Impact of Neighbor Selection on Performance and Resilience of Structured P2P Networks

Byung-Gon Chun[1], Ben Y. Zhao[2], and John D. Kubiatowicz[1]

[1] Computer Science Division, U. C. Berkeley
{bgchun, kubitron}@cs.berkeley.edu
[2] Department of Computer Science, U. C. Santa Barbara
ravenben@cs.ucsb.edu

Abstract. Recent work has shown that intelligent neighbor selection during construction can significantly enhance the performance of peer-to-peer overlay networks. While its impact on performance has been recognized, few have examined the impact of neighbor selection on network resilience. In this paper, we study the impact with a generalized cost model for overlay construction that takes into consideration different types of heterogeneity, such as node capacity and network proximity. Our simulation results show that the resulting performance improvement comes at the cost of static resilience against targeted attacks and adding random redundancy can improve the resilience significantly.

1 Introduction

Recent research has shown structured peer-to-peer overlay networks to provide scalable and resilient abstractions to large-scale applications [9, 11, 12, 15, 20]. They support routing to endpoints or nodes inside a network requiring only logarithmic routing state at each node. Nodes in structured peer-to-peer networks choose their neighbors based on optimization metrics. A recent study by Gummadi et al. [7] shows that neighbor selection based on network proximity significantly improves overall performance.

However, such neighbor selection can lead to an unbalanced overlay structure. Figure 1 shows a snapshot of the number of incoming edges (in-degree) and outgoing edges (out-degree) of nodes in a Bamboo [11] overlay running on PlanetLab [5]. Because the overlay uses proximity neighbor selection, some nodes in the system are more popular (have higher in-degree) than others. The impact of such a skewed degree distribution on the static resilience of networks has yet to be quantified. The focus of our study is to look at the impact of different neighbor selections on static resilience and performance of networks.

To better model neighbor selection across these networks, we first present a generalized cost model. While the heterogeneity of Internet hosts in bandwidth, inter-node latency and availability are well measured [13], most current protocols only consider network proximity in neighbor selection. Thus we use different neighbor selection models based on network proximity and node capacity. We study the impact they have on lookup latency and static resilience by incorporating the neighbor selection algorithms into ring and tree geometries, and show that the performance improvement from exploiting network proximity or node capacity comes at a price of increased vulnerability

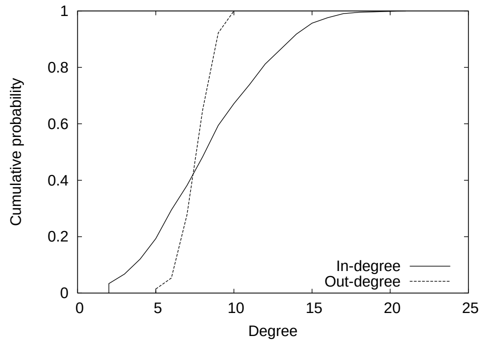


Fig. 1. Cumulative distribution of node degrees of a 205-node Bamboo overlay running on PlanetLab. In-degree and out-degree represent the number of incoming edges and the number of outgoing edges of each overlay node. The graph does not include default links (*i.e.*, leafset) used for failure tolerance. This is a snapshot taken on August 26, 2004.

against targeted attacks. Finally, we show that adding random redundancy can significantly improve static resilience against targeted attacks.

The paper is organized as follows. We discuss related work in Section 2 and describe details of the neighbor selection model in Section 3. We then measure the impact of different cost functions on both resilience and performance in Section 4 and conclude in Section 5.

2 Related Work

The closest work to ours was done by Gummadi et al. [7]. The authors quantified the impact of *routing geometry* on performance and static resilience. In contrast, we focus on the impact of *neighbor selection* on these factors. Albert et al. [1] show a clear correlation between the scale-free nature of networks and resilience to attacks and failures. Chun et al. [6] show the tradeoff between performance and network resilience of selfishly constructed unstructured overlays.

Castro et al. studied a defense mechanism against Eclipse attacks where attackers fake proximity to increase the fraction of bad routing entries in structured peer-to-peer networks using proximity neighbor selection [2]. They proposed to use two routing tables — proximity-based one and constrained one. Singh et al. proposed to bound the degree of overlay nodes in one proximity-based routing table to defend against Eclipse attacks [14]. In our work, attackers affect network connectivity by taking down nodes with high degree.

Several research efforts propose optimizing overlay construction of structured overlays using the network proximity metric [3, 10, 16, 18, 20], but generally ignore other factors such as CPU load, storage and bandwidth capacity. Brocade [19] proposes the use of supernodes for more efficient routing, but requires static selection of supernodes.

Other work [8] proposes the use of multiple "virtual servers" for load balancing among nodes of varying resource capacity, but does not consider network proximity for routing performance. Gia [4] performs continuous topology adaptation on an unstructured overlay such that nodes participate with network degree matching their resource capacity, without considering network proximity.

3 Structured Overlay Construction

In the construction of structured peer-to-peer networks, each node chooses neighbors that meet logical identifier constraints (e.g., prefix matching or identifier range), and builds directional links to them. These constraints are flexible such that a number of nodes are possible neighbors for each routing table entry. Intelligent selection of neighbors from the set of possible neighbor nodes significantly impacts the overlay's performance, resilience, and load balancing properties.

The neighbor selection problem can be reduced to a generalized cost minimization problem. We present here a generalized cost model that captures general node and link characteristics during neighbor selection. Ideally, optimizing neighbor selection for node i means minimizing the sum of the cost from i to all other nodes. The cost from i to j consists of two factors: cost incurred by intermediate overlay nodes (node cost: c_n) and cost incurred by overlay network links (edge cost: c_e). Let N be the network size. The cost of node i (C_i) is:

$$\begin{aligned} &C_i = \textstyle\sum_{j=1}^{N} t(i,j)c_p(i,j) \ \text{ where} \\ &c_p(i,j) = \textstyle\sum_{n \in V(i,j)} c_n(n) + \sum_{e \in P(i,j)} c_e(e) \end{aligned} \tag{1}$$

where $t(i,j)$ is the traffic from i to j, $c_p(i,j)$ is the cost of the path from i to j, $P(i,j)$ is the path (a set of edges) from i to j, $V(i,j)$ is the set of intermediate overlay nodes in the path $P(i,j)$ (it does not include i and j), e is an edge in the path $P(i,j)$, n is a node in $V(i,j)$, $c_n(n)$ is the cost of node n, and $c_e(e)$ is the cost of edge e. If $t(i,j)$=0, there is no incentive for the node to optimize the path from i to j. In this model, c_n captures the heterogeneity in node capacity, which is a function of bandwidth, computation power, disk access time, and so on. c_e captures network proximity.

For structured networks such as Chord, Pastry, and Tapestry, the cost function can be rearranged as follows:

$$\begin{aligned} &C_i = \textstyle\sum_{b=1}^{N_b} \sum_{j \in R_b} t(i,j)c_p(i,j,n_b) \ \text{ where} \\ &c_p(i,j,n_b) = [c_n(n_b) + c_e(i,n_b)] + [\textstyle\sum_{n \in V(n_b,j)} c_n(n) + \sum_{e \in P(n_b,j)} c_e(e)] \\ &\qquad\qquad\ \ = [c_n(n_b) + c_e(i,n_b)] + c_p(n_b,j) \end{aligned} \tag{2}$$

where b is the neighbor index, n_b is the neighbor indexed by b, N_b is the number of neighbors, R_b is the set of destinations routed through the neighbor n_b, $c_n(i)$ is the node cost value of i, $c_e(k,l)$ is the edge cost between two nodes k and l, and $c_e(e)$ is the edge cost of e. $c_p(i,j,n_b)$ is the cost of the path from i to j with n_b as a first hop; we see that this includes terms from the first hop $[c_n(n_b) + c_e(i,n_b)]$ and terms from the remainder of the path $c_p(n_b,j)$.

Depending on the optimization goal, we can choose different metrics for c_n and c_e, including latency, throughput, reliability, availability, monetary cost, or any combination thereof. For example, choosing high capacity nodes as neighbors can decrease lookup latency and increase the overall lookup processing capacity of the system. On

Model	Cost (C_i)
Random	None
Dist	$\sum_{b=1}^{N_b} c_e(i, n_b)$
Cap	$\sum_{b=1}^{N_b} c_n(n_b)$
CapDist	$\sum_{b=1}^{N_b} \{ c_n(n_b) + c_e(i, n_b)\}$

Table 1. Cost functions studied. $c_n(i)$ represents the processing delay in node i. This is a decreasing function of capacity of node i. $c_e(i, n_b)$ represents the direct overlay link delay between node i and node n_b.

the other hand, using availability as a metric creates a more stable network or using monetary cost can create a network that is more economically incentivized.

Note that our idealized cost function assumes full knowledge of the network components, and is therefore not feasible in practice. Since most peer-to-peer protocols focus on optimizing neighbor tables locally, we will focus on the application of our cost function to the cost of the first overlay hop. In this work we focus on neighbor selections that consider the first hop and optimize latency under uniform traffic ($t(i, j) = 1, \forall i, j$).

Table 1 shows the four neighbor selection cost functions. *Random* chooses neighbors randomly. *Dist* chooses neighbors physically closest in the network to adapt to the underlying network topology. Currently, Bamboo, Pastry, and Tapestry use this mechanism. *Cap* chooses neighbors that have the smallest processing delay. *CapDist* chooses neighbors that gives the smallest combined latency, which is the sum of the node processing delay and the overlay link delay.

4 Simulation Results

In this section, we first present simulation results that quantify the performance benefits of using intelligent neighbor selection algorithms. We then examine the impact such algorithms have on the static resilience of the resulting overlay to randomized failures and targeted attacks.

4.1 Simulation Setup

We simulate the Tapestry [20] and Chord [15] protocols as representatives of their respective geometries (tree and ring). When each node optimizes its cost function, it performs random sampling to select neighbors and choose the best one among the samples. In our experiments, we use 32 samples for each routing level in Tapestry or each finger in Chord.

We use practical greedy routing algorithms for both Tapestry and Chord. For Tapestry, each node forwards messages to the first live neighbor matching one more prefix digit. The lookup fails if all primary and backup links in the routing entry fail. For our Chord experiments, each node forwards messages to the live neighbor that is closest to the destination in the identifier space. The lookup fails if all neighbors before the destination in the namespace fail. Note that the measured network resilience depends on the routing algorithms we use.

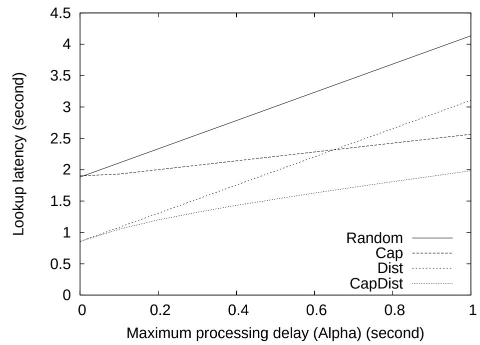


Fig. 2. Average lookup latency for uniform processing delay distribution. When processing delay variation is low, neighbor selections that exploit network proximity (*Dist* and *CapDist*) have low latency. However, when processing delay variation is high, neighbor selections that exploit node capacity (*Cap* and *CapDist*) have low latency.

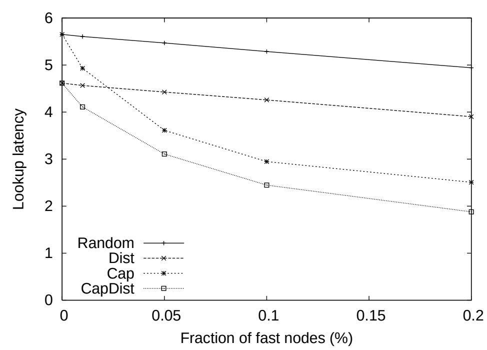


Fig. 3. Average lookup latency for bimodal processing delay distribution. As the fraction of fast nodes increases, neighbor selections using node capacity can have better lookup latency than those that do not use node capacity.

Our simulations use 5100 node transit-stub network topologies generated using the GT-ITM library [17]. We construct Chord and Tapestry overlays of 4096 nodes by placing overlay nodes to random physical locations. We gather results with 9 different configurations for GT-ITM, generate 3 transit-stub topologies each, and choose 3 overlay node placements on each topology.

4.2 Performance

We begin by quantifying the effects of neighbor selection algorithms on performance. We look at two different distributions of node processing delay: uniform and bimodal. Because Tapestry and Chord results are similar in both cases, we will only show Tapestry results.

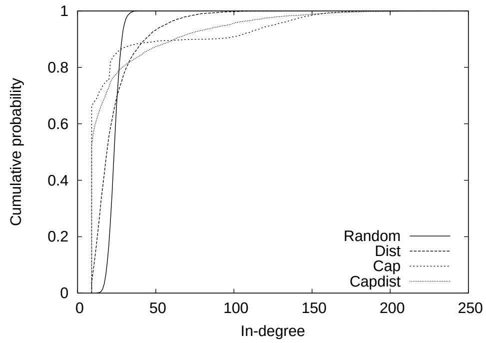


Fig. 4. CDF of the number of incoming edges for uniform processing delay distribution. *Random* shows an even in-degree distribution, but the others have very skewed distributions.

We start by assigning node processing delay from a coarse-grained uniform distribution. We choose one of 10 values uniformly from the range $(0, \alpha]$, where α is the maximum processing delay. Figure 2 shows average lookup latency over all node pairs in Tapestry. By exploiting network proximity and heterogeneous capacity, *CapDist* achieves the best lookup performance. When processing delay variation is high (α=1s), *CapDist* performs 30% better than *Dist* and 48% better than *Random*. When no variation exists (*i.e.*, α=0s), *Dist* and *CapDist* exploit network proximity to outperform *Random* and *Cap*.

We now look at a bimodal model for processing capacity, where nodes are either fast or slow. Fast nodes process 100 lookup messages per second while slow nodes process 1 message per second. Figure 3 shows that as we vary the fraction of fast nodes from 0% to 20%, neighbor selection using capacity (*Cap* and *CapDist*) favors routes through fast nodes and achieves better performance. For instances where the variation in processing capacity is extremely high, we expect that capacity utilization at fast nodes will be limited by the routing constraints of the protocol, and the deployment of virtual nodes is necessary to fully exploit the excess processing capacity.

Using latency optimization creates uneven distributions of nodes' incoming node degrees. Nodes near the center of the network (*i.e.*, transit domains) and nodes with high capacity are preferred, and minimize path latency by utilizing low latency links or low processing delay. Figure 4 shows the cumulative distribution function (CDF) of nodes' in-degrees in Tapestry networks with different neighbor selection algorithms. Unlike *Random*, results from cost-optimized overlays show slow transitions and long tails. We also observe that the CDF of nodes in transit domains is more skewed and has longer tails than that of nodes in stub domains.

4.3 Static Resilience

Previous work by Albert et al. showed an inherent tradeoff for unstructured networks between resilience against random node failures and resilience against targeted at-

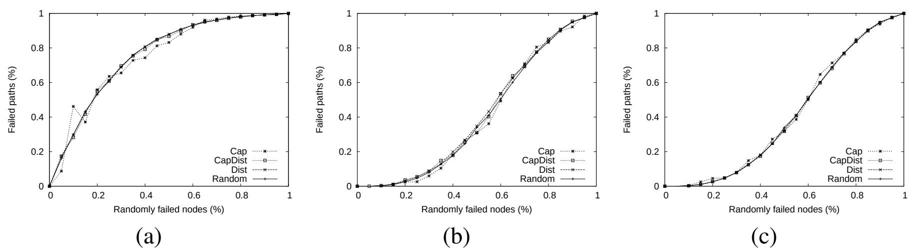


Fig. 5. Tapestry under random node failures. (a) Tapestry varying neighbor selection on one primary link (e.g., $Dist$: primary link chosen to optimize the $Dist$ cost function), (b) Tapestry varying neighbor selection on one primary link and two backup links (e.g., $Dist$: all three links chosen to optimize the $Dist$ cost function), (c) Tapestry varying neighbor selection on one primary link and choosing two backup links randomly (e.g., $Dist$: primary link chosen to optimize the $Dist$ cost function and two backup links chosen randomly).

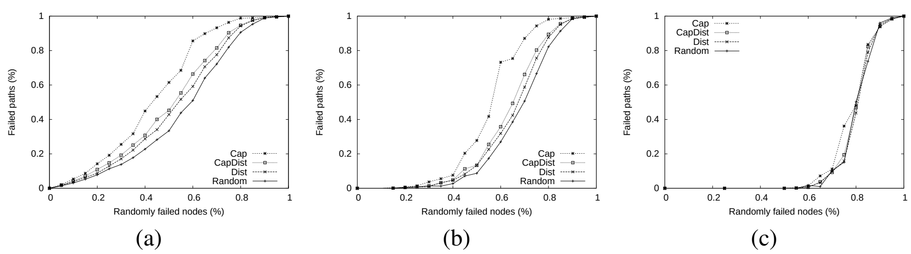
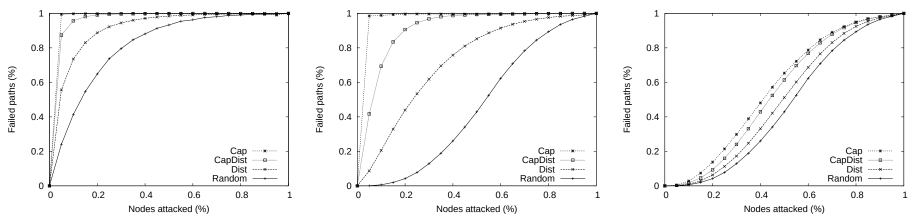


Fig. 6. Chord under random node failures. Chord varying finger selection on finger table having (a) one sequential neighbor, (b) four sequential neighbors, and (c) 12 sequential neighbors.

tacks [1]. In this section, we explore the impact that neighbor selection algorithms have on static resilience.

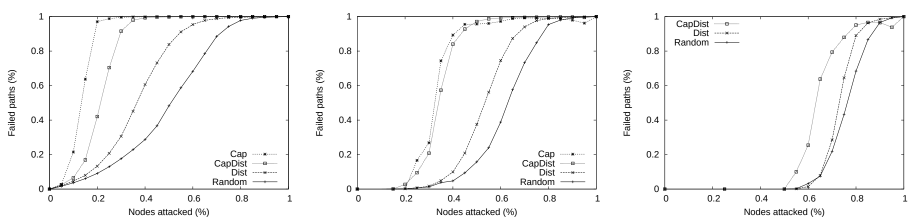
We measure resilience as the proportion of all pairs of *live* endpoints that can still route to each other via the overlay after an external event, either randomized node failures or targeted attacks. We assume attacks focus on removing nodes with the highest in-degree in order to maximize damage to overall network reachability. For these experiments, we assume nodes have a uniform processing delay distribution with α = 0.5s.

For Tapestry, we examine resilience of the base protocol, the base protocol plus additional backup links (all chosen using a number of neighbor selection algorithms), and the base protocol plus backup links chosen at random. We maintain backup links for each routing level, so adding two backup links triples the number of neighbors. For Chord, we examine the base protocol (i.e., protocol with one sequential neighbor) and the base protocol plus multiple sequential neighbors. Sequential neighbors are successors in the identifier space. They can make progress to route to all destinations.



(a) (b) (c)

Fig. 7. Tapestry under attack. (a) Tapestry varying neighbor selection on one primary link (e.g., $Dist$: primary link chosen to optimize the $Dist$ cost function), (b) Tapestry varying neighbor selection on one primary link and two backup links (e.g., $Dist$: all three links chosen to optimize the $Dist$ cost function), (c) Tapestry varying neighbor selection on one primary link and choosing two backup links randomly (e.g., $Dist$: primary link chosen to optimize the $Dist$ cost function and two backup links chosen randomly).



(a) (b) (c)

Fig. 8. Chord under attack. Chord varying finger selection on finger table having (a) one sequential neighbor, (b) four sequential neighbors, and (c) 12 sequential neighbors. For (c), we do not present *Cap*. We cannot find an order to launch targeted attacks, since *Cap* creates networks where many nodes have the same node degree.

Random Node Failures We first examine the impact of randomized node failures. In general, we would expect that using selection algorithms that prefer high capacity nodes results in more hierarchy in the network, where many weaker nodes are connected by highly interconnected high capacity nodes. In such cases, we expect that randomized failures will disconnect weaker nodes from the network, but have a relatively low impact on overall connectivity.

Figures 5 and 6 show the failure tolerance of Tapestry and Chord, respectively. Surprisingly, we see the failure tolerance is a little affected by neighbor selections. The tighter outgoing link constraints of structured peer to peer networks allow less variation in the resulting topology than unstructured networks. Every node has at least $O(logN)$ outgoing links, and randomized naming also smoothens out distribution of outgoing links. Since each lookup takes $O(logN)$ hops regardless of neighbor selection cost functions, the probability of meeting randomly failed nodes in a lookup will be similar.

Adding backup links in Tapestry and sequential neighbors in Chord dramatically improves failure tolerance (Figures 5(b), 5(c), 6(b), and 6(c)). Note that in Tapestry, failure behavior changes from extremely brittle (concave downward with increasing node failure) to smoothly varying (an S-shaped curve with increasing node failure) with the addition of path diversity.

Targeted Node Attacks While structured peer to peer overlays define a minimum number of outgoing links per node, a node's number of incoming links is unrestricted. This means that neighbor selection algorithms considering capacity or network proximity will skew the network such that powerful or central nodes have significantly higher in-degrees than weaker or boundary nodes. This means that structured peer to peer overlays that consider capacity or network proximity in neighbor selection can be vulnerable to attacks.

As shown in Figures 7(a) and 8(a), attacking nodes with high in-degree affects network connectivity severely. *Random* shows the best attack tolerance among neighbor selections. *CapDist* has worse attack tolerance than *Dist*, although it has the best performance among neighbor selections we examine. In Tapestry, when 30% of nodes are attacked, 0.4% of pairs of live nodes can communicate in the networks created with *CapDist*, but 20.4% of pairs of live nodes can still communicate in the networks created with *Random*. In Chord, when 50% of nodes are attacked, 0.2% of pairs of live nodes can communicate in the networks created with *CapDist*, but 51.8% of pairs of live nodes can still communicate in the networks created with *Random*.

This result demonstrates a fundamental tradeoff between performance and attack resilience in structured overlay construction. The performance gain from neighbor selection algorithms increases the variability of in-degrees among nodes. Nodes with high capacity or nodes near the center of the network end up with high in-degrees and have a disproportionately large impact on network connectivity when they are attacked.

Adding Redundancy From Figures 5(a) and 7(a) we observe that the resilience of *Random* under random failures is the same as that under targeted attacks. This result shows that randomness can shield against attacks targeting biases. If we can bring the randomness back into the system, we may improve the resilience against targeted attacks.

Paying the additional cost of maintaining extra links improves static resilience against targeted attacks. Figures 7(c) and 8(c) show that adding backup links or sequential neighbors can increase attack tolerance significantly. When 30% of nodes are attacked in Tapestry with one primary link optimizing *CapDist* and two random backup links, 76% of pairs of live nodes can communicate. When 50% of nodes are attacked in Chord with 12 sequential neighbors, all nodes can still communicate. Randomly choosing backup links in Tapestry and sequential neighbors in Chord avoids routing hotspots that are vulnerable to targeted attacks. In Tapestry for example, cost-optimized backup links are less effective at improving attack tolerance than random backup links (Figure 7(b)). Using sequential neighbors gains good attack resilience with the overhead of high lookup latency under attacks.

5 Conclusion

Previous research argued for the consideration of network or physical characteristics of nodes in overlay construction. In this paper, we take a quantitative approach to examining the benefits and costs of considering such criteria in overlay construction.

We present a generalized model for neighbor selection that incorporates metrics for network proximity and available resources (capacity), and show that while considering these factors can lead to significant gains in routing performance, these benefits come with their associated costs. We find that the choice of neighbor selection algorithm drives a tradeoff between performance and resilience to attacks.

Optimized structured overlays have unbalanced structures. These overlays do not bound the number of incoming links per node. Thus central nodes in a network or nodes with more resources will have much higher in-degree than others. Should high degree nodes be attacked, the impact on network connectivity is severe. On the other hand, the minimum out-degree means even for overlays that optimize towards proximity or available resources, most nodes achieve enough resilience against randomized failures. Finally, we show that adding random redundancy can improve the resilience significantly.

As future work, we intent to investigate the resilience of different geometries under different neighbor selection algorithms. We also plan to investigate the impact of these neighbor selection algorithms on dynamic resilience, such as when maintenance algorithms repair failures over time.

References

1. ALBERT, R., JEONG, H., AND BARABASI, A.-L. Error and attack tolerance of complex networks. *Nature 406* (July 2000), 378–381.
2. CASTRO, M., DRUSCHEL, P., GANESH, A., AND ROWSTRON, A. Secure routing for structured peer-to-peer overlay networks. In *Proc. of USENIX OSDI* (December 2002).
3. CASTRO, M., DRUSCHEL, P., HU, Y. C., AND ROWSTRON, A. Exploiting network proximity in peer-to-peer overlay networks, technical report msr-tr-2002-82, 2002.
4. CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., LANHAM, N., AND SHENKER, S. Making gnutella-like p2p systems scalable. In *Proc. of ACM SIGCOMM* (2003).
5. CHUN, B., CULLER, D., ROSCOE, T., BAVIER, A., PETERSON, L., WAWRZONIAK, M., AND BOWMAN, M. Planetlab: An overlay testbed for broad-coverage services. In *ACM Computer Communication Review* (July 2003).
6. CHUN, B.-G., FONSECA, R., STOICA, I., AND KUBIATOWICZ, J. Characterizing selfishly constructed overlay routing networks. In *Proceedings of IEEE INFOCOM* (2004).
7. GUMMADI, K. P., GUMMADI, R., GRIBBLE, S. D., RATNASAMY, S., SHENKER, S., AND STOICA, I. The impact of dht routing geometry on resilience and proximity. In *Proc. of ACM SIGCOMM* (2003).
8. RAO, A., LAKSHMINARAYANAN, K., SURANA, S., KARP, R., AND STOICA, I. Load balancing in structured p2p systems. In *Proc. of IPTPS* (2003).
9. RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A scalable content-addressable network. In *Proc. of SIGCOMM* (August 2001), ACM.
10. RATNASAMY, S., HANDLEY, M., KARP, R., AND SHENKER, S. Topologically-aware overlay construction and server selection. In *Proc. of IEEE INFOCOM* (2002).

11. Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. Handling churn in a dht. In *Proc. of the USENIX Annual Technical Conference* (June 2004).
12. Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware* (Nov 2001), ACM.
13. Saroiu, S., Gummadi, P. K., and Gribble, S. D. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (2002).
14. Singh, A., Castro, M., Druschel, P., and Rowstron, A. Defending against eclipse attacks on overlay networks. In *Proc. of the ACM SIGOPS European Workshop* (September 2004).
15. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM* (August 2001), ACM.
16. Waldvogel, M., and Rinaldi, R. Efficient topology-aware overlay network. In *Proc. of HotNets* (2002).
17. Zegura, E. W., Calvert, K. L., and Bhattacharjee, S. How to model an internetwork. In *Proc. of IEEE INFOCOM* (1996).
18. Zhang, H., Goel, A., and Govindan, R. Incrementally improving lookup latency in distributed hash table systems. In *Proc. of ACM SIGMETRICS* (2003).
19. Zhao, B. Y., Duan, Y., Huang, L., Joseph, A., and Kubiatowicz, J. Brocade: Landmark routing on overlay networks. In *Proc. of IPTPS* (2002).
20. Zhao, B. Y., Huang, L., Rhea, S. C., Stribling, J., Joseph, A. D., and Kubiatowicz, J. D. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC 22*, 1 (January 2004), 41–53.