Hierarchical Location Service for Mobile Ad-Hoc Networks

Wolfgang Kieß[a] *kiess@cs.uni-duesseldorf.de*
Holger Füßler[b] *fuessler@informatik.uni-mannheim.de*
Jörg Widmer[c] *joerg.widmer@epfl.ch*
Martin Mauve[a] *mauve@cs.uni-duesseldorf.de*

[a]Lehrstuhl für Rechnernetze, University of Düsseldorf, Germany
[b]Lehrstuhl für Praktische Informatik IV, University of Mannheim, Germany
[c]Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Position-based routing has proven to be a scalable and efficient way for packet routing in mobile ad-hoc networks. To enable position-based routing, a node must be able to discover the location of the node it wants to communicate with. This task is typically accomplished by a location service. In this paper, we propose a novel location service called HLS (Hierarchical Location Service). HLS divides the area covered by the network into a hierarchy of regions. The top level region covers the complete network. A region is subdivided into several regions of the next lower level until the lowest level is reached. We call a lowest level region a cell. For a given node A, one specific cell is selected on each level of the hierarchy by means of a hash function. As A changes its position it transmits position updates to these responsible *cells. If another node wants to determine the position of A it uses the same hash function to determine the cells that may hold information about the position of A. It then proceeds to query the nodes in these cells in the order of the hierarchy until it receives a reply containing the current position of A. Because of its hierarchical approach HLS is highly scalable and particularly well suited for networks where communication partners tend to be close to each other. Due to the inherent scaling limitations of ad-hoc networks it is very likely that most ad hoc networks will display this property. Furthermore HLS is very robust to node mobility and node failures since it uses regions to select location servers and not a chain of mobile nodes as it is the case, e.g., for the well known Grid Location Service (GLS). We demonstrate these traits by providing extensive simulation data on the behaviour of HLS in a wide range of scenarios and by using GLS as a benchmark.*

I. Introduction

Mobile ad-hoc networks (MANETs) enable wireless communication between mobile devices without relying on a fixed infrastructure. In these networks the mobile devices themselves forward data from a sender to a receiver, acting both as router and end-system at the same time. MANETs have a wide range of applications, e.g., range extension of WLAN access points, data transmission in disaster areas and inter-vehicular communication. Studies [5] have shown that *position-based* routing is a well-suited solution for the challenging task of routing in highly dynamic MANETs. As a prerequisite of position-based routing algorithms, each node in the network must be able to determine the position of the target node it wants to communicate with.

We assume that the nodes are able to determine their own position, e.g. by means of a positioning service such as GPS. The task of locating the destination is then accomplished by a location service, a distributed service maintained directly by the participating nodes. In this paper, we present the *Hierarchical Location Service* (HLS).

The basic operation of HLS is as follows: the area occupied by the network is divided into a hierarchy of regions. The lowest level regions are called cells. Regions of one level are aggregated to form a region on the next higher level of the hierarchy. Regions on the same level of the hierarchy do not overlap. For any given node A one cell in each level of the hierarchy is selected by means of a hash function. These cells are called the responsible cells for node A. The hash function takes a node ID (e.g. an IP Address), the level of the hierarchy and the node's position as an input and determines the responsible cell for a node

with this ID on this level of the hierarchy as an output. As a node moves through the area covered by the network, it updates its responsible cells with information about its current position. When another node B needs to determine the position of node A it uses the hash function to determine those cells that may potentially be responsible for A. It then queries those cells in the order of the hierarchy, starting with the lowest level region. On the first level that contains both nodes A and B the query will arrive at a responsible cell of A where it can be answered.

Due to its hierarchical approach and because of using the concept of responsible cells HLS has some very desirable properties:

- it does not use flooding
- the number of location servers scales logarithmically with the size of the network
- it is well-suited for high node mobility
- it specifically supports non-uniform communication patterns
- it is robust to node failures

The remainder of this paper is structured as follows: In Section II we give an overview of related work. The HLS algorithm is described in detail in Section III. Section IV contains the results of the simulation of HLS with ns-2. The paper is concluded with a summary and an outlook to future work in Section V.

II. Related Work

A location service consists of two algorithmic components, the *location update* and the *location request* component. The location update is responsible for distributing information about the current location of a target node T to a set of nodes called the *location servers* of T. If a source S wants to discover the location of T, it launches a location request which is routed through the network to one of the location servers of T. The location server can either answer the request itself or forward it to T for answering.

The range of possible designs of the update and request component are limited by two extremes: flood position updates or do not send any updates at all. If updates are flooded, each node becomes location server for each other node and no requests are necessary. If no updates are sent, each node is its own and only location server, therefore requests need to be flooded in the network.

A location service which uses flooding to spread position information is DREAM [1], the *Distance Routing Effect Algorithm for Mobility*. With DREAM, each node floods its position information in the network with varying flooding range and frequency. The frequency of the flooding is decreased with increasing range. Thus, each node knows the location of each other node whereas the accuracy of this information depends on the distance to the node. The *Reactive Location Service* (RLS) [7] marks the other extreme of the design space: it uses flooding in its request component. RLS is only active if a node S needs to discover the location of another node T. The request is flooded to all nodes in the network. Upon receiving the request, T generates an answer and sends it back to S. Both of these extremes involve flooding the network. In contrast, an efficient location service should balance the cost of updates and requests to avoid this.

A location service that does not require flooding is *Homezone* [3]. In the Homezone location service, each node is assigned an area (the so called *Homezone*) in the ad-hoc network via a hash function. Location updates are sent to all the nodes in the homezone. Location requests are answered by one of the nodes in this area. A major disadvantage of this design is the single fixed homezone. Nodes are not limited in their movements. As a result, nodes may be far away from their homezone and their updates may have to travel long distances. Furthermore, even requests from nodes close to the target node must be forwarded all the way to the homezone. This can lead to high network load and latency. In contrast to Homezone our approach does not rely on a single homezone. Instead a hierarchy of regions is responsible for maintaining the location of a node. As a consequence both updates and requests from nodes that are close to the target remain mostly local while additionally the robustness of the location service is increased.

The *Grid Location Service* (GLS) [10] divides the area containing the ad-hoc network into a hierarchy of squares forming a quad-tree. Each node selects one node in each element of the quad-tree as a location server. Therefore the density of location servers for a node is high in areas close to the node and becomes exponentially less dense as the distance to the node increases. The update and request mechanisms of GLS require that a chain of nodes based on node IDs is found and traversed to reach an actual location server for a given node. The chain leads from the updating or requesting node via some arbitrary and some dedicated nodes to a location server. As investigated in [8], traversing the chain of mobile nodes may lead

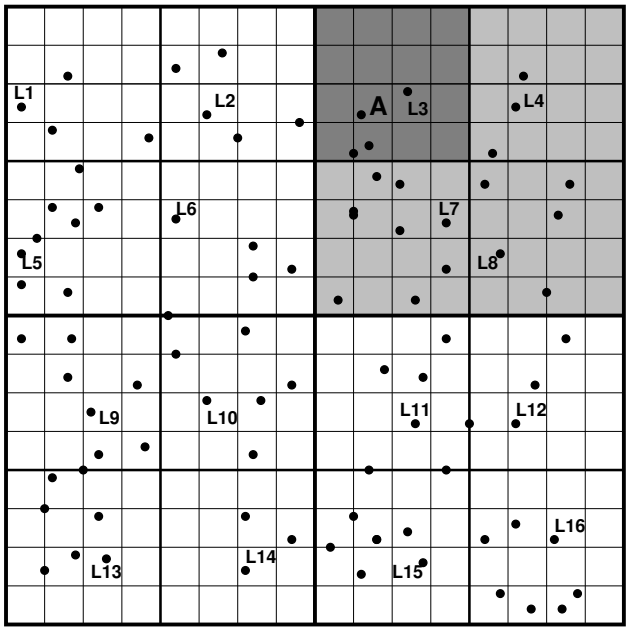


Figure 1: DLM Location Servers (L1—L16) for a node A, inspired by [15]

to significant update and lookup failures if node mobility is high: as soon as one of the dedicated nodes in the chain cannot be reached the update or request message is lost. In order to avoid this problem, HLS uses the concept of responsible cells and does not rely on a chain of mobile nodes for update and lookup of position information.

DLM [15], the *Distributed Location Management* scheme, uses a grid to partition the area of the network as shown in Figure 1. In DLM the smallest squares are assigned the highest level. Squares of the highest level are aggregated to form an area of the next lower level. This is repeated until the whole area of the network is covered by a single area which is assigned the level 0. In the example the smallest squares are assigned the level 3, sixteen of them are aggregated to form a region of level 2. Four level 2 regions form a level 1 region, four level 1 regions form the whole network as a single level 0 region. DLM then selects one level as the granularity for the distribution of location servers. In the example this is level two. For each node DLM selects one highest level (smallest) square in each of these regions. In each of the selected highest level squares, one node is chosen as location server. In the example the location servers for node A are designated as $L1$ to $L16$. In order to reduce update overhead, the information about the location of a node becomes more detailed the closer a location server is to a node. DLM does not rely on a chain of mobile nodes and is therefore more robust to mobility than GLS. On the other hand the even distribution of location servers reduces its scalability: whenever a node crosses the borders of a level-1 region or when periodic updates for maintaining a soft state are transmitted, all location servers have to be updated. Since the location servers are evenly distributed in the network the resulting message overhead is similar to that of network-wide flooding. In contrast, HLS uses a hierarchy of location servers and thereby completely avoids communication patterns that are similar to flooding the network. In addition the hierarchical approach of HLS specifically supports communication patterns where communicating nodes tend to be close to each other. Given the well known scaling behaviour of ad-hoc networks [4] it is very likely that most ad-hoc networks will display this property.

III. Hierarchical Location Service

To show how HLS works in detail, we first describe the structure of the hierarchy of regions. We then explain how cells responsible for the tracking of a node are selected and how location updates and request are sent. Finally we show how the algorithm deals with cells that do not contain any nodes.

III.A. Area partitioning

HLS partitions the area containing the ad-hoc network in *cells*. This partitioning must be known to all participating nodes. The shape and size of the cells can be chosen arbitrarily according to the properties of the network. The only prerequisite is that a node in a given lowest-level cell must be able to send packets to all other nodes in the same cell. This can either be achieved by choosing an appropriate cell size, i.e., the distance between any two points in the cell must be smaller than the radio range, or by implementing a cell-wide broadcasting mechanism. [1] It is not required that the area in which the MANET is deployed is fully covered by cells. Thus, HLS is applicable to areas containing obstacles such as buildings.

The cells are grouped hierarchically into *regions* of different levels. A number of cells forms a region of level one, a number of level-one regions forms a level-two region and so on. Regions of the same level must not intersect, i.e., each region of level n is member of exactly one region of level $n + 1$. An example for the area partitioning is shown in Figure 2.

[1] Note that this does not amount to flooding since such broadcasts are limited to the size of a lowest-level cell.

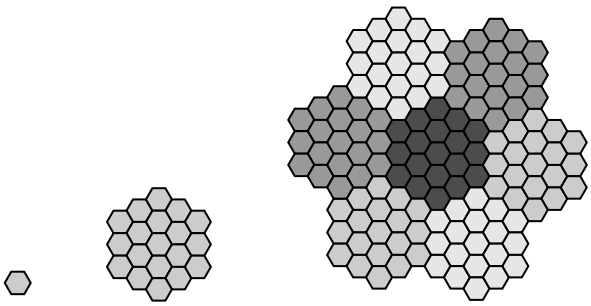


Figure 2: Grouping cells to form regions: cell, region on level one and region on level two

III.B. Responsible cells

HLS places location information for a node T in a set of cells. We call these cells the *responsible cells* (RC) of T. In the following, we say "T updates a responsible cell R" when T sends an update packet to an arbitrary node within or close to R. This node becomes *location server* for T. It is possible that subsequent updates arrive at different nodes within that cell, for example because nodes have moved. A cell may therefore contain multiple location servers for a node. Moreover, we assume that all necessary routing for HLS is done with a position-based routing protocol like GFG [2] or GPSR [5].

A node T selects one responsible cell for each level in the hierarchy. For a given level n, the RC is selected according to the following algorithm:

1. Compute the set $S(T, n)$ of cells belonging to the region of level n which contains T.

2. Select one of these cells with a hash function based on characteristics of S (like the number of cells in S) and the node ID of T.

A possible hash function is the simple, modulo-based function: $H(T, n) = \text{ID}(T) \mod \|S(T, n)\|$. With the number calculated with this hash function, one of the cells in $S(T, n)$ is chosen. As a result of the above selection, T has exactly one responsible cell on each level and it is guaranteed that the RC for T of level n and node T share the same level-n region. An example for the selection of RCs is shown in Figure 3 for a three-level hierarchy. The large circles mark the regions and the cells with the numbers are the responsible cells. The node and its RC on level one share the same level-one region, the RC of level two lies within the same level-two region as the node and so on.

An interesting observation is the following: if all cells are computed and marked which may become responsible cells as a node moves through the network,

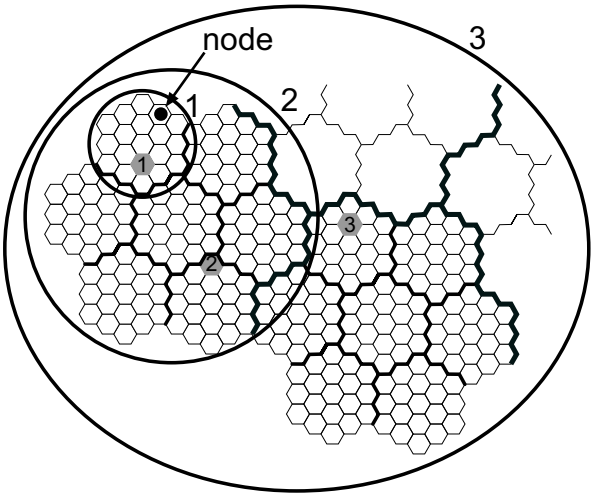


Figure 3: Example for responsible cells of a node

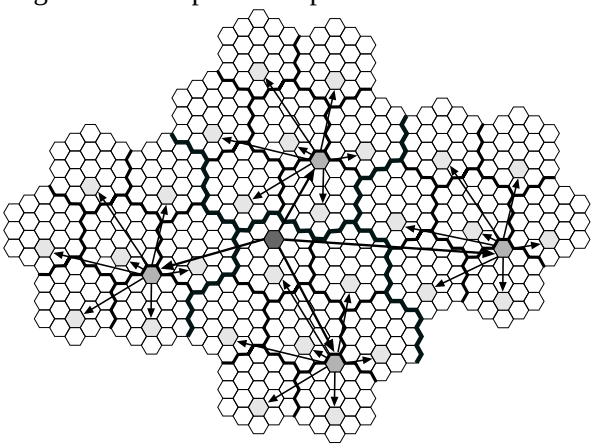


Figure 4: Candidate tree in a three level hierarchy

we get a structure similar to the one shown in Figure 4. All cells marked here are candidates for responsible cells. These *candidate cells* are connected with arrows to visualize their hierarchical, tree-like structure. We call this structure *candidate tree*. The root of the tree is the single RC candidate on the highest level. The leaves are the candidates for responsible cells on the lowest level. As a node T moves through the network, a different subset of candidate cells will become responsible cells.

The candidate tree is different for each node and can be computed from the hash function and the node ID. Selecting the responsible cells for a node T can be seen as selecting the branch in the tree which ends in the current level-one region containing T.

III.C. Location update

There are two different methods for HLS to update location servers, the *direct location scheme* and the *indirect location scheme*.

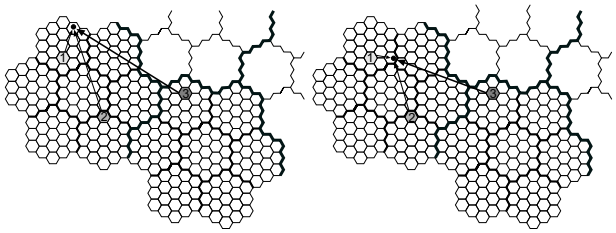
To update its location servers according to the di-

rect location scheme, a node computes its responsible cells as explained in Section III.B. Position updates are then sent to all RCs at the same rate. This update scheme is called "direct" because a location server directly knows the position of the node. In Figures 5(a) and 5(b), the location information in the RCs is represented as a pointer to the position of the node. In this scheme all responsible cells must be updated whenever the node has moved a distance large enough to render the position information in the location servers useless (Figure 5(b)). While this can be done without much overhead for close location servers, updating the RCs on a higher level which tend to be farther away may cause a significant amount of network load.

The number of necessary single hop transmissions to update a location server on level l depends on the size of a region on that level. In the following we assume that the regions have a quadtree topology with n being the number of nodes in the network and d being the diameter of the level-1 region. In the worst case, the update packet for a given level must traverse the whole region of that level. Therefore, if a node needs to update all its location servers, the cost is $O(d+2d+4d+8d+\cdots+2^{l-1}d) = O((2^l-1)d)$. With the diameter D of the whole area being $D = 2^{l-1}d$, this can be rewritten as $O((2^l - 1)d) = O((2^l - 1)\frac{D}{2^{l-1}}) = O(D)$. As the size of an area covered by n nodes grows with $O(\sqrt{n})$ for a fixed node density, the worst case update costs for an update to all levels is therefore $O(D) = O(\sqrt{n})$.

The network load can be reduced with the indirect location scheme where the location servers on higher hierarchy levels only know the region of the next lower level a node is located in. More precise location information is not necessary on higher levels. As shown in Figure 6(a), the pointers which represent the location information do no longer point to the last known position. They point to the responsible cell on the next lower level. Thus, this update scheme creates indirect location knowledge in the location servers. In contrast to GLS, the chains established here do not consist of moving nodes but of cells with fixed positions making the scheme robust to high node mobility.

Given an ideal environment with no packet loss, a location server on level n needs to be updated only when the node moves to another level-$(n-1)$ region. Thus, the responsible cell on level one will be the only cell which is updated if the node moves within the boundaries of the level-1 region (Figure 6(b)). Cells on higher levels need to be updated only if the RC on the next lower level changes (Figures 6(c), 6(d)). Hence, update traffic generated by a node is mostly lo-



(a) all RCs have exact knowledge of the nodes position

(b) the node moves: all RCs must be updated

Figure 5: Direct location scheme

cal. The majority of the update packets have to travel only a few hops whereas long-distance updates are rarely sent. Depending on the movement pattern of the nodes, the indirect location scheme may therefore reduce the costs for updates significantly. However, the worst case where all location servers have to be updated is still possible (e.g., when a node crosses the boundaries of the highest level region), thus the worst case effort remains at $O(\sqrt{n})$.

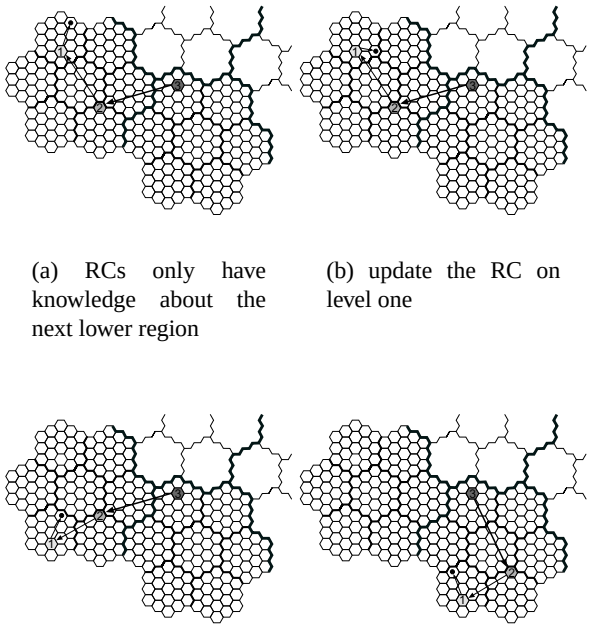
To overcome location server failures which can occur for both update schemes, we have chosen a soft state approach. As this leads to possibly unnecessary updates-especially for the indirect update scheme-other mechanisms to overcome these losses such as the use of backup location servers could be investigated in the future to further reduce network traffic.

III.D. Handovers

Since the identification of a location server depends only on its position, a node leaving a responsible cell can no longer be location server for information belonging to this responsible cell. In this case, the information belonging to the cell just left is handed over to this cell and treated like an update: the handover packet is forwarded to a node in or close to the cell which becomes the new location server.

III.E. Position Requests

To successfully query the current location of a target node T, the request of a source node S needs to be routed to a location server of T. When querying the position of T, S knows the ID of T and therefore the structure of the candidate tree defined via the hash function and T's ID. It furthermore knows that T has selected a RC for each region it resides in. Thus, the request only needs to visit each candidate cell of the regions containing S. The candidate cell of the region with the lowest level containing both S and T is per



(a) RCs only have knowledge about the next lower region

(b) update the RC on level one

(c) update RC on level two, change RC on level one

(d) update RC on level three, change RC on level one and two

Figure 6: Indirect location scheme

definition a responsible cell, and so are the candidate cells of all higher levels.

S computes the cell which T would choose as responsible cell if it were in the same level-one region and sends its request to this cell. When the request packet arrives at the first node A within the boundaries of the candidate cell, it is processed as follows:

1. Node A broadcasts the request to all nodes within the candidate cell. This is called *cellcast request*.
2. Any node which receives this cellcast request and has location information in its location database sends an answer to A.
3. If A receives an answer for its cellcast request, the request is forwarded to the target node T.
4. Otherwise A forwards the request to the corresponding candidate cell on the next level.

With this mechanism, the request is forwarded from candidate cell to candidate cell until a location server for T is found or the highest level candidate cell has been reached.

The algorithm ensures that in the worst case the request is forwarded to the top-level RC, the candidate cell which is guaranteed to be also a responsible cell. In this worst case, the request has a complexity similar to the one given above for the updates, $O(\sqrt{n})$. Usually, the request reaches a responsible cell on a lower level, in particular if communication partners tend to be close to each other. The level of the first candidate cell which is also a responsible cell for T depends on the distance between S and T. The closer the two nodes are, the earlier the branch of the candidate tree selected by T for its updates and the one calculated by S for its request will match. The branches "meet" on level l, if a level-l region is the lowest-level region which contains S and T. An upper bound for the average distance the request packet has to travel to find a location server is $O(2 * d) = O(d)$ with d being the diameter of a level-l region. Obviously, all candidate cells computed for this request with a level higher than l are responsible cells as well.

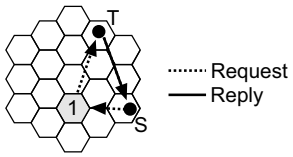
Examples for requests are given in Figure 7. Here, location servers are updated according to the indirect location scheme. If the two nodes are located in the same level-1 region as shown in Figure 7(a), the candidate cell on level one also is a responsible cell and should contain a location server. The request can be delivered and answered directly. In Figure 7(b), S is located in the same level-2 region as T. The request is forwarded via the candidate cell on level one to the responsible cell on level two which should contain a location server. In the third example presented in Figure 7(c), S and T are located in different level-2 regions. The request is forwarded to the candidate cells on level one and two, then it reaches the RC on level three and eventually finds a location server for T.

As shown in the examples, a request packet is forwarded only within the boundaries of the lowest level region where both nodes reside in. The communication complexity of a request depends on the distance between sender and target of the request. If communication is local, so is the request behaviour of HLS.

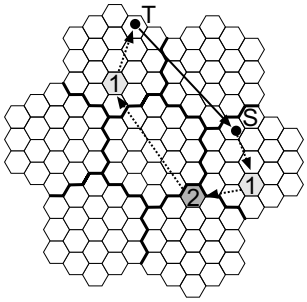
III.F. Empty cells

A problem which has not been addressed so far are empty or unreachable cells, i.e. a location update or request packet is sent to a cell which does not contain a node or which is unreachable due to a partitioned network. In HLS this problem is solved as follows: if an update can not be forwarded to the target cell, the node detecting this becomes a temporary location server. Thereafter, the information is treated by the handover mechanism explained above like any other location information outside its target RC: the temporary location server regularly tries to hand the information over to the target cell.

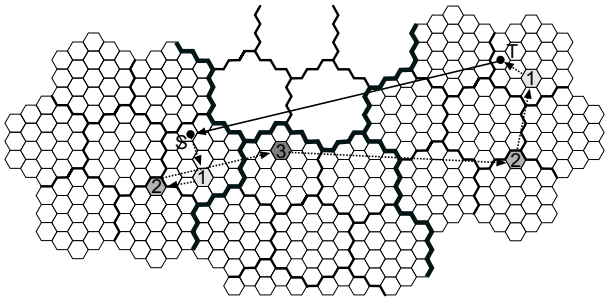
If a request looks for a location server in an empty cell, it cannot be determined if the cell is a responsible cell or a candidate cell. There are two ways to



(a) A request from a node in the same level-1 region

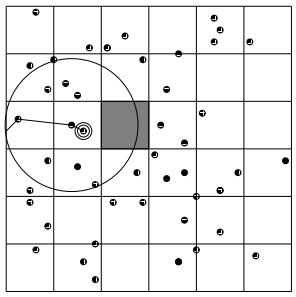


(b) A request from a node in the same level-2 region

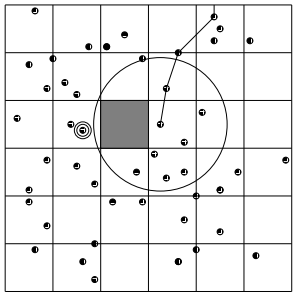


(c) A request from a node in the same level-3 region

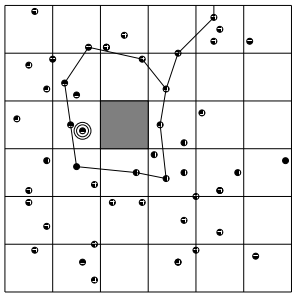
Figure 7: Example requests



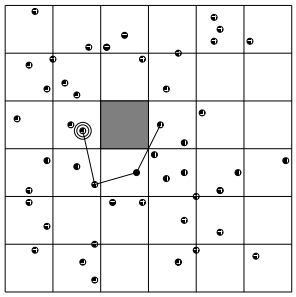
(a) empty RC, the update is stored outside of RC



(b) the request does not find the location server



(c) the request is sent to the neighbors of the RC



(d) the request is answered by the temporary location server

Figure 8: The area-extension mechanism

proceed if a request cannot reach the cell it is sent to: either search the neighbourhood for a temporary location server or forward the request to the next higher level. For HLS we have chosen a combination: for all but the highest level requests are forwarded to the next level while on the highest level the neighbourhood of the cell is searched for a temporary location server. This does minimize the likeliness that a search has to be performed while still providing a high probability of finding a location server.

An example of the search strategy is shown in Figure 8. To simplify the figure, we use quadratic cells in this example[2]. Whenever an update packet cannot reach any node within the target cell (marked dark gray) it is stored by a temporary location server (Figure 8(a), the node marked with two circles). If a top-level request is sent to the respective cell, the location server is not in the correct cell and thus may be unreachable (Figure 8(b)). In this case, the request is successively routed to the neighbors of the RC (Figure 8(c)). On overhearing the request the temporary location server sends an answer packet (Figure 8(d)) which allows the request to find the target node.

[2]Remember that shape and size of cells can be chosen arbitrarily

IV. Simulation

For the simulations, the discrete event simulator ns-2 [12] version 2.1b8a was used with the IEEE 802.11 2 MBit/s MAC layer of version 2.1b9 with additional bug fixes. Due to limitations of computing capacity, we utilized a simplified MAC layer called NULL MAC [6] in some runs to be able to simulate scenarios with up to 1875 nodes. The nodes moved according to the Modified Random Direction Mobility Model [14]. In the Modified Random Direction Mobility Model, a node arbitrarily chooses speed[3], time and direction and moves into that direction until the chosen time period has expired. Then, other values for these parameters are selected. If the node hits the border of the simulation area, it bounces back according to the physical law of reflection.

As routing protocol, GPSR [2, 5] with activated perimeter mode was used. We also did simulations with pure greedy forwarding which produced low success rates especially in scenarios with a low node density. The results of these runs can be found in [9].

We selected the Grid Location Service (GLS) [10] as a benchmark because it is scalable (it does not use

[3]Limited by a maximum speed

flooding), well understood and offers localized information [11]. The GLS implementation utilized for our simulations is the same which has been used in [8]. It has been optimized by its authors and makes extensive use of caching which influences the simulation results. In order to maintain comparability with previous studies we did not modify the caching strategy of GLS even though the caching of HLS is much less aggressive. Furthermore rectangular shaped cells were used for HLS to improve comparability with GLS.

In the HLS implementation the use of caching is minimized in order to avoid obscuring the performance of HLS with the effects of caching. The cache only contains location information about direct neighbors acquired through beacons, the information obtained through requests and information for which the node is location server. Furthermore, all location information older than 20 seconds is removed from the local location database. Location updates are sent according to the direct location scheme in regular, timer-based intervals to overcome losses (soft-state) and the automatic dropping of location information described above. Updates and Handovers are routed only in greedy mode. Early simulations indicated that routing them in perimeter mode can lead to unnecessary detours. All other packets use standard GPSR with activated perimeter mode. The cells are quadratic and have a size of 176×176 meters. Further information on the HLS implementation can be found in [9].

In a first set of simulation runs, we simulated scenarios of size 2×2 kilometers. In [5], a node density of 111 nodes per square kilometer is suggested. Since high node densities improve position based routing, we chose a conservative value of 75 nodes per square kilometer for most simulation runs. In order to understand the impact of node density on HLS we also performed simulations with 25 and 100 nodes per square kilometer. The nodes selected their movement speed out of the intervals $[0, 10]$, $[0, 30]$ and $[0, 50]$ m/s and did not pause between changing directions. This results in an average node speed of approximately 5, 15 and 25 m/s. Table 1 contains the parameters used for these runs.

All of our simulations were conducted without any data traffic on the network. While this is somewhat artificial, it removes hard-to-detect influences like congestion induced losses and allows us to better judge the performance of the algorithm itself. The performance evaluation is based on queries. A *query* is a location lookup for a node T of the routing agent at a node S. It can be either answered locally at node S by looking up T's position in the cache or by sending a request. The request must reach T which then produces a reply packet.

number of nodes	300
node density per square kilometer	75
area size (kilometer×kilometer)	2×2
max speed (m/s)	10, 30, 50
simulation time (seconds)	300
requests per node	4
requests per simulation run	1200
number of runs	10
MAC layer	IEEE 802.11

Table 1: Parameters for the basic simulations

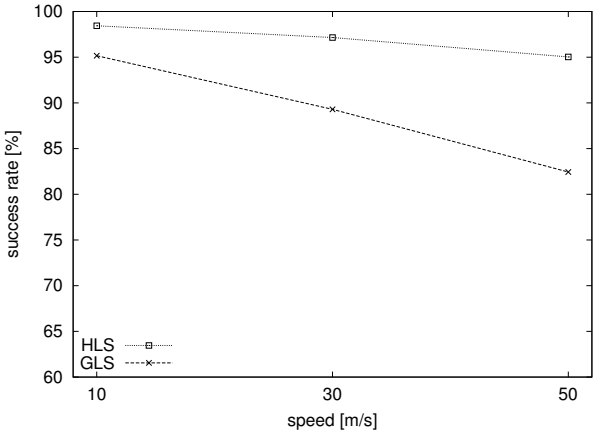


Figure 9: The percentage of successfully answered queries of HLS and GLS with perimeter forwarding for 75 nodes per square kilometer

IV.A. Success rate

The *success rate* is the percentage of queries which have been successfully answered by the location service. A query is answered successfully if the location service can determine the position of the target node with a precision of at least 250 m, either by looking it up in the local cache or by sending a request and receiving a reply. The results are shown in Figure 9.

In the curves of Figure 9, HLS achieves success rates of 95 to 98 %. This demonstrates that the mechanism which keeps the location information in the responsible cells works well even for fast moving nodes and that requests are able to find this information. The Grid Location Service performance varies between 82 and 95 %. The drop of the GLS success rate at higher speed is partly an effect of its aggressive caching: for fast moving nodes, cached information is quickly outdated. As a consequence an increasing number of requests answered through local lookups fail to provide a result that is within the 250m range of the target's true position.

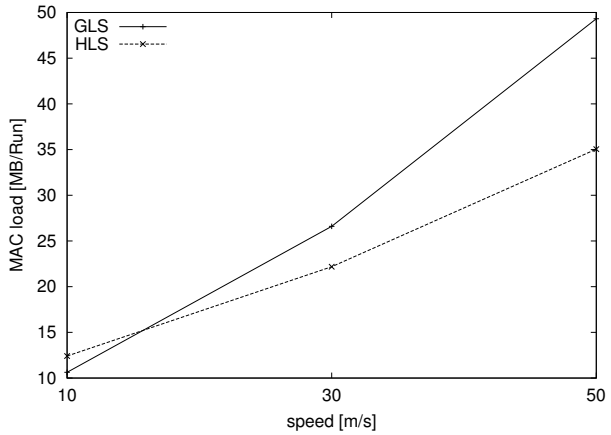


Figure 10: MAC Load for 75 nodes per square kilometer for the same runs as in Figure 9

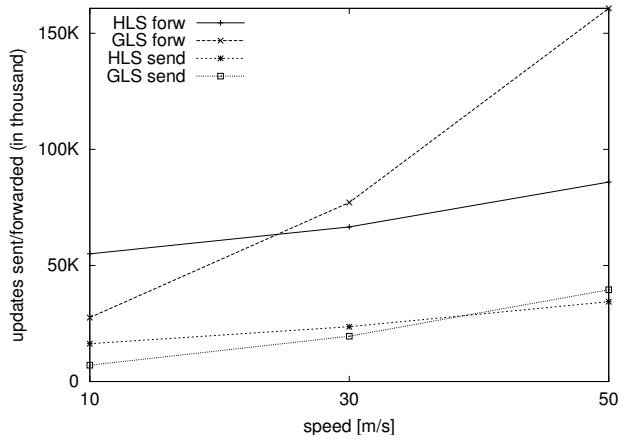


Figure 11: Sent and forwarded updates of HLS and GLS for 75 nodes per square kilometer

IV.B. MAC Load

As a second criterion, we were interested in the MAC load of the location services. The result can be found in Figure 10. HLS produces more traffic for low speeds, mainly due to time-triggered updates. As speed increases, it uses less bandwidth than GLS. The reasons for this can be found in Figure 11 where the number of sent and forwarded update packets is shown. While both location services tend to produce nearly a similar number of updates, HLS does not forward them as often as GLS, it can be concluded that the average hops per update packet for GLS increases steeper than for HLS. The explanation for this lies in the target of update packets: while a GLS packet must find a special node within a square, HLS forwards its updates to cells with well-known positions. This is independent of node speed.

number of nodes	300, 675, 1200, 1875
node density per square kilometer	75
area size (kilometer×kilometer)	2×2, 3×3, 4×4, 5×5
max speed (m/s)	10, 30, 50
simulation time (seconds)	150
requests per node	2
requests per simulation run	600, 1350, 2400, 3750
number of runs	5
MAC layer	NULL MAC

Table 2: Parameters for the large scale simulations

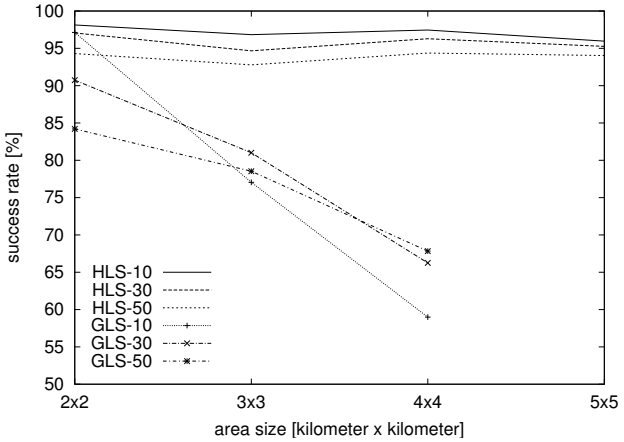


Figure 12: The percentage of successfully answered queries of HLS and GLS in the large scale simulations

IV.C. Large scale simulations

To evaluate the scalability of HLS with respect to network size, we increased the area in which the network is deployed with a constant node density of 75 nodes per square kilometer. Due to memory limitations, a NULL MAC was used instead of IEEE 802.11. The parameters of the large scale simulations are presented in Table 2.

Figure 12 shows the success rates achieved by both location services. Note that the x-axis models area size here in contrast to speed in previous graphs. The curves are labelled with the name of the location service and the maximum node speed. For the area size of 5×5 kilometers, only the success rates of the Hierarchical Location Service are shown, the GLS simulations needed too much memory in this scenario.

In all of the simulations, HLS' success rate varies only slightly for all simulated area sizes which shows the good scalability of HLS with growing area size. In contrast to this, the success rate of the Grid Location Service drops as the network gets larger. The

GLS routing scheme for updates and requests seems responsible for this descent: update and request packets rely on other nodes' location servers such that they are forwarded to the correct node. Inconsistencies in location information may lead to detours or even routing loops [8]. Following the GLS algorithm, updates and requests are forwarded to the node T which is closest to the sender S in ID space in a sibling of the square which contains S [10]. These siblings can have a size of 2×2 kilometers in the 4×4 kilometer simulation. Accordingly, updates and requests must be able to find T in an area with a size of 4 square kilometers which is a difficult task in a MANET with fast moving nodes. The Hierarchical Location Service does not face this problems. The target of updates and requests is always a cell with a fixed position. Node speed and area size do not have any influence on this.

An interesting effect which is not directly related to our Location Service comparison is indicated by the crossing curves of GLS: the success rate at a maximum speed of 10 m/s is the highest for the area size of 2×2 kilometers whereas it is the lowest for the 4×4 kilometer area compared to the success rates at higher node speeds. In general, low speeds seem to be good for smaller areas and bad as the area size increases. This is not at all intuitive, therefore we need to investigate Figure 13. It shows the percentage of queries which are answered locally from the cache[4] (CLR: Cache Lookup Rate) at different node speeds for both location services and thus the percentage of nodes about which a node has local knowledge. It can be seen that more local knowledge is available with growing node speed. This has two reasons: if nodes move faster, a higher number of updates are sent which increases the availability of information. Moreover, a node driving at high speed receives beacons of more nodes passing by than a node at low speed.

The high success rate of GLS at 10 m/s in the 2×2 kilometer area suggests that GLS is able to locate the target in this scenario. The lower success rates for 50 m/s in contrast show the negative impact of outdated information in the cache. As the area size increases, it gets more difficult to find a location server for the target of a request as explained above. The information in the cache is very valuable in these scenarios and significantly contributes to the success rate.

IV.D. Impact of node density

In order to get an impression of the impact that node density has on HLS, we also simulated scenarios with

[4]This also includes the cache lookups with a deviation of more than 250 meters.

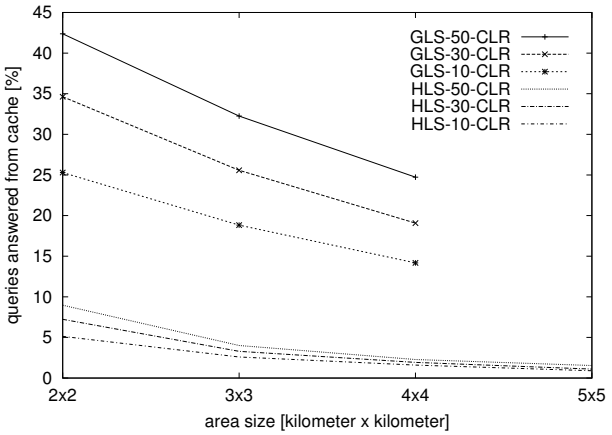
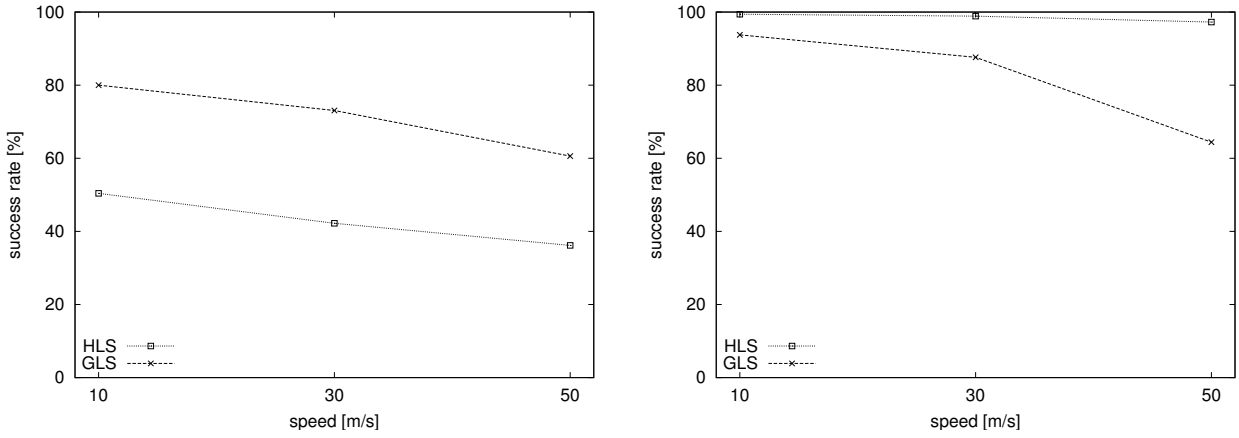


Figure 13: Cache lookup rates of HLS and GLS in the large scale simulations for different nodes speeds

higher and lower node densities than 75 nodes per square kilometer. The HLS success rates improve with the higher node density as one can see in Figure 14(b). As node density increases, the probability of empty cells is reduced. Requests routed to a responsible cell find a location server in most of the cases. It is furthermore interesting that node speed only slightly decreases the performance of HLS: even in the extreme scenario with 100 nodes per square kilometer at a maximum speed of 50 m/s, 97.9% of all queries are answered correctly. GLS achieves a much lower success rate here because it produces a lot of update packets resulting in high network load and, finally, in network congestion.

In the graph with 25 nodes per square kilometer (Figure 14(a)), HLS achieves success rates of around 40 % whereas GLS correctly answers 60 % to 80 % of all queries. One reason for this difference are the different levels of caching used by the two implementations, GLS answers 45 % to 55 % of all queries from the cache in contrast to a maximum of 10 % for HLS. Another issue is a problem of the association of location information with cells in HLS. If cells do not contain a node as it often happens at this low node density, it is difficult to find a location server.

The results show that the hierarchical location-based system is very efficient especially at high speeds. At low node densities, HLS' performance is mainly degraded due to empty cells. This can be enhanced either by a more aggressive caching strategy or by implementing a more sophisticated storage system for the location information which is better able to cope with empty cells. If node density is above a certain minimum, in our simulations 75 nodes per square kilometers, HLS reaches overall success rates of at least 92 % at all simulated node speeds and area sizes.



(a) 25 Nodes per square kilometer

(b) 100 Nodes per square kilometer

Figure 14: The percentage of successfully answered queries of HLS and GLS at varying nodes densities

V. Conclusions and Outlook

In this paper we have presented the Hierarchical Location Service (HLS). HLS uses a hierarchy of regions to achieve scalability and remain robust at high node speeds. We have shown by means of simulation that HLS achieves good results at low communication cost. It is only slightly influenced by high node speeds and scales well with the geographic extension of the network due to its pure position-based routing of updates and requests. Even in the largest scenario simulated, an area of 5×5 kilometers populated with 1875 nodes, HLS' performance did not degrade perceptibly compared to smaller scenarios[5].

Our simulations also revealed areas where HLS can be improved further, in particular for networks with low node density where empty cells are frequent. We see two main directions that can be followed in order to alleviate this problem: the use of more aggressive caching strategies and reconsidering how position information is stored. Even though the former requires a careful consideration regarding the time a cached entry remains valid, it can be realized as a conceptually straight forward adaptation of existing approaches. The latter requires more extensive modifications: In HLS there exists one location server in each responsible cell. This causes problems when the location server leaves the cell (because the location information has to be handed back into the cell) or simply fails. More importantly it reduces the success rate for low node densities because an increasing number of responsible cells no longer contain any nodes that can act as a location server. It may therefore be worthwhile to investigate the use of the Geographic Hash Table (GHT) system proposed in [13] which has been initially developed for data storage in sensor networks. GHT stores information in those nodes that form a perimeter around a point. In case of HLS this point would be the center of the responsible cell. The node closest to this point is responsible for regularly sending the information to the other members of the perimeter. If the closest node fails, another member of the perimeter will sent an update after a timer has expired and a new closest node can be elected. The main benefits of using GHT would be that responsible cells may be empty without any negative impact (the perimeter used for storing the location information would simply be outside of the responsible cell) and that the robustness is increased. A disadvantage of using GHT may be the increased complexity of the mechanism in combination with a higher network load for maintaining the information in the GHT system.

In addition to these two main directions of future work dead reckoning could be employed to prolong the validity of location information. Furthermore, if all nodes in the network move in the same direction and therefore most of the predefined cells are empty, the HLS algorithm should be adapted to this situation. Another interesting idea are cells moving according to a predictable pattern, e.g. along a highway. With the time information of the GPS signal, it can be computed at which point of the highway the cells are located. A location server moving along the highway may stay location server for a long time because it stays in the cell which is also moving. Therefore, less

[5] Due to computational limitations, this scenario has been simulated with a simplified MAC layer which does not produce collisions.

handovers are necessary and communication overhead is reduced while the functionality stays the same.

Concluding, we believe that a hierarchical approach where cells are responsible to maintain position information is very promising as location service for MANETs. While there are still many ways to improve upon HLS as presented here, we were able to show that in particular for settings with large networks, high node mobility or moderate to high node density HLS already outperforms existing approaches.

References

[1] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the fourth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '98)*, pages 76–84, Dallas, Texas, October 1998.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc Wireless Networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIAL-M '99)*, pages 48–55, Seattle, WS, August 1999.

[3] S. Giordano and M. Hamdi. Mobility Management: The Virtual Home Region. Technical Report SSC/1999/037, EPFL-ICA, October 1999.

[4] P. Gupta and P. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[5] B. N. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the sixth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '00)*, pages 243–254, Boston, Massachusetts, August 2000.

[6] M. Käsemann. Beaconless Position-Based Routing for Mobile Ad-Hoc Networks. Master's thesis, Department of Mathematics and Computer Science, University of Mannheim, February 2003.

[7] M. Käsemann, H. Füßler, H. Hartenstein, and M. Mauve. A Reactive Location Service for Mobile Ad Hoc Networks. Technical Report TR-02-014, Department of Computer Science, University of Mannheim, November 2002.

[8] M. Käsemann, H. Hartenstein, H. Füßler, and M. Mauve. Analysis of a Location Service for Position-Based Routing in Mobile Ad Hoc Networks. In *Proceedings of the 1st German Workshop on Mobile Ad-hoc Networking (WMAN 2002)*, GI – Lecture Notes in Informatics, pages 121–133, March 2002.

[9] W. Kieß. Hierarchical Location Service for Mobile Ad-hoc Networks. Master's thesis, Department of Computer Science, University of Mannheim, 2003.

[10] J. Li, J. Jannotti, D. S. J. DeCouto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proceedings of the sixth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '00)*, pages 120–130, Boston, Massachusetts, August 2000.

[11] M. Mauve, J. Widmer, and H. Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network*, 15(6):30–39, November/December 2001.

[12] The ns-2 network simulator. http://www.isi.edu/nsnam/ns/.

[13] S. Ratnasamy, B. N. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications (WSNA 2002)*, pages 78–87, Atlanta, Georgia, September 2002.

[14] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proceedings of the IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

[15] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. In *Proc. of the IEEE Conference on Local Computer Networks (LCN'2001)*, Tampa, Florida, November 2001.