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

Jun Xu, *Member, IEEE,* Abhishek Kumar, and Xingxing Yu

***Abstract*— In this work, we study a fundamental tradeoff issue in designing distributed hash table (DHT) in peer-to-peer networks: the size of the routing table v.s. the network diameter. It was observed by 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 straightforward 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 rigorously define the notion of "congestion" and conjecture that the above tradeoffs are asymptotically optimal for a congestion-free network. We show that the answer to this conjecture is negative in the strict sense. However, the answer becomes positive if the routing algorithm is required to eliminate congestion in a "natural" way by being *uniform*. Our second major result is to prove that the aforementioned tradeoffs are asymptotically optimal for *uniform* algorithms. Furthermore, for uniform algorithms, we find that the routing table size of $O(\log_2 n)$ is a magic threshold point that separates two different "state-efficiency" regions. Our third result is to study the exact (instead of asymptotic) optimal tradeoffs for uniform algorithms. 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 protocol overhead, based on a novel number-theoretical technique. Our fourth and final result is to present Ulysses, a congestion-free *non-uniform* algorithm that achieves a better asymptotic "state-efficiency" tradeoff than existing schemes in the probabilistic sense, even under dynamic node joins/leaves.**

***Index Terms*— Modeling, Communication system routing, Routing, System analysis and design.**

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 from being scalable: Napster uses centralized directory service and 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 the hash table data structure, in the face of node joins/leaves, to all participating P2P nodes. In DHT schemes, each node stores objects 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".

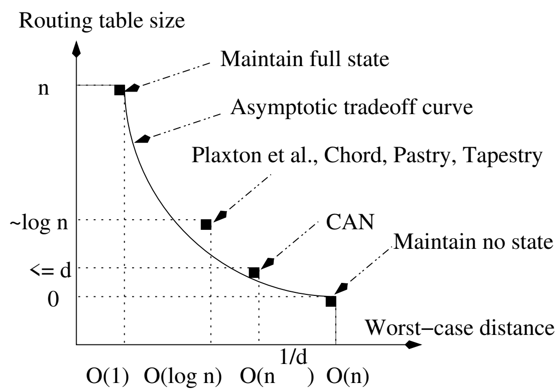

Fig. 1. Asymptotic tradeoff curve between the routing table size and the network diameter

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 end points on the tradeoff curve[1] 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,

[1]Note that the curve is symbolic in the sense that the coordinates are in asymptotics rather than in exact values.

The manuscript was received on 11/15/2002 and revised on 5/30/2003. Xu and Kumar are supported in part by NSF Grant ITR/SY 0113933. Yu was supported in part by NSF grant DMS-9970527 and NSA grant MDA 904-031-0052. A preliminary version of the paper was presented at IEEE Infocom 2003, held in San Francisco, California, USA, from April 1st to 3rd, 2003.

Jun Xu is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: jx@cc.gatech.edu).

Abhishek Kumar is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: akumar@cc.gatech.edu).

Xingxing Yu is with the School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: yu@math.gatech.edu).

and the $O(n)$ diameter incurs intolerable network delay. Such a tradeoff has been referred to as the "state-efficiency" tradeoff[2] 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], Tapestry [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. Since the answer to the above questions in the strict sense is negative, as we will show later, the existing algorithms [1], [2], [3], [4], [5] are not placed on the optimal tradeoff curve in Fig. 1.

The first major result of this paper is to clarify the 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[3]. However, these algorithms all cause severe congestion on certain network nodes even when the load is assumed to be uniform. Based on this observation, we define the notion of "congestion". We initially conjecture that if the network is required to be "congestion-free", the aforementioned tradeoffs (a) and (b) are asymptotically optimal. However, the answer to this conjecture is negative in the strict sense. As pointed out by Karp [8], the butterfly network[4] can achieve better bounds than both (a) and (b). However, if the algorithms are required to be congestion-free "in a natural way" by being *uniform* (defined later), the answer to this conjecture becomes positive. In studying this conjecture, we will thoroughly clarify the role that "congestion-free" plays in this "state-efficiency" tradeoff.

The second major result of this paper is that, if the routing algorithms are *uniform*, we prove that the aforementioned tradeoffs (a) and (b) are indeed optimal. This result is practically important since almost all existing DHT schemes [4], [1], [2], [3], [5] are uniform. Furthermore, we show that $\Omega(\log_2 n)$ is a magic threshold point for the routing table size. If the routing table size is asymptotically smaller or equal to $\Omega(\log_2 n)$, then for any algorithm, "congestion-free" constraint prevents it from achieving the smaller network diameter. When the routing table size is asymptotically larger than $\Omega(\log_2 n)$, 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 major result is to study the exact (contrary to asymptotic) tradeoff between the routing table size and the network diameter. We first 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 protocol overhead, based on a novel number-theoretical technique.

Our fourth and final result is Ulysses, a congestion-free DHT scheme that achieves better "state-efficiency" tradeoffs than both (a) and (b) mentioned above, by giving up the *uniformity* constraint. In particular, with an average routing table size of $O(\log_2 n)$, Ulysses can reach the diameter of $O(\frac{\log_2 n}{\log_2 \log_2 n})$, as compared to $O(\log_2 n)$ in existing schemes [1], [2], [3], [4] with the same asymptotic routing table size. Our design is based on the *butterfly network*, suggested to us by Karp [8] as a counterexample to the aforementioned conjecture. However, three challenges need to be addressed in order to design a low-diameter DHT based on the butterfly network: (a) the butterfly still has *edge congestion*, (b) a sparse network needs to be "mapped" to the "fully-meshed" static butterfly, and (c) a self-stabilization scheme is needed to handle dynamic node joins/leaves, without degrading the size of the routing table. We will discuss how to address these challenges in Sec. VI.

The rest of the paper is organized as follows. In Section II, we discuss the background and related work. The aforementioned four major results are established in Sections III, IV, V, and VI, respectively. Section VII 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-stabilization (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 $\{(S_{id,i}, J_{id,i})\}_{1\le i\le k}$. 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). To draw an analogy between this routing table and the forwarding table in an IP router, we can view $S_{id,i}$ as IP prefixes and $J_{id,i}$ as "next hops" at node id. For the correctness of routing, $J_{id,i} \neq J_{id,j}$ and $S_{id,i} \bigcap S_{id,j} = \emptyset$ when $i \neq j$, and $\bigcup_{1\le i\le k} S_{id,i}$ consists of all the keys not handled by the node id. A analogy to IP routing will be that all IP addresses should be covered by the union of IP prefixes in the forwarding table. In *uniform* DHT algorithms (defined rigorously later in Definition 2), 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

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

[3]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.

[4]Introduced first in parallel computing.

[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 ($O(\log_2 n)$, $\Omega(\log_2 n)$) and (d, $\Omega(n^{1/d})$) are 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 four sections. The closest work to ours in the theoretical computer science domain is [6], which studies "state-efficiency" 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-stabilization overhead. Both issues make a major difference in the tradeoff results and also the techniques needed to derive such results.

Viceroy, also based on butterfly, is proposed in [9] to achieve $O(\log_2 n)$ network diameter with a constant routing table size. The expected diameter of the Viceroy network is about $3\log_2 n$, which is larger than many existing schemes [1], [2], [3], [4]. There is no straightforward way to port the "mapping" (mentioned in Sec. I) and self-stabilization techniques of Viceroy to Ulysses, in which the routing table size is $O(\log_2 n)$.

III. Rigorous Characterization of the tradeoff problem

In this section, we first rigorously characterize the metrics involved in the tradeoff. Then we show that $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ 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 all cause severe congestion to certain network nodes. After we define the notion of congestion, we conjecture that if "congestion-free" is added as an additional constraint, $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ will be 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 Sec. 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 in IP routers (counting the number of IP prefixes). Counting the number of neighbors makes 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 of self-stabilization for adapting to node joins/leaves. The storage cost metric used in [6] and in 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-stabilization aspects of the DHT. Whether such "stateful routing" will bring some performance benefit (e.g., better load balancing) and hopefully outweigh its overhead remains an interesting topic for future research. Throughout this paper, we will only study "stateless" algorithms.

Recall that n denotes the size of the name space. In Sections III and IV 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, we implicitly assume here that every node in the name space exists and is alive[5] (i.e., "everywhere dense"). This assumption is acceptable since we only establish "negative results" in Sections III and IV: no way for an algorithm to achieve a lower diameter than the bound even if it does not need to deal with node joins/leaves. In Sections V and VI where we establish "positive results", however, we will no longer use this assumption and will address the issue of stabilization under joins/leaves.

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 that employ heuristics (e.g., route caching) to enhance 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 (average) tradeoff results achieved using such heuristics.

B. Network diameter lower bounds

It has been asked in [7] whether $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ 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 are 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.

[5]This 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 for a node is actually $O(n)$.

We formulate a DHT scheme as a directed graph (V, E), where V is the set of all participating DHT nodes and E is the neighbor relationships among them. There exists an edge from a node i to a node j if node j is one of node $i's$ neighbor in the DHT. We further require the network to be *strongly connected* (i.e., every one can reach everyone else), which is clearly required of all DHT schemes. Under this formulation, the questions above become whether $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ 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 otherwise.

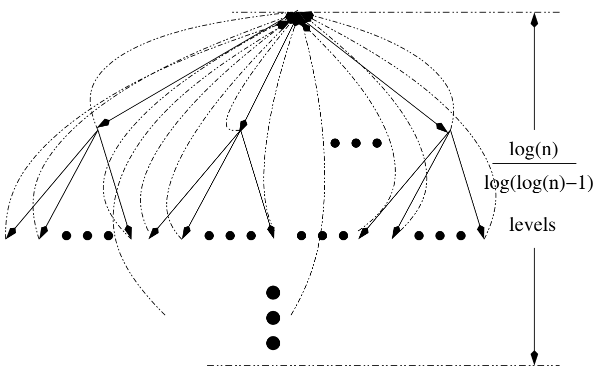

Fig. 2. The constructive proof of Proposition 1

Proposition 1 (Reachable lower 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. The network diameter lower bound is $\Omega(\frac{\log_2 n}{\log_2(\log_2 n)})$ or $\Omega(\log_d n)$ when the routing table size is no more than $\log_2 n$ or d respectively.

Proof: We prove the first assertion first. Fig. 2 shows such a graph that satisfies the aforementioned condition. There is a pseudo "root" in this graph and a directed perfectly-balanced ($\log_2 n$-1)-ary tree[6] 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[7]. 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 $\log_2 n$. The second assertion follows by similar arguments.

As to the third assertion (the lower bound), note that 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. If l is the diameter, then $x^l \geq n-1$ since there are n nodes in the graph. When $x = \log_2 n$, we get $l \geq \frac{\log_2(n-1)}{\log_2 \log_2 n}$; when $x = d$, we get $l \geq \log_d(n-1)$. ■

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

[7]Note that the "pointers" in DHT are unidirectional. In Fig. 2, although the in-degree of the root is $O(n)$, its out-degree, which is also its routing table size, is only $\log_2 n - 1$.

Remark: In later discussion, we refer to the proof of the third assertion (lower bound) as the *reachability argument*.

We can see that the routing algorithm used in the network shown in Fig. 2 is hierarchical: 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 $O(n^{1/d})$ should become optimal. This is unfortunately false, as shown by the following proposition.

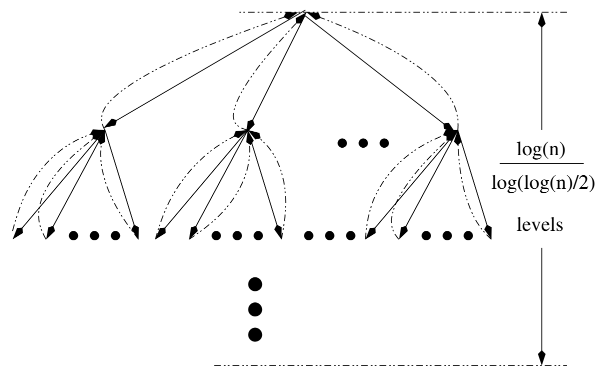

Fig. 3. The constructive proof of Proposition 2

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-degree plus out-degree) is no more than $O(\log_2 n)$. 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: We only prove the first assertion since the arguments for the second assertion 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 perfectly-balanced ($\frac{\log_2 n}{2}$)-ary tree grows from this "root". This allows the "root" to reach everyone else in at most $\log_{(\frac{\log_2 n}{2})} n = \frac{\log_2 n}{\log_2(\frac{\log_2 n}{2})}$ 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 $\log_2 n+2$, which is $O(\log_2 n)$. ■

Observant readers can see that the network construction in Fig. 3 is still a "cheat": intuitively, the root is still the point of congestion. This leads us to the conjecture that if we impose an additional "congestion-free" constraint, the aforementioned diameter lower bounds $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ might actually be optimal. In the next section, we define the notion of congestion and introduce this conjecture.

C. The notion of congestion and our main conjecture

In this section we precisely define the notion of "congestion" 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* load. In other words, for each pair of nodes i, j, $i \neq j$, we impose a unit of traffic from i to j. 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 it is both *c-node-congestion-free* and *c-edge-congestion-free*. A network is said to be *c-node-congestion-free* if no node is handling more than c times the average traffic per node. Likewise, a network is said to be *c-edge-congestion-free* if no edge is handling more than c times the average traffic per edge. When $c = 1$, we simply say that the network is node-congestion-free or edge-congestion-free.

These definitions need 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 [10] argument. This means that, if a network is c-node-congestion-free, no node should route more than $c(n-1)l$ traffic. Likewise, if a network is c-edge-congestion-free, no edge should carry more than $\frac{cn(n-1)l}{|E|}$ traffic, where $|E|$ is the number of edges/links in the network. It can be shown that all existing DHT schemes [1], [2], [3], [4], [5] are congestion-free, when all nodes in the identification space exist and are alive. The node-congestion-free part can be proven from Theorem 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 $n(n-1)l$, 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 $(n-1)l$. ■

Based on the intuition we have obtained from Propositions 1 and 2, we initially have the following conjecture on the role that congestion-free plays in the tradeoff between the network diameter and the routing table size.

Conjecture 1: When the network is required to be c-congestion-free for some constant $c \geq 1$, $\Omega(\log_2 n)$ 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.

Unfortunately, the answer to this conjecture is negative in the strict sense. Karp [8] points out to us that the static *butterfly network*[8] can be configured to reach the diameter of $O(\frac{\log_2 n}{\log_2 \log_2 n})$, when the routing table size is no more than $\log_2 n$. Note that this is exactly the lower bound when the congestion has not become a factor. Although the static butterfly is free of *node congestion*, it has *edge congestion*. Also, there is no straightforward way to apply the static butterfly network to the real P2P networks that are "sparse" (in the name space) and has dynamic node joins/leaves. In Sec. VI, we introduce the static butterfly network and show how to address the above design issues.

Interestingly, as we will show in the next section, the answer to the Conjecture 1 is positive for a class of routing algorithms known as *uniform*. We show that uniform algorithms eliminate node-congestion in a natural way. This result is both theoretically and practically important: all existing DHT[9] algorithms [1], [2], [3], [4], [5] are uniform.

IV. Asymptotic Tradeoffs for Uniform Algorithms

In this section, we show that when the routing algorithms are *weakly uniform* (defined below), $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ are the lower bounds of the diameter for any network with routing table size $O(\log_2 n)$ and d, respectively. This result is practically important since all existing schemes [1], [2], [3], [4], [5] except Viceroy [9] are uniform. In other words, *as uniform algorithms*, these algorithms all have achieved the optimal asymptotic "state-efficiency" tradeoffs. Then we show that, for uniform algorithms, $\Omega(\log_2 n)$ is a magic threshold point for the routing table size. If the routing table size is asymptotically smaller than or equal to $O(\log_2 n)$, "congestion-free" constraint prevents the algorithm from achieving the smaller (optimal) network diameters established through *reachability argument* in Proposition 1. However, when the routing table size is asymptotically larger than $\Omega(\log_2 n)$, 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 Sec. II that the routing table at node id consisting of entries $\{(S_{id,i}, J_{id,i})\}_{1 \leq i \leq k}$. We also recall that $S_{id,i}$ and $J_{id,i}$ are analogous to "i_{th} IP prefix" and corresponding "next hop" in the forwarding table of an IP router respectively. 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-exist 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 concepts of *weak* and *strong uniformity* are defined in the following. For the correctness of routing, $J_{id,i} \neq J_{id,j}$ and $S_{id,i} \bigcap S_{id,j} = \emptyset$ 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.

Definition 2: A routing algorithm is said to be *weakly uniform*, 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 said to be *strongly uniform* if it is weakly uniform, and for any pair of nodes id and id', $S_{id,i} = S_{id',i}$ for all $1 \leq i \leq k$.

Intuitively, the weak uniformity only requires the "jump sizes" to be the same at all nodes. Strong uniformity, in addition, requires all "routing tables" to be homogeneous. All existing algorithms [1], [2], [3], [4], [5], except Viceroy [9], are strongly uniform and therefore node-congestion-free, due to the following Theorem 1.

In the following discussion, we will use the notation J_i (instead of $J_{id,i}$) in weakly uniform algorithms, and S_i (instead

[8] It was originally proposed in the context of parallel computing.

[9] Except for Viceroy [9] which is based on butterfly network.

of $S_{id,i}$) in strongly uniform 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 J_i's specify how much a request (packet) will advance ("jump") in the name space from its current node, during the next step on its way to the destination.

Theorem 1: A strongly uniform algorithm is node-congestion-free, when all nodes in the name space exist and are alive.

Proof: See Appendix. ■

Remark: From this theorem, we can see that in strongly uniform algorithms, the node-congestion is not dependent on the configurations of the routing tables (i.e., $S_i^{\prime}s$) and the jump sets. Edge-congestion, however, will be dependent on both, to be shown in Part E of Sec. V. Note also that in general weak uniformity implies neither node-congestion-free nor edge-congestion-free.

We are now ready to prove two main theorems of this section, which states that the $\Omega(\log_2 n)$ and $\Omega(n^{1/d})$ are indeed the optimal achievable network diameters for uniform routing algorithms, when the routing table sizes are no more than $O(\log_2 n)$ and d, respectively. Note that in the following theorems we only assume weak uniformity, which does not imply congestion-free in general.

Theorem 2: Let k be the number of neighbors each node maintains. Suppose each node in the name space $\{0, 1, ..., n-1\}$ exists and is alive, and the network employs a weakly uniform routing algorithm. The following are true:

(a) The diameter lower bound for the network is $\lfloor \frac{1}{2} \log_2 n \rfloor$, which 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 $k \leq d$, where $d > 2$.

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 uniform 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) and let $\mathcal{P}$ denote the set of those n paths. We define a function $f : \mathcal{P} \rightarrow (\mathcal{N} \bigcup \{0\})^{k+1}$, where $\mathcal{N} \bigcup \{0\}$ 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 size x_i used in the path, for each $1 \leq i \leq k$. 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 $a_{p,0} \geq 0$. Let

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

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 $\Omega(n^{1/d})$.

■

Theorem 2(a) essentially shows 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[10]. In fact, such a tradeoff might not be achievable at all. 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 $(a_1, a_2, ..., a_k) \neq (a_1^{\prime}, a_2^{\prime}, ..., a_k^{\prime})$ in the range such that $\sum_{i=1}^{k} a_i J_i = \sum_{i=1}^{k} a_i^{\prime} J_i$. 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.

Using the intermediate result $\binom{l+k}{k} \geq n$ in the above proof, we can prove the following result, which is stronger and more general than Theorem 2(a).

Theorem 3: Let k be the number of neighbors each node maintains. Suppose each node in the name space $\{0, 1, ..., n-1\}$ exists and is alive, and the network employs a weakly uniform routing algorithm. Then the diameter l is at least $\Omega(\log_2 n)$ when $k = O(\log_2 n)$.

Proof: See Appendix. ■

A. $O(\log_2 n)$ as a Magic Threshold for Routing Table Size

We can see from Theorems 2 and 3 that $k = \Omega(\log_2 n)$ 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

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

the following sense. Recall from Proposition 1 that for a general network (without assuming uniformity) the diameter of a network is at least $\Omega(\log_k n)$ through simple reachability arguments. Theorems 2 and 3 show 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 $\Omega(\log_2 n)$. 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 4: There exists a strongly uniform 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: See Appendix. ■

Remark: Note that the network is automatically node-congestion-free due to Theorem 1. When $n = x^d$, it can be shown that the network is also edge-congestion-free.

With the routing table size in Proposition 4, the reachability argument gives us the diameter lower bound $\log_{(n^\alpha)} n = 1/\alpha$, which is equal to the bound established in Proposition 1. This shows that when the routing table size is asymptotically larger than $\Omega(\log_2 n)$, the congestion no longer becomes a limiting factor.

In this section, we have shown that when the routing algorithms are weakly uniform, $\Omega(\log_2 n)$ 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. This shows that existing DHT schemes, as strongly uniform algorithms, have achieved the optimal asymptotic tradeoffs. We have also shown that $\Omega(\log_2 n)$ 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

We have shown in the previous section that, as uniform algorithms, all existing DHT schemes have achieved 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 or an efficient algorithm for 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"). We also introduced a set of novel mathematical techniques in estimating the increase of the average hop count and the edge congestion in our new routing scheme. This result is interesting in two aspects:

1) Since the number of neighbors is directly proportional to the self-stabilizing overhead, any sizable reduction is desirable. Moreover, There is no extra fault-tolerance overhead for this reduction: the network diameter is also reduced and there is no other protocol overhead. The increase of the average hop count and the edge congestion in our new scheme is moderate.
2) Our result shows that, if the low diameter is the only goal, Chord's tradeoff is not optimal down to the constant factor, among uniform algorithms. This opens the door for further optimization.

We have also introduced a set of novel number theoretical techniques in estimating the worst and average behavior of the scheme. They are 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 uniform[11]. 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

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

This is because $\min_{(a_1, a_2, \ldots, a_k) \in P_\delta(J_1, J_2, \cdots, J_k)} \sum_{i=1}^k a_i$ 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 uniform routing scheme that is able to achieve a network diameter of $0.786 \log_2 n$ when the number of neighbors of each node is 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.

[11] Note that a weakly uniform algorithm can be stateful.

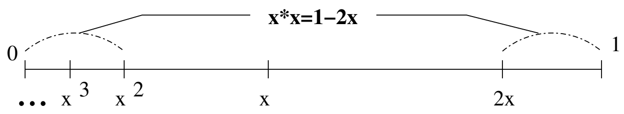

Fig. 4. Our "number system" in a normalized name space

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 in a "greedy" fashion, 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 and "greedy" 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 , ..., k, until the remainder is in $[0, 1/n)$. The intuition of the 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 (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.

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 $\lceil x^i n \rceil \leq id' - id < \lceil x^{i+1} n \rceil$. The maintenance of the neighbors in the face of node joins/leaves (i.e., self-stabilization) 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. It is also easy to see that our routing algorithm is strongly uniform. So by Theorem 1, it is node-congestion-free. Compared to Chord, it reduces the network diameter by 21.4%, shown in the following Theorem.

Theorem 4: 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: It suffices to prove the following invariant: for any $0 \leq i \leq k-1$, 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 of the following is true: (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 at least an additional exponent on x, and after at most $\lceil \log_{(1/x)} n \rceil + 1$ jumps we are done.

Given $y \in [0, \lceil nx^i \rceil)$, we consider three cases. First, if $y \in [0, \lceil nx^{i+1} \rceil)$, then (a) is automatically satisfied and we are done. Otherwise, suppose $\lceil nx^{i+1} \rceil \leq y < 2\lceil nx^{i+1} \rceil$. Then, a jump of size $\lceil nx^{i+1} \rceil$ is made due to "greedy" routing, and the remainder is $y - \lceil nx^{i+1} \rceil < \lceil nx^{i+1} \rceil$, which satisfies (a). Otherwise, $2\lceil nx^{i+1} \rceil \leq y < \lceil nx^i \rceil$. Then the routing algorithm dictates that two jumps, each of size $\lceil nx^{i+1} \rceil$, be made. The remainder is $y - 2\lceil nx^{i+1} \rceil < \lceil nx^i \rceil - 2\lceil nx^{i+1} \rceil \leq \lceil nx^i - 2nx^{i+1} \rceil = \lceil nx^{i+2} \rceil$, which satisfies (b). ∎

However, setting the jump sizes to $\lceil x^i n \rceil$, $i = 1, 2..., k$, clearly makes them dependent on the size of name space n. This is undesirable since the name space may need to grow on demand and we do not want the whole set of jumps be reconfigured as a result of that. So we would like to find a set of "universal" jump sizes that do not change with respect to n and still achieve the equivalent reduction on network diameter. We found such a set that satisfies these requirements, characterized by the following theorem. We omit the proof of the theorem since it is similar to that of Theorem 4.

Theorem 5: When the jump sizes are set to J_i, $i = 1, 2, ..., k$, where $J_1 = 1$, $J_2 = 2$, and $J_{i+2} = 2J_{i+1} + J_i$ for $i > 1$, both k and the network diameter is approximately $0.786 \log_2 n$.

Example: When $n = 1,000,000$, the jump sizes are 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860, 33461, 80782, 195025, and 470832 according to the theorem.

D. Analysis of the average path length

There is one (minor) drawback in this picture, however, which is the average path length, averaged over all pairwise 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 have explained 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 6: $\lim_{n\to\infty} \frac{f(n)}{\frac{1}{2}\log_2 n} = 2(2x+1)c_1 \log_{(1/x)} 2 \approx 1.227$, where $c_1 = \frac{\sqrt{2}+1}{4\sqrt{2}}$ and $x = \sqrt{2}-1$. Here we assume that all nodes in the name space exist and are alive.

Proof: See Appendix. ■

E. How about edge congestion?

Although the new routing scheme is 1-node-congestion-free due to Theorem 1, it is not 1-edge-congestion-free. Instead, it is 1.2336-edge-congestion-free by the following theorem. In other words, in the worst case, certain logical links carries 1.2336 times more traffic than average. However, when logical links are mapped to physical paths, the load on a physical link is typically a superposition of multiple (about $O(\log n)$) logical links of different "jump sizes", which averages each other out during the aggregation with high probability. In other words, with high probability, the physical links are close to 1-congestion-free in the sense of Definition 1.

Theorem 7: Suppose the jump sizes are as specified in Theorem 5. Let $n = 2 * J_k + J_{k-1} - 1$, which represents the worst case for edge congestion[12]. Then the scheme is 1.2336-edge-congestion-free. Here we assume that all nodes in the name space exist and are alive.

Proof: See Appendix. ■

Lemma 1: $\tilde{L}(i-1) = 2x\tilde{L}(i) + x^2\tilde{L}(i+1)$ for $i = k-1, k-2, ..., 1$.

This lemma can be proven by the fact that $x^2 + 2x = 1$ and standard techniques in calculus such as change of variables in integration. We omit its proof here due to lack of space.

VI. Ulysses: A Low Diameter Peer-to-Peer Network

In Sec. III, we conjecture that if the network is required to be congestion-free, $\Omega(\log_2 n)$ will be the network diameter lower bound when the routing table size is no more than $O(\log_2 n)$. In this section, we show that the answer to this question is negative when we do not consider the overhead of maintaining the routing invariant under dynamic node joins and leaves: a static butterfly network achieves a diameter of $O(\frac{\log_2 n}{\log_2\log_2 n})$ and $O(\log_d n)$ when the routing table size is $\log_2 n$ and d respectively. We then provide a succinct description of Ulysses, a *congestion-free* DHT scheme that maintains the routing table size of $O(\log_2 n)$ *with high probability* and always maintains the lower diameter of $O(\frac{\log_2 n}{\log_2\log_2 n})$, in spite of node joins and leaves. Note that the only purpose of introducing Ulysses here is to show that it is possible to construct DHT schemes that have optimal diameter, yet are congestion-free with high probability. This is by no means a full exploration of its design, fault-tolerance, and performance evaluation, which can be found in [11]. Finally, it remains an open question whether or not one can achieve such an optimal tradeoff in a deterministic way (instead of "with high probability"), under dynamic joins and leaves.

[12]We omit the proof to this claim, which is involved and less interesting.

A. The static butterfly

The general static[13] butterfly network can be defined as follows. A (k,r)-butterfly is a *directed* graph with $n = k * r^k$ vertices, where k and r are referred to as the *diameter* and the *degree*, respectively. Note that throughout this section, k no longer denotes the routing table size as before. Each vertex is of the form $(x_0, x_1, \cdots, x_{k-1}; i)$, where $0 \le x_0, x_1, \cdots, x_{k-1} \le r-1$ and $0 \le i \le k-1$. For each vertex $(x_0, x_1, \cdots, x_{k-1}; i)$, we refer to i as its *level*[14] and $(x_0, x_1, ..., x_{k-1})$ as its *row*. From each vertex $(x_0, x_1, \cdots, x_{k-1}; i)$, there is a directed edge to all vertices of the form $(x_0, x_1, \cdots, x_i, y, x_{i+2}, \cdots, x_{k-1}; i+1)$ when $i \ne k-1$, and $(y, x_1, ..., x_{k-1}; 0)$ when $i = k-1$. The routing path from vertex $(x_0, x_1, \cdots, x_{k-1}; i)$ to vertex $(y_0, y_1, \cdots, y_{k-1}; j)$ successively changes x_{i+1} to y_{i+1} while going from level i to level $i+1$, x_{i+2} to y_{i+2} while going from level $i+1$ to level $i+2$, and so on. This process proceeds until all of the x's have been changed to y's, and then continues along row $(y_0, y_1, ..., y_{k-1})$ to level j.

Note that in the static butterfly, the size of the routing table is r since each node $(x_0, x_1, \cdots, x_{k-1}; i)$ is connected to all nodes that have the same coordinates as the node in all dimensions except for the $(i+1)_{th}$. The diameter is $2k-1$ since a query may, in the worst case, change all coordinates (there are k of them) to the right value and then travel another $k-1$ steps to go to the right level. Since $n = kr^k$, depending on the routing table size r, we have two cases: (1) when $r = \log_2 n$, we have $k \le \frac{\log_2 n}{\log_2\log_2 n}$, and (2) when $r = d$, we have $k \le \log_d n$. In other words, if we do not consider node joins/leaves, we can achieve $O(\frac{\log_2 n}{\log_2\log_2 n})$ and $O(\log_d n)$ network diameter when the routing table size is $\log_2 n$ and d respectively. In this paper, we will only explore the first case in depth.

As pointed out by Karp [8], the static (k,r) butterfly is *node-congestion-free*. However, it is not *edge-congestion free*. Consider the edges going form a node $(x_0, x_1, \cdots, x_{k-1}; i)$ to $(x_0, x_1, \cdots, x_{k-1}; i+1)$. In the static $(\frac{\log n}{\log\log n}, \log n)$ butterfly, each node has exactly one such *horizontal* edge, and the remaining $k-1$ are non-horizontal edges. However, a query traverses $\frac{k-1}{2}$ horizontal links and k non-horizontal links on average. Therefore, a horizontal link carries about $\frac{\log n}{2}$ times as much traffic as a non-horizontal link. In other words, its edge congestion factor is $O(\log_2 n)$.

[13]We use the term static to emphasize that this topology works only under the "all-exist all-alive" condition.

[14]Throughout this paper, it is assumed that additive operations on *level* are modulo k.

B. The Ulysses butterfly

In Ulysses, we first solve the aforementioned edge congestion problem by adding $k-2$ "shortcut" links from each node $(x_1, x_2, \cdots, x_k; i)$ to nodes $(x_1, x_2, \cdots, x_k; j), j \neq i, i+1$. This way, in the aforementioned butterfly routing, once x's have all become y's, only one jump is needed to reach the destination through one of these "short-cut" links. This has the additional benefit of reducing the network diameter from $2k-1$ to $k+1$, which is about $\lceil \frac{\log n}{\log \log n} \rceil$. The increase in the routing table size is moderate: from ($\log_2 n$) to ($\log_2 n + \frac{\log_2 n}{\log_2 \log_2 n} - 2$). For example, when there are $2^{20} \approx 1,000,000$ nodes in the network, this represents an increase from 20 entries to 23 entries in the routing table.

However, for Ulysses to be operational in real-world P2P environment, where there are dynamic node joins and leaves, we need to address two additional challenges. First, we must embed the butterfly structure onto a peer-to-peer network that changes dynamically due to node joins and departures. Second, a *self-stabilization* scheme is needed to handle node joins/leaves in P2P networks[15] without degrading its routing table size. Since the detailed design is a part of our ongoing research [11], due to the lack of space, here we will only provide its succinct description.

The name space of a Ulysses k-butterfly consists of *k level-cuboids*. Each of these level-cuboids is a k-dimensional cuboid and corresponds to one of the k levels. Each DHT node represents a *zone* in the name-space. Let n' be the number of nodes inside a Ulysses DHT. Then the name space is partitioned into n' disjoint zones. Each zone is a subcuboid of a level-cuboid, the level of which is called the level of the zone. Each node that joins the network is randomly assigned a level l. The level-l cuboid will then be re-partitioned to accommodate the new node. For simplicity of discussion, here we assume that there is at least one node at each level[16], despite joins and leaves.

A new node that would like to join the network first randomly generates a search key and sends a query for this key, through a node α that is already in the Ulysses network. To find such an α, the new node can use any of the discovery mechanisms proposed in the literature [12]. This query, routed through the Ulysses network, will eventually reach the node currently responsible for the key. This node then splits its zone of responsibility by two and assigns one half to the new node. The keys that are stored at the original node and should now be handled by the new node will be forwarded to the new node. Due to such "merging" and "splitting" in response to dynamic node leaves and joins , the "volume" of a bigger zone is always 2^i $(i > 0)$ times the size of a smaller zone.

Having specified the nodes (the zone-cuboids), we proceed to define the links between these nodes. A zone-cuboid in the l^{th} level-cuboid has links to all those zone-cuboids in the $(l+1)^{th}$ level-cuboid which have an "overlap" with it in the dimensions $0, 1, \cdots, l, l+2, \cdots, k-1$, but not necessarily in the $(l+1)^{th}$ dimension. The geometric intuition of this linking relationship is the following. Given any zone at level l, we first map it to its "shadow" at "the same location" at level $l+1$. Recall that each cuboid (level or zone) is a k-dimensional object. Then we "slide" the "shadow" (in a wrap-around way) along the direction of the $(l+1)^{th}$ dimension. This zone will have a link to all the level-$(l+1)$ zones that its "shadow" will "pass through" ("touching" does not count). This intuition is captured precisely in Fig. 5. Additionally, each node also has links to all nodes that overlap with its shadow (here without sliding) at each level. These links correspond to the "shortcut" links discussed earlier. In [11], we show that routing can be performed through these links in a similar way as in a static butterfly. The size of the routing table is exactly the out-degree of a node handling a zone. We have also shown in [11] how to update a routing table in response to node joins and leaves.

This topology suggests that the routing table size of a node is approximately proportional to the volume of the zone handled by the node. Ideally, the sizes/volumes of these zones/cuboids are identical, and each cuboid keeps about $\log_2 n$ neighbors. However, due to dynamic joins and leaves, there can be certain zones which are larger. In [11], we show through rigorous mathematical analysis, that even without further optimization, the volume of a cuboid stays within the region $[(1/2 - \epsilon_1)V, (2 + \epsilon_2)V]$ with very high probability. Here V is the average volume of a cuboid, and $\epsilon_1, \epsilon_2 > 0$ are small constants. So *with high probability*, each node maintains a routing table no larger than $O(\log_2 n)$. Also, the network always achieves a low diameter of $O(\frac{\log_2 n}{\log_2 \log_2 n})$.

VII. Conclusions

In this paper, we study the fundamental tradeoffs (both asymptotic and exact) between the routing table size and the network diameter. We rigorously formulate this tradeoff problem and show that there are 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", we conjecture that the tradeoffs achieved by existing DHT schemes are asymptotically optimal if the network is required to be "congestion-free". The exploration of this conjecture ramifies the role that congestion-free plays in the "state-space" tradeoff. We then prove that, as uniform algorithms, the existing DHT schemes are indeed asymptotically optimal. Furthermore, we find that, for uniform algorithms, $O(\log_2 n)$ is a magic threshold on the routing table size that separates the tradeoff region dominated by congestion and the region dominated by reachability. We also formulate the exact (instead of asymptotic) "state-efficiency" tradeoff problem for uniform algorithms. 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. Finally, we present Ulysses, a *non-uniform* algorithm that achieves a better asymptotic "state-efficiency" than existing schemes in the probabilistic sense, under dynamic node joins/leaves.

[15] We assume that all node departures are graceful. This is also an implicit assumption in many existing DHT schemes, including CAN [5]. Mechanisms to handle ungraceful node departures will be a part of our future research.

[16] A straightforward initialization protocol can make this happen.

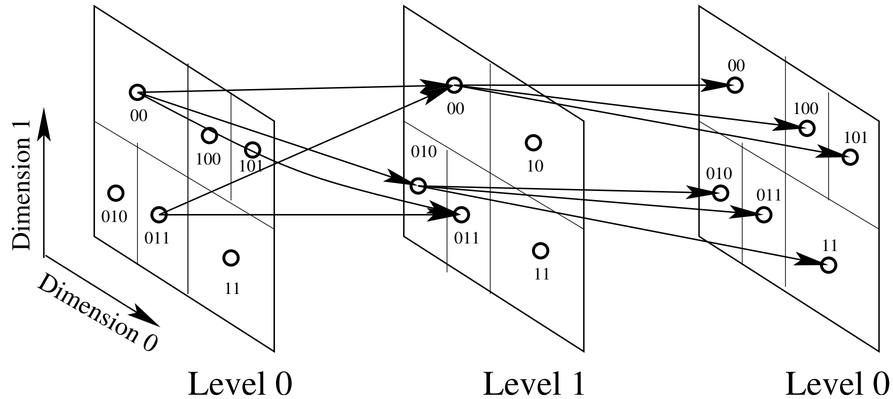

Fig. 5. A Ulysses butterfly with k=2. Links from only 2 nodes in each level are shown for clarity. It is "wrapped around" with level 0 shown twice. If we project the zone of node $\{011, 0\}$ on level 1 and slide its "shadow" along the vertical direction (dimension 1), it overlaps with zones $\{011, 1\}$ and $\{00, 1\}$. This explains why node $\{011, 0\}$ has links to nodes {011,1} and {00,1}. Note that edges from a node at level 0 to nodes at level 1 essentially "slide" in the "vertical" direction and those from level 1 to level 0 "slide" in the "horizontal" direction.

ACKNOWLEDGMENTS

We thank guest editors for coordinating an expeditious review of our submission. We also thank anonymous reviewers for their constructive suggestions that helped improve the quality and readability of this paper. We thank Prof. Richard karp for suggesting the butterfly topology as a possible counterexample to our conjecture.

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*, Newport, Rhode Island, June 1997, pp. 311–320.

[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, pp. 161–172.

[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 1st Workshop on Peer-to-Peer Systems (IPTPS'01)*, Cambridge, MA, USA, 2001, pp. 45–52.

[8] Richard Karp, "A Counter Example," *Private correspondence*, Sept. 2002.

[9] Dahlia Malkhi, Moni Naor, and David Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly," in *Proc. of the 21st ACM PODC*, 2002, pp. 183–192.

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

[11] A. Kumar, S. Merugu, J. Xu, E. W. Zegura, and X. Yu, "Ulysses: A Robust, Low-Diameter, Low-Latency, Peer-to-Peer Network," Tech. Rep., Collge of Computing, Georgia Inst. of Tech., May 2003.

[12] P. Francis, "Yoid: Extending the internet multicast architecture," Unrefereed report, 38 pages, Apr 2000.

[13] K. C. Knowlton, "A fast storage allocator," *Communications of the ACM*, vol. 8, no. 10, pp. 623–625, Oct 1965.

[14] D. E. Knuth, *The Art of Computer Programming*, vol. 1, Addison Wesley Longman Publishing Co., Inc., 3 edition, 1997.

[15] P. W. Purdom, Jr. and S. M. Stigler, "Statistical properties of the buddy system," *Journal of the ACM*, vol. 17, no. 4, pp. 683–697, Oct 1970.

APPENDIX

Proof of Theorem 1

Proof: Let $P(i,j)$ be the routing path from node i to node j. We define a function $I(i,j,t)$ as follows: $I(i,j,t) = 1$ if $P(i,j)$ contains the node t and $i \neq t$. Otherwise, $I(i,j,t) = 0$. Let $T(t)$ be the amount of traffic that goes through the node t, when the uniform all-to-all communication load is imposed. Then for any $0 \leq t' \leq n-1$ and $t' \neq t$,

$$\begin{aligned} T(t) &= \sum_{0 \leq i,j \leq n-1} I(i,j,t) = \sum_{0 \leq i,j \leq n-1} I(i+t-t', j+t-t', t) \\ &= \sum_{0 \leq i,j \leq n-1} I(i,j,t') = T(t') \end{aligned}$$

The first and the last equalities are from the definitions of T and uniform all-to-all communication. The second equality is due to the standard change of variable technique in combinatorial summation. The third equality is from the fact $I(i+t-t', j+t-t', t) = I(i,j,t)$, which follows from Lemma 2 below.

For any $0 \leq t \leq n-1$, since $T(t) = T(t')$ for any $t \neq t'$, the total amount of traffic in the network is $\sum_{0 \leq t \leq n-1} T(t) = nT(t)$ and the average per node is $T(t)$. Therefore, the network is node-congestion-free. ∎

Lemma 2: For any $0 \leq i,j,t,\delta \leq n-1$, $I(i,j,t) = I(i+\delta, j+\delta, t+\delta)$.

Proof: Again, let $P(i,j)$ be the routing path from node i to node j as above. We perform induction on $|P(i,j)|$, the length of $P(i,j)$:

- Initial step: When $|P(i,j)| = 0$, we know that $i = j$. So for any t and δ, $i+\delta = j+\delta$ and $I(i,j,t) = 0 = I(i+\delta, j+\delta, t+\delta)$.
- Induction hypothesis: Suppose $I(i,j,t) = I(i+\delta, j+\delta, t+\delta)$ holds for all (i,j) pairs such that $|P(i,j)| \leq m$.
- Induction step: Let $P(i',j')$ be any path of length $m+1$. We would like to show that for any t and δ, $I(i',j',t) = I(i'+\delta, j'+\delta, t+\delta)$ holds. Since $I(i',j',t)$ is either 1 or 0, let us consider the case $I(i',j',t) = 1$ first. Let v denote the second vertex on the path $P(i',j')$. Then according

to the definition of strong uniformity, $v+\delta$ must be the second vertex on the path $P(i'+\delta, j'+\delta)$. We again consider two cases: (1) if $t=v$, then $t+\delta = v+\delta$ and therefore $I(i'+\delta, j'+\delta, t+\delta)=1$, (2) if $t \neq v$, then $t+\delta \neq v+\delta$ and $I(v,j',t)=1$, and by induction hypothesis (since $P(v,j')$ is of length m) $I(i'+\delta, j'+\delta, t+\delta) = I(v+\delta, j'+\delta, t+\delta) = I(v,j',t)=1$. Similarly, when $I(i',j',t)=0$, we can also show $I(i'+\delta, j'+\delta, t+\delta)=0$. Therefore, $I(i',j',t) = I(i'+\delta, j'+\delta, t+\delta)$ holds for all (i',j') pairs such that $|P(i',j')| = m+1$.

■

Proof of Theorem 3

Proof: From the above proof we know that $\binom{l+k}{k} \geq n$, denoted as (*). In the following, we write l and k as $l(n)$ and $k(n)$ to emphasize the fact that they are functions of n. Since $k(n) = O(\log_2 n)$, there exists $c>0$ and $N_1 > 2$ such that $k(n) \leq c \log_2 n$ when $n > N_1$. We then choose $c_1 > c + 1/2$ and fix it. We need to show that there exists $c_2 > 0$ and $N_2 > 0$ such that $l(n) \geq c_2 \log_2 n$ for all $n > N_2$.

We define $a(x,y) = \frac{(x+y)^{(x+y)}}{x^x y^y}$. Given any $x, y > 0$, it is straightforward to see that, $\lim\limits_{y \to 0^+} a(x,y) = 1$. So given $c_1 > 0$ as above, there exists $c_2 > 0$ such that $a(c_1, 2c_2) < \sqrt{2}$. We let $N_2 = max(\lceil 2^{2/c_1} \rceil, \lceil 2^{2/c_2} \rceil, 4, N_1)$. We claim that when $n > N_2$, $l(n) \geq c_2 \log_2 n$. We prove this by contradiction. Suppose there exists $n' > N_2$ such that $l(n') < c_2 \log_2 n'$. Without loss of generality (WLOG), we choose c_1' such that $c < c_1' < c_1$ and $c_1' \log_2 n'$ is a positive integer (recall that $c_1 > c + 1/2$). We know that $k(n') \leq c \log_2 n' \leq c_1' \log_2 n'$. WLOG, we can also choose c_2' such that $c_2 < c_2' < 2c_2$ and $c_2' \log_2 n'$ is a positive integer. Then $\binom{l(n')+k(n')}{k(n')} = \binom{l(n')+k(n')}{l(n')} \leq \binom{c_1' \log_2 n' + c_2' \log_2 n'}{c_2' \log_2 n'} \leq 2(a(c_1', c_2'))^{\log_2 n'} \leq 2(a(c_1, 2c_2))^{\log_2 n'} < 2\sqrt{n'} < n'$, which contradicts (*) above. The first inequality holds because $\binom{l+k}{l}$ is an increasing function of both l and k. The second inequality is due to Lemma 3 in the following. The third inequality holds since $a(x,y)$ is an increasing function of both x and y when $x, y > 0$. Therefore, $l(n) = \Omega(\log_2 n)$ when $n > N_2$. Note that all the complications in choosing c_1' and c_2' are due to the fact that the formula $\binom{c_1' \log_2 n' + c_2' \log_2 n'}{c_2' \log_2 n'}$ needs to be *defined*.

■

Lemma 3: Let c_1', c_2', and $a(x,y)$ be defined as above. Then $\binom{c_1' \log_2 n' + c_2' \log_2 n'}{c_2' \log_2 n'} \leq 2(a(c_1', c_2'))^{\log_2 n'}$.

Proof: Let $x = c_1' \log_2 n'$ and $y = c_2' \log_2 n'$. We know that $x, y \geq 1$ when $n' > N_2$. Then $\binom{x+y}{y} = \frac{(x+y)!}{x!y!} < \frac{2\sqrt{2\pi(x+y)}(\frac{x+y}{e})^{(x+y)}}{\sqrt{2\pi x}(\frac{x}{e})^x \sqrt{2\pi y}(\frac{y}{e})^y} \leq \frac{2(x+y)^{(x+y)}}{x^x y^y} = 2a(x,y) = 2a(c_1', c_2')^{\log_2 n'}$. The first inequality is by the extended form of the Stirling's formula $\sqrt{2\pi z}(z/e)^z < z! < 2\sqrt{2\pi z}(z/e)^z$ for $z > 2$ and $z \in \mathcal{N}$. Here $\mathcal{N}$ is the set of natural numbers. The second inequality uses the fact $x \geq 1$ and $y \geq 1$. ■

Proof of Proposition 4

Proof: 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 $S = \bigcup\limits_{i=1}^{d} S_i$, 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 uniform. 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 $a_0 x^0$. ■

Proof of Theorem 6

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

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

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

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

where $\{a_i\}_{1 \leq i \leq k}$ 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

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

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

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

■

Proof of Theorem 7

Proof: [Sketch] Let E_i denote the set of edges (links) that are of jump size J_i, i.e., $E_i = \{j \to (j + J_i) | 0 \leq j \leq n-1\}$ where $j \to (j + J_i)$ denotes a link from node j to node $j + J_i$. We claim that given a uniform all-to-all communication load (introduced in Part C of Sec. III), all edges in E_i are of the same load. The proof of this claim is omitted since it is similar to that of Theorem 1. However, the load of an edge in E_i may be different from the load of an edge in E_j when $i \neq j$. Now let L_i be the load of an edge in E_i, $i = 1, 2, ..., k$. It is hard to work with L_i since it involves complicated "floors" and "ceilings" operations. Instead, like in the proof of Theorem 6, we work with its "extrapolation" to the real domain ($\tilde{L}$) as follows.

It is implicitly shown in the proof of Theorem 6 that $J_i \approx c_1(1/x)^i$, where $c_1 = \frac{\sqrt{2}+1}{4\sqrt{2}}$. We define a new set of jump sizes that are of real values: $J_i^! = c_1(1/x)^i$, $i = 1, 2, ..., k$. Clearly $J_i \approx J_i^!$ for $i = 1, 2, ..., k$. The new name space in the real domain, denoted as $\mathcal{S}$, is set to $[0, c_1(1/x)^{k+1})$, where $x = \sqrt{2} - 1$ as in Theorem 6. Similar to our "new number system" discussed in Part B of Sec. V, the "routing problem" in the integer domain can be converted to the problem of "greedily representing" a real number in $\mathcal{S}$ using these real jump sizes $\{J_i^!\}_{1 \leq i \leq k}$. We define $f(y, i) = j$ if y's greedy representation contains J_i for j times. We know that the possible j values are 0, 1, and 2, from the aforementioned properties of the "new number system". We define $\tilde{L}(i) = \int_{y \in \mathcal{S}} f(y, i)d$. We claim without proof that $\tilde{L}(i) \approx L(i)$, $i = 1, 2, ..., k$. In the following we will only work with $\tilde{L}$, the extrapolation of L to the real domain.

From simple calculation, we get $\tilde{L}(k) = 2x^2n + xn \approx 0.7574n$ and $\tilde{L}(k-1) = 2(nx^3 + nx^2) \approx 0.4853n$. Clearly $\tilde{L}(k-1) < \tilde{L}(k)$. We claim that $\tilde{L}(k-1) < \tilde{L}(k-3) < \tilde{L}(k-5) < ... < \tilde{L}(k-4) < \tilde{L}(k-2) < \tilde{L}(k)$. In other words, for any $i = k-1, k-2, ..., 1$, $\tilde{L}(i-1)$ is between $\tilde{L}(i)$ and $\tilde{L}(i+1)$. To show this, we use the recurrence relations $\tilde{L}(i-1) = 2x\tilde{L}(i) + x^2\tilde{L}(i+1)$, $i = k-1, k-2, ..., 1$, from Lemma 1. Since $2x + x^2 = 1$, $\tilde{L}(i-1)$ is a convex combination of $\tilde{L}(i)$ and $\tilde{L}(i+1)$ and must lie between $\tilde{L}(i)$ and $\tilde{L}(i+1)$.

Therefore, we know that $\tilde{L}(k) = max_{1 \leq i \leq k}\tilde{L}(i)$. We already know from Theorem 6 that the average hop count is 0.614 per node. Therefore, the average amount of traffic per node is 0.614 n by Little's Law argument similar to that of Proposition 3. Therefore, the edge congestion is no more than $\tilde{L}(k)/(0.614n)$ = 0.7574/0.614 = 1.2336.

■

PLACE
PHOTO
HERE

Jun Xu (S'98–M'00) is an Assistant Professor in the College of Computing at Georgia Institute of Technology. He received the B.S. degree in computer science from Illinois Institute of Technology in 1995 and a Ph.D. degree in computer and information science from The Ohio State University in 2000. His current research interests include computer and network security, theoretical computer science, discrete algorithms for high-speed networks, and performance modeling and simulation.

PLACE
PHOTO
HERE

Abhishek Kumar received the B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Delhi, India in 2002. He is currently a Ph.D candidate in the college of computing, Georgia Institute of Technology. His research interests include overlay networks, algorithms with applications in networking, and network measurement and performance modeling.

PLACE
PHOTO
HERE

Xingxing Yu is a Professor in the School of Mathematics at Georgia Institute of Technology. He received the M.S. degree in Mathematics from East China Normal University in 1986 and a Ph.D. degree in Mathematics from Vanderbilt University in 1990. His current research interests include graph theory, knot theory, and discrete algorithms.