# 5 Experimental Results

This chapter evaluates the core performance of MADPastry. For this purpose, we implemented MADPastry as a routing agent in the popular network simulation environment ns-2.27 [37]. To put the performance of MADPastry into perspective, MADPastry is always compared against two other routing agents: a simple broadcast-based one and a Pastry-style DHT substrate without Random Landmarking.

The ns-2 routing agent implements the MADPastry protocol as described in Chapter 4. Nodes send out cluster beacons every 30s and ping their left and right leafs every 60s. 16 landmark keys are used in the simulations. Additionally, to further increase the success rate, the MADPastry routing agent of a lookup initiator always also issues a secondary, backup lookup. That backup lookup is first sent to the "left" or "right" leaf (depending on the packet's key) who will then regularly forward the backup lookup, which, thus, can be expected to take a (partially) different route to the eventual target node. If both lookups arrive at the eventual target, the second one is dropped. Please refer to Appendix 9.2 for a list of MADPastry's system parameters and their values used throughout this section.

The fundamental question to be answered when deploying a DHT substrate in MANETs is whether the extra overhead of maintaining the DHT structure is really worth the effort. Or, is the benefit gained from using a DHT so miniscule that we would have, indeed, been better off just broadcasting the lookups in the first place. Therefore, we also implemented a Gnutella-style broadcast routing agent. The broadcast agent maintains no overlay structure and, thus, has no extra maintenance overhead. It broadcasts a packet to all its one-hop neighbors who, then, forward the packet to all their one-hop neighbors and so forth. Nodes keep track of the packet sequence number so that already forwarded packets will not be sent a second time.

Due to mobility, nodes in a MADPastry network will eventually change their cluster memberships. This means that nodes might repeatedly assign themselves new overlay IDs. The subsequent reorganization (leaving, rejoining, coping with invalid overlay identifiers, etc.) can generate a sizeable amount of traffic. To verify whether MADPastry's extra overhead stemming from these cluster changes is justified, we also implemented a routing agent that integrates regular Pastry and AODV. It works very similar to MADPastry except that it does not employ *Random Landmarking*. Thus, there are no physical clusters of nodes sharing a common overlay ID prefix and, thus, there is no overlay ID reassignment – i.e. leaving and rejoining the network – either. Since Pastry's standard routing table and leaf set maintenance can be prohibitively expensive in MANETs, the integrated Pastry routing agent, too, only fills its routing table by forwarding and overhearing live packets and also only pings its left and right leaf

proactively. This, in fact, closely resembles the related system Ekta [42] (also see Section 3.3.2). Furthermore, beacons as well as lookups for which no physical route is known are broadcast throughout the entire network – as there are no clusters. Also, the integrated Pastry routing agent does not issue any secondary lookups (as the MADPastry routing agent does) since its overhead is already drastically higher than MADPastry's – as the simulation results will show.

Again, MADPastry is a routing agent, not an application as such. Therefore, for the simulations, a simple random traffic generator was implemented as application running on top of one of the three routing agent. With this application, each node periodically sends a packet with a random key (i.e. starts a random lookup) to whichever node is currently responsible for the packet's key.

To compare the performances of the three routing agents, the following metrics are analyzed:

**Success Rate.** This represents the percentage of random lookups that are eventually delivered to the correct responsible node.

**Packet Overhead.** This is the total number of packets that are forwarded during the entire simulation. This count is increased whenever a node forwards a packet to the next physical hop. In the case of MADPastry and MADPastry without clusters, this figure comprises *all* router and application packets that are created by a node: lookups, leaf pings/pongs, join requests (only MADPastry), join replies (only MADPastry), leave messages (only MADPastry), node beacons, route requests, route replies, etc. In other words, this count is increased whenever the MAC layer of a node is being passed a packet down from an upper layer. In the case of the Gnutella-style broadcast router, this figure only consists of lookups as there simply are no maintenance messages.

**Overall Traffic.** This figure counts the total network traffic in Kbytes that is generated during the entire simulation. Whenever a node forwards a packet, this figure is increased by the packet size – i.e. whenever the MAC layer of a node receives a packet from an upper layer. Again, this figure includes *all* router and application packet types for MADPastry (with and without clusters). Here, it is important to mention that MADPastry packets on average are about 4 times larger (excluding the IP header) than the corresponding broadcast agent's packets as they carry additional information such as the last hop's overlay ID and so forth.

All simulations that were carried out modeled wireless networks over the course of one (simulated) hour. Nodes are always moving around according to the random way point model with 0s pause time and at a constant speed. For data transmission, nodes are using the 802.11 communication standard with a transmission range of 250m. Furthermore, a 32-bit overlay ID space is assumed with hexadecimal overlay IDs. In other words, each overlay ID consists of 8 hexadecimals digits.

**Table 5.1 Simulation parameters and values.**

| Simulation Parameter | Value range |
|---|---|
| Simulation duration | 3600s (simulated) |
| Network size | 100 and 250 |
| Network density | 100 nodes/km² |
| Node mobility | 1.4m/s (constant, 0s pause time) |
| Random lookup interval (per node) | 10s |

## 5.1 Basic Results

In the first set of simulations, the performances of MADPastry, the MADPastry routing agent without clusters and the Gnutella-style broadcast agent are compared in networks of 100 and 250 nodes. In all simulations, square planes are used with a node density of 100 nodes/km². Nodes are moving around at a constant speed of 1.4 m/s, which corresponds to a fast walking speed. For this first set of simulations, the random lookup application of each node sends out a random key lookup every 10s. Note that, after the start of the simulation, each node's random lookup application commences after a uniform random delay between 0s and 10s so as to avoid a traffic pattern consisting of lookup bursts every 10s. For the 100-node network, MADPastry uses 8 landmark keys. Table 5.1 provides an overview of the chosen simulation parameters and their

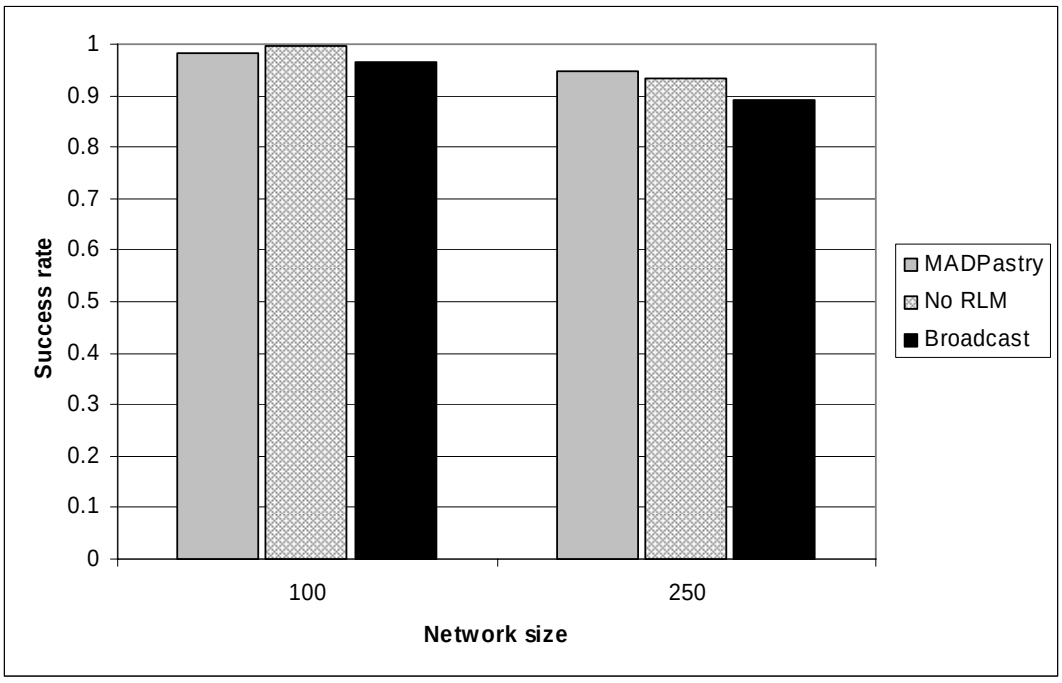


**Figure 5.1 Success rates of the respective routing agents – 1.4m/s.**

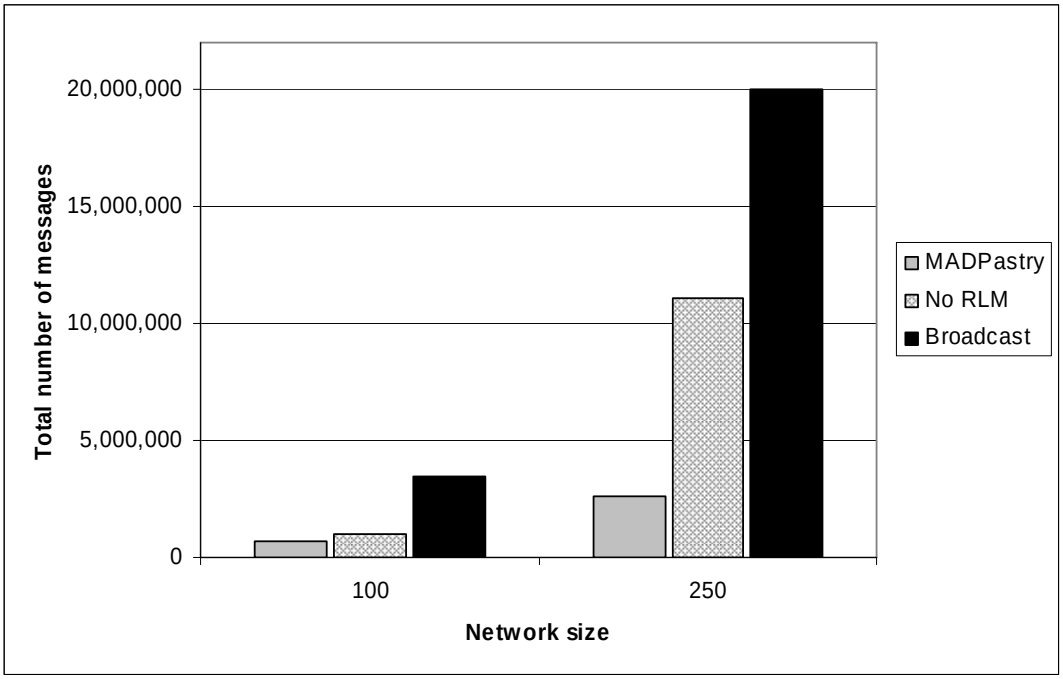


**Figure 5.2 Total number of messages - 1.4m/s.**

respective values.

Figure 5.1 shows the success rate of the three routing agents for the random lookups. As can be seen, MADPastry achieves better success rates in both 100 and 250-node networks compared to the broadcast agent. Furthermore, MADPastry retains success rates of well above 90% for both network sizes, whereas the broadcast agent's rate drops below 90% in a 250-node network. The success rate of the MADPastry router without clusters ("No RLM") is practically the same as MADPastry's (slightly higher in a 100-node network and slightly

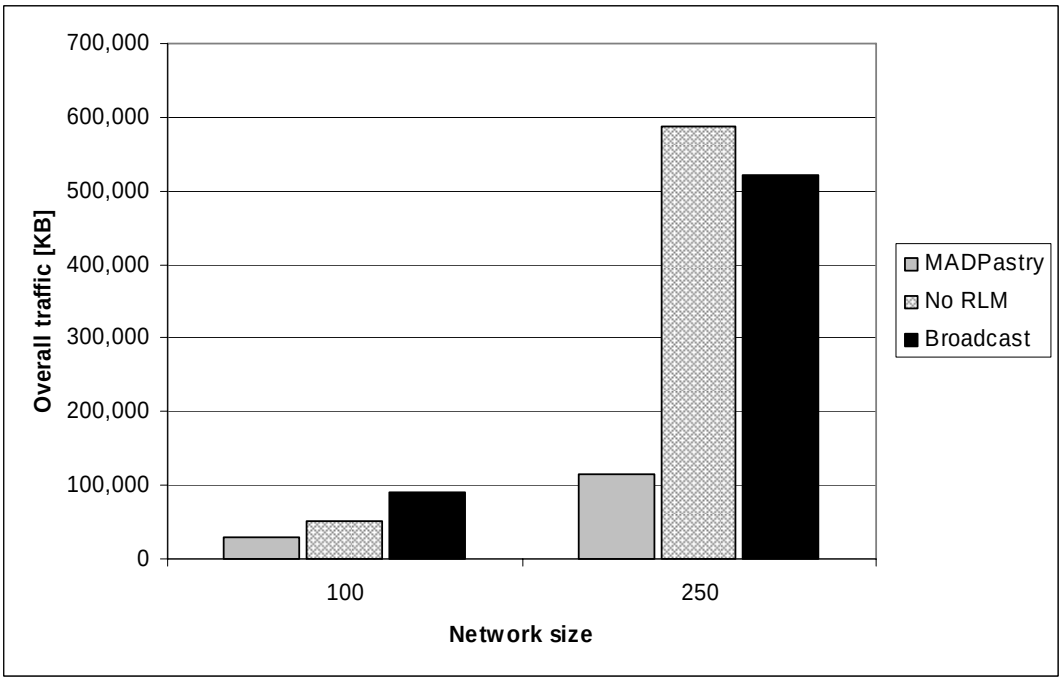


**Figure 5.3 Overall generated traffic in Kbytes - 1.4m/s.**

lower in a 250-node network).

Figure 5.2 shows the total number of messages that the routing agents send or forward during the simulated hour in order to achieve their respective success rates. Again, these figures include all router and application messages that the nodes' MAC layers receive from above. It becomes clear that MADPastry produces drastically less network traffic than the Gnutella-style broadcast agent does. In a 100-node network, the broadcast router needs about 5 times and in a 250-node network even about 7 times the number of messages that MADPastry needs. The MADPastry router without clusters ("No RLM") incurs roughly 1/3 of the message traffic of the broadcast agent in a 100-node network and roughly 1/2 in a 250-node network, which is well above MADPastry's message traffic.

However, it is important to bear in mind that a MADPastry packet header is longer than that of the broadcast router due to the extra information included in it (see 4.5). To make sure we are not comparing apples and oranges, Figure 5.3 shows the traffic in forwarded Kbytes instead. Again, these figures include all router and application messages that the nodes' MAC layers receive from above. Even with this metric, MADPastry still produces several times less traffic than either the broadcast router or the MADPastry router without clusters. An interesting observation can be made here for the MADPastry router without clusters ("No RLM"). While still below the broadcast agent's overhead in a 100-node network, its overall traffic becomes larger than the broadcaster's in a 250-node network. This can easily be explained by the fact that Pastry's overlay routing usually requires several overlay hops per lookup. Since there are no clusters, successive overlay hops can crisscross the physical network. Furthermore, when the MADPastry router without clusters has to resort to broadcasting a lookup (because the physical route to carry out the next overlay hop is unknown), the lookup could already have crossed the network several times. Obviously, one would have been better off if one had broadcast the lookup right away – which is exactly what the broadcast agent does. Furthermore, even if the lookup could be delivered without being broadcast (i.e. the routes for all overlay hops involved were known), the accumulated physical path lengths of the overlay hops might only be slightly more light-weight than a broadcast. Additionally, the required periodic beacon broadcasts are added on top. Since both physical and overlay paths are much shorter in a 100-node network, this effect is less pronounced there.

This is further confirmed by Figure 5.4. It displays the overlay stretch as generated by the random lookups in both 100 and 250-node networks. Note that, trivially, the overlay stretch of the broadcast router is always at the optimum of 1.0. This is simply due to the fact that, here, the lookups are always broadcast throughout the network. Thus, the first copy (of possibly many copies) of a particular lookup will arrive at the destination node on the shortest path from the source. For MADPastry, Figure 5.4 shows that, even in a small network, MADPastry achieves a smaller overlay stretch than the MADPastry router without clusters does – 1.33 vs. 1.46. Practically speaking, this means that the accumulated path length during a MADPastry lookup will be 33% longer than the direct path from the initiator of the lookup to the eventual target node. Without the utilization of cluster, however, this figure will already deviate by

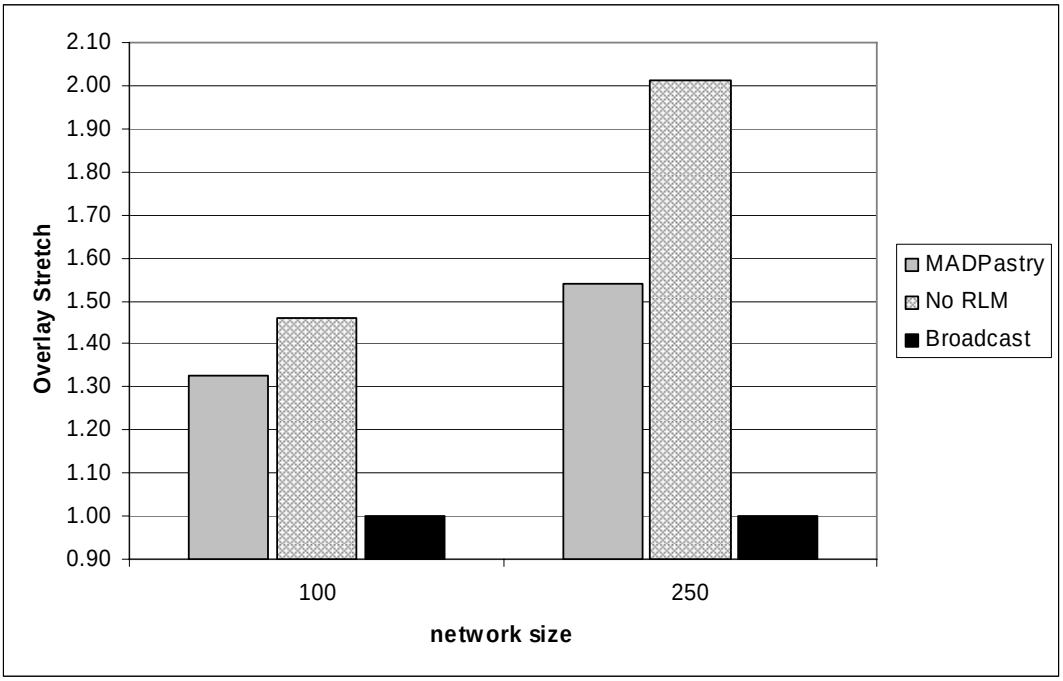


**Figure 5.4 Overlay stretch.**

46% from the direct path. In other words, the key-based routing without clusters will generate approx. 10% more traffic than routing with clusters does for *each* lookup. This effect becomes even more pronounced in larger networks (1.54 vs. 2.01) where physical routes between two arbitrary nodes also become longer.

This clearly demonstrates the benefits for DHT substrate in MANETs to consider physical locality – as MADPastry does.

## 5.2 Load Distribution

As just seen in the previous section, due to its consideration of physical locality using RLM, MADPastry achieves comparable or better success rates than a Pastry-based routing agent without RLM does while producing significantly less overall traffic. The question that arises is whether MADPastry attains this efficacy at the expense of an uneven distribution of the node traffic load. It is, indeed, conceivable that, in a MADPastry network, a situation could occur where a relatively small and isolated group of nodes form an overlay ID cluster. The size of the segment of the overlay ID space, that a member node of a cluster is responsible for, is on average given by the overlay ID range of the cluster (i.e. the range of overlay IDs that start with the given cluster prefix) divided by the number of cluster members. Therefore, in the case of a relatively small (in terms of the number of member nodes) cluster, the nodes of that cluster would be responsible for disproportionately large segments of the overlay ID space and might, thus, have to handle a disproportionate amount of traffic load. In other words, the overlay ID distribution in a MADPastry network is no longer strictly uniform but dependent on the spatial distribution of the nodes themselves.

**Table 5.2 Individual accumulated node traffic load.**

| | MADPastry | No RLM |
|---|---|---|
| Minimum individual accumulated load (in bytes) | 2,110,429 | 16,828,793 |
| Maximum individual accumulated load (in bytes) | 5,054,556 | 29,009,904 |
| Ratio | 2.40 | 1.73 |

To evaluate how severely MADPastry deviates from an ideal, uniform load distribution, the individual accumulated load during a simulation run was recorded for each node. Whenever a node received a packet (AODV, Pastry, application), the individual accumulated load count of that node was increased by the length of the received packet. Table 5.2 provides the average (over all simulation runs) minimum individual accumulated node load and the average maximum individual accumulated node load encountered in the 250-node networks from the previous section. Note that, in a mobile ad hoc network, a single sent packet (as counted in Figure 5.2 and Figure 5.3) can be received and processed by multiple nodes – generally by all nodes in the sender's transmission range – so that each sent packet can potentially increase the individual accumulated load of several nodes.

Two observations can be made in Table 5.2. First of all, one notices that nodes in a network that does not employ RLM have to, on average, handle around 6 times as much load as nodes in a MADPastry network do. This corresponds with the observation made in Figure 5.3. Again, the reason for this is that the router

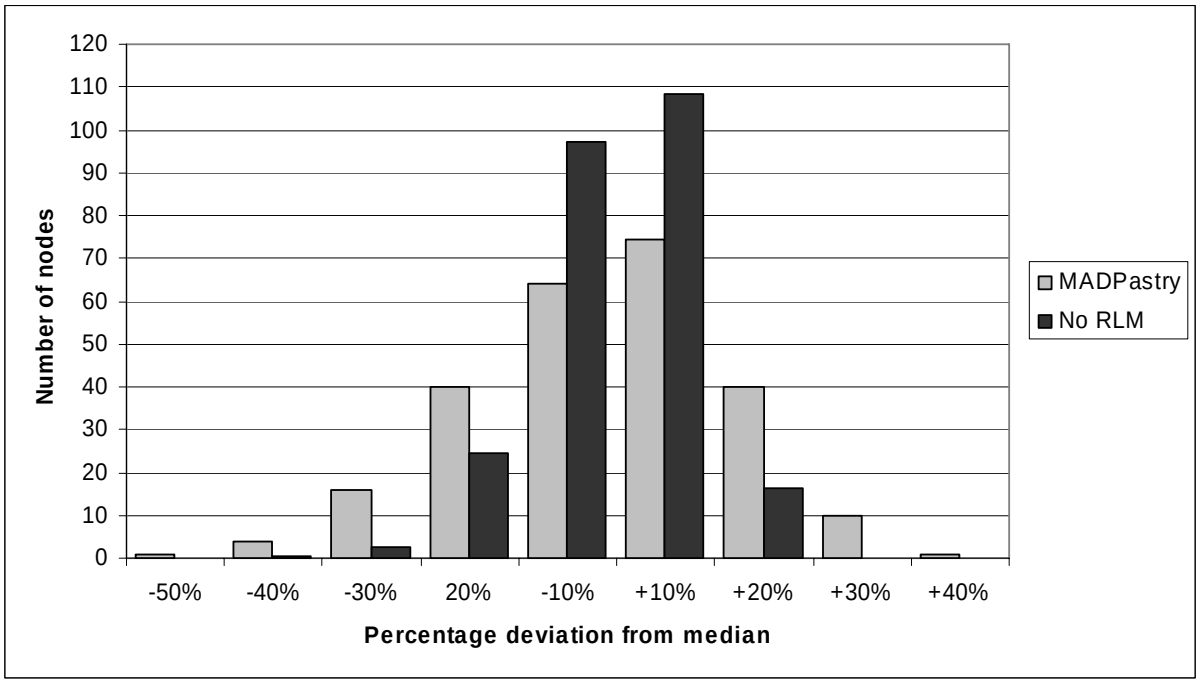


**Figure 5.5 Percentage deviation of the individual accumulated node loads from the median.**

without RLM needs to issue significantly more network-wide broadcast messages (such as AODV route discoveries) that, in turn, are received and processed by numerous nodes. Secondly, it can be seen that MADPastry's load distribution is somewhat less optimal than the distribution achieved without RLM. Whereas, without RLM, the node with the highest individual accumulated load has to, on average, handle 1.73 times the amount of traffic that the node with the lowest individual accumulated has to handle on average, this ratio is mildly higher at 2.40 in a MADPastry network with RLM.

Figure 5.5 presents a different view of the individual accumulated node load. It depicts the percentage deviation and the quantity of the individual accumulated node loads from the median in the 250-node network from the previous section. In other words, its shows the average number (over 10 simulation runs) of nodes whose individual accumulated loads were within a range of 0%-10% (both smaller and larger), 10%-20%, 20%-30%, etc. of the median. As could be expected from Table 5.2, MADPastry's load distribution fans out somewhat compared to the router's without RLM but otherwise resembles it closely: While in a network without RLM 98% of the nodes need to handle loads that do not deviate by more than 20% from the median, this is still true for 88% of all nodes in a MADPastry network.

In order to consider physical locality in its overlay structures, MADPastry deliberately sacrifices an ideal, uniform overlay ID distribution, which results in a mildly more fanned-out load distribution compared to a network that does not employ RLM. However, we strongly believe that the expense of having a somewhat less optimal load distribution ( a maximum/minimum load ratio of 2.40 vs. 1.73) is practically negligible compared to the advantage of achieving comparable and better success rates with individual node loads that are around *6 times* less than those accumulated without RLM.

## 5.3 Node Velocity

In the first set of simulations, nodes were moving at a constant speed of 1.4 m/s.

**Table 5.3 Simulation parameters and values – varying node velocities.**

| Simulation Parameter | Value range |
|---|---|
| Simulation duration | 3600s (simulated) |
| Network size | 250 |
| Network density | 100 nodes/km² |
| Node mobility | 0.1, 0.6, 1.4, 2.5, 5.0m/s (constant, 0s pause time) |
| Random lookup interval (per node) | 10s |

Of course, the node velocity can be expected to have a significant impact on the performance a routing agent. Therefore, in the next set of simulations, 250-node networks will be examined with varying node velocities: 0.1 m/s (practically stable network), 0.6 m/s (slow walking speed), 1.4m/s (fast walking speed), 2.5 m/s and 5.0 m/s. A request frequency of one random lookup every 10s per node will be used. Table 5.3 provides an overview of the simulation parameters and their respective values.

Figure 5.6 shows the success rates of the three routing agents in reference to the different node velocities. One can see that both MADPastry and MADPastry without clusters ("No RLM") achieve better success rates than the Gnutella-style router does for speeds up to a fast walking speed (1.4 m/s). At a speed of 2.5 m/s, the success rates of MADPastry and MADPastry without clusters start falling below the broadcast router's. The reason for this is that, with fast speeds, routes break so frequently that MADPastry without clusters can no longer keep its routing table and leaf set sufficiently valid – hence its success rate drops below the broadcast agent's success rate. With MADPastry this problem is further aggravated by the fact that nodes move from cluster to cluster so rapidly that a) they spend a significant amount of their time leaving and rejoining the network, and thus b) their overlay routing tables frequently contain stale entries. With stale entries, the likelihood that a packet might have to be sent back to the previous overlay hop (see Section 4.4) so that an alternative and ideally valid next overlay hop destination can be chosen increases. This, of course, will result in larger overlay stretches, which in turn decreases the probability of a successful packet delivery.

Figure 5.7 shows the total number of messages (packets) produced by the routing agents and the application during an average simulation run. Again, these figures include all router and application messages that the nodes' MAC layers

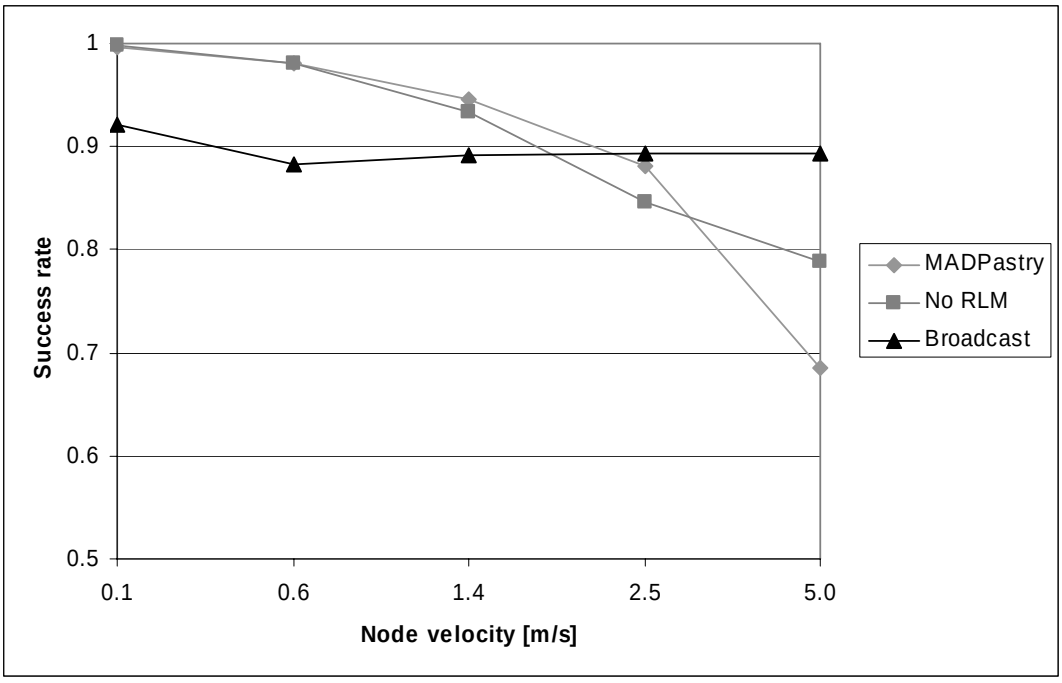


**Figure 5.6 Success rates vs. node velocity.**

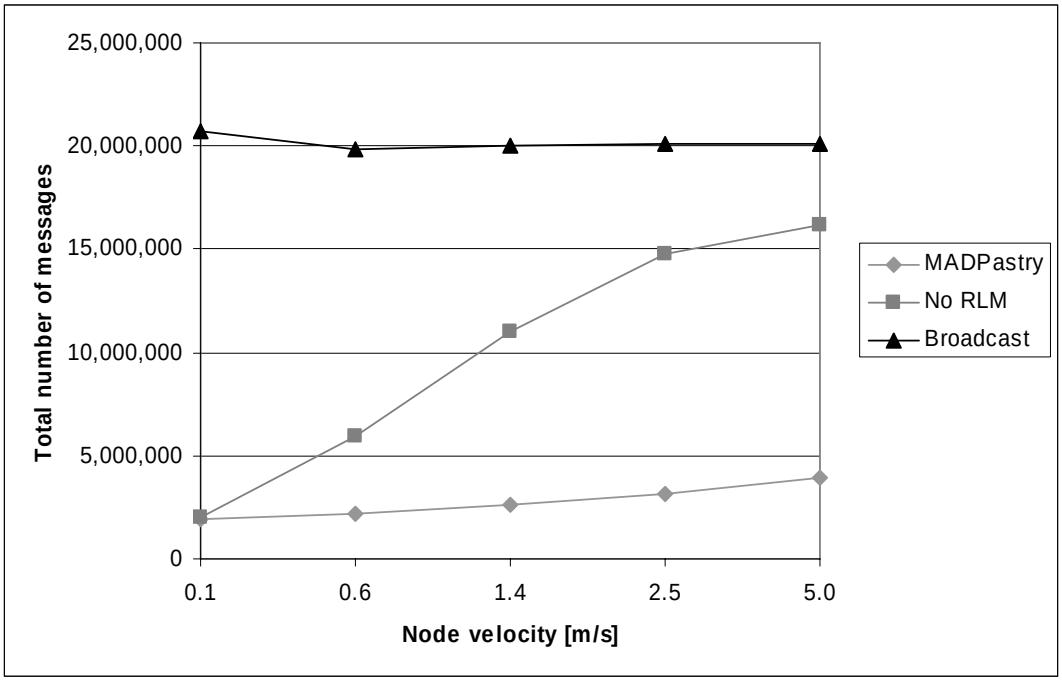


**Figure 5.7 Total number of messages vs. node velocity.**

receive from above. As can be expected, the overhead of the broadcast agent is practically independent of the node velocity. As can be seen, MADPastry produces significantly less packets for all considered node velocities than the other two routing agents do since broadcasts in MADPastry are restricted to their respective cluster.

Furthermore, Figure 5.8 also demonstrates that MADPastry's overall traffic stays significantly below that of the other two routing agents. For Pastry without clusters, the overall traffic quickly surpasses even that of the broadcast agent as

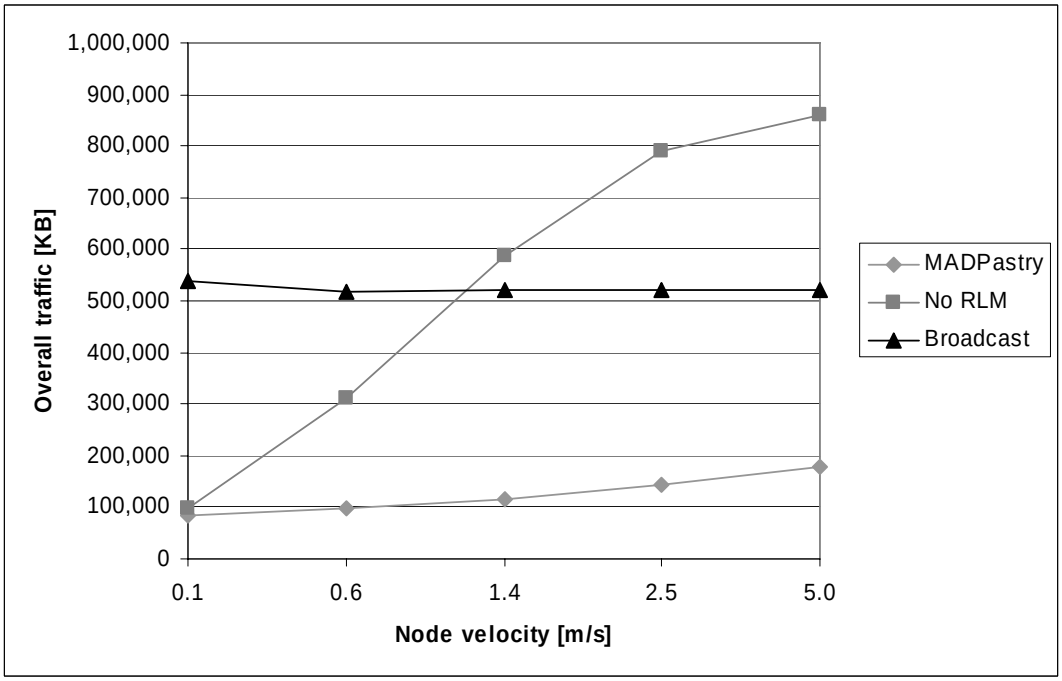


**Figure 5.8 Overall traffic vs. node velocity.**

**Table 5.4 Simulation parameters and values – varying lookup rates.**

| Simulation Parameter | Value range |
|---|---|
| Simulation duration | 3600s (simulated) |
| Network size | 250 |
| Network density | 100 nodes/km² |
| Node mobility | 1.4m/s (constant, 0s pause time) |
| Random lookup interval (per node) | 1s, 10s, and 60s |

route failures occur more and more frequently and the effects described in Section 5.1 become ever more pronounced.

## 5.4 Lookup Rates

All simulations thus far all have assumed a node lookup rate of one lookup per 10s. Next, the impact of the lookup rate on the overall performance will be evaluated. The following lookup intervals will be examined: 1s, 10s, and 60s in a 250-node network with a node velocity of 1.4 m/s. Table 5.4 provides an overview of the simulation parameters and their respective values. Note that for the lookup interval of 1s, MADPastry does not issue any backup lookups, whereas for the interval of 60s, two backup lookups are employed.

Figure 5.9 shows the success rates of the three routing agents in reference to the

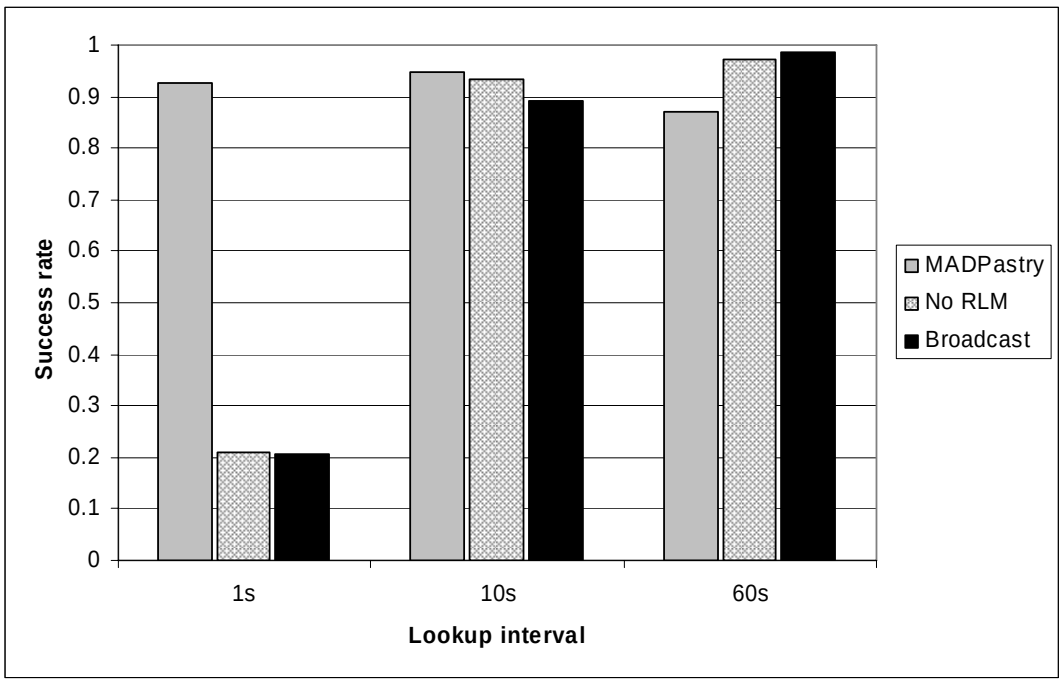


**Figure 5.9 Success rates vs. lookup intervals.**

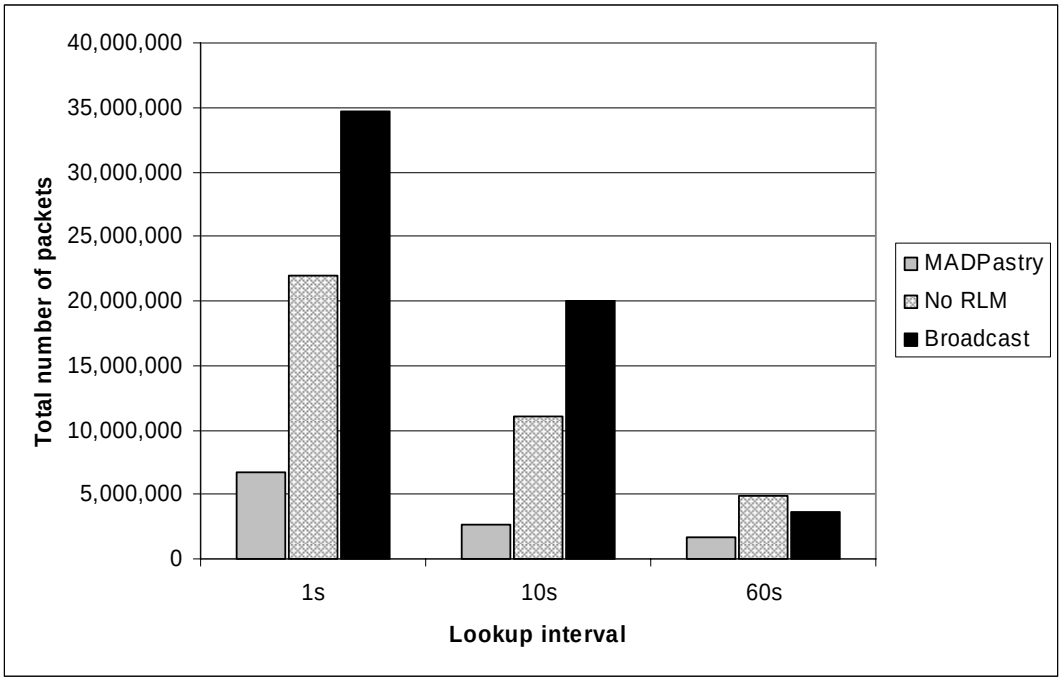


**Figure 5.10 Number of packets vs. lookup interval.**

lookup rate. As already seen in Section 5.1, MADPastry and MADPastry without clusters ("No RLM") achieve comparable success rates well above 90% for a per-node lookup interval of 10s. The Gnutella-style router's success rate here drops below 90%.

A very interesting observation can be made in networks with high lookup rates of 1 lookup per second per node. At such high lookup rates, both the broadcast agent and MADPastry without clusters can no longer keep up with MADPastry. Their (frequent) network-wide broadcasts of the lookup requests or route discoveries

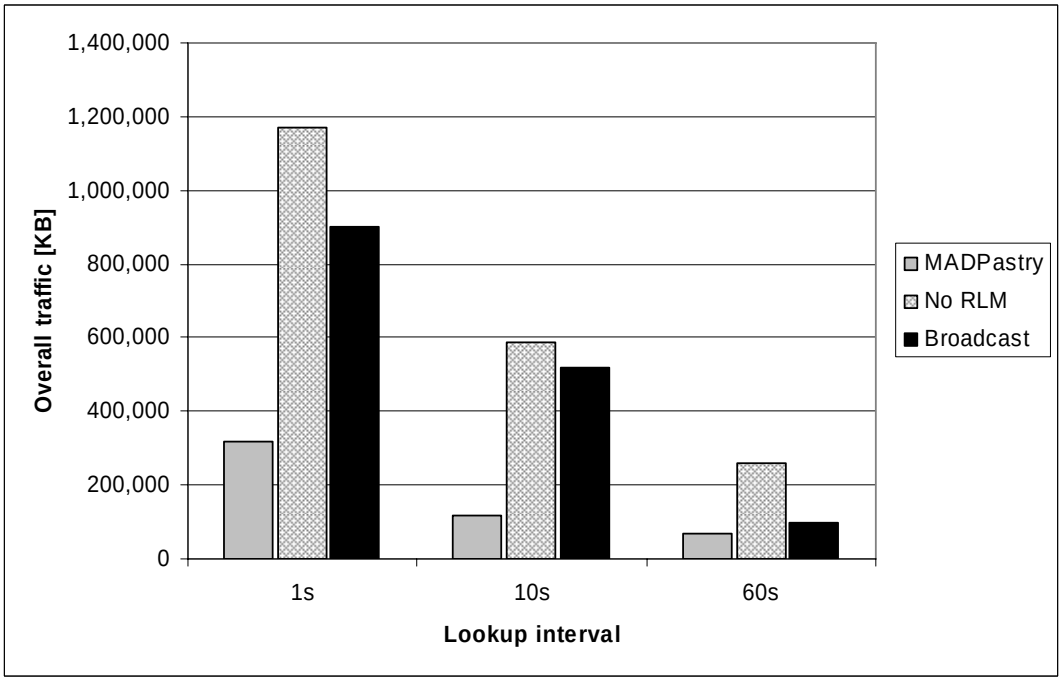


**Figure 5.11 Overall traffic vs. lookup intervals.**

clearly overwhelm the wireless physical network, resulting in so many packet collisions that the majority of lookups fail to be delivered. Thus, their success rates drop to 20%. On the other hand, MADPastry's physically shorter overlay hops (compared to MADPastry without clusters) and its local cluster broadcasts allow it to still maintain a success rate of 92% in the presence of such high lookup rates. Again, MADPastry's overall traffic remains significantly below that of the broadcaster and MADPastry without clusters both in terms of the total number of packets sent and the generated traffic, as Figure 5.10 and Figure 5.11 show.

On the other hand, if there is only one lookup per minute, MADPastry's lookup rate falls below 90% (87%). This is due to fact that the nodes overhear much less packets with which to update their routing tables. Furthermore, nodes often do not detect other nodes' cluster changes, which can result in packets being routed to stale overlay addresses. However, we believe that a request rate of one lookup per minute is too low to justify the effort of maintaining a DHT in the first place. When nodes only issue one lookup per minute, they might just as well broadcast their occasional requests and not bother to maintain a DHT structure.

## 5.5 Churn

Thus far, the scenarios considered have assumed that all nodes participate in the network during the entire simulated hour. However, in mobile ad hoc networks, it can occur that mobile nodes abruptly leave the network because they might have drained their batteries or moved out of the transmission range of other nodes – to name but a few factors. This dynamic and erratic joining and leaving of nodes is often referred to as *churn*.

This section examines how MADPastry can adapt to various churn rates in the network. For this purpose, it is necessary to first understand how churn is modeled in the following experiments. Since the dynamic creation and removal of nodes during simulation runtime is not directly supported in ns-2, the modeling of churn in ns-2 is a non-trivial task. Therefore, we use the following churn model. At the beginning of a simulation, each node is assigned a random uptime

**Table 5.5 Simulation parameters and values – varying churn rates.**

| Simulation Parameter | Value range |
|---|---|
| Simulation duration | 3600s (simulated) |
| Network size | 250 |
| Network density | 100 nodes/km² |
| Node mobility | 1.4m/s (constant, 0s pause time) |
| Random lookup interval (per node) | 10s |
| Node uptime intervals | (60-3600), (600-1200), (300-600), (60-300)s |

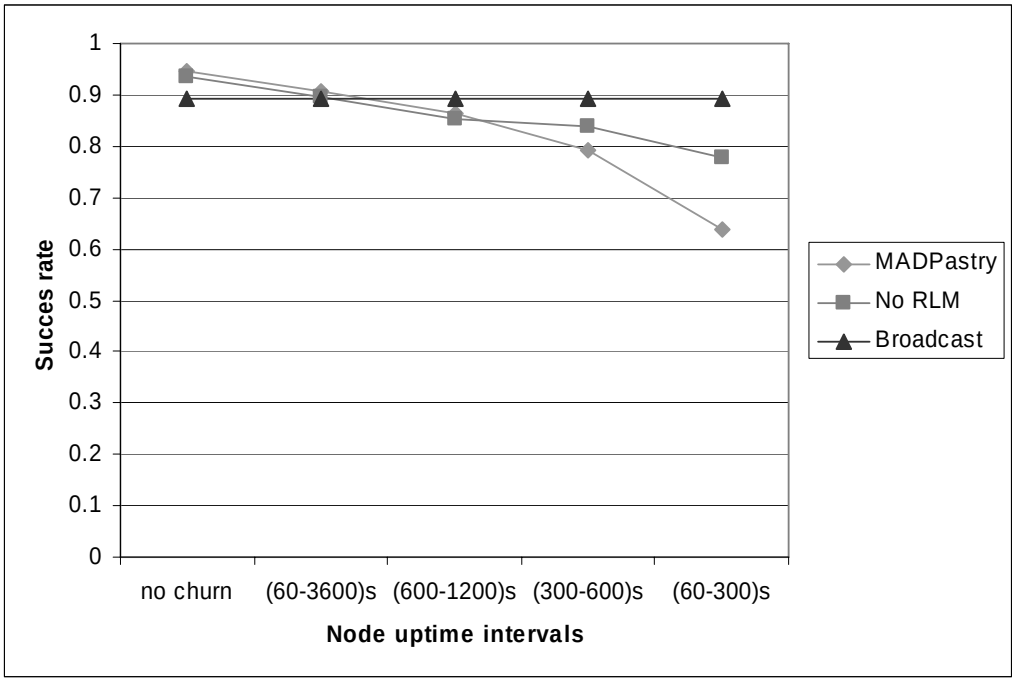


**Figure 5.12 Success rates vs. churn rates.**

after which the node will fail and leave the network abruptly. When a node fails, it will at once clear all its routing state (i.e. AODV routing table, Pastry routing structures, etc.) and immediately move to a random position in the network at a speed of 999,999 m/s. There, after overhearing packets from its one-hop neighbors, it will use one of those neighbors as bootstrap node to rejoin the network under a new overlay ID (but retaining its original node (IP) address). After the node has, thus, rejoined the network, it will be assigned a new random uptime and restart its random waypoint node movement. This way, the number of participating nodes in the network is kept constant. Note that ns-2 does not support the dynamic changing of node addresses (i.e. IP addresses) during simulation. Thus, it could happen that a node receives a packet that is destined for it under one of its former overlay IDs. In such a case, the node will simply return the packet to its sender – i.e. to the previous overlay destination – stating its new overlay ID.

For the churn experiments, we choose the 250-node network with a constant node velocity of 1.4 m/s from the previous sections and a request rate of 1 request per 10s. To examine various churn rates, the following node uptime intervals are considered:

- (60-3600)s – after (re-)joining the network, each node is assigned a randomly chosen uptime between 60s and 3600s. This represents a *mild* churn rate.

- (600-1200)s – after (re-)joining the network, each node is assigned a randomly chosen uptime between 600s and 1200s. This represents a *medium* churn rate.

- (300-600)s – after (re-)joining the network, each node is assigned a randomly chosen uptime between 300s and 600s. This represents a *high* churn rate.

- (60-300)s – after (re-)joining the network, each node is assigned a randomly chosen uptime between 60s and 300s. This represents a *very high* churn rate.

Table 5.5 provides an overview of the simulation parameters and their respective values.

Figure 5.12 shows the success rates that MADPastry and the MADPastry routing agent without Random Landmarking achieve in under the various churn rates. Note that the success rate of the broadcast router from Figure 5.1 is included as reference line as the broadcast router would not be affected by churn. As can be expected, the success rates of both MADPastry and the DHT router without RLM start declining the higher the churn rate becomes. Furthermore, MADPastry achieves slightly better success rates (over 90%) than the DHT router without RLM does for mild and medium churn rates. For higher churn rates, however, it becomes more and more difficult for MADPastry to maintain its clusters. The frequent node failures trigger an ever increasing cluster reorganization process, which results in MADPastry's lower success rates compared to the router without RLM.

Figure 5.13 depicts the total amount of network traffic generated by the respective routing agents under the various churn rates. Again, for the broadcast router, the amount from Figure 5.3 is included as reference line. As can be seen, with an increasing churn rate, MADPastry produces more and more network traffic. This is due to two factors. First of all, the higher the churn rate, the more often node swill fail and rejoin the network under a new overlay ID. Trivially,

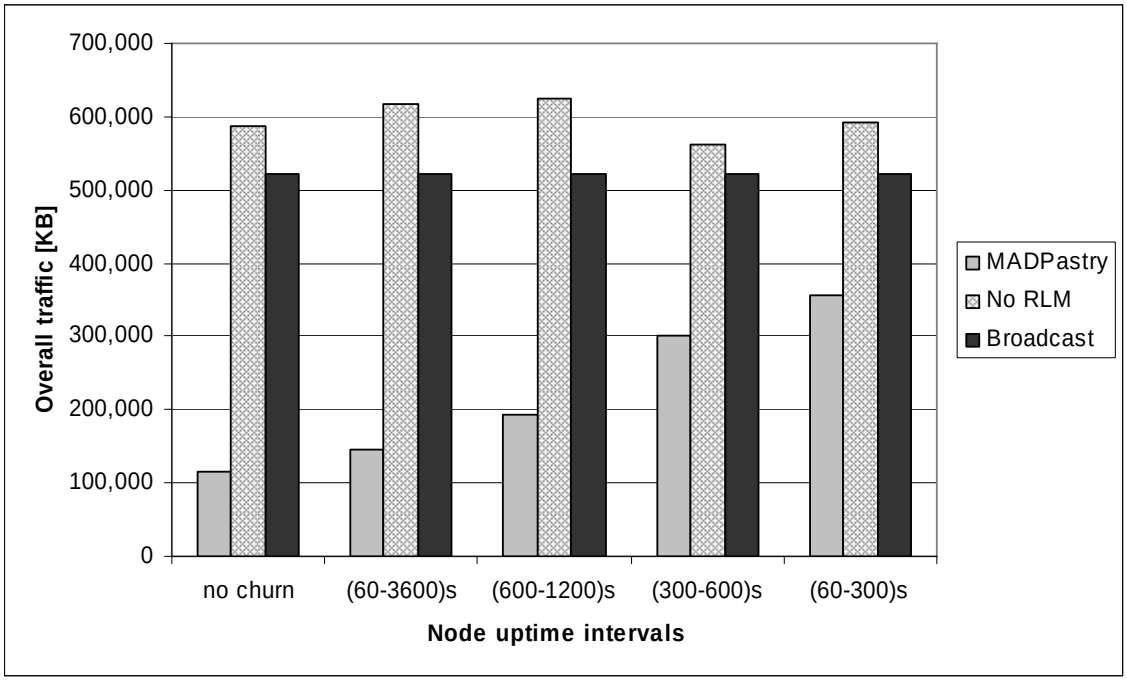


**Figure 5.13 Overall traffic vs. churn rates.**

this will increase the probability of a node to receive a packet under one of its old overlay IDs, in which case the packet would be send back to the previous overlay hop. Thus, the overlay stretch will be markedly increased, which, in turn, reduces the success rate. Secondly, the more frequently nodes fail, the more cluster reorganization traffic this will trigger, which, in turn, will also increase the number of times where MADPastry has to resort to AODV-style network-wide route discoveries. In the case of the DHT router without RLM, this increase in network traffic can also be observed up to a medium churn rate. For high churn rates, however, the routers seem to generate slightly less network traffic. The reason for this is that, with such high churn rates, the average number of participating nodes drops below 250 as failed nodes have to wait longer and longer before their rejoin requests succeed (the correct delivery of such rejoin requests can fail, for example, because nodes on the request route fail during the forwarding or because the requests might increasingly be forwarded to failed nodes, etc.). Thus, less and less nodes that would generate traffic actively participate in the network. In the case of a very high churn rate and "No RLM", the figure increases again slightly as the effect just described is outweighed by the traffic caused by the very frequent rejoin requests. In MADPastry networks, however, the ever increasing efforts to reorganize the clusters outweigh this effect.

## 5.6 Handovers

The experimental results thus far have shown that MADPastry produces drastically less overhead than the broadcast agent and MADPastry without clusters do. It is important to realize, though, that these experimental results present the *gross* overhead savings of MADPastry.

The reason for this is that, when MADPastry nodes change their cluster membership, they effectively change their overlay ID. Therefore, when a MADPastry node changes its overlay ID, it would have to pass the objects (or, more likely, references to them) that it was responsible for under its old overlay ID to its old left and right leaf before leaving the network and acquire the new objects (or, more likely, references to them) that it is now responsible for from its new left and right leaf. However, the nature of that additional handover traffic entirely depends on the actual application running on top of MADPastry, as well as the amount and distribution of the objects in the network.

To evaluate the maximum potential of MADPastry, all simulations thus far have assumed that, after a cluster change, a node is able to handover its object references to both its old left and right leaf in *one* message each, and, conversely, also acquire its new object references from its new left and right leaf in one message each. However, the number of handover messages that have to be exchanged following a cluster change will certainly have an impact on the overall performance of MADPastry. Therefore, the random lookup application was, next, slightly extended. Aside from periodically issuing random lookups, varying numbers of objects were also uniformly distributed inside the overlay ID space. Since it can be prohibitive to transfer large objects in MANETs, the DHT actually

**Table 5.6 Simulation parameters and values – handovers.**

| Simulation Parameter | Value range |
|---|---|
| Simulation duration | 3600s (simulated) |
| Network size | 250 |
| Network density | 100 nodes/km² |
| Node mobility | 1.4m/s (constant, 0s pause time) |
| Random lookup interval (per node) | 1s and 10s |
| Total number of objects | 1,000, 10,000, 100,000, and 1,000,000 |

stores references to the objects. A reference contains the object's ID (i.e. hash key) and the physical address of the node where the object resides – hence, 8 bytes per reference (32 bits for the hash key and 32 for the network address of the provider). When a MADPastry node changes its overlay ID, it hands over and acquires the respective old and new references. To minimize the handover overhead, the application does not hand over each object reference individually. Instead, each handover packet contains multiple object reference. As a heuristic to balance the trade-off of having to send numerous small handover packets as opposed to a small number of large handover packets, the application tries to maintain a ratio of 1:4 between the number of handover packets that a node needs send after an overlay ID change has occurred and the number of object references that are to be handed over. Again, a 250-node network was employed. The simulations examined the effect that a total of 1,000, 10,000, 100,000, and 1,000,000 uniformly distributed distinct objects have on the overall performance. Table 5.6 provides an overview of the simulation parameters and their respective values.

In a first set of simulations, a lookup rate of one request per 10s per node was considered. Figure 5.14 shows the success rate in reference to the total number of objects in the network (please note the logarithmic scale on the x-axis). Up to a total of 100,000 distinct objects, i.e. on average 400 distinct object references per node, MADPastry can sustain success rates of above or equal to those of the router without clusters – around 94% – as the additional handover packets actually help spread node information, that can be used to update the routing tables, through the network, thereby mitigating the negative effects of an increased number of packet collisions. At the same time, MADPastry's success rates remains well above that of the broadcast router. With 1,000,000 objects in the network (on average 4,000 distinct objects per node), however, the handover packets start markedly interfering with lookup packets (e.g. through collisions), as they now dominate the overall traffic, so that MADPastry's success rate starts falling slightly below that of the router without clusters. Note that figures for the broadcast router and the router without clusters remain unaffected by the number of objects as they do not need to hand over packets since no overlay ID changes occur during the simulations.

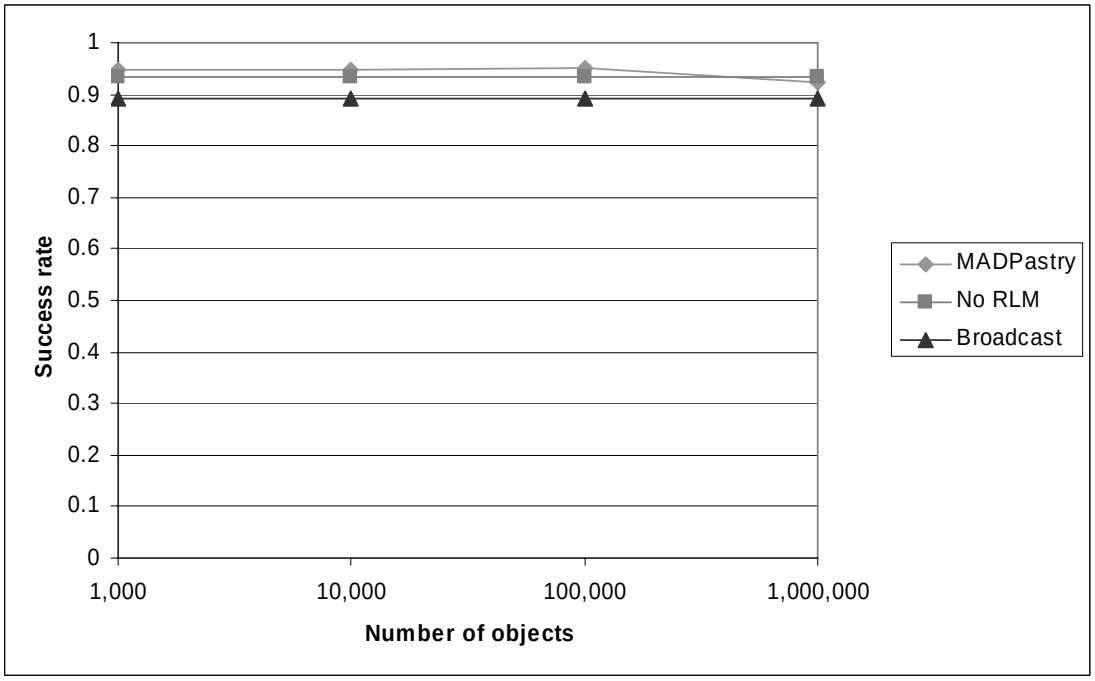


**Figure 5.14 Success rate vs. number of objects – 1 lookup per 10 seconds per node.**

Figure 5.15 shows the total number of generated packets. Note that MADPastry figures increase only slightly with higher objects totals. This is due to the heuristic that the application tries to keep a ratio of 1:4 between the number of hand over packets and the number of object references that a node needs to hand over after an overlay ID change. As can be seen, the number of packets exchanged by MADPastry remains markedly below the figures of the broadcast router and the router without cluster for all object counts. However, this figure is

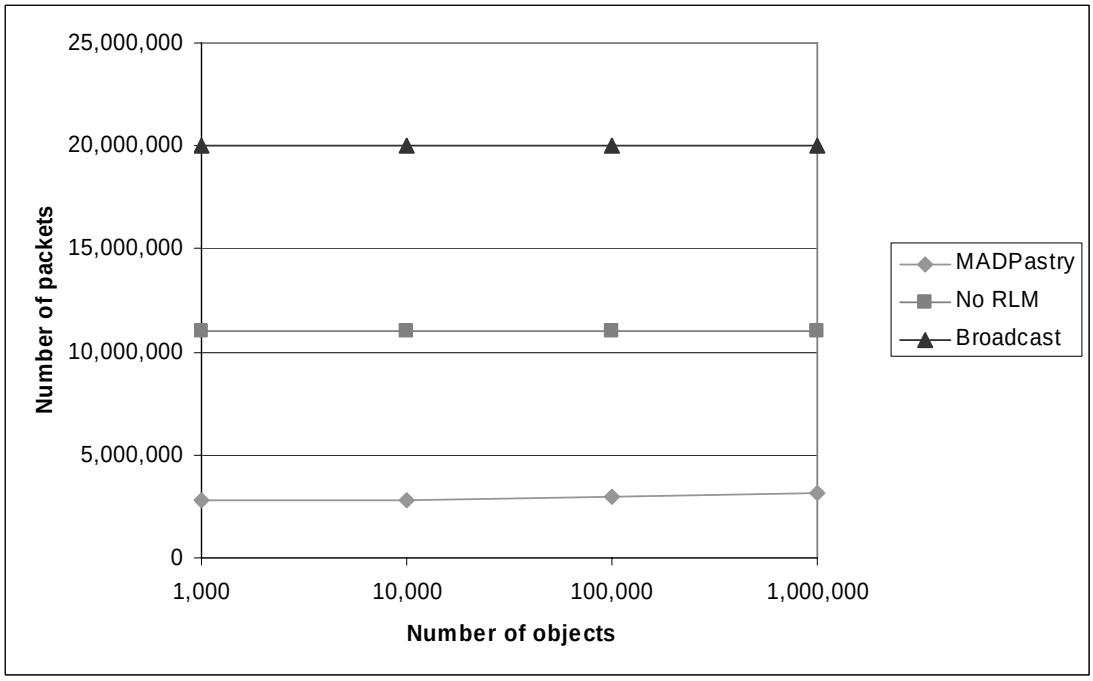


**Figure 5.15 Number of packets vs. number of objects – 1 lookup per 10 seconds per node.**

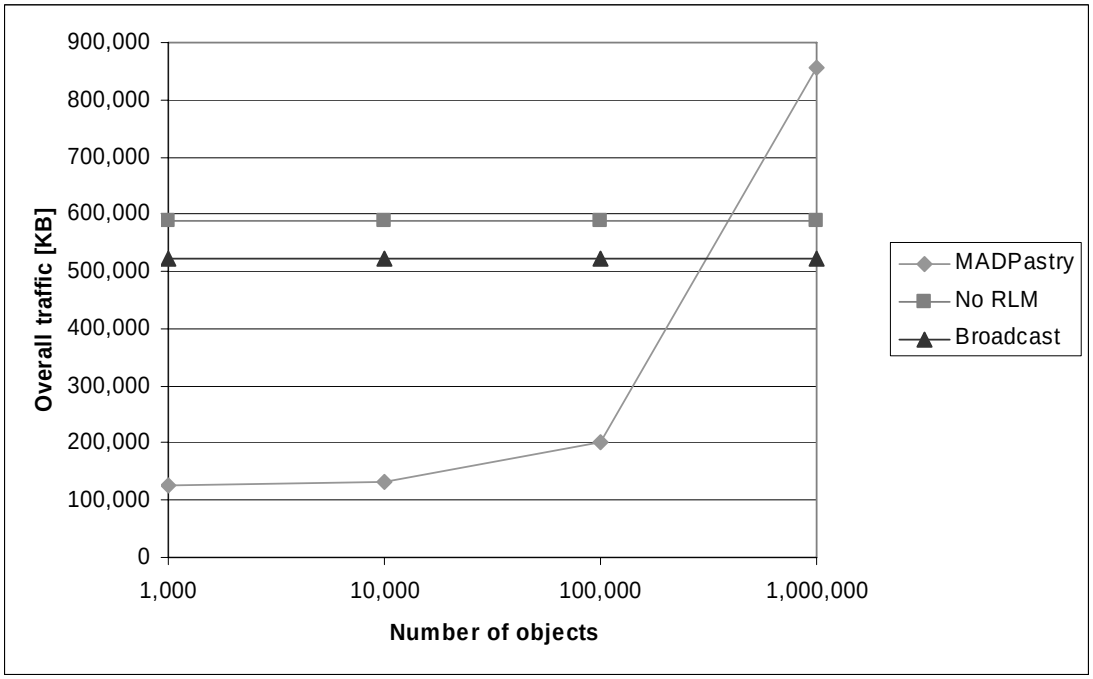


**Figure 5.16 Overall traffic vs. total number of objects – 1 lookup per 10 seconds per node.**

only of limited significance here as the length of a handover packet will increase decidedly with a growing number of objects. Figure 5.16 demonstrates this. Up to an object count of 100,000, MADPastry produces significantly less traffic than the two other routing agents do. For 1,000,000 objects, on the other hand, MADPastry's figure clearly surpasses the other routers. This explains why, for this high total number of objects, lookup packets noticeably start colliding with handover packet and the success rate of MADPastry starts declining.

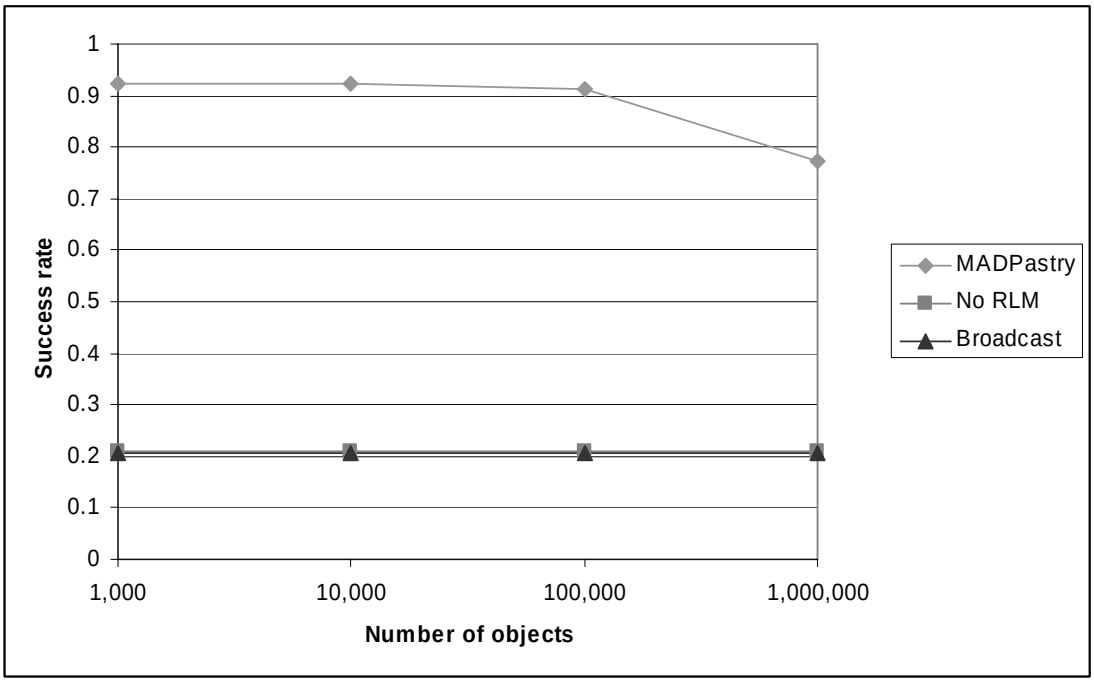


**Figure 5.17 Success rate vs. number of objects – 1 lookup per second per node.**

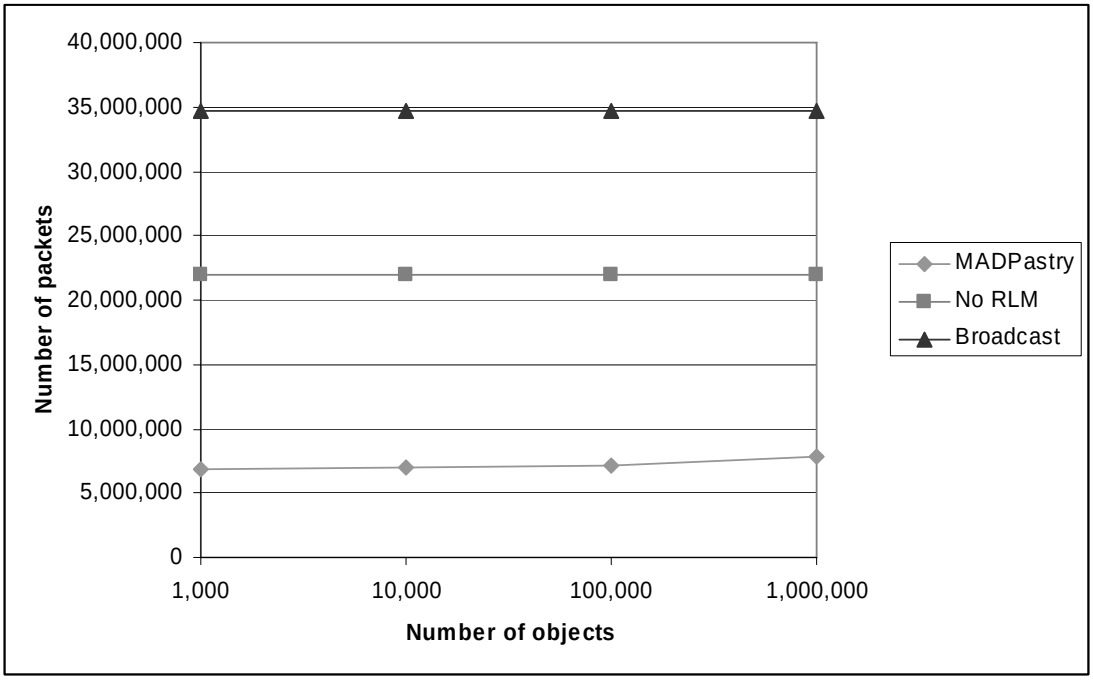


**Figure 5.18 Number of packets vs. number of objects – 1 lookup per 1 second per node.**

Next, the effect of handover packets was evaluated using a higher lookup rate of 1 lookup per second per node. Note, again, that the figures of the broadcast router and the router without clusters are not affected by handovers due to the lack of overlay ID changes during the simulations.

Figure 5.17 shows the respective success rates. Even with 1,000,000 distinct objects in the network, MADPastry can still achieve significantly higher success rates than the other two routing agents do. As with the lower lookup rate before,

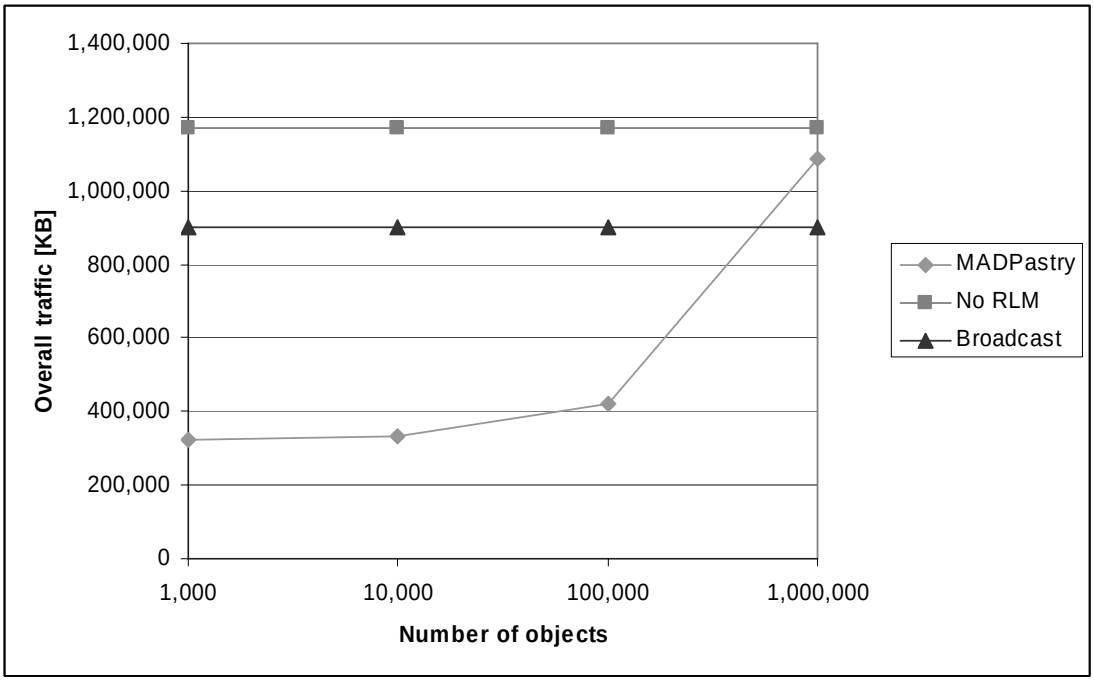


**Figure 5.19 Overall traffic vs. total number of objects – 1 lookup per second per node.**

MADPastry's success rates remains practically unaffected by the handover traffic up to an object count of 100,000. With 1,000,000 distinct objects, the (large) handover packets starts interfering with the lookups, which results in lower lookup success rate of slightly below 80%. The traffic development is depicted in Figure 5.18 (number of packets) and Figure 5.19 (generated traffic in kilo bytes). Since the broadcast router and the router without clusters are overwhelmed with the high lookup rate, the interference of MADPastry's handover packets at 1,000,000 distinct objects does not have as much overall effect as before.

## 5.7 Summary

The simulation results presented in this chapter show that MADPastry achieves comparable or better lookup success rates at *significantly* less overall traffic compared to a reference broadcast application and a reference DHT substrate without locality awareness for most scenarios considered. Notably, MADPastry's strengths become especially apparent when the network has to handle high request rates.

Two basic rules of thumb can be drawn from the experimental results. First of all, since MADPastry markedly outperformed the reference DHT substrate without Random Landmarking in the vast majority of the considered scenarios, it can be concluded that it is, indeed, essential for a DHT substrate in MANETs to explicitly consider physical locality. Secondly, a DHT substrate explicitly designed for the usage in MANETs (as demonstrated by MADPastry) can provide very efficient key-based routing for MANETs that display a certain degree of stability – e.g. node velocities in the range of fast walking speeds, mild churn rates, etc. However, in MANETs with high volatility – e.g. high node velocity and/or churn rates – it is very difficult to maintain any sort of routing structure – including a DHT. In such MANETs, one might, indeed, be well advised to resort to a broadcast-based approach.