An IP Address Based Caching Scheme for Peer-to-Peer Networks

Ronaldo A. Ferreira, Ananth Grama and Suresh Jagannathan
Department of Computer Sciences, Purdue University
West Lafayette, IN, 47907-2066, USA
{rf,ayg,suresh}@cs.purdue.edu

***Abstract*— Distributed hash tables (DHTs), used in a number of current peer-to-peer systems, provide efficient mechanisms for resource location. Systems such as Chord, Pastry, CAN, and Tapestry provide strong guarantees that queries in the overlay network can be resolved in a bounded number of overlay hops, while preserving load balance among the peers. A key distinction in these systems is the way they handle locality in the underlying network. Topology-based node identifier assignment, proximity routing, and proximity neighbor selection are examples of heuristics used to minimize message delays in the underlying network. In this paper, we investigate the use of source IP addresses to enhance locality in overlay networks based on DHTs. We first show that a naive use of source IP address potentially leads to severe resource imbalance due to nonuniformity of peers over the IP space. We then present an effective caching scheme that combines a segment of the source IP with the queried hash-code to effectively localize access and affect replication. Using detailed experiments, we show that this scheme achieves performance gains of up to 41%, when compared to Pastry in combination with the proximity neighbor selection heuristic.**

I. Introduction

The past few years have seen considerable research activity in the area of peer-to-peer (P2P) systems and applications. File-sharing systems, such as Napster [10], and Gnutella [5], have gained immense popularity and attracted research attention. As has been well documented, [2], [18], [17], [6] these early systems had major scalability problems – Napster due to its use of a central server for queries, and Gnutella due to an inefficient search protocol that floods the entire network, or at least a significant part of it, in order to find an item of interest. The scale of P2P networks with a large number of participating nodes requires these networks to be highly scalable in terms of aggregate resource requirement as well as end user performance. The latter requirement translates to minimizing the number of hops a message must travel in order to satisfy a query.

A number of researchers have addressed the problem of scalability of P2P networks [19], [15], [24]. Systems such as Chord [19], Pastry [15], and Tapestry [24] provide a simple primitive for name resolution: given a file name, return the IP addresses of the nodes that currently have references to the file. To support this primitive, these systems rely on a *distributed hash table* (DHT) abstraction, and provide an upper bound on hop-count of $O(\log n)$, where n is the total number of nodes in the network. This upper bound is achieved using a small amount ($O(\log n)$) of routing information per node. Other systems such as CAN [12] support similar primitives, but have different upper bounds on hop-count subject to varying constraints on per-node routing information.

While much of the current work has focused on minimizing the number of overlay hops, the delay experienced by messages in the underlying network can be a major performance bottleneck. Since nodes and objects are identified by random strings, lookup messages may travel around the world before reaching a destination node that is in the same LAN as the source node. To try to minimize the effects of randomization, several heuristics have been incorporated into these systems. These include *proximity routing*, *topology-based ID assignment*, and *proximity neighbor selection* [14].

While these heuristics produce good results when compared to a standard implementation, they, nevertheless, rely exclusively on a single global overlay. Consequently, no guarantees can be provided that a popular object has its key installed very far from the majority of nodes that access it. Moreover, a response to a lookup message does not contain any locality information about the nodes holding a copy of the object. Thus, a node receiving a query response has no information about which nodes are close or far from it.

A natural way of building locality in an overlay network is to explore the addressing scheme of the underlying network, and to try to relate an overlay identifier with its host address in the underlying network. The allocation of IP addresses in the Internet has the desirable property that addresses that are numerically close to each other are, in most cases, physically proximate. In Section III, we explore this idea and show, using traces collected over the Internet, that a naive adaptation of IP addressing to overlay networks results in severe load imbalances resulting from a highly nonuniform population of the IP space by peers. Using the same traces, we investigate the prefix lengths of CIDR blocks in different autonomous systems, and show that the average prefix length is long enough that it is unlikely to produce good locality results in current DHT systems. We subsequently propose a caching scheme for Pastry that uses the geographic allocation of IP addresses to create different overlay domains where copies of objects are replicated. The caching scheme is implemented using additional routing information that is a constant factor larger than that required by Pastry. We show, using extensive simulation studies, that this scheme can reduce the response delay by up to 41%, when compared with an optimized version

of Pastry, depending on the cache hit ratio.

II. Background

Current routing schemes in P2P networks such as Chord [19], Pastry [15], and Tapestry [24], work by correcting a certain number of bits at each routing step. These schemes can be viewed as variants of hypercube (or dimension ordered) routing [9]. The routing scheme of Plaxton [11] for accessing nearby copies of objects in distributed systems is also related. In these systems, nodes and objects share the same address space. Nodes have their addresses assigned randomly and uniformly. This is generally achieved by computing a hash function on their IP addresses. Objects are identified by computing a hash function on their names. The uniform distribution of the identifiers (nodes and objects) is desirable for providing load balance in the system; that is, all nodes are expected to store roughly the same number of object keys. Due to space limitations, we present a brief description of Pastry and refer interested readers to the bibliography for other protocols.

In Pastry, objects and nodes are assigned random and uniformly distributed identifiers of length 128 bits. An object is stored in the node that is numerically closest to the object's identifier. Each node maintains routing information (overlay IDs and IP addresses) about a limited number of neighbors, with the size of the routing table varying depending on a configuration parameter, b, which indicates how many bits are resolved at each routing step. To route a message, each intermediate node, along the message path, tries to forward the message to a node whose node ID shares a prefix with the destination identifier that is at least b bits longer than the current node's shared prefix. If the routing table does not have any entries to such a node, the message is forwarded to a node that shares a prefix of the same length as the prefix shared by the current node, but is numerically closer to the destination. In addition to the routing table, each node maintains two other pieces of information: a *leaf set* and a *neighborhood set*. The leaf set stores information about l nodes that are numerically closest to the current node, with $l/2$ nodes having smaller IDs than the current node, and $l/2$ having larger IDs. This information is stored to provide reliable message routing, and is normally used when the routing table does not have an entry for a node that shares a prefix longer than the current node. The neighborhood set stores information about m nodes that are closest in the underlying network to the current node. This information is not normally used for routing, rather it is useful in applications that exploit locality properties. Figure 1 illustrates an example of the state stored at a particular Pastry node.

An arriving node needs to initialize its tables (routing table and leaf set) and inform the other nodes of its arrival. A new node initializes its state tables by contacting a node already present in the network, and asking it to route a join message to its node ID. The nodes along the path from the contact node to the node responsible for the new node's ID, send their state information to the new node. The new node uses this information and may contact additional nodes to complete its state tables.

Pastry State Information			
Leaf Set			
Smaller		Greater	
31001	31021	32021	32031
31200	32101	32101	32110
Routing Table			
0α	1α	2α	–
30α	31α	–	33α
–	321α	322α	323α
3200α	–	3202α	3203α
		–	
Neighborhood Set			
10223	03111	11330	23111
31001	32110	21333	22203

Fig. 1. Pastry state information stored at node with identifier 32012, and parameter b equal to 2. The digits are in base 4, and α is an arbitrary suffix.

The departure of a node x, either by voluntary termination or by failure, triggers an update at its neighbors. The nodes that are close in the ID space to the departing node must update their leaf sets. If x appears in the leaf set of a node y, y can replace x by asking the leaf set of the node with highest index in the same side of y's leaf set that x was present. If x appears in the routing table of a node w, this will be detected when w tries to use that entry. When detecting the failure of x, node w can use its leaf set to forward the message. To repair its routing table, node w can contact any node in the same row of the missing entry, and ask for its routing table. For a more detailed explanation of Pastry and its algorithms, we refer the reader to [15].

III. IP Addresses and Network Locality

Locality is a highly desirable property of DHT systems. This property refers to an overlay mapping in which nodes that are close in the overlay space are also close in the physical network. An overlay network that incorporates geographic locality, among other advantages, has the potential for isolating traffic based on the peer interests. For example, in a music sharing application, users from a particular region will most likely share the same interests. A user in China will most likely be interested in Chinese music, therefore, it would be desirable to isolate corresponding network traffic largely to this region.

We motivate our locality scheme with a brief description of the Internet organization and IP addressing. The Internet is composed of several autonomous systems, each autonomous system is further comprised of multiple LANs. Within a LAN, IP addresses are normally allocated sequentially. This means that if we could use the IP addresses of the nodes as overlay node identifiers, we would be able to build overlay networks with good locality properties. The problem with this approach is that the IP space is not uniformly populated by peers. This can be easily seen by considering just the addresses whose first octet values are greater than or equal to 224. IP addresses in this range are reserved for multicast and for future use. If we were to use IP addresses as overlay identifiers, this part of the ID space would have no node. All keys mapped to this region would have to be deterministically remapped, possibly

to one or two hosts that are closest to the region depending on the DHT scheme used. Another problem with a nonuniform address space is that the upper bound of $O(\log n)$ overlay hops cannot be guaranteed.

In the early days of the Internet, IP address assignment was completely controlled by the Internet Assigned Number Authority (IANA) and by the Internet Registry (IR). With its growth and globalization, this task was decentralized and delegated to Regional Internet Registries (RIRs). Each regional registry became responsible for large geographic areas, with the world being divided into four major areas: North America (ARIN), Asia and Pacific (APNIC), South and Central America (LACNIC), and Europe (RIPE NCC). The address allocation for the RIRs is in blocks of contiguous addresses, the size of blocks may vary, and the RIRs may have multiple blocks from different parts of the address space. For example, the addresses from 193.0.0.0 to 195.255.255.255 and from 80.0.0.0 to 82.255.255.255 are allocated to Europe. For more details about the Internet address allocation policy, we refer the reader to [8], [4], and for the current allocation we refer the reader to [7]. What can be observed with the current allocation policy is that IP addresses that share the first octet are normally allocated to the same geographic region. This is, of course, not always the case, with the initial class B allocation being a counter-example. Since the great majority of class B addresses were allocated to North America, we do not expect this to be a severe deviation from geographic allocation.

Instead of trying to build a unique overlay network using the entire IP address as an overlay identifier, we limit our approach to a caching scheme that can be incorporated into Pastry, or any one of the other DHT systems. The idea is that nodes in the network are grouped into different domains based on the first eight bits of their IP addresses. Each domain is an additional Pastry overlay, where the routing of messages is performed in the same manner as in the original scheme. For example, all nodes with IP addresses with first octet 128 are placed in the same domain. In this manner, we group the entire network into as many domains as there are unique first octets. When a node joins or leaves the network, it must join or leave both the main overlay network and the cache network. Each domain has a *rendezvous point* in the main network that can be contacted by a new node to find nodes in the same domain. Both routing tables (main and cache) can be constructed using the proximity neighbor selection heuristic. When searching an object, nodes issue a query in the cache network. If it is found, it is immediately returned, otherwise, the query is reissued in the main overlay network. Once an object is accessed in one domain, it is automatically replicated in the cache network for that domain. With this scheme, we can have objects replicated in as many domains as the number of distinct eight-bit prefixes present in the network. In practice, however, due to bounded buffer (cache) sizes this does not generally occur.

After grouping nodes into domains, we investigate the possibility of using the remaining 24 bits of the IP addresses as overlay identifiers for the cache network. Such an identifier allocation has the desirable property that as we progress through the routing process we move closer to the prefix lengths of autonomous systems, with the last few hops being inside one particular autonomous system. Such a scheme, if successful, would avoid the higher delays that dominate the last hops in Pastry [1].

A key issue in this approach is the load imbalance that results from the nonuniform occupancy of the IP space and a restrictive hash-code. To further explore this issue, we populate the network with clients from Gnutella (Zeinalipour-Yazti and Folias [22] collected and analyzed more than 56 million messages from a Gnutella network in June 2002 and have kindly made these traces available to us). Using these traces, we isolate the IP addresses of the nodes present in the network. Since it is possible to have bogus advertisements in Gnutella, we first validate the IP addresses in the trace using a whois server[1] to map them to their respective autonomous systems, verifying, in this way, their allocation to an existing organization. From the total addresses present in the trace, more than 62,000 are valid IP addresses. We classify the IP addresses into domains based on the first eight bits and build a Pastry network for each domain. The node identifiers for each network are the last 24 bits of the IP addresses. For each node in the network, we issue a number of queries for randomly and uniformly selected destinations in the address space $[0 - 2^{24}]$ to analyze how many queries each node receives. Figure 2(a) illustrates the results for 2,420 nodes with IP addresses whose first octet is 200, the number of queries originating at each node is fixed at 20 per node, for a total of 48,400 queries. Figure 2(b) shows an experiment with the same number of nodes and queries, however, here the node identifiers are drawn from a uniform distribution.

We note in the case of a uniform node distribution that each node is the target of no more than 125 queries (indeed it can be shown that this number varies roughly as $k \log n$ for k source queries generated at each node and n being the total number of nodes). While this situation is reasonably implementable, notice what happens when we use actual data from nodes on the Gnutella network. In this case, the number of queries that are mapped to a target node may be as high as 3000! Recall that each source node only originates 20 queries and under optimal load distribution, each peer must also be a target of only 20 queries. Evidently, under real Gnutella population, load imbalance may be as high as two orders of magnitude with this mapping scheme. A direct consequence of this mapping is that a few nodes will be overloaded with queries, thus posing a resource (network and replication/disk) bottleneck. We observe similar results for the other IP domains as well.

As previously mentioned, another source of locality in the Internet is its organization into autonomous systems (ASs). Hosts in an AS are generally located in a limited geographic area. With this in mind, we investigate, in the Gnutella trace, the prefix length of CIDR (Classless Inter Domain Routing) blocks in the autonomous systems. The trace contains 3,264

[1] whois.radb.net

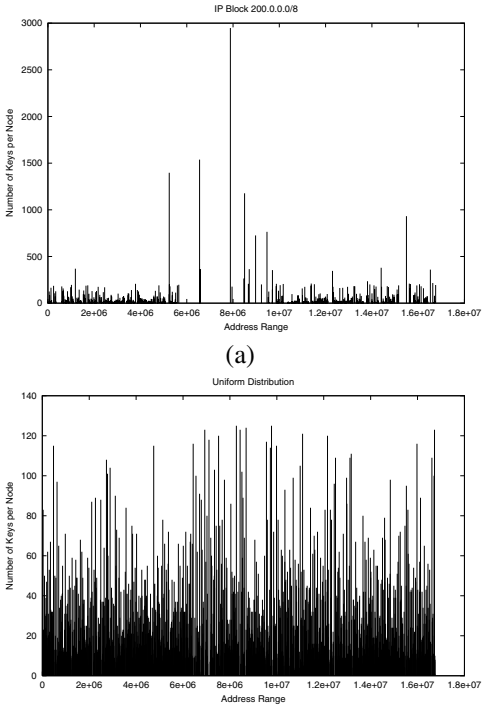

Fig. 2. (a) Observed (actual) distribution of Gnutella clients and key mapping in the 200.x.y.z domain; (b) Synthetic (uniform) distribution of clients with corresponding key mapping.

different ASs, which constitutes a representative sample of the Internet. The average prefix length of the AS's CIDR blocks is a little over 19 bits. This implies that if we use a routing scheme like Pastry, where we resolve four bits at each hop, we would need on average 5 hops to get inside an AS. Such a scheme would become effective only if the number of nodes in the network were greater than 16^5 (approximately sixteen million). As shown in [15], the expected number of overlay hops in Pastry is $\log_{16} n$, where n is the total number of nodes.

We can conclude from the above discussion that, other than the geographic location encompassed in the first eight bits of the IP addresses, it is unlikely that we can employ IP addresses to build locality into systems such as Pastry (where routing is performed by bit correction) without compromising the load balance of the overlay network. Consequently, we use a localization scheme that groups peers into domains based on the first eight bits of their IP addresses. These domains are used as caching networks within the global network. Inside the cache network, the node identifiers are randomized in the same way as in the main network, avoiding, in this way, load imbalances at peers. We now characterize the performance of such a caching scheme using detailed experimental analyses.

IV. Experimental Results

In this section, we present simulation results for our caching scheme. We have developed a network simulator that implements the routing scheme of Pastry for the overlay networks (cache and main), and emulates the underlying network. The topology of the underlying network is generated using the Georgia Tech. transit-stub network topology model [21]. All experiments are performed on a 32-processor Origin 2000 with 16GB of memory, running IRIX64.

The topology generated by the GT-ITM topology generator does not contain node addresses that can be linked to real Internet addresses. In our experiments we use the IP addresses collected from the Gnutella network. To assign these addresses to the generated topology, we use the following approach: a transit network in the generated topology corresponds to a geographic region whose nodes' IP addresses share the first eight bits; a stub network, which is linked to a transit network, corresponds to an autonomous system whose addresses share the same prefix as the transit network. The IP addresses are grouped in the autonomous systems obtained from the mapping using a whois server, as described in the previous section. Each autonomous system in the trace is linked to a stub node, and the IP addresses of the autonomous system are assigned to LANs connected to the stub node. The resulting topology used in the experiments below has 117,825 nodes. Link delays are random values within the following intervals: the delay of an edge between two transit domains is in the interval [20-80]ms; the delay of a transit-transit edge in the same transit domain is in the interval [3-23]ms; the delay of a transit-stub edge is in the interval [2-7]ms; the delay of a stub-stub edge is in the interval [1-4]ms; and the delay of an edge connecting a host to a stub node is fixed at $1ms$. These figures are similar to those used by Xu et al. [20].

The evaluation of a caching scheme requires appropriate dimensioning of the storage available for caching at each node, and a realistic workload. Since there are no publicly available traces that contain file sizes for existing peer-to-peer systems, we use web proxy logs for the distribution of file sizes in the network. The same approach was used to validate PAST in [16]. We use a set of web proxy logs from NLANR[2] corresponding to eight consecutive days in February 2003. The trace contains references to 500,258 unique URLs, with mean file size 5,144 bytes, median file size 1,663 bytes, largest file size 15,002,466 bytes, and smallest file size 17 bytes. The total storage requirement of the files in the trace is 2.4GBytes.

The overlay nodes are selected randomly from the Gnutella trace, resulting in 75 different domains. The distribution of nodes per domain is illustrated in Figure 3. The source nodes of the queries were chosen randomly and uniformly, and the objects are accessed according to a Zipf-like distribution, with the ranks of the objects being determined by their position in the original trace.

As described in the previous section, in our scheme a query that cannot be satisfied by the cache network causes an access to the main overlay network and the automatic caching of the object in the cache network of the node issuing the query. Here, we assume that the objects being cached are the actual files and not the keys, as generally assumed in Pastry. Due

[2]http://www.ircache.nlanr.net

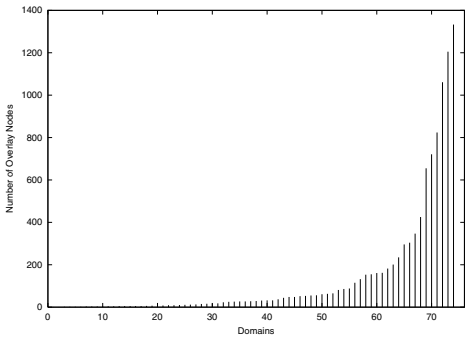

Fig. 3. Distribution of nodes per cache domain.

to the storage requirements of our workload, and to guarantee that the simulation would reach a steady state, we restrict the size of the overlay network to 10,000 nodes, with each node having a cache of size 5MB. If the number of overlay nodes is increased beyond this value, without reducing the cache size per node to insignificant values, the total cache capacity available in the network exceeds the total storage requirement, which causes the cache hit ratio to keep increasing with the number of queries. With the parameters above, we are able to reach steady state after 75 million queries.

We measure the impact of the cache hit ratio on the response delay for the queries. The Pastry parameters are set to: $b = 4$, and leaf set size $l = 32$, the neighborhood set is not used. LRU (Least-Recently-Used) is the cache replacement policy used on the nodes in the cache overlays. Figure 4 summarizes the results for the delay measurements. For each value of α in the Zipf distribution, we run simulations of the network without and with cache. The parameter α has no effect on queries performed only over the main overlay network. This is because the sources are chosen uniformly, there is no replication, and the objects are always placed in the nodes numerically closest to the objects' IDs. The performance gain shown in the table is defined as: $g = \frac{d_1 - d_2}{d_1}$, where d_1 is the average query delay in the network without cache, and d_2 is the average query delay in the network with cache. Observe that both overlay networks implement the proximity neighbor selection heuristic to minimize the delays in the underlying network. The results are very promising, since the gains are relative to a topology-aware scheme with relative delay penalties on average less than 2, as shown in [1].

α Parameter	Cache Hit Ratio	Gain
0.75	76%	31.0%
0.80	79%	33.5%
0.85	81%	36.0%
0.90	83%	38.7%
0.95	86%	41.3%

Fig. 4. Simulation results illustrating percent improvement in response time and corresponding cache hit ratios.

V. Related Work

Three major approaches have been proposed for topology-aware overlay construction in DHT networks: *proximity routing*, *topology-based node ID assignment*, and *proximity neighbor selection* [14].

In proximity routing, each node determines the next hop for a message taking into consideration not only the node that makes more progress in the resolution of the virtual identifier, but also the proximity of its neighbors in the underlying network. This technique has been used with some success in CFS [3]. The key idea is to select, among all neighbors, the node that is closest in the underlying network or one that balances the proximity with progress in the resolution of the virtual ID. The problem with this approach is that the number of overlay hops may increase considerably.

In topology-based node ID assignment, the overlay node IDs are chosen based on the location of the nodes in the underlying network. The idea here is that nodes that are close in the underlying network are assigned overlay IDs that are numerically close. Ratnasamy et.al [13] demonstrate the use of this technique in CAN. In this particular example, before joining the network, nodes measure their distances to a set of landmarks and use the measurements to position themselves in the multi-dimensional address space of CAN. Even though this technique is able to achieve considerable reduction in routing delays, it also presents a few problems. By biasing identifier assignments, it potentially destroys the uniform distribution of nodes in the ID space.

In proximity neighbor selection, the topology of the underlying network is explored during the construction and maintenance of the routing table. For a given routing table entry, the idea is to choose, among all the nodes that qualify for that entry, the one that is closest in the underlying network to the current node. This heuristic is suitable for prefix-based protocols like Tapestry and Pastry. In these systems, the first rows of the routing tables have many options for each entry, with latter rows having exponentially fewer options. As a consequence, the total delay during the routing process is dominated by the last hop. Castro et. al.[1] present a detailed study of proximity neighbor selection in Pastry. They show that the heuristic results in small relative delay penalties, without compromising the load balance of the system. Relative delay penalty is defined as the ratio of the delay experienced by a Pastry message to the delay from its source and destination nodes in the underlying network. We use this approach in the routing tables of the cache overlay nodes.

The use of a two-level architecture to improve overlay performance is not new. Brocade [23] uses a secondary overlay network of supernodes. The supernodes are nodes with good capacity and network connectivity and are assumed to be close to network access points such as routers. Nodes inside an autonomous system use the supernodes to access objects in the global overlay. Our approach differs from Brocade in several important respects. In Brocade, a normal node (not a supernode) participates in the overlay via supernodes. A

normal node first needs to contact a supernode and to ask it to route its messages. Supernodes have to keep information about all overlay nodes inside their autonomous systems. The Brocade organization is basically an overlay of servers that have several clients connected to them. There is no deterministic distributed way for a normal node to find a supernode. It is assumed that the supernodes are able to snoop the underlying network and detect overlay traffic, or that the supernodes have well known DNS names.

Xu et al. [20] also proposes a two level overlay, consisting of one auxiliary overlay, called *expressway*, composed of supernodes, as in Brocade, and a global overlay. Nodes in the expressway exchange routing information, much in the same way as routers exchange BGP reports in the Internet. A key difference w.r.t. our scheme is that the auxiliary network in [20] is intended to speed up the communication of nodes far apart in the Internet. Our auxiliary network (cache overlay) is intended to speed up the communication among nodes that are close to each other, taking advantage of possible common interests, and reducing traffic in the global network. Furthermore, the expressway requires significant work to be performed by the supernodes. The supernodes try to emulate, in the overlay network, the BGP routing protocol of the Internet, by running a distance-vector routing protocol to summarize routes.

VI. Concluding Remarks

In this paper, we investigate the use of IP addresses to incorporate locality into structured P2P networks. We show that this approach has some fundamental limitations with respect to load imbalance and associated resource requirements. This is a consequence of the nonuniform occupancy of the address space by overlay nodes. We subsequently propose a caching scheme that uses the geographic allocation of IP addresses to partition nodes into replication domains. This approach yields excellent results, when compared to an overlay network without replication. Specifically, we demonstrate performance gains of up to 41%, depending on the cache hit ratio, for realistic network and workload scenarios. We are currently investigating and comparing other aspects of our caching scheme with existing systems. Link congestion, aggregate resource utilization, and replication factors of objects are all parameters that we expect to quantify in the near future.

Acknowledgments

This research is partially funded by the National Science Foundation grants ACI 0082834, ACI 9875899, and EIA 0216131. The first author is partially funded by CNPq and UFMS, Brazil.

References

[1] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Exploiting Network Proximity in Peer-to-Peer Overalay Networks, Technical Report MSR-TR-2002-82, 2002.

[2] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *ACM SIGCOMM'02 Conference*, 2002.

[3] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-Area Cooperative Storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Lake Louise, Canada, October 2001.

[4] E. Gerich. Guidelines for Management of IP Address Space, RFC 1466, IETF, May 1993.

[5] Gnutella. http://gnutella.wego.com/.

[6] Theodore Hong. Performance. In *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[7] http://www.iana.org/assignments/ipv4-address space.

[8] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel. Internet Registry IP Allocation Guidelines., RFC 2050, IETF, November 1996.

[9] Frank Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.

[10] Napster. http://www.napster.com/.

[11] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Theory of Computing Systems*, 32:241–280, 1999.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 247–254, San Diego, CA, August 2001.

[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proceedings of IEEE INFOCOM 2002*, New Yourk, NY, June 2002.

[14] S. Ratnasamy, S. Shenker, and I. Stoica. Routing Algorithms for DHTs: Some Open Questions. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Boston, MA, March 2002.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 247–254, San Diego, CA, August 2001.

[16] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Lake Louise, Canada, October 2001.

[17] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[18] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *ACM Transactions on Networking*, *To appear*.

[19] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 149–160, San Diego, CA, August 2001.

[20] Z. Xu, M. Mahalingam, and M. Karlsson. Turning Heterogeneity into an Advantage in Overlay Routing. In *Proceedings of the IEEE INFOCOM 2003*, San Francisco, CA, April 2003.

[21] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE INFOCOM 1996*, March 1996.

[22] D. Zeinalipour-Yazti and T. Folias. Quantitative Analysis of the Gnutella Network Traffic, Technical Report, Department of Computer Science, University of California at Riverside, June 2002.

[23] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. Brocade: Landmark Routing on Overlay Networks. In *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.

[24] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-0101141, UC Berkeley, Computer Science Division, April 2001.