RAQNet: A Topology-Aware Overlay Network

Seyed Iman Mirrezaei[1], Javad Shahparian[1], and Mohammad Ghodsi[1,2,⋆]

[1] Computer Engineering Department, Sharif University of Technology, Tehran, Iran
[2] IPM School of Computer Science,Tehran, Iran
{mirrezaei,shahparian}@ce.sharif.edu, ghodsi@sharif.edu

Abstract. Peer-to-peer overlay networks provide a useful infrastructure for building distributed applications. These networks provide efficient and fault-tolerant routing and object locating within a self-organizing overlay network. This paper presents a multi-dimensional overlay network called RAQNet which is based on RAQ[1]. RAQ supports exact match queries and range queries over multi-dimensional data efficiently. Moreover, its routing cost does not depend on the dimension of the search space. In RAQNet, we have improved its original routing algorithms and extended it to have topology awareness property. In RAQNet, nodes are connected to each other if their labels are "close" to each other with respect to the topology of its underlying network. A topology match between the overlay and underlying network results in reduced routing delay and network link traffic. In comparison with RAQ, we will describe different node-join algorithms and routing table maintenance in order to provide the topology awareness. We present the experimental results through a prototype implementation of two emulated networks.

Keywords: Overlay Network, Topology Awareness, Proximity Metric.

1 Introduction

A peer-to-peer (P2P) overlay network is a logical network on the top of its physical layer. The overlay organizes the computers in a network in a logical way so that each node connects to the overlay network through its neighbors.

Several recent systems (CAN [10], Coral [12], Chord [11], Pastry [6] and Tapestry [4]) have recently appeared as flexible infrastructure for building large P2P applications. A DHT can be built using these networks, which allows data to be uniformly distributed among all the participants in such systems.

In these overlays, any item can be found within a bounded number of routing hops, using a small per-node routing table. While there are algorithmic similarities among these overlays, one significant difference lies in the approach they take to consider topology awareness in the underlying network. Chord, for instance, does not consider topology that it rides. As a result, its protocol for maintaining

⋆ This work has been partially supported by IPM School of CS (contract: CS1385-2-01) and Iran Telecommunication Research Center (ITRC).

[1] A Range-Queriable Distributed Data Structure [1].

A.K. Bandara and M. Burgess (Eds.): AIMS 2007, LNCS 4543, pp. 13–24, 2007.

the overlay network is very light weight, but queries may travel arbitrarily long distances in the underlying network in each routing hop.

Content Addressable Network (CAN) is another overlay network which assumes the attendance of a set of nodes that act as landmarks on the Internet, in order to optimize distances among nodes. Each CAN nodes measure their relative distances from this set of landmarks and measures its round-trip time to each of these landmarks and orders these values in order of increasing RTT. According to these values, topologically close nodes are likely to have the same ordering and so neighbors in the overlay are likely to be topologically close on the Internet [10].

Coral is a P2P content distribution system which is based on a distributed sloppy hash table (DSHT) [12]. In order to restrict queries to close nodes, Coral gathers nodes in groups called clusters. The diameter of a cluster is the maximum desired round-trip time (RTT) between any two nodes that it contains. So, Coral uses round-trip time as distance metric obtained from the underlying topology to obtain better performance [12].

We see that the mentioned overlays use underlying topological information to improve their communication performance. These overlays are aware of their underlying network and use this to improve their performance.

A mathematical model for topology awareness of P2P overlay networks has been introduced by Rostami et al [3]. They constructed their model based on an optimization problem called IP labeling. They also proved that IP labeling optimization is an NP-hard problem. So, it is impossible to build a perfect topology aware overlay network, but it can be solved in certain situations.

Based on RAQ [1], we present a new multi-dimensional overlay network, called RAQNet, with the topology awareness and improve its routing algorithms. RAQ supports exact match queries and range queries over multi-dimensional data efficiently. The routing cost in RAQ does not depend on the dimension of search space. In RAQNet overlay, nodes are connected to each other if they have the same labels and also are close to each other with respect to the topology of the underlying network. A topological match between overlay and underlying network resulted in reduced routing delays and network link traffic. We will describe a refined protocol for joining nodes and failure recovery in order to provide a topology-aware overlay network.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of the RAQ. Design of RAQNet and the new protocols for joining a node and failure recovery are presented in Section 3. Section 4 presents experimental results. We will conclude the paper in Section 5.

2 Basic RAQNet Structure

In this section, we introduce the basic design of RAQ and present a brief overview of its data structure.

2.1 Overview of RAQ Data Structure

In RAQ, the search space is d-dimensional Cartesian coordinate space which is partitioned among n nodes of the overlay network by a partition tree. Each node has $O(\log n)$ links to other nodes. Each single point query can be routed via $O(\log n)$ message passing. In addition, RAQ supports range queries as well as single point query through $O(\log n)$ communication steps. Each node is corresponded to a region and it is responsible for the queries targeting any point in its region. Furthermore, out-degree of a node and routing cost in RAQ is not dependent on the dimension of the search space. The partition tree splits the search space with no attention to the dimension of the search space.

2.2 Space Partitioning

The partition tree is the main data structure in RAQ which partitions the search space into n regions corresponding to n nodes. Assuming r is the root of partition tree and representing the whole search space, each internal node divides its region into two smaller regions using a hyperplane equation. Although only leaves in the partition tree represent actual network nodes, each node in this tree has a corresponding region in the search space. Every network node x which corresponds to a leaf in the partition tree assigned a *Plane Equation* or PE to specify its region in the whole space. Each PE consists of some paired labels which is defined as $X_{PE} = ((p_1, d_1), (p_2, d_2), \cdots, (p_{r(x)}, d_{r(x)}))$. In each label, $r(x)$ presents the distance of x from the root of the tree and p_i shows the plane equation that partitions the ith region into two regions and d_i determines one side of the plane p_i (left or right). Every leaf node in the RAQ stores its own PE as well as the PE of its links. Using theses information, every node like x can locally recognize whether a requested query belongs to a node to the left or the right of x or to the left or right of any of its links in the partition tree. Figure 1 (right) portrays partitioning of 2-dimension search space. In figure 1 (left), the PE of node c is $[(x = 4, -), (y = 2, +), (x = 2, +), (y = 1, -)]$. We use "+" and "-" in the PE of nodes to determine one side of the plane (left or right).

2.3 Network Links in RAQ

Every node has some links to other nodes of the network. Each link is the addressing information of the target node which can be its IP address and its PE. Connection rule in RAQ is based on partition tree. Consider the node x and its PE, x has link to one of node in each of these sets: $[((p_1, \bar{d_1}))]$,$[((p_1, d_1), (p_2, \bar{d_2}))]$,$\cdots$, $[((p_1, d_1), (p_2, d_2), \cdots, (p_{r(x)}, \bar{d}_{r(x)}))]$, where $\bar{d_i}$ is the opposite side of d_i. It is easy to show that each node has links to $O(\log n)$ nodes in RAQ.

2.4 Query Routing in RAQ

Whenever a node in the network receives a single point query, it must route the query to the node which is responsible for the region containing the point. Once

the query Q is received by a node z, if destination point matched with PE of node z completely, then routing is finished. Otherwise, node z sends the query via a network link to a node y with a PE that matches the destination point at a higher level. This will go on further until the query reaches the destination node.

3 Design of RAQNet

In this section we modify RAQ to build a topology aware overlay network. We select node's link based on RAQ data structure and also based on topology of underlying network. Additionally, we hold more node pointers in routing tables in comparison to the basic data structure. A new routing table is also added. We thus propose different procedures for join, departure, and maintenance of RAQNet overlay in order to provide topology awareness.

Each RAQNet node has a fairly random point in a d-dimensional Cartesian search space. As in RAQ, search space is a logical space that is divided among network nodes and each node is responsible for responding to the queries matching with its PE. We suppose that PE of nodes are strings and contains some paired label as we mentioned before. We enforces some constraints on the plane equations that a node may choose when it joins the network and splits another region node. These constraints cause the PE of nodes remain simple after node's join or departure. The constraints that we enforce are the following:

- Each plane should be perpendicular to a principal axis. Hence, in a d-dimensional space of $(x_1, x_2, \cdots, x_d)$ each plane takes the form of $x_i = c$ for some $1 \leq i \leq d$ and some value of c. This effectively means that each plane equation partitions the regions in the space based on the value of x_i for some i.
- If the search space is d-dimensional, we define the form of the plane equation that may be assigned to an internal node depending on the depth of that node. If A is an internal node, the plane equation assigned to A must be of the form $x_i = c$ for an arbitrary value of c, that is for any given i, all of the nodes whose depth numbers are i are assigned plane equations of the form $x_i = c$, so that regions can be re-merged when node leaves the overlay. For a 2-d search space, All the internal nodes which are in depth i split the search space along dimension X.

This implies that whenever a new node joins the RAQNet and divides the region of another node which leads to a new internal node, the plane equation of that internal node must obey the above constraints.

3.1 Routing Tables in RAQNet

The routing state maintained by each node consists of a *routing table* and a *hop table*. Each entry in the routing tables contains the PE and IP address of a node.

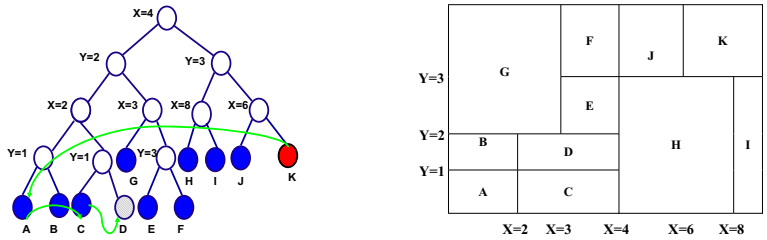

Fig. 1. Left: Routing a query from node whose PE is $[(x = 4, +), (y = 3, +), (x = 6, +)]$ to destination point $(2.5, 1.5)$, Right:2-dimension search space

Routing Table. The routing table is organized with $O(\log n)$ rows and 2^t columns, where t is a configuration parameter with typical value of 2. The entry in rth row and nth column of the routing table refers to a node whose PE that shares the first r labels with the local node's PE, and its $(r + 1)$th label of node's PE, corresponds to plane like $x_{r+1} = c$. All entries in row r were sorted increasingly according to values of $(r + 1)$th label of PE. Figure 2 depicts a sample routing table. This routing table is similar to those used by Tapestry[4] and RRR[5].

Each entry in the routing table contains the IP address of one of potentially many nodes whose PE have the appropriate prefix; in practice, a node is chosen that is close to the local node, according to the topology of underlying networks. We will show in 3.3.

Hop Table. The hop table is the set of H nodes with half of label's of their PE that are shared with the present node's PE's. All nodes in hop table are sorted increasingly according to $\left|\frac{present\ node_{PE}}{2}\right| + 1$th label of their PE. A typical value for H is approximately 2^t or $2 * 2^t$. Figure 2 shows a routing table and hop table of node c whose PE is $(X = 4, -), (Y = 2, -), (X = 2, +), (Y = 1, -)$.

3.2 Query Routing

At each routing step, the current node usually sends the query to a node whose PE shares at least one label longer with the destination point than the prefix with the local node's PE. If no such node is known, the query is sent to a node whose PE is closer to the destination point and shares a prefix with the destination point having the same length. If there is no such node, the query is delivered to the local node because it is closest node to the destination point. Before sending a query to the one of the nodes in rth row , we search for the proper node whose $(r + 1)$th label of its PE is also matched with the destination point.

3.3 Neighbor Selection Based on Topology Awareness

This section focuses on topology aware property. RAQNet seeks to exploit topology awareness from the underlying network in order to fill its routing table rows

Routing Table

(X=4,+)	H	I	J	K
(X=4,-),(Y=2,+)	G	E	F	*
(X=4,-),(Y=2,-),(X=2,-)	*	A	B	*
(X=4,-),(Y=2,-),(X=2,+),(Y=1,+)	*	D	*	*

Hop Table

(X=4,-),(Y=2,-)	A	B	D

Fig. 2. Left: Routing table of node c, Right: hop table of node C. The associated IP addresses are not shown. If no node was known with a suitable PE, then the routing table entry is filled with "*".

```
1.  if ( z.isInPlaneEquationofHopTable(l)
2.       //Use the hop table
3.       forward to H_i such that H_i is closer to z than   other nodes in H table
4.  else
5.       //Use the routing table
6.       Let r = PlaneEquationMatch(z, l)
7.       Let c = FindingProperColumn(z, r)
8.       if (R_r^c exists)
9.            Forward to R_r^c
10.      else
11.           //Rare case
12.           forward to t ∈ H ∪ R such that
13.                PlaneEquationMatch(z, l) ≥ r
```

Fig. 3. RAQNet Routing procedure, when a query with destination point z arrives at a node whose PE is l. R_r^c is the entry in the routing table R at cth column and rth row.

effectively. Hence, any node with the required prefix in PE can be used to fill an entry, topology aware neighbor selection selects the closest node in the underlying network among nodes whose PE have the required prefix. Topology awareness relies on a proximity metric that indicates the "distance" between any given pair of nodes. The choice of a proximity metric depends on the desired quality of the resulting overlay (e.g., low delay, high bandwidth). Our proximity metric in RAQNet overlay network is round trip time.

Topology aware neighbor selection was first proposed in PRR [5] and pastry [6]. In RAQNet, the expected distance traveled in the beginning routing hop is small and it increases at each successive routing step. Because the number of nodes decreases with the increasing length of the prefix match between their PE and the destination point.

3.4 Node Join

When a new node ,x , joins the overlay, it chooses a fairly random point X in the search space and contacts an existing close node e sending its join request. The close node e can be found using IP multi-cast in some applications or the algorithm described in Section 3.7. RAQNet uses the join mechanism similar to pastry [6] as follows.

Node e routes join request using X as the query, and x gets the first row of its routing table and first label of its PE from the node e. Then e forwards the join request to the second node which sends second row of its routing table and second label of its PE to x and so forth. We will show that x's resulting routing table is filled with close nodes if node e is close to x, according to the proximity metric.

We assume that triangle inequality holds in the proximity space and entries of each node's routing table refers to overlay nodes close to itself according to proximity metric.

x is close to e because we search for a close node to send join request. Also, the nodes in the first row of e's routing table are close to e. Due to triangle inequality, these nodes are also close to x. This holds for the next rows in the same way.

It is also important to update other node's routing tables to ensure that they are filled with close nodes after new nodes join the overlay network. Once x has initialized its own routing table, it sends the each row of its routing table to each node that appears as an entry in that row. This causes both to announce its attendance and to spread information about new nodes that joined before. Each node receives a row then checks the nodes in the row to measure if x or one of the entries is closer than the corresponding entry in its own routing table, and updates its routing table properly. This procedure ensures that routing tables filled with close nodes. Additionally, x and the nodes that appear in nth row of x's routing table form a group of 2^t close nodes whose PEs share in the first n labels. It is clear that these nodes need to know of x's entrance since x may displace a more distant node in one of the node's routing tables. In an opposite way, a node with same prefix in the first n labels of its PE that is not a member of this group is more distant from the members of the group, and therefore from x. Thus, x's entrance is not likely to affect its routing table and it does not need to be informed of x's entrance.

3.5 Node Departure

According to RAQ [1], each node has departure links to the nodes which have links to it. When a node decides to leave overlay, it informs their neighbors by departure links. All nodes that receive this message, mark their corresponding entry in the routing table. Instead of using a marked entry to route a query, RAQNet routes the query to another node in the same row whose PE also matches the destination point. If the next node has a proper entry that matches the next label of the destination point, it automatically informs the previous node of that entry. The next node is usually an entry in the same row as the failed node. If that node provides an alternative entry for the failed node, its expected distance from the local node is low since all three nodes were member of the same group of close nodes with same PE prefix. If no replacement node is supplied by the next node, a replacement is found by triggering the routing table maintenance task, which is described next.

3.6 Routing Table Maintenance

Whenever an overlay node could not find an alternative entry for its failed entry, it triggers the maintenance procedure to handle this problem.

Another concern is that deviations could cascade and lead to a slow deterioration of the topology aware properties gradually. To prevent a deterioration of the route quality, each node runs a periodic routing table maintenance task (e.g., every 20 minutes). The maintenance task performs the following procedure for each row of the local node's routing table. It selects a random entry in the row, and requests a copy of associated node's row. Each entry in that row is compared to the corresponding entry in the local routing table. If they differ, the node probes the distance to both entries and puts the closest node in its own routing table.

3.7 Locating a Nearby Node

When a new node x want to join to overlay, it should contact the close node e around itself to fill its routing table with close nodes properly. Karger et al [8] proposed an algorithm to find close node but this would require maintaining additional information. In Figure 4 we describe an algorithm to find a close overlay node to x. This algorithm is interesting because it does not need any other information beyond the routing table and hop table that are already preserved by RAQNet nodes.

```
1.  discover (anyNode)
2.      nodes = getHopTable (anyNode)
3.      nearNode = pickClosest(nodes)
4.      depth = getMaxRoutingTableLevel(nearNode)
5.      closest = nil
6.       while (closest ! = nearNode)
7.           closest = nearNode
8.           nodes = getRoutingTable(nearNode,depth)
9.           nearNode = pickClosest(nodes)
10.          if (depth > 0 ) depth = depth −1
11.     end
12.     return closest
```

Fig. 4. Finding near node

This algorithm exploits position of node in the network. In each step, distance of all nodes in the same row is checked in order to find closer node from joining node. This is achieved bottom up by picking the closest node at each level and getting the next level from it. This performs a constant number of probes at each level but the probed nodes get closer at each step. The last phase repeats the process for the top level until there is no more progress. As it was showed in RAQ, routing tables have $\log n$ rows. Hence, the complexity of this algorithm is $O(\log n)$ too (n is number of nodes in the overlay network).

4 Experimental Results

In this section, we present experimental results quantifying the performance of topology aware neighbor selection in RAQNet under realistic conditions. The results were obtained using a RAQNet implementation running on top of a network simulator, using Internet topology models. The RAQNet parameter was set to $d = 2$. Higher dimensions can be used without imposing extra over-head because routing mechanism of RAQNet does not depend on the dimension of the search space. Our results obtained with a simulated RAQNet overlay network of 10,000 nodes.

4.1 Network Topologies

Two simulated network topologies were used in the experiments. In the "Sphere" topology nodes are placed at uniformly random locations on the surface of a sphere with radius 1000. The distance metric is based on the topological distance between two nodes on the sphere's surface. However, the sphere topology is not realistic, because it assumes a uniform random distribution of nodes on the Sphere's surface, and its proximity metric satisfies the triangulation inequality. A second topology was generated by the Georgia Tech transit-stub network topology model[9]. The round trip delay (RTT) between two nodes, as provided by the topology graph generator, is used as the proximity metric with this topology. As in the real Internet, the triangle inequality does not hold for RTTs among nodes in the this topology. Our experimental results are significantly good for both topologies although our assumption of triangle inequality does not hold for the second topology.

4.2 Routing Hops and Distance Ratio

In the first experiment, 200 lookup queries are routed using RAQNet from randomly chosen nodes, using a random point. Figure 5 (left) shows the number of RAQNet routing hops and the distance ratio for the sphere topology. Distance ratio is defined as the ratio of the distance traveled by a RAQNet query to the distance between its source and destination nodes, measured in terms of the proximity metric. The distance ratio can be interpreted as the penalty, expressed in terms of the proximity metric, associated with routing a query through RAQNet instead of sending the query directly in the Internet.

Two sets of results are shown. "RAQ" shows the corresponding experimental results with RAQ. "RAQNet" shows results of experiments in RAQNet overlay network. According to analysis in RAQ [1], the expected number of routing hops is slightly below $\frac{\log 10000}{2} = 6.64$ and the distance ratio is small. The reported hop counts are independent of the network topology, therefore we present them just for the sphere topology.

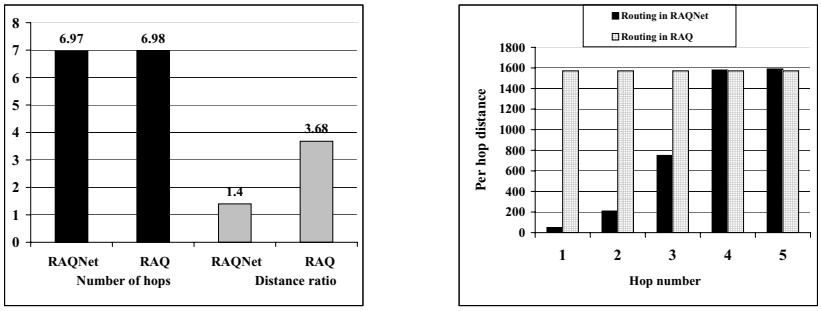

Fig. 5. Left: Number of routing hops and distance ratio in the sphere topology, Right: Distance traversed per hop in the sphere topology

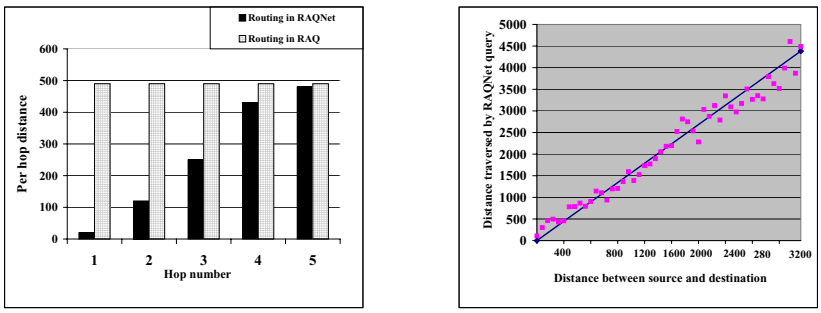

Fig. 6. Left: Distance traversed per hop in the GT-ITM topology, Right: Number of routing hops and distance ratio in the sphere topology

4.3 Routing Distance in RAQNet

Figure 5 (right) shows the distance messages travel in each following routing hops. The results confirm the increase in the expected distance of following hops up to the fourth hops. Moreover, in the absence of the topology awareness, the average distance traveled in each hop is constant and corresponds to the average distance between nodes which are placed on the surface of a sphere $1571 = \frac{\pi * r}{2}$ (where r is the radius of the sphere).

Figures 6 (left) shows the same results for the GT-ITM topology respectively. Due to the nonuniform distribution of nodes and the more complex proximity space in this topology, the expected distance in each following routing step still increases monotonically. However, the node join algorithm continues to produce routing tables that refer to close nodes, as indicated by the modest difference in hop distance to the routing tables in the first three hops.

Figures 6 (right), and 7 (left) show raster plots of the distance query travel in RAQNet, as a function of the distance between the source and destination nodes, for each of the two topologies. Queries were sent from 50 randomly chosen source nodes to random destination points in this experiment. The mean distance ratio

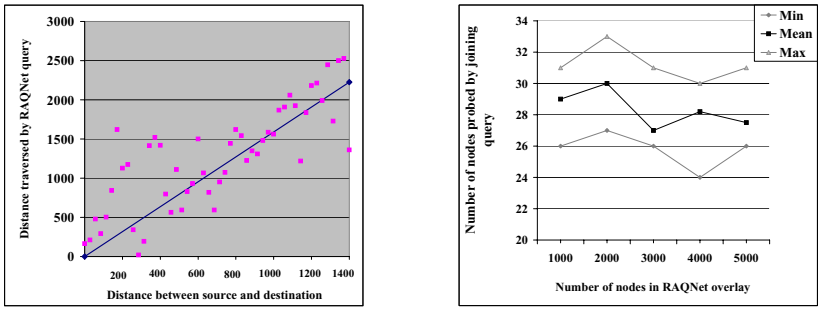

Fig. 7. Left: Number of routing hops and distance ratio in the GT-ITM topology, Right: number of probes by a newly joining node

is shown in each graph as a solid line. The results show that the distribution of the distance ratio is relatively firm around the mean. Not surprisingly, the sphere topology produces the best results, because of its uniform distribution of nodes and the geometry of its proximity space. However, the far more realistic GT-ITM topology produces still good results, with a mean distance ratio of 1.63, a maximal distance ratio of about 8.3, and distribution that is fairly firm around the mean.

4.4 Overhead of Node Join Protocol

In this section, we measure the overhead incurred by the node join protocol to preserve topology awareness in the routing tables. We measure this overhead in terms of the number of probes, where each probe corresponds to the communication required to measure the distance according to the proximity metric between two nodes. Of course, in our simulated network, a probe simply involves looking up the corresponding distance according to the topology model. However, in a real network, probing would likely require at least two message exchanges. The number of probes is therefore a meaningful measure of the overhead required to maintain the topology awareness. Figure 7 (right) shows the maximum, mean and minimum number of probes performed by a node joining the RAQNet overlay network. This overhead is independent of number of nodes which we varied from 1,000 to 5,000 nodes. In each case, the probes performed by the last ten nodes that joined the RAQNet overlay network were recorded. It is assumed here that once a node has probed another node, it stores the result and does not probe again.

5 Conclusion

This paper presented a new multi-dimensional topology aware overlay network and analysis as well as an experimental evaluation of the RAQNet. A refined protocol for node joining and node failure recovery achieves in order to provide

topology awareness in RAQNet overlay network. Experimental results showed that topology aware properties can be achieved with low overhead in network topologies. Additionally, simulations on two different Internet topology models show that these properties can hold in more realistic network topologies. The results also show that considering topology awareness can be provide a significant performance improvement relative to topology unaware routing.

Acknowledgments. The authors would like to thank Payam Bahreini, Hesam Chiniforoushan and Hojatollah Vaheb for their reviews and supports.

References

1. Nazerzadeh, H., Ghodsi, M.: RAQ: A range queriable distributed data structure (extended version). In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 264–272. Springer, Heidelberg (February 2005)
2. Alaei, S., Toossi, M., Ghodsi, M.: SkipTree: A Scalable Range-Queryable Distributed Data Structure for Multidimensional Data. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 298–307. Springer, Heidelberg (2005)
3. Rostami, H., Habibi, J.: A Mathematical Foundation for Topology Awareness of P2P Overlay Networks. In: Zhuge, H., Fox, G.C. (eds.) GCC 2005. LNCS, vol. 3795, pp. 906–918. Springer, Heidelberg (2005)
4. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-resilient wide-area location and routing, Tech. Rep. UCB//CSD-01-1141, U.C. Berkeley (April 2001)
5. Plaxton, C.G., Rajaraman, R., Richa, A.W.: Accessing nearby copies of replicated objects in a distributed environment. In: Proc. 9th ACM Symp. on Parallel Algorithms and Architectures, June 1997, Newport, Rhode Island, USA, pp. 311–320 (1997)
6. Rowstron, A., Druschel, P.: Pastry: Scalable,distributed object location and routing for large-scale peer-to-peer systems. In: Proc. IFIP/ACM Middleware 2001, Heidelberg, Germany (November 2001)
7. Costa, M., Castro, M., Rowstron, A., Key, P.: PIC: Practical Internet Coordinates for Distance Estimation. In: 24th IEEE International Conference on Distributed Computing Systems (ICDCS' 04), Tokyo, Japan (March 2004)
8. Karger, D.R., Ruhl, M.: Finding nearest neighbors in growth-restricted metrics. In: STOC'02 (July 2002)
9. Zegura, E., Calvert, K., Bhattacharjee, S.: How to model an internetwork. In: INFOCOM96 (1996)
10. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. of ACM SIGCOMM (August 2001)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM '01 Conference, San Diego, California (August 2001)
12. Freedman, M., Mazieres, D.: Sloppy hashing and self-organizing clusters. In: Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS03) (2003)