Providing Administrative Control and Autonomy in Structured Peer-to-Peer Overlays

Alan Mislove Peter Druschel
{amislove, druschel}@rice.edu

Rice University, Houston, TX, USA

Abstract

Structured peer-to-peer (p2p) overlay networks provide a decentralized, self-organizing substrate for distributed applications, and support powerful abstractions such as distributed hash tables (DHTs) and group communication. However, in most of these systems, lack of control over key placement and routing paths raises concerns over autonomy, administrative control, and accountability by participating organizations. Additionally, structured p2p overlays tend to assume global connectivity while in reality, network address translation and firewalls limit connectivity among hosts in different organizations. In this paper, we present a general technique that lends structured overlays content/path locality and support for NATs and firewalls. Instances of conventional overlays are configured to form a hierarchy of identifier spaces (i.e., rings) that reflects administrative boundaries and respects connectivity constraints among networks.

1 Introduction

Structured peer-to-peer (p2p) overlay networks provide a decentralized, self-organizing substrate for distributed applications, and support powerful abstractions such as distributed hash tables (DHTs) and group communication [13, 18, 19, 20, 22, 15]. Most of these systems use randomized object keys and node identifiers, which yields good load balancing and robustness to failures. However, in such overlays, applications cannot ensure that a key is stored in the inserter's own organization, a property known as *content locality* [13]. Likewise, one cannot ensure that a query which originates within an organization *O* and is resolved to a key that is stored within *O* is routed along a path that remains entirely within *O*, a property known as *path locality* [13]. In an open system where participating organizations have conflicting interests, this lack of control can raise concerns about autonomy, administrative authority and accountability [13].

The SkipNet [13] structured overlay protocol addresses this problem by assigning node identifiers and keys based on the owner's organization and/or location, thus ensuring content and path locality. However, this choice constrains the design space for overlay protocols and the approach has some weaknesses with respect to security. An attacker, for instance, can intercept traffic from and to an organization by creating nodes that are near the victim organization in the namespace. Our aim is to offer an alternative that achieves content and path locality while maintaining the advantages of random identifier assignment and leveraging other work on structured overlay protocols, e.g. on secure routing.

Additionally, most structured p2p overlay protocols assume that the underlying network is fully connected. In the real Internet, however, communication among host in different organizations is often constrained. Security firewalls and network address translation (NAT) often prevent nodes exterior to an organization from contacting interior ones.

In this paper, we present a general technique to configure structured p2p overlay networks into a hierarchy of identifier spaces that reflects administrative and organizational domains. The technique provides content locality, path locality, and respects connectivity constraints along organizational boundaries. Our solution generalizes existing protocols with a single id space, thus leveraging prior work on all aspects of structured p2p overlays, including secure routing [2].

The rest of this paper is organized as follows. Section 2 describes in detail the design of our system and explains how messages are routed across multiple rings. Section 3 discusses the costs, benefits and limitations of our technique. Section 4 details related work and Section 5 concludes.

2 Design

In this section, we describe a hierarchical configuration of overlays that reflects organizational structure and connectivity constraints. A *multiring* protocol stitches together the rings and implements global routing and lookup. To applications, the entire hierarchy appears as a single instance of a structured overlay network that spans multiple organizations and networks. The design can be applied to any structured overlay protocol that supports the key-based routing (KBR) API defined in Dabek et al. [7].

Our design relies on a group anycast mechanism, such as Scribe [5, 6]. Scribe maintains spanning trees consisting of the overlay routes from group member nodes towards the overlay node that is responsible for the group's identifier. These trees are then used to implement multicast and anycast. Scribe can be implemented on top of any structured overlay that supports the KBR API. If the underlying overlay protocol uses a technique such as proximity neighbor selection [3, 12], then the Scribe trees are efficient in terms of network proximity and anycast messages are delivered to a nearby group member [6].

For convenience, we will refer to an instance of a structured overlay as a "ring", because the identifier spaces of protocols like Chord and Pastry form a ring. However, we emphasize that our design can be equally applied to structured overlay protocols whose identifier spaces do not form a ring, including CAN, Tapestry, and Kademlia [17, 22, 15].

Figure 1 shows how our multiring protocol is layered above the KBR API of the overlay protocols that implement the individual rings. Shown at the right is a node that acts as a gateway between the rings. The instances of structured overlays that run in each ring are completely independent. In fact, different protocols can run in the different rings, as long as they support the KBR API.

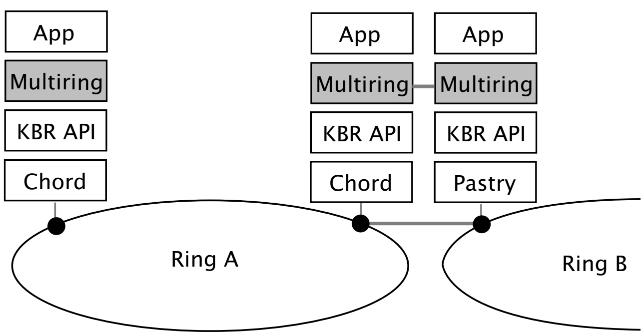

Figure 1: Diagram of application layers. The two nodes on the right are actually instances of the same node in two different rings.

2.1 Ring structure

The system forms a tree of rings. Typically, the tree consists of just two layers, namely a *global ring* as the root and *organizational rings* at the lower level. Each ring has a globally unique *ringId*, which is known to all members of the ring. The global ring has a well-known ringId consisting of all zeroes. It is assumed that all members of a given ring are fully connected in the physical network, i.e., they are not separated by firewalls or NAT boxes.

All nodes in the entire system join the global ring, unless they are connected behind a firewall or a NAT. In addition, each node joins a ring consisting of all the nodes that belong to a given organization. A node is permitted to route messages and perform other operations only in rings in which it is a member.

The global ring is used primarily to route inter-organizational queries and to enable global lookup of keys, while application keys are stored in the non-global rings. Each non-global ring defines sets of nodes that wish to ensure content and path locality for keys that they are inserting into the overlay. In addition, a non-global ring may also define a set of nodes that are connected to the Internet through a firewall or NAT box.

An example configuration is shown in Figure 2. The nodes connected by lines are actually instances of the same node, running in different rings. Ring A7 consists of nodes in an organization that are fully connected to the Internet. Thus, each node is also a member of the global ring. Ring 77 represents a set of nodes behind a firewall. Here, only two nodes can join the global ring, namely the firewall gateway nodes.

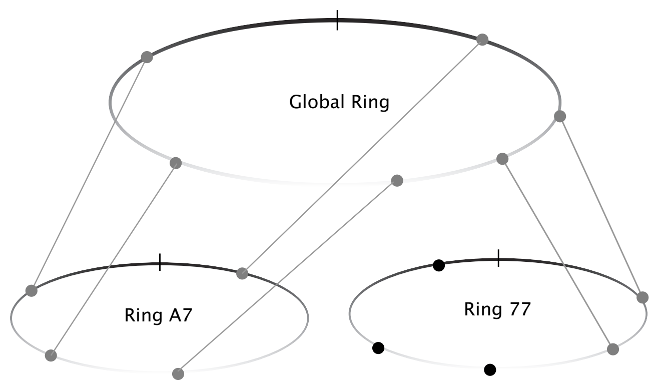

Figure 2: Example of a ring structure. Nodes shown in gray are instances of the same node in multiple rings, and nodes in black are only in a single ring due to a firewall.

2.2 Gateway Nodes

A node that is a member of more than one ring is a *gateway node*. Such a node supports multiple virtual overlay nodes, one in each ring, but uses the same node identifier (id) in each ring. Gateway nodes can forward messages between rings, as described in the next section. In Figure 2 above, all of the nodes in ringId *A7* are gateway nodes between the global ring and ring *A7*. To maximize load balance and fault tolerance, all nodes are expected to serve as gateway nodes, unless connectivity limitations (firewalls and NAT boxes) prevent it.

Gateway nodes announce themselves to other members of the rings in which they participate by subscribing to an anycast (Scribe) group in each of the rings. The group identifiers of these groups are the ringIds of the associated rings. In Figure 2 for instance, a node *M* that is a member of both the global ring and *A7*, joins the Scribe groups:

Scribe group *A700...0* in the global ring
Scribe group *0000...0* in ringId *A7*

2.3 Routing

Next, we describe how messages are routed in the system. We assume that each message carries, in addition to a target key, the ringId of the ring in which the key is stored. In the subsequent section, we will show how to obtain these ringIds.

Recall that each node knows the ringIds of all rings in which it is a member. If the target ringId of a message equals one of these ringIds, the node simply forwards the message to the corresponding ring. From that point on, the message is routed according to the structured overlay protocol within that target ring.

Otherwise, the node needs to locate a gateway node to the target ring, which is accomplished via anycast. If the node is a member of the global ring, then it forwards the message via anycast in the global ring to the group that corresponds to the desired ringId. The message will be delivered to a gateway node for the target ring that is close in the physical network, among all such gateway nodes. This gateway node then forwards the data into the target ring, and routing proceeds as before.

If the node is not a member of the global ring, then it forwards the message to a gateway node to the global ring by sending the message via anycast to the group whose identifier corresponds to the ringId of the global ring. Routing then proceeds as described above.

As an optimization, it is possible for nodes to cache the IP addresses of gateway nodes they have previously obtained. Should the cached information prove stale, a new gateway node can be located via anycast. This optimization drastically reduces the need for anycast messages during routing.

2.4 Global Lookup

In the previous discussion, we assumed that messages carry both a key and the ringId of the ring in which the key is stored. In practice, however, applications often wish to lookup a key without knowledge of where the key is stored. For instance, keys are often derived from the hash of a textual name provided by a human user. In this case, the ring in which the key is stored may be unknown.

The following mechanism is designed to enable the global lookup of keys. When a key is inserted into a non-global ring and that key should be visible at global scope, a special indirection record is inserted into the global ring that associates the key with the ringId(s) of the non-global ring(s) where (replicas of) the key is(are) stored. The ringId(s) of a key can now be looked up in the global ring. Note that indirection records are the only data that can be stored in the global ring.

2.5 Multi-level Ring Hierarchies

We believe that a two-level ring hierarchy is sufficient in the majority of cases. Nevertheless, there may be situations where more levels of hierarchy are useful. For instance, a world-wide organization with multiple campuses may wish to create multiple rings for each of its locations in order to achieve more fine-grained content locality. In these cases, it may be advantageous to group these machines into subrings of the organization's ring, further scoping content and path locality.

In order to provide for such extensions, the ring hierarchy described above can be naturally extended. To do so, we view ringIds as a sequence of digits in a configurable base b, and each level of ring hierarchy will append an extra digit onto the parent ring's ringId. Thus, organizations which own a given ringId can dynamically create new rings by appending digits to their ringId.

The routing algorithm can be generalized to work in a multi-level hierarchy as follows. When routing to a destination ring R, the node first checks to see if it is a member of R. If so, it simply routes the message in R using the normal overlay routing.

If the node is not a member of R, it must forward the message to a gateway. First, however, the node must choose which of the rings in which it is a member to forward the message in. This is done by comparing the shared prefix length of each local ringId and R and picking the ring with the longest shared prefix. In the case of multiple ringIds with the longest prefix, the node should pick the shortest one in total length. This process guarantees that the node picks the local ring which is "closest" to the destination ring R.

Once the node has chosen which local ring L to send the message in, it the must determine if it should route the message up, towards the global ring, or down. This is an easy computation, as it is dependent only upon the length of the shared prefix of L and R.

```
route(dst, msg) {
  if (local == dst) {
    route_normally(msg)
  } else {
    len = length(local)

    if (dst.hasPrefix(local))
      forward(substring(dst, len+1), msg)
    else
      forward(substring(local, len-1), msg)
  }
}
```

Figure 3: The pseudocode for routing between rings, which is executed at each node along the route.

If R has L as a prefix, the node should route the message downwards since R is "below" this ring. Thus, the node should forward the message via an anycast to the Scribe group rooted at $substring(R, length(L)+1)$. The gateway node which receives the message can then use the routing algorithm again in the other ring.

If R does not have L as a prefix, the node should route the message upwards, towards the global ring. This is done by routing the message to the parent ring, or to ring with ringId $substring(L, length(L) - 1)$. As can clearly be seen, messages are routed efficiently by forwarding the message until a ring which is a prefix of the destination ring is found, and then routing the message downwards towards the destination ring.

The pseudo-code for routing a message *msg* to the ringId *dst* at a node in ringId *local* is shown in Figure 3. Figure 4, below, shows an example a node in ring *D1A8* routing to a location in the ring *63*.

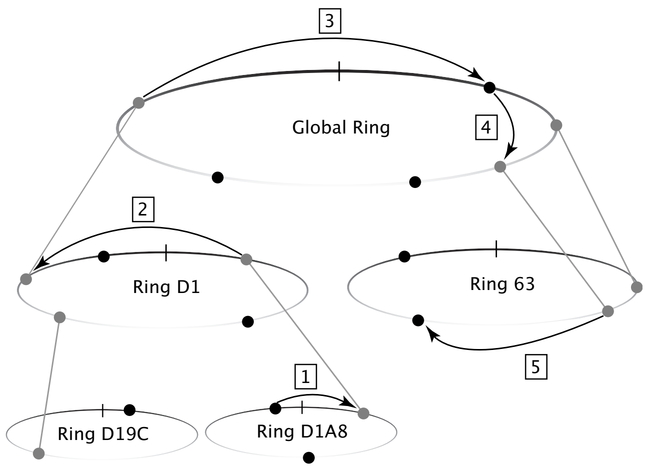

Figure 4: Diagram of a the routing process with multiple levels of hierarchy. Gray nodes are gateways, which exist in multiple rings and route between them. Numbers 1-5 note the steps in routing.

3 Discussion

In this section, we discuss the costs, benefits and limitations of our proposed technique.

Partitioning an overlay network into organizational rings affords content and path locality, but reduces diversity among the set of nodes that store a given key. If the diversity of nodes in an organizational ring is not sufficient to provide the desired durability of keys, then replicas must be stored in different organizations' rings via an appropriate replica placement strategy. The lookup of keys replicated in this manner proceeds by first looking up the key in the local ring and should that fail, looking up the key's indirection record in the global ring, which refers to other rings containing the key.

In deciding on the ring structure, organizations need to strike the right balance between content locality and diversity. An organizational ring should be large enough to contain nodes with different physical network links to the Internet, independent power sources and locations in different buildings if not cities.

To retain the robustness of a single global overlay network, all nodes without connectivity constraints should join the global ring. All such nodes act as gateway nodes among the rings, thus ensuring load balancing, efficient routing across rings, and fault tolerance. In the case of rings behind firewalls, some loss of these properties is unavoidable due to the limitations of the physical network.

In an organizational ring, keys can be inserted only by a member of the same ring, providing organizations with autonomy and authority over their resources. Likewise, it alleviates the threat of denial-of-service attacks that aim at filling up the storage space, which are a security threat in open rings. However, nodes that participate in the global ring must store indirection records and forward routing request on behalf of arbitrary other organizations. This is unavoidable as some resource sharing is central to the idea of a cooperative overlay network. Our system limits data stored in the global ring to indirection records and due to the small size of these records, space-filling attacks are more difficult to mount.

3.1 Performance

The cost of routing a message within a given ring depends on the overlay protocol used within the ring, typically $O(log\,\mathrm{N})$ routing hops and, if proximity neighbor selection is used, a delay stretch below two.

Routing a message between two non-global rings requires, in the worst case, three intra-ring routes plus two anycast transmissions. However, caching of gateway nodes eliminates the two anycasts in most cases. Also, all nodes in non-global rings without connectivity constraints are gateways to the global ring, thus eliminating the need for one anycast and one overlay route if the source is such a node.

With proximity neighbor selection used in the overlay protocols, the gateways located via anycast are nearby in the physical network. Thus, the gateway nodes are likely to lie along or near the shortest path from source to destination node in the physical network. Combined with an expected delay stretch of under two for the route segments between the gateways, this suggests that the total delay stretch for an inter-ring route is also around two in the common case. We are currently in the process of verifying this hypothesis experimentally.

In terms of maintenance, the principal overhead of our system results from the fact that gateways nodes must join multiple rings, and thus require additional control messages for maintaining the routing state in each ring. In what we consider the most common case of a two-level hierarchy, the worst case overhead is twice that of a single ring. The overhead is lower when many nodes are behind firewalls or NAT boxes. Moreover, a large fraction of the additional control traffic for maintaining non-global rings remains internal to a given organization. Since the basic maintenance overhead of the most efficient structured overlays has been reduced to less than half a message per second and node [1], we believe that the overhead imposed by hierarchical rings is not a concern.

In addition, various optimizations are possible that exploit overlap among the routing state of a given node in the different rings. For instance, the size of the neighbor set (e.g., leaf set in Pastry, successor set in Chord) can be reduced in non-global rings, as the global ring can be used to repair a non-global ring that has become disconnected due to many simultaneous node failures. Since the details depend on the specific overlay protocols used in each ring, we don't discuss them here.

3.2 Security

Our system does not constrain the structured overlay protocols used in the individual rings. This allows us to leverage existing work on secure routing in the presence of malicious participants [2]. The nodeId certificates used in this work can be extended to bind a node's IP address to both its nodeId and ringId. When a node joins an anycast group or offers to forward a request into a different ring, it presents its certificate demonstrating that it is actually a member of the ring in question. With both nodeId and ringIds certified, the techniques described in Castro et al. [2] can then be applied to our hierarchical ring structure. A full analysis, however, remains the subject of ongoing work.

3.3 Locality-based Id Assignment

The main alternative to our proposed technique of hierarchical rings is the use of locality-based id and key assignment, as used, for instance, in SkipNet and a version of CAN [18]. The advantage of these techniques is that they can achieve content and path locality without the additional maintenance overhead of multiple rings. On the other hand, obtaining robustness and load balancing requires a different protocol design (as in SkipNet), and these systems are more vulnerable to certain security attacks. Our approach to hierarchical rings applies to existing structured overlay protocols and can leverage existing work on, for instance, secure overlay routing. Moreover, our system can stitch together rings that run different overlay protocols and it respects connectivity constraints due to NATs and firewalls without additional engineering.

3.4 Status

The system as described is actively used within POST, a serverless infrastructure for collaborative applications including email, instant messaging, and shared whiteboards [16]. Users' desktops are collectively hosting the service, and organizational rings provide content and path locality.

An implementation of this system will be available as part of the upcoming FreePastry 1.4 release. The implementation is designed using only the KBR API [7], and can be used with any structured overlay protocol supporting this API. The release is open source and can be downloaded from `http://freepastry.rice.edu`.

4 Related Work

The use of multiple coexisting rings has been described before, most notably in the context of Coral [9] and SkipNet [13]. In Coral, multiple rings are used to provide data locality, and are built dynamically using the ping values to existing rings as a metric. The system does not provide guarantees over data placement and administrative autonomy.

Harvey et al. have first articulated the case for content and path locality [13]. SkipNet uses location-based id assignment in order to provide content and path locality. It employs a skiplist-based search structure to ensure robustness and load balancing despite the inherently uneven population of the identifier space. However, the system is more vulnerable to certain types of attacks that place makicous node near the boundaries of an organization's segment in the namespace [13]. Our multiring approach offers an alternative that can leverage existing protocols and work on secure routing at the expense of a slighly higher overhead for maintaining multiple rings.

Recently, virtual coexisting rings have been used to allow nodes to select which services they will opt to run [14]. The multiple rings are virtualized by providing alternate routing mechanisms, allowing a node to route to the nearest live node to a given key with the constraint that the node is a member of a certain group. Also, the use of multiple physical rings has been discussed in order to provide universal service discovery and code maintenance [4]. Such work is complementary to this paper, as the issue of path and content locality is not addressed in either of these approaches.

The Brocade [21] system, based on Tapestry, provides more efficient routing and path locality by using a secondary network of supernodes. Each administrative domain chooses a supernode, and inter-domain routing is accomplished via DHT lookups and landmark routing. This system is complementary to our work as it focuses on routing efficiency and provides neither content nor path locality.

Hierarchical peer-to-peer systems have also been explored in Garces-Erce et al. [10], but only with the goal of improving performance of the overlay network routing performance. A system of hierarchal rings was mentioned in the SkipNet paper as a design alternative. The authors opted for a different design due to the overhead associated with routing between multiple rings. We believe that in our system, this overhead is small enough to provide a practical alternative that can leverage existing work on structured overlays.

Additionally, none of the projects described above address the problem of deploying peer-to-peer overlays over networks with connectivity constraints. Many unstructured peer-to-peer overlays [11] solve this problem through network engineering, including push requests and rendezvous points, but these approaches add complexity and may not scale. Bryan Ford [8] has attempted to solve this problem in general with the use of a new network-layer protocol, Unmanaged Internet Protocol (UIP). However, the deployment of such technology or IPv6, is still, at best, years away.

5 Conclusions

Structured p2p overlay networks provide a decentralized, self-organizing substrate for large-scale distributed applications. However, most of the existing overlays either cannot ensure content and path locality, or they sacrifice some security. Also, the Internet has become increasingly fragmented - many hosts are not reachable due to firewalls and NATs. We presented a hierarchical configuration of structured overlays that reflects organizational boundaries and respect connectivity constraints. A multiring protocol stitches together organizational overlay that can run different overlay protocols that support the KBR API. To applications, the entire system appears like a single structured overlay that provides content and path locality. Since our solution works with any structured overlay protocol, it is able to leverage existing work, e.g., on secure overlay routing.

6 Acknowledgments

This work was supported in part by NSF (ANI-0225660), by Texas ATP (003604-0079-2001) and by Intel Research. We thank Miguel Castro and Antony Rowstron for discussions that lead to the ideas presented in this paper. We would also like to thank the anonymous IPTPS reviewers for their comments.

References

[1] M. Castro, M. Costa, and A. Rowstron. Performance and dependability of structured peer-to-peer overlays, 2003. Technical report MSR-TR-2003-94.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Security for structured peer-to-peer overlay networks. In *Proc. of the Fifth Symposium on Operating System Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[3] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Proximity neighbor selection in tree-based structured peer-to-peer overlays, 2003. Technical report MSR-TR-2003-52.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. One ring to rule them all: Service discovery and binding in structured peer-to-peer overlay networks. In *In Proceedings of the SIGOPS European Workshop*, Saint-Emilion, France, 2002.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE JSAC*, 20(8), Oct. 2002.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scalable application-level anycast for highly dynamic groups. In *Proc. NGC 2003*, Munich, Germany, 2003.

[7] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *Proc. IPTPS 2003*, Berkeley, California, 2003.

[8] B. Ford. Unmanaged internet protocol: Taming the edge network management crisis. In *In Proceedings of the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, 2003.

[9] M. Freedman and D. Mazieres. Sloppy hashing and self-organizing clusters. In *In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, 2003.

[10] L. Garces-Erce, E. Biersack, P. Felber, K. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. In *Proc. Euro-Par 2003*, Klagenfurt, Austria, 2003.

[11] The Gnutella protocol specification, 2000. `http://dss.clip2.com/GnutellaProtocol04.pdf`.

[12] R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *Proc. ACM SIGCOMM'03*, Karlsruhe, Germany, 2003.

[13] N. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proc. USITS 2003*, Seattle, WA, 2003.

[14] D. Karger and M. Ruhl. An augmented chord protocol supporting heterogeneous subgroup formation in peer-to-peer networks. In *In Proceedings of the 3nd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, San Diego, CA, 2004.

[15] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003.

[16] A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. Wallach, X. Bonnaire, P. Sens, J. Busca, and L. Arantes-Bezerra. Post: A secure, resilient, cooperative messaging system. In *Proceedings of the Ninth Workshop on Hot Topics in Operating Systems (HotOS '03)*, Lihue, HI, 2003.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM'01*, San Diego, CA, Aug. 2001.

[18] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *NGC*, Nov. 2001.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware 2001*, Heidelberg, Germany, Nov. 2001.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM'01*, San Diego, CA, Aug. 2001.

[21] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. Brocade: Landmark routing on overlay networks. In *In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, MA, 2002.

[22] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.