Improved Locality-Aware Grouping in Overlay Networks

Matthias Scheidegger and Torsten Braun

IAM, Universität Bern, Neubrückstrasse 10, 3012 Bern, Switzerland

Abstract The performance of peer-to-peer and overlay networks depends to a great extent on their awareness of the underlying network's properties. Several schemes for estimating end-to-end network distances have been proposed to simplify this task. The mOverlay framework identifies groups of nodes that are near to each other in the network topology. Instead of distances between nodes mOverlay measures distances between groups. However, mOverlay's locating procedure has a number of drawbacks. We propose an alternate method for identifying groups using Meridian's closest node search. Simulation results based on PlanetLab measurements indicate that the Meridian-based approach is able to outperform mOverlay in terms of joining delay, the size of the identified groups, and their suitability for a distance estimation service.

1 Introduction

Peer-to-peer and overlay networks use logical topologies rather than the physical topology of the underlying network. This allows them to achieve many desirable aims like high scalability or high resilience to node failure. Nevertheless, creating a logical topology without considering the properties of the underlying network may lead to considerable inefficiency. A single logical link connecting two nodes may in fact span many links on the physical network. This property is commonly referred to as the stretch of an overlay topology. Overlay networks usually perform better if neighbors in the overlay topology are also close to each other in the physical network.

Most early designs for peer-to-peer and overlay networks, like the unstructured peer-to-peer networks Gnutella [7] and Freenet [8], are unaware of the underlying network. More recent designs often consider the properties of the underlying network in some way. For example, a binning scheme based on round-trip times [10] can be used to optimize content-addressable networks (CANs) [9]. The routing algorithm in Pastry [11] also considers the round-trip time between overlay nodes to find the optimal next hop when forwarding queries. Application-level multicast is especially sensitive to the choice of topology. Accordingly, these systems focus on optimizing this aspect. Scribe [13] uses Pastry's routing algorithm to build efficient multicast trees. The MULTI+ approach [12] hierarchically groups overlay nodes according to their IP network prefixes to create a multicast tree. This is based on the idea that similar IP prefixes indicate a neighborship in the network.

Since adaptation to network properties is a common problem in peer-to-peer and overlay networks, several frameworks and services have been proposed to help with this task. One of them is mOverlay [1], a locality-aware overlay that assigns nodes to groups depending on their distance in the underlying network. A dynamic landmark procedure determines the closest group for each overlay node. Each mOverlay group has a set of leader nodes, which store measured distances to other groups. Using this structure, overlay applications can easily distinguish between local links and more expensive links to other groups. Furthermore, these groups can be used as equivalence classes for distance estimation. If two nodes are in the same group, their respective distances to a remote node should be approximately the same. The distance between two nodes can therefore be estimated by the distance between their respective groups. In many cases an application only needs to find the closest overlay node to a given IP address and is not interested in the exact distance between the overlay node and the end system in question. Meridian [2] is a lightweight framework that efficiently solves this closest node problem. Each Meridian node keeps track of an incomplete, fixed-size set of other Meridian nodes, which is updated using a gossip protocol. When a node receives a closest node request it will choose the nearest node to the destination from its set and forward the request to that node.

In recent work [4] we have proposed an overlay distance measurement service using local groups similar to those in mOverlay. In addition, our approach is also able to detect if remote nodes are close together. This is achieved by analyzing time series of distance measurements to remote hosts (obtained using `ping` for example). Similarities in any two time series indicate that the respective remote nodes are close to each other. This enables two improvements. First, the service can be deployed locally because remote hosts only need to respond to standard tools like `ping` and do not have to run any special software. Second, when looking at the network from a given position, far away groups are often indistinguishable from each other and can be viewed as a single entity. In such cases we only need to store a single data record for a set of groups, which improves the scalability of the service. Initially, we planned to use mOverlay as a mechanism to identify local groups. However, we have found that replacing a part of mOverlay's locating algorithm with Meridian's closest node search improves its accuracy and reduces the time it takes for a node to join the overlay network. Our contribution in this paper is the modified locating algorithm, which we compare to the dynamic landmark procedure originally proposed for mOverlay in [1]. The resulting group structure is also useful to simplify the task of optimizing overlay topologies, which is a hard problem if the topology is large. Topologies based on groups preserve inter-node distances but are much smaller than topologies based on individual nodes and are thus well-suited for solving optimal routing problems.

The remainder of the paper is organized as follows: Section 2 discusses related work, including mOverlay and Meridian. Section 3 describes the modified locating algorithm. In Section 4 we present the simulators used to compare the two approaches, and we discuss the simulation scenarios and results in Section 5. Section 6 concludes the paper.

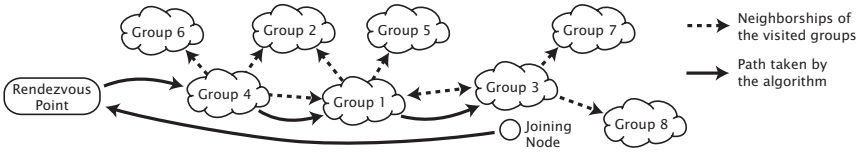

Figure 1. A joining node contacts the mOverlay rendezvous point and finds the nearest group 3 via groups 4 and 1.

2 Related Work

2.1 mOverlay

The mOverlay [1] framework uses a two tier overlay structure. At tier one, nodes that are close to each other form groups and communicate directly with other members of the same group. At tier two, groups select a number of nearby groups as their neighbors. The groups are chosen such that they can be used as equivalence classes concerning the distance metric. This reduces the endpoint-to-endpoint distance estimation problem to the much smaller problem of estimating distances between groups. Moreover, this structure can serve as a basis for constructing efficient overlay topologies because it distinguishes between efficient short distance links inside the groups and potentially inefficient long distance links between groups. In order to decide whether or not a joining node belongs to a given group the following *grouping criterion* is used [1]:

> When the distance between a new host Q and group A's neighbor groups is the same as the distance between group A and group A's neighbor groups, then host Q should belong to group A.

New nodes iteratively search for a group that meets this grouping criterion. When a node joins the overlay network it first contacts a rendezvous point and obtains contact information for a set of randomly chosen boot hosts. For each boot host it starts a locating process, which tries to find a suitable group for the node. Using several locating processes increases the robustness of the approach. The algorithm starts by contacting a boot host, which returns a set of distances between the boot host's own group and its neighbors. The joining node then measures and compares its own distances to these neighbor groups. If the grouping criterion is met the process terminates and the node joins the group of the boot host. Otherwise, the new node chooses the neighbor group that is nearest to it and repeats the process. After a predefined number of unsuccessful iterations, or if all available groups have been visited, the new node creates its own group. When a node creates a new group it selects its neighbors from the closest groups it has seen during the locating process, and their neighbors. It then contacts each of the selected neighbors in order to allow them to adjust their own neighbor tables if needed. Figure 1 illustrates the locating algorithm. Solid arrows indicate steps in the algorithm and dashed ones indicate neighborships between the groups that are used to check the grouping criterion.

2.2 Meridian

Meridian [2] is a "framework for performing network positioning without embedding nodes into a global virtual coordinate space." It has a another focus than mOverlay. Its three main functions are closest node discovery, central leader election, and multi-constraint search. For our work we use Meridian's closest node discovery. Meridian nodes form a loosely connected overlay network. They exchange information about other overlay nodes using a gossip protocol and keep track of a fixed number of peer nodes. These nodes are sorted into non-overlapping, concentric rings of exponentially growing width around the Meridian node (see Figure 2). The ith ring contains nodes with latencies between αs^{i-1} and αs^{i} from the center, and the outermost ring contains nodes with latencies αs^{i^*} and more. Within each ring, the nodes are selected to maximize diversity, which is quantified through the hyper-volume of the k-polytope formed by the selected nodes. A closest node search aims to identify the Meridian node that is

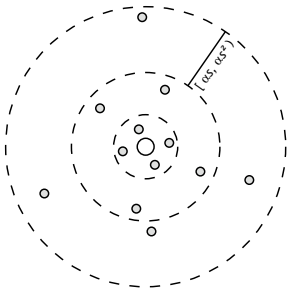

Figure 2. Meridian nodes arranged into rings based on their distance

closest to a given end system E in the network. To start the procedure we send a request to an arbitrary Meridian node. This node measures its latency to E and selects the nodes with similar latencies from its cache. It then contacts each of these nodes and asks them to measure and report their latency to E. The node with the smallest latency to E becomes the next hop, and the procedure repeats. When the next hop is only insignificantly closer than the current one the closest node search terminates and the current node is selected.

2.3 Other Work on Distance Estimation

A considerable amount of work on network distance estimation has been published in recent years. One of the earliest designs, IDMaps [14], is a distance estimation service that relies on tracers placed at key locations throughout the network. The distance between two clients is estimated by the sum of the distances between the nodes and their respective nearest tracers, plus the distance

between those tracers. Dynamic Distance Maps [15] uses a similar way to estimate distances, but uses the tracers to hierarchically decompose the Internet into regions. An important part of the work on network distance estimation focuses on coordinate-based approaches, which normally embed measured network distances in n-dimensional Euclidean space such that the Euclidean distance between two nodes is a good estimate of their distance in the network. GNP [16] is a prominent member of this family. Clients measure their distance to a fixed set of landmark nodes with known coordinates and compute their own coordinates using simplex downhill minimization. It has been argued that its fixed set of landmarks impairs GNP's scalability and makes it vulnerable to attacks and node failures. Consequently, more robust approaches like Lighthouse [17] have been proposed. Here, subset of overlay nodes may be used as landmarks, or lighthouses. Vivaldi [18] does not use any landmarks. Instead, it passively monitors network traffic to obtain distance measurements and applies a distributed algorithm to iteratively adjust the coordinates of the nodes.

3 An Alternative Locating Algorithm

A problem of mOverlay is its topology. Because the groups choose neighbors from their close proximity the logical links between the groups are very short. This affects the performance of the locating algorithm, since the algorithm follows the topology and thus can only make small steps towards the target node. If the target node is far away, taking bigger steps would be more efficient. Another problem is that mOverlay's topology is prone to so-called net-splits.

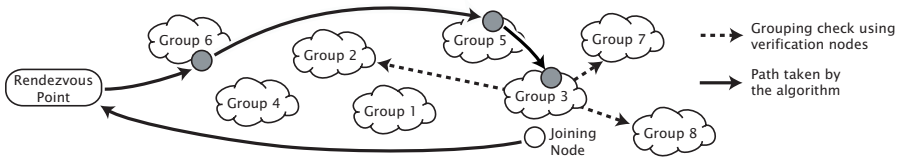

Figure 3. Alternative algorithm using Meridian's closest node search and mOverlay's grouping criterion

In order to overcome these problems we have defined an alternative group locating algorithm based on both mOverlay and Meridian. We take the group concept from mOverlay but change the overlay structure. The groups no longer have neighbors. Instead, the group leaders become Meridian nodes. When a new node wants to join, it locates a boot node using the rendezvous point. Then, it asks this boot node to start a Meridian closest node search with the joining node itself as target. The search returns the address of the closest group leader. At this point, the new node checks the grouping criterion to find out whether or not to join this group. If the criterion is met the new node joins the group. Otherwise, it creates a new group and becomes a Meridian node itself.

Unfortunately, we cannot directly use mOverlay's grouping criterion, because in our alternative approach groups do not have neighbors. We solve this problem using Meridian's node cache. The group leader found by Meridian's closest node search selects a randomly chosen set of *verification nodes* from its node table and creates a list of addresses and latencies to these nodes. The new node receives this list and in turn measures its latency to each of the verification nodes. This provides us with two comparable sets. Furthermore, because mOverlay's grouping criterion is formulated in general terms we also need to specify exactly when two distances can be considered "the same." We say distances x and y are the same if

$$\begin{array}{rcl} x \geq y & \wedge & (1-g) \cdot x \leq y \\ \vee\, x < y & \wedge & (1-g) \cdot y \leq x \end{array} \tag{1}$$

for a *grouping threshold* $g \in [0, 1)$. The test checks the relative difference between two distances. For example, with a grouping threshold of 0.05 we consider two distances the same if they are within $\pm 5\%$ of each other. A new node joins a group if test (1) succeeds for every verification node. The algorithm is illustrated in Figure 3.

We believe that this combined approach to grouping nodes solves the problems discussed above. Meridian's closest node search makes the search more efficient. The approach is also less prone to net-splits than mOverlay because Meridian nodes maintain a more diverse set of peer nodes. The loose overlay structure also makes the system more resilient to node failure. A possible drawback of our algorithm is that it only checks the grouping criterion for a single group, which bears the danger that the algorithm might skip over the optimal one. Fortunately, the results in Section 5 suggest that this kind of error is rare.

4 Implementation of Simulations

In order to compare the performance of mOverlay's locating algorithm to our alternative algorithm we have implemented simulators for the two approaches. Both simulators are based on a black box network model given by a matrix of the end-to-end latencies between each pair of endpoints in the simulation. For our experiments we use a matrix derived from all-sites ping data measured on PlanetLab [5]. In both simulators the nodes join the overlay network one after the other, in pseudo-random order (given by the seed value). For each node we record the time that expires until it joins a group or creates its own group. When the simulation ends we examine the resulting groups according to several criteria, which we discuss in Section 5.

4.1 mOverlay

We simulate mOverlay with a simple message-based approach where each message fits into a single packet and the message processing at a node does not take any time. Thus, a request-response message exchange takes exactly one round-trip time to complete, which is a lower bound for any real implementation of the

framework. Furthermore, we skip mOverlay's initial request to the rendezvous point because the performance of this step depends heavily on the implementation of the mechanism (e.g. a well-known address, a DNS-based approach, etc.) and possibly on the placement of the rendezvous point.

In the simulator, the locating processes of a joining node run in parallel and stop when one of them finds a group that meets the grouping criterion. A locating process also stops if its next hop would be a group it has already visited. If none of the locating processes are successful, the joining node gives up and creates a new group. Locating processes keep a list of visited groups. When a new group is created its neighbors are selected from the lists of all its locating processes. The first two joining nodes are special cases. They automatically create new groups because the grouping criterion cannot be evaluated without further nodes. As mentioned in Section 3, mOverlay does not define how to test if two distances are the same. However, we need to test this to check the grouping criterion. We have used test (1) from Section 3 also for the mOverlay simulation because it is a natural choice.

4.2 Meridian

In contrast to the mOverlay simulator, where we implemented all necessary messages, we did not implement the Meridian approach ourselves. Instead, we have used the official Meridian C++ implementation [3]. We have written wrapper code to redirect any messages to a simulation back-end instead of the network, and we have changed Meridian's time-keeping code to use the simulation time instead of the system clock. Each Meridian node is now a C++ object in the simulator rather than a physical node on the network. When it sends a packet the simulator determines the appropriate transmission latency using the underlying network model and schedules the packet arrival at the destination node accordingly. The wrapper objects also evaluate the grouping criterion at the end of a joining procedure and create a new group if necessary.

The simulation back-end is event-based. There are three kinds of events: one for inserting a new node into the scenario, one for triggering Meridian's periodic gossip protocol, and one for packet arrivals at a Meridian node. We start the simulation by scheduling node join events every seven seconds (which corresponds to Meridian's default gossip interval). When a node joins it starts by sending a closest node query to a Meridian node. This search is handled entirely by the original code. When the query returns, the joining node contacts the identified closest node to retrieve a list of verification nodes, which the wrapper code extracts from the Meridian object's latency cache. In the simulator we use a maximum of five verification nodes.

5 Simulation Results

5.1 Simulation Scenario

For the simulations we have used a matrix of round-trip times between 77 PlanetLab nodes, based on all-sites ping data from PlanetLab [5]. The simulator

estimates the one-way delay between two endpoints by dividing the appropriate round-trip time by two. At the time of writing, 694 machines hosted by 335 sites were part of PlanetLab. This means that each site hosts only slightly more than two machines on average. Consequently, we can expect to find groups of only a few nodes each in our scenario, especially since the 77 nodes in the network model were randomly selected from the available PlanetLab nodes. For each node pair we have also acquired a time series of round-trip times, which we use for evaluation. The time series contain round-trip time measurements made every 15 minutes during one day. We have originally planned to use more than 77 nodes, but we have removed several nodes from the original set due to missing values in the time series. Simulations have been run with different values for various parameters. Furthermore, each set of parameters was simulated using 100 different seeds, which we obtained from random.org [6].

5.2 Evaluation Criteria

We get the joining delays for every node, and the identified groups from a simulator run. While the comparison of the joining delays is straightforward, quantifying the quality of the identified groups is not. Grouping can exhibit two kinds of errors, false positives and false negatives. A node joining a group when it should not is considered a false positive and increases the error of grouping. A false negative occurs when a node erroneously does not join a group and creates a new one instead. This results in too many groups and impairs the efficiency and scalability of the overlay network. Unfortunately, due to the black box nature of our network model, we cannot say a priori whether a node should join a group or not. Nevertheless, we can define three criteria for the quality of the identified groups.

- First, the members of a group should be close to each other. Accordingly, we compute the mean round-trip time between members of the same group. Groups with only one node are ignored in this case.
- Second, bigger groups are preferable because they reduce the complexity of the overlay network. We use the average number of nodes per group as the second criterion.
- The third criterion stems from the use of the identified groups as a basis for a distance estimation service. One important assumption in mOverlay is that if two nodes A and B are in the same group, the distances $\overline{AC}$ and $\overline{BC}$ to a node C outside the group are virtually the same. This property enables significantly better scalability of the service. However, it must also hold over time. Otherwise, we would have to reorganize the groups constantly. We define the third criterion accordingly: If A and B are in the same group, $\overline{AC}_t$ should be a good prediction for $\overline{BC}_t$, where t is the time of measurement. We verify this using the time series of round-trip times between the two nodes. Two measurements $\overline{AC}_t$ and $\overline{BC}_t$ are out-of-band of each other if

$$\begin{array}{rcl} \overline{AC}_t \geq \overline{BC}_t & \wedge & (1-b) \cdot \overline{AC}_t > \overline{BC}_t \\ \vee\, \overline{AC}_t < \overline{BC}_t & \wedge & (1-b) \cdot \overline{BC}_t > \overline{AC}_t \end{array} \tag{2}$$

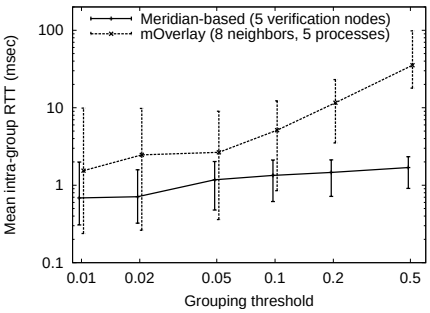

Figure 4. Mean intra group distance for several grouping thresholds

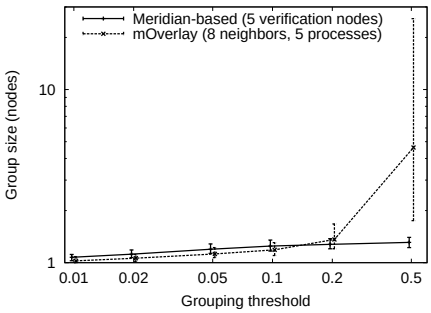

Figure 5. Avg. nodes per identified group for several grouping thresholds

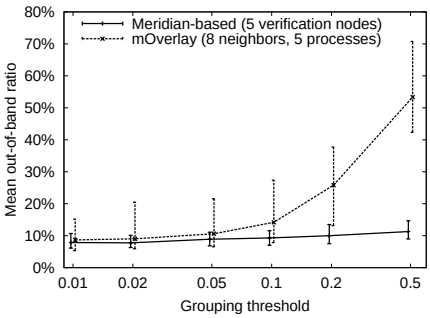

Figure 6. Mean out-of-band ratio using a 10% band, for several grouping thresholds

for a relative bandwidth $b \in [0, 1)$. The *out-of-band ratio* between two nodes is the ratio of out-of-band measurements in the respective time series. In this paper we use a bandwidth of 10% ($b = 0.1$).

The graphs in the remainder of this section use a dot-and-whisker format showing the mean with a 95% confidence interval, obtained by running the simulation with 100 different seeds. We have also slightly staggered the graphs along the horizontal axis to improve readability.

5.3 Quality of the groups

As a first comparison we look at the quality of the groups identified by mOverlay and our alternative approach. For both we use parameters that we have found to produce near optimal results. We set the maximum number of neighbors for mOverlay groups to eight and the number of parallel locating processes to five. For the Meridian-based approach we set the maximum number of verification nodes to five.

Figure 4 shows the mean round-trip times between group members for the grouping thresholds 1%, 2%, 5%, 10%, 20%, and 50% (using a logarithmic scale

for better readability). For both approaches a lower threshold also leads to smaller distances between group members. The effect is much bigger for mOverlay because the grouping threshold affects every iteration of the locating process, while the Meridian-based locating algorithm only uses the grouping threshold for its final step. Nevertheless, the round-trip times between group members of mOverlay are always bigger on the average than those of the Meridian-based approach. Moreover, the confidence intervals for mOverlay are bigger. We conclude that the Meridian-based approach performs better than mOverlay with respect to the first criterion.

The second aspect we examine is the average number of nodes per group. Figure 5 shows the group size for the same grouping thresholds as Figure 4. The groups identified by the alternate approach are bigger for grouping thresholds up to 10%. In contrast, mOverlay identifies much bigger groups with grouping thresholds above 10%, but this comes at the price of much greater round-trip times between group members. As expected, group sizes are rather small because of the wide distribution of the nodes.

If the identified groups shall be used as a basis for a distance estimation service they must also have a low out-of-band ratio. We look at this aspect using again the same parameters for grouping threshold and a 10% band for the out-of-band test. The results can be seen in Figure 6. The Meridian-based approach has shows a smaller out-of-band ratio than mOverlay for all grouping thresholds, and it shows less variance. Again, mOverlay shows high sensitivity towards the grouping threshold while the out-of-band ratio of the Meridian-based approach only increases slightly with growing grouping threshold.

5.4 Joining delay

In addition to a good quality of the identified groups it is also desirable to find the groups in the shortest time possible. We compare the two approaches using the same parameters as in Section 5.3. Figure 7 shows the joining delay per node for several grouping thresholds. Again, mOverlay proves to be much more sensitive towards the grouping threshold than the Meridian-based approach. Moreover, unless the grouping threshold is extremely high the alternate algorithm finds the local group much faster than mOverlay.

The joining delay of mOverlay nodes is not only sensitive to the choice of grouping threshold. Figure 8 shows the influence of the maximum number of neighbors per group and the number of parallel locating processes. Here we used a grouping threshold of 5% and a maximum of 2–10 neighbors per group. Furthermore, the three graphs show the effect of using 1, 5, or 10 parallel locating processes. We observe that a lower maximum of neighbors per group and fewer locating processes running in parallel cause a significant reduction in joining delay. Furthermore, the increase in joining delay appears to become smaller the more parallel locating processes we employ. However, regardless of the parameters the variance of the results is always quite large.

It appears that mOverlay can match the speed of the alternate approach if we reduce the number of parallel locating processes and the maximum number of

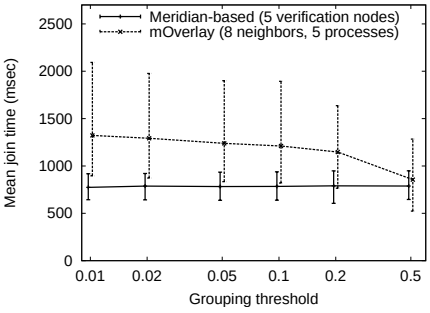

Figure 7. Mean joining delay per node for several grouping thresholds

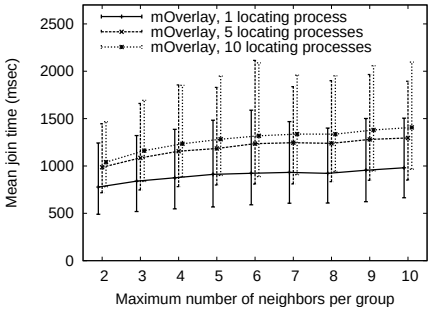

Figure 8. Mean joining delay in mOverlay for various parameters

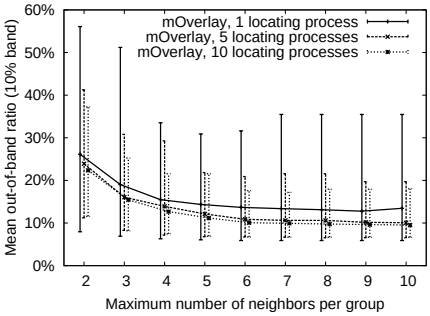

Figure 9. Mean out-of-band ratio in mOverlay for various parameters

neighbors per group. Nevertheless, the effect on the quality of the groups is severe as Figure 9 shows. Less than ca. 6 neighbors per group cause a significant increase in the out-of-band ratio. Using only one locating process also has a noticeable negative effect. On the other hand, the benefit from using more locating processes rapidly declines. The difference between five and ten parallel locating processes is mainly the size of the respective confidence intervals. Figure 9 also justifies our choice of parameters for mOverlay. The improvement for more than a maximum of eight neighbors per group and five locating processes is small while the increase in joining delay is still noticeable.

6 Conclusions

The locating algorithm of mOverlay has a number of drawbacks. We have presented an alternate algorithm that combines Meridian's closest node search with mOverlay's grouping criterion. In order to compare the mOverlay algorithm to the Meridian-based approach we have implemented simulators for each, using a black box network model based on PlanetLab measurements. We compare the

performance of both approaches based on the time it takes for a new node to find a group, the round-trip time between group members, the average number of nodes per group, and the suitability of the grouping for distance estimation, measured by the so-called out-of-band ratio.

From the simulation results we conclude that the Meridian-based locating algorithm is faster in most cases. It also identifies larger groups, and the nodes inside the groups are closer together than the nodes in mOverlay groups. Moreover, the groups identified with the alternate algorithm also have a smaller out-of-band ratio, which indicates better suitability for a distance estimation service.

References

1. X. Y. Zhang et al., "A construction of locality-aware overlay network: mOverlay and its performance," *IEEE JSAC*, vol. 22, no. 1, pp. 18–28, January 2004.
2. B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," ACM SIGCOMM'05, Philadelphia, Pennsylvania, USA, August 21–26, 2005.
3. Meridian C++ code, http://www.cs.cornell.edu/People/egs/meridian/code.php
4. M. Scheidegger, T. Braun, and F. Baumgartner, "Endpoint Cluster Identification for End-to-End Distance Estimation," ICC'06, Istanbul, Turkey, June 11–15, 2006, ISBN 1-4244-0355-3.
5. J. Stribling, "All-pairs-pings for PlanetLab," http://pdos.csail.mit.edu/~strib/pl_app.
6. random.org – True Random Number Service, http://www.random.org.
7. The Annotated Gnutella Protocol Specification, http://rfc-gnutella.sourceforge.net/developer/stable/index.html.
8. Clarke et al., "Protecting Free Expression Online with Freenet," IEEE Internet Computing, February 2002, pp. 40–49
9. S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network," SIGCOMM 2001, Aug 2001.
10. S. Ratnasamy, M. Handley, R. Karp and S. Shenker, "Topologically aware overlay construction and server selection," IEEE Infocom'02, New York, NY, June 2002.
11. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM Middleware 2001, Heidelberg, Germany, pp. 329–350, November 2001 .
12. L. Garcés-Erice, E. W. Biersack, and P. A. Felber, "MULTI+: Building topology-aware overlay multicast trees," in QofIS'04, September 2004.
13. M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," IEEE JSAC, Vol. 20, No. 8, pp. 1489–1499, 2002.
14. P. Francis, S. Jamin, J. Cheng, Y. Jin, D. Raz, and Y. Shavitt, "IDMaps: A global internet host distance estimation service," IEEE/ACM ToN, Vol. 9, No. 5, pp. 525–540, October 2001.
15. W. Theilmann and K. Rothermel, "Dynamic distance maps of the Internet," IEEE Infocom 2000.
16. T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," ACM IMC 2003.
17. M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," IPTPS 2003.
18. F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," ACM SIGCOMM 2004.