient rules for regulating between "tug-of-war" (forcing the clients to share) and "own local copy" (allowing multiple clients to have local copies when the corresponding request rates are high enough, which in turn limits the number of hosted distinct documents) should be identified. This fundamental trade-off between the number of replicas of each hosted document and the number of distinct documents is at the heart of an efficient utilization of the storage resource.

An interesting possibility is to consider autonomic content entities by assigning the responsibility for movement and duplication decisions to the content itself, rather than the content creator and the origin server that first injects the content to the network. To be able to make such decisions, content must be accompanied by a set of attributes that will allow it to act in an autonomic manner. As an example, imagine a movie file that is being injected in the network from the location of its origin server. The creator of the movie supplies it with attributes like storage credits (i.e., a budget for buying storage at replication points), maximum lifetime, "geographical" boundaries (set of ISPs in which it may spread) and other general characteristics for empowering its ability to manage itself. One such ability is the ability to split itself. This is stimulated by an interesting categorization of the targeted content among integral documents (i.e., documents that are indivisible, one document=one file) and non-integral documents (one document divided into multiple parts). The first case to be used with small/medium sized documents (e.g., html pages, images) while the second to be used with voluminous documents (e.g., software updates). The case of voluminous non-integral documents calls for special handling as compared to the case of integral documents (whose relatively small size permits them to move and duplicate as a whole). For voluminous content, applying movement and duplication to the entire document might be restricting due to large size; this is because potential movement and duplication actions become infeasible as few hosting nodes can accommodate such large object in their entirety. Segmenting such documents into multiple parts, that may be handled by different nodes, partly alleviates the problem of volume, but creates new challenges for orchestrating among the different constituent parts. In such cases, additional rules must be defined so as to maintain some degree of coordination among the multiple parts, as they move and duplicate about the network in response to the demand. A foreseeable target for such a coordination is, for example, to guarantee that the multiple parts constituting an entire document remain within a maximum distance of each other, so as to facilitate an uninterrupted (parallel or sequential) streaming towards a receiver.

Off-line algorithms that have a priori knowledge of the demand and topology are able to optimize the trade-off between the aforementioned concepts of *content movement* and *content duplication* by computing the relative value of each additional replica of a document and balancing it against the relative value of hosting a new (not yet replicated) document. Achieving this optimal trade-off in a distributed, on-line manner is challenging and yet unexplored and could be pursued by the proposed combination of the proposed concepts of content movement and duplication, possibly including additional recently proposed ideas such as parallel downloads and appropriate encoding schemes.

3 Storage Perspective

In this section we discuss the role of storage in an autonomic content distribution framework. We assume that storage is employed by the nodes for replicating content so that they may provide it to local users promptly, while limiting the consumption of bandwidth; essentially, the installed storage is used for absorbing the local demand for content, and not letting it flow to the network. Traditionally, a node's storage may be either managed by a central authority (e.g., owner of a CDN) in a way that maximizes the network's benefit, or by the individual node in isolation (e.g., typical user caches), in a way that maximizes the specific node's benefit.

The huge proliferation of the installed interconnected node storage calls for a reconsideration of the aforementioned traditional storage management paradigms. On one hand, centralized decisions are less feasible due to the lack of a single owner of the resources. On the other, the autonomous node storage facilities should not be managed in isolation catering to their own needs in a selfish and myopic manner but, instead, cooperation among the otherwise autonomic node storage entities should be considered.

The way that nodes cooperate in utilizing their storage resource is ultimately shaped by the scope of their utility, whether local (selfish behavior) or global (social aware behavior). We discuss such issues using the abstraction of a distributed replication group [14]. Such an abstraction is commonly employed for studying content distribution application such as web caching, web mirroring, content distribution networks (CDNs), and peer-to-peer applications.

Under this abstraction, nodes utilize their storage capacity to replicate information objects that they make available to local and remote users. A request that is issued by a local user and can be serviced locally (i.e., it involves a locally replicated object) is served immediately thus incurring a minimal cost. Otherwise, the requested object is searched in other nodes of the group and if not found, it is retrieved from the origin server; the access cost, however, increases with the distance. Depending on the particular application, the search for objects at remote nodes may be conducted through query protocols [22], succinct summaries [5], DNS redirection [17] or distributed hash tables [20].

Several placements problems can be defined regarding a distributed replication group. The proxy (or cache, or mirror, or surrogate) placement problem has been studied in several works, including [15, 18, 4]. The object placement problem refers to the selection of objects for the nodes, under given node locations and capacities [14, 10, 9, 2]. Finally, works such as [12, 13] combine node placement, node dimensioning and object placement in one problem.

All the aforementioned work in the field has centered around the optimization of the so called *social utility*, which is made of the sum of the individual *local utilities* of the nodes; here the term utility refers to delay and bandwidth gains from employing replication. The quest for optimizing the social utility arises naturally in applications where a centralized authority dictates replication decisions to the nodes. It suits well applications such as web mirroring and CDNs, which are operated under centralized control (the content creator or content distributor playing that role). Applications that are run by multiple authorities, such as web caching networks and P2P networks may too adhere to the goal of an optimized social utility, but this may come only as an act of voluntarism, as the optimization of social utility is often harmful to several local utilities.

Take as an example a group of nodes that collectively replicate content. If one of the nodes is generating the majority of requests, then a socially optimal (SO) object placement ends up using the storage capacity of other nodes to replicate objects that do not fit in the over-active node's cache. The users of these other nodes experience a service deterioration as a result of their storage being hijacked by potentially irrelevant content; in fact, such nodes are better off acting on their own, and employing a greedy local (GL) placement (i.e., replicating the most popular objects as pertaining to the local demand). The same situation arises if caching, rather that replication, is in place; remote hits originating from other nodes may evict objects of local interest in an LRU operated cache that participates in a web caching network. Fear of such exploitation may prevent nodes from participating in such groups and instead lead them to operate in isolation in a greedy local manner.

Being greedy local is often ineffective not only to the social utility, but to one's local utility too. For example, when nodes have similar demand patterns and inter-node distances are small, then replicating multiple times the same most popular objects, as done by a GL object placement, is highly ineffective. Clearly, there is a substantial gain for all nodes in that case, if they cooperate and replicate different objects; in fact, all local utilities may improve as compared to the GL performance, if such cooperation takes place. Nodes, however, are generally not aware of the remote demand patterns, thus cannot recognize such opportunities for cooperation. On the other hand, as discussed earlier, they cannot blindly trust a SO object placement as they do not know whether it will be for good or bad as pertaining to their local utility.

To address such problems equilibrium (EQ) placement strategies, which can guarantee that a node's local utility will always be better under EQ than under GL, may be used. A node has no reason not to participate in such placement strategies, as it has only to benefit from such participation.

For example in [11] an EQ strategy has been presented, which is based on the notion of Nash equilibrium, and extends the replication problem defined by Leff et al. [14] (who have developed the SO replication strategy) to the case of multiple local utilities. A two-step local search (TSLS) algorithm is derived that computes the EQ strategy. The TSLS algorithm can be implemented in a distributed manner, and for its execution each node needs to know only its local demand pattern and the objects selected for replication by remote nodes, but not the remote demand patterns (as required by centralized replication algorithms that compute the SO strategy). In addition, a distributed protocol that implements TSLS and requires minimal exchange of information has been developed. In the sequel we give a numerical example with the aim of highlighting the aforementioned placement strategies and their relevance to the self-interest of individual nodes.

3.1 A Numerical Example

In this section we give a numerical example to demonstrate the potential benefits of the TSLS algorithm. This is an example of an algorithm that is run by each autonomic node in order to take decisions in a cooperative (not isolated) framework utilizing (some) limited network information and yielding decisions that increase the global gain (benefit) without ever reducing the individual gain (benefit) enjoyed when acting in a self-serving manner only, in isolation.

There are two nodes that generate requests from the same Zipf-like distribution that assigns to the ith most popular object a request probability K/i^a, where $K = (\sum_{i'=1}^{N} \frac{1}{i'^a})^{-1}$; N denotes the number of distinct objects, a the skewness parameter of the distribution, and ρ_j the total request rate from the jth node (here $j = 1$ or 2). The local access cost is, $t_l = 0$, the remote one, $t_r = 1$, and the cost of accessing the origin server, $t_s = 2$; this leads to a hop-count notion of distance. It is assumed that there exist $N = 100$ distinct objects, and that each node has a storage capacity for $C = 40$ objects.

In Table 1 we show the objects replicated under the GL, SO, and EQ replications strategies for fixed $\rho_1 = 1$ and varying ρ_2. The GL strategy selects for each node the first 40 most popular objects, i.e., those with ids in {1:40}, independently of ρ_2. The SO strategy, however, is much different. As the request rate from Node 2 increases, SO uses some of the storage capacity of Node 1 for replicating objects that do not fit in Node 2's cache, thereby depriving Node 1 of valuable storage capacity for its own objects. For $\rho_2 = 10$, Node 1 gets to store only 3 of its most popular objects, while it uses the rest of its storage for picking up the next 37 more popular objects for Node 2, starting with the one with id 41. Under the EQ strategy Node 1 (v_1) stores 23 of its most popular objects. Node 2 (v_2) is the second one (i.e., the last one) to improve its placement, and it naturally selects the initial 40 most popular objects.

Table 1. An example with v_1, v_2 having the same Zipf-like demand pattern with $a = 0.8$. The number of available objects is $N = 100$ and the storage capacity of each node is $C = 40$. Also, $t_l = 0$, $t_r = 1$, $t_s = 2$, $\rho_1 = 1$

placement strategy	Node 1 objects	Node 2 objects
GL, $\rho_2 = X$	{1:40}	{1:40}
SO, $\rho_2 = 1$	$\{1:16\} \cup \{41:64\}$	{1:40}
SO, $\rho_2 = 2$	$\{1:12\} \cup \{41:68\}$	{1:40}
SO, $\rho_2 = 3$	$\{1:9\} \cup \{41:71\}$	{1:40}
SO, $\rho_2 = 4$	$\{1:7\} \cup \{41:73\}$	{1:40}
SO, $\rho_2 = 5$	$\{1:6\} \cup \{41:74\}$	{1:40}
SO, $\rho_2 = 6$	$\{1:5\} \cup \{41:75\}$	{1:40}
SO, $\rho_2 = 7$	$\{1:4\} \cup \{41:76\}$	{1:40}
SO, $\rho_2 = 8$	$\{1:4\} \cup \{41:76\}$	{1:40}
SO, $\rho_2 = 9$	$\{1:3\} \cup \{41:77\}$	{1:40}
SO, $\rho_2 = 10$	$\{1:3\} \cup \{41:77\}$	{1:40}
EQ, $\rho_2 = X$	$\{1:23\} \cup \{41:57\}$	{1:40}

We turn our attention now to the average individual and social access costs under the various placement strategies. The local access cost for node v_j is $\sum_{o_i \in P_j} r_{ij} \cdot t_l + \sum_{\substack{o_i \notin P_j \\ o_i \in P_{-j}}} r_{ij} \cdot t_r + \sum_{o_i \notin (P_j \cup P_{-j})} r_{ij} \cdot t_s$, whereas the social cost is the weighted sum of access costs of individual nodes, the weighing factor being the normalized request rate $\rho_j / \sum_{v_{j'} \in V} \rho_{j'}$; r_{ij} denotes the request rate at node v_j for object o_i, P_j denotes the *placement* of node v_j, i.e., the set of objects that it replicates, whereas P_{-j} denotes the collective set of objects that are replicated at all nodes except v_j. Figure 1 shows that as ρ_2 increases, the access cost for v_2 under SO decreases as it intercepts storage from v_1 for replicating objects according to its preference; v_1's access cost under SO increases rapidly as a result of not being able to replicate locally some of its most popular objects. In fact, for $\rho_2 > 2$, v_1's cost is worse (higher) that the corresponding one under GL. From this point and onwards, v_1 is mistreated by the SO strategy and thus it has no incentive in participating in it, as it can obviously do better on its own under a GL placement. Notice also that as a consequence of v_2's higher request rate, the social cost under SO follows in profile v_2's cost under SO.

By following the EQ strategy, v_1's cost cannot become higher than under GL, that is, v_1 cannot be mistreated, independently of ρ_2 and other parameters. In fact, both nodes succeed in doing better under EQ than under GL. Node v_2, however, benefits the most, and thus incurs a lower cost than v_1. This owes to the fact that v_2 is the second (last) one to improve its placement and, thus, has an advantage.

4 Related Work

We are aware of only few very recent works on game-theoretic aspects of caching and replication. Hadjiefthymiades et al. [7] (May, 2004), have studied the contention between different users that compete for storage in a single cache, and have

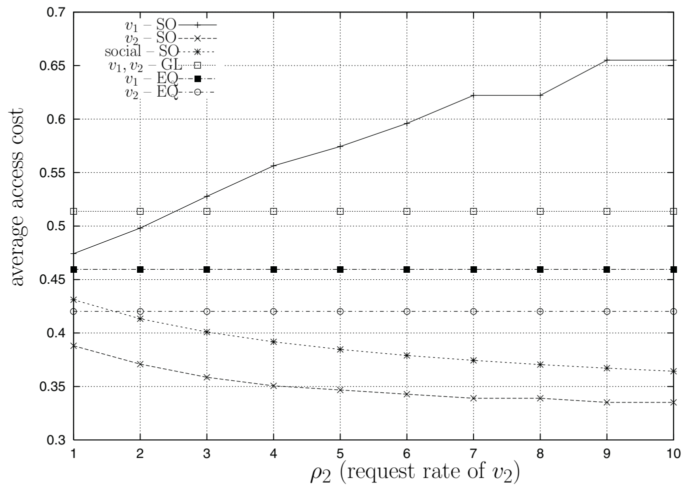

Fig. 1. Average cost for the example of Table 1: "v_j – XX" denotes the local cost for node v_j under the placement strategy XX; "social XX" denotes the social cost under the placement strategy XX

modeled it as a continuous game. More relevant to our work is the work of Chun et al. [3] (July, 2004), which studies distributed selfish replication. However, this work does not consider storage capacity limits on the nodes and, thus, differs substantially from our approach. Recent works on incentives in P2P networks, e.g., Antoniadis et al. [1], study the problem of attracting users to a P2P network and making them contribute more content. The aforementioned work, however, formulates the problem at a completely different level as compared to the current work, as it focuses on the number of files shared by each node, without identifying the identities of these files, whereas we focus on identifying the exact set of files shared.

References

1. Panayotis Antoniadis, Costas Courcoubetis, and Robin Mason. Comparing economic insentives in peer-to-peer networks. Computer Networks, 46(1):1–146, September 2004.
2. Ivan D. Baev and Rajmohan Rajaraman. Approximation algorithms for data placement in arbitrary networks. In Proceedings of the 12th Annual Symposium on Discrete Algorithms (ACM-SIAM SODA), pages 661–670, January 2001.
3. Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz. Selfish caching in distributed systems: A game-theoretic analysis. In Proc. ACM Symposium on Principles of Distributed Computing (ACM PODC), Newfoundland, Canada, July 2004.
4. Eric Cronin, Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz, and Yuval Shavitt. Constraint mirror placement on the internet. IEEE Journal on Selected Areas in Communications, 20(7), September 2002.
5. Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Transactions on Networking, 8(3):281–293, 2000.

6. Garth A. Gibson and Rodney Van Meter. Network attached storage architecture. Communications of the ACM, 43(11):37–45, November 2000.
7. S. Hadjiefthymiades, Y. Georgiadis, and L. Merakos. A game theoretic approach to web caching. In Proceedings of IFIP Networking 2004, Athens, Greece, May 2004.
8. IBM Corp. Autonomic Computing initiative, 2002. http://www.research.ibm.com/autonomic/.
9. Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. Computer Communications, 25(4):376–383, March 2002.
10. Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Placement algorithms for hierarchical cooperative caching. In Proceedings of the 10th Annual Symposium on Discrete Algorithms (ACM-SIAM SODA), pages 586–595, 1999.
11. Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. Distributed selfish replication, 2004. [submitted]
12. Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. Joint object placement and node dimensioning for internet content distribution. Information Processing Letters, 89(6):273–279, March 2004.
13. Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. On the optimization of storage capacity allocation for content distribution. Computer Networks, 2005. [to appear]
14. A. Leff, L. Wolf, and P. S. Yu. Replication algorithms in a remote caching architecture. IEEE Transactions on Parallel and Distributed Systems, 4(11):1185–1204, 1993.
15. Bo Li, Mordecai J. Golin, Giuseppe F. Italiano, Xin Deng, and Kazem Sohraby. On the optimal placement of web proxies in the internet. In Proceedings of the Conference on Computer Communications (IEEE Infocom), New York, March 1999.
16. J. Li, P.L. Reiher, and G.J. Popek. Resilient self-organizing overlay networks for security update deliver. IEEE Journal on Selected Areas in Communications, 22(1):189–202, January 2004.
17. Jianping Pan, Y. Thomas Hou, and Bo Li. An overview DNS-based server selection in content distribution networks. Computer Networks, 43(6), December 2003.
18. Lili Qiu, Venkata Padmanabhan, and Geoffrey Voelker. On the placement of web server replicas. In Proceedings of the Conference on Computer Communications (IEEE Infocom), Anchorage, Alaska, April 2001.
19. Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), December 2002.
20. I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking, 11(1):17–32, 2003.
21. David A. Turner and Keith W. Ross. A lightweight currency paradigm for the P2P reseource market, 2003. [submitted work].
22. Duane Wessels and K. Claffy. ICP and the Squid web cache. IEEE Journal on Selected Areas in Communications, 16(3), April 1998.