

Discovering Network Neighborhoods Using Peer-to-Peer Lookups

Li-wei Lehman

Center for Educational Computing Initiatives
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: lilehman@mit.edu

Steven Lerman

Center for Educational Computing Initiatives
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: lerman@mit.edu

Abstract— We present a distributed rendezvous mechanism to solve the network neighborhood discovery problem in large-scale distributed systems. Our scheme uses the existing distributed peer-to-peer lookup mechanisms such as Chord to provide network neighborhood discovery. Our idea is for each host to compute a *location key* that characterizes its network location on the Internet. A node then uses this location key to perform a Chord lookup on its nearby peer node. We focus on the issue of how to generate the location key in a distributed fashion such that nodes that are close to each other in the actual network will have similar location key values, and will therefore be mapped to nearby locations on the Chord ring. Using real network measurements from NLANR and large-scale topologies generated by GT-ITM, we compare the performance of our scheme with a random mapping scheme. Simulation results show that significant performance gains can be achieved by probing only a small number of nodes using our scheme.

I. INTRODUCTION

In many distributed applications, end hosts need to know the network locations of other nearby participating hosts in order to enhance their performance. Example applications include server selection in content distribution networks, proximity routing in a peer-to-peer (P2P) system, and neighbor selection in overlay network constructions. We focus on the network neighborhood discovery problem in large-scale distributed peer-to-peer systems. We use round-trip latency as a measurement of nearness between network hosts. Our goal is for each node to discover other “nearby” participating nodes in a completely decentralized manner, where each node probes only a small subset of other members in the system.

We introduce a rendezvous-based scheme for distributed topology discovery using a peer-to-peer lookup system such as Chord [1]. Given a key, the Chord protocol maps the key onto a node. Our approach is for each host to compute a key which characterizes its network location on the Internet. We call such a key the location key, and the nodes that these location keys are mapped to the Rendezvous Points (RPs). To lookup other nearby participating nodes, a node queries its corresponding rendezvous point using its location key. One issue is how to map Internet network locations to Chord identifier space such that nodes that are close to each other in the actual network will be mapped to similar key values in the Chord identifier space. On one hand, we would like to exploit the natural clustering among nodes so that nodes that are close to each

other are mapped to the same rendezvous point. On the other hand, the location keys should not be clustered too tightly in order to avoid overloading any rendezvous points.

We examine the performance tradeoffs of our rendezvous-based scheme by using the GNP (Global Network Positioning) [2] generated coordinates as a basis to compute the location keys. In GNP, each node measures its network distances to a few landmark nodes to derive its coordinates in a D-dimensional geometric space. We generate a host’s location key by applying a scaling function to its 1-dimensional coordinate in GNP, and use coordinates from a higher dimensional space to refine the searching process for the closest node.

Our location-key based rendezvous mechanism has the following characteristics.

- **Efficiency.** The computation of the location key based on the distances to the landmarks is fast and efficient. Once the location key is computed, the lookup cost required to locate a rendezvous point is same as performing a key look up in Chord, which is $O(\log N)$ [1].
- **Scalability.** Instead of using a fixed set of server nodes, the task of location lookup is distributed among a subset of the participating peer nodes. Since the number of nodes (i.e., RPs) that respond to location queries automatically increases as the number of participating peer nodes increases, our scheme can scale to support a large number of nodes. Our simulation results suggest that the average number of location queries at the rendezvous points remains at a very reasonable load even as the system size increases to thousands of nodes.
- **Fault-tolerance.** Since the location queries are routed to the appropriate RPs using distributed peer-to-peer lookups, our scheme adapts to dynamic membership and network changes well. In contrast to schemes with pre-designated location servers, the RPs in our scheme can leave and join the system dynamically with minimal disruption to the network neighborhood lookup process.

In the next section, we contrast our approach with related works in this area. Next, we present a detailed description of the rendezvous-based network neighborhood discovery scheme. Finally, we present our simulation results and conclude with summary and future work.

II. RELATED WORK

The IDMaps [3] and GNP [2] projects are both architectures for a global distance estimation service. Our work, in contrast, does not focus on the infrastructure, but rather on a distributed peer finding scheme that is built on top of the infrastructure support. Anycast [4], [5] can be used as a basis to solve the peer finding problem by having all peer nodes subscribe to the same anycast service, but requires global deployment of such service. More recently, the Internet Indirection Infrastructure (I3) [10] presents a new Internet rendezvous-based communication architecture using Chord [1]. In contrast to our scheme, I3's focus is on flexible and application-specific routing using the so-called "triggers" as Chord keys.

In [11], Castro *et al.* presented a distributed algorithm to discover a nearby seed node for proximity routing in Pastry [7]. In [12], Karger and Ruhl describe a data structure for nearest neighbor queries using random sampling. In contrast to the above approaches, our scheme does not go through a series of probing in order to converge to the nearby node; the computed location key in our scheme allows the nodes to quickly discover their network neighbors by querying their RPs. Further, our scheme does not rely on all nodes along the request routing path to store proximity information about other nodes in order to locate nearby peer nodes.

Brocade [13], built as a second overlay on top of Tapestry [8], allows nearby nodes to locate each other through "supernodes" that are situated near network access points. In contrast, our scheme does not involve any election process to find a subset of nodes to function as "supernodes"; instead we use the Chord lookup mechanism to locate the RPs based on the location keys. In Tiers [14], peer nodes are organized in a hierarchy in order to allow nodes to locate nearby application peers.

In CAN [9] with the distributed binning scheme [15], proximity routing is supported by having nodes that are close-by in network space map to nearby CAN key space. Similar to our scheme, they assume the existence of a set of landmarks in the Internet. In contrast to our scheme, the location of an end host i in their scheme is characterized by the ordering of landmarks in terms of their distances to i .

III. RENDEZVOUS-BASED TOPOLOGY DISCOVERY USING STRUCTURED P2P SYSTEMS

In this section, we describe a rendezvous-based scheme for distributed topology discovery. There are two main components to our scheme: (1) a mapping scheme that maps nodes to the Chord identifier space such that nodes that are close-by in terms of the actual network latency can discover each other quickly through the Chord lookup mechanism (2) a distance comparison mechanism for RPs to compare distances between nodes.

A. A Location Key Based Mapping Scheme

In this section, we describe how nodes can compute location keys in a distributed fashion, and how nodes with similar

location key values can rendezvous using the Chord mechanism. For simplicity, we assume that all nodes in our peer-to-peer system already formed a Chord lookup structure amongst themselves¹. Each node is assigned a random identifier. Identifiers are ordered in an identifier circle modulo 2^m as in Chord.

As mentioned earlier, we use GNP [2] to compute the coordinates of each host in a geometric space to characterize its network location. Our approach to compute the location key is based on the 1-D coordinate values generated by GNP. More specifically, let k_i be the 1-D coordinate value of node i generated by GNP based on node i 's RTT measurements to P landmarks. We apply a scaling function $f(k)$ to map k to a Chord identifier k' which we call the location key. Each node i then uses its scaled 1-D coordinate value, k' , as a Chord key to look up its rendezvous point. The successor node of key k' on the Chord identifier circle is the Rendezvous Point (RP) of node i . Each rendezvous point maintains information of a "pool" of nodes that are rendezvousing at itself. The "pool size" of a rendezvous point refers to the number of nodes that are rendezvousing at that RP. We assume that each RP periodically exchanges its "pool" list with its immediate L successors and predecessors on the Chord ring. The combined "pool" list is then called the "augmented pool" list.

Since in a distributed environment, the actual range of the 1-D coordinate value is unknown, each node needs to use an estimated coordinate range to map its 1-D coordinate value to the Chord identifier space. To automatically scale a node's 1-D coordinate value to the Chord identifier space, we use the function $f(k)$ as described below. We take advantage of the fact that each node has information about the the 1-D coordinate values of the P landmarks. Let X_l and X_h be the lowest and highest 1-D coordinate values of the P landmarks respectively. Each node with 1-D coordinate value k computes its location key k' as follows:

$$\begin{aligned} S &= X_l - C \\ E &= X_h + C \\ R &= |E - S| \\ M &= 2^m \\ k' &= f(k) = \left(\frac{|k - S|}{R} * M \right) \text{mod} M \end{aligned}$$

where C is a positive constant that is set based on assumptions of node distributions on the 1-D location key space. It can be used to adjust how close the location keys are spaced from each other on the Chord ring.

B. Distance Comparisons for Query Refinement at RPs

As part of the neighborhood discovery process, a node i sends its location information to its RP by inserting a key/value

¹The assumption is not required to guarantee the correctness of our network neighborhood discovery scheme. In fact, the nodes which are querying their respective network neighborhoods can be a completely different and independent set of nodes from the Chord nodes.

pair into the Chord ring, where the key is i 's computed location key, and the value is i 's IP address, and possibly other higher dimensional location information. The higher dimensional location information can be used by the RP to compare the distance between each pair of nodes in its (augmented) pool list. The location information is time-stamped and the RPs periodically refresh its pool list information.

In response to each node i 's query for network neighborhoods, an RP sorts its augmented pool list based on some distance metrics. The RP returns the closest q nodes in the sorted list to node i as an estimate of the closest nodes to node i . To complete the search process, node i then pings each of the q nodes returned by its RP, and declares the one with the closest RTT measurement its closest network neighbor.

To restrict the number of nodes for each node to ping, an effective distance comparison mechanism is needed. We examine three different ways to compare distances between nodes at an RP's pool list.

- Location key. An RP can use the location keys of nodes to compare distances. In this case, distance between two nodes is estimated as the absolute value of the difference between their location key values. The advantage of this scheme is that distance comparison is computationally inexpensive, and no other higher dimensional location information needs to be communicated to the RP.
- RTT (Round Trip Time) to landmarks. Each node can use its round trip latency to the P landmarks as its coordinate values in a P -dimensional space. The Euclidean distance between each node in this P dimensional space will be used by the RP to sort its pool list.
- GNP computed D-dimensional coordinates. Based on the P landmark RTT measurements, GNP can be used to compute coordinates in a D-dimensional space to characterize a node's location on the Internet.

IV. SIMULATION RESULTS

In this section, we evaluate the above location key based rendezvous scheme for discovering nearby nodes. We compare our location key based mapping with a randomized mapping scheme and show that the location key based scheme is much more effective in finding nearby nodes. The randomized mapping scheme works the same way as the coordinate-based scheme, except that each node, instead of using the 1-D coordinate as a key to locate its RP, uses a randomly generated key to locate its RP. We evaluate our scheme using both real network measurements and simulated topologies:

- The Active Measurement Project (AMP) at the National Laboratory for Applied Network Research (NLNR) collects network measurements between over 100 active monitors distributed over the Internet [16]. We use the round trip time (RTT) measurements between 110 of such monitors on July 16, 2002 for our experiments.
- The GT-ITM Internet Topology Generator is used to generate transit stub topologies of different network sizes: network with 100, 600, 1470, 3492 and 8730 nodes were used for simulations.

To make minimum assumptions about the landmark placement, we randomly chose 10 nodes from the N nodes as the landmarks for each experiment. Ten experiments in total were performed for each topology, each with a different random selection of the 10 landmarks.

We use the RTTs between each host and the 10 randomly selected landmarks as input to GNP, and generated two sets of coordinates for each host: coordinates in 5-dimensional and 1-dimensional space. The 1-dimensional coordinate was computed using GNP to minimize the sum of the normalized squared errors. The normalized squared error term is defined as $(\frac{d_{ij}-O_{ij}}{d_{ij}})^2$, where d_{ij} is the actual distance measurement between host i and j , and O_{ij} is the computed distance (Euclidean in our case) between host i and j using the coordinates.

For ease of implementation, we chose the identifier space to be in the range of 0 to $2^{16} - 1$. We examine our rendezvous algorithm in terms of three performance metrics: distance ratio, rendezvous hop distances, and rendezvous point poolsize distribution.

A. Distance Ratio Comparison

The distance ratio R_i of a node i is defined as, $R_i = \frac{RTT_f}{RTT_a}$, where RTT_f is the RTT measured between node i and its closest found node, and RTT_a is the RTT measured between node i and its actual closest network neighbor in the system. In this section, we examine the distance ratios of both the coordinate-based and the random mapping schemes. We declare a node to have found its "closest" neighbor if the distance ratio is less than 1.0001. The results presented here are combined distributions of distance ratios from all 10 experiments (with different landmark selections).

We examine the distance ratios of the coordinates-based mapping scheme when q equals 1, 3 and infinity. Recall that, in response to a node i 's query to its closest node, an RP will return the q closest nodes (sorted based on GNP generated 5-Dimensional coordinates) to i in its augmented pool. When q equals infinity, the RP returns the entire augmented pool list to node i , which in turn pings each node to locate the closest neighbor.

From Figure 1, we note that a node derives most of the benefits by pinging only the three closest nodes (in terms of GNP generated 5-dimensional Euclidean distance) in its RP's augmented pool. It also shows that the location key based scheme significantly outperforms the randomized-mapping scheme. Using the location key based mapping algorithm with $C = 200$, close to 50% of the nodes found their actual closest neighbor, and 70% of the nodes yield distance ratios less than 1.5 by pinging the first three nodes from its RP's sorted augmented pool. The augmented pool list is sorted based on the 5-Dimensional coordinate values generated in GNP. In comparison, using the randomized mapping scheme, less than 8% of the nodes found their actual closest neighbor by pinging all nodes in the RP's pool list.

Figure 2 shows the distance ratio results using the GT-ITM generated topology with 3492 peer nodes with $L = 8$. It shows

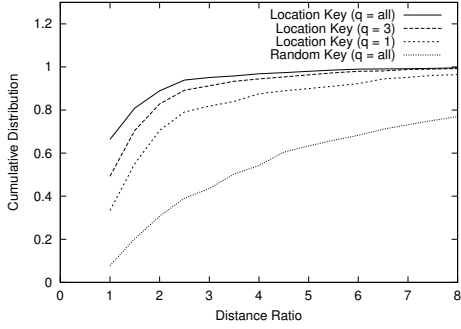


Fig. 1. Distance Ratio Cumulative Distribution: Random vs. location key based mapping. AMP, $N = 110$, $L = 1$.

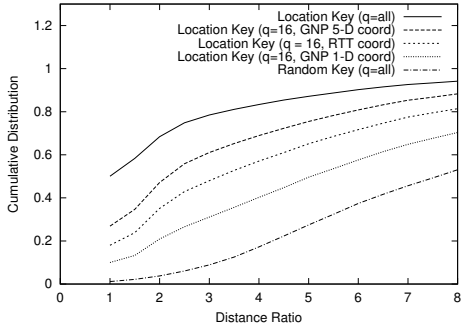


Fig. 2. Distance Ratio Cumulative Distribution: Random vs. location key based mapping. Also compares 3 different distance comparison schemes in sorting an RP's augmented pool list. GT-ITM results, $N = 3492$, $L = 8$.

that when each node pings the entire augmented pool list at its RP, over 50% of the nodes can find its nearest network neighbor, and close to 70% of the nodes can find a nearby network node whose distance is less than a factor of two than the actual closest peer node. The next three curved lines compare the efficiency of three different distance metrics used for sorting an RP's augmented pool list. In all three cases, the nodes are sorted based on their Euclidean distances to the querying node, and the top q ($q = 16$ in this case) nodes in the sorted list are pinged. Our results indicate that using the GNP generated 5-dimensional coordinates perform the best; using a node's RTTs to the P (10 in this case) landmarks comes second; and, as expected, using the 1-dimensional location key to compute the Euclidean distance between nodes performs worst, but still outperforms the random mapping results.

B. Rendezvous Hop Distance Comparison

The second metric measures how far apart two nearby network nodes are mapped on the Chord ring. Ideally, if node j is the closest network neighbor to node i , we would like node i and j to be mapped to the same rendezvous point on the Chord ring. We measure the effectiveness of the mapping scheme as the hop distance between node i 's RP and its nearby network neighbor node j 's RP. For example, if node i and j are mapped to the same RP, the rendezvous hop distance is zero in this case. If node j 's RP is the immediate predecessor

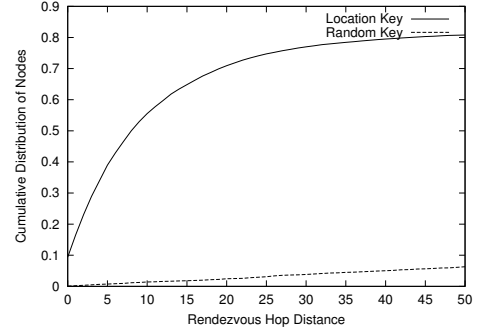


Fig. 3. Rendezvous hop distance: cumulative distribution of nodes which found a nearby neighbor (distance ratio is no larger than 1.01 in this case) with rendezvous hop distance no more than x hops. GT-ITM, $N = 3492$, $C = 500$.

or successor of node i 's RP, then the rendezvous hop distance is one.

Since we assume that each RP exchanges its pool list with its immediate L neighbors on the Chord ring, as long as the rendezvous hop distance between a node i and its network neighbor j is no more than L hops away from each other on the Chord ring, node i can find j by a simple query message to its own RP.

In figure 3, we evaluate our location key based mapping scheme in a network with 3492 nodes. We compare our scheme with the baseline random key scheme in terms of the cumulative distribution of nodes which found a nearby network neighbor with rendezvous hop distance no more than x hops. For each node i , we declare i to have found one of its closest network neighbors once it locates a node with distance ratio no more than 1.01 (i.e., once it finds a node whose latency to i is no more than 1% larger than i 's latency to the actual closest).

Using the location key mapping scheme, close to 10% of the nodes can find one of its closest network neighbors at their RPs in a network with 3492 nodes. More than half of the nodes can find their closest network neighbors in no more than 8 hops from their RPs. In contrast, using the random key mapping scheme, less than 0.06% of the nodes can find a nearby network neighbor at their randomly mapped RPs; and for more than half of the nodes their closest neighbors are mapped to some RPs that are at least 500 hops away from their RPs.

C. Poolsize Distribution

The pool size of an RP represents its querying load. One issue is whether we can maintain a relatively stable querying load at RPs as the number of hosts in the system increases. In this section, we examine the poolsize distribution for network of size 110, 3492, and 8730 respectively. Our results indicate that the mean querying load maintains at a very reasonable number (average about 6 queries per RP) even for large network sizes. Although the distribution of the pool sizes tend to be skewed because nodes are clustered in the actual network

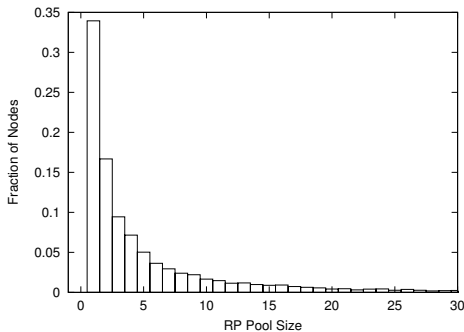


Fig. 4. Location key based mapping poolsize (includes only samples with non-empty poolsize). GT-ITM results, $N = 3492$, $C = 500$.

TABLE I
RP POOL SIZE DISTRIBUTION FOR DIFFERENT NETWORK SIZES.

N	Mean	Std	Avg Worst	Fraction of RPs
110	4.37	5.48	29	0.21
1470	4.19	6.97	66	0.23
3492	5.03	8.27	90.7	0.20
8730	5.43	8.47	114.8	0.18

space, the worst case pool size is still very reasonable for large number of peer nodes.

For each node with a non-empty pool, we take the number of nodes rendezvousing at that node, and plot the overall distributions from the 10 experiments. Figure 4 shows the RP pool size distribution for the location key based mapping with 3492 nodes. Compared to the random key based mapping (plot not shown due to space), the location key based mapping scheme has a more skewed poolsize distribution because the nodes are clustered in the actual network space. About 17% of the nodes function as RPs.

Table I shows the mean and standard deviation of the RP pool size for different system sizes. It also shows the average worst case pool size, which is the average of the largest pool size from ten different experiments. The last column of the table shows the fraction of nodes in the system that function as rendezvous points. In general, the mean and worst case pool size increases as the scaling constant increases; while the fraction of nodes that serve as RPs decreases as the scaling constant increases. The scaling constants used in table I is 200 for all network sizes, except for the 8730 nodes case where $C = 500$.

From Table I, for 8730 nodes with a scaling constant of 500, the mean pool size is 5.43 with a standard deviation of 8.47. Although the worst case pool size increases as the number of hosts increases, it is still at a reasonable number for large N . For 8730 nodes, the average worst case query load at an RP is approximately 115 which is 1.3% of the total number of queries. About 18% of the nodes serve as rendezvous points, which means that the location queries are distributed over approximately 1607 nodes.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a location key based rendezvous mechanism to look up network neighborhood in a distributed P2P environment. Through large-scale simulation studies, we examined the performance tradeoffs of our system in terms of lookup efficiency, fraction of nodes that serve as RPs, and the querying load at the RPs. As part of the future work, we will explore alternative mapping functions to map Internet network locations to Chord key space. In particular, we will focus on the tradeoffs between the rendezvous point poolsize and the lookup efficiency of the rendezvous mechanism. Further, nodes which are isolated in network space will likely be mapped to a rendezvous point with no other nodes in the pool. In this case, we need an efficient lookup algorithm for these nodes to locate its “closeby” nodes on the Chord ring. We will also investigate the use of the coordinates-based rendezvous scheme in other scenarios, such as discovering network clusters among distributed peer-to-peer hosts and construction of efficient overlay networks.

ACKNOWLEDGMENT

This research is funded by the Singapore-MIT Alliance.

REFERENCES

- [1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *SIGCOMM’01*, 2001.
- [2] T. E. Ng and H. Zhang, “Predicting internet network distance with coordinates-based approaches,” in *INFOCOM*, 2002.
- [3] P. Francis, S. Jamin, C. Jin, Y. Jin, V. Paxson, D. Raz, Y. Shavitt, and L. Zhang, “Idmaps: A global internet host distance estimation service,” in *IEEE Infocom’99*, New York, NY, March 1999.
- [4] C. Partridge, T. Mendez, and W. Milliken, “Host anycasting service,” in *Request for Comments 1546, IETF*, November 1993.
- [5] S. Bhattacharjee, M. Ammar, E. Zegura, V. Shah, and Z. Fei, “Application-layer anycasting,” in *Infocom*, 1997.
- [6] C. Plaxton, R. Rajaraman, and A. W. Richa, “Accessing nearby copies of replicated objects in a distributed environment,” in *Theory of Computing Systems*, no. 32, 1999, pp. 241–280.
- [7] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *International Conference on Distributed Systems Platforms*, November 2001.
- [8] B. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An infrastructure for fault-resilient wide-area location and routing,” UCB/CSD, Tech. Rep., 2001.
- [9] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, “A scalable content-addressable network,” in *SIGCOMM’01*, San Diego, CA, 2001.
- [10] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, “Internet indirection infrastructure,” in *SIGCOMM’02*, 2002.
- [11] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, “Exploiting network proximity in peer-to-peer overlay networks,” in *Proceedings of the International Workshop on Future Directions in Distributed Computing*, Bertinoro, Italy, June 2002.
- [12] D. Karger and M. Ruhl, “Finding nearest neighbors in growth-restricted metrics,” in *ACM Symposium on Theory of Computing (STOC’02)*, May 2002.
- [13] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, “Brocade: Landmark routing on overlay networks,” in *First International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.
- [14] S. Banerjee, C. Kommareddy, and B. Bhattacharjee, “Scalable peer finding on the internet,” in *Globecom*. Taipei, Taiwan: IEEE, 2002.
- [15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” in *INFOCOM’02*. New York: IEEE, 2002.
- [16] T. Hansen, J. Otero, T. Mcgregor, and H.-W. Braun, “Active measurement data analysis techniques,” <http://amp.nlanr.net/>, 2002.