Appears in *Proceedings of the 1st Workshop on Secure Network Protocols* (NPSec 2005)

Identity Theft Protection in Structured Overlays *

Lakshmi Ganesh and Ben Y. Zhao
Computer Science Department, U. C. Santa Barbara
{lakshmi, ravenben}@cs.ucsb.edu

Abstract

Structured peer-to-peer (P2P) overlays rely on consistent and robust *key-based routing* to support large-scale network applications such as multicast and global-scale storage. We identify the main attack in these networks as a form of P2P identity theft, where a malicious node in the path of a message claims it is the desired destination node. Attackers can hijack route and lookup requests to forge and destroy data to disrupt applications. We propose a solution where nodes sign proof-of-life certificates for partial node ids and distribute them to randomly chosen proof managers in the network. Source nodes can evade attackers by requesting proofs from multiple proof managers. Analysis and simulation show the approach is effective and imposes storage and communication costs that grow logarithmically with network size.

1 Introduction

Structured peer-to-peer overlays [15, 8, 14] provide scalable and resilient infrastructures for Internet-scale applications. A variety of Internet-scale applications have been built on them, including application-level multicast [18, 10], distributed file systems [7, 9] and distributed query processing [5]. While many projects have studied these overlays, few have examined their security issues [12, 1]. Given their global scale, use of low cost identities, and distribution across independent network domains, we cannot treat the presence of malicious nodes as aberrations, but must expect them as part of normal operations.

The core functionality applications leverage is *key-based routing* (KBR) [2], where all messages with the same destination-key route to the same node consistently across changes in the network. Applications use this mechanism to store and locate data using location-independent names, much like a distributed hash table [2]. Since the overlays use large sparse namespaces (160 bits) to avoid name collision, nodes must deliver each message by choosing a single node closest to the destination key. This is often called the key's *root* node.

For any route request to key K, a malicious node on the routing path can hijack the key-based routing primitive by claiming that it is K's root node. Since nodes only keep state about $logN$ nodes for a N node network, they must rely on intermediate nodes to determine the key to root mapping. For example, an attacker with ID `12340` can hijack a message destined for node `12345` by claiming it is the only node with prefix `1234`. We call this the *Identity Attack*, since the attacker is stealing the identity of the true root node. To attack a file system, several malicious nodes close to a target node can claim they (or their colluding neighbors) are the root nodes for all outgoing read requests, and return arbitrary data in response.

In this paper, we describe the Identity attack, and present a solution where client nodes find self-certifying proofs to verify the existence of their desired destinations. Nodes periodically push signed proofs of their existence out to a random subset of network nodes. Client nodes use their routing table to estimate namespace density and determine when a root node is suspicious. They verify authenticity of root nodes by requesting existence proofs for closer IDs. Our detailed simulations show that namespace density estimation is effective at detecting suspicious nodes, and existence certificates provide proof of an attack while requiring reasonable traffic overhead.

We begin in Section 2 with a discussion of structured overlay security and related work. We then describe the identity attack and our defense in Section 3. Next in Section 4, we explore its efficiency and cost tradeoffs via detailed simulations, followed by conclusions in Section 5.

2 Background and Related Work

In this section, we describe structured overlays and key-based routing. We then discuss known attacks on these systems and other work related to this paper.

Key-based Routing A structured overlay is an application-level network connecting any number of nodes, each representing an instance of an overlay participant. The nodes are assigned nodeIds uniformly at random from a large identifier space. Application-specific

*The authors gratefully acknowledge support from the DARPA Control Plane project: BAA04-11.

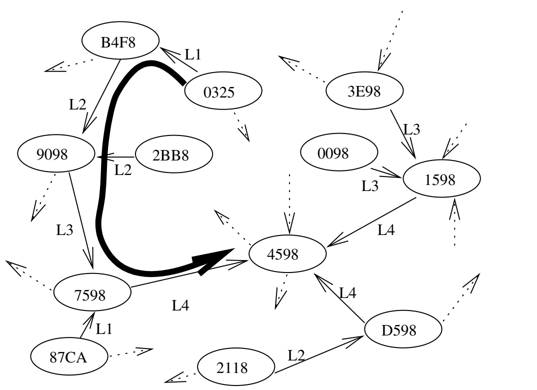

Figure 1: Base 10 prefix routing in Tapestry from `5230` to `8954`.

0	**0** 12021	**10** 0031	**120** 021	**1230** 02	**12300** 2	~~**123000**~~
1	**1** 23002	**11** 0321	**121** 301	**1231** 23	~~**12301**~~	~~**123001**~~
2	**2** 22031	**12** 3002	**122** 223	~~**1232**~~	~~**12302**~~	**123002**
3	**3** 30110	**13** 2100	**123** 002	**1233** 31	~~**12303**~~	~~**123003**~~

Node 123002

Figure 2: Base 4 routing table for node `123002`.

objects are assigned unique identifiers called keys from the same space.

Each key is dynamically mapped by the overlay to a unique live node, called its *root node*. While a key's root can change with network membership, a single node is responsible for a key in a consistent network at any given time. To deliver a message to its root node (key-based routing), each node forwards messages using a locally maintained routing table of overlay links. Figure 1 shows an example of Tapestry's prefix routing algorithm, and Figure 2 shows a node's routing table.

Each system defines a function that maps keys to nodes. For example, keys can be mapped to the live node with the closest nodeId as in Pastry [8], or the closest nodeId clockwise from the key as in Chord [14].

P2P Attacks and Defenses Previous work describes two attacks on structured overlays, the Sybil attack [3] and the Eclipse attack [1]. In the Sybil attack, an attacker generates a large number of identities and uses them together to disrupt normal operation. In the Eclipse attack, attackers try to organize to disproportionately populate routing tables inside target nodes to affect routing operation. Both attacks can increase the probability that a malicious node can intercept a desired route request, resulting in an Identity attack. The Identity attack is more general, however, since it can be launched by a single malicious node, and affects every route, lookup or store operation.

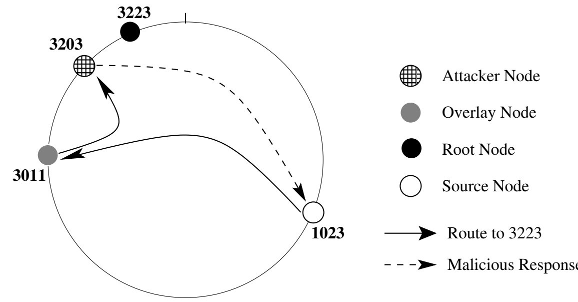

Figure 3: In a network using digits of base 4, node `1023` routes a message towards key `3223`. Before it reaches the root (`3223`), an attacker intercepts the message and responds as the root.

Several approaches limit the Eclipse attack by constraining connectivity in the network [1, 4, 11]. However, these defenses can only limit attackers from attracting more than their share of normal traffic, but cannot protect traffic that routes to malicious nodes. They also require external mechanisms to verify node properties such as in-degree and location in the network. In contrast, our approach requires only key-based routing, and can significantly reduce the impact of malicious nodes in the routing path.

Finally, public-private keypairs based on prefix IDs was also used in the Cashmere [17] anonymous routing system. Network indirection across an overlay is generalized for mobility and resilience in the Internet Indirection Infrastructure (I3) project [13].

3 Defending Against the Identity Attack

3.1 The Identity Attack

To perform an Identity Attack, a malicious node hijacks an overlay connection at setup time by spoofing the destination node. When an overlay node routes a message to some key K, it wants to connect to K's root node (generally the node in the overlay with ID closest to the key). A malicious node on the routing path intercepts the message and responds to the source claiming that it is K's root node.

By claiming to be K's root node, the attacker can intercept application requests and return data of its own choosing. For example, the attacker can hijack a request for a block in a distributed file system and respond with arbitrary data. At worst, she can arbitrarily manipulate application behavior; at best, the application invalidates the data, reducing this to an effective denial-of-service attack. Figure 3 shows an example of the identity attack.

While a single node can perform the attack, malicious parties can increase the effectiveness of the attack by using the Sybil attack [3] to generate a large number of col-

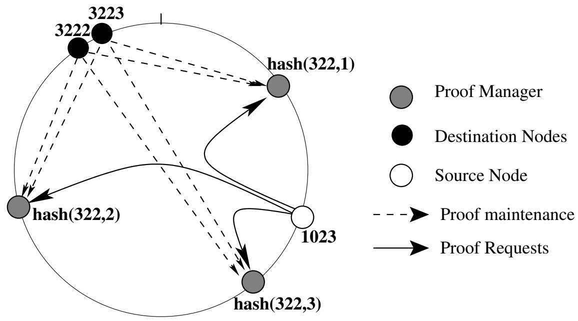

Figure 4: Nodes 3222 and 3223 periodically send signed existence proofs for prefix 322 to 3 randomly chosen proof managers in the overlay; node 1023 requests proofs.

luding attacker nodes. Attackers can generate numerous identities to perform a *client-based* or *key-based* identity attack. In a client-based attack, multiple malicious nodes collude to fill a target node's routing table using the Eclipse attack [11]. They can then isolate the node and hijack all outgoing application requests. In a key-based attack, the attacker targets a specific application-level key, and generates identities until it obtains a substantial number of identities close to the target key. Distributed across the network, these nodes will intercept most routing paths to the target, and effectively isolate the real root node (and content) from the network.

3.2 Existence Proofs

Our defense uses signed certificates to prove the existence of nodes with IDs in a namespace range. Online nodes periodically sign and send these *existence proofs* to a random subset of nodes, *proof managers*, for storage. When a node responds to a message with key K, the message source uses a namespace density estimate to determine whether the responder is a likely root. If not, the source node guesses the prefix that K's real root will share with K, and sends verification requests to the proof managers responsible for that prefix. If a better root exists, it will have signed a recent certificate with which the source can prove the identity theft. See Figure 4 for an example.

In a prefix routing protocol such as Tapestry [15], we use prefixes of different lengths to identify specific ranges of nodeIDs. In a namespace where nodes have IDs of L digits, we use a prefix of length $l < L$ to define a *prefix group* corresponding to the set of all nodes whose IDs start with that prefix. Shorter prefixes will have prefix groups with more members. If nodes use L-digit IDs, each node potentially belongs to L different prefix groups. This corresponds to different sized partitions of the namespace whose members are nodes with IDs in the range.

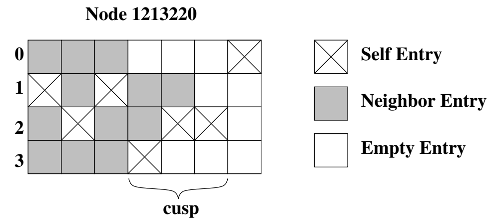

Figure 5: Routing table for node 1213220, showing the *cusp region*.

To certify that a prefix-group is non-empty, a member node uses a public key to sign an existence certificate embedded with a nonce. To simplify verification, a central offline CA distributes a unique public-private key pair to all members of each unique prefix group when they join the network. For example, a node ABCD would receive key pairs for prefix groups A, AB, ABC, and ABCD. Existence proofs are signed with the private key for the corresponding prefix. At regular intervals according to a network-wide parameter I, a node N signs certificates proving the existence of various prefix groups it belongs to. To determine the proof managers for prefix group P, N applies a SHA-1 hash to P with several salts (*i.e.*, 1,2,3) to generate several random keys in the namespace. The proof-managers are the root nodes of those keys.

A node initiates verification when it thinks a responder to a message for key K is suspicious. It examines its local routing table, to find the longest prefix column for which it has all entries filled. This threshold T is a measure of the density of the network. The responder should match the desired key with at least T prefix digits. If not, the node tries to verify the existence of nodes matching longer prefixes of key K. It searches for nodes matching a prefix by calculating its proof managers using the salted hash as described above, and queries them for the relevant certificates. If any queries are successful, it has discovered an attempted identity attack.

3.3 Limiting Prefix Groups

Certifying every possible prefix group in the network would be effective, but prohibitively expensive. In this section, we show how to validate the entire network by certifying only a small number of prefixes.

Our goal is to provide existence proofs for all prefixes of a certain length L. A small L means many nodes will matching the prefix, resulting in large bandwidth and storage overheads for verification. A large L means the client node may need to verify many prefixes in order to "find" a suitable root node.

Recall that nodeIDs are chosen uniformly at random using a secure hashing function (*e.g.* SHA-1). To choose an appropriate prefix length to store proofs for, the client node examines its own routing table to measure the density of nodes in the namespace. It chooses several prefix lengths for which its routing table contains columns with a mixture of empty and nonempty entries. We call this region the *cusp*. Figure 5 shows a node's routing table along with its cusp region. Choosing a prefix length L in this region means that each prefix of length L is likely to match a "small" number of nodes in the system.

Prior work [16] proved that with a high probability, such a cusp will have a size ≤ 2, independent of network size. The proof is an application of the Coupon Collector problem, where entries in a row of the routing table are coupons, and collecting coupons is the act of assigning random IDs to fill a particular entry. The result says the probability of the cusp including more than 2 routing levels is $P \leq b/e^b$, where b is the base of the prefix digit. P is less than 0.07 for $b = 4$ and less than $1.8 * 10^{-6}$ for $b = 16$.

As a result, we assume a cusp size of 3. A client node searches its routing table, and marks the start of the cusp as the first routing level that contains an empty entry. For example, node `1213220` in Figure 5 uses prefixes of length 4, 5, and 6 when sending certificates and requesting verifications. Verifications start from the longest possible prefix and work downwards. To test for the existence of a node `12301230`, our node would first test prefix `123012`, then `12301` if necessary, and finally `1230`.

3.4 Replicating Proof Managers

Several factors can affect the success rate of verification requests. Node churn can limit the availability of proof managers for a given prefix. Malicious nodes on the path between the client and proof managers can hijack and drop the verification request. Finally, if proof managers themselves are compromised, they can simply deny ever seeing the requested existence certificate.

We can improve the verification success rate by increasing the number of randomly chosen proof managers. A larger replication factor means more managers will be online despite network churn and more of them will be nonmalicious. Verification requests will take a larger number of random routes, increasing the number of requests that will avoid malicious nodes. We evaluate the impact of these factors on verification in Section 4.

3.5 Extension to Other Protocols

While much of our discussion assumes the use of prefix routing, our technique easily generalizes to other protocols. For example, a range-based routing protocol like Chord simply routes "towards" a given value. Instead of certifying existence of nodes matching a given prefix `0123`, we would certify existence of nodes in a certain value range (*e.g.* `12300-12399`). Similarly, our mechanism for choosing prefix lengths to certify reduces to finding several range sizes that have the right node density. We are studying these mechanisms in ongoing work.

Topology	Random
Length of run	7200s
Base	16
Prefixes certified (cusp size)	3
Certification interval	500s
Certificate time-out period	1500s

Table 1: Simulation Settings

4 System Evaluation

In this section we describe some preliminary results based on detailed simulations on the P2PSim simulation platform. P2PSim [6] is a multithreaded discrete event simulator with full implementations of several protocols. Our experiments are run using the implementation of Tapestry [15] included with P2PSim. The simulation settings are listed in Table 1.

4.1 Overhead of Existence Proofs

In order to defend against Identity Attacks, each node in the network incurs bandwidth overhead in sending existence proofs (certificates) to proof managers, and storage overhead in storing existence proofs for other prefixes. We use simple analysis to quantify these overheads.

Bandwidth Cost. Nodes certify their prefix groups every T seconds ($T = 500s$ in our simulations). Let v denote the number of prefix groups each node certifies ($v = 3$ in our simulations), and r denote the replication factor (number of proof managers per prefix). Thus the rate that each node sends out certificates is $\frac{v \cdot r}{T}$. With our simulation parameters, we expect each node to send $\frac{3 \cdot 4}{500} = 0.024$ certificates/second. This is confirmed by our measurements that show the rate to be 0.025 certificates per second for all network sizes. As expected, this overhead increases linearly with replication factor. Certificates should be no larger than 50 Bytes, resulting in bandwidth cost of 1.25 Bytes/second or 10 bps.

Storage Cost. If we assume that a new certificate replaces previous certificates from the same host, then each node generates vr certificates to be stored. Assuming the hash function to generate proof manager IDs spreads the load evenly across all nodes, each node only needs to store vr certificates, for a total cost of 600 Bytes of storage per node using our parameters. Clearly, the overhead from

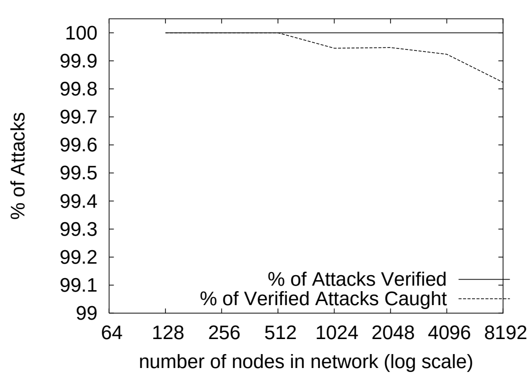

Figure 6: Effectiveness of the verification system under ideal conditions (no denials, no certificate hijacks, no node churn).

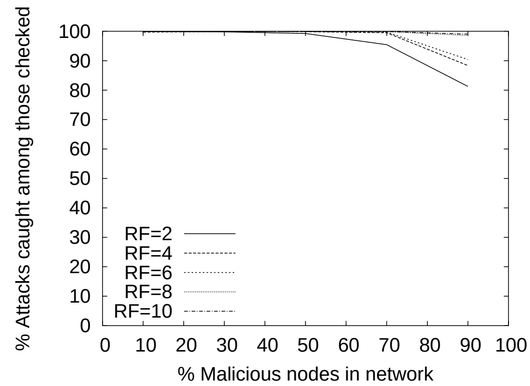

Figure 7: Use of replication factor to increase verification effectiveness (certificate denials, no hijacks, no node churn).

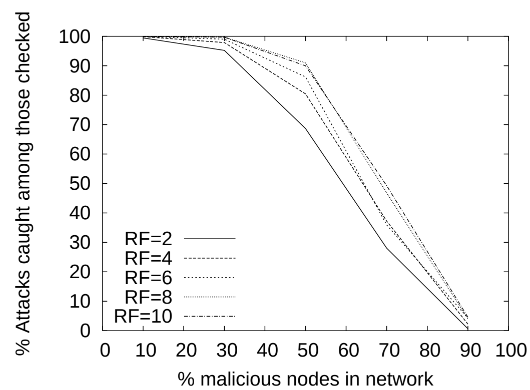

Figure 8: Use of replication factor to increase verification effectiveness (certificate denials and hijacks, no node churn).

generating and storing certificates is small enough to not impact overall system performance.

4.2 Resilience Against the Identity Attack

We evaluate our defense against the Identity Attack under a variety of conditions. We first consider the basic identity attack in a stable network where all mechanisms function normally. Next, we consider the case when compromised proof managers deny the presence of certificates. This models colluding malicious attackers, where colluding attackers cover each others' tracks by refusing to service verifications. Another factor is certificate hijacks, where malicious nodes intercept certificates on the path between the signer and a proof manager. This is a stronger attack, where we assume in-path routers has read the message payload to determine if it is a certificate (*i.e.* no overlay link level encryption). This attack also applies if malicious nodes indiscriminately hijack all messages they see without interpretation. Finally, we examine the impact of nodes entering and leaving the network (node churn).

Performance Metrics To quantify the effectiveness of our defense, we examine two metrics, the *trigger rate*, how often does an attack trigger a verification request, and the *verification rate*, how often do requests succeed in locating an existence proof proving the attack. Our simulations show that we get a trigger rate of 100% using our threshold detection scheme, with roughly 3 verifications requested per actual attack. We are currently tuning our threshold to reduce the verification overhead. Since the trigger rate is 100%, our experiments measure the verification rate under different conditions.

Ideal Conditions We assume that malicious nodes hijack non-certificate messages, but do not hijack certificates, and compromised proof managers perform normally. We vary the network size and assume no node churn. Figure 6 shows that for all network sizes, our trigger rate is 100%, and verification rate is over 99.8%.

Verification Denials Here we assume malicious nodes hijack messages and deny verification requests, but do not hijack certificates. We simulate a network of 4K nodes while varying the percentage of malicious nodes. Figure 7 shows that the verification rate falls as the proportion of malicious nodes increases. For a given proportion of malicious nodes, however, performance improves if we increase the number of proof managers. This improvement is significant when a large number of nodes is malicious. Note that even when 90% of nodes are malicious, over 80% of attacks are caught with just two proof managers.

Certificate Hijacks This model is similar to the last model, with the addition that malicious nodes also hijack certificates en route to proof managers. Our results in Fig-

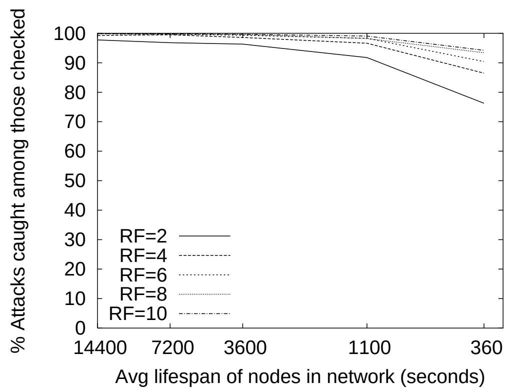

Figure 9: Use of replication factor to increase verification effectiveness (certificate denials, hijacks, and node churn assuming 20% malicious nodes).

ure 8 show that this additional factor causes a steep fall in performance. For a given % of malicious nodes in the network, certificate hijacks have a much stronger impact than verification denials. This is because the likelihood that a malicious node is in the path of a certification is higher than the likelihood that it is a proof manager. For predominantly malicious (over 50%) networks, the percentage of attacks caught is very poor, even when many proof managers are used. But for networks with up to 40% malicious nodes, nearly 80% of the attacks are caught.

Churn Finally, we examine the effect of adding churn to the network. We simulate an attack model that includes Identity Attacks, verification denials as well as certificate hijacks, and ran it on networks of 4096 nodes, 20% of which are malicious. Figure 9 shows that increasing churn degrades performance. With 8 or more proof managers, it is highly likely that at least one non-malicious proof manager can service each verification request; hence over 95% of the attempted hijacks are caught.

5 Conclusion

In this paper, we described the Identity Attack, a simple attack that subverts the fundamental key-based routing functionality in structured peer-to-peer overlays. Unlike the Sybil and Eclipse attacks, the Identity attack directly impacts application level behavior, and can leverage both prior attacks for increased effectiveness. We propose a defense that uses the placement of signed existence proofs at randomized node subsets. After routing requests, source nodes use estimates of namespace density to trigger verification, where they determine whether "better" root nodes exist by searching for their existence proofs.

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Proc. of OSDI*, Dec 2002.

[2] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured P2P overlays. In *Proc. of IPTPS*, Feb 2003.

[3] J. R. Douceur. The Sybil attack. In *Proc. of IPTPS*, Mar 2002.

[4] K. Hildrum and J. Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. of DISC*, Oct 2003.

[5] R. Huebsch et al. Querying the internet with pier. In *Proc. of VLDB*, Berlin, Germany, Sept. 2003.

[6] p2psim, a simulator for peer-to-peer protocols. `http://pdos.csail.mit.edu/p2psim/`.

[7] S. Rhea et al. Pond: The OceanStore prototype. In *Proc. of FAST*, San Francisco, Apr 2003.

[8] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, Nov 2001.

[9] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. of SOSP*. ACM, Oct 2001.

[10] A. Rowstron, A.-M. Kermarrec, P. Druschel, and M. Castro. SCRIBE: The design of a large-scale event notification infrastructure. In *Proc. of NGC*. ACM, Nov 2001.

[11] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *Proc. of ACM SIGOPS European Workshop*. ACM, Sept. 2004.

[12] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. of IPTPS*, Mar 2002.

[13] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. of SIGCOMM*, Aug 2002.

[14] 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. of SIGCOMM*, Aug 2001.

[15] B. Y. Zhao et al. Tapestry: A global-scale overlay for rapid service deployment. *IEEE JSAC*, 22(1), Jan 2004.

[16] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report CSD-01-1141, UCB, Apr 2001.

[17] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proc. of NSDI*, Boston, MA, May 2005. ACM/USENIX.

[18] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of NOSSDAV*. ACM, June 2001.