Emulating Small-World Networks on Content-Addressable Networks

Po-An Chen
Institute of Information Science
Academia Sinica
poanch@iis.sinica.edu.tw

Yih-Kuen Tsay
Department of Information Management
National Taiwan University
tsay@im.ntu.edu.tw

Abstract

Distributed Hash Tables (DHTs) are prevalently used in resolving the routing problem in peer-to-peer networks. Many schemes achieve short routing paths with small state per node, considering dynamic networks, yet performs in a complicated way except CAN. Kleinberg's small-world networks are a family of random graphs allowing short routing paths. We emulated small-world networks on CAN to catch merits from both peer-to-peer routing networks and small-world networks. Our small-world CANs are simple and efficient at handling node arrivals and departures while maintaining short routing paths and ensuring load-balance inherently by clustering. Greedy routing is used first on our small-world CANs to show a competitive performance compared with many other schemes' logarithmic length of average routing paths with logarithmic size of state per node, and then a routing algorithm modified from greed routing is devised to arrive at a better result in reducing the state size by a $O(\sqrt{\log n})$ factor or the cost of node joining and exiting by a $O(\log n)$ factor.

1. Introduction

Since the seminal PRR scheme [12] lit up a way for locating digital objects or routing messages on large-scale networks, many schemes such as Chord [15], Tapestry [18, 2], Pastry [14], and Content-Addressable Networks (CANs) [13, 17] have emerged for the same purpose in the context of peer-to-peer overlay networks. Their approaches can be more generally described by Distributed Hash Tables (DHTs) where the object identifier space is partitioned among all the participant nodes. The tradeoff between short routing path and small state is a basic concern in designing DHTs [16]. When the network is dynamic with frequent arrivals and departures of nodes, the design problem is further complicated by a requirement of achieving small cost for state updates. Fault-tolerance, load balancing and locality are also other important issues. Recent schemes like Simplified PRR [6], Viceroy [7], and Koorde [3] strive for small state ($O(\log n)$ or $O(1)$) while maintaining the same routing cost of $O(\log n)$ hops (with high probability or on average). They may seem successful in the performance measures but they work in a complicated way and fail to keep simple the network construction and maintenance or even routing.

On the other hand, Kleinberg's construction of small-world networks [4, 5] brought up a family of random graphs that allow short routing paths of $O(\log^2 n)$ hops with only one long-range contact per node using greedy routing. The goal of such small-world networks is immediately recognized to coincide with that of the forementioned peer-to-peer routing networks in the sense of ensuring short routing paths. For example, Aspnes et al. [1] extend Kleinberg's result to a more general family of random graphs and derive both lower bounds and upper bounds on greedy routing given different sizes of routing state and probability of failure. Symphony [9] adapts Kleinberg's construction to arrive at a randomized peer-to-peer routing network by placing nodes in a ring and equipping each node with multiple long-range links. A major challenge of adapting small-world networks for a dynamic peer-to-peer setting is how to keep some desirable link properties of the random graphs, since nodes constantly come and go.

We observed that a relatively simple yet efficient and scalable randomized routing scheme for dynamic peer-to-peer networks can be obtained by emulating Kleinberg's small-world networks on an existing DHT implementation. For the overall routing scheme to be as simple as small-world networks, the underlying DHT imple-

mentation has to be simple as well. In fact, a similar idea is applied in Symphony, which used a Chord-like ring as the underlying DHT. However, there are DHT implementations with a more suitable model, other than a ring which Symphony used, to absorb small-world networks so we aims at a more elegant emulation taking arrivals and departures into a good consideration. A natural choice of the underlying DHT implementation is CAN, a scheme that is not only similar to small-world networks in the model of d-dimensional spaces, onto which the physical Internet may be most directly mapped, where $d \geq 1$ but also is simple enough and good at dealing with arrivals and departures of nodes.

Our Contributions. The scheme of a DHT implementation (a peer-to-peer routing network) can be divided into two parts: the formation of the overlay network and a routing algorithm operated on the network. Our emulation of small-world networks on CAN concentrates on the first part. By three different approaches with incremental considerations, we emulate small-world networks on the CAN scheme, and demonstrate greedy routing as an example of routing algorithms on our small-world CAN to achieve average routing paths of $O(\frac{\log^2 n}{\ell})$ hops with ℓ long-range links per node. The point-based naïve approach is simply convenient for dynamic networks. The cluster-based approximation approach is designed for dynamic networks considering load-balance and is based on the node-based approximation with the help of clustering. In the naïve approach, node joining requires $2d + \ell$ nodes to change their states and constructing $2d + \ell$ state, which induces routing of $O(\log^2 n)$ hops on average; node exiting requires $2d + \ell$ nodes to change their states, which induces routing of $O(\log^2 n)$ hops on average. In the approximation approach, node joining requires $2d$ nodes to change their immediate CAN neighbors without routing and constructing $2d + \ell$ state, which induces routing of $O(\log^2 n)$ hops on average; node exiting requires $2d+\ell$ nodes to change their states, which induces routing of $O(\log^2 n)$ hops on average.

Nevertheless, greedy routing is not the only choice so we also explore other possibilities of routing algorithms besides greedy routing and some optimal ones on randomized routing networks (the NoN-GREEDY routing algorithm [10] and the optimal randomized protocol of [8]) facilitated by our emulation of small-world networks on CAN. We simply devise a routing algorithm, which is modified from greedy routing and performs between greedy routing and optimal routing algorithms, to arrive at average routing paths of $O(\frac{\log^2 n}{d\ell})$ hops with $2d + \ell$ state per node on average. With $d = \Theta(\sqrt{\log n})$ and $\ell = \Theta(\sqrt{\log n})$, average routing paths become of $O(\log n)$ hops with $\Theta(\sqrt{\log n})$ state per node, which reduces node joining and exiting cost from $O(\log^2 n)$ to $O(\log n\sqrt{\log n})$; with $d = \Theta(\log n)$ and $\ell = \Theta(1)$, node joining and exiting cost is further reduced to $O(\log n)$ while the same average path length is attained with $\Theta(\log n)$ state per node. So, this result is better than many other schemes' load-balanced result of $O(\log n)$ hops with $\Theta(\log n)$ state per node.

Road Map. In Section 2, we highlight some essential preliminaries about CAN for a basis of following discussion. In Section 3 and Section 4, we present small-world CANs considering different approaches for different requirements, and performances are analyzed for each way of emulation. In Section 5, we search for other routing algorithms feasible in small-world CAN. In Section 6, some other related work mentoined in the itroduction of Section 1 is highlighted for reference. In Section 7, conclusions and future work are given.

2. Preliminaries: CAN

Our scheme is built on the CAN scheme of [13] for its generality, simplicity, and advantages of dealing with nodes joining and exiting. We briefly review it here and leave other essential background knowledge about Kleingerg's construction of small-world networks introduced later when necessary along our discussion.

In the CAN scheme, there is a virtual d-dimensional Cartesian coordinate space on a d-torus as a key space, which is dynamically partitioned among all the nodes in the network such that every node is assigned and thereafter owns its individual, distinct zone. The virtual coordinate space is targeted to store (key,value) pairs where, in the context of object location, a key is an object identifier and a value is an object pointer to a copy of this object, i.e., the hosting node of this object. Thus, a key is deterministically mapped to a destination point P in the virtual space by a uniform hash table and, along with its value, stored at the indexing node that is assigned the zone containing P for later being accessed by any other node using the same hash function to get P. By the routing table consisting of immediate CAN neighbors with adjacent zones, a node sends a message towards the node whose zone contains the destination point by greedy routing: A node forwards the message to one of the neighbors with the zone of coordinate spans closest to the coordinates of the destination point.

There are two optimizing algorithms necessary in the following discussion going to be briefly reviewed here: The more uniform partitioning requires a node under a partition request for a new node's joining to render the partition request induced by the new node to one of its CAN neighbors with the largest zone. The background zone reassignment is periodically executed when a node that temporarily handle zones more than one seeks to hand-off the zone that is not originally its own. A depth-first search in the subtree of the partition tree rooted at the sibling of the zone that is going to be handed-off will find two sibling leaves in the subtree. The merger of these two zones makes one of the two nodes originally responsible for the two zones available for taking over the handed-off zone. Both algorithms prevent small zones from being further partitioned and therefore contribute to a more uniform zone assignment.

Consequently, in the CAN scheme, for a d dimension virtual space shared by n nodes, the number of network nodes can grow without increasing the expected state per node, which is $O(d)$, if d is set independent of n while the expected routing path grows in the rate of O($d \cdot n^{\frac{1}{d}}$) hops. Also, node joining and exiting requires $2d$ nodes to change their state on average without any routing, which is quite efficient.

3. Small-World CAN

We present two approaches in this section to emulate small-world networks on CAN, considering arrivals and departures of nodes, i.e., dynamic networks. A point-based naïve approach is directly a modification from Kleinberg's construction of small-world networks and uses a mechanism adapted from CAN for node arrivals and departures . A node-based approximation approach ignores the need for long-range link changes caused by node arrivals and still results in a small-world competitive network. It is the basis of a cluster-based approximation approach presented in the next section where load-balancing issues are inherently considered using a clustering method. Our small-world CANs can be generalized to an arbitrary dimension. However, we focus on a 2-dimensional space spanned by coordinates x and y for the sake of clarity.

3.1. Point-Based Naïve Approach

In this approach, it is emphasized to distinguish a point, denoted as (i, j), in the virtual space from a node in the network where $i, j \in \{1, 2, .., m^{\frac{1}{2}}\}$. We define the virtual space in integers. $m^{\frac{1}{2}}$ is the number of points along a dimension in the virtual space and the total number, the whole virtual space size, is m. It is required that $m \leq c \cdot n$ for some constant c. This means that the size of object identifiers is bounded by the number of nodes in the network, which is a crucial condition for this approach to attain a performance bound in the analysis later. A node is responsible for all the points in its assigned zone. To make feasible the emulation of a small-world network on CAN, we modify the definition of distance in our scheme from "lattice distance" in Kleinberg's grid model, where nodes are uniformly distributed, to "point distance" in CAN's virtual space. The lattice distance and our point distance are actually the Manhattan distance measured in nodes and measured in points, respectively. Specifically, the *point distance* between two points (i, j) and (k, l) is $d((i, j), (k, l)) = |i - k| + |j - l|$. A node's long-range contacts are decided with the probability in terms of such point distance instead of lattice distance.

We let a node u choose a most central point in its assigned zone $[a, b] \times [c, d]$ as this node's representative point, denoted as $p(u) = (\lfloor \frac{a+b}{2} \rfloor, \lfloor \frac{c+d}{2} \rfloor)$. A node has $2d$ immediate CAN neighbors on average and ℓ long-range contacts, each of which is chosen independently. A long-range link of node u to a point x not in its zone is built with probability proportional to $[d(p(u), x)]^{-2}$. Note that a long-range contact is still a node though it is identified by the point to which this long-range link is pointing. In short, we change the atomic unit in the model from a node in [4] to a point in the virtual space of CAN, and thereby resolves the problem of conformance to the condition of uniform (grid) distribution of atomic units for applying Kleinberg's construction. Therefore, long-range link probabilities can be accordingly defined in terms of point distance.

Node Joining and Exiting. The advantage of this approach is immediately found in handling node joining and exiting. The algorithms for node joining and exiting in CAN can be adapted by taking each node's long-range links into consideration. Because long-range link probabilities are defined using points which are assumed static not like nodes, existing long-range links except those pointing to the points in the splitting zone for node joining or those pointing to the taken-over zone for node exiting can stay unchanged without affecting the link probabilities defined above. Since CAN's algorithms for node joining and exiting need no further discussion, we put more attention to considering long-range links.

Assume a new node v joins at some point that is in node u's zone $[a, b] \times [c, d]$. So, the zone $[\lceil \frac{a+b}{2} \rceil, b] \times$

$[c, d]$ is reassigned to node v if node u partitions its zone along coordinate x; the zone $[a, b] \times [\lceil \frac{c+d}{2} \rceil, d]$ is reassigned to node v if the partition is along coordinate y. CAN's algorithm for node joining can be run for node v, node u, and its old immediate CAN neighbors to learn or update their immediate CAN neighbors. For long-range contacts, the nodes whose long-range links are pointing to the points in the half zone reassigned to node v are notified to change their long-range contacts to node v. Node v choose a long-range contact to point x with probability in proportion to $[d(p(v), x)]^{-2}$ and routes to find the node whose zone contains point x. This process of constructing a long-range contact is repeated ℓ times to get ℓ long-range links independently.

Node exiting results in a taken-over zone. A taken-over zone is a zone that is under control of a node wanting to hand-off its zone or one of its zones. It can appear when the zone of a leaving node can be merged with a neighbor's zone to become a valid zone. Also, it can appear twice during one run of the background zone reassignment mentioned in the preliminaries. Again, CAN's algorithm for exiting node is applied for updating the immediate CAN neighbors of the affected nodes. For long-range contacts, the nodes whose long-range links are pointing to the points in the taken-over zone are notified to change their long-range contacts to a node that takes over such taken-over zone.

Performance Analysis. The performance analysis for our small-world CAN in point-based naïve approach is analogous to the analysis for Kleinberg's construction of small-world networks but in a point-based perspective. We discuss greedy routing in this section as an example of routing algorithms on our small-world networks to arrive at a result between the original CAN and Kleinberg's small-world networks with a grid model and leave other possibilities of routing algorithms in Section 5 to achieve an even better result.

Theorem 1 *Greedy routing on small-world CANs constructed by the point-based naïve approach achieves average routing paths of $O(\frac{\log^2 n}{\ell})$ hops where $\ell \leq \log n$.*

PROOF OF THEOREM 1. By this approach, we get a network where each node u is linked to its 4 immediate CAN neighbors on average and has ℓ long-range contacts. The probability that node u chooses point x not in its zone $[a, b] \times [c, d]$ as one of its long-range contact is

$$\frac{[d(p(u), x)]^{-2}}{\sum_x [d(p(u), x)]^{-2} - \sum_{x \in [a,b] \times [c,d]} [d(p(u), x)]^{-2}}$$

where $\sum_x [d(p(u), x)]^{-2} - \sum_{x \in [a,b] \times [c,d]} [d(p(u), x)]^{-2} \leq \sum_{x \neq p(u)} [d(p(u), x)]^{-2} \leq \sum_{a=1}^{2m^{\frac{1}{2}}-2} (4a)(a^{-2}) \leq 4 \ln(6m^{\frac{1}{2}}) \leq 4 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}})$. The last step is by $m \leq c \cdot n$ for some constant c. Therefore, point x is chosen with probability of at least $[4 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}}) d(p(u), x)^2]^{-1}$.

Greedy routing is considered in phases: For $j > 0$, the routing is said in phase j when the "point distance" from the current point to a target point t is greater than 2^j and at most 2^{j+1}; the routing is said in phase 0 when the point distance to t is at most 2. If the routing is in phase j where $0 \leq j \leq \log m^{\frac{1}{2}}$ and the current node is u, we can calculate the probability that phase j will end at this node, i.e., the probability that the point to which the next hop leads enters the set B_j of points within point distance 2^j from t. There are more than 2^{2j-1} points in B_j, each is within point distance $2^{j+1} + 2^j < 2^{j+2}$ from u so each point in B_j is a long-range contact of u with probability of at least $[4 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}}) 2^{2j+4}]^{-1}$. Thus, if each node has only one long-range link, the probability that the point to which the next hop leads enters the set B_j with probability of at least

$$\frac{2^{2j-1}}{4 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}}) 2^{2j+4}} \geq \frac{1}{128 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}})}$$

and the expected total number of hops needed in phase j will be at most $128 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}})$.

Let X_j denote the total number of hops needed in phase j where $0 \leq j \leq \log m^{\frac{1}{2}}$. If each node only has one long-range link, it has only one trial for the nect hop to enter B_j and since each node has ℓ long-range links instead of one, it has ℓ trials. If $\ell \leq \log n$, $E[X_j]$ is at most one ℓth of the expected total number of hops needed in phase j with one long-range link per node, which we have known. So, we conclude $E[X_j] \leq \frac{128 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}})}{\ell}$. In sum, if let $X = \sum_{j=0}^{\log m^{\frac{1}{2}}} X_j$ be the number of hops of a routing path, then $E[X] \leq (1 + \log m^{\frac{1}{2}}) \cdot \frac{128 \ln(6c^{\frac{1}{2}} \cdot n^{\frac{1}{2}})}{\ell} \leq c' \cdot \frac{\log^2 n}{\ell}$ where c' is a suitable constant. □

Theorem 2 *In small-world CANs constructed by the point-based naïve approach, node joining requires $2d+\ell$ nodes to change their states and constructing $2d + \ell$ state, which induces routing of $O(\log^2 n)$ hops on average; node exiting requires $2d + \ell$ nodes to change their states, which induces routing of $O(\log^2 n)$ hops on average.*

PROOF OF THEOREM 2. The algorithms for node joining and exiting include connections and reconnections of long-range links and CAN's algorithms for node

joining and exiting. CAN's algorithms for node joining and exiting only requires $2d$ nodes to change their immediate CAN neighbors, constructing $2d$ immediate CAN neighbors for the new node on average, and induces no routing; connections and reconnections of long-range links require ℓ nodes to change their states and therefore induce routing of $O(\frac{\log^2 n}{\ell})$ hops for each long-range link, i.e., a total routing of $O(\log^2 n)$ hops. □

The point-based naïve approach creates a simple small-world CAN scheme that performs at least competitively with many other schemes. However, this scheme does not naturally provide load-balance since the load (both routing and indexing storage) is proportional to the area of a node's zone (the number of points in a node's zone) and a random entrance point for a new node's joining does not necessarily make the ratio between the area of the largest and that of the smallest zone be $O(1)$. We still need other load-balancing methods. For example, a new node can first choose $O(\log n)$ points at random and then select to join at the point that splits the largest zone, which is shown to achieve a ratio of $O(1)$ in [11].

3.2. Node-Based Approximation Approach

We try to deal with the problem of emulating small-world networks on CAN from another node-based point of view. The major challenge that stems from a node-based perspective right away is the immediate impact of a new node's joining on long-range link probabilities formed by the existing nodes. In the point-based approach, because points are static, node joining and exiting only affects the existing long-range links pointing to the points in the splitting zone when a node joins and those pointing to the taken-over zone when a node exits. The node-based approach, not like the point-based one, must face and solve the challenge, which may induce a lot of reconnections for maintaining the desirable link probabilities if not being handled carefully. Thus, an approximation may work here as long as the resulting small-world CAN scheme performs well enough. Our idea is ignoring all the reconnections needed to adjust long-range links to conform to the link probabilities in Kleinberg's construction of small-world networks, and showing our small-world CANs by a node-based approximation approach still permits short routing paths.

We basically follow the link probabilities in Kleinberg's construction of small-world networks defined in terms of lattice distance. Yet, for discussing a network's evolving in our approach, we define "node distance" between two nodes as the number of "node steps" separating them and differentiate node distance at different stages of a network's evolution. We identify the network with the number of nodes equal to or greater than 2^i and less than 2^{i+1} as at stage i. The node distance between node u and node v at some current stage i of a network is denoted as $d_i(u, v)$. Then, the node distance between node u and node v at the jth previous stage is d_{i-j}. Hence, $2^{j-1}d_{i-j}(u, v) \leq d_i(u, v) \leq 2^{j+1}d_{i-j}(u, v)$, which will be useful inequalities relating node distances at different stages in our analysis. In this approach, a node has $2d$ immediate CAN neighbors on average and ℓ independent long-range contacts. If node u joins at stage i of the network, a long-range link of node u to a node $v \neq u$ is built with probability proportional to $[d_i(u, v)]^{-2}$.

Node joining and exiting. The advantage of this approach is the ease of handling arrivals of nodes since our approximation strategy simply ignores reconnections of long-range links that are affected by a new node's joining. Because this approach is node-based, not point-based, CAN's algorithms for node joining and exiting can be more conveniently applied when necessary. The algorithm for node exiting is nearly the same as that in the point-based naïve approach so we just skip this part and pay more attention to the algorithm for node joining in this approach.

Assume a new node v joins at a point in the zone of node u. Besides the reassignment of a half of u's zone to v and learning or updating the immediate CAN neighbors for node v, node u, and its old immediate CAN neighbors by CAN's algorithm, all the work left is constructing node v's ℓ long-range links independently. An existing node w is chosen as one of v's long-range contacts with probability in proportion to $[d_i(v, w)]^{-2}$ and v needs routing to find w if the network is currently at stage i, and this constructing process is repeated ℓ times. The probabilities' dependency on the current node distance between any two existing nodes precludes the necessity of considering future nodes' effect impacting on link probabilities between any existing nodes.

Everything seems completed but in fact it does not. It must be noticed how v knows the node distance to some node. This problem does not exist in the point-based naïve approach because v simply can calculate the point distance to some point if the point is chosen. Yet, the problem emerges in a node-based approach and has to be solved for node joining to proceed. We sketch our solution idea borrowed from [8] here: nodes are grouped into clusters to make each cluster spans nearly equal size of the virtual space so the point distance between two nodes in the virtual space can help estimate the number

of nodes between them, i.e, the node distance. Nevertheless, we just assume each node v knows how to get $d_i(v,w)$ for each node w when the network is at stage i in this section, for the sake of our theory developing of this node-based approximation approach, which though may not be actually implemented but will be used as a basis for developing our cluster-based approximation approach that can be implemented. In other words, the node-based approximation approach can be though as a virtual version of the cluster-based approximation approach.

Performance Analysis. Again, we use greedy routing as an example of routing algorithms on our small-world CAN by a node-based approximation approach to show a competitive result just like that in a point-based naïve approach. The analysis for load-balanced small-world CANs by a cluster-based approximation approach in the next section is still relying on the analysis here as a basis (with substitution of clusters for nodes).

Theorem 3 *Greedy routing on small-world CANs constructed by the node-based approximation approach achieves average routing paths of $O(\frac{\log^2 n}{\ell})$ hops where $\ell \leq \log n$.*

PROOF OF THEOREM 3. Recall that greedy routing can be considered in phases. Instead of using the definition of phases in a point-based approach, we define phases in a node-based approach in "node distance at the current stage": For $j > 0$, the routing is in phase j when the node distance from the current node to a target node t is greater than 2^j and at most 2^{j+1}; the routing is in phase 0 when the node distance to t is at most 2.

If the routing is now in phase j where $0 \leq j \leq \log n^{\frac{1}{2}}$ and the current node is u, as in the analysis of Section 3.1, we can still calculate the probability that phase j will end at this node, i.e., the probability that the node to which the next hop leads enters the set B_j of nodes within node distance 2^j from t. We can get the following useful lemma for this purpose.

Lemma 1 *Assume that each node has only one long-range link instead of ℓ. In phase j, if the network is currently at stage l and the current node u joins at the kth previous stage, phase j ends at this node with probability of at least*

$$\frac{c}{2^k \ln(6n^{\frac{1}{2}})}$$

where c is a suitable constant and n is the current number of nodes in the network.

PROOF OF LEMMA 1. Let B'_j be the subset of B_j that node u has probability to choose as long-range contacts when it joins because at the kth previous stage there are only part of B_j existing. For some node v, we have $[d_{l-k}(u,v)]^{-2} \geq 2^{2(k+1)}[d_l(u,v)]^{-2}$ and $[d_{l-k}(u,v)]^{-2} \leq 2^{2(k-1)}[d_l(u,v)]^{-2}$ from the inequalities relating node distances at different stages. By the definition of stages, $2^{k-1}|B'_j| \leq |B_j| \leq 2^{k+1}|B'_j|$ so $|B'_j| \geq \frac{|B_j|}{2^{k+1}}$. At the kth previous stage, node v is chosen as a long-range contact of u with probability of

$$\frac{[d_{l-k}(u,v)]^{-2}}{\sum_{v \neq u}[d_{l-k}(u,v)]^{-2}}$$

where $\sum_{v \neq u}[d_{l-k}(u,v)]^{-2} \leq \sum_{v \neq u} 2^{2(k-1)}[d_l(u,v)]^{-2} \leq 2^{2(k-1)} \sum_{a=1}^{2n^{\frac{1}{2}}-2}(4a)(a^{-2}) \leq 2^{2k}\ln(6n^{\frac{1}{2}})$.

If each node has only one long-range link, the probability that the node to which the next hop leads enters the set B_j is

$$\frac{[d_{l-k}(u,v)]^{-2}}{\sum_{v \neq u}[d_{l-k}(u,v)]^{-2}} \cdot |B'_j| \geq$$

$$\frac{2^{2(k+1)}[d_l(u,v)]^{-2}}{2^{2k}\ln(6n^{\frac{1}{2}})} \cdot \frac{|B_j|}{2^{k+1}} \geq \frac{c}{2^k \ln(6n^{\frac{1}{2}})}$$

where c is a suitable constant. The last step follows from the fact d_l is within 2^{j+2} and $|B_j|$ is at least 2^{2j-1}.

We are ready to use Lemma 1 for our proof. In phase j, the stage at which the current node joins is a random variable with with some probability distribution. By the definition of stages, if the network is currently at stage l, there are at most $2^{l+1} - 2^l$ nodes joining at this stage and 2^{l-k} nodes joining at the kth previous stage where $0 \leq k \leq l$ and $k = 0$ refers to the current stage. The current node joins at the kth previous stage with probability of $\frac{2^{l-k}}{n} > \frac{2^{l-k}}{2^{l+1}} = \frac{1}{2^{k+1}}$. Therefore, if each node has only one long-range link, then phase j ends at this node with probability of at least $\sum_{k=0}^{l} \frac{1}{2^{k+1}} \cdot \frac{c_k}{2^k \ln(6n^{\frac{1}{2}})} \geq \frac{1}{\ln(6n^{\frac{1}{2}})} \sum_{k=0}^{l} \frac{c_k}{2^{2k+1}} \geq \frac{c'}{\ln(6n^{\frac{1}{2}})}$ by Lemma 1 where c' is some suitable constant.

In sum, if let $X = \sum_{j=0}^{n^{\frac{1}{2}}} X_j$ be the number of hops of a routing path where X_j is the number of hops spent in phase j, then $E[X_j] \leq \frac{\log n}{\ell}$ because each node's ℓ long-range links gives it ℓ trials for ending this phase where $\ell \leq \log n$. At last, $E[X] \leq c'' \cdot \frac{\log^2 n}{\ell}$ where c'' is a suitable constant. □

The node joining part of Theorem 4 follows from Theorem 3 for average routing paths of $O(\frac{\log^2}{\ell})$ hops

make constructing ℓ long-range links cost routing of $O(\log^2 n)$ hops. Besides, the algorithm for node exiting is nearly the same as that in the point-based naïve approach so the same result as the node exiting part of Theorem 2 can be attained.

Theorem 4 *In small-world CANs constructed by the node-based approximation approach, node joining requires $2d$ nodes to change their immediate CAN neighbors without routing and constructing $O(2d + \ell)$ state, which induces routing of $O(\log^2 n)$ hops on average; node exiting requires $2d+\ell$ nodes to change their states, which induces routing of $O(\log^2 n)$ hops on average.*

4. Load-balanced Small-World CAN

Our load-balanced small-world CAN is built by the cluster-based approximation approach, which is based on the node-based approximation presented in Section 3.2 with the help of a clustering technique that is adapted from [8] by replacing nodes with clusters. We will first introduce the clustering technique that we need and combine the node-based approximation approach with clustering to get the cluster-based approximation approach. This later developed approach also solves the problem of node distance that was brought up in Section 3.2. It also provides load-balance in an inherent way by clustering.

4.1. Clustering

In CAN, randomly selecting an entrance point in the virtual space for node joining does not necessarily result in a $O(1)$ ratio between the area (or load) of the largest zone and that of the smallest zone, which has been mentioned in Section 3.1. Even the two optimizing algorithms mentioned in Section 2, the more uniform partitioning and the background zone reassignment, which though are useful heuristics, do not give provable results about the ratio. Our idea for giving a bound about routing and indexing load per node is grouping nodes to make each group responsible nearly equal load. LEMMA 3.3 of [8] that is applied in a Chord-like environment can be adapted to fit our need in a CAN environment. The lemma is restated as follows.

LEMMA 3.3 of [8]. Let k be such that $2^k \leq \frac{\epsilon^2 n}{8 \ln n}$. With probability at least $1 - \frac{2}{n}$, the number of points in each of 2^k equi-sized non-overlapping sub-intervals of $[0, 1)$ lies in the range $\frac{(1\pm\epsilon)n}{2^k}$.

The points in this statement is actually we know as nodes so it says that if a ring of $[0, 1)$ is divided into 2^k equal sized non-overlapping sub-intervals where 2^k cannot be not too large, the number of nodes in each sub-onterval is about $\frac{n}{2^k}$. With a little modification, it can be applied on CAN: CAN's virtual space can be set as $[0, 1) \times [0, 1)$ if $d = 2$; 2^k equal sized groups can be done by dividing each dimension into $2^{\frac{k}{d}} = 2^{\frac{k}{2}}$. We can use this lemma to group nodes into 2^k clusters by making the zone in each cluster span the size of $\frac{1}{2^k}$. Clustering would be easy if it could be done following this way yet what has to be noticed is that the network size n is unknown. Besides, a clustering scheme must consider arrivals and departures of nodes. This means we cannot get a fixed n to group nodes into a fixed number of clusters but have to *estimate* n to make clusters grow with n.

We can still adapt the network size estimation scheme and THEOREM 3.1 of [8] and use them to get an accurate estimate of n. The idea is, for a node, measuring the density of node identifiers close to itself to deduce n. By the following scheme, we can know the sufficient number of nearby nodes to look over and the size of interval spanned by these nodes to arrive at an estimate of n accurate enough: Consider some node with identifier x and let n_i denote the number of nodes that share the most significant bits with x; node x identifies the largest ℓ such that $n_\ell \geq 16(1 \pm \delta)\delta^{-2} \ln(2^\ell n_\ell)$ where δ is used in the corresponding theorem, which provides a bound for this estimate of n. The theorem is restated as follows.

THEOREM 3.1 of [8]. With probability at least $1 - \frac{2}{n}$, the estimate of network size made by every node lies in the range $\frac{n}{1\pm\delta}$.

4.2. Cluster-Based Approximation Approach

The problem about how to get the node distance between two nodes in the node-based approach can be solved by transforming the problem to get the "cluster distance" between two clusters, using the clustering method in Section 4.1. The number of the nodes whose zones totally span $\frac{1}{2^k}$ of the whole virtual space is about $\frac{n}{2^k}$ with high probability where k is bounded by a function of n, proportional to $\frac{\epsilon^2 n}{\ln n}$. By the network estimation scheme for n and assuming $\delta < \frac{1}{3}$, we can also get a good estimate of k, i.e., $\tilde{k}$ where $k - 1 \leq \tilde{k} \leq k + 1$. So, in the network, there are at most three $\tilde{k}$ values, which are constantly bounded. Hence, each node can know the zone boundaries of each cluster that has $\Theta(\ln n)$ nodes and thereby calculate the cluster distance to each cluster.

Our cluster-based approximation approach is now obvious. This approach is based on the node-based approx-

imation, i,e., ignoring long-range link changes when a new node comes, but instead long-range links between clusters are built with probabilities defined in terms of cluster distance at an estimate of k that can be calculated in the above way. A node u has average $2d$ immediate CAN neighbors and ℓ long-range constacts chosen independently. If node u joins at an estimate $\tilde{k}$ which has at most a difference of 1 to k, it chooses a cluster y as one of its long-range contact with probability in proportion to $[d_{\tilde{k}}(c_{\tilde{k}}(u), y)]^{-2}$ where $c_{\tilde{k}}$ returns u's cluster at $\tilde{k}$ and $d_{\tilde{k}}$ is the cluster distance between cluster $c_{\tilde{k}}$ and cluster y at $\tilde{k}$. Note that k is simlar to the stage defined in Section 3.2 that represents the evolution degree of the network. A long-range contact is one of the nodes in the identifying cluster to which this long-range link is leading.

Routing. Routing must be reconsidered. It can perform among clusters using long-range links between clusters just like among nodes, but how it should be done when the target cluster is entered. The so-called intra-cluster or local routing toward the target node that should follow. Since each cluster has only $\Theta(\ln n)$ nodes, routing among them is quite easy and is nearly impossible to become a bottleneck for the whole routig process. Many routing algorithms can do this job. Even by following the immediate CAN neighbor links, intra-cluster routing costs only $O(\log n)$ hops. Therefore, as our so-called inter-cluster or global routing achieves average routing paths of $O(\log n)$ hops, we do not have to do anything, like adding links or choosing sophisticate routing algorithms, for intra-cluster routing. We can put our focus back on inter-cluster routing.

Node Joining and Exiting. The algorithms for node joining and exiting in the node-based approximation approach can be run here with some modifications described as follows. Assume a new node v joins at a point within the zone of node u when the network is with $\tilde{k}$. Besides the reassignment of a half of u's zone to v and learning or updating the immediate CAN neighbors for node v, node u, and its old immediate CAN neighbors by CAN's algorithm, the main tast is constructing node v's long-range links independently since we still apply an approximation strategy. With the clustering technque in Section 4.1, if the network currently has an estimate $\tilde{k}$, then an existing cluster y is chosen as one of v's long-range contacts with probability proportional to $[d_{\tilde{k}}(c(v), y)]^{-2}$ and v needs routing to find a node in y. Notice that the long-range contacts, each of which is leading to a cluster, should be uniformly assigned among all the $\ln n$ nodes in this cluster to ensure load-balance. This can simply be done by traverse all the nodes in the cluster, which merely induces routing of $O(\log n)$ hops. Repeating this constructing process ℓ times, node v can have ℓ independent long-range links.

Performace Analysis. We use greedy routing as an example of routing algorithms for the inter-cluster part. The analysis for routing paths and node joining and exiting of our load-balanced small-world CANs by a cluster-based approximation approach is easy to get simply by substituting 2^k for n in $O(\frac{\log^2 n}{\ell})$ of Theorem 3 and $O(\log^2 n)$ of Theorem 4. $2^k \leq \frac{\epsilon^2 n}{8 \ln n}$ so Theorem 5 and Theorem 6 immediately follow.

Theorem 5 *Greedy routing on small-world CANs constructed by the cluster-based approximation approach achieves average routing paths of $O(\frac{\log^2 n}{\ell})$ hops where $\ell \leq \log n$.*

Theorem 6 *In small-world CANs constructed by the cluster-based approximation approach, node joining requires $2d$ nodes to change their immediate CAN neighbors without routing and constructing $O(2d + \ell)$ state, which induces routing of $O(\log^2 n)$ hops on average; node exiting requires $2d+\ell$ nodes to change their states, which induces routing of $O(\log^2 n)$ hops on average.*

The cluster-based approximation approach deals with load-balancing issues inherently by clustering because the clustering technique ensures that the zones of the $\ln n$ nodes in a cluster span totally $\frac{1}{2^k}$ of the whole virtual space. The ratio between the area of the largest summation of the zones in a cluster and the area of the smallest summation of the zones in a cluster should become $O(1)$. Since the routing and indexing load per node is proportional to the area of a node's zone, with the uniform selection of a node in a cluster when a long-range link is constructed for node joining, the ratio between the largest load per node and the smallest load per node is $O(1)$.

5. Other Possibilities of Routing Algorithms

Greedy routing is prevalently used in many schemes such as Chord to achieve short routing paths, i.e., routing paths of $O(\log n)$ hops with $\Theta(\log n)$ state per node (on average or with high probability) and we also use it as an example of routing algorithms on our small-world CANs. However, greedy routing has been shown not optimal on randomized routing networks in the work of

[1, 10]. By optimum, we mean routing paths of $O(\frac{\log n}{\log \ell})$ hops with ℓ links per node (on average or with high probability), i.e., routing paths of $O(\frac{\log n}{\log \log n})$ hops with $\ell = \Theta(\log n)$ state. Greedy routing on our small-world CANs arrives at average routing paths of $O(\frac{\log^2 n}{\ell})$ with ℓ links, which is $O(\log n)$ hops with $\ell = \Theta(\log n)$ state. So, our scheme is at least competitive with many other schemes.

There are other routing algorithms that can be chosen such as NoN-GREEDY of [10] and the optimal randomized protocol of [8]. NoN-GREEDY is a routing algorithm that passes messages to the farest one among the neighbors of a node's neighbors'greedily toward the target and has been shown effective on many randomized routing networks with small-world properties in [10]. If we choose to use NoN-GREEDY on our small-world CANs, it is reasonable to conjecture that average routing paths are of $O(\frac{\log n}{\log \log n})$ hops with $\Theta(\log n)$ state per node because a small-world CAN is a result of emulationing Kleinberg's construction of small-world networks on CAN, and NoN-GREEDY on Kleinberg's small-world networks (called as small-world percolation networks in [10]) has been shown to have such routing paths of $O(\frac{\log n}{\log \log n})$ hops with high probability.

NoN-GREEDY lets a node know the neighbors of its neighbors to get more information without increasing each node's state where neighbors may actually be long-range contacts in the sense of small-world networks. This gives us an idea of using just the long-range contacts of a node's immediate CAN neighbors to devise a routing algorithm though obviously not competitive with optimal routing yet at least better than greedy routing. We only use a node's immediate CAN neighbors' information because communication with immediate CAN is cheap. This routing algorithm is simply modified from greedy routing that we have carefully studied. The first good thing about this modified greedy routing algorithm is its simplicity no matter whether it is run or analyzed. We modify greedy routing as follows: At each node, it not only use its own long-range contacts to route but ask its immediate CAN neighbors about their long-range contacts and thereby takes $d\ell$ long-range links into consideration to pass messages to the link leading to a node nearest to the target.

The analysis for the average routing paths using this routing algorithm is basically like the analysis in Section 3 and Section 4 with substitution of $d\ell$ for ℓ in $O(\frac{\log^2 n}{\ell})$ where $d\ell \leq \log n$. As each immediate CAN neighbor v of node u's is located at nearly at the same place as u, a node w will be chosen as a long-range contact of v with nearly the same probability as the probability that w is chosen as a long-range contact of u. Therefore, average routing paths are of $O(\frac{\log^2 n}{d\ell})$ hops with $2d + \ell$ state per node on average. There are two flexibilities. With $d = \Theta(\sqrt{\log n})$ and $\ell = \Theta(\sqrt{\log n})$, attaining average routing paths of $O(\log n)$ hops, this routing algorithm reduces the state per node to $\Theta(\sqrt{\log n})$ rather than greedy routing's $\Theta(\log n)$; correspondingly, node joining or exiting requires $\Theta(\sqrt{\log n})$ nodes to change states, which induces routing of $O(\log n\sqrt{\log n})$ hops on average. With $d = \Theta(\log n)$ and $\ell = \Theta(1)$, attaining average paths of $O(\log n)$ with $\Theta(\log n)$ state per node, this routing algorithm reduces the number of nodes that need to change states to $\Theta(1)$, therefore reducing routing to $O(\log n)$ hops on average.

6. Related Work

PRR [12], for providing provable locality properties for a restricted class of metric spaces, makes maintaining a node's state a nontrivial task, especially when nodes are constantly joining and exiting from the network. On the other hand, Chord [15] disregards locality for simplicity. Simplified PRR [6] searches for combining desirable advantages in both of these extreme schemes and design a scheme that is simple yet exploits locality still. Besides an identifier ring, by correlating the distance in an auxiliary logical ring with the distance in physical networks, locality can be exploited.

Combining Chord with de Bruijn graphs, Koorde [3] looks up a key by contacting $O(\log n)$ nodes with $O(1)$ state per node. To embed a de Bruijn graph on a sparsely populated identifier ring (of 2^b identifier space), each node m in the network maintains information about two other nodes, namely its successor on the ring and its first de Bruijn node. Koorde also achieves routing paths of $O(\frac{\log n}{\log \log n})$ hops with $O(\log n)$ state per node for fault-tolerance. Koorde can use the node joining and exiting algorithm in Chord so it induces as much cost as Chord does for state updates.

Viceroy [7] uses a composition of an approximate butterfly network and the connected ring of predecessor and successor links to achieve a routing path length of $O(\log n)$ hops with $O(1)$ state per node on average. In Viceroy, each node, located at a level at random among $O(\log n)$ levels, maintains a routing table of five links to upper, lower, and the same levels in addition to its predecessor and successor. Although a node in Viceroy has a constant out-degree and a constant average in-degree, a node can at most have a logarithmic in-degree with high probability. Nodes joining and exiting in Viceroy induce

$O(\log n)$ hops and requires $O(1)$ nodes to change their state with high probability.

7. Conclusions and Future Work

We emulated small-world networks on CAN to obtain a new scheme of peer-to-peer routing network, small-world CANs. For CAN is a suitable basis, our emulation gracefully avoids unnecessary complication. Small-world CANs preserve CAN's simplicity and ease of node arrivals and departures as well as inherit small-world networks' short routing paths. With help of clustering, our scheme can ensure load-balance inherently. We used greedy routing on our small-world CANs to show a cometitive performance compared with many other schemes, and also devised a routing algorithm modified from greedy routing to arrive at a better result.

We are interested in exploring other routing algorithms suitable on small-world CANs, further leveraging inter- and intra- cluster routing, and considering more heterogeneity of nodes in the network to generalize the scheme. Besides, each hop is not weighted in the model but if it is, how the routing stretch should be considered. Also, small-world CANs as a practical lightweight peer-to-peer infrastructure deserves experiments.

References

[1] J. Aspnes, Z. Diamadi, and G. Shah. Fault-tolerant routing in peer-to-peer systems. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'02)*, pages 223–232, 2002.

[2] K. Hildrum, J. D. Kubiatowicz, S. Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'02)*, pages 41–52, August 2002.

[3] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proceedings of the 2nd International Peer To Peer Systems Workshop (IPTPS'03)*, 2003.

[4] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC'00)*, 2000.

[5] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Proceedings of the 15th Annual Conference on Advances in Neural Information Processing Systems (NIPS'01)*, 2001.

[6] Xiaozhou Li and C. G. Plaxton. On name resolution in peer-to-peer networks. In *Proceedings of the 2002 Conference on Principles of Mobile Computing*, pages 82–89, 2002.

[7] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'02)*, 2002.

[8] G. S. Manku. Routing networks for distributed hash tables. In *Proceedings of the 23nd ACM Symposium on Principles of Distributed Computing (PODC'03)*, July 2003.

[9] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, pages 127–140, 2003.

[10] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *Proceedings of the 36nd ACM Symposium on Theory of Computing (STOC'04)*, June 2004.

[11] M. Naor and U. Wieder. Novel architectures for p2p applications: The continuous-discrete approach. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'03)*, June 2003.

[12] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(1/2):241–280, 1999.

[13] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161–172, 2001.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware 2001, LNCS 2218*, pages 329–350, 2001.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, 2001.

[16] Jun Xu, A. Kumar, and Xingxing Yu. On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, April 2003.

[17] Zhichen Xu and Zheng Zhang. Building low-maintenance expressways for p2p systems. Technical Report HPL-2002-41, Internet Systems and Storage Laboratory, HP Laboratories Palo Alto, March 2002.

[18] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Computer Science Division, April 2001.