Topology-Aware Overlay Construction in Dynamic Networks

Rolf Winter, Thomas Zahn, Jochen Schiller
Institute of Computer Science
Freie Universität Berlin, Germany
{winter, zahn, schiller}@inf.fu-berlin.de

Abstract

Lately, peer-to-peer overlay networks and their ability to reflect the underlying network topology have been a focus in research. The main objective has been to reduce routing path lengths, stretched by the overlay routing process. In most solutions developed, a kind of fixed infrastructure in the form of so called landmarks or excessive message exchange are necessary to guarantee good overlay locality properties. Some solutions also deliberately give up even overlay ID distribution when constructing an overlay network with locality information.

This paper presents a topology-aware overlay network based on Pastry which does not rely on any fixed set of infrastructure nodes. Additionally, the approach presented here tries to construct the overlay with only little communication overhead and still tries to distribute overlay IDs as evenly as possible. Two bootstrap strategies were developed and analyzed, both explicitly designed to work in dynamic networks.

Keywords: peer-to-peer overlays, topological proximity, DHTs, dynamic networks

1 Introduction

Peer-to-peer (P2P) systems have recently seen a tremendous surge in popularity which has lead to the development of a variety of such systems. However, first-generation systems such as Gnutella [1] suffer from serious scalability problems [2]. Thus, current research efforts have been devoted to distributed hash tables (DHTs) to overcome these scalability obstacles.

DHTs are self-organizing overlay networks especially tailored towards the need of large-scale peer-to-peer systems. The general idea of DHTs is that each node participating in the (overlay) network is assigned a random ID. Each object that is to be stored on the network is also assigned a random ID. An object is now stored at the node whose ID is closest to the object's ID. All DHTs provide one basic operation: lookup(key) → node. Given an object's ID, a DHT is capable of locating the responsible node within a bounded amount of overlay routing steps. Prominent representatives of DHTs are CAN, Chord, Pastry, and Tapestry [3, 4, 5, 6].

In the overlay network, a node maintains an overlay routing table containing the IDs of a small set of other overlay nodes. Each such entry can be thought of as a virtual, direct link between the current node and the table entry. In overlay terms that means that messages can be exchanged directly between a node and the nodes in its routing table or, in other words, a node can reach all nodes in its routing table with a single overlay hop.

However, a single overlay hop is likely to involve multiple physical routing hops. For example, consider two overlay nodes A and B connected to the Internet. A is located in London and B in Chicago. It is quite obvious that even if B resides in A's overlay routing table, the one overlay hop between A and B would amount to several IP hops.

The main advantage of DHTs is that they provide a guaranteed bound on the number of overlay routing hops that have to be taken to locate any given object (i.e. any given key) on the overlay network. For [4, 5, 6] this bound is O(log N), where N is the number of nodes participating in the overlay network.[1] Due to the discrepancy between overlay hops and physical hops, as explained above, it is very likely that a significantly larger amount of physical hops compared to the logarithmic amount of overlay hops is involved in locating an object on the overlay network. With high probability, the following issue arises from this discrepancy: The number of physical hops induced by the overlay routing process can be decidedly greater than the direct physical routing path between the source node and the target node.

Consider the overlay routing example given in Figure 1. Overlay node S initiates a lookup that will eventually be routed to overlay node T. Since every overlay node only has very limited knowledge of other overlay nodes, nodes usually try to forward a lookup request to other nodes that are closer (in terms of the overlay ID space) to the key than they are themselves.[2] In this example, three intermediate overlay routing steps are involved until the request reaches its final destination, clearly traveling a highly suboptimal physical route.

As can be seen, although the target node can be located with logarithmic overlay hops, the physical path traveled during the overlay routing process is often less than optimal. More technically speaking, the ratio between the number of physical hops induced by overlay routing and

[1] CAN employs a more general approach involving d-dimensional virtual coordinate spaces. CAN has a routing effort of $O(d(n^{1/d}))$. However, if the number of dimensions d is chosen to be $d=(\log_2 n)/2$, CAN achieves the same effort as Chord, Pastry and Tapestry.

[2] Exactly how many nodes an overlay node knows about and how message forwarding is done, is an implementation-specific detail of the respective DHT.

the number of physical hops on a direct physical routing path is often markedly lopsided.

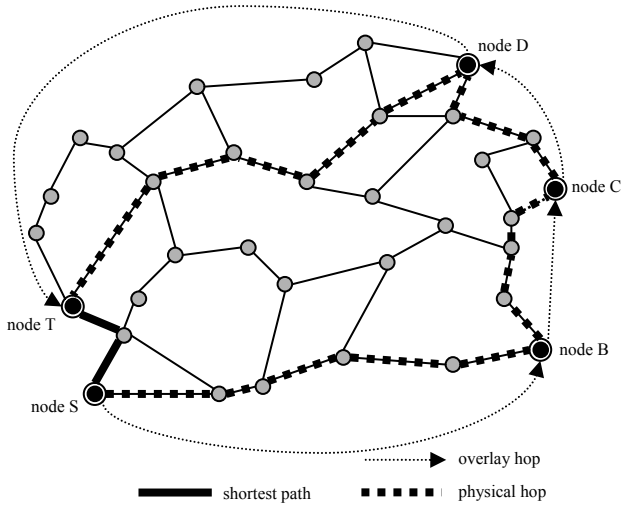


Figure 1: Overlay routing

The main contribution of this paper is the design and analysis of two approaches that optimize this ratio. Most importantly, these two approaches have been designed to maintain optimized routing properties even in the presence of network dynamics such as frequent node failures (i.e. network degression). Thus, they are well suited for highly dynamic networks, such as ad-hoc networks.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we present in detail our two approaches. Section 4 analyzes and evaluates various experimental results achieved with these approaches. Section 5, concludes this paper and gives a brief outlook on our future work.

2 Related Work

A significant amount of work has been dedicated to the development of P2P overlay networks, but so far only few approaches explicitly focus on making overlay networks reflect the locality properties of the underlying physical networks.

One of the general concepts used to close the gap between physical and overlay node proximity is landmark clustering. Ratnasamy et al. [7] use landmark clustering in an approach to build a topology-aware CAN [3] overlay network. They require a fixed set of landmark nodes that all participating nodes have to know about. Prior to joining the overlay network, a joining node has to measure its distance (e.g., RTT, hop count, or any other appropriate metric) to each landmark. The node then orders the landmarks according to its distance measurements. Nodes with the same such landmark ordering fall into the same bin. The intuition behind this idea is that nodes with the same landmark ordering, i.e. nodes that have similar distances to all landmark nodes, are also quite likely to be close to each other topologically. Each bin is now mapped to a region in CAN's virtual coordinate space. After having *binned* itself, a joining node assumes a random point in the region associated with its bin. An immediate issue with landmark binning is that it can be rather coarse-grained depending on the number of landmarks used and their distribution. Furthermore, a fixed set of landmarks renders this approach unsuitable for dynamic networks, such as ad-hoc networks. The most significant downside of this approach, however, is that it can lead to an extremely uneven overlay ID distribution. This means that a small set of nodes could be responsible for a very large part of the ID space, essentially turning them into hot spots. Xu et al. [8] have verified this in their study.

[8] presents a method to fine-tune landmark binning for the construction of overlay networks. They introduce maps containing information on close-by nodes in a specific regions to allow nodes to join the overlay network with a more accurate reflection of its own position in the physical network. A map is stored as global soft state among the nodes of a region. This approach, however, comes with a significant overhead. Potentially, there could be a very large number of regions (e.g., in Pastry such a region is considered to be a set of nodes sharing a certain ID prefix), all of which have to maintain their map. Moreover, to achieve a finer granularity, additional inner-bin measurements are required.

Waldvogel and Rinaldi [9] propose an overlay network (Mithos) that focuses on reducing routing table sizes. Mithos also tries to establish overlay locality from physical network proximity. A new node is assigned an overlay ID based on the IDs of its (physical) neighbors. They employ virtual springs to make the ID fit into the neighborhood range. In order to avoid local minima, substantial probing has to be undertaken. Unfortunately, only very small overlay networks (200 – 1000 nodes) are used for simulations and the impact of network degression is not considered.

As a DHT, Pastry [5] uses certain heuristics to exploit physical network proximity in its overlay routing tables. In a thorough analysis [10], Castro et al. examine the impact of various network parameters and node degression on Pastry's locality properties. Unlike the other approaches presented, Pastry does not construct its overlay structure from the underlying physical network topology. Instead, Pastry distributes its node evenly in the overlay ID space regardless of the actual physical topology. One way in which Pastry tries to exploit physical proximity is that a new node should bootstrap itself using a node close-by. During the join process, it then tries to choose among the candidate nodes for a particular routing table entry a node that is "close" to itself. During its lifetime, a node periodically performs routing table maintenance and improvement by asking other nodes for "better" routing table entries. Obviously, those are mere heuristics and, therefore, Pastry does not guarantee optimal routing table states.

3 Random Landmarking and Closest Neighbor Prefix Assignment

As mentioned previously, our approaches actively exploit physical proximity in the creation of overlay networks. Their main focus is on achieving good locality properties in the overlay network by inducing as little construction and maintenance overhead as possible while still maintaining an even overlay ID distribution. This will translate into an optimized overlay vs. physical routing distance ratio, which is particularly crucial in dynamic networks. Maintaining an even overlay ID distribution is especially important in ad-hoc networks with extremely heterogeneous devices where devices with scarce resources should not become hotspots.

The implementation of our approaches is based on a Pastry overlay network. Pastry is a very well-known DHT that provides built-in locality heuristics. We chose Pastry because these heuristics – as mentioned above – have been thoroughly analyzed [10]. This analysis makes a good background against which to compare the experimental results achieved with our approaches. However, we believe that their mechanisms are DHT-independent and could, thus, be ported to other DHTs.

Our approaches differ primarily from Pastry's approach by the way in which overlay IDs are assigned. Pastry's overlay construction basically works in a top-down fashion, i.e. Pastry randomly assigns overlay IDs regardless of the underlying topology. It, then, tries to make the physical proximity fit into the overlay routing state through the join process and table maintenance. In contrast, we construct the overlay network in a bottom-up fashion, i.e. the overlay is built considering locality information from the underlying network. Before a node joins the overlay, it gathers information concerning its physical neighborhood and uses it to assign itself an appropriate overlay ID. Two approaches are examined in this context: *random landmarking* (RLM) and *closest neighbor prefix assignment* (CNPA).

To analyze the different effects of Pastry's and our approaches, at this point Pastry's overlay routing is discussed briefly (for a thorough discussion see [5, 10]). Each Pastry node essentially maintains a routing table and a leaf set. The routing table consists of a number of rows equal to the number of digits in an overlay ID and a number of columns equal to the ID base. From row to row, the matching prefix between the current node's ID and the row's entries increases by one. The leaf set contains the numerically closest nodes to the current node regardless of physical proximity. When a node has to forward a lookup, it first checks whether the requested ID is covered by its leaf set and forwards the lookup directly to the corresponding leaf. Otherwise, it uses its routing table to identify a node that has a matching prefix with the requested key that is one digit longer than the current node's matching prefix. This process continues until the node numerically closest to the requested ID is located. Intuitively, this approach allows Pastry to locate a node responsible for a certain key with logarithmic effort because in each routing step the matching prefix length is likely to be increased by one.

Since the prefix increases by one from routing table row to routing table row, there are also exponentially less candidates with which to fill a routing table entry as the row number increases. In Pastry, this leads to the effect (see [5, 10]) that from overlay routing step to overlay routing step the physical distance between nodes is likely to increase. Thus, the last routing step tends to dominate the overall physical routing path length of a key lookup.

Since the last overlay routing step is usually taken from the leaf set, with our approaches this routing step is likely to be close. This is because leaf set entries are numerically closest to the current node, and thus they are also likely to be physically close to the current node due to our ID assignment strategies. In other words, our approaches promise to optimize the "last mile" of the overlay routing process.

3.1 Random Landmarking

Conventional landmarking, as introduced in [7, 8], suffers from the limitation that it assumes a set of fixed, stationary landmark nodes. All overlay nodes are expected to know the landmark nodes and to measure their respective distances to those landmarks. This, obviously, reintroduces the client-server concept into the bootstrap process. Especially in networks where nodes are expected to fail frequently, there are usually no sets of fixed nodes available, which renders this approach infeasible. Therefore, we introduce random landmarking (RLM) into the overlay construction process.

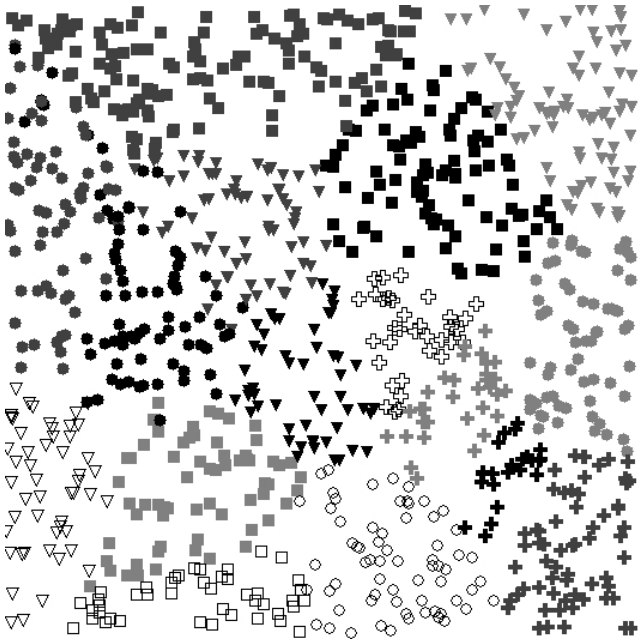


Figure 2: Prefix distribution as generated by RLM. Equal symbols and colors represent equal prefixes.

RLM utilizes the overlay lookup capabilities to locate overlay nodes responsible for a fixed set of landmark keys

(overlay IDs). These nodes serve as temporary landmarks for a joining node. It is important to understand that the keys have to be chosen in a way that they divide the overlay ID space into equal portions. For example, in a network with an ID base of 16, an appropriate set of landmark keys would be: 000..00, 100..00, 200..00, …, F00..00. The joining node then measures the distances to those temporary landmarks and assigns itself an ID based on its landmark ordering. The advantage of this approach is that "landmark nodes" can fail and others will simply step in as Pastry will automatically redirect future key lookups to those nodes now responsible for the landmark keys. After having measured its landmark distances, the joining node adopts an ID prefix of a certain length from the landmark node closest[1] to itself. The ID remainder can be assigned randomly or can be based on an algorithm that further takes into account the physical neighborhood. The length of the ID prefix that the new node shares with its closest landmark node can be determined using the following formula:

$$prefix\ length = \lfloor \log_b k \rfloor$$

where b is the ID base and k the number of landmark keys. As can be seen, the number of landmark keys should preferably equal a power of b.

This approach has the following effects. First of all, it leads to physically close nodes forming regions with common ID prefixes, which means these nodes are also likely to be numerically close to each other in the overlay ID space, as can be seen in figure 2. This, in turn, leads to the desired effect that a node's leaf set is likely to reference physically close nodes (bear in mind that the leaf set of a node contains the numerically closest nodes). Since the leaf set is normally utilized for the last routing step, that step is likely to travel a short physical distance. Note that there are still less and less candidate nodes to choose from to fill a certain overlay routing table entry as the row number increases, but with our approaches the likelihood of these candidates being physically close to the current node also increases from row to row.

Special care has to be taken when a network is first created from scratch. To prevent temporary landmark nodes from being located too close to each other in the underlying network, the notion of a *landmark gravitation range* is introduced. If a new node discovers during its landmark measurement process that a temporary landmark node is responsible for a landmark key with which it shares no common prefix – i.e. that landmark must, therefore, be responsible for more than one landmark key – the new node should make itself a new landmark. However, it will only do so if its physical distance to *any* other landmark node exceeds a certain threshold, the landmark gravitation range. Again, various distance metrics are conceivable. The gravitation range is only a measure of reassurance that no physical landmark clusters form. It is therefore only significant during the initial network build-up because after all landmark keys are properly covered, this process ceases to have any importance.

To make the whole landmarking process more lightweight and efficient, a node obtains from its bootstrap node a list of the landmarks that the bootstrap node itself had used when it first joined the network. The idea here is that those "old" landmarks could still be valid. Thus, the new node is spared to initiate lookups for all landmark keys. The joining node now measures the distances to its inherited landmarks. When one of these landmarks receives the measure request from the joining node, it checks whether it is still responsible for the corresponding landmark key. If it is not, it will signal so in its measure response. Only in this case will the joining node reinitiate a landmark key lookup. Afterward, it will measure its distance to the proper landmark. This can reduce the overall bootstrap traffic significantly.

3.2 Closest Neighbor Prefix Assignment

Obviously, RLM will drive more node bootstrap / join traffic to those nodes temporarily responsible for a given landmark key. Although RLM is likely to cause only marginal overhead, in some network settings it can be desirable to be able to join the network without such additional overhead. Closest Neighbor Prefix Assignment (CNPA) was designed explicitly for such situations.

CNPA takes advantage of Pastry's specification that a new node should always bootstrap itself using the physically closest neighbor. After having identified its closest neighbor employing methods such as expanding ring search, hill climbing strategies, etc., a new node assumes the ID prefix of that neighbor. The remainder of its ID can be determined in the same fashion as described with RLM. The prefix length which has to be adopted by a new node is expected to be known by all nodes.

With CNPA the problem of the initial network construction is solved slightly differently. Since this approach employs no landmarks, a balanced ID prefix distribution is achieved using another strategy. If the node is close to its bootstrap node according to a metric similar to RLMs gravitation range, it only then assumes the bootstrap node's ID prefix and generates the rest of the ID as mentioned previously. According to Pastry's join specifications, a new node always receives the first row of its routing table from its bootstrap node. If the node is further away from the bootstrap node than the threshold mentioned above, it would also query for landmark prefix length - 1 more table rows. Taking advantage of this, the joining node inspects these inherited rows for empty entries. Intuitively, an empty row entry hints at the fact that there is no node on the overlay having the landmark ID prefix. The joining node now assumes the missing ID prefix for itself, generates the rest of its ID randomly and then goes through Pastry's usual join procedure. As in RLM, this process is only significant during the initial network build-up.

[1] Conceivable metrics include hop count, RTT etc.

The effects of CNPA are exactly the same as with RLM. CNPA, though, induces less overhead at the expense of being more coarse-grained.

4 Experimental Results

In order to examine the behavior and performance of our approaches, we simulated both a Pastry and a network utilizing both our approaches with the discrete event simulator Omnet++ [11]. Since we wanted to especially evaluate the overlay behavior and resilience in extremely dynamic networks, we chose to run Pastry and our overlays in ad-hoc scenarios employing an AODV [12] physical routing layer.
To put our simulation results into perspective, we implemented a Pastry reference overlay in Omnet++ in strict conformance with the Pastry papers [5, 10]. Although these papers describe simulation results in detail, we implemented Pastry to have a reference basis which can be exposed to exactly the same networking conditions as our implementations. Additionally, the result of our Pastry implementation can be compared to the paper results to give confidence in the correctness of our implementation details.
We evaluated our approaches in two main network settings:

- static networks
- networks with degression

In a static network, the initial network topology remains unchanged over the entire course of a simulation run. In networks with degression, random nodes leave and join the network with a certain rate. As our initial physical topology, we chose a plain where nodes are distributed randomly and with a certain density.
For each network setting and scenario, we conducted multiple simulation runs.

4.1 Static Networks

The first set of simulations were run in order to verify the correctness of our Pastry reference implementation and to, thereby, create a background against which to compare our results. For our simulations, we considered randomly distributed plain topologies with 1,000, 2,000, 5,000, and 10,000 participating nodes. All participating nodes form an underlying ad-hoc network. The average node connectivity is about 14, i.e. on average each node is within the transmission range of 14 other nodes. Furthermore, each physical node also participates in the overlay network. During a simulation run, 20,000 random key lookups are initiated by randomly picked overlay nodes.
We examined various Pastry bootstrap mechanisms. As a lower bound, we implemented an artificial bootstrap procedure where we used global knowledge to fill all overlay routing tables. This means that for each routing table entry the physically closest candidate is always known and chosen. However, global knowledge is an absolutely unrealistic assumption and, thus, this was only utilized to be able to compare further results to Pastry's theoretical best state in our scenarios.
We also examined a bootstrap mechanism that uses Pastry's standard join procedure. According to [10], after a new node has bootstrapped itself, it sends the nth row of its routing table to each entry in that row. These entries, then, update their own routing tables. This optimization serves both to propagate information about newly joined nodes and to avoid cascading routing table inefficiencies. Obviously, it also induces a hefty network overhead.

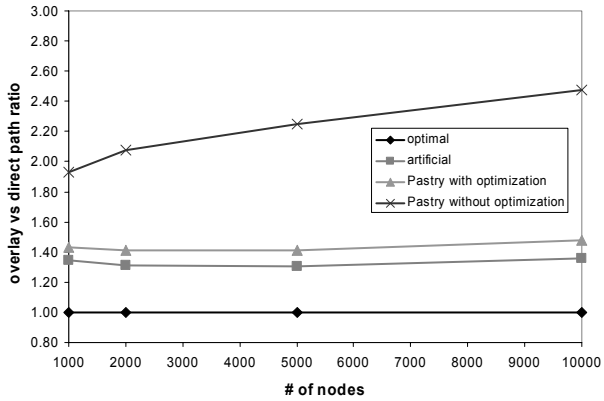


Figure 3: Overlay vs. direct path ratio of the various Pastry bootstrap mechanisms.

To study what a Pastry network without any such overhead performs like, we also implemented a bootstrap mechanism that does not try to optimize the routing tables after node arrivals. With this mechanism, Pastry's locality properties have to rely on the mere heuristic embedded in its join process. This approach can be viewed as the upper bound on Pastry's performance in our scenarios.
Figure 3 shows the average ratio between the number of physical hops induced by an overlay lookup and the direct physical routing path between the source node and the target node for the Pastry bootstrap mechanisms mentioned above. As expected, the results correlate directly with the original Pastry results in [5, 10]. When global knowledge is applied during the bootstrap process, Pastry can achieve a ratio of around 1.33 on average. In the more practical case of Pastry's original bootstrap strategy, an average ratio of 1.45 is achieved. With no optimization, the ratio rises to around 2.47 as the number of participating nodes increases.
Figure 4 depicts the total number of messages that have to be exchanged among all nodes during the bootstrap process in order to build up and optimize the overlay routing tables. Obviously, these different message efforts cause the varying ratios between Pastry with and without optimization as displayed in Figure 3.
As can be seen, Pastry's bootstrap optimization introduces a significant overhead. With optimization, 6 to 7 times more messages have to be exchanged compared to Pastry's bootstrap without optimization. These messages include join requests and forwards, distance measurements, and messages containing routing table state information. In a

network of 10,000 nodes, 6.88 million messages have to be processed in order to optimize the overlay routing tables so that a ratio of 1.47 can be achieved. Bear in mind that without such optimization the ratio deteriorates to 2.47. Note that there is no data about the artificial bootstrap mechanism. This is because in this case the routing tables were not constructed using the actual overlay join procedure, but instead all entries were artificially selected employing global knowledge.

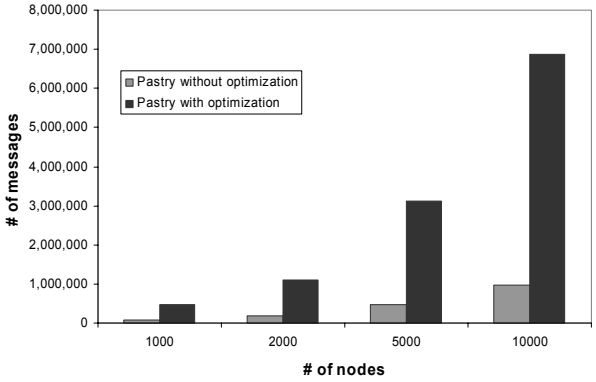


Figure 4: Total number of messages exchanged during bootstrap (Pastry).

Next, we used the same network settings (static, random plain topologies of sizes 1000, 2000, 5000, and 10,000) to evaluate the performance of our approaches in static networks. We considered 4 different approaches: RLM, RLM with bootstrap optimization, CNPA, and CNPA with bootstrap optimization. For RLM we used 16 landmark keys.

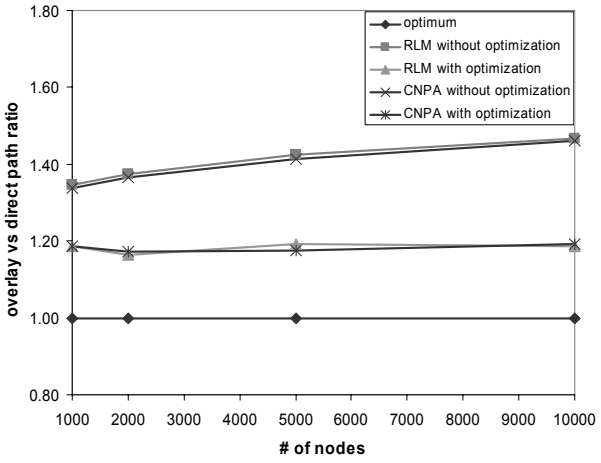


Figure 5: Overlay vs. direct path ratio with RLM and CNPA

Figure 5 shows the overlay vs. direct path ratios achieved with our approaches. As can be observed, both approaches achieve better or equal ratios in all tested networks *without* any optimization than Pastry does *with* its optimization. If they also utilize the same bootstrap optimization as Pastry, both RLM and CNPA gain a ratio of 1.19, which is significantly lower than the best possible ratio that Pastry can only score when artificially bootstrapped. This is due to the fact that we construct the overlay network exploiting physical proximity directly whereas Pastry assigns its IDs oblivious to the physical topology. Pastry can only subsequently try to optimize its routing tables to reflect physical proximity as best as possible to supplement its built-in heuristics.

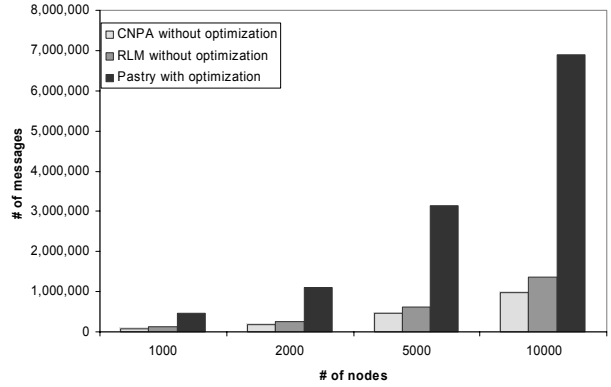


Figure 6: Total number of messages exchanged during bootstrap (Pastry, RLM, CNPA).

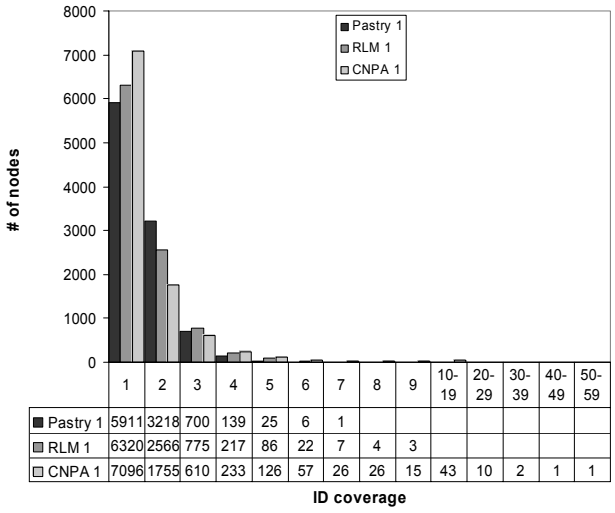
Again, one of the main goals was to achieve those ratios with as little overhead as possible. Figure 6 presents RLM's and CNPA's costs in terms of the total number of messages exchanged. As already observed, both our approaches perform comparable to Pastry with optimization, but they produce *significantly* less overhead. For example, in a network of 10,000 nodes, Pastry has to exchange 5 to 7 times more messages to obtain the same ratio (1.42) as RLM and CNPA do without optimization (6.88 million compared to 1.35 and 0.98 million, respectively).

If one focused on optimizing the ratio instead of the network traffic, RLM and CNPA could also utilize bootstrap optimization. In this case, their advantage in terms of the message total would be given up deliberately with the effect of lowering the factor to 1.19, as described above.

Another crucial issue with overlay networks is the equal distribution of overlay ID ranges that an individual node is responsible for, i.e. the number of overlay IDs that every node covers. Clearly, in a perfectly distributed overlay network, each node would be responsible for an ID range of the size of the total ID space divided by the number of participating nodes, the *optimal ID range*. Figure 7 shows the average ID coverage distribution in a 10,000 node Pastry, RLM, and CNPA overlay network.

In true DHT spirit, Pastry achieves a very well distributed ID coverage. On average, 5,911 nodes cover the optimal ID range, 3,318 nodes are responsible for twice the optimal ID range, and so forth, and only a single node is responsible for 7 times the optimal ID range. As can be seen, RLM manages to retain a comparable distribution.

6,320 nodes cover the optimal ID range, 2,566 are responsible for twice the optimal ID range, and so forth. At its extreme, merely 4 nodes have to handle 8 times more IDs than optimal, and only 3 nodes cover 9 times the optimal ID range.



	1	2	3	4	5	6	7	8	9	10-19	20-29	30-39	40-49	50-59
Pastry 1	5911	3218	700	139	25	6	1							
RLM 1	6320	2566	775	217	86	22	7	4	3					
CNPA 1	7096	1755	610	233	126	57	26	26	15	43	10	2	1	1

ID coverage

Figure 7: Average overlay ID coverage distribution in a 10,000 node network (Pastry, RLM, CNPA).

On the other hand, CNPA does not perform quite as well. The vast majority of nodes appears well distributed, but at its extreme a single node covers more than 50 times the optimal ID range. It is, thus, likely to attract 50 times more traffic than in an ideal overlay network. This is the trade-off of CNPA's lower message overhead in achieving the same ratio as RLM. Both RLM and CNPA obtain equal ratios, but RLM induces a higher message total whereas CNPA sacrifices a perfectly equal ID distribution.

4.2 Networks with Degression

In a next step, we evaluated the performance of Pastry, RLM, and CNPA in networks with degression. For all test runs, we used 5,000-node networks. In the first set of simulations, randomly selected 40% of all nodes leave the network and the same amount of new nodes join the network at random. In the second set, the degression rate is 100%, i.e. the entire network topology is changed once before the 20,000 random lookups are issued. The focus of these experiments is to analyze how the different approaches deal with network degression *without* inducing any additional overhead. Therefore, in all scenarios, none of the joining nodes employed bootstrap optimization in order to keep the network traffic as low as possible.
The Pastry networks were always bootstrapped artificially, but the new nodes joined at random using the standard Pastry join procedure without bootstrap optimization. In the RLM networks, the nodes also failed and joined the network at random, but extra care was taken to ensure that one in every 10 failing nodes was a temporary landmark node. Figure 8 depicts the overlay vs. direct routing path ratios as maintained in the different degression scenarios.

As can be observed, without bootstrap optimization for the new nodes, Pastry's ratio that was artificially lowered to 1.35 deteriorates significantly with both degression rates. As expected, with 100% degression, Pastry's ratio reaches the level (2.3) that Pastry without optimization achieves in static networks (see figure 3). For mild degression rates (40%), CNPA slightly outperforms RLM as it does not have the problem of frequently changing landmark nodes. However, as the degression rate increases (100%), CNPA's ID assignment becomes too coarse-grained and its ratio deteriorates markedly. On the other hand, RLM maintains a ratio clearly below 2.

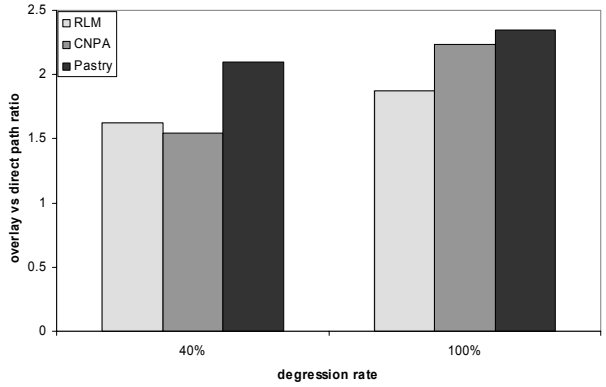


Figure 8: Overlay vs. direct routing path ratios with different degression rates (without bootstrap optimization).

5 Conclusion & Future Work

In this paper we have analyzed two approaches designed to construct topology-aware overlay networks. Our goal was to reduce the ratio between overlay routing path lengths and the length of direct physical routing paths. Compared to Pastry, the basis for our approaches, we were able to achieve comparable ratios with significantly less communication effort. Up to 5 to 7 times more messages have to be exchanged using Pastry compared to RLM and CNPA achieving the same ratio. On the other hand, when adding the same optimization effort as with Pastry, our approaches were able to achieve a ratio decidedly lower than the theoretical Pastry optimum. Furthermore, we were able to preserve an even ID distribution employing RLM or at least being close to such a distribution using CNPA. In networks with mild degression rates, we showed that without any overhead our two approaches can keep up a better ratio than Pastry. RLM can retain a good ratio even in highly dynamic networks.

The next evaluations of our approaches will include different network topologies, parameters and settings. Another focus of our work is the consideration of mobile nodes common in ad hoc networks. We also want to examine the effect of a combination of RLM and CNPA on the path length ratio and overlay ID distribution and to extend our study to include even more degression rates and to evaluate overlay ID reassignment strategies.

References

[1] The Gnutella Protocol Specification v0.4.www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[2] J. Ritter. Why Gnutella Can't Scale. No, Really. http://www.darkridge.com/~jpr5/doc/gnutella.html

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In Proc. of ACM SIGCOMM, Aug. 2001.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. of ACM SIGCOMM, Aug. 2001.

[5] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In International Conference on Distributed Systems Platforms (Middleware 2001). Nov. 2001.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U.C. Berkeley, April 2001.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In IEEE Infocom '2002.

[8] Z. Xu, C. Tang, and Z. Zhang. Building Topology-Aware Overlays using Global Soft-State. In ICDSC'2003, May 2003.

[9] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. HotNets 2002, SIGCOMM/CCR 2003.

[10] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting Network Proximity in Peer-to-Peer Overlay Networks. International Workshop on Future Directions in Distributed Computing (FuDiCo), June 2002.

[11] Omnet++. Discrete Event Simulation System. http://www.omnetpp.org

[12] Ad hoc On-Demand Distance Vector Routing. http://www.ietf.org