Designing a Decentralized Peer-to-Peer Network with Rapid Service Discovery and Effective Load Balancing[1]

Ching-Wen Chen[2]
Department of Computer Science and Information Engineering
Chaoyang University of Technology, Taiwan
chingwen@mail.cyut.edu.tw

Phui-Si Gan
Department of Computer Science and Information Engineering
Chaoyang University of Technology, Taiwan
s9227601@mail.cyut.edu.tw

Chao-Hsiang Yang
Department of Computer Science and Information Engineering
Chaoyang University of Technology, Taiwan
s9227616@mail.cyut.edu.tw

Abstract

Peer-to-peer systems (P2P) have attracted much current attention as a popular way to share huge volumes of resources directly within peers. Efficient content search and load balancing are both important research issues in P2P environments, especially in those without centralized global indexes. Although a number of content search schemes are known to alleviate these problems, they cannot provide flexible object searches while balance the load for content demand in P2P network, thereby resulting in severe load imbalance and consequently increased user response times. In this paper we tend to address these problems by harnessing all available resources in the P2P network and balance the load without decreasing user response time. We proposed a novel decentralized service discovery mechanism by creating a new service call P2P Registry (P2PReg) in Domain Name Server (DNS). In our scheme, each peer is able to register and discover desirable services automatically through current DNS within a short duration in an efficient way. The proposed method not only performs least searching time and low bandwidth but also balances load in P2P systems in order to avoid hot spot problems naturally. It is obvious that the more replicas there are, the better the media delivery quality of the specific content object is. We also present the results of simulation work to validate the viability of our approach. Our simulation study showed that serving data discovery over local search is indeed effective in improving the system performance significantly.

Keywords: Service Discovery, Contents Discovery, Load Balancing, Decentralized Peer-to-Peer Networks (P2P), Domain Name Server (DNS)

[1] This research was supported by the National Science Council NSC- 92-2213-E-324-006-

[2] Corresponding author
Tel: +886-4-23323000 Ext. 4534 Fax: +886-4-23742375

1. Introduction

Recent years have seen a tremendous proliferation of peer-to-peer (P2P) file sharing applications such as Napster [1] and KaZaA [2] as a popular way to share huge volumes of resources. In this context, P2P systems allows users to share, search for and download files. P2P is not new as it has been investigated for decades [3], [4]. Many researchers have been focused on understanding the issues surrounding these systems and built several P2P applications to demonstrate the usefulness of this new technology. It is clear that P2P research is an old but rich area for versatile research.

The most distinct characteristic of P2P computing is that there is symmetric communication between the peers; each participating autonomous computing node or device represents as a peer and behaving as both client and server, which can collaborate with each other in symmetric. It is clear that network bandwidth is better utilized rely on the direct communication within peers. Disk space for storing the files is also distributed across the peers in the P2P networks. These networks scale indefinitely without decreasing search time and without the need for costly centralized resources. They utilize the processing and networking power of the end-users machines since these resources always grow in direct proportion to the network itself.

Although P2P distributed computing paradigm alleviates the scalability problem that has dogged client-server systems and enable a lot of interesting and useful applications, there are limitations that come with these advantages of its complexity in peer discovery in P2P network that lacks a centralized server. Hence the ability to effectively locate a peer and retrieve services from a potentially huge number of decentralized peers in a decentralized network within a short duration is rather important and challengeable. We would like to address these problems by investigating the benefit of Domain Name Server (DNS) hierarchical structure into the P2P architecture and sufficient query communication.

While much work has been done to harness the huge computing resources of P2P systems for high-performance computing and scientific applications, issues concerning load-balancing with a view towards faster access to data for normal users have not received adequate attention. Global search results in an unfair allocation of user objects to peers and creates *hotspots* in certain parts of the zone. A hotspot refers to a node which receives a large volume of requests in rapidly and dramatically, often resulting in the node being overwhelmed and response times shooting up. Notably, load-balanced P2P system not only provides considerable savings in network bandwidth, but also ensures that the benefit of powerful networks extends well beyond e-commerce and scientific applications to individuals living in this world.

The purpose of this paper is to investigate the problem of efficient content discovery and load balance in the form of a DNS based directory hierarchy. We proposed a novel decentralized service discovery mechanism by creating a new service call *P2P Registry* (*P2PReg*) in DNS. With the help of DNS, we can discover desirable peers and obtain services from the peer within a short duration. In our scheme, each DNS server has a local peer index for quickly finding local peer when a query is received. Hence, a requesting peer can find all its potential serving peers and send query *q* to search for desired files directly to peers which holding related file. We also capture the interests of the peers through the number and types of files maintained and provided by the peers in the network.

According to the inherent hierarchical architecture of DNS, the proposed method not only performs least searching time and low bandwidth but also ensure that the search process result in a nearest peer that has the desired content tends to balance load in order to avoid hot spot problems naturally. This is all achieved without the need for costly centralized resources.We also present the results of simulation work to validate the viability of our approach.

This paper is organized as follows. We point out the major research issues of P2P systems and various related work in section 2 and describe how the trees architecture is formed and contents are delivered upon request with our proposed service discovery method in section 3. Then, in Section 4, we will describe how to balance load in P2P systems with our proposed method in order to avoid hot spot problems naturally. Section 5 presents the simulation model for a more realistic system and reports results. The last section gives the conclusion for the paper.

2. Related Work

Enormous discussions on P2P computing with definitions, current trends and related issues and tradeoffs in designing a scalable P2P system can be found at [5, 6]. Tsoumakos [7] gave a through comparisons and analysis on P2P systems, and pointed out the trends and potential profits of P2P systems over arbitrary network topology. A number of open problems have been addressed such as traffic problem, cost and vulnerable central index, yet each of the architecture possesses its own advantages and disadvantages with a significant amount of heterogeneity among them.

As an example for a pure peer-to-peer system, the original Gnutella implements fully distributed searching which does not build indices, every query is propagated to every peer, users search for files by flooding the network with queries; this may have difficulty to deal with the large numbers of sites or complex queries. Instead, resulting in a simple but very costly approach as just a simple query can flood the Gnutella network [8].

Napster is not really considered as real P2P systems in the strictest sense as it uses a centralized server to maintain a directory of MP3 music files that are on users' computer. Although it is simple and able to locate files quickly and efficiently due to the central index database it maintains. It remains as one of the easiest networks to target and can be shut down by a court order or hacker attack.. As a result, it will likely end up with unacceptable delay and user frustration.

However, Freenet [9] is open source, provide an anonymous file sharing service to guarantee clients and publishers anonymity, and uses no centralized server. It is decentralized and symmetric and automatically adapts when hosts leave and join. Freenet allows data to be published, replicated and retrieved while maintaining the anonymity of data producers and consumers.

Yang [10] proposed efficient search methods for P2P environments. They considered the trade-off between the number of messages and response times in searches and proposed methods combining the Gnutella-based breadth-first search and Freenet-based depth-first search.

The problem of the efficient search in a P2P network is also addressed in [11] by introducing the concept of routing indices, which allow nodes to forward queries to their neighbors that are more likely to have answers.

Brocade [12] proposed hierarchical virtual network infrastructure based on physical topology. Brocade constructs a secondary overlay to be layered on top of peer-to-peer lookup systems. The secondary overlay builds a location layer between super-nodes. A super-node acts as a landmark for each network domain.

Several researchers have previously proposed interesting dynamic load balancing and channel assignment schemes to overcome the congestion problem. Generic load balancing is a relatively old and very and well-researched area [13] [14]. Existing load-balancing algorithms proposed in the distributed systems literature are not appropriate for a peer-to-peer network. We find that load-based algorithms do not handle the heterogeneity that is typical in a peer-to-peer network. The work in [15] addressing the problem of load balancing in order to face “flash crowds”. However, they assume a static system architecture and global knowledge.

In [16], a load balancing scheme for Chord is introduced that is based on the notion of virtual servers. Each physical node can support multiple virtual servers. For overloaded nodes several strategies for moving virtual servers to underloaded nodes are discussed. However, the authors mention that there might be negative effects on search efficiency with their approach. Our work aims to solve a much more general problem, as stated above.

3. Design Description

In this section, we will present a brief overview of DNS and discuss our proposed method in more details.

3.1 Overview of DNS

The Domain Name Server (DNS) is a distributed Internet directory service which provides a distributed database of records spread across a semi-static hierarchy of servers. DNS is used mostly to translate human readable domain names (e.g. www.csie.cyut.edu.tw) to numerical IP addresses (e.g. 163.17.10.11) or vice versa aim to identify hosts on the Internet, and to control Internet email delivery [17, 18, and 19]. The DNS namespace is divided into a hierarchy of domains and each sub-domain can be locally administered independently by an authoritative name server that are responsible for keeping information about that domain up-to-date.

3.2 *P2P Registry* (*P2PReg*)

When new hosts are added to a domain, the administrator of DNS must edit the database manually to make the new hosts public. The other authoritative servers periodically fetch the contents of the master file whenever a new client joins or leaves in order to keep their records up-to-date.

To address this problem with least effort, we present a mechanism by creating a new service call *P2P Registry* (*P2PReg*) in DNS to enable peers register and leave the DNS server automatically. Our scheme is based on DNS domain hierarchy, although some adjustments have been made in order to make use in the domain tree.

When a host logs on to Internet, metadata on her entire library is automatically uploaded to the DNS server and added to the peer index as a peer (e.g. if the register user hosted as *“Personal Computer (PC) 1”*, the registered domain name represent as *“PC1_P2P”*) in order to distinguish from normal node. We suppose a peer who departs the DNS server either purposely or accidentally due to a failure. As a result of these situations, when a peer logs off, all of her library information is automatically removed from the peer index which done by *P2PReg* to reduce the bandwidth burden. At any given time, only the libraries of connected, or active, peers are in the index. While this policy allows the index to remain small and thereby increases query efficiency.

3.3 Peer Register and Peer Logoff

There is no connection between peer and DNS server at the beginning stage. The connection established whenever a host starts on a P2P program. When the connection is established, the peer informs DNS of the host name and request to register as a peer in P2P networks through *P2PReg* automatically.

No matter a peer logs off purposely or accidentally, all of her library information is automatically removed from the peer index which done by *P2PReg*. While this policy guarantee that peers in the index are active and connected to P2P network at any given time.

3.4 Content Discovery

We have packaging the resources being shared by peers as different services in order to avoid in returning large unneeded peer list in a request. We capture the interests of the peers through the number and types of files maintained and provided by the peers in P2P networks. As evident, services can be classified as categories, or simply software services that are device independent. For simplicity, we assume that there are only four topics of interest: Application Software (AS), Operation System (OS), Music (M), Movie (MV) and Others (O). If a register peer provides application software services, we designate an alias to the registered domain name represent *“PC1_P2P”* as *“AS_P2P”*. In case of a peer which holding many topics of services, the alias act to be longer, and thereby increases query efficiency, it also avoids confusion for user in identification of peer interest.

When DNS server receives a query from hosts which registered as peer, it first uses the local database to answer the query. If not enough answers are found or the requestor wish to look for more related peers, the service retrieval request is redirected along the hierarchical tree upward until finding a proper peer which has certain services or reach the TTL. Eventually, the searching process results in a list of peers that have the desired content.

With instant reply from DNS server, the requestor peer has a sufficient knowledge of the services that can access from these peers and uses the results to determine which peer is suitable to connect to. For each request, the peer utilizes the searching mechanism to discover the serving peers and related information, such as the uplink bandwidth and number of current uploading sessions of the peers.

Hence, a requesting peer can find all its potential serving peers and send query q to search for desired files (e.g. music file) directly to peers which holding related file. In this case, the reply message will follow the request path to reach the peer that initiated the request. These domain transfers are done using the special zone transfer query type in DNS (AXFR query type [12]).

4. Load Balancing

Load balancing is a critical issue for the scalability achievement of peer-to-peer networks. A good peer can be a hot spot if all of the requesting clients direct their request to the peer that considered as good peer. Nodes with a high degree of similar interests are considered good peers. Hence, a good peer may receive more load than it can handle. In order to ensure the high performance of the system, we have to avoid bottlenecks and balance the load across all system nodes. In fact, serving data discovery over local search eliminates the need to have each good peer become a hot spot. In this section, we will describe how to balance load in P2P systems with our proposed method in order to avoid hot spot problems naturally.

4.1 Finding Nearby Peers

In decentralized peer-to-peer network, peers are dynamic, often without permanent IP addresses and potentially located in more topologically diverse locations in the Internet. Clearly, a client would be desirable to find nearby peers that are well-connected without resorting to expensive network measurements. Finding nearby peers can greatly increase the efficiency of peer-to-peer communications.

Two peers are deemed to be topologically close if their IP addresses share a common address prefix. In our scheme, each DNS server has a local peer index for quickly finding local peer when a query is received. According to the inherent hierarchical architecture of DNS, we can ensure that the peer that discovered is topologically nearest to us, hence, yield better performance in searching a peer.

4.2 Replicas

Ideally, the request forwarding decision should be made at the global level. Requestor is willing to forward their request to a good peer indeed. Hence, global-wise searching results in an unfair allocation of user objects to peers and creates *hotspots* in certain parts of the zone as stated in Figure 1. The basic premise in our proposed networks is that any one of a set of "replica" nodes can provide the requested content, increasing the availability of interesting content without requiring the presence of any particular area. In fact, serving data discovery over local search eliminates the need to have each good peer become a hot spot and distributes load separately.

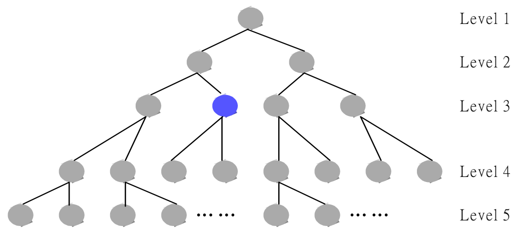
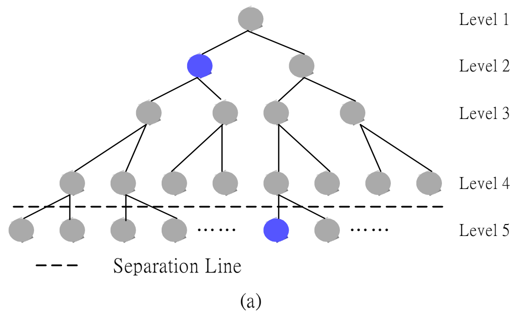


Figure 1 Global-wise searching



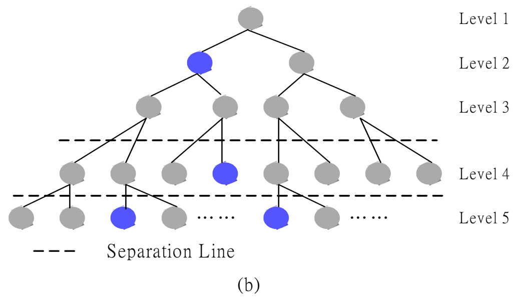
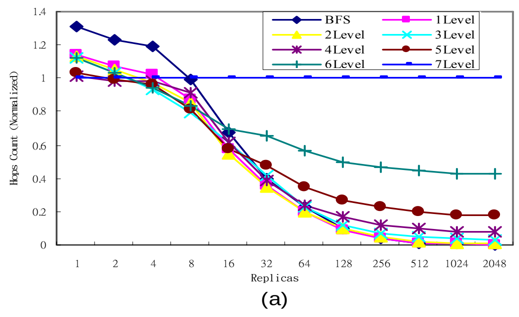


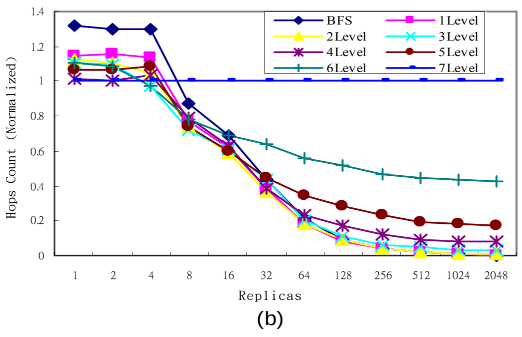
Figure 2 (a) two replicas (b) four replicas

For example, a complete tree with k level holds 2^k-1 node as each parent node holds two children nodes. When there is the only replica in this complete tree, yields 2^k-1 average searching time units. However, when the replica is 2^1, yields 2^{k-2} average searching time units. Yet, when the replica is 2^n, the average searching time units must be 2^{k-n-1}. Hence, we can conclude that average searching time units of a complete k level tree is x^{k-n-1} whenever the replicas is n with each parent node holding x children nodes. From the above analysis, we can set the searching area (how many levels) depends on practical situation and user requirement. It is quite sure that when the replicas reach an optimal level, the performance of content searching may approximate to global-wise searching. From Figure 2, it is obvious that the more replicas there are, the better the media delivery quality of this specific content object is.

5. Experimental Results

In this section, we implement our proposed architecture and algorithms. We present the results of simulation work to validate the viability of our approach. Three experiments were performed using different number of contents as 500 thousands, 1500 thousands and 2500 thousands, each experiment being tested with different searching area (global-wise searching, searching within one level, two levels and the like). In these experiments, we examine the performance of load balancing with a complete tree with seven levels in a P2P environment. For each experiment we will highlight the important conclusions and give a condensed, high-level explanation for the results.





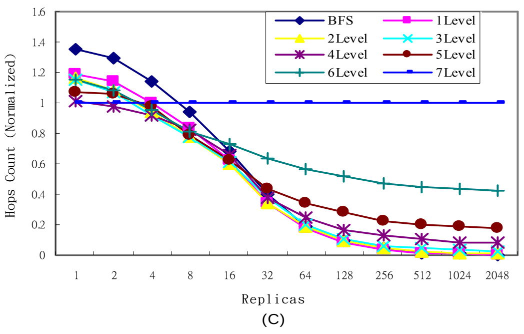


Figure 3 Total Hop count with varying the number of replica when the numbers of contents are (a) 500 (b) 1500 (c) 2500 thousands

5.1 Load balancing

In Figures 3, we plot the total number of hops count taken for a peer to retrieve the hot object over the number of replicas for each of the number of contents (500, 1500, 2500 thousands of contents) in P2P network. We see that hops count here means average hops numbers in location. It is clearly shown that our proposed method with local search is capable of distributing the load more evenly than Breadth First Search (BFS), especially, reducing the loads of the hot node. The load balancing is significantly better in this case, but we note that it is very close to that global-wise searching. Overall, this means that even given large amount of contents, the end result has not much different.

5.2 Standard Deviation

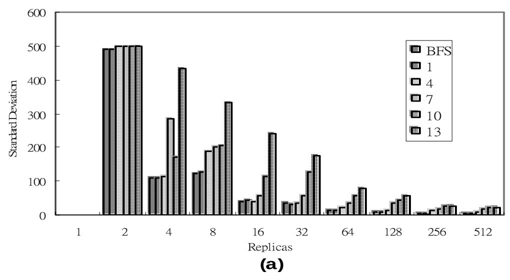
In order to ensure that the load distribution is equally, we consider the ability of our proposed method to balance the load locally by standard deviation to measure the gap between practical situation and ideally value. The standard deviation of different searching method and replicas is then given by the following formula:

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x}{n} - s_i\right)^2}$$

In our problem setting, n stands for the number of replicas, x stands for the load, and Si for the number of service in practical.

The standard deviation is equal to zero whenever the result of load distribution in practical is equal to the average in mathematical calculation. It is approaching ideally load balance status as the value of standard deviation more closely to zero.

Due to the reason that all nodes may send request for searching specification content, so the whole framework and the depth of the tree is different depends on each peer. The depth of a tree is the number of levels of links, not including the top. As the simulation we stated here, we adopt a complete tree with seven levels, so the largest depth of the nodes is 13 from the formula (n-1)*2 + 1.



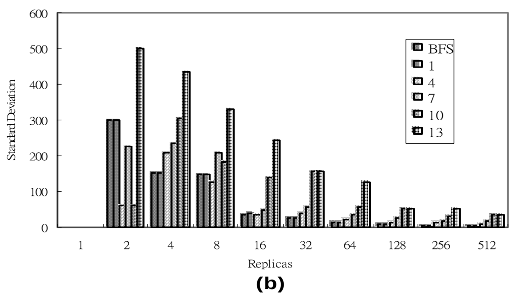


Figure 4 Standard deviation vs replicas in (a) 500 (b) 1500 thousands contents

In Figures 4 and 5, we plot the standard deviation for each of the number of contents (500, 1500, 2500 thousands of contents). In summary, our initial results suggest that our proposed architecture can improve overall system performance. Distributed streaming and hierarchical architecture are promising solutions to reduce load at individual peers while improving robustness.

Table 1 Standard Deviation with different searching area in 2500 thousands contents

Search Metho	Replicas									
	1	2	4	8	16	32	64	128	256	512
BFS	0	112	127	145	48.7	23.1	18.2	7.5	4.76	3.19
1 level	0	112	148	145	49.5	25.8	19.3	7.99	4.86	3.19
2 level	0	112	148	129	48.6	27.3	18.7	8.74	5.94	4.34
3 level	0	170	108	107	48.5	32.4	19.7	10.3	8.14	6.48
4 level	0	201	119	95.1	47.9	32	21.3	13.2	10.6	9.33
5 level	0	242	129	102	48.5	39.4	25.8	16.5	12.5	10.9
6 level	0	254	275	118	49.8	45	29.3	18.6	14.7	12.7
7 level	0	382	234	137	55.4	48.6	34.5	23.4	17.1	16.4
8 level	0	338	145	90.3	66.9	58.8	38.6	25.7	24	27
9 level	0	269	165	127	94.6	85.9	49.3	30.6	28.2	32.4
10 level	0	250	215	146	109	112	60.7	38.5	33.5	44.2
11 level	0	205	277	219	153	130	88.1	51.9	34.1	44.2
12 level	0	313	433	236	174	174	94.4	52.1	34.1	44.2
13 level	0	500	433	331	242	174	124	62.6	34.1	44.2

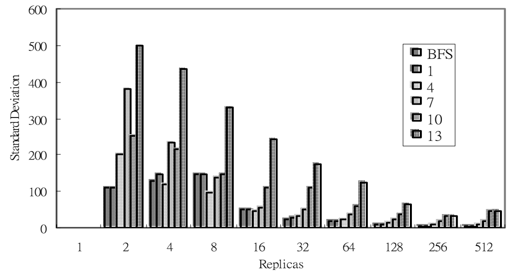


Figure 5 Standard Deviation vs Replicas in 2500 thousands contents

Our simulation study showed that serving data discovery over local search is indeed effective in improving the system performance significantly. We can conclude that when the number of replica reach an optimal level, the performance of content searching may approximating to global-wise searching by just searching a few levels. It is obvious that the more replicas there are, the better the media delivery quality of the specific content object is.

6. Conclusions

While efficient content search in decentralized P2P networks within a short duration is important, issues concerning load-balancing with a view towards faster access to data for normal users is rather challengeable. In this paper, we proposed a decentralized service discovery mechanism by harnessing all available resources in the P2P network and balance the load without decreasing user response time. We can conclude that when the numbers of replicas reach an optimal level, the performance of contents searches approximating to global-wise search by just searching a few levels. Our simulation study showed that serving data discovery over local search is indeed effective in improving the system performance significantly. In our design, it is obvious that the more replicas there are, the better the media delivery quality of the specific content object is.

References

[1] Napster, Inc., http://www.napster.com
[2] KaZaA, http://www.**kazaa**.com
[3] Young.K., "Look no Server", *Network*. pp.21, 22 & 26, March 1993.
[4] Simon.S., "Peer-to-Peer Network Management in an IBM SNA Network", *IEEE Network Magazine*, Vol. 5, pp. 30-34, March 1991.
[5] Peer-to-peer working group, *http://www.peer-to-peerwg.org/*.
[6] Open P2P website: http://www.openp2p.com.
[7] D. Tsoumakos and N. Roussopoulos, "A Comparison of Peer-to-Peer Search Methods", *International Workshop on the Web and Databases (WebDB)*, June 12-13, 2003.
[8] Gnutella homepage, *http://www.gnutella.wego.com*.
[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System", *In Work on Design Issues in Anonymity and Unobservability,* pages 311-320, July 2000.
[10] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks." *ICDCS'02,* 2002.
[11] A. Crespo, H. Garcia-Molina, "Routing Indices for Peer-to-Peer systems", *In Proc. ICDCS'02*.
[12] B.Y. Zhao, Y. Duan, and L. Huang, "Brocade: Landmark Routing on Overlay Networks", *In Proceedings of the 1st International Workshop on Peer-to-Peer Systems,* March 2002.
[13] R.M. Karp, M. Luby, and F. Meyer auf der Heide, "Efficient PRAM Simulation on a Distributed Memory Machine", *ACM STOC,* May 1992.
[14] M. Mitzenmacher, "The Power of Two Choices in Randomized Load Balancing", *Ph.D.thesis,* 1996.
[15] T. Stading, P. Maniatis, M. Baker, "Peer-to-Peer Cashing Schemes to Address Flash Crowds", *In Proc. IPTPS'02.*
[16] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. "Load Balancing in Structured P2P Systems", *In 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.
[17] P. Mockapetris, "Domain names—concepts and facilities", *Internet Request for Comments (RFC 1034),* February 1987.
[18] P. Mockapetris, "Domain names—Implementation and specification", *Internet Request for Comments (RFC 1035)*, February 1987.
[19] A. Kumar, J. Postel, C. Neuman, P, Danzig, and S. Miller, "Common DNS implementation error and suggested fixes", *Internet Request for Comments (RFC 1536)*, October 1993.