Telecommunication Systems 27:2–4, 321–345, 2004

DynaMO: A Topology-Aware P2P Overlay Network for Dynamic, Mobile Ad-Hoc Environments

ROLF WINTER, THOMAS ZAHN and JOCHEN SCHILLER {winter;zahn;schiller}@inf.fu-berlin.de
Institute of Computer Science, Computer Systems and Telematics Group, Freie Universität Berlin, Germany

Abstract. Due to the key differences between wired and ad-hoc wireless networks, traditional networking services and techniques are not always easily portable from an infrastructure based network to a wireless environment. One of the most prominent examples is the TCP transport protocol, which performs only poorly in wireless ad-hoc networks. The Peer-to-Peer (P2P) overlay networks recently developed all target the Internet where a lot of performance issues can be neglected or can be completely ignored. In addition, assumptions made for infrastructure based networks cannot be made in an ad-hoc environment, such as a fixed set of nodes which are always available. This article presents a P2P network tailored towards mobile ad-hoc environments. It utilizes proximity information to efficiently generate an overlay structure which reflects the underlying physical network topology. This way, physical routing path lengths stretched by the overlay routing process are reduced. As a novelty it does not rely on a fixed set of nodes and adapts to changes in the physical network topology. A prominent property of the overlay construction process is that the communication overhead is reduced to a minimum. Additionally, the P2P network presented maintains an even Overlay ID distribution which is deliberately given up by some solutions previously developed for wired networks. The basis of this new overlay network is Pastry, a P2P substrate based on the concept of a distributed hash table. Two different bootstrap strategies were developed and analyzed, both explicitly designed to work in dynamic and mobile networks such as ad-hoc networks.

Keywords: peer-to-peer, ad-hoc networks, mobility, DHTs, topological proximity

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 [10] suffer from serious scalability problems [Ritter, 7]. 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 toward 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 on the node whose ID is closest to the object's ID. All DHTs provide one basic operation: `lookup(key)` $\rightarrow$ `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 [Ratnasamy et al., 5; Stoica et al., 9; Rowstron and Druschel, 8; Zhao et al., 13].

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 though 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 Los Angeles. 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 [Stoica et al., 9; Rowstron and Druschel, 8; Zhao et al., 13] 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 the number of physical hops on a direct physical routing path (in the following referred to as overlay stretch) is often markedly lopsided.

This article describes the design and analysis of two approaches that optimize the overlay stretch. Most importantly, these two approaches have been designed to maintain optimized routing properties even in the presence of network dynamics such as frequent

[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.

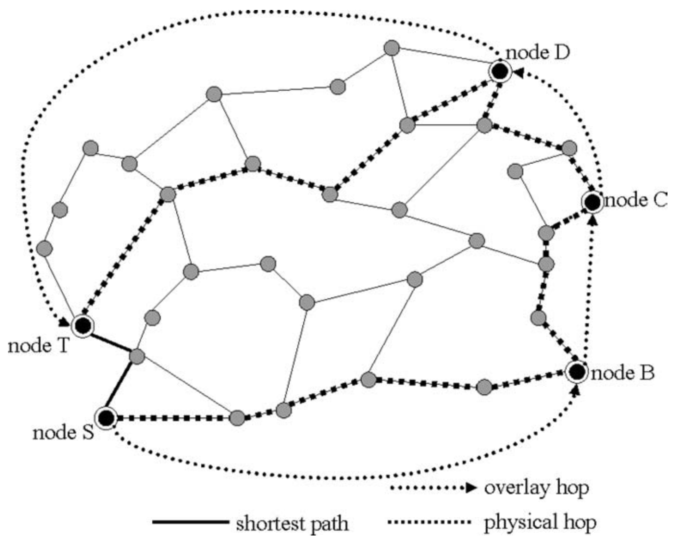


Figure 1. Overlay routing.

node arrivals and departures (i.e. network churn) and, especially, node mobility. Thus, they are well suited for highly dynamic networks, such as ad-hoc networks.

The remainder of this article is organized as follows. Section 2 discusses related work. In section 3, we present in detail the two approaches taken by DynaMO. Section 4 analyzes and evaluates various experimental results achieved with DynaMO. The fifth section describes application scenarios and shows how to efficiently utilize DynaMO's unique characteristics. Section 6 concludes this article 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. Furthermore, none of the related approaches consider node mobility.

One of the general concepts used to close the gap between physical and overlay node proximity is landmark clustering. Ratnasamy et al. [6] use landmark clustering in an approach to build a topology-aware CAN [Ratnasamy et al., 5] 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. [12] have verified this in their study.

Xu et al. [12] present 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 [11] 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 churn is not considered.

As a DHT, Pastry [Rowstron and Druschel, 8] uses certain heuristics to exploit physical network proximity in its overlay routing tables. In a thorough analysis, Castro et al. [1] examine the impact of various network parameters and network churn 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. DynaMO

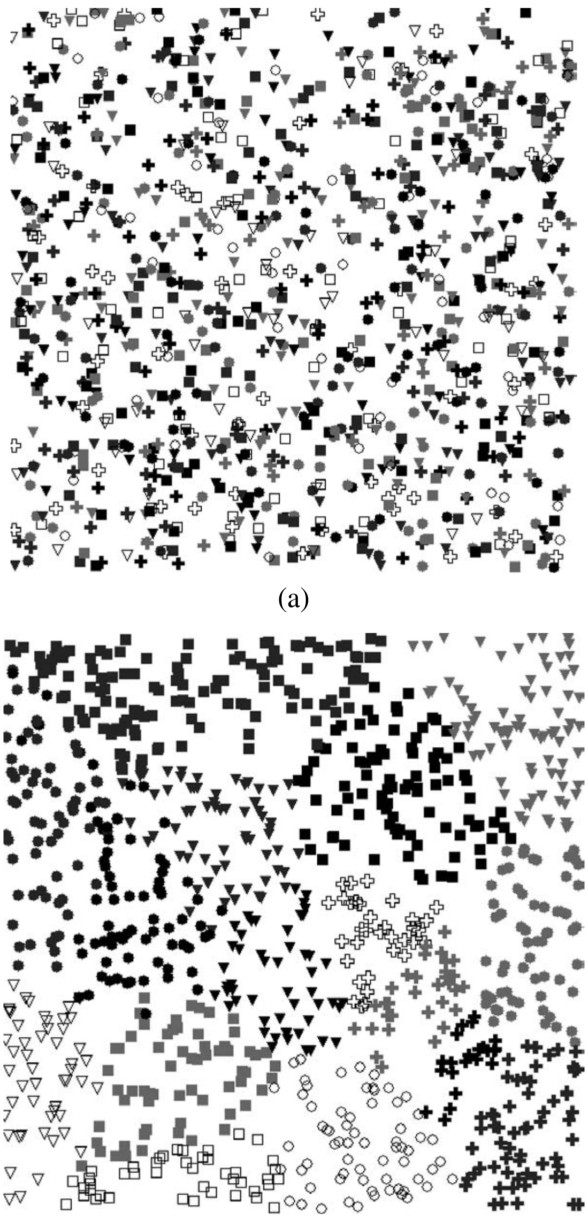
As mentioned previously, DynaMO actively exploits physical proximity in the creation of overlay networks. DynaMO's main focus is on achieving good locality properties in the overlay network by inducing as little construction and maintenance overhead as possible while still maintaining an even overlay ID distribution. This will translate into an optimized overlay stretch, 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.

Our implementation of DynaMO 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 [Castro et al., 1]. This analysis makes a good background against which to compare the experimental results achieved with DynaMO. However, we believe that DynaMO's mechanisms are DHT-independent and could, thus, be ported to other DHTs.

DynaMO's 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. This is reflected in the total randomness of the spatial overlay ID distribution, as depicted in figure 2(a). It, then, tries to make the physical proximity fit into the overlay routing state through the join process and table maintenance. In contrast, DynaMO constructs 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 DynaMO's approaches, at this point Pastry's overlay routing is discussed briefly (for a thorough discussion see [Rowstron and Druschel, 8; Castro et al., 1]). 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 [Rowstron and Druschel, 8; Castro et al., 1]) 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.


(a)

(b)

Figure 2. Spatial prefix distribution as generated (a) by Pastry and (b) by RLM. Equal symbols and colors represent equal prefixes.

Since the last overlay routing step is usually taken from the leaf set, with DynaMO 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 DynaMO's ID assignment strategies. In other words, DynaMO promises to optimize the "last mile" of the overlay routing process.

3.1. Random landmarking

Conventional landmarking, as introduced in [Ratnasamy et al., 6; Xu et al., 12], 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, DynaMO introduces random landmarking (RLM) into the overlay construction process.

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[3] 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:

$$\textit{prefix length} = \lfloor \log_b k \rfloor,$$

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(b). 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 DynaMO the likelihood of these candidates being physically close to the current node also increases from row to row.

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

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 nodes 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.

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 DynaMO's behavior and performance, we simulated both a Pastry and a DynaMO network using the discrete event simulator Omnet++ [3]. Since we wanted to especially evaluate the overlay behavior and resilience in extremely dynamic networks, we chose to run Pastry and DynaMO overlays in ad-hoc scenarios employing an AODV [Perkins, 4] physical routing layer.

To put DynaMO's simulation results into perspective, we implemented a Pastry reference overlay in Omnet++ in strict conformance with the Pastry papers [Rowstron and Druschel, 8; Castro et al., 1]. 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 DynaMO implementation. Additionally, the result of our Pastry implementation can be compared to the paper results to give confidence in the correctness of DynaMO's implementation details.

We evaluated DynaMO in three main network settings:

- static networks,
- networks under churn,
- mobile networks.

In a static network, the initial network topology remains unchanged over the entire course of a simulation run. In networks under churn, random nodes leave and join the network at a certain rate. In mobile networks, the set of participating nodes remains the same, but nodes are constantly changing their physical position. As our initial physical topology, we chose a plane 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 DynaMO's results. For our simulations, we considered randomly distributed plane 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 [Castro et al., 1], 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.

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 overlay stretch, which is the 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 [Rowstron and Druschel, 8; Castro et al., 1]. When global knowledge is applied dur-

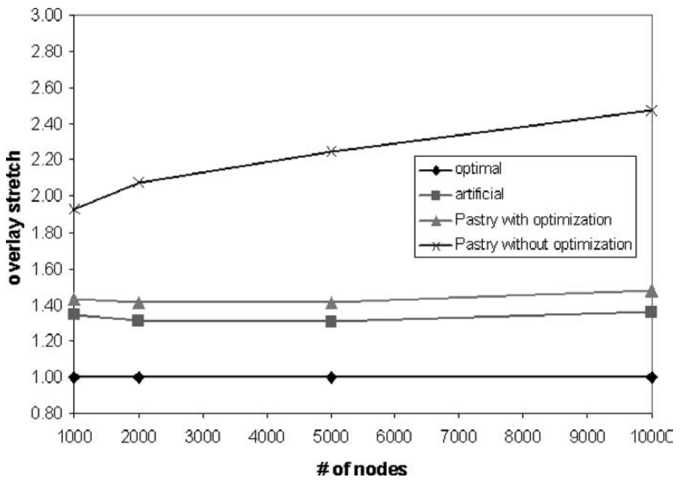


Figure 3. Overlay stretch of the various Pastry bootstrap mechanisms.

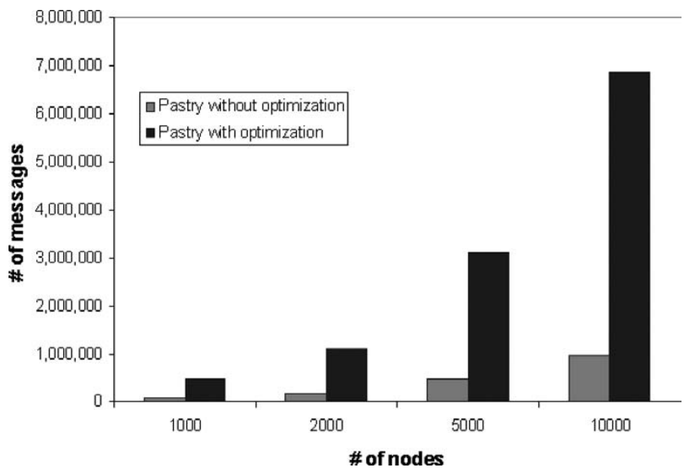


Figure 4. Total number of messages exchanged during bootstrap (Pastry).

ing the bootstrap process, Pastry can achieve a stretch of around 1.33 on average. In the more practical case of Pastry's original bootstrap strategy, an average stretch of 1.45 is achieved. With no optimization, the overlay stretch 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 overlay stretches 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 stretch of 1.47 can be achieved. Bear in mind that without such optimization the overlay stretch 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.

Next, we used the same network settings (static, random plane topologies of sizes 1000, 2000, 5000, and 10,000) to evaluate DynaMO's performance 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. This means that a new node "inherits" its first ID digit from its closest temporary landmark node whereas its ID suffix is assigned completely randomly.

Figure 5 shows the overlay stretches achieved with DynaMO's approaches. As can be observed, both DynaMO approaches achieve better or equal stretches in all tested networks *without* any optimization than Pastry does *with* its optimization. If DynaMO also

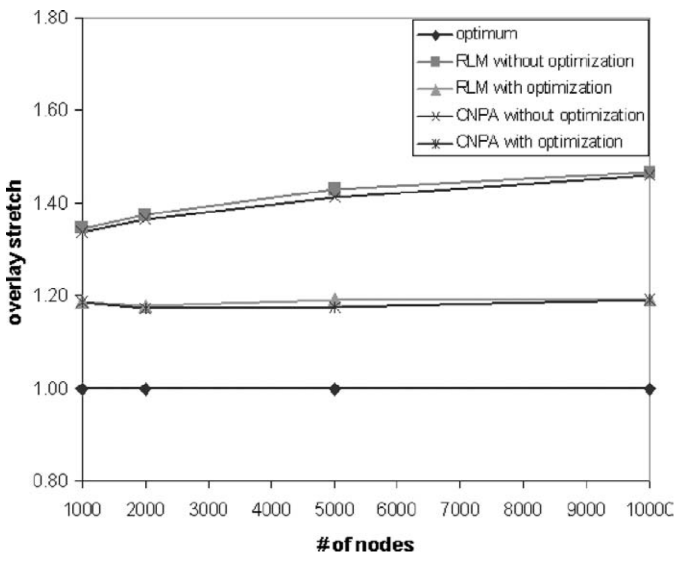


Figure 5. Overlay stretch with DynaMO.

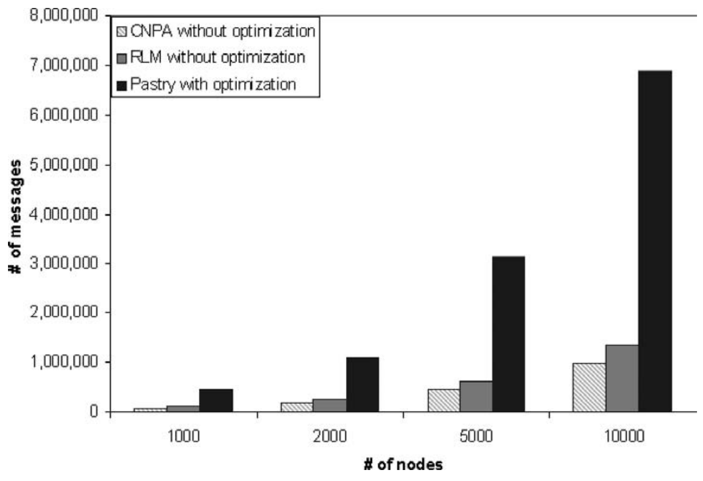
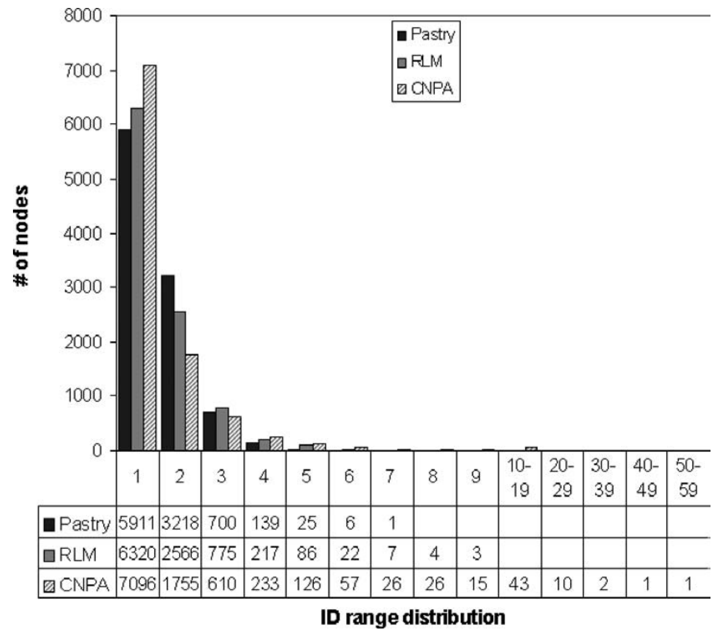


Figure 6. Total number of messages exchanged during bootstrap (Pastry, RLM, CNPA).

utilizes the same bootstrap optimization as Pastry, both RLM and CNPA gain a stretch of 1.19, which is significantly lower than the best possible overlay stretch that Pastry can only score when artificially bootstrapped. This is due to the fact that DynaMO constructs its 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.

Again, one of the main goals was to achieve those overlay stretches with as little overhead as possible. Figure 6 presents the costs of DynaMO's bootstrap variations



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

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

in terms of the total number of messages exchanged. As already observed, both DynaMO 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 stretch (1.42) as RLM and CNPA do without optimization (6.88 million compared to 1.35 and 0.98 million, respectively). Pastry's additional overhead is generated primarily during the optimization process. A small portion, though, stems from the structural differences that affect Pastry's join procedure. Since in DynaMO networks ID clusters form, the distance to the final node in the join process is on average shorter than in Pastry overlays.

If one focused on optimizing the overlay stretch instead of the network traffic, DynaMO could also utilize bootstrap optimization. In this case, DynaMO's 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 or less, 3,318 nodes are responsible for up to twice the optimal ID range, and so forth, and only a single node is responsible for up to 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 or less, 2,566 are responsible for up to twice the optimal ID range, and so forth. At its extreme, merely 4 nodes have to handle up to 8 times more IDs than optimal, and only 3 nodes cover up to 9 times the optimal ID range.

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 overlay stretch as RLM. Both RLM and CNPA obtain equal stretches, but RLM induces a higher message total whereas CNPA sacrifices a perfectly equal ID distribution.

4.2. Networks under churn

In a next step, we evaluated and compared the performance of Pastry, RLM, and CNPA in networks under churn. For all test runs, we used 5,000 node networks. In the first simulation setting, randomly selected 40% of all nodes leave the network and the same amount of new nodes join the network at random. After these random node substitutions occurred, the usual 20,000 random lookups were issued. In the second setting, the churn rate is 100%, i.e. the entire network topology is replaced completely before the lookups are issued. It is important to note that we did not implement network churn as a two-phase process (i.e. first nodes fail or leave, then rejoin), but instead it is a continuous process. When a node fails, it is immediately replaced by a new random node.

The focus of these experiments is to analyze how the different approaches handle network churn *without* inducing additional overhead. Therefore, during these simulations, 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 and CNPA networks, the nodes also left and joined the network at random. In the RLM networks, however, extra care was taken to ensure that one in every 10 failing nodes was a temporary landmark node. In this case, the underlying DHT would implicitly redirect future landmark key lookups to the node that is now numerically closest to that key. Obviously, with this approach, landmark nodes are decidedly more likely to fail than non-landmark nodes. This emphasizes the temporary nature of RLM's landmark nodes and helps to demonstrate that RLM is still able to maintain a good overlay stretch in the presence of high landmark churn rates. Figure 8 depicts the overlay stretches as maintained in the different network churn scenarios.

As explained above, in the first setting, 40% of all nodes were replaced with random new nodes before any lookups were issued. We believe this is a severe case of network churn because during the leave and rejoin phase 40% of the entire network is replaced without any routing table maintenance taking place besides Pastry's simple join

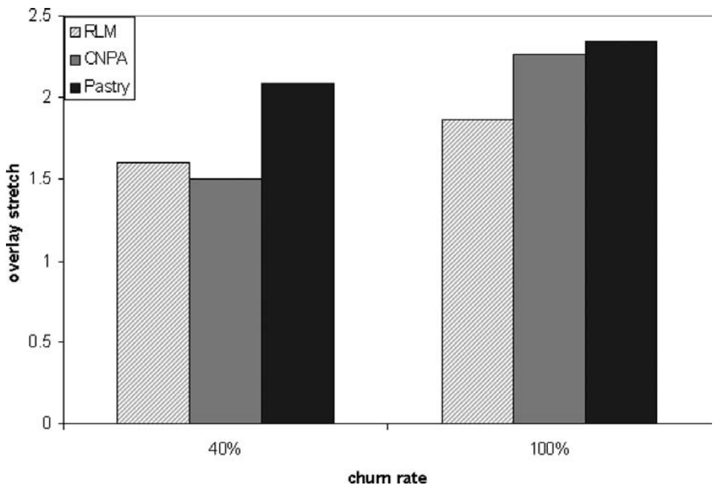


Figure 8. Overlay stretches with different network churn rates (without bootstrap optimization).

heuristics. As can be observed, without bootstrap optimization for the new nodes, Pastry's stretch, that was artificially lowered to 1.30 prior to the network churn, deteriorates significantly with a churn rate of 40%. Both RLM and CNPA maintain very good overlay stretches, but CNPA slightly outperforms RLM as it does not have the problem of frequently changing landmark nodes.

As a worst-case scenario, we considered a network where *all* nodes are replaced by new nodes with no routing table recovery mechanisms employed except for the above mentioned join heuristics. With 100% network churn, Pastry's stretch reaches the level (2.3) that Pastry without optimization achieves in static networks (see figure 3). With RLM, newly joining nodes have to frequently re-look up landmark nodes as they fail significantly more often than other nodes. During those lookups, intermediate nodes can discard invalid routing table entries. This leads to an increased number of empty routing table entries (compared to CNPA and Pastry), which are subsequently replaced automatically with new nodes during their join process. Since this mechanism is missing in CNPA, RLM now outperforms CNPA significantly. CNPA-bootstrapped routing tables contain many more invalid entries in the presence of 100% churn causing longer routing paths. This effect was not as pronounced with a churn rate of 40%.

At this point it has to be mentioned that RLM's better overlay stretch in networks with 100% churn comes with a certain price tag. CNPA and Pastry have the same overhead whereas RLM has to perform landmark measurements and lookups as described above. On the other hand, Pastry started out with a factor of 1.30, which was achieved artificially and would have cost a routing table maintenance overhead decidedly larger than RLM's overhead. If a Pastry network without bootstrap optimization is exposed to a churn rate of 100%, its stretch deteriorates even further and peaks out at 2.60.

Perhaps the most interesting feature of both DynaMO approaches is that they maintain a key property even in the light of 100% network churn: overlay ID locality. Figure 9 shows the nearly unaffected ID clustering after 100% churn has occurred.

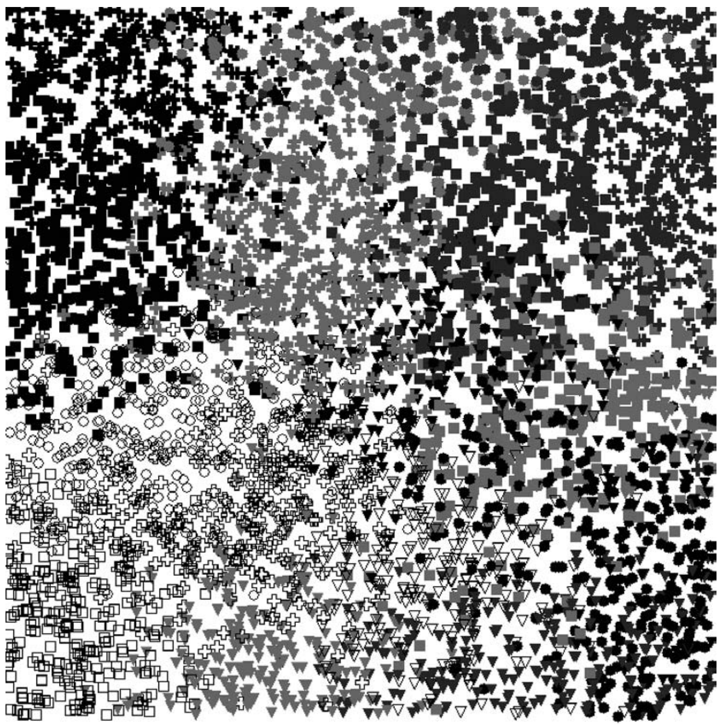
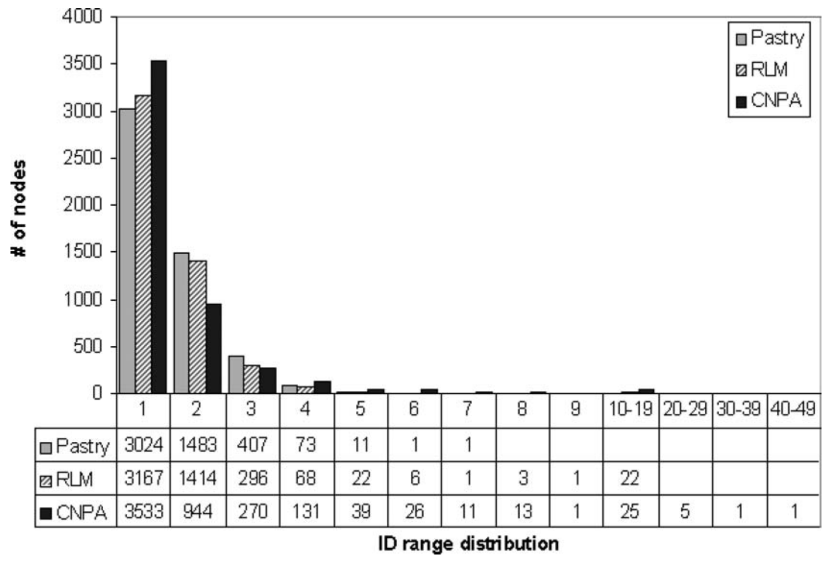


Figure 9. Prefix distribution as generated by RLM after complete network churn (100%). Equal symbols and colors represent equal prefixes.

As mentioned previously, another crucial factor with DHT-based peer-to-peer systems is the distribution of overlay IDs. Figure 10 shows the ID range distribution in terms of the *optimal ID range* (see section 4.1) for Pastry, RLM, and CNPA after all nodes have left the overlay and new replacement nodes have rejoined. As expected, Pastry's overlay ID distribution is completely unaffected by the churn as it assigns its overlay IDs in a totally random fashion. RLM and CNPA roughly maintain their ID distribution after the churn has taken place. In some test runs, the ID distribution even improved slightly after network churn had occurred.

4.3. Mobility

Mobile networks represent the biggest challenge when building topology-aware overlay networks because the underlying physical network changes constantly. In order to evaluate the performance of DynaMO in such networks, we conducted several simulations comparing Pastry and RLM. For RLM, we implemented an ID reassignment strategy to deal with mobility. As CNPA alone (without additional maintenance mechanisms) seems to cope with network churn less effectively than RLM in extreme churn scenarios, it was not taken into consideration here since mobility can be considered an extreme case of churn if ID reassignment is employed.



	1	2	3	4	5	6	7	8	9	10-19	20-29	30-39	40-49
▨ Pastry	3024	1483	407	73	11	1	1						
▨ RLM	3167	1414	296	68	22	6	1	3	1	22			
■ CNPA	3533	944	270	131	39	26	11	13	1	25	5	1	1

Figure 10. Average overlay ID range distribution in a 5,000 node network (Pastry, RLM, CNPA) after network churn (100%).

Pastry has no explicit mechanisms to deal with rapid topological changes in its underlying physical network. The only way to adapt its routing tables to reflect a modified physical underlay is to periodically run routing table maintenance tasks. These tasks are not run explicitly to detect mobility-induced changes, but instead are performed to compensate for any effects causing routing table deterioration.

The routing table maintenance as described in [Castro et al., 1] performs the following task. In certain intervals, each node randomly selects an entry from each row of its routing table. It then asks each of those nodes for its corresponding routing table row. After that, it compares each entry in the row received with the corresponding entry in its own row by probing the distances to them. If need be, it replaces its own entry.

RLM uses a different strategy that deals explicitly with mobility-induced topology changes. Every node periodically re-measures its distance to the current landmark nodes. If its ID prefix is still congruent with the prefix of its closest landmark node, it will increase its re-measure interval by some factor. Otherwise, it will re-assign itself a new overlay ID based on the same strategy as used during its bootstrap and will rejoin the network with its new ID. Due to the extremely dynamic nature of mobile networks, a node uses the standard Pastry bootstrap optimization as explained in section 4 after it has rejoined the network under its new ID. This serves to propagate its new ID faster.

Both Pastry and RLM were evaluated with the following network settings. Due to the added simulation complexity, a 2,000 node physical network served as the underlay. Each test run lasted 24 simulated hours. During those 24 h, the usual 20,000 random lookups were issued. The nodes in the network moved according to the random waypoint mobility model [Johnson and Maltz, 2] with a speed of 0.6 m/s and a pause time of 30 s.

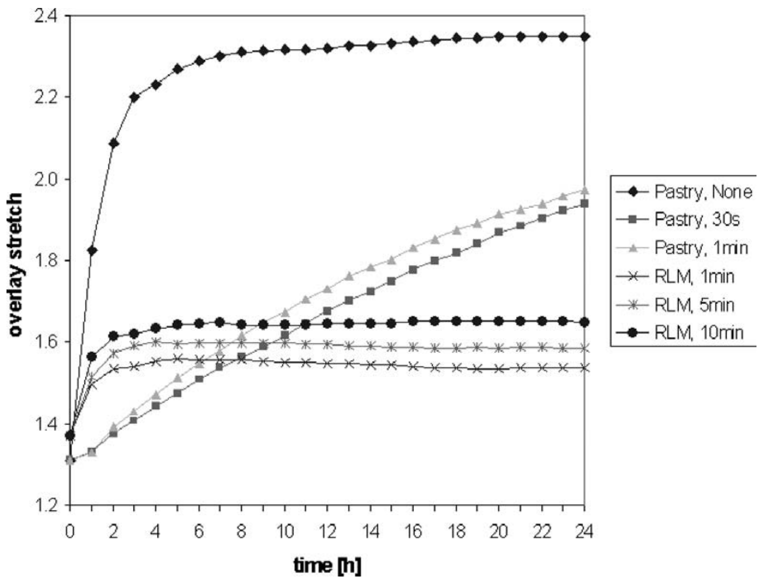


Figure 11. Overlay stretch change over time with Pastry and RLM.

For our Pastry experiments, we evaluated two different routing table maintenance intervals, as well as Pastry networks with no maintenance at all. Before mobility set in, all Pastry networks were artificially bootstrapped to start out with an overlay stretch of 1.31. As figure 11 shows, if no routing table maintenance is performed, Pastry's stretch quickly deteriorates to a level of 2.35 that it would also roughly achieve in static networks without any optimization. Therefore, we considered next a maintenance interval of 1 minute so that each node runs the routing table maintenance task as explained above every minute. As can be seen, Pastry is unable to maintain a stable factor over time. Its stretch deteriorates nearly linearly until it reaches 1.97 after 24 hours. Clearly, this stretch does not peak here but would further rise. For this reason we conducted a second set of experiments, lowering the interval to 30 s thereby increasing the maintenance effort markedly. The results indicate that even with this increased effort Pastry still fails to reach a stable overlay stretch level after 24 hours with a stretch of 1.94 that is only negligibly lower than before.

With the same mobility parameters, we next evaluated RLM. Before mobility set in, all RLM networks were bootstrapped without optimization yielding an initial stretch of 1.37 slightly above Pastry's initial artificial overlay stretch. Figure 11 shows that with a re-measure interval of 1 minute, RLM quickly reaches a stable stretch level of 1.54. If the re-measure interval is increased to 5 minutes, RLM achieves a stable overlay stretch level of 1.58. Even if the nodes re-measure their landmark distances only every 10 minutes, RLM still maintains a stable stretch level of 1.65.

Figure 12 shows the number of overlay messages exchanged during an average 24 h simulation run. These messages include rejoin requests and forwards (only with RLM), distance measurements, and messages containing routing table state information. In the

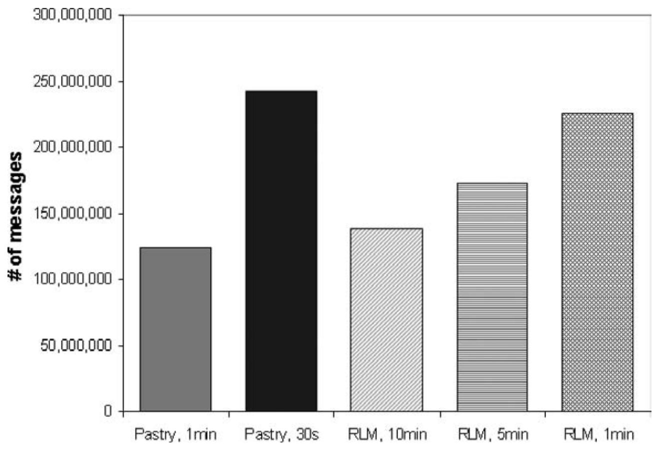


Figure 12. Number of overlay messages exchanged during an average 24 h simulation.

case of RLM, those numbers also include on-demand routing table maintenance to take care of "false positives". A false positive in this case is a routing table entry that refers to a node by its old overlay ID although the node has rejoined the network under a new overlay ID in the meantime. In order to rid our routing tables of such false positives, we employ a form of on-demand maintenance. When a node receives a request, it checks whether it still has the overlay ID that the sending node thinks it does. If not, it will acknowledge that to the requesting node, so that the false positive gets removed from the sending node's routing table. With an increasing request rate, that extra effort becomes negligible as false positives are removed quickly due to the high request frequency. The higher the request rate, the lower the probability will be for a single request to encounter false positives as previous requests are likely to have already cleaned up the routing table. If one was not interested in other effects, such as routing table locality and clustering, and only cared about lowering the overall message overhead, RLM per se requires high request rates in order for its lower overlay stretch to outweigh its construction overhead. Therefore, we believe that the extra effort induced by false positives is negligible.

Figure 12 shows that Pastry exchanges large volumes of overlay messages during the simulated 24 h with both maintenance intervals (124 million and 241 million, respectively). Despite that, Pastry is not able to reach a stable stretch level after 24 h as figure 11 showed. RLM, on the other hand, achieves a stable stretch level (1.65) significantly better than Pastry's with a comparable amount of messages exchanged when the re-measure interval is 10 minutes. If one is willing to accept a message total above Pastry's 1-minute interval total but still well below Pastry's 30 s interval total, RLM's overlay stretch can be lowered even further (see figures 11, 12). It has to be mentioned that in object storage overlays the rejoin of a node induces some additional overhead. Before a node assigns itself a new ID, it would have to pass on the references to the objects it is currently responsible for to its left and right neighbor in the ID space as they now become responsible for them. After rejoining the network with its new ID, the node would have to acquire the references to the objects it has now become responsible for

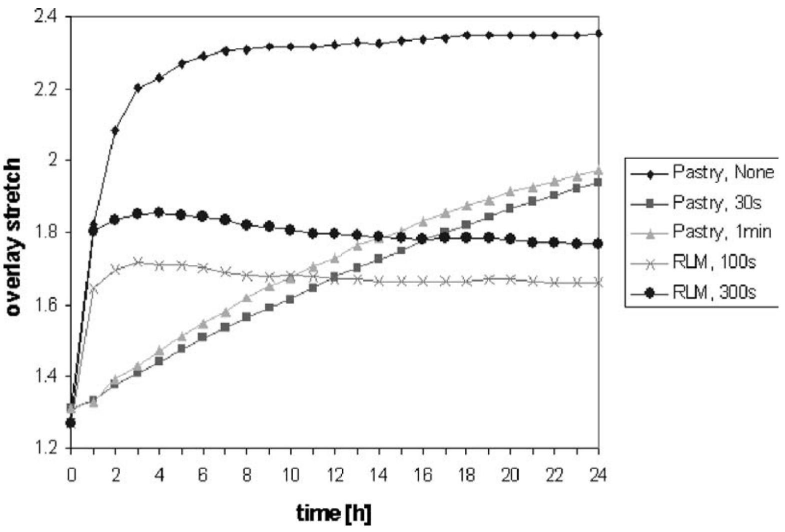


Figure 13. Overlay stretch change over time with Pastry and RLM (with periodic table clean-ups).

from its new left and right ID space neighbor. Obviously, this overhead depends largely on the number of objects being stored on the network and the number of participating nodes.

The next set of simulations we conducted decouples false positive maintenance from the request rate. In order to make that maintenance independent from the requests, we periodically clean up the entire routing table to remove all false positives. Thus, in case a request still encounters a false positive (due to recent ID reassignment after the last routing table clean-up), the extra effort of backtracking the request is added to the overall path length of that request.

Figure 13 shows the development of the overlay stretch over time of Pastry and RLM with periodic routing table clean-ups. Pastry, again, was bootstrapped artificially resulting in an unrealistically good routing table state. In the case of RLM, we employed an ID reassignment interval of 10 minutes. In this scenario, a table clean up works as follows. A node sends a keep-alive message to each node in its routing table. A receiving node only responds if its overlay ID has recently changed, whereupon the initiating node removes that node with its old ID from its table and re-inserts it under its new ID. As can be seen, RLM maintains a stretch of 1.77 over time when a clean-up period of 300 s is employed. For a clean-up period of 100 s, the stretch remains at 1.66. Obviously, cleaning up routing tables periodically is more coarse-grained than on-demand maintenance, which is reflected in the slightly worse overlay stretches compared to figure 11.

Figure 14 depicts the total number of overlay messages exchanged during an average 24 h simulation. Even when cleaning up the *entire* routing table periodically, RLM still produces less or comparable message overhead compared to Pastry with a maintenance interval of 30 s while maintaining a decidedly better overlay stretch over time.

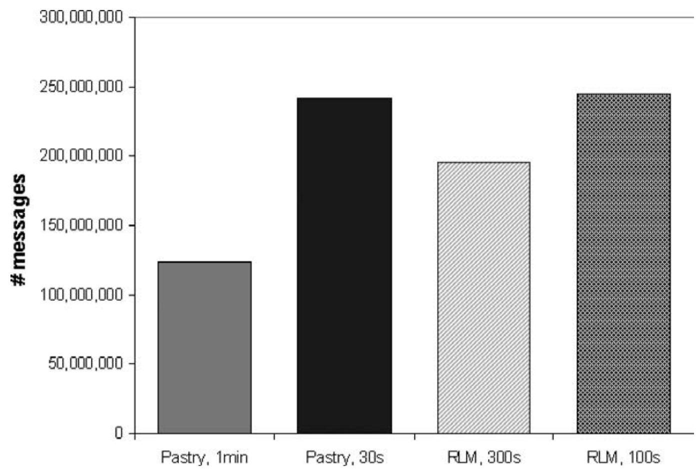


Figure 14. Number of overlay messages exchanged during an average 24 h simulation.

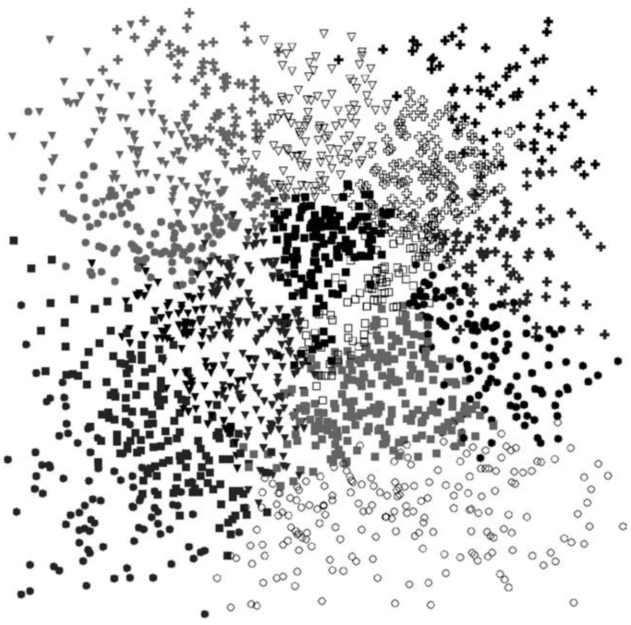
We argue that periodically cleaning up the entire routing table independent of request rates is suboptimal. First of all, it produces an unnecessary amount of network traffic since those routing table entries that turn out not to be needed any time soon are also contacted as well as those that will be needed for the next request forwards. Secondly, the overhead of periodic clean-ups is static since it is not affected by request rates at all. However, as the effect of false positives becomes less and less relevant with increasing request rates, having a static maintenance overhead is even less desirable. On the other hand, a low request rate renders the cleaning up of the entire routing table useless since very few routing table entries are ever used between two clean-ups.

Figures 15(a) and (b) give a visual interpretation of the spatial overlay ID distribution in RLM networks with re-measure intervals of 1 and 10 minutes at the end of a 24 h simulation. With the additional message overhead that a 1-minute interval generates, the spatial distribution remains nearly perfect. With the markedly smaller overhead of the much more coarse-grained 10-minute interval, the overlay ID clustering deteriorates visibly, but still remains acceptable.

Figure 16 shows the ID range distribution in terms of the *optimal ID range* (see section 4.1) for Pastry, RLM with a 1-minute re-measure interval, and RLM with a 10-minute re-measure interval in a 2,000 node mobile network at the end of a 24 h simulation. With both re-measure intervals, RLM retains very good ID range distributions.

5. Application scenarios

DynaMO groups physically close nodes into common overlay regions, i.e. physically close nodes are likely to share a common overlay ID prefix. Conversely, DynaMO makes it very likely for two physically close nodes to be also close to each other in the overlay ID space. Practically, one could consider these regions in the overlay network as certain, common prefix peer-to-peer clusters. Since DynaMO forms these peer-to-peer clusters


(a)

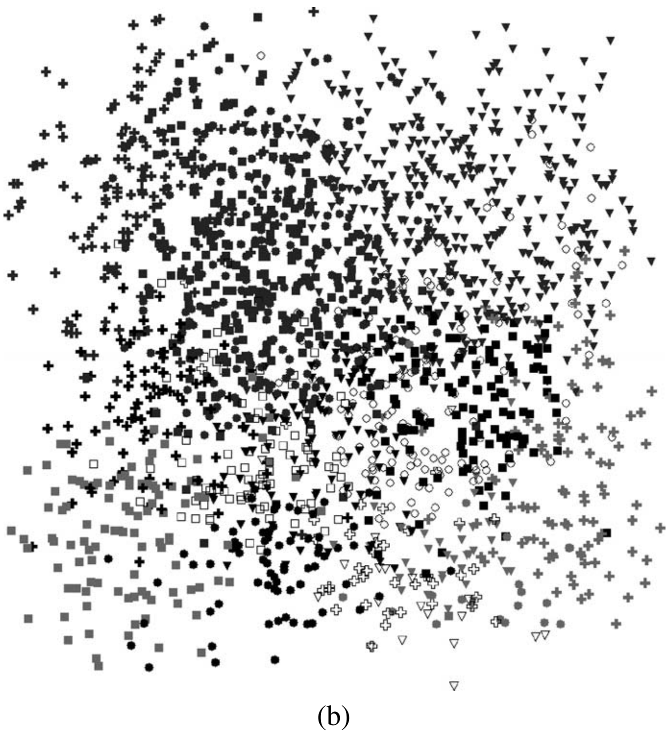
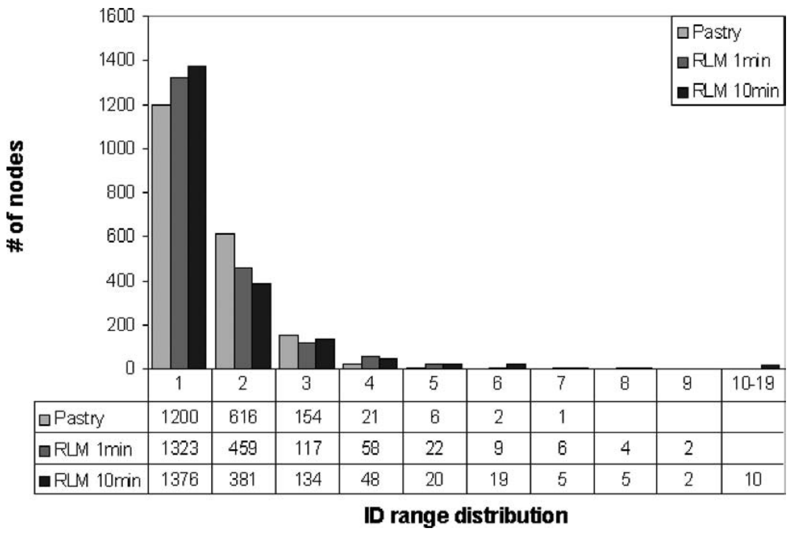

(b)

Figure 15. Spatial overlay ID distribution in an RLM-based mobile network with a (a) 1-minute, (b) 10-minute re-measure interval at the end of a simulation. Equal symbols and colors represent equal prefixes.



	1	2	3	4	5	6	7	8	9	10-19
▨ Pastry	1200	616	154	21	6	2	1			
▨ RLM 1min	1323	459	117	58	22	9	6	4	2	
■ RLM 10min	1376	381	134	48	20	19	5	5	2	10

Figure 16. Average overlay ID range distribution in a 2,000 node mobile network (Pastry, RLM with 1-minute re-measure interval, RLM with 10-minute re-measure interval) at the end of a simulation.

based on physical proximity (among the cluster members), intra-cluster communication with DynaMO has the following properties:

1. Since nodes in a DynaMO cluster share a common overlay ID prefix, intra-cluster communication will likely involve only *very* few overlay routing steps – in fact, it will often be done with only one overlay hop as many of its fellow cluster members will be covered by a node's leaf set.

2. Since DynaMO clusters are formed based on physical proximity, intra-cluster communication will likely incur short physical routes.

While these characteristics are certainly useful in all sorts of networks, they are especially well-suited for mobile, wireless ad-hoc networks. In mobile, wireless networks, with their limited bandwidth, packet losses and volatile routes between nodes occur much more frequently than in wired, stationary networks. Hence, one should strive to keep communication in such networks as local as possible and routes as short as possible. DynaMO effectively helps achieve these goals in the deployment of peer-to-peer overlays in mobile, wireless environments.

One appropriate application of DynaMO could be Peer-to-Peer e-mail/messaging services in mobile networks. Such an application would benefit from DynaMO in two ways:

1. Since DynaMO lowers the overlay stretch, it delivers messages more directly (i.e. with shorter physical routes) to their recipients than a regular DHT (such as Pastry) would. This would clearly increase the message delivery rate as shorter physical routes decrease the likelihood of packet losses and routes are less likely to be broken.

2. Under the assumption that nodes are more likely to communicate with nodes that are in their physical vicinity rather than with nodes across the networks, DynaMO would help reflect this communication pattern also in the overlay traffic. Since DynaMO's overlay clusters are based on physical proximity, such a communication pattern would result in a large fraction of intra-cluster communication. As intra-cluster communication involves only few overlay hops (many requests could be directly serviced by a node's leaf set, e.g.), DynaMO would thus also help to reduce overlay traffic, which in turn decreases real, physical traffic.

DynaMO's overlay clusters could also be directly used for general Peer-to-Peer lookup services in mobile networks. For this purpose, objects would be stored on the overlay under two different keys. The first one is a global key as produced by a conventional hash function and the object is stored on a random (as determined by the hash function) node on the overlay. The object (or a reference to it) is also stored locally inside its owner's cluster. For this matter, it is also assigned a local key. This local key can be produced by replacing the global key's prefix with the object's owner's ID prefix. When an overlay nodes wants to look up an object, it will now first query its own cluster (by replacing the prefix of the query key with its own prefix). Only if that query fails (i.e. a NULL response is returned from the node in its cluster currently responsible for the local key), will the node start a global query. This way, local objects (such as files, services, etc.) would be favored over far-way objects, thereby reducing unnecessary cross-network traffic.

6. Conclusion and future work

Adopting "traditional" networking technologies and techniques from wired networks into dynamic, mobile networking environments often proves to be inefficient. Even worse, approaches designed for infrastructure based networks are often unsuitable for ad-hoc networks since the infrastructural conditions these approaches are based on are simply not available.

The presented P2P system DynaMO adopts a general idea from overlay networks designed to be used on the Internet. It re-designs the idea of landmarking to a well suited approach in wireless networks. Additionally, DynaMO's design goes a step further and includes a mechanism which prevents the topological awareness from decreasing due to node mobility. Still, DynaMO is light weight and is able to maintain an even overlay ID distribution even in highly dynamic conditions cause by network churn and node mobility.

In this article we have analyzed DynaMO's two approaches designed to construct topology-aware overlay networks. Our first goal was to reduce the overlay stretch, which is 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 overlay stretches with significantly less communication effort. Up to 5–7 times more messages have to be exchanged using Pastry compared to RLM and CNPA achieving the same stretch. On the other hand, when adding the same optimization effort as with Pastry, our approaches were able to achieve a stretch decidedly lower than even the artificial 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.

Our second goal was to design our approaches in a way that they can maintain their key properties even in the presence of high network dynamics, especially node mobility. We showed that DynaMO maintains a significantly lower overlay stretch than Pastry does in mobile networks while exchanging a smaller or comparable amount of messages.

DynaMO's lower overlay stretch becomes even more important as both the request rate and the network size increases. Even in networks with low request rates, DynaMO maintains valuable properties such as locality and clustering. These properties could also be used for location-based services, e.g. Especially in wireless ad-hoc networks the lower overlay stretch is particularly beneficial since messages traveling physically shorter overlay hops are less likely to be dropped due effects such as collision, bit errors, etc.

In the future, we plan to evaluate DynaMO's performance in various network topologies. We will extend DynaMO's approaches to lower the message overhead in mobile networks even further. Additionally, the effect of a combination of RLM and CNPA on the overlay stretch and overlay ID distribution will be examined and different overlay ID reassignment strategies will also be evaluated.

References

[1] M. Castro et al., Exploiting network proximity in peer-to-peer overlay networks, in: *Internat. Workshop on Future Directions in Distributed Computing (FuDiCo)* (2002).

[2] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad-hoc wireless networks, in: *Mobile Computing*, Vol. 353, eds. Imielinski and Korth (Kluwer Academic, Dordrecht, 1996).

[3] Omnet++, Discrete event simulation system, `http://www.omnetpp.org` (2003).

[4] C.E. Perkins, Ad-hoc on-demand distance vector routing, `http://people.nokia.net/~charliep/` (2003).

[5] S. Ratnasamy et al., A scalable content-addressable network, in: *Proc. of ACM SIGCOMM* (2001).

[6] S. Ratnasamy et al., Topologically-aware overlay construction and server selection, in: *IEEE INFOCOM* (2002).

[7] J. Ritter, Why Gnutella cannot scale, no, really, `http://www.darkridge.com/~jpr5/doc/gnutella.html` (2001).

[8] A. Rowstron and P. Druschel, Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, in: *Internat. Conf. on Distributed Systems Platforms*, Middleware (2001).

[9] I. Stoica et al., Chord: A scalable peer-to-peer lookup service for Internet applications, in: *Proc. of ACM SIGCOMM* (2001).

[10] The Gnutella protocol specification v0.4, `www9.limewire.com/developer/gnutella_protocol_0.4.pdf` (2001).

[11] M. Waldvogel and R. Rinaldi, Efficient topology-aware overlay network, in: *SIGCOMM/CCR 2003*.

[12] Z. Xu, C. Tang and Z. Zhang, Building topology-aware overlays using global soft-state, in: *ICDSC'2003*.

[13] 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, Berkeley (2001).