Know thy Neighbor's Neighbor: Better Routing for Skip-Graphs and Small Worlds.*

Moni Naor[†‡] Udi Wieder [‡]

Abstract

We investigate an approach for routing in p2p networks called *neighbor-of-neighbor greedy*. We show that this approach may reduce significantly the number of hops used, when routing in skip graphs and small worlds. Furthermore we show that a simple variation of Chord is degree optimal. Our algorithm is implemented on top of the conventional greedy algorithms, thus it maintains the good properties of greedy routing. Implementing it may only improve the performance of the system.

1 Introduction

Our aim in this paper is to propose an approach for routing in DHT's which is better than greedy routing. Greedy routing is a common approach in many DHT constructions. Typically some metric is imposed on the key space and then routing is performed by moving the message to the closest neighbor to the target. Examples include Chord [15], Skip Graphs or Skip Nets [2],[6], Pastry [14], Tapestry [16] and more. We discuss constructions which do not employ greedy routing towards the end of this section. The greedy routing approach has many advantages, among which are the following:

Simplicity - greedy routing is easy to understand and implement. The routing is oblivious in the sense that the link used depends only on the destination of the message.

Fault Tolerance - In greedy routing closer is better. So when nodes or links fail, as long as each node has some edge towards the target, it is guaranteed that the message would reach its destination. In many constructions (such as Chord and Tapestry) it is assumed that a ring like structure exists. It is clear that as long as the ring edges exist, greedy routing would eventually succeed.

Locality in the key space - Messages do not 'wander' in the key space. If the source and the target are close to each other in the key space, then during the routing, the message would stay between the source and the target, in a small portion of the key space. If the key of a resource is associated with its location in the physical world then greedy routing might have proximity preserving properties; i.e. it might minimize the physical distance a message travels. If keys have *semantic* meaning, such as file names and sizes, then greedy routing would supply prefix search (as in skip graphs, see Section 1.1).

If greedy routing is so good then why use something else? Greedy routing has one major disadvantage - it requires a large number of hops, at least larger than what is dictated by the degree. In the previous examples, the degree is logarithmic in the network size, while the number of hops used by greedy routing is $O(\log n)$. Logarithmic degree may permit theoretically path lengths of $O(\log n/\log\log n)$, thus greedy routing is *not* degree optimal. Furthermore, there are lower bounds that show that under some general conditions, greedy routing uses $\Omega(\log n)$ hops, when the degree is logarithmic, see [1],[11]. The latter bounds any routing algorithm which uses the immediate neighbor only.

Recently constructions with optimal path length were suggested. De-Bruijn based DHT's [13],[8],[4] offer an optimal tradeoff between degree and path length for every degree, in particular logarithmic degree permits routing in $O(\log n/\log\log n)$ hops. These routing algorithms however, are *not* greedy and rely on some arithmetic manipulation of the keys. Thus routing in these graphs is not local in the key space. Furthermore these algorithms assume keys are *random*, so there is no immediate way to make the keys carry semantic meaning.

Can we have the advantages of greedy routing together with optimal path length? In this paper we answer this question affirmatively. We show that a variation of the

*Research supported in part by the RAND/APX grant from the EU Program IST

†Incumbent of the Judith Kleeman Professorial Chair.

‡Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science. Rehovot 76100 Israel. {moni.naor,udi.wieder}@weizmann.ac.il

greedy algorithm called 'neighbor-of-neighbor (NoN) greedy' enjoys the advantages of greedy routing while being degree optimal in a large family of constructions. We show that our algorithm reduces substantially the latency of skip graphs and of small worlds. Furthermore while it is known that Chord is not degree optimal [5], we show that a variation of Chord is indeed degree optimal.

1.1 Small Worlds and Skip Graphs

The notion of 'small worlds' originated as a term to describe the high connectivity of social networks. Kleinberg [9] modelled social networks by taking a two dimensional grid and connecting each node u to q edges when edge (u, v) exists with probability proportional to $||u - v||^{-2}$. For simplicity, we remove the parameter q and assume that each edge (u, v) is connected with probability $||u - v||^{-2}$, thus creating a graph with average degree $\Theta(\log n)$. For any dimension $d > 1$, the small world graph of dimension d has n^d nodes associated with the points of a $d-$dimensional mesh, where edge (u, v) is occupied with probability $||u - v||^{-d}$. Small world graphs serve as a motivation for p2p networks. Indeed they were analyzed in this context by Aspnes *et al* [1] who proved that the one dimensional small world graph permits greedy routing of $O(\log n)$ even if nodes and links fail independently.

Skip-Graphs or Skip-Nets is a dynamic data structure meant to serve as a tool for locating resources in a dynamic setting. It was proposed independently by Aspnes and Shah in [2] and by Harvey *et al* in [6]. The main advantage of skip graphs is that they supply prefix search and proximity search. In a skip graph each nodes chooses randomly a string of bits called the *membership vectors*. The links are determined by the membership vectors, and therefore the keys could be arbitrary. In other words, there is no need for the keys to be randomized and they may maintain *semantic* meaning. It is therefore essential that routing algorithms remain local in the key space. This is achieved without compromising the complexity of the Insert and Delete operations, thus skip graphs(nets) are an especially attractive p2p construction. It is shown in [2] [6] that greedy routing takes $O(\log n)$ hops. It is shown in [11] that greedy routing takes $\Omega(\log n)$ hops.

2 The NoN-Greedy algorithm

The NoN approach originates from a work by Coppersmith *et al* [3], which used it to prove bounds on the diameter of the small world graph, though not in an algorithmic perspective. It was also used by Manku *et al* (under the name 'lookahead') as a heuristic for the 'Symphony' P2P construction [10] which is based upon small world graphs. We assume that each node holds its own routing table, and on top of that it holds its neighbors routing tables. Thus each node has knowledge of a neighborhood of radius 2 around it. The NoN algorithm is presented in Figure 1.

Algorithm for routing to node t.

1. Assume the message is currently at node $u \neq t$. Let $w_1, w_2, \ldots, w_k$ be the neighbors of u.
2. For each i, let $z_{i_1}, z_{i_2}, \ldots, z_{i_k}$ be the neighbors of w_i.
3. Among these k^2 nodes, assume z_{i_j} is the one closest to t.
4. Route the message from u via w_i to z_{i_j}.

Figure 1: The NoN-Greedy Algorithm. It is assumed for simplicity that the degree of each node is k.

The NoN algorithm could be thought of as greedy in two hops instead of just one. One might think that practically the NoN algorithm uses a routing table of size $O(\log^2 n)$. This however is not the case. In order to apply NoN, the *memory* allocated to hold the routing tables of the neighbors is $O(\log^2 n)$ instead of $O(\log n)$. The main cost of an entry in the routing table lies in its *maintenance* (pinging periodically etc.). The cost in terms of memory of simply holding the entry is marginal.

Step (2) of the algorithm is implemented internally by putting all the z in a search tree. Thus the time it takes to find the next link is the time it takes to find the correct link is the search time of the tree which is $O(\log(k^2))$. Assuming that $k = O(\log n)$, Step (2) takes $O(\log \log n)$ time, which is the same as in greedy routing.

NoN is not greedy since w_i might not be the closest node to t among the neighbors of u. While not being greedy per se it is clear that NoN enjoys the advantages of greedy. The following theorem is taken from a companion paper [11], and shows that asymptotically the NoN algorithm is optimal both for skip graphs and for small world graphs.

Theorem 2.1. *When using the NoN algorithm:*

(a) *The average number of hops it takes to route a message in a skip graph is* $O(\log n / \log \log n)$.

(b) *Using the NoN algorithm, the number of hops it takes to route a message in a small world graph of any dimension is* $O(\log n / \log \log n)$ *with high probability.*

2.1 An Interesting Phenomena

A common approach when constructing p2p systems is to try and 'emulate' dynamically a good static network.

Thus Chord, Pastry and Tapestry are 'inspired' by the hypercube. A perfect skip graph has a topology similar to that of Chord. A deterministic protocol that achieves it is presented by Harvey and Munro in [7]. The work of Ganesan and Manku [5] implies that the average diameter of a perfect skip graph is $\Omega(\log n)$, thus we conclude that the randomization of edges *reduces* the expected path length. See [11] for a discussion of this phenomena.

3 Simulation Results

We ran simulations in which we compared the performance of the greedy algorithm and the performance of the NoN greedy algorithm. We constructed a skip graph of up to 2^{17} nodes and a small world graph of up to 2^{24} nodes. In a small world graph it is not necessary to create the full graph in advance. Each time the message reached a node, we randomly created the neighborhood of radius 2 around the node. This is a negligible compromise over the definition of the model, since the edge in which the node was entered might not be sampled. This technique allowed us to run simulations on much larger graphs. For each graph size we ran 150 executions. A substantial improvement could be seen. Figures 2,3 demonstrate an improvement of about 48% for skip graphs of size 2^{17} and an improvement of 34% for small world graphs of size 2^{24}. Figure 2 also depicts the average shortest path in the graph. We see that the shortest paths may be 30% shorter than the paths found by NoN, yet even for moderate network sizes, the NoN algorithm performs substantially better than the Greedy one.

An even more impressive improvement could be seen when the size of the graph is fixed and the average degree changes. We fixed a small world graph of size 2^{20}. After that we deleted each edge with a fixed probability which varied from 0 (the usual small world graph) to 0.9 (a graph with roughly one tenth of the edges). Figure 4 depicts the results of these simulations. It shows that the reduction in the number of hops is more or less independent from the number of edges. The latency achieved by the Greedy algorithm when the degree is 26 is achieved by the NoN algorithm when the degree is merely 12. In the case of skip graphs we ran the simulation for a graph of size 2^{17} and varied the size of the alphabet of the membership vectors. When the alphabet size is s the average degree is $O(\log_s n)$. We can see in Figure 4 that NoN with alphabet size 20 is better than Greedy with alphabet size 2, i.e. when the average degree is $\log_2 20 \simeq 4.3$ bigger.

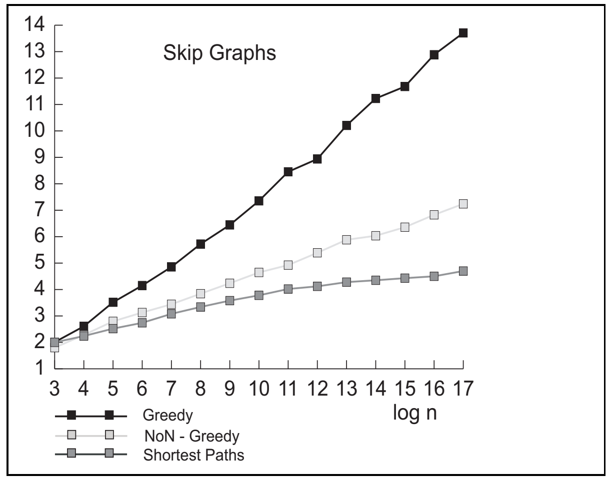

Figure 2: The number of hops in skip graphs.

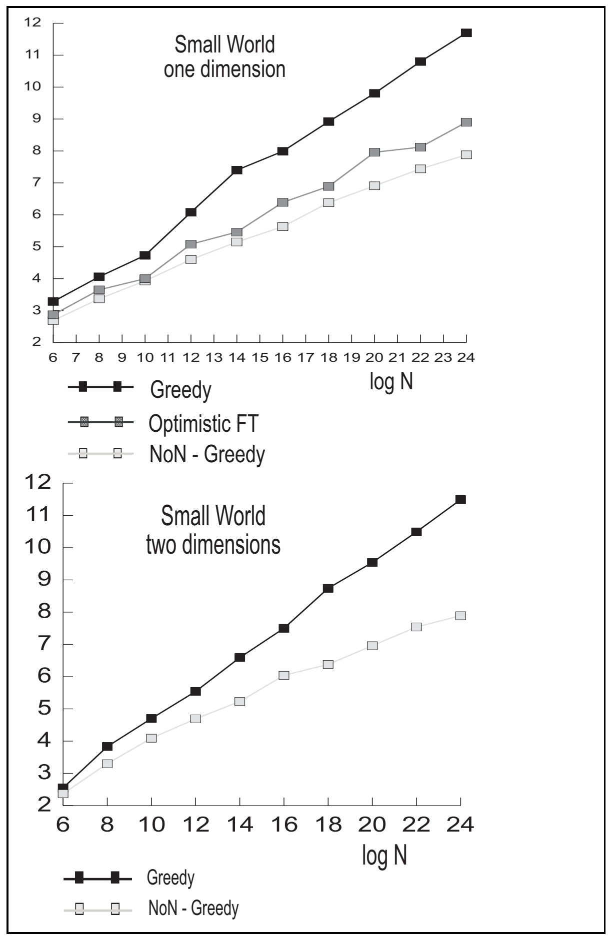

Figure 3: The number of hops in a small world of dimensions 1, 2.

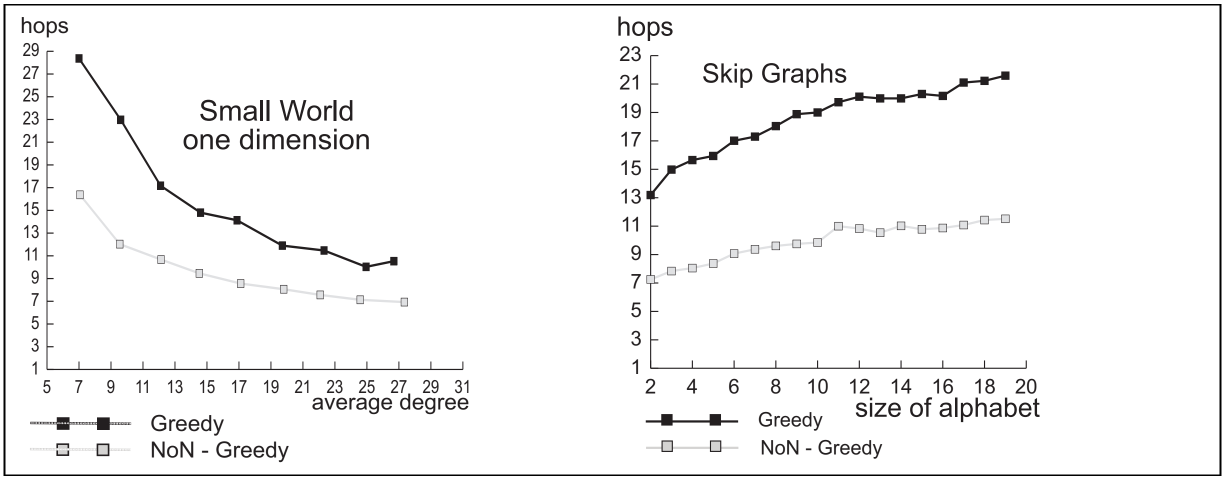

Figure 4: The tradeoff between average degree and latency in a small world with 2^{20} nodes (left) and skip graphs of 2^{17} nodes (right).

3.1 A different Implementation

The algorithm presented in Figure 1 is somewhat unnatural. Each NoN step has two phases. In the first phase the message is sent to a neighbor whose neighbor is close to the target. In the second phase a greedy step is taken (i.e. the message moves to the neighbor of neighbor). A $1-$phase implementation would let each node initiate a NoN step again, i.e. each node upon receiving a message, finds the closest neighbor of neighbor, and passes the message on. This variant is harder to analyze, indeed Theorem 2.1 holds for the $2-$phase version only. Yet, as Figure 5 shows, in practice the two variants have basically the same performance.

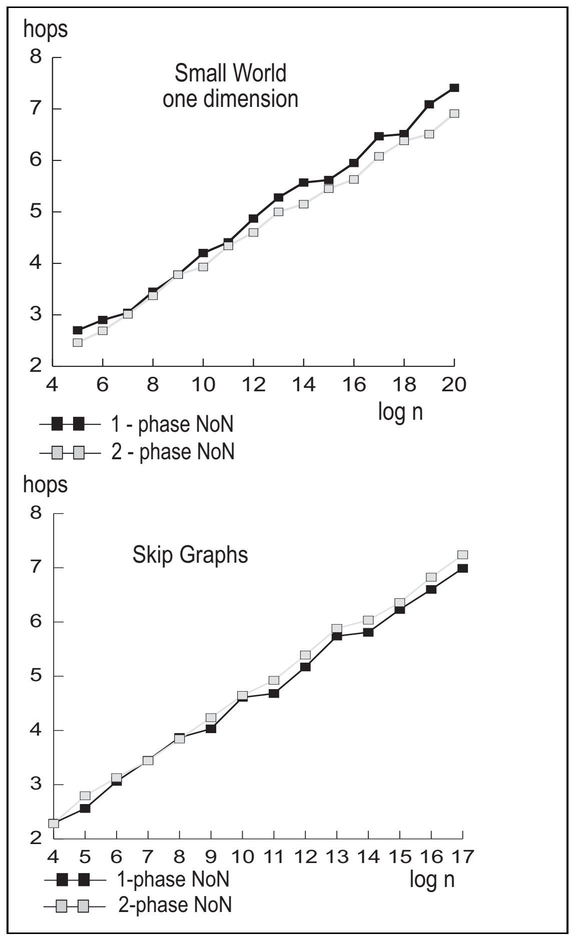

Figure 5: Comparison between the two variants of NoN

3.2 Fault Tolerance

The previous simulations assumed that the list of neighbor's neighbors each node holds is always correct. In reality this might not be the case. We examine two scenarios which capture the two extremes of this problem.

Optimistic Scenario: In this case we assume that a node knows whether its neighbors of neighbors lists are up-to-date or not. Whenever a node has a stale list is performs a greedy step. If a node cannot perform the NoN step from any other reason, it performs the greedy step instead. We ran simulations in which each node performs with probability $\frac{1}{2}$ a NoN step, and with probability $\frac{1}{2}$ a greedy step. Whenever a NoN step is performed, both phases of it are performed correctly. Figures 3,6 show that the total performance is hardly compromised. A small world of size 2^{22} suffered a relative delay of less than one hop, A skip

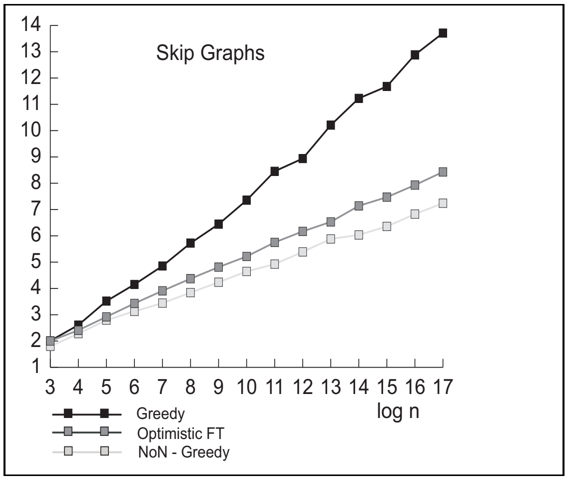

Figure 6: Optimistic fault tolerance in Skip Graphs

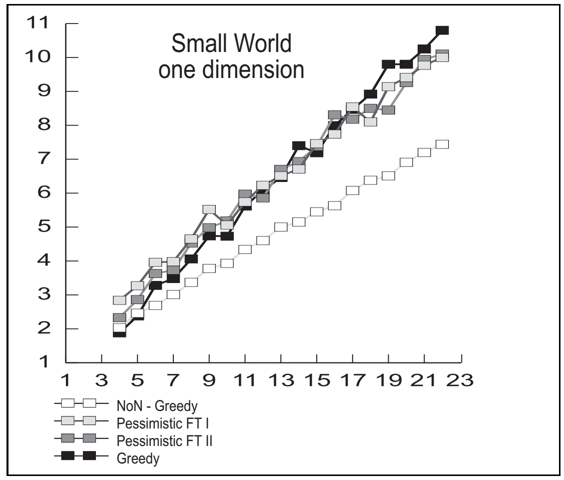

Figure 7: Pessimistic Fault Tolerance

graph of size 2^{17} suffers a relative delay of 1.2 hops. But why is the optimistic scenario justified? Our suggestion is that each node would calculate a hash of its neighbors list. This hash would be sent to all its neighbors on top of the maintenance messages. Thus with a minuscule overhead in communication each node would know whether its lists are up-to-date. We discuss specific hash functions in Section 4.

Pessimistic Scenario: In this scenario we assume that a node is unaware that its neighbor's neighbors lists are not up-to-date. So when node u passes a message to node w expecting it to move on to node z, with probability $\frac{1}{2}$ the edge (w, z) no longer exists. We tested two variants: in the first one, whenever this occurs the intermediate node w performs a greedy step. In the second variant the intermediate node w initiates another NoN step. The results of the simulations appear in Figure 7. It could be seen that in the pessimistic scenario, the performance of NoN is approximately the same as the Greedy algorithm.

We conclude that even if we assume that the neighbor lists are error prone, still the use of the NoN algorithm may be beneficial.

4 Implementation Issues

The execution of a NoN hop requires a node to store the neighbor lists of its neighbors. This implies that nodes should update each other regarding their own lists of neighbors. Such an update occurs in two scenarios:

1. Each node upon entrance, must send its list of neighbors to its neighbors.
2. Whenever a node encounters a change in its neighbor list (due to the entrance or exit of a node), it should update its neighbors.

The extra communication imposed by these updates is not heavy due to the two following reasons. First, assume nodes u, v are neighbors. Node u periodically checks that v is alive (for instance by pinging it). Checking whether v's neighbor list has changed could be added to the maintenance protocol by letting v send a hash of its neighbor list on top of the maintenance protocol. A possible hash function may be `MD5` (though the cryptographic properties of this hash function are not needed). Another possibility is simply to treat the id of neighbors as coefficients of a polynomial, and evaluate this polynomial at a random point. Either way the actual cost in communication is very small. When an actual update occurs there is no reason for v to send its entire neighbor list. It may only send the part of it which u misses. If it does not know which part it is then u, v may participate in a very fast and communication efficient protocol that reconciles the two sets, see e.g. [12] for details. The second reason the communication overhead is small is that the the actual updates are not urgent (as the simulations of the optimistic scenario show) and may be done when the system is not busy.

It is important to notice that the implementation of the NoN algorithm does not affect the Insert/Delete operations. Once a node enters, the needed updates should occur. We conclude that implementing NoN has very little cost both in communication complexity and in internal running time. It is almost a free tweak that may be implemented on top of the previous constructions.

5 Other Constructions - NoN-Chord

Does NoN-Greedy improve more p2p systems? In this section we show that a variation of Chord [15] is degree optimal. In Chord the key space is a ring $[0, 1, \ldots, n]$. A node whose i.d. is x is connected (more or less) to $x + 1, x+2, x+4, \ldots, x+\frac{n}{2} \mod n$. It was shown by Manku [5] that the *average* diameter of Chord is $\Theta(\log n)$, thus no algorithm may significantly reduce the path length. We show a slight variation of Chord, (in search of a better name lets call it NoN-Chord). The idea is to make Chord resemble the Small-World graph. Now each node x is connected to $\log n$ nodes $y_0, y_1, y_2, \ldots$ such that y_i is a *random* point in the segment $[x + 2^i, x + 2^{i+1}]$. An easy adaptation of the proofs in [11],[3] shows that w.h.p the path length used by NoN is $O(\log n/\log\log n)$. The Insert operation now, would take $\log^2 n/\log\log n$ operations, in $\log n/\log\log n$ parallel time. So we have an interesting tradeoff of parameters. An increase in the communication complexity of the Insert operation (though not in the time complexity) reduces the latency of the paths.

Acknowledgments

We thank James Aspnes and Gauri Shah for supplying us with their implementation of skip graphs, and the anonymous referees for useful comments.

References

[1] James Aspnes, Zoe Diamadi, and Gauri Shah. Fault-tolerant routing in peer-to-peer systems. In *Proceedings of the twenty-first symposium on Principles of distributed computing (PODC)*, pages 223–232, 2002.

[2] James Aspnes and Gauri Shah. Skip graphs. In *fourteenth ACM SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[3] Don Coppersmith, David Gamarnik, and Maxim Sviridenko. The diameter of a long-range percolation graph. *Random Structures and Algorithms*, 21(1):1–13, 2002.

[4] Pierre Fraigniaud and Philippe Gauron. The content-addressable network d2b. Technical Report LRI 1349, Univ. Paris-Sud, 2003.

[5] Prasanna Ganesan and Gurmeet Singh Manku. Optimal routing in chord. In *fifteenth ACM SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[6] Nicholas Harvey, John Dunagan, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *4th USENIX Symposium on Internet Technologies and Systems, (USITS)*, 2003.

[7] Nicholas Harvey and J. Ian Munro. Deterministic skipnet. In *Twenty Second Annual ACM Syposium on Priciples of Distributed Computing (PODC)*, pages 152–153, 2003.

[8] Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Second International Workshop on Peer-to-peer Systems IPTPS*, 2003.

[9] Jon Kleinberg. The Small-World phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, 2000.

[10] Gurmeet Singh Manku, Mayank Bawa, and Prabhakar Raghavan. Symphony: Distributed hashing in a small world. In *4th USENIX Symposium on Internet Technologies and Systems, (USITS)*, 2003.

[11] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *STOC*, 2004.

[12] Yaron Minsky and Ari Trachtenberg. Practical set reconciliation. Technical Report 2002-03., Boston University, 2002.

[13] Moni Naor and Udi Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures SPAA*, pages 50–59, 2003.

[14] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[15] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Frank Dabek Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. Technical Report TR-819, MIT LCS, 2001.

[16] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.