Degree–Optimal Deterministic Routing for P2P Systems

Gennaro Cordasco* Luisa Gargano* Mikael Hammar† Vittorio Scarano*

Abstract

We propose routing schemes that optimize the average number of hops for lookup requests in Peer–to–Peer (P2P) systems without adding any overhead to the system. Our work is inspired by the recently introduced variation of greedy routing, called neighbor–of–neighbor (NoN), which allows to get optimal average path length with respect to the degree. Our proposal has the advantage of first "limiting" and then "eliminating" the use of randomization. As a consequence, the NoN technique can be implemented with our schemes without adding any overhead. Analyzed networks include several popular topologies: Chord, Hypercube based networks, Symphony, Skip-Graphs. Theoretical results and extensive simulations show that the proposed simplifications (while maintaining the original node degree) do not increase the average path length of the networks, which is often improved in practice. The improvement is obtained with no harm to the operational efficiency (e.g. stability, ease of programming, scalability, fault–tolerance) of the considered systems.

1 Introduction

Peer–to–Peer file sharing applications quickly became very popular in the recent years. Several of the recently proposed systems are completely distributed and use a scalable Distributed Hash Table (DHT) as a sublayer. A DHT is a self–organizing overlay network that allows to add, delete, and look up hash tables. Several proposals have been made recently of systems where hosts configure themselves into a structured network such that lookups require a small number of hops.

RELATED WORK. The greedy routing approach, in which the message is routed through the neighbor which is nearest to the target, has been used in most of the proposed P2P networks. These include [9, 3, 10, 2, 7]. Several reasons make greedy a popular strategy. In particular, one of the main advantages is that greedy routing is very simple to implement and has some "implicit" fault-tolerance capability. It was however noticed that greedy routing usually produces paths of length larger than what would be required in a network of the given node degree. As an example, some popular topologies like Chord have degree $O(\log n)$ and the greedy routing produces an average path length $O(\log n)$ whereas the lower bound is $\Omega(\log n/\log\log n)$.

The use of randomization allowed to exhibit networks with optimal average path length [5]. Recently, constructions based on de Bruijn graphs [4] exhibited optimal trade–offs between degree and latency with deterministic routing; these algorithms are not greedy and present some other disadvantages as discussed in [8].

Recently a novel approach for routing in DHTs which improves on greedy routing has been proposed [8]. This approach, called *NoN* (Neighbors–of–Neighbors) or *1-lookahead routing*, substantially consists in making the greedy choice by looking not only at the neighbors of a node but at all the nodes at distance at most two from the node itself. The NoN approach together with the use of randomization in establishing the neighbors of the nodes which are present in the network, can optimally reduce the latency in several well known topologies [8].

OUR RESULTS. We show how to retain the improvements given by the NoN routing over randomized networks, while eliminating the drawback in system overhead implied by this technique.

It is well known that an average path length $O(\log n/\log\log n)$ is optimal when the degree is $O(\log n)$; however, this low latency is obtained by NoN routing over randomized networks to the prejudice of the system overhead. As a matter of fact, the result in [8] is obtained by trading off programmability and feasibility with efficiency since in NoN routing over randomized networks a certain amount of overhead is heavily underestimated. In this paper, we show how to eliminate the extra-communication at all and, similarly, eliminate the need of storing in each node its neighbors of neighbors.

By using both randomization and NoN routing it is necessary to transmit to a node its neighbors of neighbors: in [8], the authors argue that it can be done without extra cost

*Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi (Salerno), Italy. E-mail: {cordasco,lg,vitsca}@dia.unisa.it. The work of these authors was partially supported by Italian FIRB project WebMinds.

†Research & Development, Apptus Technologies AB, IDEON, 223 70 Lund, Sweden, E-mail: Mikael.Hammar@apptus.com.

by using keep-alive TCP messages. We believe that this is not a suitable solution since transport protocols should not be tampered with by any application protocol, because of abstraction requirements. Anyway, by following their suggestion, performances predicability of a NoN implementation can be seriously limited by underestimating this overhead[1] in the analysis.

In order to eliminate at all these drawbacks, we need to eliminate the random factor in establishing each neighbor of a node. In fact, determinism allows each node to calculate locally the neighbors of its neighbors. Randomness is eliminated as follows. First, we prove that it is possible to reduce the use of random values from $\log n$ to *one random number* for each node without degrading the efficiency of the system with respect to the NoN routing as proposed in [8]. Actually, we show that with such a modification efficiency is improved. On the basis of the above result we can then replace the random number by a hash-function, thus having a completely deterministic network. Indeed, hashing can be done (as an example) on the node ID so that node y, knowing the ID x also knows x's neighbors[2]. Hence, no additional information is transmitted nor stored for NoN routing. Analyzed networks include several popular topologies: Chord, Hypercube based networks, Symphony and Skip-graphs.

Theoretical results and extensive simulations show that the proposed simplifications (while maintaining the original node degree) do not increase the average path length of the networks, on the contrary it is often improved in practice. The experiments performed exploits the Secure Hash Algorithm (SHA-1).

It follows that our approach leads to "deterministic" overlay networks which maintain the original degree and the optimal average path length of the corresponding randomized versions, while improving on the system overhead.

2 Preliminaries

In this section we report the definition of the existing networks that are of interest for the rest of the paper and of the NoN greedy algorithm.

We consider a set N of n nodes lying on a ring of 2^m identifiers (labeled from 0 to $2^m - 1$ in clockwise order). Each node x, has an m bit ID and is connected[3] with its predecessor and its successor on the ring.

- **Chord:** For each $0 \leq i < m$, node x is connected by edges to the nodes[4] $x + 2^i$.

  R-Chord:[6] For each $0 \leq i < m$, let $r(i)$ denote an integer chosen uniformly at random from the interval $[0, 2^i)$, node x is connected by edges to the nodes $x+2^i+R(i)$.
- **Hypercube:** For each $0 \leq i < m$ node x is connected with the node y_i where x and y_i differ in exactly the position i.

  R-Hypercube:[6] For each $1 \leq i \leq m$, node x makes a connection with node y_i where y_i is defined as follows: The top $i-1$ bits of y_i are identical to those of x. The i^{th} is flipped. The remaining $m - i$ bits are chosen uniformly at random.
- **Symphony*:**[6] Symphony* is derived from Symphony [7]. Let δ denote a real number satisfying[5] $\ln \delta = (\ln 2^m)/k$. Let $\mathcal{I}_1 = [1, \delta]$. For $1 < i \leq k$, let $\mathcal{I}_i = (\delta^{i-1}, \delta^i]$. For interval $\mathcal{I}_i$, let ϕ^i denote a probability distribution over integers in $\mathcal{I}_i$ such that the probability at integer d is proportional to $1/d$. For each $1 \leq i \leq k$ an edge is established from node x to a node $y_i = x + a_i$, where a_i is an integer drawn from ϕ^i.
- **Skip-graph:** [2] In a Skip-graph, the out-going edges of a node x are determined by $m(x)$, its *membership vector*, which is an infinite string of *random bits*[6]. Two nodes are connected by an edge if their corresponding membership vector share some prefix which is not shared by any of the nodes between them. Formally, let $m_k(x)$ denote the first k bit of $m(x)$. The nodes x and y are connected by an edge if there exists some k such that $m_k(x) = m_k(y)$ and $\forall z$ in the interval (x, y) it holds that $m_k(z) \neq m_k(x)$. The cycle edges could be viewed as corresponding to the empty prefix. It is easy to see that w.h.p., all nodes have a logarithmic degree.

Skip-graphs [2] are different from the other three networks that we just presented. In fact, in Skip-graphs, links are determined by the membership vector, 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. Hence the main advance of Skip-graphs is that they supply prefix search and proximity search. However, Skip-graphs do not address the issue of load balancing the number of items per node.

THE NON-GREEDY PROTOCOL. We assume that each node holds its own routing table, and on top of that holds its neighbors routing tables. Let $d(x, y)$ be a metric for the nodes in the network. Here is the description of the routing algorithm NoN-Greedy:

1. Assume the message is currently at node $u \neq$target.
2. Let $V = \{v_1, v_2, \ldots, v_k\}$ be the neighbors of u. For each $1 \leq i \leq k$, let $w_{i1}, w_{i2}, \ldots, w_{ik}$ be the neighbors of v_i and let $W = \{w_{ij} \mid 1 \leq i, j \leq k\}$.

[1] Later, in Sec. 5, we prove that the overhead is, indeed, significative.

[2] Node y knows the exact position of x's neighbors only if the network is full (i.e. each ID is present). If the network is not full, node y knows the IDs whose successors are the x's neighbors (as defined in Section 2). We stress that this is sufficient for the correct functioning of the network.

[3] When we say that a node x is connected to y, we means that x is connected to y if y is a node in N, otherwise x is connected to the first node that follows y.

[4] All the arithmetic operation on the ring are done $\mod 2^m$.

[5] Symphony* can have arbitrary degree k.

[6] We think of the length of $m(x)$ as infinite for convenience even though $O(\log n)$ bits suffice with high probability, in a network of n nodes.

3. Among the k^2+k nodes in $V \cup W$, assume that z is the closest to the target (with respect to metric d).
4. If $z \in V$ route the message from u to z else $z = w_{ij}$, for some i and j, and we route the message from u **via** v_i to z.

Remark: *We call the (standard) definition given above 2-phase NoN that has been theoretically analyzed. A more efficient definition, call it 1-phase NoN, can be obtained by replacing Step 5 as follows:*

4'. If $z \in V$ route the message from u to z else $z = w_{ij}$, for some i and j and we route the message from u **to** *v_i.*

Intuition can easily support the claim of efficiency of 1-phase NoN: Re-applying the whole algorithm at node v_i can only extend the choices.

In the NoN-Greedy algorithm, v_i may not be the neighbor of u which is closest to the target (with respect to metric d). The algorithm could be viewed as a greedy algorithm on the square of the graph – a message gets routed to the best possible node among those at distance two.

3 H-Networks

In this section we describe the novel version of Chord, Hypercube and Symphony* based networks, called H-Chord, H-Hypercube and H-Symphony*, and show that the average path length is $O(\log n/\log\log n)$ hops for performing lookups. The routing table size is $O(\log n)$ for Chord and Hypercube while Symphony* has routing table size k and the latency is $O\left((\log^2 n)/(k\log k)\right)$ hops on average.

The key observation in our results is that generating one single random number for each node in the network is enough to preserve the $O(\log n/\log\log n)$ hops (on avg). Routing occurs through canonical NoN. The random number is replaced by a hash-function defined on the node IDs to get a deterministic algorithm with the same properties.

H-CHORD. Below we give a new network based on Chord. By using hashing together with the new network we get the same result as the NoN protocol for R-Chord [6], but with the same message complexity for maintaining the network intact but without any additional overhead as described in the Introduction. The network is defined as follows.

Definition (H-Chord) *Let $H()$ denote a hash function, that maps an id on the interval $[0,1)$. Each node x is connected by an edge to the node $x + 2^i + \lfloor H(x)2^i \rfloor$, for $i = 0, \ldots, m-1$.*

We analyze the protocol assuming to have a good hash function, that generates integers taken uniformly at random from the interval $[0,1)$.

We first consider the case of a full network, that is, $n = 2^m$; afterwards we extend to any $n \leq 2^m$.

Lemma 1 *The average path length is $O(\log n/\log\log n)$ hops for the NoN Greedy algorithm on H-Chord with $n = 2^m$ nodes.*

Proof : We consider a source node s that wants to send a message to a node t at distance $d(s,t) = d$. Let p be the unique number with $2^p \leq d < 2^{p+1}$. There are two cases to consider. In the first case $p \leq \log n/\log\log n$. In this case it suffices with $O(p)$ hops to reach the destination, since as for R-Chord, the distance decreases at least with a factor of $3/4$ for each hop executed.

In the second case $p > \log n/\log\log n$. Let $I = (d-d', d]$, where $d' = \lceil d\log\log n/\log n \rceil$. Now, s has at least $p-1$ neighbors $s_1, \ldots, s_{p-1}$ in the interval $[s, s+d]$. Moreover, let $s_0 = s$ and $S = \bigcup_{i=0}^{p-1} s_i$, we are investigating the probability of S having an outgoing edge entering the interval I, i.e.,
$P = Pr[J_k(s_i) \in I$ for some $0 \leq i < p$ and $0 \leq k < m]$
where $J_k(s_i) = s_i + 2^k + \lfloor H(s_i)2^k \rfloor$ denotes the k^{th} neighbor of s_i. Let us assume that, for any node $s_i \in S$, the probability that it has an outgoing edge entering the interval I is $d'/2^p$. Each of these nodes have chosen its neighborhood independently from the others, hence, the probability that none of these nodes have an outgoing edge reaching the interval I is $\left(1-\frac{d'}{2^p}\right)^{|S|} \leq \left(1-\frac{2^p\log\log n}{2^p\log n}\right)^{\frac{\log n}{\log\log n}} \leq e^{-1}$, since $p > \log n/\log\log n$. Hence, the probability that s can reach the interval in two hops is at least $1-e^{-1}$. Thus, in $O(\log d/\log\log n)$ hops, the distance is decreased to $2^{p'}$, where $p' < \log n/\log\log n$, and we have reduced case two into case one.

Left is to prove the assumption. Consider a node s_i at most a distance d from I. We are investigating the probability of s_i having an outgoing edge entering the interval I, i.e., $Pr[J_k(s_i) \in I$ for some $0 \leq k < m]$.
There are two cases to consider.

1. $d - d' \geq 2^p$. In this case the probability that $J_p(s_i)$ reaches the interval is equal to $d'/2^p$.
2. $d - d' < 2^p$. The probability that one of the hops reaches the interval I is equal to
   $Pr[J_{p-1}(s_i) \in (d-d', 2^p)$ or $J_p(s_i) \in [2^p, d]]$.
   We can observe that

   Claim 1 *$J_p(s_i) \in (2d-2d', 2^{p+1})$ implies that $J_{p-1}(s_i) \in (d-d', 2^p)$.*

   By noticing that, for any $n > 16$, the intervals $(2d-2d', 2^{p+1})$ and $[2^p, d]$ do not overlap,

$$\begin{aligned} &Pr\left[J_{p-1}(s_i) \in (d-d',\ 2^p) \text{ or } J_p(s_i) \in [2^p,\ d]\right] \\ &= Pr\left[J_p(s_i) \in (2d-2d', 2^{p+1})\right] + Pr\left[J_p(s_i) \in [2^p, d]\right] \\ &= \frac{2^{p+1} - 2d + 2d' - 1 + d - 2^p + 1}{2^p} > \frac{d'}{2^p}. \end{aligned}$$

Actually, if we go into the details of Theorem 5.2 in [6], we can see that the H-Chord network works a little better in Case 2, because of Claim 1. □

We can also generalize the results to hold in a ring where not all nodes are present. Due to Chord constraints the n

nodes can be assumed to be uniformly distributed [9].

Theorem 1 *The average path length is $O(\log n/\log\log n)$ hops for the NoN Greedy algorithm on H-Chord in a ring of size 2^m where the number of nodes alive is $n < 2^m$.*
Proof. Omitted.

H-HYPERCUBE. We can obtain the same improvement in the Hypercube.
Definition (H-Hypercube) *Let $H()$ denote a hash function mapping an id on a sequence of m bits. For each $1 \leq i \leq m$, node x makes a connection with node y_i where y_i is defined as follows: The top $i-1$ bits of y_i are identical to those of x. The i^{th} is flipped. The remaining $m-i$ bits are identical to those of $H(x)$.*

Theorem 2 *The average path length with NoN greedy routing on the H-Hypercube is $O(\log n/\log\log n)$ hops.*
Proof. Omitted.

H-SYMPHONY*. We are also able to improve the Symphony* protocol so that we generate only one random number that determines all the outgoing edges of a node.
Definition (H-Symphony*) *Let $H()$ denote a good hash function that maps an id on the interval $[0,1)$ and let $r = H(x)$. The procedure for constructing the edges of a node is changed as follows: For each ϕ^i, let $a_1, \ldots, a_q$ denote the integers in the interval $I_i = (\delta^{i-1}, \delta^i]$. To choose the destination of the ith out-going edge y_i of a node x, we use the mapping*

$$f_i(r) = a_l, \text{ if } \sum_{j=0}^{l-1} \frac{1}{s_i a_j} \leq r \leq \sum_{i=0}^{l} \frac{1}{s_i a_j},$$

where $s_i = \sum_{j=1}^{q} 1/a_j$.

By definition of H-Symphony*, the probability of choosing the destination a_j is equal to $1/s_i a_j$.

By choosing only one random number we gain in network maintenance, since in order to compute the neighbors of a neighbor, it is sufficient to know the random number chosen by the node. Thereafter we can directly compute the out-going edges of the neighbor using the mappings f_i.

As we will also see, we can get a slightly smaller average path length due to the following lemma.

Lemma 2 *Consider an interval $I = [a, b]$, with $a \in I_i$ and $b \in I_{i+1}$. The probability that a node s has an out-going edge entering I is $Pr[J_i(s) \in I \cap I_i] + Pr[J_{i+1}(s) \in I \cap I_{i+1}]$, where $J_k(s)$ denotes the k^{th} neighbor of s.*
Proof. Omitted.

Consider now the routing procedure in the H-Symphony* network. In the following, we will show that average path length when using the greedy algorithm or the NoN greedy algorithm remains the same as for the original network.

Theorem 3 *The average path length is $O\left((\log^2 n)/k\right)$ hops for the greedy routing on H-Symphony*.*
Proof. Omitted.

Theorem 4 *The average path length is $O\left((\log^2 n)/(k \log k)\right)$ hops for the NoN greedy algorithm on H-Symphony*.*
Proof. Omitted.

Actually, if we go into the details of Theorem 5.3 in [6], we see that the H-Symphony* network works a little bit better in some cases, because of the independency shown in Lemma 2.

4 H-Skip-graph

In this section, we, first, comment on the fact that the technique used for Chord, Hypercube and Symphony* cannot be fruitfully applied on skip-graphs. Nevertheless, we show that, in the modified version of Skip-graphs by changing the metric we can obtain better results.

Let us define, first, the modified version of Skip-graphs.
Definition (H-Skip-graph) *Let H() denote a hash function that maps an id on a sequence of $O(\log n)$ bits. H-Skip-graphs are identical to the skip-graphs, but with $m(x)$=$H(x)$.*

Let S be a generic Skip-graph with n nodes, for each possible binary sequence p of size k, we denote with S_p a ring that contains all the nodes of S such that p is a prefix of their membership vector. Formally $S_p = \{v \in S \text{ s.t. } m_k(v) = p\}$. In particular, if we denote with ϵ the empty sequence, $S_\epsilon = S$. It is easy to show that the expected number of nodes on a Ring S_p is $\frac{n}{2^k}$.

NoN greedy routing on the modified version of Skip-graphs does not avoid the system overhead of knowing the neighbors' neighbors. To wit, a node in H-Skip-graphs, in spite of using a deterministic hash function, has no way of deriving the IDs of its neighbors' neighbors from the membership vectors, since that information depends globally on the network topology. The membership vectors of its neighbors does not help in this case. As a consequence, when we use NoN greedy routing the system overhead of transmitting and storing neighbors' neighbors cannot be avoided.

Nevertheless, by using a deterministic hash function the *membership vector* of the destination becomes available to the source, and a more efficient search is now possible.

In the three networks previously analyzed, the metric that was used in the routing was the distance between nodes in terms of IDs (i.e. the distance on the ring):

$$d(x, y) = (y + 2^m - x) \bmod 2^m.$$

When we know the *membership vector* of the destination we can define a new metric and we can use both the greedy and NoN routing strategy with the new metric. Our goal is to evaluate the distance from two nodes on the smallest ring that contains both nodes. We define the new metric as:

$$d'(x,y) = \frac{(y+2^m-x) \bmod 2^m}{2^{|lcs(m(x),m(y))|}}$$

where $lcs(m(x), m(y))$ computes the longest common prefix between $m(x)$ and $m(y)$.

This new metric is a better estimate of d since it measures the distance between x and y on the smallest ring that contains both x and y. In particular, if $m(x)$ and $m(y)$ share no prefix (i.e. $lcs(m(x), m(y)) = \epsilon$) then $d(x,y) = d'(x,y)$. To provide more intuition: two nodes x and y can be far apart on the ring S_ϵ but can be very close within a small size ring S_p, where p is their common prefix.

5 Maintenance Overhead

In this section we consider the communication complexity of maintaining neighborhood information. We restrict our discussion to the Chord-like systems although the same line of arguments are suited also for the Hypercube, Symphony* and Skip-graphs networks. Note that R-Chord uses $\Omega(log^2 n)$ words of memory, since it keeps track of all the NoN information, whereas the H-Chord network only needs a storage of $O(\log n)$.

As explained by Manku et al. [8], R-Chord uses the same network update algorithms as Chord. In addition, the neighbors of neighbors information is communicated during basic network maintenance.

We already commented on the feasibility of this suggestion in the Introduction. Here, we try to estimate the overhead caused by this communication. The careful reader already noticed that H-Chord uses the same network maintenance protocol as Chord, i.e., without any extra overhead. This means that the exact neighbors of neighbors information is not available in H-Chord, since the nodes suggested by the hash-function might not be alive.

However, the hash-function provides us with the node IDs of the neighbors' neighbors. This information is sufficient to perform efficient lookups. The node ID is namely a lower bound on the distance to the neighbor in question. Hence, we can use this distance to estimate the search progress in each step. It follows that using the estimated distances in Theorem 1 we get an upper bound on the number of hops needed to reach the target.

Below we describe the additional communication costs of R-Chord. Since Manku et al. do not provide exact algorithms we give lower bounds on the communication cost and remarks stating the probable overhead in practice.

Theorem 5 *The communication complexity, using R-Chord together with the 1-lookahead protocol, for maintaining the network structure is $\Omega(\log n)$ messages for each node and each round if no failures, joins or leaves occur.*
Proof. Omitted.

Remark: *If the probability of neighborhood-updates is high, it is more likely that the protocol directly checks which neighbors of the neighbor have changed using a $\log n$ bit word. Hence, the communication complexity will be $\Omega(\log^2 n)$ bits/node and round in practice.*

Theorem 6 *Every* join, leave *or* failure *incur a message overhead of $\Omega(log^2 n)$ messages and $\Omega(\log^3 n)$ bits.*
Proof. Omitted.

6 Experimental Results

We report some results of our validating simulations. We ran simulations to compare the performances of the greedy routing and NoN routing on the randomized networks with the same routing algorithms on our deterministic version of each network. The performance are measured in terms of average path length.

Our goal was to show that no hidden constant (in the big-Oh notation) in the theoretical results of the previous sections could limit the significance of the deterministic networks that we propose here.

The results are encouraging: our deterministic networks behave equally well as the randomized version in our simulations and do not suffer the drawback of system overhead for keeping and transmitting information on neighbors of neighbors.

In Fig. 1 we report the results on Chord. Similar results are obtained for the Hypercube while experiments for Symphony* are in progress. In Fig. 1 (left) we show the average path length for Chord with up to 2^{18} IDs and nodes, i.e. a ring full of nodes. In Fig. 1 (right) we show the results for a ring of 2^{32} IDs where the number of nodes varies from 2 to 2^{18}. In both figures, we can see that H-Chord behaves as well as R-Chord. It should be noticed that, in our evaluation, we used, for all the networks, the 1-phase NoN that is more efficient than the 2-phase NoN.
In Fig. 2, we show simulation results for the average path length of Skip-graphs. In this case, we first need to compare the (randomized) Skip-graphs with our definition of H-Skip-graphs. In Fig. 2 (left) we show that the deterministic version of Skip-graphs works as efficiently as the randomized, standard one with respect to Greedy and NoN routing strategies. As noticed in Section 4, using a hash function does not avoid the costly operation of knowing the neighbors' neighbors (as happens, on the contrary, for Chord, Hypercube and Symphony*). In Fig. 2 (right) our goal is to evaluate the improvement by using the new metric d' in applying greedy and NoN routing strategies. We show that a significant improvement is obtained by using d' as a metric in applying both routing strategies, but the improvement is more marked for the greedy strategy. In particular, we would like to emphasize that the greedy routing with our new metric is only slightly less efficient than the standard NoN routing, but has no additional overhead.

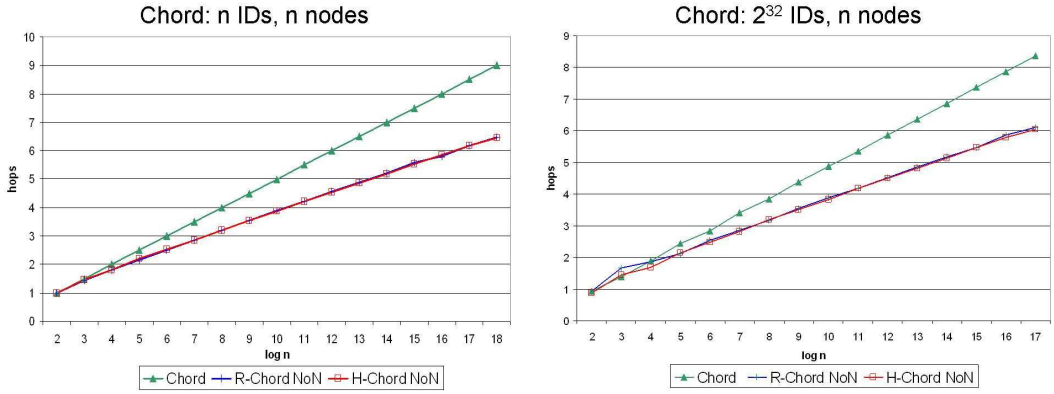

Figure 1: The randomized version of Chord is compared with H-Chord (left) with full network of sizes from 2 to 2^{18}; (right) with nodes from 2 to 2^{18} on a fixed size ring with key space size 2^{32}.

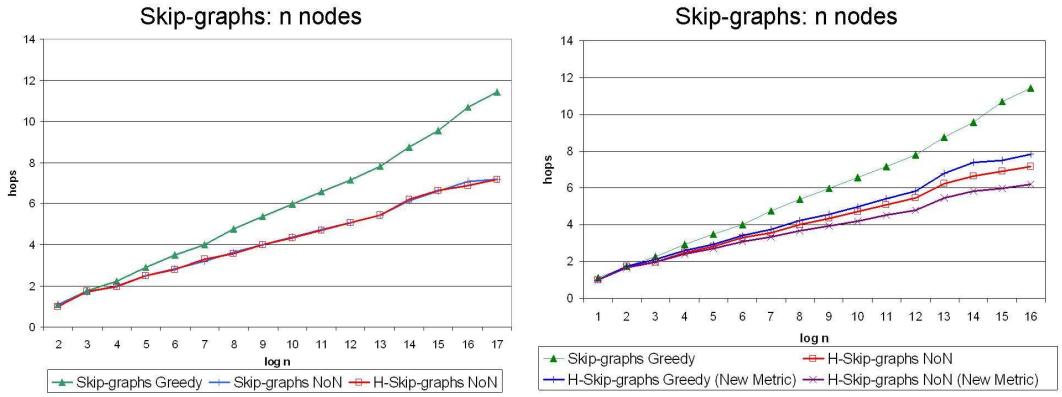

Figure 2: (left) The standard Skip-graphs compared with H-Skip-graphs. (right) H-Skip-graphs compared with a different distance metric, evaluating two strategies: Greedy and NoN with network of sizes from 2 to 2^{17}.

7 Conclusions and Discussion

We propose routing schemes that optimize the average number of hops for lookup requests in Peer–to–Peer systems. Unlikely other proposed system our scheme does not add any overhead to the system. Recently introduced variation of greedy routing, called neighbor–of–neighbor (NoN), allows to get optimal average path length with respect to the degree; overhead is payed compared to previous systems due to additional network maintenance. Our proposal has the advantage of "limiting" randomization to such an extent that neighborhood information can be encoded within the hash-value of the node ID. This enables us to use NoN lookup routing without any additional overhead.

References

[1] National Institute of Standards and Technology, Secure Hash Standard. http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[2] J. Aspnes and G. Shah. Skip graphs. Proc. of 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Jan 2003.

[3] P. Druschel and A. Rowstron. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science, 2218, Nov 2001.

[4] M. Kaashoek and D. Krager. Koorde: A simple degree-optimal distributed hash table. Proc. of IPTPS, Feb 2003.

[5] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. Proc. of ACM Symp. on Principles of Distr. Comp. (PODC), Aug 2002.

[6] G. Manku. The power of lookahead in small-world routing networks. Tech. Rep., CS Dept, Stanford Univ., Nov 2003.

[7] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. Proc. USENIX Symp. on Internet Tech. and Systems (USITS), Mar 2003.

[8] G. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: The power of lookahead in randomized p2p networks. Proc. of STOC, June 2004.

[9] I. Stoica, R. Morris, D. Liben-Nowell, M. F. K. D. R. Karger, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In IEEE/ACM Transactions on Networking, 12(2):205, Apr 2004.

[10] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep., CS Dept, Univ. of California at Berkeley, 2001.