2 Background

This chapter provides a closer look into the technologies that serve as a background for this thesis. At first, the conceptual differences between unstructured and structured peer-to-peer overlays will be explained in detail. Following that, the typical issues in mobile ad hoc networks will be examined. The final part of this chapter will look at optimizations for structured overlays that consider physical locality in the overlay network.

2.1 Peer-to-Peer Overlay Networks

As already mentioned briefly in the introduction, peer-to-peer networks are decentralized, self-organizing application-layer networks where – ideally – all nodes (i.e. peers) assume equal roles. That is, each peer will usually function as both a client and a server in the network. In other words, a peer will issue requests of its own, forward requests on the behalf of other nodes, and respond to requests from other nodes. P2P networks are also referred to as *overlay networks*. This means that P2P networks form a virtual topology on top of an underlying physical network. Figure 2.1 shows the relationship between a P2P overlay

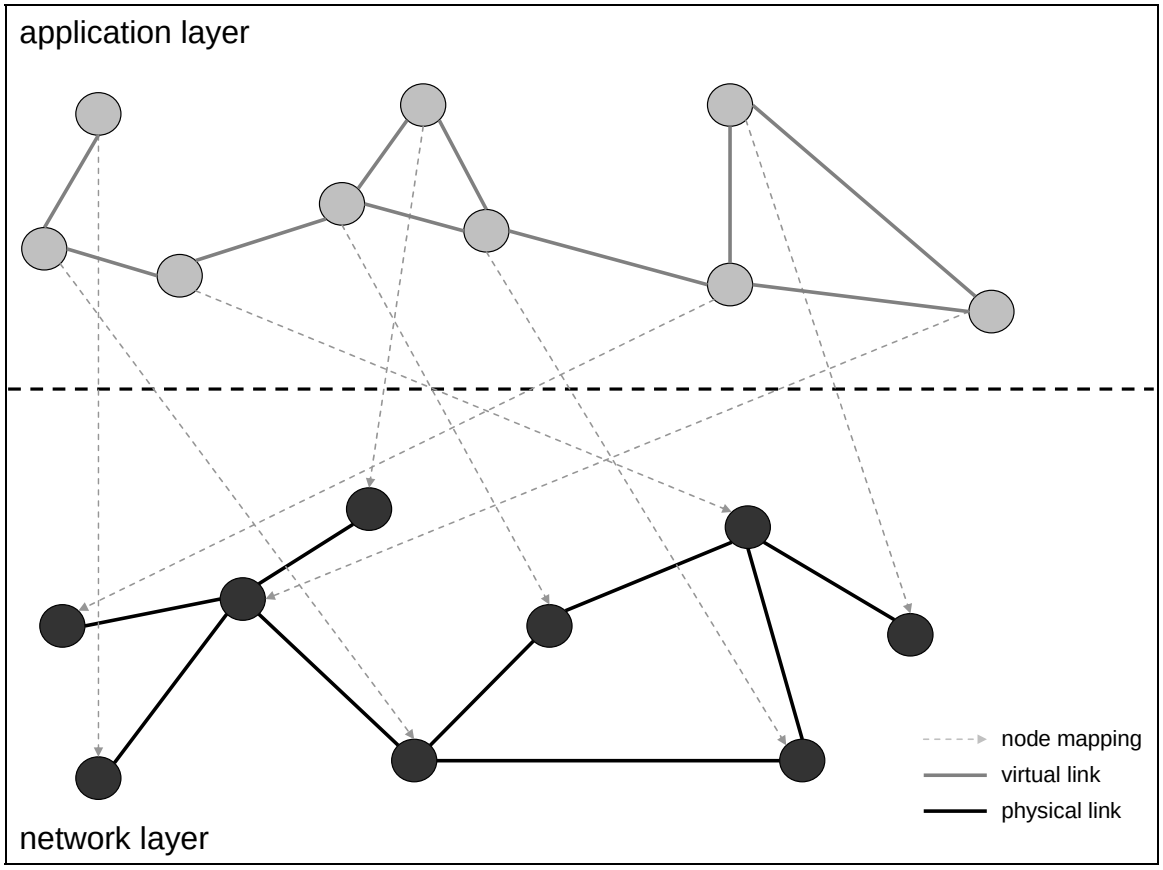

Figure 2.1 P2P overlay network vs. physical network.

network and the underlying physical network. It depicts both the overlay and physical topology and the position of each node in those two topologies. Links between nodes in the overlay are referred to as virtual links as they do not exist in a physical sense. Rather, a virtual link is merely given by an entry in a node's overlay routing table. Since neighboring nodes in the overlay network are very likely not be neighbors in the underlying physical network, an overlay hop (i.e. the passing of a message from a peer to one of its neighbor peers) will usually translate into a physical route that likely comprises numerous physical hops. Moreover, as conventional P2P overlay networks have usually been designed for the use in the Internet, they take physical routing for granted. Thus, they are largely oblivious to the physical aspects of the underlying network.

2.1.1 Unstructured Overlays

As P2P networks are decentralized, the most fundamental issue for any peer is to find the node that has the information that the current peer is interested in. First-generation P2P networks, especially those popularly used for file-sharing such as Gnutella [17], construct so-called *unstructured overlays*. In principal, there are no restrictions on which nodes a peer can include in its routing table. Therefore, the resulting overlay will exhibit a random topology with no explicit structure. Such unstructured overlays are easy to form and practically require no explicit routing table maintenance as failed entries can be replaced with any other nodes. On the other hand, having no explicit structure, it is virtually impossible to predict the best node to forward a request for a certain piece of information to. The requested information could potentially reside with any node in the network and each routing table entry is equally likely to provide it. For this reason, unstructured P2P overlays employ *flooding* to locate information in the network. When a node wants to find a certain piece of information, it will forward the request to *all* entries in its routing table. Each recipient node will then check whether it has the requested information. If so, it will reply to the originator with the requested information. Otherwise, it, too, will forward the request to *all of its* routing table entries. In general, this process continues until all nodes in the network have been contacted or the TTL (time-to-live) parameter of the request has expired. It is clear to see that this flooding-based information discovery in unstructured overlays does not scale to a growing number of nodes and requests and that it can easily overwhelm the underlying physical network [48]. Again, one could reduce this overhead (and thus increase the scalability) by restricting the broadcasts to a certain radius, but this would come at expense of a decreased recall.

To curb the network traffic produced by them, unstructured overlays have more recently started to impose a two-level hierarchy on their topologies [18, 26]. The general idea is to restrict the information discovery process to more powerful nodes – so-called super-peers. Super-peers usually possess higher bandwidth, more computational power, and/or longer up-times than general peers do. A weaker node will connect to a super-peer and send a list of all the information (e.g. files) that it is providing to its super-peer. The super-peer then acts an index server for its client peers. When a peer now requests some information, it will send the request to its super-peer. The request is then propagated only among the superpeers. Each super-peer checks whether any of its client peers can

provide the requested information and, if so, replies to the originator with the appropriate client's address. While thus the flooding traffic is restricted to the super-peers, the information discovery process is not changed fundamentally. The problem of scalability is merely shifted to a – albeit more powerful – subset of nodes.

2.1.2 Structured Overlays

To overcome the scalability issues of unstructured P2P networks without sacrificing the recall, Distributed Hash Tables (DHTs, e.g. [45, 50, 56, 73]) have been introduced. At the core of each DHT lies the ability to route a packet based on a key to the node in the network that is currently responsible for the packet's key. This process is referred to as indirect or key-based routing. To do this, DHTs form a virtual overlay identifier space (e.g. 128-bit identifiers) from which each peer assigns itself a random overlay ID – e.g. by hashing its IP address into the overlay ID space. Analogously, each object is also hashed into the same overlay ID space. A node is now responsible for an object if the node's own overlay ID is closest to the object's ID among all live nodes in the network. Closeness in this aspect depends on the nature of the overlay ID space and the characteristics of the respective DHT, but common examples are numerical distance or Euclidean distance.

Unlike unstructured P2P networks with their random topology, DHTs impose a structure on the overlay topology by no longer choosing routing table entries arbitrarily. Instead, routing table entries have to satisfy certain criteria depending on the respective DHTs. That topological structure enables DHTs to introduce an upper bound on the number of overlay hops that have to be taken to route a given packet to the node currently responsible for the packet's key. This upper bound is commonly O(log N), with N being the number of nodes in the network. How this bound is achieved, fundamentally depends on the routing strategies employed by the respective DHTs. Those strategies include reducing the Euclidean distance in the overlay ID space to the destination in each overlay routing step [45], halving the numerical distance to the destination in each overlay routing step [56], or increasing the length of the matching prefix/suffix between the current node's overlay ID and the key in each overlay routing step [50, 73]. While DHTs can route packets very efficiently in comparison to unstructured P2P networks, they, on the other hand, usually induce a higher traffic overhead due to the maintenance of their routing tables.

To understand the concept of DHTs more clearly, Pastry [50] shall be presented in more detail here as an exemplary DHT as it also serves as a basis for this thesis. Pastry employs a 128-bit module ring overlay ID space – i.e. all Pastry overlay IDs range from 0 to 2^{128}-1. Pastry overlay IDs are organized in digits of base 2^b (typically, b is set to 4 to obtain hexadecimal digits). In Pastry, a node is responsible for a given key if the node's overlay ID is *numerically* closest to the key among all live nodes. To provide key-based overlay routing, each Pastry node maintains two node sets: a *routing table* and a *leaf set*.

A Pastry routing table consists of $\log_{2^b}N$ rows and 2^b-1 columns. Each routing table entry consists of the node's overlay ID and IP address. In row i, Pastry

Local Overlay ID: 34035761

Routing Table

0	1	2	3	4	5	6	7
03761261	13541241	27235106		45521251	53610176	65547772	71154601
30771255	31521225	32236556	33400741		35716236	36124163	37715026
	34102724	34203162	34303735	34404216	34576555	34653437	34734152
34006624	34012773	34025354		34045631	34050032	34064645	34076214
34030342	34031643	34032346	34033652	34034040		34036123	34037437
34035073	34035152	34035231		34035470		34035667	
34035701	34035717			34035740			34035774

Leaf Set

smaller	34035667	34035701	34035717	34035740
larger	34035774	34036012	34036042	34036076

Figure 2.2 Example of a Pastry routing table and leaf set. Overlay ID digits have base 8 (b=3) and the leaf set contains 2^b entries.

stipulates that each entry have an overlay ID that shares a matching prefix of length i-1 with the local node's own overlay ID. This means that the entries in the first routing table row do not have to share any overlay ID prefix with the local node, all entries in the second row have to have overlay IDs with the same first digit as the local node, all entries in the third row have to share the same first two overlay digits with the local node, and so forth. Furthermore, a Pastry routing table consists of 2^b columns – one column for each value that an overlay ID digit can assume (e.g. 0,1,2,…,E,F for hexadecimal digits). As described above, each entry in row i shares the first i-1 digits with the local node's overlay ID but differs in digit i. The value of an entry's i^{th} digit is given by its column index j ($0 <= j <= 2^b-1$). Note that the local node trivially satisfies this condition for the slot that corresponds to the value of its own i^{th} digit. Therefore, a Pastry routing table row will contain at most 2^b-1 entries.

A Pastry leaf set, on the other hand, consists of l entries, with l commonly set to 2^b or $2*2^b$. One half of the leaf set contains those $l/2$ nodes whose overlay IDs are numerically closest but smaller than the local node's own overlay ID. The other half contains the l/2 nodes with the numerically closest larger overlay IDs. In other words, one can think of the nodes in the leaf set as the "left" and "right" overlay neighbors of the local node in the overlay ID ring. Figure 2.2 shows an exemplary Pastry routing table and leaf set.

Overlay routing in Pastry works prefix-based. The general idea in Pastry's overlay routing is to increase the length of the matching prefix between the intermediate (i.e. the forwarding) node and the key of the packet, that is being forwarded, by one with each overlay hop, or, if that is not possible, to at least reduce the numerical distance between the intermediate node's overlay ID and the packet's key. For this purpose, each forwarding node first determines whether it is itself already responsible for the packet's key. This is the case when the node's own overlay ID is numerically closer to the packet's key than each of the node's leaves' IDs. By definition, this would imply that the node's overlay ID is numerically closest to the packet's key among all live nodes – provided that the node's leaf set is correct, of course. Otherwise, the forwarding node checks whether one of its leaves is responsible for the packet's key, and, if that is the case, it forwards the packet to that leaf. If none of the leaves is responsible for the packet's key, the forwarding node consults its routing table. First, the forwarding node determines the length of the matching prefix between its own overlay ID and the packet's key. This number establishes the routing table row from which to select the destination of the next overlay hop (for example, if the matching prefix length is 0, the first routing table row will be consulted, if the length is three, the fourth row will be considered, etc.). Once the routing table row is established, the forwarding node selects the entry from the column that corresponds with the key's digit that immediately follows the matching prefix. If an entry can be found at the given routing table slot, the forwarding node will send the packet to that entry, thereby increasing the matching prefix length by one with the ensuing overlay hop. In case no such entry exists, however, the forwarding node will select the node that is numerically closest to the packet's key among all the entries in its routing table and leaf set and the packet will be forwarded to that node. This process continues at each intermediate node until the packet eventually reaches the node whose overlay ID is closest to the packet's key among all live nodes. As can easily be seen, the expected number of overlay hops during a Pastry lookup is O(log N).

Figure 2.3 shows an example of the Pastry overlay routing process. Suppose a node whose overlay ID is `65A1FC` intends to send a packet with key `D469FC` to the node currently responsible for that key. Since node `65A1FC` does not share any prefix with the key, it consults the first row of its routing table and selects node `D1A08E` that has a matching prefix with the key of length 1. Therefore, in the first overlay hop, node `65A1FC` forwards the packet to node `D1A08E`. Node `D1A08E` then consults the second row of its routing table to find a node that has a matching prefix with the key of length 2. Thus, in the second overlay hop, node `D1A08E` forwards the packet to node `D4213F`. Node `D4213F` now checks the third row of its routing table and forwards the packet to node `D462BA` that shares the first three digits with the key. Node `D462BA` then determines that one of its leaves, node `D46A01`, is responsible for the packet's key and forwards the packet to its final destination.

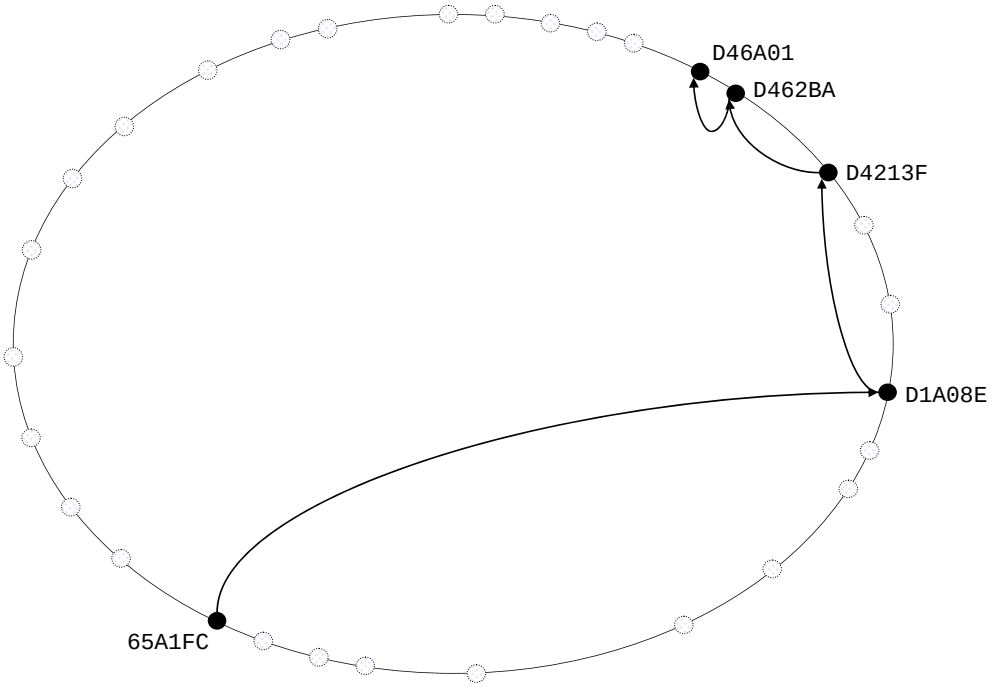

Figure 2.3 Example of the Pastry overlay routing process.

It is obvious that in order for Pastry's routing to maintain its O(log N) bound, nodes need to have accurate routing tables and in particular accurate leaf sets (without accurate leaf sets, the overlay routing might no longer converge). For this reason, Pastry nodes periodically maintain their routing state. When a node notices that one of its leaves is no longer alive, it will contact its numerically largest live leaf (if the failed leaf was inside the leaf set half containing the numerically larger leaves) or its numerically smallest live leaf (in case the failed leaf was inside the leaf set half containing the numerically smaller leaves), respectively. The contacted leaf will then respond with its own leaf set so that the originating node can replace the failed leaf with an appropriate new live entry. In practice, a Pastry node A will usually contact its leaves periodically to check whether they are still alive and to receive their leaf sets. It will then inspect those remote leaf sets to determine if any nodes have joined or left the network that node A has not heard about, but that would also fall into node A's own leaf set. Aside from maintaining its leaf set, a Pastry node also performs routing table maintenance. For this purpose, a Pastry node will periodically select a random node from each of its routing table rows and request the corresponding routing table row from that node. Upon reception of such a remote routing table row, a Pastry node will pairwise contact the entries in the remote row and those in its own corresponding routing table row. This process actually serves two purposes. First, a Pastry node thus checks if (some of) its routing table entries are still alive and learns about new nodes that could be inserted into its routing table to replace empty slots or failed entries. Second, it allows a Pastry node to optimize its routing table entries with respect to the underlying physical network. When comparing two possible entries (i.e. its own and the remote) for a given routing table slot, preference can be assigned according to an appropriate physical metric such as latency or hop count. This process is often referred to as Proximity Neighbor Selection [5].

2.2 Mobile Ad Hoc Networks

Generally speaking, mobile ad hoc networks (MANETs) consist of mobile devices (e.g. PDAs, laptops, sensor nodes, etc.) that communicate amongst each other in a wireless fashion – for example using the IEEE 802.11 standard. Those devices spontaneously form a network amongst themselves without any central infrastructure or fixed topology. Having no central infrastructure as well as a highly dynamic topology, arguably the most fundamental task in a MANET is routing. How can a packet be delivered from a source node to a specified destination node, most likely via a chain of forwarding node? This routing process is also often referred to as *unicasting*.

There exists a plethora of routing protocols for MANETs. It is clearly beyond the scope of this thesis to present a comprehensive analysis of all MANET routing protocols. Instead, this section will provide an overview of the different concepts behind the various protocols. In general, one differentiates between flat routing and hierarchical routing [53].

2.2.1 Flat Routing Protocols

Flat routing protocols operate on the assumption that all nodes in the network move around autonomously. Therefore, all nodes in the network generally assume equal roles in the routing process. One can further subdivide flat routing protocols into *proactive* and *reactive* protocols.

Proactive routing protocols update route information in their routing tables independent of actual demands. This means that nodes maintain routes to other nodes even if those routes are not currently needed for any data packets. Many proactive protocols are link state based. To keep routes up-to-date in the presence of node mobility and failures, nodes broadcast information about their network neighbors periodically or event-driven. The main advantage of proactive protocols is that they can allow for low data packet transmission delays. When a node is about to send a data packet to a target node, the route will likely be known and up-to-date so that the data packet will not have to be cached to wait for the response of a lengthy and costly route discovery. This advantage becomes even more pronounced in traffic scenarios where nodes send data packets to frequently changing target nodes. The obvious disadvantage of proactive routing protocols is their constant route maintenance traffic can easily constitute a significant part of the overall traffic and can lead to an increased number of collisions with actual data packets. Since routes are also maintained even when they are not currently needed, proactive protocols are also not well-suited for stream-based traffic scenarios where nodes tend to communicate with an unchanging set of nodes over a certain period of time. Furthermore, due to the maintenance even of unused routes, proactive protocols can quickly drain the energy resources of the participating nodes. Popular representatives of proactive MANET routing protocols include DSDV [40] and OLSR [7].

Reactive routing protocols, on the other hand, discover routes on-demand. This means that a route from some node S to another node T is only established when node S actually is about to send a packet to node T. When routes have not been

Destination	Next Hop	Sequence Number	Distance (hop count)	Other fields
17	54	5	4	...
23	23	11	1	...
41	23	7	3	...
54	54	7	1	...
62	54	9	7	...
71	54	10	5	...
110	23	6	7	...

Figure 2.4 Example of an AODV routing table.

used in a certain while, they usually expire and are expunged from the routing tables. A clear advantage of reactive routing protocols is that they do not generate any significant maintenance traffic. Only those routes are discovered/maintained that are currently needed. Thus, reactive protocols can scale to a large number of nodes as long as each source node tends to communicate with the same target node (or set of target nodes) over a longer period of time – so that costly route discoveries are limited – and as long as node mobility is rather mild – so that recently discovered routes remain valid for a certain period. Another advantage of reactive routing protocols is that, in scarce or bursty traffic scenarios, nodes can usually significantly preserve energy as they will not have to consume any energy for the maintenance of currently unneeded routes. The main disadvantage of reactive protocols is that they do not scale well to high and arbitrary traffic. When nodes send out packets at a high rates to frequently changing target nodes, the MANET will be flooded with route discoveries. Obviously, this problem would be further aggravated by high mobility rates where recently cached routes break quickly. Another negative effect of discovering routes on-demand is the increased delay between initiating the sending of a data packet until the routes has been discovered and the final delivery of the data packet to the target node. Popular representatives of reactive MANET routing protocols are DSR [24] and AODV [41].

As AODV [41] also serves as a basis for this thesis, its general concepts shall be outlined here. Being a reactive routing protocol, AODV nodes only discover routes when they are being needed. The AODV routing table of a node A contains information on recently discovered or used routes. Practically speaking, for each such route, the routing table contains the next hop that has to be taken from node A to get to the target, a sequence number that serves as a timestamp of this route information, and the length of this route as determined by the hop count. Figure 2.4 shows an example of an AODV routing table. If not used or updated during a certain period, entries expire and are expunged from the table. AODV routing is very straight-forward. When a node wants to send out a data packet, it

looks up the entry for the packet's destination in its routing table and sends the packet to the next hop as indicated by the routing table entry. Upon reception of the packet, that next hop node looks up the packet's destination in its routing table and forwards the packet to the next hop as indicated by its entry. This process continues at each intermediate node until the packet reaches its destination.

If a node that is about to send out a data packet cannot find a routing table entry for the packet's destination, a route discovery will be started. Among other fields, such a route request contains the source address (i.e. the address of the requesting node), the destination address (i.e. the address of the node to whom a route is request), and the last known sequence number for the destination (to make sure that no older entries are sent back than the entries the source has already known). The route request is then broadcast throughout the network. Each receiving node temporarily stores the previous hop of the request as the next hop for the reverse route back to the source node. If an appropriate entry (i.e. an entry whose sequence number is larger than the one included in the request) can be found in the routing table, the node will send back a route reply containing the destination sequence number. Otherwise, the request is forwarded further. When the destination node itself receives the route request, it will increment its local sequence number and respond with reply containing the new sequence number.

Each intermediate node that receives a route reply will increment the hop count and forward that reply using the reverse route from the corresponding route request until the reply is received by the source of the request. Furthermore, each intermediate node as well as the source node of the request, will update their routing table with the information from the route reply. This information will be stored in the routing table if i) no entry can be found for the request's destination, ii) there is already an entry for the request's destination but its sequence number is smaller than the sequence number included in the reply (i.e. the entry is outdated), or iii) the existing entry's sequence number is equal to that of the reply but the reply's route is shorter.

2.2.2 Hierarchical Routing Protocols

In contrast to flat routing protocols, that usually scale only to a limited number of nodes, hierarchical routing protocols try to exploit a common node behavior where, instead of moving around completely independently of each other, nodes often move around in clusters or groups. Therefore, information about topological changes in some group A have to be broadcast only inside that group A. Nodes from other groups are not affected as long as they still know how to reach any node – often a specific group head that serves as gateway to and from its group – in group A so that they can still deliver data packets to that node who will then take care of the routing inside its own group. An advantage of having node groups is that route maintenance can be markedly reduced. For example, nodes could maintain the routes to nodes inside their own group at regular intervals. Only the node that serves as group head or gateway will further have to maintain routes to heads of other groups. Furthermore, hierarchical routing protocols can scale well to growing network sizes. It is, for example, conceivable that, as the

number of nodes increases, groups start forming super-groups, super-super-groups, etc. to further divide the route maintenance responsibilities.

However, hierarchical routing protocols can also have significant disadvantages. Since node heads serve as gateways to and from their groups, they will have to handle higher traffic rates than regular nodes do, which, in turn, will also drain their energy resources more quickly. Thus, group heads can become bottlenecks in the routing process. Another problem can be the organization and maintenance of the hierarchy itself. Due to node mobility, nodes will not permanently stay in the same group. As nodes leave their groups and enter new groups, the hierarchy will constantly have to be adapted to this. Especially when nodes serving as heads change groups, this can trigger cascading group reorganizations. An often cited example of a hierarchical routing protocol is HSR [39].

Often, concepts from hierarchical and flat routing are combined. ZRP [21] is a classic example of such hybrid routing strategies. Each node in ZRP forms a group of a certain radius (e.g. two hops) and uses proactive routing inside its group. For all other destinations outside their groups, nodes employ a reactive routing protocol. While ZRP alleviates the costs of (re-)organizing the hierarchy and generally avoids the formation of bottlenecks, it quickly assumes the behavior and runs into the same problems as reactive flat routing protocols as the number of nodes in the network increases and, thus, most destinations start lying outside the nodes' respective groups.

2.3 Topology-Aware Structured Overlay Networks

It is important to bear in mind that a single overlay hop actually constitutes a physical route most likely consisting of multiple physical hops from the source node to the destination node of the overlay hop. At the same time, conventional Distributed Hash Tables were designed for the Internet where physical routing is practically taken for granted. For this reason, DHTs are largely oblivious to the underlying physical network during the construction of their overlay topology. The practical implication of this is that two overlay neighbors – i.e. two nodes whose overlay IDs are close to each other in the overlay ID space – will usually not be in close proximity of each other in the underlying physical network. This leads to a common effect referred to as *overlay stretch*. Technically speaking, the overlay stretch is the ratio between the length of the accumulated physical route traveled during an overlay lookup process compared to the length of the direct physical path from the source to the eventual target node.

Figure 2.5 illustrates the overlay stretch. In this example, node S wants to send a packet with key `D46A17` to the node currently responsible for the packet's key. In the first overlay hop, the DHT (in this example Pastry) then routes the packet to node A with overlay ID `D1A08E`. Node A forwards the packet in the overlay network to node B with overlay ID `D4213F`. Node B then forwards the packet to node C (overlay ID `D462BA`) who eventually delivers it to node T (overlay ID `D46A01`). Node T is responsible for the packet's key and, thus, the packet has

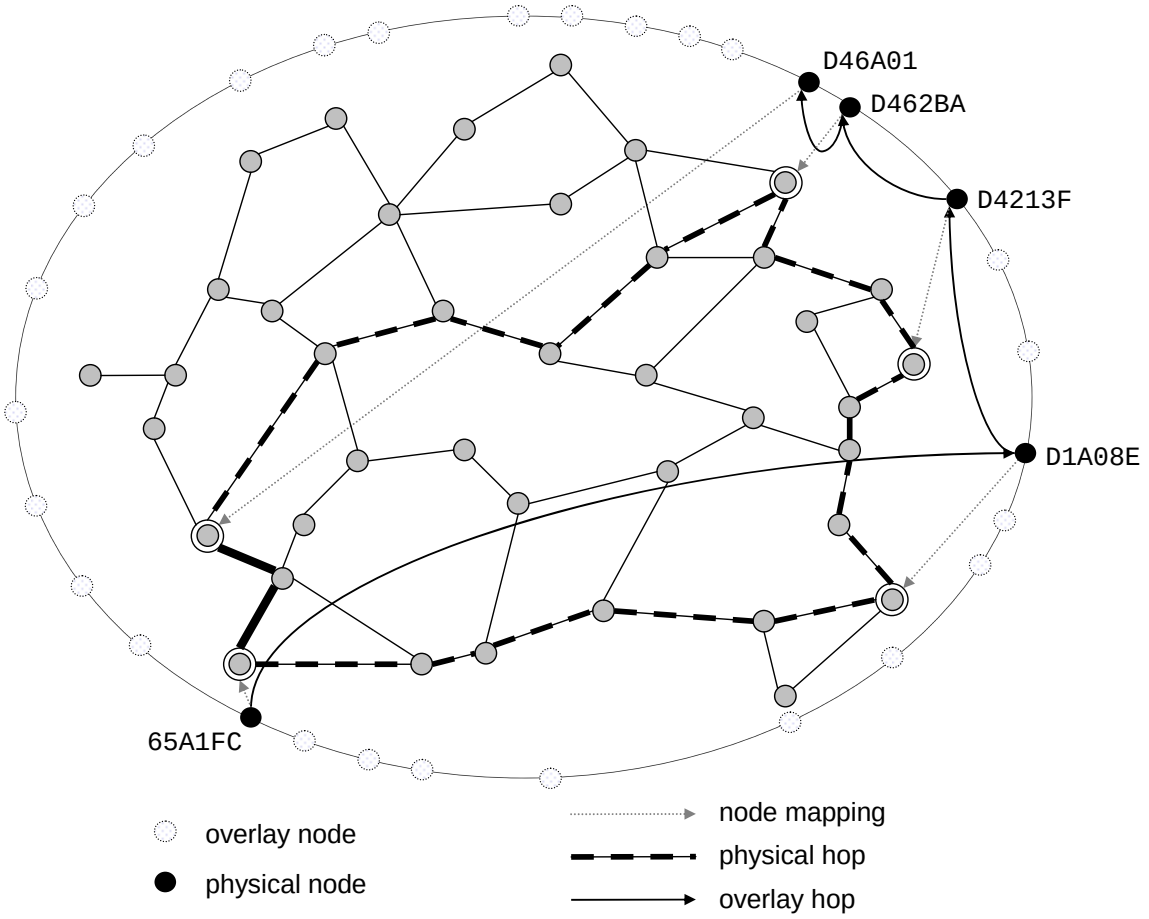

Figure 2.5 Overlay stretch.

reached its final destination. While with respect to the overlay network the packet is delivered very efficiently with only 4 hops, this example shows that in terms of the physical network the packet actually traverses the network twice to eventually reach its final destination that turns out to be only two physical hops away from the source node of this overlay lookup. In other words, the packet has traveled 17 physical hops during the overlay routing process when it could have reached its destination in 2 hops had the destination been known in advance. Therefore, the overlay stretch in this example is 17/2 = 8.5. It needs to be pointed out that this is obviously an extreme example whose purpose is to provide a clear illustration of the overlay stretch. In practice, most DHTs employ Proximity Neighbor Selection (PNS) to alleviate the overlay stretch and Pastry's overlay stretch is generally observed to be between 1.6 and 2.2 [5].

While overlay stretches can at best be considered suboptimal in the Internet where physical routing is practically taken for granted, they pose a severe problem in MANETs where each additional physical hop decreases the delivery probability of a packet. Therefore, several approaches have been proposed to alleviate the overlay stretch in structured overlay networks.

For example, in [46] the concept of *Landmarking* is proposed. Since nodes cannot be assumed to have exact positioning devices such as GPS available, the idea here is to introduce a fixed set of so-called landmark nodes into the network. This set of landmark nodes is known to all nodes and nodes will periodically measure

their distances – e.g. as given by the hop count – to those landmark nodes (usually by sending a simple PING message to each landmark node that will be answered with a PONG message). Nodes will then order the landmark nodes according to their distance to them. The intuition is now that nodes that share the same such landmark order are quite likely to be physically close to each other. Thus, nodes with the same landmark order are mapped into the same region of the overlay ID space – e.g. by assigning them coordinates in the same CAN region or by assuming overlay IDs with a common prefix in the Pastry overlay ID space.

Another example was proposed in [60]. Here, a joining node establishes its overlay ID by first measuring the distances to its physical neighbor nodes. These distances are then used to establish virtual springs between the nodes. The value needed to achieve the minimum energy state for the new node in that system of springs is used to assign the new node its overlay ID. However, in MANETs where a node's immediate physical neighborhood changes frequently, this approach can be expected to be quite volatile, generating quite a few ID reassignments to keep the system of virtual springs in check.

An extensive study of all proposed mechanisms to provide locality-awareness in structured networks would be beyond the scope of this work. Therefore, we will focus on a light-weight and efficient approach called *Random Landmarking* [63, 64, 65, 66, 72] as its general concept serves as a basis for this theses.

The goal of Random Landmarking (RLM) is to form clusters where physical close nodes share a common overlay ID prefix. Thus, two nodes that are physically close to each other will also likely be "close" to each other in the overlay. In order to achieve this goal, RLM is based on the general concept of Landmarking. However, fixed and stationary landmark nodes are usually not available in the context of mobile ad hoc networks. Since Random Landmarking was explicitly designed for its application in such MANETs, it does not assume the presence of fixed landmark nodes. Instead, it uses a set of *landmark keys*. A landmark key is simply an overlay ID. Rather than having dedicated landmark nodes, with RLM those nodes become temporary landmark nodes that are currently responsible for one of the landmark keys (i.e. whose own overlay identifiers are currently closest to one of the landmark keys as defined by the respective DHT). Therefore, when one of the current landmark nodes fails or resigns, another node (that whose overlay ID is *now* closest to the landmark key) will automatically assume its role.

Landmark keys should be chosen so that they divide the overlay ID space into equal-sized segments. For example, in a hexadecimal-based Pastry ID space, an appropriate set of landmark keys could be: 0800…000, 1800…000, 2800…000, ..., E800…000, F800…000.

To form clusters of common overlay ID prefixes, nodes associate themselves with the temporary landmark node that is currently closest to them (e.g. as determined by the hop count) by adopting its overlay ID prefix. For that purpose, each node periodically measures its distance to the temporary landmark nodes. This can easily be achieved by issuing a DHT lookup for the respective landmark keys. The temporary landmark nodes will then directly respond to the requester

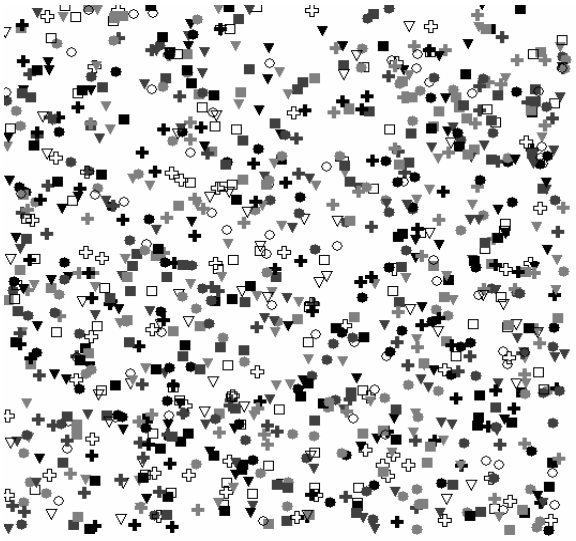

Figure 2.6 Spatial distribution of overlay ID prefixes in a Pastry network. Equal symbols and shades of grey represent equal overlay ID prefixes.

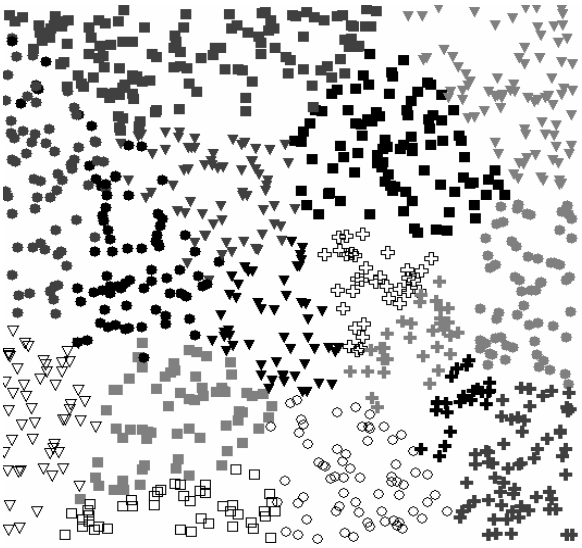

Figure 2.7 Spatial distribution of overlay ID prefixes in a RLM network. Equal symbols and shades of grey represent equal overlay ID prefixes.

and that node can use the hop count of the response as a distance metric to the various landmarks. Nodes periodically examine their landmark list to determine whether they have moved closer to a new landmark, i.e. whether they have moved – with high probability - into a new overlay cluster. If so, a node will assign itself a new random overlay ID with its new cluster's overlay ID prefix, resign from the overlay network with its old ID, and rejoin the overlay network with its new ID. The length of the ID prefix that a 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.

RLM was built into the DHT Pastry and thoroughly evaluated in various network settings. It was shown that, in static networks, RLM is able to significantly decrease the overlay stretch compared to standard Pastry. When additionally employing the same routing table optimization techniques that Pastry uses, RLM is even able to achieve an overlay stretch even below Pastry's optimal achievable overlay stretch. Furthermore, a Pastry network with RLM obtains its lower overlay stretches while generating up to 75% less overlay traffic in total compared to standard Pastry. Figure 2.6 and Figure 2.7 illustrate the effect that RLM has on the network topology. Both figures depict the spatial distribution of overlay ID prefixes where equal symbols and colors represent equal overlay ID prefixes. In a standard Pastry network, as shown in Figure 2.6, overlay ID prefixes are randomly distributed among the participating nodes. There is no correlation between overlay proximity (i.e. nodes that have a common overlay ID prefix and that are, thus, close to each other in the overlay ID space) and actual physical proximity of nodes. However, Figure 2.7 demonstrates that, with RLM, clusters form where physically close nodes share a common overlay ID prefix.

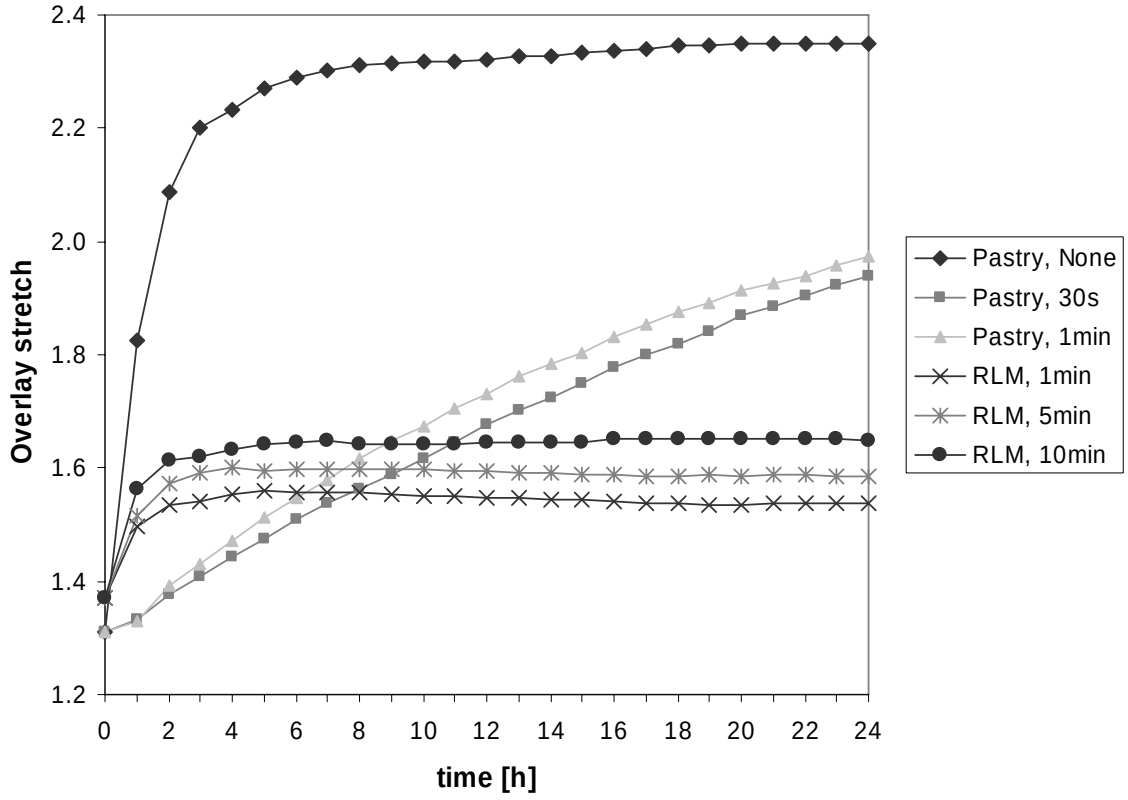

Figure 2.8 Overlay stretch for Pastry and RLM mobile networks.

Hence, with RLM, physically close nodes are also likely to be close to each other in the overlay ID space.

Random Landmarking was also evaluated in mobile networks. Figure 2.8 compares the performance of RLM and standard Pastry in a 2,000 node network where nodes moved according to the random waypoint mobility model with a speed of 0.6m/s and a pause time of 30s. Each simulation run lasted 24 simulated hours. During those 24h, 20,000 random lookups were issued, uniformly distributed among the 2,000 nodes. As can be seen, Pastry's overlay stretch quickly deteriorates, even when the Pastry nodes maintain their routing tables every 30s and 60s, and approaches the overlay stretch that Pastry displays without any routing table maintenance.

RLM, on the other hand, achieves a stable and decidedly lower overlay stretch than Pastry does with a comparable amount of messages exchanged when the landmark re-measure interval is 10 minutes. If one is willing to accept a message total above Pastry's total with a 1-minute routing table maintenance interval but still well below Pastry's 30s interval total, RLM's overlay stretch can be lowered even further with shorter landmark re-measure intervals of 1 and 5 minutes (see Figure 2.9).

So far, to evaluate RLM's general performance, ideal networks have been used where packets were always correctly delivered. Next, RLM was evaluated in ns-2 [37], simulating a complete 802.11 based physical network. Figure 2.10 shows the success rates in a 100 node network where, again, nodes move at a

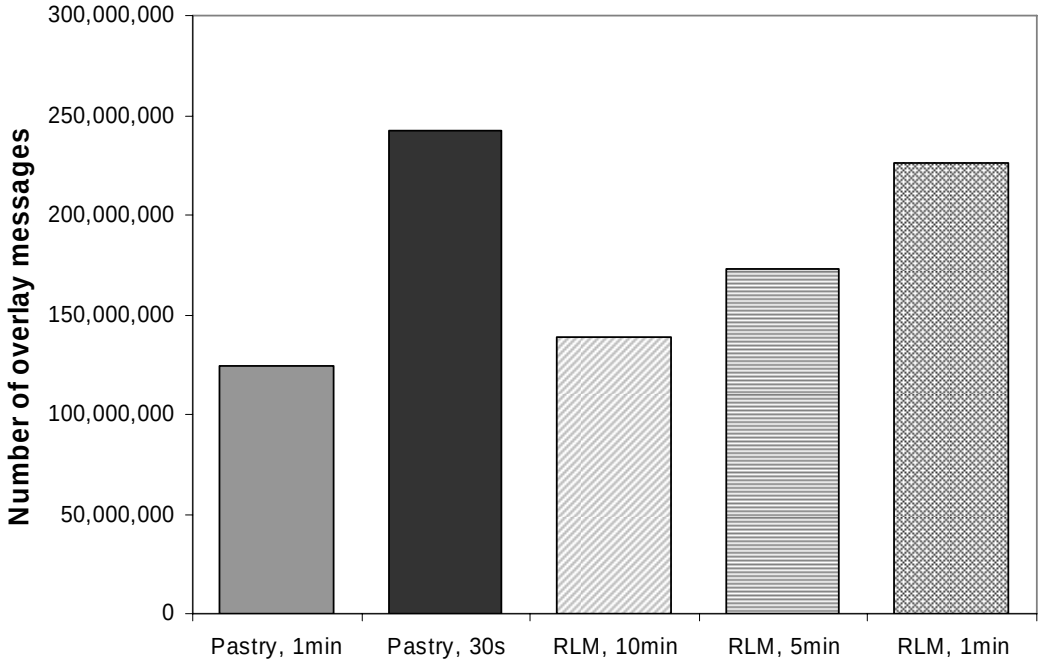

Figure 2.9 Total number of overlay messages exchanged during an average 24h simulation.

steady speed of 0.6m/s with a pause time of 30s. The Pastry and RLM overlay networks were deployed on top of AODV as network routing protocol. Each simulation run lasted one simulated hour during which each node periodically issued random key lookups.

The success rate is simply defined as the ratio between the total number of issued random key lookups and the number of those key lookups that were delivered to the correct destinations – i.e. to the nodes currently responsible for the respective keys. As can be seen, RLM can help improve the success rate slightly (from 55%

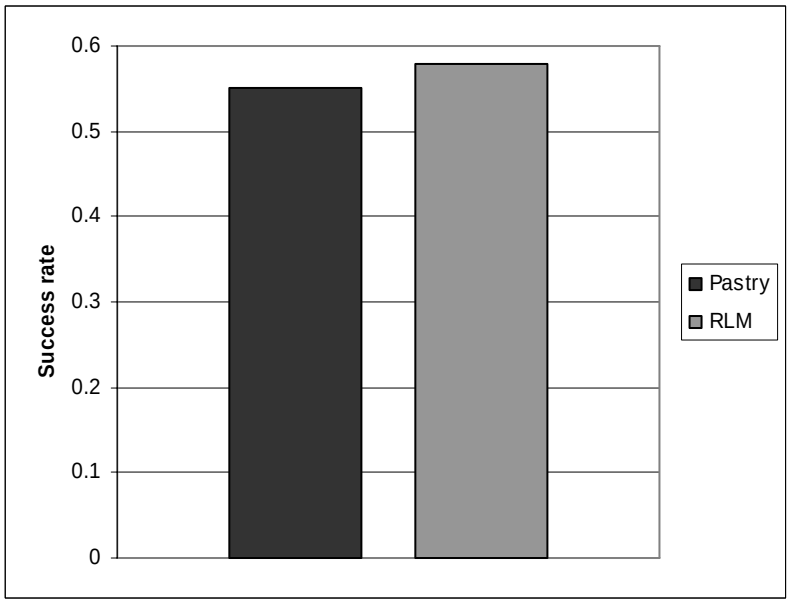

Figure 2.10 Success rate of Pastry and RLM in a 100-node AODV network.

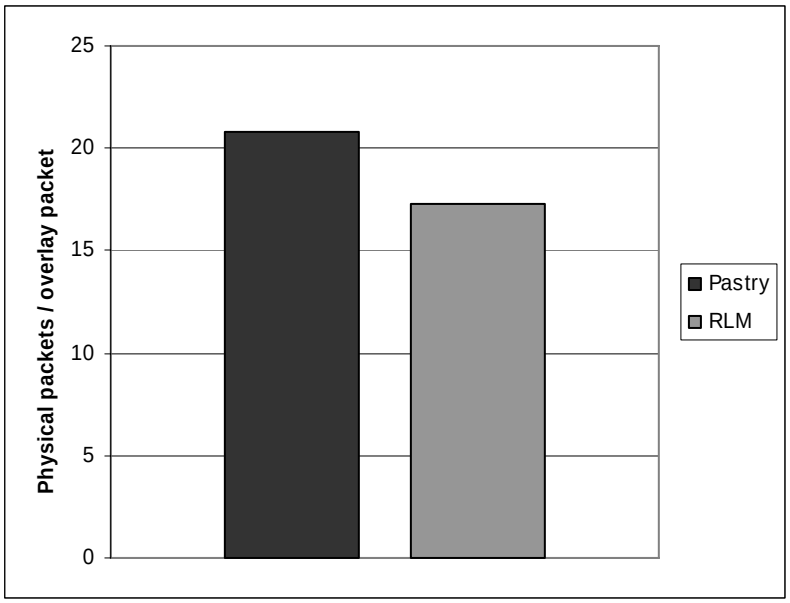

Figure 2.11 Average number of physical packets per overlay hop.

to ca. 58%). This is because, on average, RLM reduces the number of actual physical packets that are triggered by an overlay hop. For example, an overlay hop from node A to destination node B might trigger a network-wide broadcast of AODV route request packets if node A does not know a physical route to node B in order to perform the overlay hop.

Figure 2.11 depicts the average number of physical messages (such as AODV route requests, replies, etc.) that are triggered by an overlay hop. Using RLM, this figure can be reduced by approx. 20%. The question arises why this considerable reduction results in only a slightly improved success rate. The reason for this is that RLM's additional overlay messages such as landmark re-measurements, node rejoins, etc. will also trigger significant amounts of physical

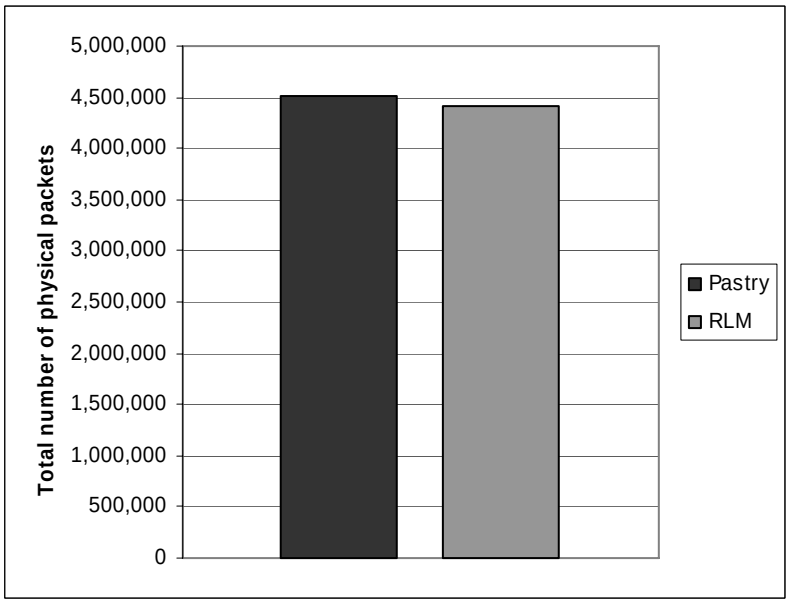

Figure 2.12 Physical traffic generated by Pastry and RLM.

messages. Those extra physical messages will, then, interfere (e.g. due to collisions, packet queue overflows, etc.) with other packets such as key lookup packets, which in turn will lower the success rate. Furthermore, not only will lookup packets be lost due to factors such as collisions but also RLM maintenance packets such as landmark re-measurement packets themselves. For example, unlike in ideal networks, a node might not hear from its closest landmark node due to packet loss and, therefore, might decide to join another cluster even though it has not actually moved out of its own cluster. This, of course, will result in an increased rejoin overlay traffic, which in turn can trigger additional physical messages and so forth. Accordingly, Figure 2.12 shows that, in a "real", non-idealized physical network, RLM generates only slightly less physical messages than standard Pastry does.

2.4 Summary

To overcome the scalability issues of the flooding-based unstructured peer-to-peer networks, Distributed Hash Tables have been proposed. DHTs impose a certain structure on their overlay networks to enable very efficient overlay routing. However, as DHTs are largely oblivious to the underlying network, the length of the accumulated physical route traveled during an overlay lookup process can be significantly greater than the length of the direct physical path from the source to the eventual target node. To alleviate this *overlay stretch*, mechanisms have been proposed to map physical proximity to the overlay ID space. For example, using *Random Landmarking*, one can significantly lower the overlay stretch in Pastry overlay networks on top of fixed or idealized mobile physical networks. However, when deployed on top of a non-idealized, "real" mobile ad hoc network, the packet delivery success rate even of RLM-enhanced overlay networks drops to unacceptably low levels (in the analyzed scenario to around 60%) as factors such as ad hoc route discoveries and packet collisions start posing a heavy burden on the efficacy of the overlay network. These findings corroborate our assumption that is does not suffice to merely deploy conventional DHTs (even with topology-awareness) on top of MANETs to provide the building blocks for distributed applications. Instead, we argue that, in order to provide efficient key-based routing in MANETs, it is necessary to integrate the concepts of ad hoc routing and DHT-based overlay routing at the network layer.