Analysis of a Location Service for Position-Based Routing in Mobile Ad Hoc Networks

Michael Käsemann, Hannes Hartenstein

NEC Europe Ltd. Network Laboratories
Adenauerplatz 6
69115 Heidelberg, Germany
{Michael.Kaesemann|Hannes.Hartenstein}@ccrle.nec.de

Holger Füßler, Martin Mauve

Praktische Informatik IV, Universität Mannheim
L 15,16
68161 Mannheim, Germany
{fuessler|mauve}@informatik.uni-mannheim.de

Abstract: Position-based routing in a mobile ad hoc network requires geographic addresses. Thus, a node that wants to send a packet to some target node has to know the target's (approximate) current position. In order to provide each node's position to the other network nodes, a distributed location service has to be used. J. Li et al. recently put forward a promising approach called the 'Grid Location Service' (GLS). In this paper we provide some analyses and evaluations of GLS by means of simulation with *ns-2* beyond the results of the original paper. We present quantitative results with respect to location query failure rate and bandwidth consumption. We analyze in detail why queries failed and how the query failure rate can be decreased for scenarios with a low density of nodes.

1 Introduction

Routing in a mobile ad hoc network is a demanding task since the network's topology is changing frequently. At the same time, the bandwidth consumption of routing-related messages should be low since the overall available bandwidth is rather limited.

'Traditional' IETF Manet [Man] protocols have either followed distance vector or link state ideas in case of so-called 'pro-active' approaches, or have employed on-demand protocols where a route is only established when there is an actual need for this specific route. On-demand approaches are generally considered to be more scalable than pro-active ones for mobile ad hoc networks when it can be assumed that not all nodes communicate with all other nodes all the time.

While known for a long time (see, e.g. [TK84]), position-based routing has recently received significant attention [BBC+01, BMSU99, Kar00] for ad hoc networking. One

main reason for this is that today it is possible for nodes to know their geographic positions, e.g., by means of GPS. The motivation to use position-based routing is that it eliminates the need to maintain routes and therefore is very well suited for highly mobile networks. In order to forward a packet a node using position-based routing only needs information about the immediate neighborhood and the position of the destination. A survey on position-based routing for mobile ad hoc networks is given in [MWH01].

For position-based routing one assumes that nodes are addressed by a node identifier (ID) together with the node's current geographic position. A forwarding node makes use of the target's position given in the packet header as well as of some locally available information: the forwarding node's own position and knowledge of the positions of its 1-hop neighbors. With a greedy forwarding strategy, the forwarding node selects as next hop the 1-hop neighbor that lies in the direction of the target and results in the 'most forward progress'.[1]

Note the main difference between traditional topology-based routing and position-based routing: with position-based routing there is no route setup or maintenance, forwarding is done 'on-the-fly'.

However, as an essential prerequisite for position-based routing, a *location service* is needed from which a node can learn the current position of its desired communication partner. A survey on various distributed location services is also presented in [MWH01]. Clearly, in order to learn the position of another node one could flood the whole network as with a route discovery of an on-demand approach. Alternatively one could let each node flood its current position in regular intervals as a 'pro-active' approach. Of course, the optimum is likely to lie somewhere in the middle of both approaches. One proposal where each node sends its current position only to a small subset of all nodes has been put forward by J. Li et al. under the name of Grid Location Service (GLS) [LJC+00]. While in our opinion GLS appears to be a promising candidate for a scalable distributed location service, the approach shows some degree of complexity and requires a deeper analysis and evaluation than has been done before. In particular, we would like to obtain answers to the following questions:

- How accurately does GLS manage locations?
- When a location query fails, what is the reason?
- What is the exact amount of signaling load that is required to provide sufficiently up-to-date information on the position of nodes?
- What density of nodes is required for successful greedy forwarding? In other words, when would position-based routing as well as a location service benefit from a more advanced forwarding strategy?

In this paper we present results derived from simulations with *ns-2* [ns] that answer the above questions. In Section 2 we first give an introduction to GLS. In Section 3 we outline

[1] When there is no 1-hop neighbor in the direction of the target, a 'recovery strategy' has to be employed.

our simulation scenarios and present results for *i)* a detailed analysis of location query failure rate vs signaling load, and *ii)* the impact of density of nodes on the performance of GLS and greedy forwarding in general. Section 4 concludes the paper.

2 Grid Location Service

GLS structures an ad hoc network's area by using a fixed grid whose grid lines are known to all nodes in the network. It is assumed that each node in a cell of the grid knows about all other nodes in the same cell. This is achieved by maintaining 1- and 2-hop neighbor lists at each node and by choosing an appropriate cell size with respect to the transmission range of the nodes. The neighbor lists are built from periodic 'hello' broadcasts that are sent by all nodes and indicate a node's location as well as other parameters like the node's 1-hop neighbors. Thus, in order to reach a node one only has to reach the corresponding cell or, in GLS terminology, the node's 'location'. Note that the location of a node, therefore, represents 'quantized' positional information in contrast to the actual geographical position.[2]

GLS represents a fully distributed location service, i.e., each node in the network maintains a part of the overall location database. In this service a node is called a location server of another node if it stores the mapping of this node's ID to its location. The main problem that has to be solved in such a system is how to find one or more location servers for a node with a certain ID. This mapping is needed in two situations: *i)* when a node changes its position it needs to store the new position in *all* of its location servers and *ii)* when a node needs the position of a communication partner it has to find at least *one* of that node's location servers. In the following we first present the basic GLS concepts inspired by 'consistent hashing' and afterwards give some more technical details.

GLS adheres to the following two design guidelines: *i)* no flooding should be used, and *ii)* the density of a node's location server should be decreasing with an increase in distance to the node. These constraints are imposed for scalability reasons. The design guidelines are fulfilled by superimposing a quadtree on top of the grid structure (Figures 1 and 2). The original grid cells are now called 1-order squares, the cells of the next upper level are called 2-order squares, and so on. For a node A, GLS selects one location server per n-order square that is in the same $n+1$-order square as node A if this n order square does not already contain a location server for A from a previous step of the recursion (see example below).

The selection of location servers is based on node identifiers (IDs consisting of a natural number). A node A's location server in a specific square is determined by a hash function that takes as input the IDs of all nodes in the respective square as well as the ID of node A: the location server for node A in an n-order square is the one that has the 'nearest' ID to node A's ID in the respective n-order square. The nearest ID to node A's ID is

[2]Greedy forwarding can now proceed as follows: first, 'candidate locations' in the direction of a target's location are selected; then, within these locations the node with the 'best' geographic position is chosen as next hop.

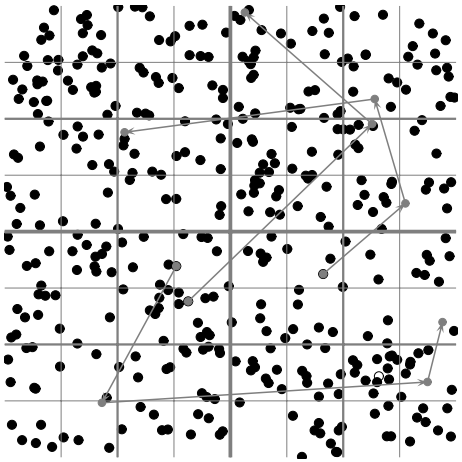

Figure 1: GLS grid structure as well as movement paths for 3 selected nodes.

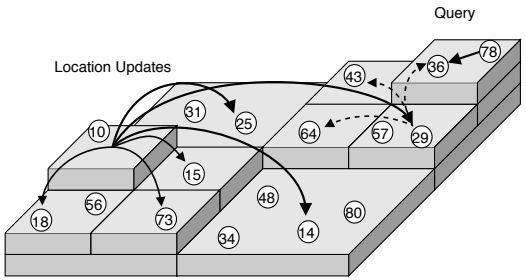

Figure 2: GLS example.

the least ID larger than node A's ID.[3] The hash function is designed in a way that *each* node can make use of the hash function for a specific square and target ID in order to find the location servers of the target ID. The location server selection process is best explained using an example depicted in Figure 2 (taken from [MWH01]). For node 10 the selected location servers in the three surrounding first order squares are nodes 15, 18, 73. In the surrounding three second-order squares, again the nodes with the nearest ID are chosen to host the node's location; in the example these are nodes 14, 25, and 29. This process repeats until the area of the ad-hoc network is covered. The 'density' of location information about a given node thus decreases logarithmically with the distance to that node.

In order to explain how a node can find a location server for the mapping of an arbitrary node ID to that node's location, we use again the example given in Figure 2. Assume

[3] ID numbers wrap around after the highest possible ID.

that node 78 wants to obtain the location of node 10. It therefore should locate a 'nearby' node that knows about the location of node 10. In the example this is node 29. While node 78 does not know that node 29 holds the required position, it is able to discover this information. To see how this works, it is useful to have a look at the location servers for node 29. Its location is stored in the three surrounding first order squares in nodes 36, 43, and 64. Note that each of these nodes and node 29 are automatically also the ones in their respective first order square with the nearest ID to 10. Thus, there exists a 'trail' with descending node IDs to the correct location server from each of the squares of *all* orders. Location queries for a node can now be directed to the node with the nearest ID the querying node knows of. In our example this would be node 36. The node with the nearest ID does not necessarily know the sought-after node, but it will know a node with a nearer node ID (node 29, which already is the sought-after location server). The process continues until a node that has the location information available is found.

When a location server is found that has the requested information, the location query is forwarded to the queried target node. The target node itself sends back a query response to the query originator indicating the current location.

After having explained the basic concepts of GLS we like to point out some technical details. It is important to note that a node does not need to know the IDs of its location servers, which makes a bootstrapping mechanism that discovers a node's location servers unnecessary. A node sends out location update messages towards the respective n-order squares for which a location server has to be selected. Within such a square, location update messages are forwarded to nodes with nearer IDs in a process closely resembling location queries that are explained above. The only difference is that information is written instead of read, ensuring that the location information reaches the correct node where it is then stored. Proof of correctness can be done in an inductive fashion (see the original paper on GLS [LJC$^+$00]). In general, forwarding decision are based either on locations or on node ID's: for example, when sending a location update, this messages is first forwarded to the respective square it is intended for by means of position-based forwarding. Within the target square the update is then forwarded with respect to node IDs.

Clearly, the 'consistency' of the location server selection might suffer from mobility. When a node moves into a new grid cell, information in the node's location servers is outdated. Furthermore, since the moving node also acts as location server for some other nodes, there might be no location server for these nodes in the square the moving node has just left. Thus, there is a clear trade-off between bandwidth consumption of GLS signaling messages (in particular, location updates) and the success rate of location queries. This trade-off is studied in detail in the following section.

A trigger for sending a location update is based on the 'update distance' parameter d that indicates the distance a node might travel until a location update has to be sent again. In particular, for $i > 1$ a node updates each ith order location server after a movement of $2^{i-2}d$ since sending the last update.

To alleviate the problem of looking for a node that has just moved to a neighboring cell, the concept of a 'forwarding pointer' has been introduced. The forwarding pointer consisting

of a node ID and its new location is broadcasted by a node shortly after the node moved into the new cell. Nodes of the previous cell that receive the 'forwarding pointer update' append it to their hello message.

To improve the performance of location queries, each node maintains location information it has obtained from forwarded packets in a 'location cache'.

For more details on GLS the reader is referred to [LJC+00].

3 Simulation Results

We ported the original *ns-2* modules [gri] to the current version *ns-2.1b8a* and extended the modules for our measurements. In particular, we added:

- Support for detailed location query failure analysis. We can check the specific reason why a query was not successfully completed.
- Support for deviation analysis. When a location query 'meets' a location table/cache with an entry for the query's target, we measure the distance between the target's current position and the target's position when the entry in the table/cache was created.
- Support for accurate bandwidth measurements. For the bandwidth measurements we set the header types and field sizes as given in Tables 1 and 2.
- Support for forwarding analysis. If a packet does not make it to its destination, we can check whether *i)* there exists a theoretical path to the destination and *ii)* whether there exists a 'greedy' path to the destination.
- Enhanced tracing facilities.

In all simulations we make use of the random waypoint mobility model [BMJ+98] as movement pattern. In this model each node randomly selects a waypoint in the area that contains the network and moves from its current location to the waypoint with a random but constant speed. Once a node has arrived at the target waypoint it immediately selects a new waypoint and speed and the process continues. IEEE 802.11 with 1Mbits/s is the simulated protocol for the wireless communication. The transmission range is set to 250m. The size of the first order squares is set to 250m $\times$ 250m. The update distance d is set to 100m. All nodes move with a maximum speed of 10 m/s (average 5 m/s). The location queries are generated by randomly selecting querying nodes and target IDs with a uniform distribution.

3.1 Location query failure rate vs costs

In this section we study how often location queries succeed or fail, the reasons for query failures, and what amount of GLS signaling is needed to achieve the observed location

Table 1: Fields and their respective sizes in GLS packet headers used in our bandwidth calculations. The Table length parameters refers to the size of the Neighbor Table and Forwarding Pointer Table in a Hello packet.

Field type	**Size in bytes**
Identifier (ID)	3
Location (LOC)	2
Speed	1
Position (X,Y)	2
Timestamp	2
Timeout value	2
Packet type and flags	1
Table length	2

Table 2: The main header types of GLS and their respective fields as used in our experiments. The forwarding pointer update is the message sent by a moving node in order to inform its previous cell about its new location.

Header type	**Fields** All packet headers include the packet type and flags as well as a timestamp field (contributing 3 bytes). Additional fields:	**Size** (Bytes)
HELLO	Source LOC, Source ID, Source (X,Y), Table length, Neighbor table, Forwarding Pointer Table (entries in a table consist of ID and LOC)	>10
LOCATION UPDATE	Source LOC, Source ID, Destination LOC, Location Timeout	12
LOCATION QUERY	Source LOC, Source ID, Destination ID, Next LOC, Next ID, Target Data Timestamp	18
QUERY RESPONSE	Source LOC, Source ID, Destination LOC, Destination ID	13
LOCATION NOTIFICATION	Source LOC, Source ID, Destination LOC, Destination ID	13
FP UPDATE	Source LOC, Source ID, Destination LOC	10

query success rate. We first study a scenario of size 2km $\times$ 2km populated with 400 mobile nodes, thus, with a density of 100 nodes per square kilometer as recommended in the original GLS paper. In the subsequent section we will vary the density of nodes and study the corresponding effects. For all results described below an average of 5 runs was taken.

From the results presented in Table 3 we see that about 1400 location queries out of a total of 6000 queries were answered by the querying node itself, i.e., via its neighbor table, location table, or location cache. Only the remaining 4598 queries were actually sent to the network. Losses of queries occur between requesting node and location server,

Table 3: Query analysis.

Event	Number of packets
Total number of queries	6000
Number of queries not resolved locally	4598
Number of queries received by appropriate location server	4311
Number of queries received by destination	4233
Number of successful queries	4170
Number of lost queries	428

between location server and target destination, and between destination and requesting node. About 7.1% of all queries, or 9.3% of all queries actually sent, were lost.

We classify the reasons for a loss of a location query into 4 classes.

- No route: GLS forwarding fails since at some forwarding node there is no option for a next-hop node.
- TTL expired: The packet either travels for too long or enters a routing loop.
- No location server found: no location server for the requested target node can be found since no node with a 'better' ID is known.
- Location error: a location for the target node is determined but the target node is no longer there and cannot be reached.

Our simulations results show that about 15% of the lost queries suffered from 'No route' and about 40% were classified as 'TTL expired'. About 40% of the lost queries could not be resolved since no appropriate location server could be found. Finally, about 5% of queries failed due to a location error.

An analysis of the successful queries as presented in Table 4 shows that most queries are resolved either using the local 2-hop neighbor table or a foreign location table or cache. The average deviation gives the deviation in meters between the current position of the target node and the target's position when the corresponding table/cache entry was created. When we assume that a node that has traveled no further than approximately 250m since its previous location update might still be reachable, we see that the location tables and caches are sufficiently 'up-to-date' on average. For a further study, we like to investigate to what degree the number of hello messages and location updates can be reduced without significantly affecting performance.

The signaling load in the current study is as follows. There were 38790 updates sent with a size of 12 bytes. Each update traveled 3.1 hops on average. Only 1.4% of the updates were lost. Thus, when we compute the average number of kilobits per second per node, we obtain an average load of about 0.1 kbps per node for the location updates.

Table 4: Shown are the sources of information for successful responses on location queries together with the amount of queries that made use of the respective source, the deviation of the target's current position compared to its position when the target's location information was inserted in the cache/table, as well as the age of the entry that was used for answering the query. Note that the deviation information is external to GLS but measured by the simulation environment.

Source of response	Proportion (%)	Deviation in m	Age in s
2-Hop Neighbor Table	20	7.58	0.44
Location cache (local)	2	122.76	15.66
Location table (local)	1	263.37	92.95
Location cache (foreign)	32	115.78	14.78
Location table (foreign)	45	237.79	83.84

The hello packets are of size 165.22 bytes on average due to the neighbor and forwarding pointer tables. A total of 60184 hello messages where transmitted, thus, the hello messages contribute 0.66 kbps per node. Thus, in total the signaling overhead is about 0.8 kbps per node.

3.2 Impact of Node Density

We now study the impact of a varying node density on GLS performance, i.e., we are interested to assess how the various performance parameters scale with the number of nodes. Furthermore, since ad hoc networks are based on the principle of mutual assistance, it is interesting to know how many nodes have to participate in GLS in order to succeed. For this reason, we varied the number of nodes between 100 and 400 for the scenario of the previous section. Figure 3 shows the amount of queries sent to the network as well as the amount of queries that failed compared to all submitted queries. The portion of queries that were not resolved locally (compared to all location queries) keeps constant, while the amount of query failures compared to all queries increases significantly with a decreasing number of nodes.

The reasons for the significant query failure rate within a low density scenario are explained in Figure 4. For the scenario with only 100 nodes the most common error (60% of all query failures) is due to 'no route'. We were now interested in the question of whether a better greedy or non-greedy forwarding strategy for queries and updates can alleviate this particular problem.

We examined whenever a 'no route' error occurred whether a greedy forwarding strategy that has full instantaneous knowledge on node positions could successfully deliver the query. In addition, we checked whether there is a path to the destination at all (using Dijkstra's algorithm). The results are presented in Figure 5. Obviously, the 'improved' greedy strategy will eliminate a significant part of the occurred errors while the introduction of a recovery strategy to the simple greedy forwarding can eliminate 70% to 90% of all 'no route' errors.

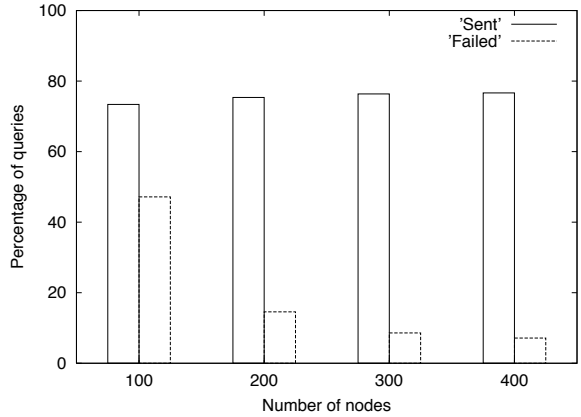

Figure 3: Shown is the amount of queries sent to the network as well as the amount of queries that failed, both in comparison to the total number of queries.

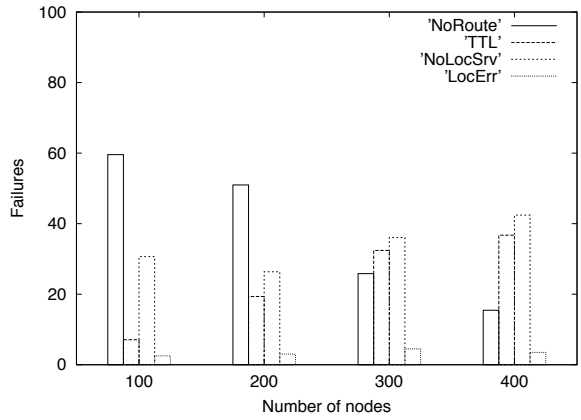

Figure 4: Shown are the distributions for the various error classes of query failures.

Another observation that can be made by viewing Figure 4 is that the proportion of 'TTL expired' errors is increasing for an increasing number of nodes. While a greedy forwarding strategy is inherently loop-free for a *static* network, node mobility can and actually does introduce some amount of 'looping' as explained in the following example.

Figure 6 (left) shows an example of a path of a location update that ended in a forwarding loop. The sending node 4 sends an update for the second order square located top left (a location update is not directly send to a specific destination since the destination, i.e., the corresponding location server, has to be determined 'on-the-fly'). The packet travels along the path via nodes 91, 25, 48, 65, to node 21. When it arrives there, node 21 has the 'misconception' that the update should be sent to node 8. However, node 21 is not aware

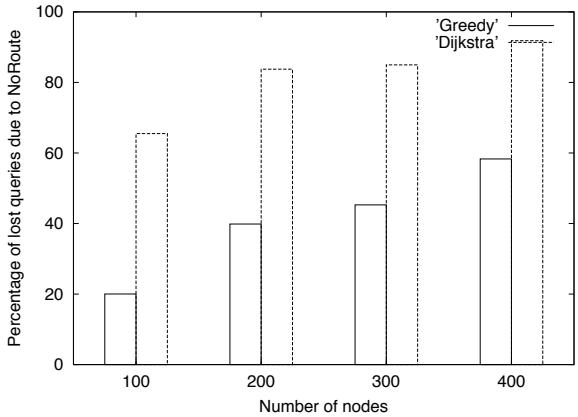

Figure 5: Compared to the number of queries that failed due to 'no route', this graph presents the proportion of queries that would have been successful with an 'ideal' greedy method as well as with using Dijkstra's shortest path algorithm.

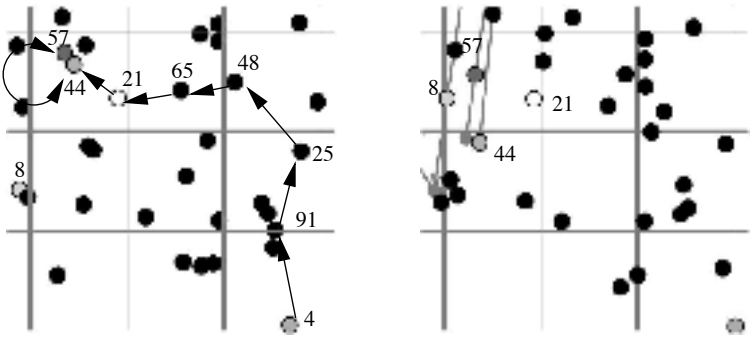

Figure 6: Two snapshots that illustrate why forwarding loops can occur. The figure on the right shows a snapshot with movement paths a few seconds before the snapshot given on the left side was taken.

of the fact that node 8 is not in the target second order square anymore. As we see from Figure 6 (right) that shows a snapshot taken a few seconds before the one of 6 (left), node 8 traveled out of the target second order square. Now, when node 21 forwards the packet to node 44, this decision is also done based on somewhat 'outdated' position information. Finally, nodes 44 and 57 have an inconsistent view of their respective positions. Again, this can be easily explained from the 'travel history' as depicted in Figure 6 (right). Basically, node 44 thinks that node 57 might still reach node 8, while node 57 thinks that node 44 might actually still reach node 8. These kinds of inconsistencies can occur simply due to motion and 'under-sampling' of positional changes. However, since hello messages are unacknowledged broadcasts, it can also happen that a node misses the announcement of a

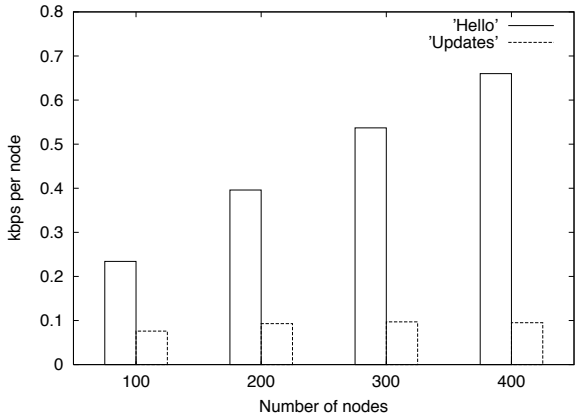

Figure 7: The per-node bandwidth consumption for hello messages and location updates.

positional change of a neighbor. Constellations of nodes with an inconsistent view of the positions might, therefore, act as 'black holes' for packets until the positional information is correctly updated again.

The analysis of the signaling load depending on node density as given in Figure 7 shows that the amount of updates per node is independent of the number of nodes. However, the amount of bytes for hello messages significantly increases with an increase in node density since the neighbor tables and forwarding pointer tables are increasing in size.

4 Conclusions and Future Work

We presented a detailed simulative study on the Grid Location Service proposal. In our 2km×2km scenario with 400 nodes, GLS consumes about 0.8kbps per node for signaling purposes and results in a location query failure rate of about 7%. The failure analysis and, in particular, the study on improved greedy or non-greedy forwarding indicate that improving the forwarding strategy by a reasonable recovery strategy will lead to a significant improvement of the location query success rate. This will also make lower density scenarios (with less than 100 nodes per square kilometer) more robust. We also showed that by increasing the node density, the bandwidth needed for hello messages dominates the costs for signaling. We are currently investigating whether reducing the frequency for sending hello messages as well as increasing the update distance parameter can save bandwidth without affecting the query success rate. In addition, we are looking into the impact of node mobility on GLS performance and check results for maximum speeds of the nodes of 30 m/s and 50 m/s. Furthermore, a study combining GLS and Greedy Perimeter Stateless Routing [Kar00] is underway.

Acknowledgments– Parts of this work were done within the framework of the 'Fleetnet – Internet on the Road' project which is partly funded by the German Ministry of Education and Research

(BMB+F) under grant 01AK025D. H. H. would like to thank his colleague Xavier Pérez Costa for valuable comments on drafts of this paper.

References

[BBC+01] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. Hubaux, and J. Le Boudec. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 2001.

[BMJ+98] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. MOBICOM'98*, pages 85–97, Dallas, TX, USA, 1998.

[BMSU99] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of 3rd ACM DIAL M99*, pages 48–55, 1999.

[gri] Grid modules for ns-2. http://www.pdos.lcs.mit.edu/grid/sim/index.html.

[Kar00] B. N. Karp. *Geographic Routing for Wireless Networks*. PhD thesis, Harvard University, 2000.

[LJC+00] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proc. MOBICOM'2000*, pages 120–130, Boston, MA, USA, 2000.

[Man] The IETF Working Group on Mobile Ad Hoc Networks (MANET). http://www.ietf.org/html.charters/manet-charter.html.

[MWH01] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.

[ns] Network Simulator (ns), version 2. http://www.isi.edu/nsnam/ns.

[TK84] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. on Comm.*, 32(3):246–257, March 1984.