On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks

Jun Xu
College of Computing
Georgia Institute of Technology
jx@cc.gatech.edu

***Abstract*—**

In this work, we study a fundamental tradeoff issue in designing dynamic hash table (DHT) in peer-to-peer networks: the size of the routing table v.s. the network diameter. It was observed in Ratnasamy et al. that existing DHT schemes either (a) have a routing table of size $O(log_2 n)$ and network diameter of $O(log_2 n)$, or (b) have a routing table of size d and network diameter of $O(n^{1/d})$. They asked whether this represents the best asymptotic "state-efficiency" tradeoffs. Our first major result is to show that there are routing algorithms which achieve better asymptotic tradeoffs. However, such algorithms all cause severe congestion on certain network nodes, which is undesirable in a P2P network. We then define the notion of "congestion-free" and conjecture that the above tradeoffs are asymptotically optimal for a congestion-free network. Though we are not able to prove (or disprove) this conjecture in full generality, our rigorous formulation of the problem and techniques introduced in proving slightly weaker results serve as the basis for further exploration of this problem. Our second major result is to prove that, if the routing algorithms are symmetric, the aforementioned tradeoffs are asymptotically optimal. Furthermore, for symmetric algorithms, we find that $O(log_2 n)$ is a magic threshold point for routing table size as follows. The "congestion" factor dominates the "reachability" factor in determining the minimum network diameter when the routing table size is asymptotically smaller than or equal to $O(log_2 n)$, and it is the other way around when the routing table size is asymptotically larger than $O(log_2 n)$. Our third and final major result is to study the exact (instead of asymptotic) optimal tradeoffs. We propose a new routing algorithm that reduces the routing table size and the network diameter of Chord both by 21.4% without introducing any other overhead, based on a novel number-theoretical technique.

I. Introduction

As peer-to-peer (P2P) file sharing systems become increasingly popular in recent years, scalability has been recognized as the central challenge in designing such systems. Early systems such as Napster and Gnutella all have some design limitations that prevent them for being scalable: Napster uses centralized directory service, while Gnutella employs flooding when searching for objects. To meet this challenge, various distributed hash table (DHT) schemes have been proposed in different P2P systems [1], [2], [3], [4], [5]. The basic idea of a DHT scheme is to use a hash table-like interface to locate the objects, and to distribute the duty of maintaining (in the face of node joins/leaves) the hash table data structure to all participating P2P nodes. In DHT schemes, each node stores files that correspond to a certain portion of the key space, and uses a routing table (referred to as a "finger table" in Chord [4]) to forward the request for an object not belonging to its key space to appropriate "next-hop" nodes. The request will eventually be forwarded to a node responsible for (the key of) the object through a chain of such "next-hops".

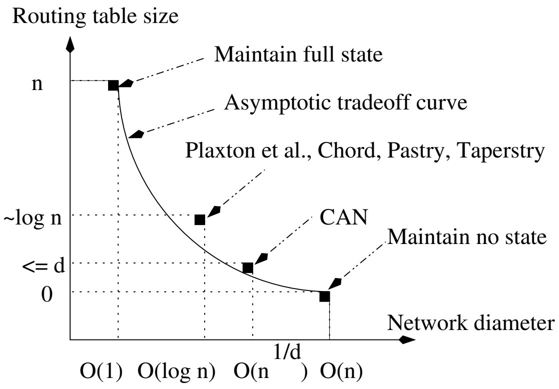

Fig. 1. Asymptotic tradeoff curve between the routing table size and the network diameter

This paper studies a fundamental tradeoff issue in designing DHT: the number of neighbors (equivalently the size of the routing table) vs. the network diameter, the number of hops a request needs to travel in the worst case. In a network consisting of n nodes, it is straightforward to see that when n neighbors are maintained (the "full directory" case) at each node, the search cost is $O(1)$, and when each node only maintains one neighbor (essentially a "logical ring"), the search cost is $O(n)$. This plots two points on the tradeoff curve shown in Fig. 1. In practical systems, neither extreme is desirable: the "full directory" approach involves heavy maintenance cost due to frequent *joins* and *leaves* of the P2P nodes, and the $O(n)$ diameter incurs intolerable network delay. Such a tradeoff has been referred to as the "state-efficiency" tradeoff[1] in [7]. It was observed in [7] that existing DHT schemes either (a) have a routing table of size $O(log_2 n)$ and network diameter of $O(log_2 n)$, which includes Chord [4], Taperstry [1], Plaxton et. al. [2], and Pastry [3], or (b) have a routing table of size d and network diameter of $O(n^{1/d})$, which includes CAN [5]. It was asked in [7] whether $\Omega(log_2 n)$ and $\Omega(n^{1/d})$ are the asymptotic lower bounds for the network diameter when the routing table sizes are $O(log_2 n)$ and d, respectively. We clarify and rigorously formulate this interesting question, and answer it in a comprehensive way.

The first major result of the paper is to clarify this tradeoff problem. We first formally characterize the metrics involved in the tradeoff. Then we show that there are routing algorithms which achieve better asymptotic tradeoffs than both (a) and (b) above[2]. However, we observe that these algorithms all cause severe congestion on certain network nodes even when the load is assumed to be uniform. In retrospect, one can intuitively see that "congestion-free" is indeed one of the key properties a good DHT routing algorithm should possess. Finally, based on this observation, we define the notion of "congestion-free". We conjecture that if the network is required to be "congestion-free", the aforementioned tradeoffs (a) and (b) are indeed asymptotically optimal. Though we are not able to prove (or disprove) this conjecture in full generality, our rigorous formulation of the problem and the techniques introduced in proving slightly weaker results (described later) serve as the foundation for further exploration of this problem.

The second major result of the paper is that, if the routing algorithms are symmetric[3] (defined later), we can prove that the aforementioned tradeoffs are indeed optimal. Furthermore, we show that $O(log_2 n)$ is a magic threshold point for the routing table size. If the routing table size is asymptotically smaller or equal to $O(log_2 n)$, "congestion-free" constraint prevents the algorithm from achieving the smaller network diameter when the "congestion-free" constraint is not imposed. When the routing table size is asymptotically larger than $O(log_2 n)$, however, the "congestion-free" condition no longer plays this "bottleneck" role. This may explain why many existing DHT algorithms [1], [2], [3], [4] stay around this magic threshold.

Our third and final major result is to study the exact (contrary to asymptotic) tradeoff between the routing table size and the network diameter. We first rigorously formulate this tradeoff problem as an optimization problem and explain that finding its solution can be prohibitively expensive in terms of computational complexity for large-size networks. Then we propose a new routing algorithm that reduces the routing table size and the network diameter of Chord [4] both by 21.4% without introducing any other overhead, based on a novel number-theoretical technique.

The rest of the paper is organized as follows. In Section II, we discuss the background and related work. The aforementioned three major results are established in Section III, IV and V, respectively. Section VI concludes the paper.

II. Background and Related Work

In this section, we survey the routing aspects of the existing DHT schemes. Throughout this paper, other aspects will be discussed only when they become relevant to routing. In a P2P system using a DHT scheme, each node is responsible for storing certain parts of the key space. The routing and self-stabilizing (reacting to node joins/leaves) algorithms running on each node collectively implement a hash table-like interface that allows each node to perform lookup, insertion, and deletion of objects.

In DHT schemes, a routing algorithm is characterized by the routing tables employed at each node. Like in Chord [4], we assume that both the name space and the key space of the network are $0, 1, \cdots, n-1$. We let k denote the size of the routing table at each node. At a node of identification id, the routing table basically consists of a set of entries $\{(S_{id,i}, J_{id,i})\}_{1\leq i\leq k}$. The routing algorithm is simply the following: forward a request for key α to node $R(id + J_{id,i})$ if $\alpha - id \in S_{id,i}$. Here $R(\beta)$ is the node currently (subject to changes due to node joins/leaves) responsible for the key β, and the arithmetic is in the cyclic sense (i.e., modulo n). For the correctness of routing, $J_{id,i} \neq J_{id,j}$ and $S_i \bigcap S_j = \emptyset$ when $i \neq j$, and $\bigcup_{1\leq i\leq k} S_{id,i}$ consists of all the keys not handled by the

[1]The term was originally introduced in [6] in a similar but different context.

[2]This is the reason why, in Fig. 1, we deliberately do not put any of the existing DHT schemes on the optimal asymptotic tradeoff curve.

[3]It can be shown that all existing DHT schemes are symmetric.

node id. In symmetric DHT algorithms (defined rigorously later), where $S_{id,i}$ and $J_{id,i}$ are all independent of id, we simply write them as S_i and J_i.

In Chord [4], $n = 2^k$, $S_i = [2^{i-1}, 2^i)$, and $J_i = 2^{i-1}$, where $i = 1, 2, \cdots, k$. The size of the routing table is exactly $log_2 n$, and the network diameter is also $log_2 n$. Algorithms used in [1], [2], and [3] are similar, except that they use different basis (Chord uses 2). In Tapestry [1], for example, $n = d^x$, $S_{i*d+j} = [j * d^i, (j+1) * d^i)$, $J_{i*d+j} = j * d^i$, where $i = 0, 1, \cdots, x-1$ and $j = 1, 2, \cdots, d-1$. Pastry [3] is similar to Tapestry except that d is chosen as a exponential of 2. In both algorithms, the network diameter ($log_d n$) is smaller than Chord's, but the routing table size is larger ($(d-1)log_d n$). However, in terms of asymptotics, these algorithms all maintain a routing table of size $O(log_2 n)$ and achieve a network diameter of $O(log_2 n)$. CAN [5], on the other end, maintains no more than a constant number d of neighbors. In CAN, $S_i = [x^{i-1}, x^i)$ and $J_i = x^{i-1}$, where $x^d = n$. The network diameter is $O(n^{1/d})$.

It is asked in [7] whether ($O(log_2 n)$, $O(log_2 n)$) and (d, $O(n^{1/d})$) are indeed the optimal asymptotic tradeoffs between the routing table size (first coordinates) and the network diameter (second coordinates). We clarify and answer this question in the next two sections. The closest work to ours in the theoretical computer science domain is [6], which studies this tradeoff in a general network. However, they do not address the important issue of congestion. Also they use the storage cost to gauge the routing table size, while we use the self-stabilizing overhead. Both issues make a major difference in the tradeoff results and also the techniques needed to derive such results.

III. Rigorous Characterization of the tradeoff problem

In this section, we first rigorously characterize the metrics involved in the tradeoff. Then we show that $O(log_2 n)$ and $O(n^{1/d})$ are not the asymptotically optimal network diameter values when the routing table size is constrained by $O(log_2 n)$ and d respectively. We show, however, that the schemes which achieve better tradeoffs cause severe congestion to certain network nodes. After we rigorously define the notion of congestion, we conjecture that if "congestion-free" is added as an additional constraint, $O(log_2 n)$ and $O(n^{1/d})$ indeed are the asymptotically optimal network diameter values.

A. *Characterization of the metrics involved in the tradeoff*

In this section, we formally characterize the notion of the routing table size in the DHT context. Recall from Section II that a routing table consists of entries $\{(S_{id,i}, J_{id,i})\}_{1 \le i \le k}$ and we use k to denote the "size" of routing table. In other words, in measuring the routing table size, we count the number of different "next-hops" (neighbors). This is different from the way they are counted in [6] (counting the storage cost of $S_{id,i}$) and IP routers (counting the number of IP prefixes). Counting the number of neighbors make sense in DHT, since there are frequent joins and leaves of nodes, and the cost of maintaining the routing table is directly proportional to the number of neighbors. In other words, the number of neighbors measures the cost to pay for self-stabilizing (adapting to node joins/leaves). The storage cost metric as used in [6] and IP routers, on the other hand, become irrelevant in the DHT context given today's storage price and technology.

Counting the number of neighbors, however, is the correct measure only for stateless routing algorithms. A *stateless routing algorithm* makes a routing decision based only on the destination address (i.e., object key in the request). Therefore, in a stateless routing algorithm, a node does not need to know about node joins/leaves other than those that change some of its "next-hop" values (i.e., identity of the neighbors), since they will not affect its routing decision. All existing DHT schemes are stateless. Contrary to stateless routing is to let the routing decision be based on both source and destination addresses. In such algorithms, a node id may have to react to the join and leave of a node even though it does not affect $id's$ neighboring relationship with other nodes. This certainly would add more complexity to both the routing and the self-stabilizing aspects of the DHT. Whether such "stateful routing" will bring some performance benefit (and hopefully outweigh its overhead) is an interesting topic for future research.

In the rest of the paper, we assume that the network under consideration consists of n nodes, $0, 1, \cdots, n-1$, handling the key spaces $\{0\}$, $\{1\}$, $\cdots$, and $\{n-1\}$, respectively. Clearly, here we implicitly assume that every node in the name space exists and is alive. That assumption, however, sounds a little ironic: if we know that all the nodes exist and are alive, why not send the request for key α to the node α directly? Note however that in this case, the routing table size is actually $O(n)$.

Tradeoff analysis is essentially to study the lower bound of one metric while fixing the other. All lower bound results target worst-case performance. Assuming certain traffic or join/leave patterns, one can design routing algorithms [8] that employ heuristics (e.g., route caching) to enhances average performance. Such heuristics, however, will not be able to improve the performance lower bound *in the worst case*. So our worst-case tradeoff results do not conflict with better tradeoff (average) results achieved using such heuristics.

B. Network diameter lower bounds

It has been asked in [7] whether $O(log_2 n)$ and $O(n^{1/d})$ are the best achievable network diameters when the routing table sizes are $O(log_2 n)$ and d respectively. Our answers to both questions are "no". We show that there exists networks of diameter $O(\frac{log_2 n}{log_2(log_2 n)})$ and $O(log_2 n)$ when the routing table sizes are $O(log_2 n)$ and d respectively.

We formulate the network as a directed graph (V, E). V is the set of all participating DHT nodes and E is the neighbor relationships among them. There exists an edge from node i to node j if node j is one of node $i's$ neighbor. We associate a cost of 1 to each edge so that the network diameter (naturally) becomes the maximum distance between any two nodes. We further require the network to be strongly connected (i.e., every one can reach everyone else). Under this formulation, the questions above become whether $O(log_2 n)$ and $O(n^{1/d})$ are the best achievable network diameters when the out-degree of each node is bounded by $O(log_2 n)$ and d, respectively. The following proposition shows the opposite.

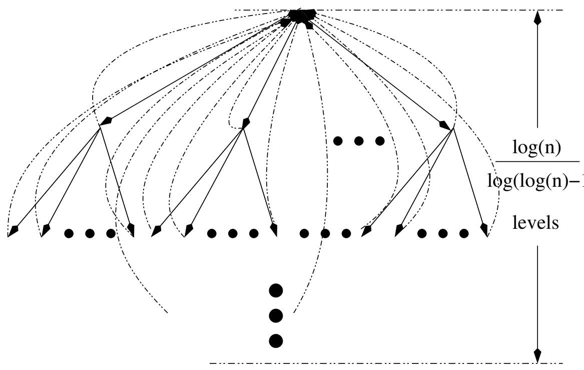

Fig. 2. The constructive proof of Proposition 1

Proposition 1—Upper bounds: There exists a strongly-connected directed graph of diameter $O(\frac{log_2 n}{log_2(log_2 n)})$ in which the out-degree of any node is no more than $log_2 n$. There also exists a strongly-connected directed graph of diameter $O(log_d n)$ in which the out-degree of any node is no more than d.

Proof: We only prove the first part and the second part follows by similar arguments. Fig. 2 shows such a graph that satisfies the aforementioned condition. There is a pseudo "root" in this graph and a directed $(log_2 n$-$1)$-ary tree[4] grows from this "root". This allows the "root" to reach everyone else in at most $log_{(log_2 n-1)} n = \frac{log_2 n}{log_2(log_2 n-1)}$ steps. Also every node other than the root has a directed edge back to the root. This allows every node to reach every other node through the root. So the network diameter is at most $\frac{log_2 n}{log_2(log_2 n-1)} + 1 = O(\frac{log_2 n}{log_2(log_2 n)})$. Note that the maximum out-degree at each node is no more than $log_2 n$. ■

Remark: Note that $O(\frac{log_2 n}{log_2(log_2 n)})$ and $O(log_d n)$ are also the lower bounds, since when each node's out-degree is bounded by x, a node can only reach x^l other nodes using paths no longer than l.

We can see that the routing algorithm used in the network shown in Fig. 2 is essentially centralized: the root has a high in-degree and handles most of the traffic. This is undesirable in P2P networks since the root will become the performance bottleneck and central point of failure. Our initial hypothesis was that if we bound the degree sum (in-degree plus out-degree) at each node to $O(log_2 n)$ and d, the network diameter bounds $O(log_2 n)$ and $O(n^{1/d})$ become optimal. This is unfortunately false, as shown by the following proposition.

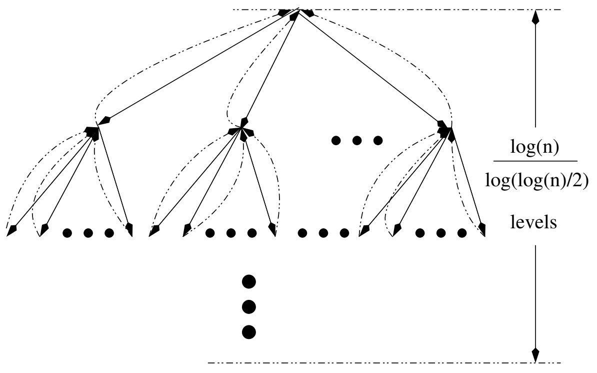

Fig. 3. The constructive proof of Proposition 2

Proposition 2: There exists a strongly-connected directed graph of diameter $O(\frac{log_2 n}{log_2(log_2 n)})$ in which each node's degree sum (in+out) is no more than $O(\frac{log_2 n}{log_2(log_2 n)})$. There exists a strongly-connected directed graph of diameter $O(log_{(d/2)} n)$ in which each node's degree sum is no more than d.

Proof: Again we only prove the first part since the arguments for the second part are similar. Fig. 3 shows such a graph that satisfies the aforementioned condition. There is again a pseudo "root" in this graph and a directed $(\frac{log_2 n}{2})$-ary tree grows from this "root". This allows the

[4]For simplicity of discussion, we omit the use of floors and ceilings when appropriate.

"root" to reach everyone else in at most $log_{(\frac{log_2 n}{2})} log_n = \frac{log_2 n}{log_2(\frac{log_2 n}{2})}$ steps. Also every node other than the root has a directed edge to its parent. This allows every node to reach every other node in $2\frac{log_2 n}{log_2(\frac{log_2 n}{2})} = O(\frac{log_2 n}{log_2(log_2 n)})$ steps (through their lowest common ancestor). Clearly, in this network, no node's degree sum is more than $log_2 n$. ■

Observant readers can see that the network construction in Fig. 3 is still a "cheat": intuitively the root is still the point of congestion. So we come up with the conjecture that if we impose an additional "congestion-free" constraint, the aforementioned diameter lower bounds ($O(log_2 n)$ and $O(n^{1/d})$ may actually be optimal. In the next section, we rigorously define the notion of congestion and introduce the our main conjecture.

C. The notion of congestion and our main conjecture

In this section we precisely define the notion of "congestion-free" and use that to formulate our conjecture. Note that it makes sense to talk about congestion only when a communication load is specified. We artificially impose a uniform all-to-all communication between any pairs of nodes. In other words, for any pair of nodes i, j, $i \neq j$, we impose a unit of traffic from i to j. In the P2P network context, this means that node i is looking for an object that node j is responsible for. Altogether a load of $n(n-1)$ units is imposed on the network. With this artificial load imposed, we define the notion of congestion-free as follows.

Definition 1: We say that a network is c-congestion-free ($c > 1$) if the amount of traffic going through or arriving at each node is no more than c times of the average.

This definition needs to be carefully explained. Suppose the average path length from a random node i to another random node j is l. We have the following proposition stating that the average load on a node is $(n-1)l$ and its proof is essentially a Little's Law [9] argument. The above definition basically says that no node should have more than $c(n-1)l$ load. Note that all existing DHT schemes are 1-congestion-free when all nodes in the identification space exist and are alive, due to Lemma 1 in the next section.

Proposition 3: The average amount of traffic going through a node is $(n-1)l$.

Proof: We write down all $n(n-1)$ sequences of node identifications corresponding to the communications paths between all pairs of nodes. Each occurrence of a node in a sequence in which it is not the source node constitute a unit load to that node. The total number of such occurrences for all n nodes are $ln(n-1)$, since the average path length is l and the first node (source) in each sequence should not be counted. So the average load on each node is $(n-1)l$. ■

We are now ready to formulate the main conjecture of this paper:

Conjecture 1: When the network is required to be c-congestion-free for some constant $c \geq 1$ and uses stateless routing (defined before), $\Omega(log_2 n)$ and $\Omega(n^{1/d})$ are the asymptotic lower bounds for the network diameter when the routing table sizes are no more than $O(log_2 n)$ and d, respectively.

This mathematical "clean" conjecture turns out to be extremely difficult to prove or disprove (even to my math colleagues working in the field of graph theory). We pose it here as an interesting and challenging open problem that merits further research. In the next section, we show that this conjecture is true for a class of routing algorithms known as *symmetric*. The proof leads to our discovery of $O(log_2 n)$ as a magic threshold point for the routing table size.

IV. Asymptotic Tradeoffs for Symmetric Algorithms

In this section, we first prove a slightly weaker result than the conjecture posed in the last section. We show that when the routing algorithms are weakly symmetric (defined below), $\Omega(log_2 n)$ and $\Omega(n^{1/d})$ are indeed the diameter lower bounds for any network with routing table size $O(log_2 n)$ and d, respectively. Then we show that $O(log_2 n)$ is a magic threshold point for the routing table size. If the routing table size is asymptotically smaller or equal to $O(log_2 n)$, "congestion-free" constraint prevents the algorithm from achieving the smaller network diameter when the "congestion-free" constraint is not imposed. When the routing table size is asymptotically larger than $O(log_2 n)$, however, the "congestion-free" condition no longer plays this "bottleneck" role. This may explain why many existing DHT algorithms [1], [2], [3], [4] stay around this magic threshold.

We again assume that the name space is $0, 1, \cdots, n-1$ and all the nodes in the name space exist and are alive. We recall from Section II that the routing table at node id consisting of entries $\{(S_{id,i}, J_{id,i})\}_{1 \leq i \leq k}$. At node id, a request for key α is forwarded to node $id + J_{id,i}$ (equal to $R(id + J_{id,i})$ under our "all-there all-alive" assumption) if $\alpha - id \in S_{id,i}$. Note that all the arithmetic is in the cyclic sense (i.e., modulo n). The following defines the notion of weak and strong symmetry.

Definition 2: A routing algorithm is said to be weakly symmetric, if for any pair of nodes id and id', $J_{id,i} =$

$J_{id',i}$ for all $1 \leq i \leq k$. A routing algorithm is strongly symmetric if it is weakly symmetric, and for any pair of nodes id and id', $S_{id,i} = S_{id',i}$ for all $1 \leq i \leq k$.

For the correctness of routing, $J_{id,i} \neq J_{id,j}$ and $S_i \bigcap S_j = \emptyset$ when $i \neq j$, and $\bigcup_{1 \leq i \leq s} S_{id,i}$ consists of all the keys not handled by the node id.

In the following discussion, we will use the notation J_i (instead of $J_{id,i}$) in weakly symmetric algorithms, and S_i (instead of $S_{id,i}$) in strongly symmetric algorithms. Also, we will refer to the set $\{J_i\}_{1 \leq i \leq k}$ as the *jump set* and $J_i's$ as *jump sizes*, since they specify how much a request (packet) will advance ("jump") in the name space from its current node, in the next step on its way to the destination.

We can see from the above definition that weak symmetry only requires the "jump sizes" to be the same in all the nodes. Strong symmetry, in addition, requires all "routing tables" to be homogeneous. Using symmetry arguments, one can show that strong symmetry implies congestion-free. Weak symmetry, however, does not carry this implication in general.

Lemma 1: A strongly symmetric algorithm is congestion-free.

Proof: [Sketchy] Straightforward through symmetry arguments. ■

Remark: When all the nodes in the name space exist and are alive, all existing DHT schemes are congestion-free due to their strong symmetry natures.

We are now ready to prove the main theorem of this section, which states that the $\Omega(log_2 n)$ and $\Omega(n^{1/d})$ are indeed the optimal achievable network diameters for symmetric routing algorithms, when the routing table sizes are no more than $O(log_2 n)$ and d, respectively. Note in the following theorem that we only require weak symmetry, which does not imply congestion-free in general. Therefore, the result is not in the strict sense weaker than conjecture 1.

Theorem 1: Let k be the number of neighbors each node maintains. For a network that has name space $0, 1, \cdots, n-1$ and uses a weakly symmetric routing algorithm, the followings are true:

- (a) The diameter lower bound for the network is $\Omega(log_2 n)$, if $k \leq \lfloor \frac{1}{2} log_2 n \rfloor$.
- (b) The diameter lower bound for the network is $\Omega(n^{1/d})$, if $k \leq d$. Here assume $d > 2$.

Proof: Let $\{J_i\}_{1 \leq i \leq k}$ be the set of jump sizes, which are the same for all nodes due to the weak symmetric assumption. Suppose the network diameter is l. We pick an arbitrary node id and consider all paths from node id to all other nodes. There are n such paths (including the empty path to itself). We define a function $f : \mathcal{P} \rightarrow (\mathcal{Z} \bigcup \{0\})^{k+1}$, where $\mathcal{P}$ is the set of such paths and $\mathcal{Z} \bigcup \{0\}$ is the set of non-negative integers, as follows. For any path $p \in \mathcal{P}$, we denote as $a_{p,i}$ the number of jumps of sizes x_i used in the path, $i = 1, 2, \cdots, k$. We know that $\sum_{i=1}^{k} a_{p,i} \leq l$ since l is the network diameter. We define $a_{p,0} = l - \sum_{i=1}^{k} a_{p,i}$, and clearly $a_{p,0} \geq 0$. Then we define

$$f(p) := (a_{p,0}, a_{p,1}, \cdots, a_{p,k})$$

We claim that f is injective (one-to-one). We prove this claim by contradiction. Suppose that there are two paths $p, q \in \mathcal{P}$, such that $a_{p,i} = a_{q,i}$, $i = 0, 1, \cdots, k$. Then clearly $\sum_{i=1}^{k} a_{p,i} * x_i = \sum_{i=1}^{k} a_{q,i} * x_i$. So starting from the node id, both paths necessarily end up at the same destination. This contradicts our definition of $\mathcal{P}$ as the set of paths used to reach different destinations.

The size of the range, which is the number of vectors $(a_0, a_1, a_2, \cdots, a_k)$ that satisfy the equation $a_0 + a_1 + \cdots + a_k = l$ and $a_i \geq 0, i = 0, 1, 2, ..., k$. We know from elementary combinatorics that this number is equal to the number of different ways to put l indistinguishable balls into $k+1$ different bins, which is equal to $\binom{l+k}{k}$. Since f is injective, the size of the range should be no smaller than the size of domain, which is n. Therefore, $\binom{l+k}{k} \geq n$. Now we are ready to prove both (a) and (b)

- (a) It suffices to show that $l \geq \frac{1}{2} \lfloor log_2 n \rfloor$. First, we show that $l \geq k$. We prove by contradiction and suppose $l < k$. Note that $\binom{l+k}{k}$ is an increasing function of l. So $\binom{l+k}{k} < \binom{k+k}{k}$. However, given any $\epsilon > 0$, by Stirling's formula ($x! \approx \sqrt{2\pi x}(\frac{x}{e})^x$), $\binom{k+k}{k} \leq (1+\epsilon) * 2^{2k} \frac{1}{\sqrt{\pi k}} < 2^{2k} \leq n$ for large enough n and k. This contradicts our prior assumption that $\binom{l+k}{k} \geq n$. Therefore $l \geq k$. We proceed to show $l \geq \frac{1}{2} \lfloor log_2 n \rfloor$. We again argue by contradiction. Suppose $l < \frac{1}{2} \lfloor log_2 n \rfloor$. Note that $\binom{l+k}{k} \leq \binom{l+l}{l}$ (easy to verify through combinatorial argument), since $l \geq k$. However, when $l < \frac{1}{2} \lfloor log_2 n \rfloor$, we have $\binom{l+l}{l} < n$ due to the same argument above. Therefore $\binom{l+k}{k} \leq \binom{l+l}{l} < n$, a contradiction.
- (b) We need to show that l has to be $\Omega(n^{1/d})$. Since $(l+d)^d > \binom{l+d}{d} > n$, we have $l + d > log_d n$ and therefore $l > n^{1/d} - d$, which is $\Omega(n^{1/d})$.

■

In fact, using the same argument ($\binom{l+k}{k} \geq n$) we can prove the following result, which is more general than Theorem 1(a).

Proposition 4: Let k be the number of neighbors (jump sizes). For a network that has name space $0, 1, \cdots, n-1$

and uses a weakly symmetric routing algorithm, its diameter lower bound is $\Omega(log_2 n)$ when $k = O(log_2 n)$.

Proof: [Sketchy] Assume that $k \leq c * log_2 n$ and apply Stirling formula to the LHS of $\binom{l+k}{k} \geq n$. ■

Although the arguments used in the proof of Theorem 1 give us accurate asymptotic bounds, it may not offer an accurate estimate on the constant factor. For example, in the proof of Theorem 1(a), we essentially show that the diameter lower bound is approximately $\frac{1}{2}log_2 n$ when k is approximately $\frac{1}{2}log_2 n$. However, we have not been able to design a new scheme that achieves the ($\frac{1}{2}log_2 n$, $\frac{1}{2}log_2 n$) tradeoff[5]. This is because in our estimation of the range size in the proof, some elements in the range may not be the image of any paths. In other words, there may exist two vectors $(a'_1, a'_2, ..., a'_k) \neq (a'_1, a'_2, ..., a'_k)$ in the range such that $\sum_{i=1}^{k} a_i J_i = \sum_{i=1}^{k} a'_i J_i$. The (unique) path in $\mathcal{P}$ of length $\sum_{i=1}^{k} a_i J_i$ will map to at most one of them, and the other one will not be the image of any path. Therefore, it can be interesting to further sharpen the estimate on the constant factor through perhaps more sophisticated combinatorial arguments.

We can also see from the proof of Theorem 1 that $k = O(log_2 n)$ is a magic asymptotic threshold. When k is a constant, $\binom{l+k}{k}$ is approximately l^k. However, when k becomes $\frac{1}{2}log_2 n$, $\binom{l+k}{k}$ is approximately 2^{2k}. It is also a magic threshold in the following sense. Recall from Proposition 1 and its remark that for a general network (without assuming symmetry) the diameter of a network is at least $O(log_k n)$ through simple reachability arguments. Theorem 1 shows that this ideal lower bound is superseded by the need to achieve congestion-free routing, when the number of neighbors k is no larger than $O(log_2 n)$. In other words, below the $O(log_2 n)$ threshold, congestion factor dominates the reachability factor. However, we can show that when the number of neighbors k is asymptotically larger than $O(log_2 n)$, we can indeed achieve the bound dictated by the reachability argument. In other words, the congestion no longer plays a "bottleneck" role. This is shown in the following proposition.

Proposition 5: There exists a 1-congestion-free network of diameter $\frac{1}{\alpha}$ $(0 < \alpha < 1)$ in which the number of neighbors at each node is bounded by $O(n^\alpha)$.

Proof: We construct the network to be strongly symmetric so that it is automatically 1-congestion-free due to Lemma 1. We let $n = x^d$ for simplicity of discussion (to avoid getting into floors and ceilings). In our construction, the jump set at each node is $S = \bigcup_{i=1}^{d} S_i$, where $S_i = \{1x^{i-1}, 2x^{i-1}, \cdots, (x-1)x^{i-1}\}$. The routing algorithm is essentially a "greedy" one: given a request for a key α that arrives at node id, id will forward it to $id + j$, where $j = \max\{s | s \in S, s \leq \alpha - id\}$. Clearly, this algorithm is strongly symmetric. Now we show why the network diameter is no more than d. Suppose that a node sends a request to another node that is δ $(0 \leq \delta \leq n-1)$ larger (in the cyclic sense) in the name space. Since $n = x^d$, we can write δ as an $x-ary$ number of at most d digits $a_{d-1}a_{d-2}...a_0$, where $\delta = \sum_{i=0}^{d-1} a_i x^i$. Since $a_i x^i \in S_i \subseteq S$, the "greedy" routing algorithm will route this message in at most d jumps: $a_{d-1}x^{d-1}$, $a_{d-2}x^{d-2}$, $\cdots$, and $a_0 x^0$. ■

Remark: Note that the reachability argument in this case will give use the diameter lower bound $log_{(n^\alpha)} n = 1/\alpha$, which is equal to the bound established above.

In this section, we show that when the routing algorithms are weakly symmetric, $O(log_2 n)$ and $O(n^{1/d})$ are indeed the diameter lower bounds for any network with routing table size $O(log_2 n)$ and d, respectively. This shows that existing DHT schemes (all strongly symmetric) indeed achieves the optimal asymptotic tradeoffs. We also show that $O(log_2 n)$ is a magic asymptotic threshold for the routing table size, which separates the tradeoff region dominated by congestion and the region dominated by reachability.

V. On the Exact Optimal Tradeoffs

The previous sections show that as symmetric algorithms, all existing DHT schemes achieve the optimal asymptotic tradeoffs. However, it is not clear whether they have achieved the optimal tradeoff down to the constant factor. In particular, we would like to know whether the $(log_2 n, log_2 n)$ tradeoff in Chord [4] is optimal. In this section, we formulate this tradeoff problem as an optimization problem: finding the minimum network diameter while fixing the number of neighbors k in a network of size n. However, we are not able to find a closed-form solution for or an efficient algorithm to compute the problem, even though such a solution obviously exists for each (n, k) pair. Nevertheless, we construct an algorithm that achieves $(0.786 log_2 n, 0.786 log_2 n)$ tradeoff using a novel number-theoretical technique. In other words, it is 21.4% smaller in diameter than Chord and uses 21.4% less neighbors ("fingers"). This result is interesting in three aspects:

1) Since the number of neighbors is directly proportional to the self-stabilizing overhead, any sizable reduction is desirable. Moreover, we pay nothing

[5]We did however achieve $(0.7864 log_2 n$, $0.7864 log_2 n)$ tradeoff in Section V.

(and even get paid!) for this reduction: the network diameter is also reduced and there is no other overhead.

2) Our result shows that Chord's tradeoff is not optimal down to the constant factor. This opens the door for further optimization.
3) The number-theoretical technique used is thought-provoking and may lead to the discovery of a general framework to optimize such tradeoffs.

A. Formulation of the problem

An optimal tradeoff problem can be viewed as an optimization problem: optimizing one metric while fixing the other. In this section, we formulate the tradeoff between the routing table size and the network diameter as the following optimization problem. We assume that the network consists of n nodes $0, 1, 2, ..., n-1$ and the routing table is weakly symmetric[6]. We assume that the jump set consists of k jumps $1 \leq J_1 < J_2 < ... < J_k \leq n-1$. The problem is to find a best jump sequence $\{J_i\}_{1 \leq i \leq k}$ that minimizes the network diameter. Let $P_\delta(J_1, J_2, ..., J_k) = \{(a_1, a_2, \cdots, a_k) : \sum_{i=1}^{k} a_i J_i = \delta \ (mod\ n), a_i \geq 0\}$. Then the network diameter $h(J_1, J_2, \cdots, J_n)$ as a function of $\{J_i\}_{1 \leq i \leq k}$ is equal to

$$\max_{1 \leq \delta \leq n-1} \min_{(a_1, a_2, ..., a_k) \in P_\delta(J_1, J_2, \cdots, J_k)} \sum_{i=1}^{k} a_i$$

This is because $\min_{a_i \in P_\delta(J_1, J_2, \cdots, J_k)} \sum_{i=1}^{k} a_i$ is the minimum cost to reach a node that is larger than the source node by δ in the name space. Therefore, we would like to find an algorithm that, given k, computes the following:

$$\underset{1 \leq J_1 < J_2 < ... < J_k \leq n-1}{argmin} [h(J_1, J_2, \cdots, J_k)]$$

Unfortunately, we are not able to find a closed-form solution to this optimization problem. Also, for large n, k, we so far are not able to find an efficient algorithm (brute-force search takes n^k steps) that computes the optimal jump set and the network diameter. Nevertheless, using a novel number-theoretical technique, we are able to construct a routing algorithm that achieves better tradeoffs than Chord.

B. Our new "number system"

We have designed a novel symmetric routing scheme that is able to achievable a network diameter of $0.786 log_2 n$ when the number of neighbors of each node are no more than $0.786 log_2 n$. In other words, it maintains 21.4% less neighbors than Chord [4] for the same network size, and achieves 21.4% less worst-case network delay. The construction of the scheme is based on the following novel number-theoretical technique.

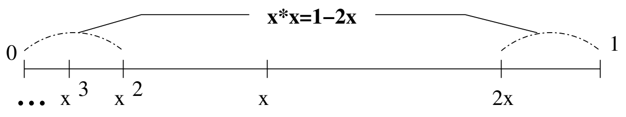

Fig. 4. Our "number system" in a normalized name space

To explain the intuition behind the scheme, we normalize the name space into a unit interval $[0, 1)$, shown in Fig. 4. In other words, the network nodes in this system are 0, $1/n$, $2/n$,..., $(n-1)/n$. The jump set used in Chord can be viewed as 1/2, 1/4, 1/8, 1/16,..., $1/n$ in the normalized name space. In our scheme, we let $x = \sqrt{2} - 1 \approx 0.414$ and the jump set consists of x, x^2, $\cdots$, x^k, where $x^k \approx 1/n$ (i.e., $k \approx log_{(1/x)} n$). Note that x is the root of the equation $1 - 2x = x^2$, as shown in Fig. 4.

Essentially, the goal here is to approximate every real number in $[0, 1)$ using these jump sizes, when allowing a "remainder" smaller than $1/n$. Given a number $y \in [0, 1)$ to approximate, there are three cases at the very beginning:

- (a) If $y \in [0, x)$ then do nothing for this step.
- (b) If $y \in [x, 2x)$, we subtract x from it (a "jump" of size x in the normalized name space) and the "remainder" $y - x$ is in $[0, x)$.
- (c) If $y \in [2x, 1)$, we subtract x from it for two times, and the "remainder" $y - 2x$ is in $[0, x^2)$.

The above procedure will be repeatedly executed in a recursive fashion. In other words, such approximation steps (like (a)–(c)) will be performed in smaller and smaller intervals $[0, \lceil x^i \rceil)$, i=0,1,2,...,k, until the remainder is in $[0, 1/n)$. The intuition of steps (a)–(c) is the following. If a number y belongs to case (a), it is already "better-off" in terms of path length (so we do nothing in the current step). This is because, if y belongs to case (b) or case (c), 1 or 2 *additional* jumps of size x are needed to reduce the remainder to the case (a). Since case (c) requires one more jump (hop) than case (b), we compensate this difference by allowing its remainder to jump to the region $[0, x^2)$ (since $1 - 2x = x^2$) instead of $[0, x)$ as in case (b). In this way, we "equalize" the cost to approximate numbers in regions $[x, 2x)$ and $[2x, 1)$. Note that such equalization is done in a recursive way, spreading its "equalization" benefit recursively.

[6]Note that a weakly symmetric algorithm can by stateful.

C. *Our new routing scheme*

Now we go back to the original (not normalized) name space 0,1,2, $\cdots$, $n-1$. In our routing scheme, the routing table consists of the following jump sizes: $\lceil xn \rceil$, $\lceil x^2 n \rceil$, ..., $\lceil x^{k-1} n \rceil = 2$, $\lceil x^k n \rceil = 1$. So the number of neighbors in this network is $k \approx log_{(1/x)} n \approx 0.786 log_2 n$, which is 21.4% less than in Chord [4]. The routing protocol is essentially the same as in Chord. When a request destined for node id' reaches node id, the current node (id) will forward it to $id + \lceil x^i n \rceil$ where $\lceil x^i n \rceil \le id' - id < \lceil x^{i+1} n \rceil$. The maintenance of the neighbors in the face of node joins/leaves (i.e., self-stabilizing) is also similar to that is used in Chord. In other words, we only change the jump sizes in the routing table and leave all other mechanisms intact. We can also see that our routing algorithm is strongly symmetric. So by Lemma 1, it is 1-congestion-free. Compared to Chord, it reduces the network diameter by 21.4%, shown in the following Theorem.

Theorem 2: Under the routing algorithm shown above, the network diameter is no more than $\lceil log_{(1/x)} n \rceil + 1 \approx 0.786 log_2 n$.

Proof: [Sketchy] It suffices to prove the following invariant: if the difference y between the destination node and the current node in the name space is in $[0, \lceil nx^i \rceil)$, then either (a) after no more than one jump, the remainder falls into the region $[0, \lceil nx^{i+1} \rceil)$, or (b) after two jumps, the remainder falls into the region $[0, \lceil nx^{i+2} \rceil)$. In other words, each jump is rewarded by an additional exponent on x, and after at most $\lceil log_{(1/x)} n \rceil + 1$ jumps we are done (we get down to $[0, 1)$, which can only be 0, as the jumps are in integers). However, this invariant is trivially true since we deliberately constructed the network this way ($x^2 = 1 - 2x$ as shown in Fig. 4). ■

Therefore, our algorithm achieves a $(0.786 log_2 n, 0.786 log_2 n)$ tradeoff, which is better than Chord's tradeoff $(log_2 n, log_2 n)$. This represents a 21.4% reduction on both metrics.

D. *Analysis of the average path length*

There is one (minor) loser in this picture, however, which is the average path length, averaged over all pair-wise communications. In this section, we show that our scheme increases the average path length by about 22.7%, compared to Chord. Nevertheless, the proposed routing scheme is still a bargain, since the scheme reduces both network diameter and the routing table size by 21.4%. Also, as we explain before, given a stochastic model of node joins/leaves, heuristics such as route caching may be used to enhance the (average) performance significantly.

In the following, we show the calculation of the increase in the average path length. Due to the recursive nature of our algorithm, the increase in the average path length can be exactly calculated: no need for simulation. Its derivation exhibits the beauty of recursion.

Let $h(\delta)$ be the exact path length that is needed to reach a node which is δ larger than the source node in the name space (in the cyclic sense). Then the average path length for the name space of size n, denoted as $f(n)$, is equal to $(\sum_{\delta=0}^{n-1} h(\delta))/n$. Note that the average path length in Chord is exactly $\frac{1}{2} log_2 n$. Therefore, our goal is to find out $\lim_{n\to\infty} \frac{f(n)}{\frac{1}{2} log_2 n}$, which is how much worse our scheme did compared to Chord. This is shown in the next Theorem.

Theorem 3: $\lim_{n\to\infty} \frac{f(n)}{\frac{1}{2} log_2 n} = 2(2x+1) c_1 log_{(1/x)} 2 \approx 1.227$, where $c_1 = \frac{\sqrt{2}+1}{4\sqrt{2}}$ and $x = \sqrt{2} - 1$.

Proof: [Sketchy] For simplicity of discussion,, we would like to avoid "floors" and "ceilings" involved in manipulating the function f, which is defined only on the integer domain. We instead work on the (approximate) extension of function f to g, which is defined on the real domain. $g(n)$ is defined as follows. We let $l(t)$ be the "hop counts" (path length) needed to represent a real number t, using the jump set nx, nx^2, nx^3, ... (these are real numbers). We define g(n) as $\frac{1}{n} \int_0^n l(t)(dt)$. It can be shown (through complicated floor and ceiling operations) that $f(n) \approx g(n)$.

We define $g'(y) := y * g(y)$ (i.e., g' is the total while g is the average). It is much easier to work with $g'(y)$. We obtain the following recurrence relations due to the recursive nature of the routing algorithm:

$$\begin{aligned} g'(n) &= 2g'(xn) + g'(x^2 n) + xn + 2x^2 n \\ g'(xn) &= 2g'(x^2 n) + g'(x^3 n) + x^2 n + 2x^3 n \\ g'(x^2 n) &= 2g'(x^3 n) + g'(x^4 n) + x^3 n + 2x^4 n \\ &\ldots\ldots \end{aligned}$$

We evaluate $g(n) = \frac{1}{n} g'(n)$ based on the recurrence relations above. We obtain

$$g(n) = \sum_{j=1}^{k-1} (a_j x^j + 2 a_j x^{j+1}) + o(log_2 n)$$

where $\{a_i\}_{1 \le i \le k}$ is in turn generated by the following recurrence relation:

$$a_{i+1} = 2a_i + a_{i-1}, i = 2, 3, \cdots, k-1$$

The initial conditions are $a_1 = 1$ and $a_2 = 2$. Solving this recurrence relation, we obtain

$$a_i = c_1 r_1^i + c_2 r_2^i, \;\; i = 1, 2, \cdots, k-1$$

where $c_1 = \frac{\sqrt{2}+1}{4\sqrt{2}}$, $c_2 = \frac{\sqrt{2}-1}{4\sqrt{2}}$, $r_1 = 1 + \sqrt{2}$, and $r_2 = 1 - \sqrt{2}$.

Note that $|r_2| < 1$ and $|r_1| > 1$, so $r_2^i \to 0$ when $i \to \infty$. So $a_i \approx c_1 r_1^i$. Also, note that $r_1 x = (1+\sqrt{2})(\sqrt{2}-1) = 1$. So we have

$$\lim_{n\to\infty}(a_j x^j + 2a_j x^{j+1}) = \lim_{n\to\infty}(2x+1)c_1 x^j r_1^j = (2x+1)c_1$$

Therefore

$$\begin{aligned} \lim_{n\to\infty} \frac{f(n)}{\frac{1}{2}log_2 n} &= \lim_{n\to\infty} \frac{g(n)}{\frac{1}{2}log_2 n} \\ &= \lim_{n\to\infty} \frac{\sum_{j=1}^{k-1}(a_j x^j + 2a_j x^{j+1})}{\frac{1}{2}log_2 n} \\ &= \lim_{n\to\infty} \frac{\sum_{j=1}^{k-1}(a_j x^j + 2a_j x^{j+1})}{k-1} \frac{k-1}{\frac{1}{2}log_2 n} \\ &= 2(2x+1)c_1 log_{1/x} 2 \\ &\approx 1.227 \end{aligned}$$

■

VI. Conclusions

In this paper, we study the fundamental tradeoffs (both asymptotic and exact) between the size of the routing table and the network diameter. We rigorously formulate this tradeoff problem and show that there exists algorithms which achieve better tradeoffs than existing DHT schemes. However, all of these algorithms cause intolerable levels of congestion on certain network nodes. After formulating the notion of "congestion-free", we conjecture that the tradeoffs achieved by existing DHT schemes, namely, $(O(log_2 n),\ O(log_2 n))$ and $(d,\ n^{1/d})$, are indeed asymptotically optimal if the network is required to be "congestion-free". We then prove that, for symmetric algorithms, these two tradeoffs are indeed asymptotically optimal. Furthermore, we find that $O(log_2 n)$ is a magic threshold on the routing table size, which separates the tradeoff region dominated by congestion and the region dominated by reachability. Finally, we formulate the tradeoff in the exact sense as a challenging optimization problem. We construct a new routing scheme based on a novel number-theoretical technique, which maintains 21.4% less neighbors than Chord and has a diameter 21.4% less than Chord.

References

[1] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep., U.C. Berkeley Tech. Report UCB/CSD-01-1141, 2001.

[2] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proc. of ACM Symposium on Parallel Algorithms and Architectures*, 1997.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001, pp. 329–350.

[4] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of ACM SIGCOMM'01*, San Diego, CA, 2001, pp. 149–160.

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

[6] D. Peleg and E. Upfal, "A trade-off between space and efficiency for routing tables," *Journal of the ACM*, vol. 36, no. 3, pp. 510–530, July 1989.

[7] S. Ratnasamy, S. Shenker, and I. Stoica, "Routing algorithms for dhts: Some open questions," in *Proc. of Ist Workshop on Peer-to-Peer Systems (IPTPS'01)*, 2001.

[8] G. Pandurangan, P. Raghavan, and E. Upfal, "Building low-diameter p2p networks," in *Proc. of IEEE FOCS*, 2001.

[9] L. Kleinrock, *Queueing Systems*, vol. I and II, J. Wiley and Sons, 1975.