Hierarchical Location Service for Mobile Ad-hoc Networks

Diplomarbeit
von

Wolfgang Kieß

aus
Beltersrot

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Prof. Dr. Wolfgang Effelsberg
Universtität Mannheim

März 2003

Betreuer:
Dr. Martin Mauve
Dipl. Wirtsch.-Inf. Holger Füßler
Dipl. Wirtsch.-Inf. Jörg Widmer

Acknowledgments

A lot of people supported me during my work on this thesis to whom I wish to express my gratitude.

First of all, I would like to thank Dr. Martin Mauve, Holger Füßler and Jörg Widmer for their support, ideas, valuable comments and the interesting discussions.

I would like to thank Michael Käsemann for helping me with his detailed knowledge about ns-2 and for all the little tools he programmed.

I would like to thank Christoph Gütter who gave me a lot of feedback both on language and on technical details of this thesis.

Furthermore I would like to thank Evelyn Naudorf, Stefan Weber, Carolin Kiess, Anne Hartkopf and Katie Haskins for removing the worst errors and Björn Münstermann for giving me support in mathematical issues.

Contents

List of Figures

List of Tables

Nomenclature

AGPF	Anchored Geodesic Packet Forwarding
AODV	Ad-hoc On-demand Distance-Vector routing
DREAM	Distance Routing Effect Algorithm for Mobility
DSDV	Destination-Sequenced Distance-Vector routing
DSR	Dynamic Source Routing
GLONASS	GLObal NAvigation Satellite System
GLS	Grid Location Service
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GRSS	Geographical Region Summary Service
HLS	Hierarchical Location Service
IEEE	Institute of Electrical and Electronics Engineers
IFQ	InterFace Queue
ISO	International Organization for Standardization
LAN	Local Area Network
LAR	Location Aided Routing

LSI	Location Server Identification
MAC	Medium Access Control
MANET	Mobile Ad-Hoc Network
OLSR	Optimized Link State Routing
OSI	Open Systems Interconnection
RC	Responsible Cell
RFC	Request For Comment
RLS	Reactive Location Service
SPA	Self Positioning Algorithm
TBRPF	Topology Broadcast based on Reverse-Path Forwarding
TDM	Time Division Multiplexing
TLR	Terminode Local Routing
TORA	Temporally-Ordered Routing Algorithm
TTL	Time To Live
VINT	Virtual InterNetwork Testbed
ZRP	Zone Routing Protocol

Chapter 1

Introduction

Most wireless, satellite- or cable-based communication networks have a major drawback: they need a pre-installed infrastructure. If this infrastructure is not available, e.g. after a disaster, during expeditions in remote regions or even other planets, on the battlefield and during inter-vehicular communication, data exchange in infrastructure-based networks becomes impossible. Here, *mobile ad-hoc networks* (MANETs) are the network technology of choice. In a MANET, autonomous nodes form a collaborative community, each node simultaneously acts as end-system and router.

1.1 Problem Statement

Routing in MANETs is a challenging task. In early approaches, proactive routing protocols were suggested. These protocols maintain information about paths in a network even if these paths are not in use. In highly mobile ad-hoc networks where the topology changes frequently, maintaining unused paths tends to consume a lot of the available bandwidth. As a consequence, reactive routing protocols were developed which only create routes if necessary. Nevertheless, these routes need to be maintained during a communication relation. This may be difficult with frequent topology changes.

Position-based routing[1], where the routing decision is based only on the posi-

[1]Also called geographic routing.

tion of the communication partners, offers a scalable solution to this problem. An important task when using position-based routing is to determine the position of a distant communication partner. This is accomplished by a location service.

1.2 Contribution

In this thesis we introduce the *Hierarchical Location Service* (HLS). The basic idea of HLS is to divide the area of the network into smaller areas called cells and assign each node a set S of these cells. Position updates and requests are sent to (possibly different) subsets of S. The selection of the subset depends on a hierarchical grouping of the cells of S and the position of the node which computes the subset. The HLS algorithm guarantees that the intersection of two subsets computed for the same node is non-empty.

We present a theoretical comparison of HLS to existing location services in the first part. The main part deals with the evaluation of a prototypical implementation of the Hierarchical Location Service which will be compared to the well-known *Grid Location Service* (GLS).

1.3 Structure

The second chapter contains an overview of mobile ad-hoc network basics, gives important definitions and introduces some well-known location services as well as position-based routing algorithms. Chapter 3 presents a detailed description of the Hierarchical Location Service and some potential improvements. In Chapter 4, we give details about our implementation of HLS and GLS with ns-2 version 2.1b8a([NS2]). Simulation results are presented in Chapter 5, comparing the performance of HLS to that of GLS. Based on these results, HLS' behavior is then analyzed in detail. This thesis is concluded by an outlook on possible extensions of the Hierarchical Location Service.

Chapter 2

Foundations of Ad-hoc Networking

In this chapter, we will explain basic concepts of mobile ad-hoc networks and introduce the most important definitions. We will then point out some of the problems that have to be faced in the context of mobile ad-hoc networks and discuss existing solutions to these problems.

2.1 Mobile Ad-hoc Networks

Ad-hoc networks are wireless communication networks that function independently of any infrastructure. If the nodes, i.e. the members of this network, are passive, the communication area of a single node is limited to its radio range, e.g. Bluetooth [Gro] or IEEE 802.11 [IEE99] in ad-hoc mode. The strength of ad-hoc networks lies in cooperation: a participating node may operate as both end-system and router. As a result, each node can communicate with other nodes that may be dozens of hops away. If we talk about ad-hoc networks in the following, we always mean networks which make use of this form of cooperation.

Ad-hoc networks can be static or mobile. With static ad-hoc networks being a subset of mobile ad-hoc networks, this work concentrates on the latter. A typical scenario for a mobile ad-hoc network is a number of cars driving on a highway and exchanging telemetry information, e.g. while driving in a group. Thus, these networks can have hundreds of members moving with high speed (not necessarily in the same direction) and spread across several kilometers.

2.2 Routing

Routing is the process of directing a data packet from its source to its destination in a communication network. Following [MWH01], routing protocols for mobile ad-hoc networks can be subdivided in

- *topology-based*: packets are routed by using topology information about the network. This information can be supplied as follows, for details see [RT99, BMJ$^+$98]:
  - *proactive*: maintaining routes before any data traffic is sent, e.g. DSDV [PB94], TBRPF [OTBL01], OLSR [CJL$^+$01]
  - *reactive*: doing a route discovery only if a connection is to be established (like DSR [JM96], AODV [PR99], TORA [PC97])
  - *hybrid*: mixing local proactive and global reactive routing to increase scalability (e.g. as proposed in ZRP [HH01])
- *position-based*: packets are routed according to the geographic position of the communication partner. This position is supplied by a location service, for details see Section 2.3.

When using topology-based routing protocols, communication overhead depends, among other things, on the change rate of the network structure. In a MANET which mainly consists of fast moving nodes, this change rate can be high, resulting in a lot of communication overhead to keep the routing information in the nodes up to date. Thus, the routing protocols further examined in the remainder of this section are all position-based.

In contrast to topology-based routing, nodes must be able to determine their own position and that of other nodes in the network when they use position-based routing. The first of these tasks is accomplished by a *positioning service*. There are quite a few of those services, satellite based ones like GPS (USA), GLONASS (Russia)[GLO] or Galileo (Europe)[GAL], where the signal propagation delay of satellites' radio emissions is used to calculate absolute positions with an accuracy of a few meters. But there are also positioning services that do not require this kind of infrastructure. As an example the Self-Positioning Algorithm (SPA) [CHH02] can be mentioned which measures the distances between nodes using radio signal propagation delay. SPA then establishes a relative network coordinate system based on the distance information.

To determine the position of other nodes within the network, a node can use the location service mentioned above. For the overview of routing protocols presented in this section, we assume that every node has the ability to learn the position of its communication partner by asking its local interface to this service.

The basic idea of position-based routing is to forward a packet to the last-known location of the target node. If the position information is sufficiently exact, the node is within radio range of its last-known position and the packet can be delivered. Local routing decisions are based on the position of the target, the position of the forwarding node itself and the positions of its neighbors. A few algorithms based on the concept of position-based routing will be presented below.

2.2.1 GPSR

Greedy Perimeter Stateless Routing (GPSR) [KK00] is a routing protocol that makes ”aggressive use of geography to achieve scalability” as expressed by [KK00]. It is a combination of *greedy routing* and *perimeter routing*. When using GPSR, each node periodically sends 1-hop broadcast messages, termed *beacons*, containing the node id and current position. Due to these beacons, every node is able to maintain a list of its 1-hop neighbors. Whenever a node does not receive a beacon from a previously known neighbor for a certain timespan, it assumes that the neighbor is out of reach and is deleted from the list of neighbors.

Greedy routing forwards a data packet to the neighbor closest to the target position of the packet. It approaches the target node with every hop until it is close enough to be delivered[1]. While this works well for dense scenarios, cases occur where packets are likely to be dropped in sparsely populated scenarios. The resulting effect is shown in Figure 2.1. S sends a packet to D, the packet is forwarded in greedy mode until it arrives at node F. Here, it reaches a local

[1]In general, this is possible as long as the target node still is in radio range of its last known position.

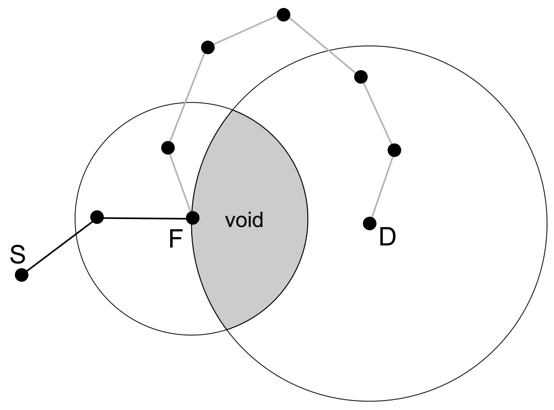


Figure 2.1: Reaching a local maximum. (inspired by [MWH01])

maximum and cannot be forwarded even though a route to the destination exists (the gray line from F to D). To take this route however, the packet would have to be forwarded over nodes farther away from D than F, which contradicts the principle of greedy routing.

The intersection of the two circles is called ”void”, greedy routing fails when the packet reaches a node with a void. Perimeter routing is a method used to route around this void. It uses the well-known *right-hand rule* to do so. As expressed in [KK00]:

> This rule states that when arriving at node x from node y, the next edge traversed is the next one sequentially counterclockwise about x from edge (x,y).

The sequence of edges which is found by using the right-hand rule is called *perimeter*. To avoid crossing edges, which can hinder a packet forwarded with the right-hand rule to find a way around a void [Kar00], the graph is planarized[2] before applying the rule. This is achieved by a distributed algorithm using only local information. The algorithm produces either a relative neighborhood graph [Tou80] or a gabriel graph [GS69].

The cooperation of greedy routing and perimeter routing is shown in Figure 2.2. The solid lines signify the hops where the packet from source S to destination D is forwarded in greedy mode. Whenever the packet reaches a node

[2] A graph is *planar* when no edges cross.

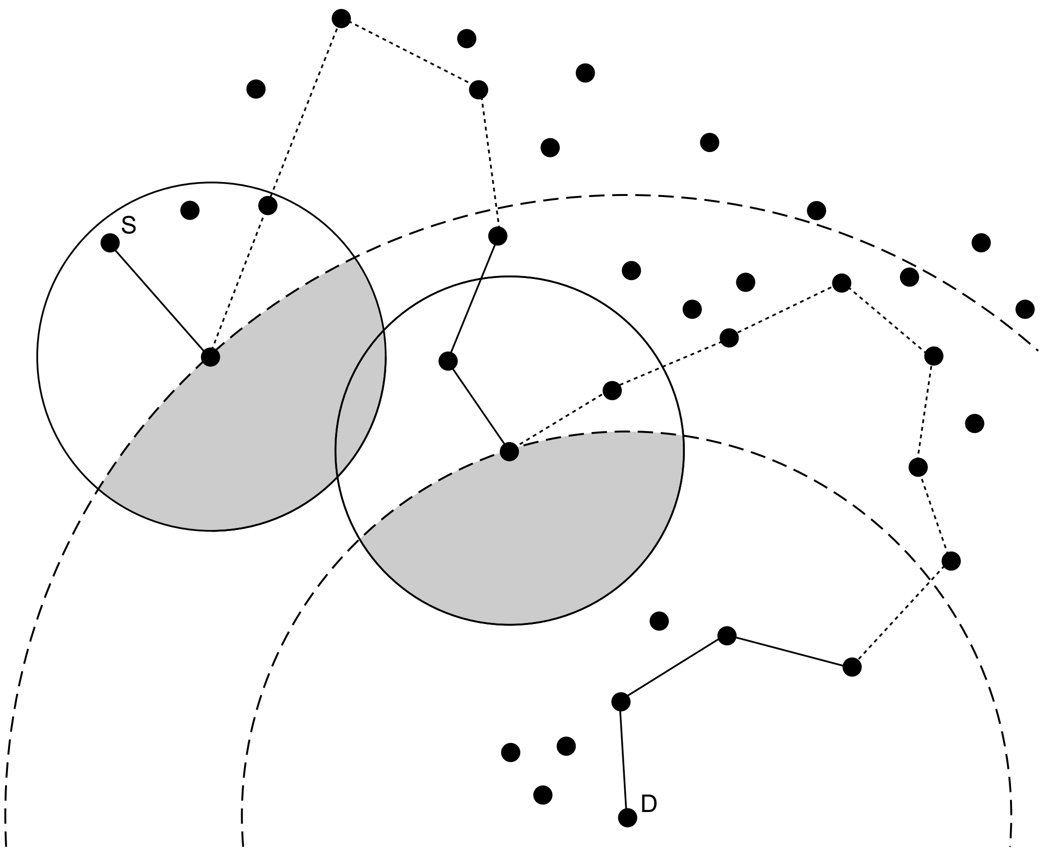


Figure 2.2: An example of GPSR routing

which has no neighbor closer to D within radio range (the dark shaded area marks the void), the packet is switched to perimeter mode and forwarded according to the right-hand rule, marked as dotted line. The solid circles show the radio range of the nodes, the dotted circle segments the distance to the destination when the packet is switched to perimeter mode. The packet leaves the perimeter mode when it reaches a node which is closer to the destination than the perimeter entry point.

The Face-2 [BMSU99] algorithm works in a similar fashion. For the remainder of this work, we use the GPSR terminology.

2.2.2 Terminodes Routing

Terminodes Routing ([BGL00, BGL01]) consists of *Anchored Geodesic Packet Forwarding* (AGPF) and *Terminode Local Routing* (TLR). It was developed within the terminodes project ([HGBV01]). TLR uses source routing to maintain routes to neighbors close to a node. It thus defines a neighborhood where all nodes know the position of each other. In AGPF, the source includes a list of anchors in the packet. An anchor is a geographic point on the route which has to be visited. The anchors are computed either based on a map or on information provided by other nodes. Between anchors, AGPF uses greedy routing similar to GPSR. If a packet reaches a node from which the anchor can be reached within a predefined number of hops, it is forwarded to the next anchor. In case the target lies within the same neighborhood, the packet is forwarded to the target.

2.2.3 LAR

Location Aided Routing (LAR) [KV98] is a method used in reactive routing protocols during route discovery to limit flooding. With the location information available, an expected zone (i.e. the expected location of the target) and a request zone are defined. The latter marks the area to which the route discovery packet is geocasted[3] [NI97], the request zone normally contains the expected zone.

2.3 Location Service

A *location service* in the context of a mobile ad-hoc network is a distributed service which answers location queries of the type: ”where is node X?”. Therefore, certain nodes in the network can act as *location servers* for X, which means that they host location information about this node. In general, a location request is sent to a location server and forwarded to the target node which generates an answer. Depending on the location service employed, different criteria are used to select these location servers.

Using the classification in [MWH01], location services can be subdivided by the number of nodes that host the service (*some* or *all* nodes act as location servers) and by the amount of information hosted in each location server (each server contains information about *some* or *all* nodes of the network).

In short:

[3] A geocast packet is delivered to all nodes in a certain geographical area.

- some-for-some
- some-for-all
- all-for-some
- all-for-all

If the selected location servers host information for all nodes (some-for-all and all-for-all approach), the update overhead is high in bigger networks because all nodes must transmit their position to these location servers. Therefore, the system may face scalability problems.

Additional criteria for the classification of location services are:

structure A location service can be *hierarchical*, which means it establishes a certain type of hierarchy. In general, this is used to achieve a higher concentration of location servers close to the actual position of a node. The greater the distance, the lower the concentration of position information. Generally speaking, the number of hierarchy levels is adapted to the size of the network. However, there are also approaches with a fixed number of levels. If the location service does not use any hierarchy, its structure is *flat*.

location server identification (LSI) If the location service uses location servers, these servers can be identified either by their node id (*id-based LSI*) or by their actual position (*position-based LSI*). In the latter case, servers are interchangeable as long as they are located in a certain area.

area division Most location services using area divisions also use a hierarchical structure. The hierarchies are based on dividing the area into different regions and use sets of these regions to form regions of a higher level.

update and request strategy The *update* and *request strategy* describes the method used by a location service to find location servers. The strategies can be flooding, geocast or unicast. To better describe the behavior of some hierarchical approaches, we introduce an additional strategy called *treewalk*.

When using treewalk, the update and request packets are forwarded according to the treelike structure of the hierarchy, following e.g. a branch from a leaf to the root.

Besides the criteria mentioned above, security and energy consumption are important design goals for mobile ad-hoc network protocols. Energy saving can be achieved by switching the nodes to a sleep mode from time to time in a first approach or is not necessary for certain applications (e.g. inter-vehicular communication). Security however, has not been addressed by any of the location services presented in the following. One of the biggest threats to security is the missing location privacy. Location services are designed to associate a node identifier with a geographical position, a member of the network can thus be constantly observed. This leads to the conclusion that location privacy is a task which should be accomplished by the combination of geographical routing protocol and location service. In fact, the current architecture, i.e. the separated design of routing protocol and location service, can only achieve *location obfuscation*. Location obfuscation denotes the shadowing of a node's actual location, not the total encryption of the position. It can be roughly subdivided in three classes (the entity which illegally wants to track a node's location is termed *intruder* in the following):

full obfuscation No information about the current location of a node is known to any other node in the network. The intruder has to send a location request to the target node that can then decide if it wants to answer the request or not.

partial obfuscation Certain location information is known to some other nodes in the network, but the accuracy of the information depends on the distance between source and target. To get a detailed position, the intruder has to get close to the node in quest or send a location request which then has to be once again answered by the requested node itself.

no obfuscation It is possible for an intruder to track the position of a node from one or more distant locations in the network without the affected node noticing.

In Table 2.1 location services are classified according to these criteria[4]. The location services will be presented in detail in the following sections. In case of a hierarchical structure, the number of hierarchy levels is given. Whenever a location service uses unicast for updates and location requests, multicast may be employed when necessary and supported by the routing protocol.

[4]An explanation of the additional criteria can be found in [MWH01].

Criterion	DREAM	Quorum System	GLS	Homezone	RLS	GRSS	HLS
Type	AfA	SfS	AfS	AfS	SfS	AfA	AfS
CC (Update)	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	0	$O(n)$	$O(\sqrt{n})$
CC (Lookup)	$O(c)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(n)$	$O(c)$	$O(\sqrt{n})$
TC (Update)	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	0	$O(\sqrt{n})$	$O(\sqrt{n})$
TC (Lookup)	$O(c)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(c)$	$O(\sqrt{n})$
State Volume	$O(n)$	$O(c)$	$O(\log n)$	$O(c)$	0	$O(n)$	$O(\log n)$
LI	Yes	No	Yes	No	No	Yes	Yes
Robustness	High	Medium	Medium	Medium	High	High	Medium
IC	Low	High	Medium	Low	Low	Medium	Medium
Structure	Flat	2	$O(\log n)$	Flat	Flat	$O(\log n)$	$O(\log n)$
LSI	-	Id-based	Pos+Id-based	Position-based	-	-	Position-based
Area Division	No	No	Yes	No	No	Yes	Yes
Update Strategy	Flooding	Unicast	Unicast	Unicast	-	Flooding	Geocast
Request Strategy	Geocast	Unicast	Treewalk	Unicast	Flooding	Unicast	Treewalk
L. Obfuscation	no	no	partial	no	full	partial	partial
Abbreviations: AfA=All-for-All, SfS=Some-for-Some, AfS=All-for-Some n=Number of Nodes c=Constant LI=Localized Information CC, TC, IC=Communication, Time, Implementation Complexity LSI=Location Server Identification							

Table 2.1: Comparison of different location services. This is an extended version of the one presented in [MWH01]

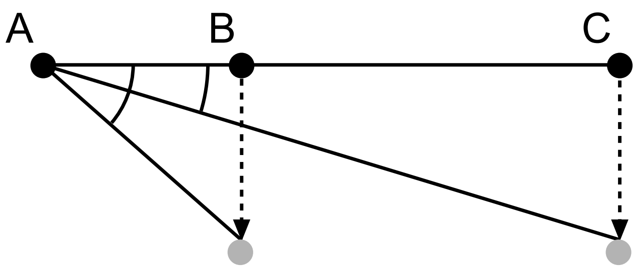


Figure 2.3: The distance effect

2.3.1 DREAM

The *Distance Routing Effect Algorithm for Mobility* [BCSW98] uses the so called *distance effect* shown in Figure 2.3: in case two nodes B and C move with similar speed, the direction to the node which is farther away changes slower. The nodes incorporate this effect in their update strategy. DREAM uses flooding to spread position information with varying flooding frequency and distance by combining the principles: the higher the flooding distance, the lower the frequency; the higher the node movement speed, the higher the frequency.

When we assume for example that a packet can reach any other node in the network with 15 hops, the following basic update strategy could be applied according to DREAM: each node floods position information to its 2-hop neighborhood[5] every 5 seconds, to its 5-hop neighborhood every 20 seconds and to its 15-hop neighborhood every 120 seconds. Nodes furthermore vary the pause times between updates according to their speed. As a consequence, each node has position information about every other node. Accuracy of position information about a node increases with reduced distance to this node.

To send a packet to a node T, a node S sends this packet to all its 1-hop neighbors that ”lie in the direction” of T (see [MWH01, BCSW98] for details). This is repeated until the packet reaches T. Figure 2.4 shows how this works schematically. The gray areas mark the zones to which updates have been flooded together with the flooding interval in seconds. The white area is the zone in which all nodes forward a packet from S to T. As soon as the packet reaches nodes which

[5]N-node neighborhood means: all nodes reachable with n hops.

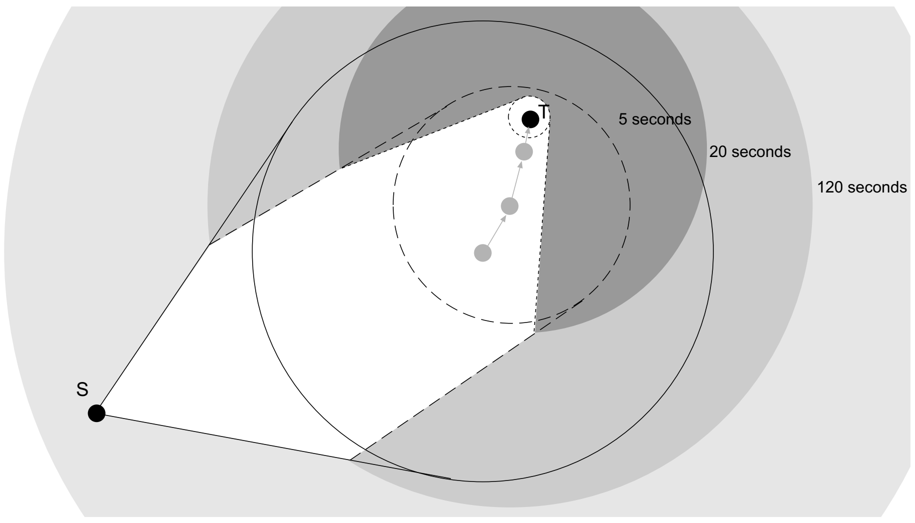


Figure 2.4: DREAM packet delivery

have more exact information about the target (in our example this is the case when entering a new gray zone), the forwarding nodes calculate a new expected zone for the target, here marked as circles.

With DREAM being an all-for-all approach, it has a high update complexity and therefore does not scale to larger networks. On the other hand, the failure of individual nodes does not lead to a collapse of the location service which makes it suitable for applications requiring robustness.

2.3.2 Homezone

The *Homezone* [GH99, Sto99] location service utilizes the concept of a virtual home region. Every node has a homezone, a zone in the area of the ad-hoc network to which the node frequently geocasts its current position. The homezone of each node is assigned by a hash function which is known to all other members of the network. Thus, every node can determine the homezone by applying the hash function on the node's id. It can then send the location request to a node in this zone where it probably can be answered.

Problems occur when there is no node in the respective homezone. This first problem can be solved by dynamically resizing the size of the zone. The second problem disables the homezone location service from scaling for large networks: if nodes are far away from their homezone, they nevertheless need to update it frequently. This leads to a lot of long-distance update traffic which in the end may congest the network.

2.3.3 Quorum System

The location service presented in [HL99], with the name *Quorum System*, is based on an idea mainly used in database systems. Location information is sent to a subset of the available nodes (called *quorum*), location requests are sent to another quorum. Each quorum is constructed in a way that the intersection with any other quorum is non-empty. As a consequence, a location request packet reaches at least one location server for the requested node, the query can be answered.

The paper mentioned above suggest the construction of a virtual backbone of inter-connected nodes which at all times know each other's position. All non-backbone nodes are connected with at least one backbone node. Position updates are sent to the next backbone node which selects a quorum to host the location information. It then sends the update information to this quorum. Location request are treated in the same manner.

2.3.4 Reactive Location Service

The basic idea used in the *Reactive Location Service* (RLS)[KFHM02] is the same as in the route discovery mechanism of DSR [JM96]. A location request for a node is flooded to the whole network. When the request reaches its target, this node generates a reply packet that is returned to the query initiator. In addition to the basic flooding mechanism which is not totally reliable, linear flooding, exponential flooding and binary flooding offer improvements. All approaches influence the flooding distance, either it is incremented in linear steps, in exponential ones or in only two steps, one for near the other for long-distance communication.

The Reactive Location Service neither generates any proactive update packets, nor are location servers necessary. It works in a fully reactive manner.

2.3.5 Grid Location Service

The *Grid Location Service* (GLS) [LJD$^+$00] belongs to the class of location services with a hierarchical structure that has already been analyzed in a number of publications ([KHFM02a, KHFM02b]).

Area Division

The variant proposed in the original GLS-paper [LJD$^+$00] divides the area of the ad-hoc network into hierarchical grids composed of quadratic squares of different sizes. This subdivision is known to all participating nodes. Four squares of the smallest order, referred to as order-1 squares, build one order-2 square, four order-2 squares build one order-3 square and so on. Furthermore, squares of the same order do not overlap. Therefore, a node is located in exactly one square of each order, the squares form a *quadtree*-like structure[6].

Location Servers

A node B recruits a location server in each of the three adjacent squares to its own square of the same order. If there is more than one node in the square, the node with the smallest id greater than that of B becomes location server as shown in Figure 2.5 for node B with the id 17. Nodes 63, 23 and 2 act as location servers in the first-order square. In the second-order square, these are nodes 26, 31 and 43 and in the highest order square nodes 37, 19 and 20.

Position Requests

To find the position of a node B, a node A sends a request to the node with the smallest number greater than that of B for which it has position information. The request is forwarded to nodes with decreasing ids until it reaches a location server for B. From the location server, it can be delivered to B where a reply packet for A is generated.

[6]A quadtree is a tree where each node is split along all d dimensions, leading to 2^d children [nis].

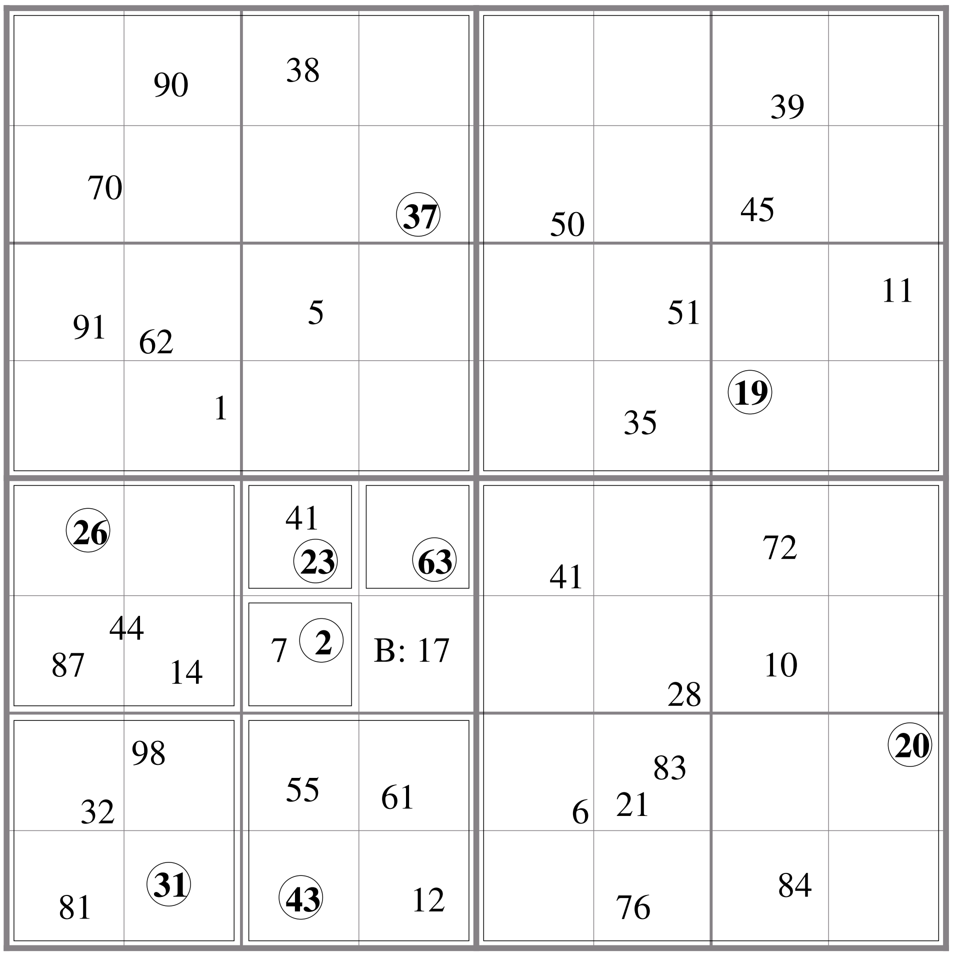


Figure 2.5: Quadtree and Location server, taken from [LJD+00]

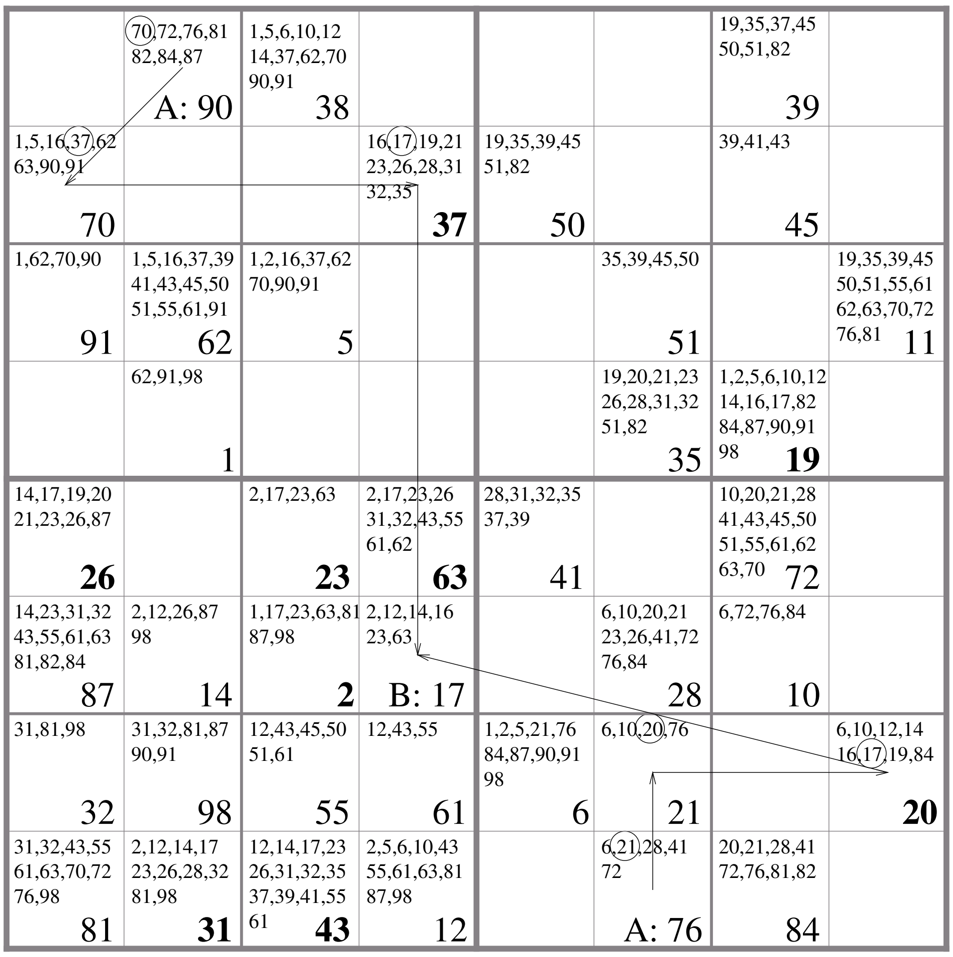


Figure 2.6: GLS location query, taken from [LJD+00]

The algorithm is visualized in Figure 2.6. Node 74 and 90 send a location query for node 17. The request of node 90 gets forwarded to the node with an id closest to that of 17 in the first order element, node 70. Here, the best node known is 37, which also is a location server of the requested node. The request gets relayed to node 17 where a reply packet can be generated and returned to 90. The request of node 74 works in a similar fashion. It is forwarded via 21 in the first order square and 20 in the second order square. Node 20 is a location server for node 17 and forwards the request to the target.

Analysis

The simulations presented in ([LJD^{+}00]) evaluate GLS' behavior within a limited set of scenarios only. A fixed node density of "around 100 nodes per square kilometer" was used as stated in the original paper. A more detailed analysis of GLS can be found in [KHFM02a]. Here, node densities are varied from 25 to 100 nodes per square kilometer in a 2×2 kilometer scenario. All simulations were done using greedy forwarding. The simulations showed that the query success rate decreased with lower node densities, mainly due to lack of greedy-connectivity. Furthermore, dense scenarios with higher speed (300 nodes and 30 m/s, 400 nodes and 30 or 50 m/s) suffered from congestion. In addition, a *forwarding loop error* scenario was presented in [KHFM02a] where inaccurate position information in one location server lead to a routing loop for a location update.

Regarding analytical issues, the Grid Location Service is a complex approach. This may result in difficulties during implementation and lead to errors. The behavior of GLS in rapidly changing networks is currently not well-understood. Inconsistencies in position information about some nodes may cause failures in position lookups and location updates for other nodes. It is furthermore a bandwidth-consuming task to update all location servers frequently in all branches of a quadtree in large networks.

2.3.6 Geographical Region Summary Service

The location service presented in [Hsi01] named *Geographical Region Summary Service* (GRSS) can be classified as an all-for-all approach. It uses a division of the ad-hoc network area in grids with different sizes similar to GLS. A local link-state routing protocol is used to exchange position information between the nodes in the square of the lowest order. In addition to the coordinates of the node itself, a beacon contains a list of all neighbors residing in the same order-0 square[7]. Therefore, each node has complete knowledge of all its 2-hop neighbors belonging to the same order-0 square which is necessary for the proper work of the algorithm. There is, however, one major drawback already stated by the authors of GRSS: ”Because only two hops of information are being exchanged, the square size can not be much larger than the radio range” (taken from [Hsi01]). Solving this by increasing the neighborhood to more than two hops can consume a lot of bandwidth in the beacon headers for the neighbor information.

Nodes close to the boundaries of a square additionally generate summaries about the nodes in their square which are sent as summary updates to the adjacent squares. These summaries can either be exact containing one entry per node (e.g. a bit which is set in a bit vector) or be generated by using a Bloom Filter [Blo70]. The summary generation process is repeated for all levels of the hierarchy, resulting in global knowledge at each node.

Forwarding decisions at intermediate nodes are based only on the id of the target node. If the target node is member of the same lowest order square, the forwarding node should know the exact location of the target and the packet can be delivered. If the exact position of the target is not known, the intermediate nodes forward the packet to the center of the lowest-order square from which it received a summary containing the target node.

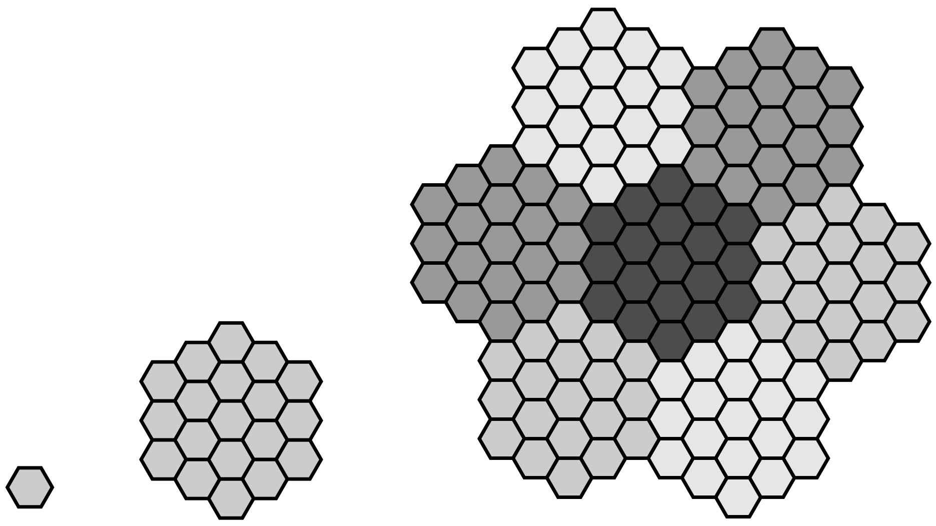
[7]The authors of the original paper referred to the lowest order squares as ”order-0 squares”, we will therefore use their terminology.

Chapter 3

Hierarchical Location Service

A pitfall of the Homezone location service is the fact that for each node exists only one region containing location servers. Updates have to be sent in short intervals to this homezone resulting in frequent (possibly long-range) update traffic. Furthermore, a request packet has to be forwarded to the homezone of the target node even if sender and target are only a few hops away from each other. Duplicating the homezone can reduce request costs but will increase the cost of updates. In MANETs, the maximum number of participating nodes is limited in this way. Analytically speaking, GLS scales better than Homezone because it uses localized information. Distant location servers may be updated less frequently than local ones without degrading the location services' performance. However, GLS is limited by its complex update and request scheme and the non position-based location server selection.

The *Hierarchical Location Service* (HLS) overcomes the limitations of the two location services described above by combining their concepts. The area of an ad-hoc network is partitioned in a way similar to a quadtree. Each node has its own view on the quadtree depending on its position. Due to this view, a node can determine the position of location servers for itself or other nodes. In contrast to GLS, the location server identification is purely position-based. A node becomes location server for another node if it is at the correct position while its node id is irrelevant. Position updates and requests are independent of other nodes' location servers. Update and request packets are delivered by plain geographic forwarding.



(a) cell (b) region on level one (c) region on level two

Figure 3.1: Grouping cells to form regions

3.1 Cells

3.1.1 Structure

The Hierarchical Location Service partitions the area of an ad-hoc network into *cells* and groups them to form *regions*. Each cell and region can be identified by a unique id. The partitioning is fix and known to all nodes in the network. Hexagonal shaped cells cover the area with the least number of cells possible. Other shapes, however, work as well. Each node in a cell c must be able to broadcast packets to all nodes which are member of c. This is achieved either by using cells with a diameter of less than the radio range, by implementing a cell-wide flooding or by any other appropriate mechanism.

The cells are grouped hierarchically as presented in Figure 3.1. A number of adjacent cells form a region of level 1, a number of adjacent level-1 regions form a level-2 region and so on. The process of building regions is continued until one region covers the whole area of the network. This region is called *top-level region*.

3.1.2 Responsible Cells

By applying a hash function on a combination of node id and current position, a *responsible cell* (RC) is assigned to each node for each level. A responsible cell is the cell in which a node recruits a location server.

In other words, a node A selects a number of cells to be its responsible cells whereas the selection depends on the id of A and its current position. To find a location server for A, another node B computes a number of candidate cells by applying the hash function on a combination of A's id and B's position. The design of the hash function guarantees that the intersection between candidate cells and responsible cells is non-empty.

In an abstract way, this can be expressed as follows: let N be the set of nodes, L the set of levels, $X \times Y$ the set of tuples with valid coordinates[1] for a node and C the set of cells. The region on level $l \in L$ covering position $pos \in X \times Y$ is defined as

$$R(l, pos) := \{c \in C \,|\, c \text{ is member of the level-}l\text{ region which contains } pos\}.$$

The hash function F can then be defined as

$$\begin{aligned} F : N \times L \times (X \times Y) &\rightarrow R(l, pos) \text{ with} \\ (n, l, pos) &\mapsto c, \end{aligned}$$

where $n \in N, l \in L, pos \in X \times Y$ and $c \in R(l, pos)$. A hash function which conforms to this specification can be found in Section 4.1.1.

The hierarchy imposed by the hash function is best understood by means of an example (Figure 3.2). It shows all cells which are candidates for responsible cells of a node A in a scenario with regions of three different levels. The *candidate cells* are connected with arrows to visualize their hierarchical, treelike structure which we call *candidate tree*. The root of the tree is the single RC candidate on the highest level, which in this example is located near the center of the area. The candidate tree may be different for each node and can be computed with the

[1] A coordinate is valid if it lies within the boundaries of the area in which the MANET is deployed.

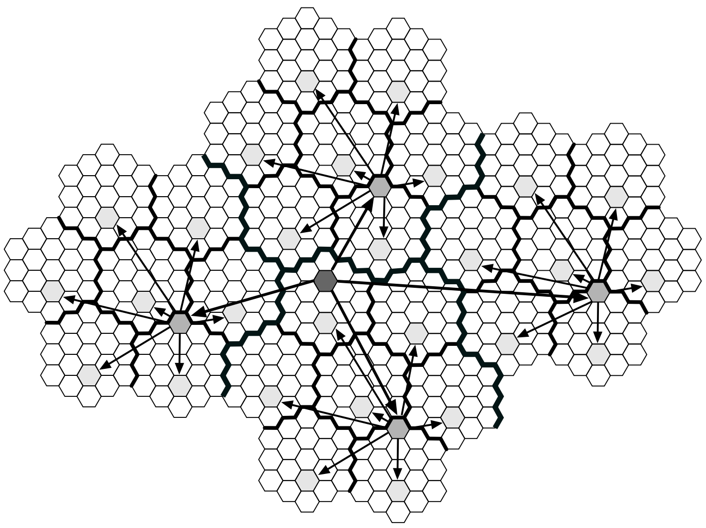


Figure 3.2: Candidate tree in a three level hierarchy

hash function and the node id. To determine which of the candidate cells become responsible cells, a node A calculates the level-1 region which contains its actual position, $r_{1A} = R(1, pos_A)$. It then selects the branch of the tree which ends in r_{1A}. All cells which belong to this branch are responsible cells as presented in Figure 3.3, the RCs and regions are marked with their hierarchy level.

The performance of the Hierarchical Location Service depends on the hash function used to calculate the responsible cells. It should therefore be adapted to the environment in which the MANET is used. We will discuss this issue in the following chapters when necessary.

3.2 Position Updates

There are two different methods for HLS to update location servers, the *direct location scheme* and the *indirect location scheme*. If we say in the following that a node updates a responsible cell, it means that an update packet is sent to the RC and stored at an arbitrary node in or close to that cell which becomes location server. It is possible that subsequent updates arrive at different nodes within a cell. A cell may therefore contain multiple location servers for a node.

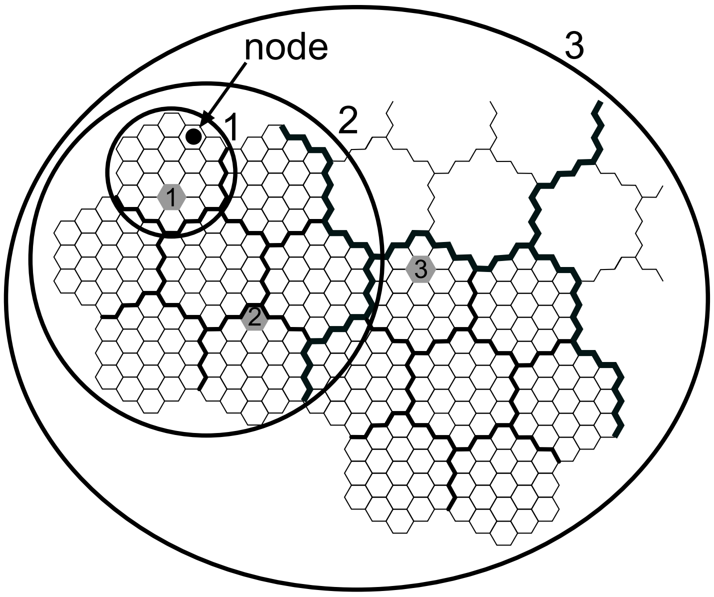


Figure 3.3: Example for responsible cells of a node

To update its location servers according to the direct location scheme, a node computes its responsible cells as explained in Section 3.1.2. 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 Figure 3.5, the location information in the RCs is represented as a pointer to the position of the node. Figure 3.4(a) shows the responsible cells after receiving an update. The RCs on all levels contain exact location information about that node. All location servers must be updated whenever the node has moved a certain distance (Figure 3.4(b)). While this can be easily done for close location servers, updating the RCs on a higher level which tend to be farther away may cause a lot of traffic.

This traffic can be reduced with the indirect location scheme. The location servers on higher hierarchy levels only know in which region of the next lower level a node is located. They do not need to know the exact location. As shown in Figure 3.5(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. For this reason, this update scheme creates ”indirect” location knowledge in the location servers. In an ideal environment with no loss of information, a location server on level n needs to be updated only when the node moves to another level-$(n-1)$ region[2]. 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 3.5(b)). Cells on higher levels need to be updated only if the RC on the next lower level changes (Figure 3.5(c), 3.5(d)). Hence, update traffic generated by a node is generally local. The majority of the update packets have to travel only a few hops whereas long-distance updates are rarely sent.

3.3 Handovers

The identification of a location server depends on its position. Thus, when leaving the responsible cell, a node is no longer location server for information of this RC. 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. This may also be the node which has generated the handover packet if no better node is available.

It can not be guaranteed that a cell always contains a node or that location information is not lost because a location server fails, a packet gets destroyed by collisions and so on. Appropriate techniques to cope with these problems are subject to future research.

3.4 Position Requests

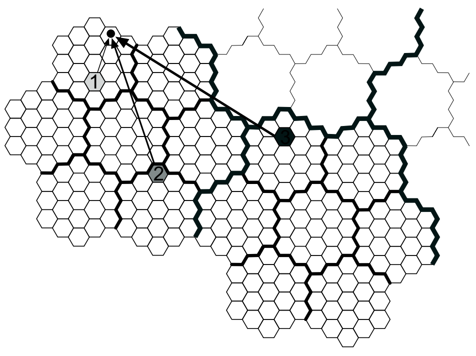
To query the current location of a node T, a node S sends a request packet according to the following algorithm:

1. Compute the candidate tree for T.

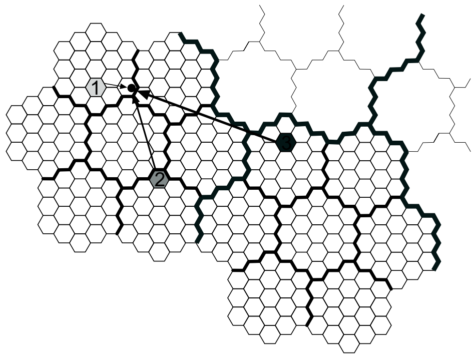
2. Select the branch of the tree which ends in $R(1, pos_S)$.

3. Send the request packet to the candidate cell on level one of this branch.

[2]The loss occurring in a real-world application could be compensated for example by duplicating location servers.

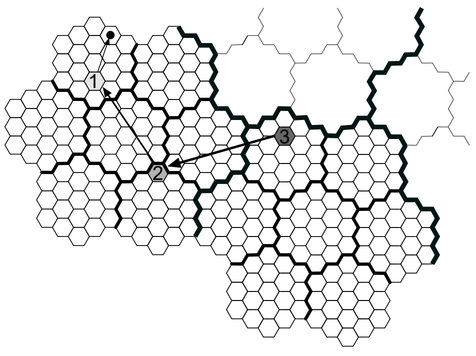


(a) direct location scheme

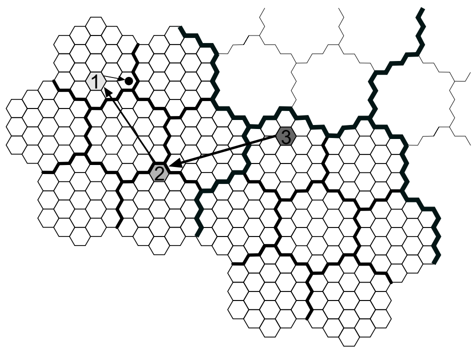


(b) all RCs must be updated

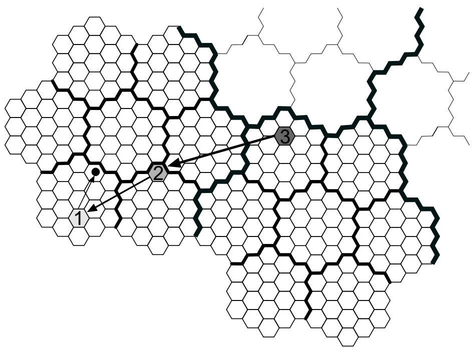
Figure 3.4: Direct location scheme



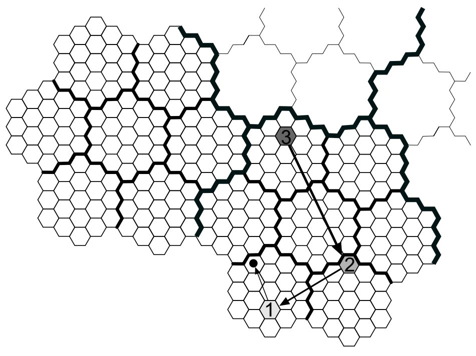
(a) indirect location scheme



(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 3.5: Indirect location scheme

Whenever the request packet arrives at a target candidate cell, more precisely at the node A within the boundaries of the candidate cell which receives the request packet first, it is processed as follows:

1. If node A is a location server for T, it forwards the request to T.
2. Otherwise, A broadcasts (see Section 3.1.1) the request to all nodes within the candidate cell. This is called *cellcast request*.
3. Any node which receives this cellcast request and has location information in its location database sends an answer to A.
4. If A receives an answer for its cellcast request, the request is forwarded to the target node T.
5. Otherwise it forwards it to the candidate cell on the next hierarchy level.

If the routing is not able to forward the request to a node within the target candidate cell, HLS redirects the request to the candidate cell on the next hierarchy level.

With this mechanism, the request is forwarded from candidate cell to candidate cell until it finds a location server for T or no more candidate cells are left. In the latter case, the request has failed[3]. If T receives the request, it sends a reply to S.

The algorithm guarantees that the request is forwarded to at least one candidate cell which is also responsible cell, the top-level RC. In more advantageous cases, the request is already forwarded to a responsible cell on a lower level. 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 selected by T for its updates and the one calculated by S for its request will converge. If r is the region with the smallest i which contains S and T (r = $R(i,pos_S) = R(i,pos_T)$), the branches converge on level i. All candidate cells on a level greater or equal i are also responsible cells.

An example for a request is given in Figure 3.6 for nodes S and T. Here, the location servers are updated according to the indirect location scheme. If the two

[3] Appropriate failure recovery mechanisms are subject to future research. One possible solution might be the artificial home perimeter technique described in Section 4.1.1.

nodes are located in the same level-1 region as in Figure 3.6(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 3.6(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 3.6(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. Therefore, the communication complexity of a request depends on the distance between sender and target of the request. A node needs a location server on each hierarchy level. With the number of hierarchy levels being $O(\log n)$, so is the number of location servers.

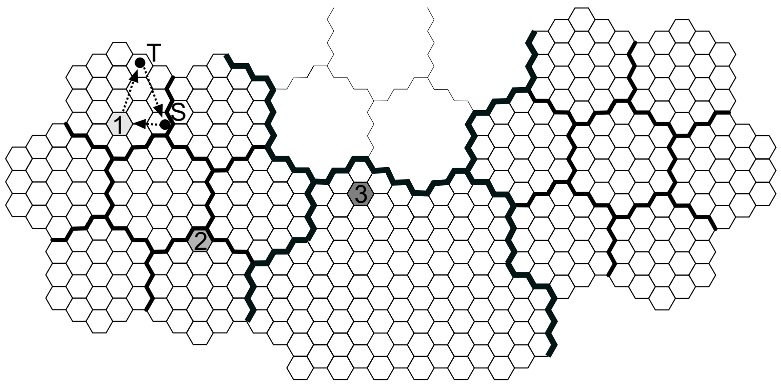
3.5 HLS Extensions

During the research on the Hierarchical Location Service, a lot of ideas for possible extensions and modifications of the basic algorithm emerged. They can be used to improve its performance and to adapt HLS to different scenarios. However, they have not been evaluated in detail. Some of these modifications will be presented in the following.

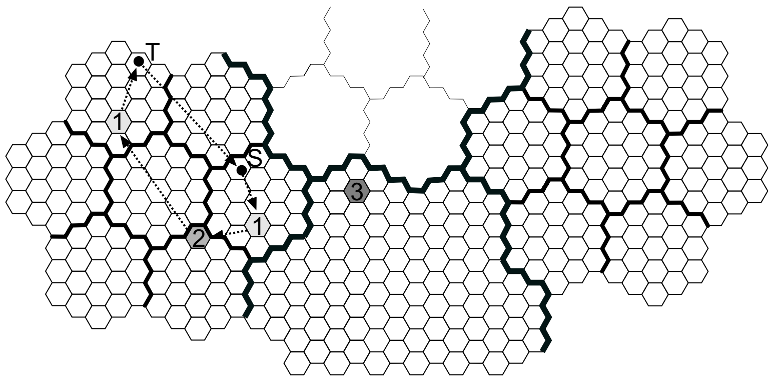
3.5.1 Hybrid Mode

The hybrid mode is an HLS variation for MANETs which are used in scenarios where most of the nodes are concentrated in small, distributed areas (e.g. cities surrounded by countryside). The users are mobile within the city boundaries but will not leave this area too often. Most connections take place between nodes of the same area. The number of connections to nodes outside the area, in contrast, is relatively small.

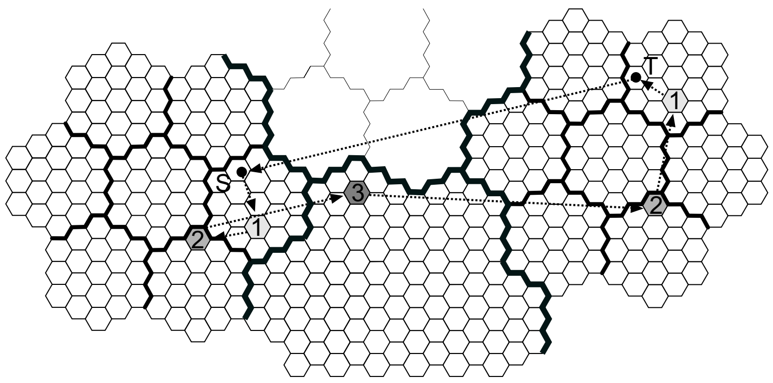
Each node's top-level RC is located in the city area where the node can be



(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 3.6: Example requests

found most of the time. If the corresponding node is in ”its” city, it sends update packets only to a reduced number of hierarchy levels. Otherwise, the basic HLS mode is used with all levels. Requests from outside the city are answered as with the standard HLS algorithm, whereas requests from inside the city area are handled by using the reduced hierarchy.

This approach minimizes the number of location updates in areas with high node density and gives HLS a more reactive character. The hierarchy can be reduced because even sending a request to the responsible cell on the highest level is not expensive in the cities. Sparsely populated zones outside the cities contain less nodes and do therefore not suffer to the same extend from congestion as cities. Furthermore, requests profit more from a fine grained hierarchy here.

3.5.2 Minimizing Proactive Costs

The direct location scheme presented in Section 3.2 produces a lot of long-distance update traffic. This traffic is proactive, i.e. it occurs even if no requests are sent. By modifying the direct location scheme, this traffic can be minimized: updates are only sent to the responsible cells on levels lower or equal a certain $n \in [1,h]$; h : highest hierarchy level.

Then, location discovery has to be performed as follows. Node A sends a request for node B as in the basic HLS. If the request reaches a candidate cell on level n and no location information is found in that cell, the request will be sent to all other candidate cells on that level. At least one of these cells is a responsible cell for node B. The request can then be redirected to node B where an answer is generated.

The advantage of this is the cut of proactive traffic, the disadvantage, however, is the increase in request costs. So far, it has not been explored how this variant of HLS will perform and should therefore be further evaluated. With n, the trade-off between update and request costs can be adjusted, the appropriate value for n is subject to further research.

3.5.3 High Mobility Scenarios

Unfortunately, geographic routing in MANETs bears problems during long-distance communication with fast-moving nodes. If a node S communicates with another node T over a MANET, it will piggyback[4] its current location on outgoing packets to keep T's knowledge about its position up to date. If T replies to these packets, it will use this position information for geographic routing. The packets sent by T can only be delivered to S if the information is sufficiently exact. In general, the deviation should be smaller than the radio range. If A moves with high speed, it may be possible that the information has a bigger deviation and the packet has to be dropped.

Let us investigate the following example: in inter-vehicular communication, mobile nodes may move with speeds of 200 km/h, about 55 m/s. It is assumed that nodes which have moved more than 250 meters away from their last known position are out of reach. As a consequence, nodes are out of reach after approximately 4.6 seconds in this scenario.

If communication distances are small and the answer can be generated quickly, the time between sending a packet and receiving a response to this packet is most likely less than 4.6 seconds. Problems occur if a node wants to communicate over longer distances or with a slowly responding system, e.g. a web email client. Packet retransmissions, timer delay of the ad-hoc routing protocol and computation time can easily amount to 4.6 seconds. As a consequence, the packet will not reach its destination.

Instead of sending these packets to the last known position, they can be directed to the respective level-1 RC where the location information is updated. Using the updated location information, the packet can then be correctly delivered.

3.5.4 Scenarios with High Failure Rates or Sleep Mode

In scenarios with high failure rates or sleep mode (where nodes switch themselves off and on to save energy), a location server may fail or be momentarily unavailable at the time a request arrives.

[4]Piggybacking means attaching data to other packets.

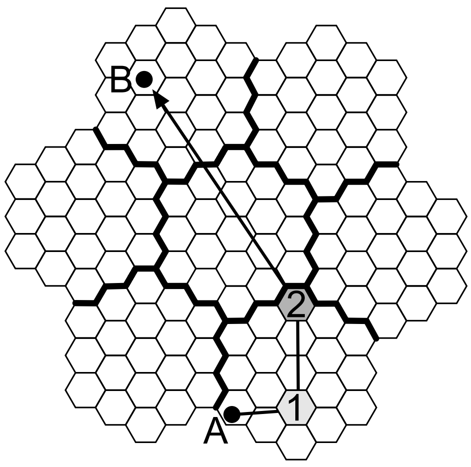
HLS avoids this partially due to its different hierarchy levels. If a node requests an information from a responsible cell in which the information should be available but the respective location server is unreachable, the request will be redirected to the next level. While this already offers a higher degree of failure protection, it does not solve the problem for the top-level RC. The solution is the duplication of the information kept in the RCs. Instead of using just one location server in these cells, the position information can be duplicated several times when arriving and sent to different nodes. While this duplication improves failure protection, it only consumes more storage space and requires a few small-distance transmissions.

3.6 Location Obfuscation

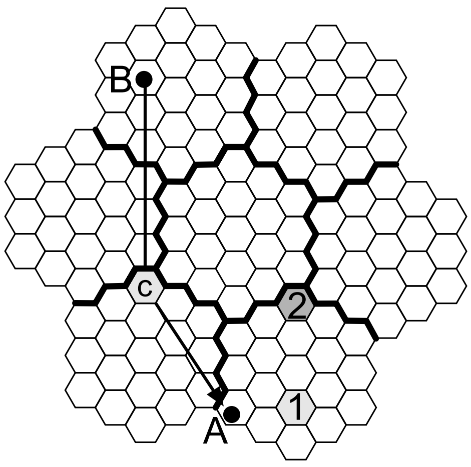
HLS does not consider location obfuscation in the basic draft (for an introduction to location obfuscation, see Section 2.3). At the moment, the most important goal is to develop a scalable location service and to simulate its behavior. Nevertheless, we present some first steps towards location obfuscation in the following sections.

3.6.1 Obfuscation by Design

When using the indirect update mode mentioned in Section 3.2, HLS already obfuscates the position of a node per default. If a location request is sent to a responsible cell, the exact location of the node is unknown. The location server in this cell only knows in which region of the next lower level the node is located. Instead of periodically updating the level-1 RC, the whole system could be changed to work on a reactive basis on this level: a location request always has to be flooded to the level-1 region. The requested node may decide whether to answer the request or not. Consequently, the location can only be determined with a certain degree of accuracy without the affected node noticing.



(a) sending the request



(b) answering via proxy cell

Figure 3.7: Obfuscation with a proxy cell

3.6.2 Proxy Cell

To obfuscate the position of a node during an active communication connection, the location service can use a proxy cell. A location request can be answered only by the requested node itself. If node B receives a request from node A (Figure 3.7(a)), B choses a connection id and a random cell c as proxy cell. This cell is supposed to share the way to the target to avoid long detours. The connection id should be computed in a way that enables A to identify the reply packet, e.g. by using public key cryptography. B sends a reply packet to c (Figure 3.7(b)), it only includes the connection id, not its own node id. At c, B's destination is removed and replaced by that of c. A now receives a reply packet with the proxy cell as source. Due to the random selection of the proxy cell, A does not know the exact location of B. All packets from that connection have to pass c where the addresses are translated. The connection id prevents forwarding nodes from knowing which nodes participate in the connection.

Chapter 4

Implementation

4.1 HLS and ns-2

Simulating the Hierarchical Location Service was performed using the discrete event simulator *ns-2* in version 2.1b8a. It was developed by the VINT-project, its wireless simulation capabilities were added by the Monarch Group at the Carnegie Mellon University [CMU]. Furthermore, the contributions and fixes of the fleetnet project [FN] have been a great support.

4.1.1 Implemented Features

This section lists all features that have been implemented for the HLS evaluation in the next chapter. The greatest difference between this section and the description of the algorithm in Chapter 3 is the cell shape which we changed to improve comparability with GLS.

Cell shape The cells are quadratic, the diagonal d of a cell is equal to the radio range. A side s of a cell has the length

$$length[s] = \lfloor\sqrt{d^2/2}\rfloor = \lfloor\sqrt{250^2/2}\rfloor = 176\ meters$$

According to these dimensions, a node being within the boundaries of a cell can be sure that every node inside the same cell is within radio range. A level-1 region has a size of 3×3 squares, a level-2 region 6×6 squares and a level-3 region, the largest one, 12×12 squares.

As a result, the area partitioning of HLS is almost equal to that of GLS which uses order-one squares with side lengths of 250 meters:

- The GLS implementation partitions the ad-hoc area into squares with side lengths of 250 meters, 500 meters, 1000 meters and 2000 meters.
- The HLS implementation partitions the ad-hoc area into squares with side lengths of 176 meters, 528 meters, 1056 meters and 2112 meters.

While the GLS squares fit the simulation area, the regions of HLS are slightly bigger. The regions and cells start in the lower left corner of the coordinate system of the area. It is therefore possible that parts of some regions exceed the simulation area on the right and the top. The resulting effects have only little influence on the simulations and will therefore not be considered in the following.

Responsible cells The hash algorithm used to calculate the responsible cells is based on a modulo operation on hierarchy level and node id and can be formulated as follows:

$$\begin{aligned} Size(l) &= (3 \cdot 2^{l-1})^2 = \text{number of cells in a region of level } l \\ I(R(l,pos),p) &= \{c \in R(l,pos) \mid c \text{ is the cell with the pth smallest id}\} \\ F(n,l,pos) &= I(R(l,pos),((n+l) \bmod Size(l))) \end{aligned}$$

with $n \in N,\ l \in L,\ pos \in X \times Y,\ p \in \mathbb{N}$. For a definition of the sets, see Section 3.1.2.

Figure 4.1 shows the candidate cells for node 11 that have been computed with the above function. At a first glance, one could think that the deterministic pattern of the candidate cells leads to uncontrolled side effects, e.g. if nodes cannot reach certain cells due to obstacles. In our simulations, we only use quadratic simulation areas without obstacles and random node movement. As a consequence, nodes are uniformly distributed on average. The deterministic pattern therefore has no negative influence on our simulations. If a MANET using

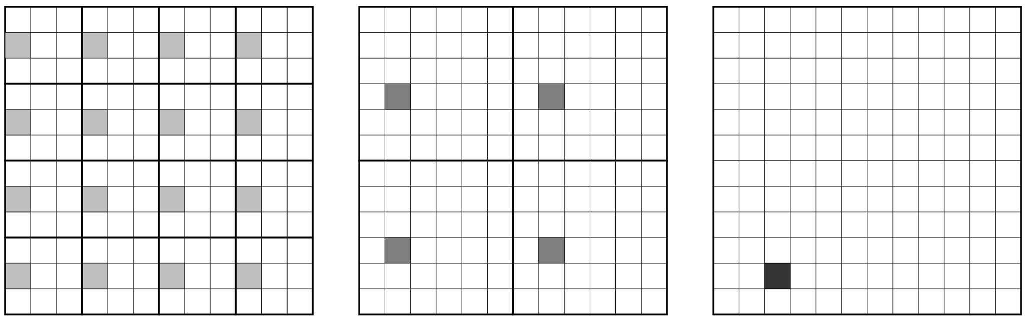


Figure 4.1: The candidate cells for node 11 as they were used in the simulations.

the Hierarchical Location Service is deployed in a real-world scenario, the hash function needs to be adapted to the environment.

Updates We have implemented the direct location scheme presented in Section 3.2. Position updates are triggered in the subsequent three cases:

- The responsible cell has changed, for example due to a region change.
- The node has covered a distance greater than the radio range since the last update.
- It is time for a periodic, time-triggered update.

Since we want to evaluate the basic idea of HLS (to find location servers in hierarchically grouped responsible cells), nodes remove location information older than 20 seconds from their location database. To compensate the dropping of information at active location servers, responsible cells must be updated periodically. We chose the update frequency to be 9 seconds for all simulations at all speeds to provide stable conditions. A negative but acceptable side effect are the high number of update packets and therefore the high basic MAC load especially for low node speeds[1]. Due to the timeout for position information, location servers which have left the responsible cell discard their information after a short period of time. As a matter of fact, most requests are answered by active location servers.

[1]A node moving with 10 m/s only needs to update its location server every 25 seconds.

To reduce traffic, these time-triggered updates are only sent to RCs on level two and three. On the one hand, unavailable location information on the first level does not affect the location service's performance much. On the other hand, this is partly compensated by the fact that nodes within the RC on level one may be in beacon range and therefore can serve as implicit location servers.

The update packets are routed with position-based routing (GPSR, see Section 2.2.1). For the selection of a location server, there are two possibilities. If the responsible cell cannot be reached e.g. because the network is partitioned or the RC does not contain any node, the node detecting this becomes location server. An example for this can be found in Figure 4.3(a) on page 42. If the update packet reaches a node which is in the correct cell, this node becomes location server.

With exception of the time-triggered updates, we tried to reduce update traffic as much as possible in our implementation of HLS. As a consequence, request packets produce more traffic because they statistically have to be forwarded more often to find a location server. However, since requests are sent less frequently than updates, the overall traffic is reduced. Following this principle, the TTL for update packets is set according to the formula

$$TTL = 10 \cdot l \ , \ l : \ level \ of \ RC$$

which is sufficient to reach the target cell.

Furthermore, update as well as handover packets never use perimeter mode in the current implementation. Early simulations showed that enabling perimeter mode for those packet types generates heavy load on the network in low-density scenarios. If the target RC is empty or cannot be reached, update and handover packets are forwarded on long perimeters until the TTL expires. GPSR tries to reach an unreachable destination while it would be sufficient to deliver the packet to a node close to the target cell.

Handovers As mentioned above, nodes can be the location server for position information of a responsible cell although they are not located within the boundaries of this RC. This may happen if a node has just left the RC or no node within the RC could be found as presented in Figure 4.3(a). Therefore, every node checks regularly if it is storing position information belonging to cells other than the one in which it is located at this point in time. In this case, the node sends the information in a handover packet to the target cell.

Due to this mechanism, the position information can always be found in or close to the target cell, even when nodes move with high speeds.

Requests Requesting the position of a node works as described in Section 3.4.

Upon receiving a request, a node processes it as follows:

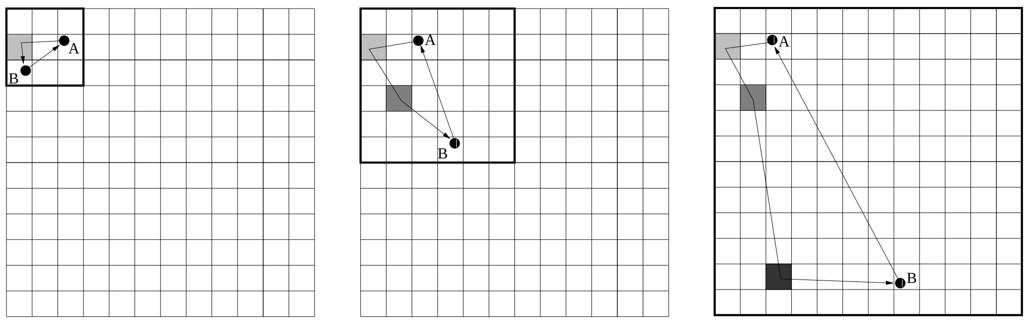
1. If the request target is the receiving node itself, a reply is generated and sent back to the requesting node.
2. If the node has location information about the target in its location database, the request is redirected to the target.
3. If the node is member of the target candidate cell but does not know the location of the target node, it sends a cellcast request (see Section 3.4). This will be answered by any node within radio range with location information about the requested node.
4. If the node detects that no location server can be found in the candidate cell, for example because a cellcast request timed out or the candidate cell could not be reached, it forwards the request to the candidate cell on the next level.

The level of the target candidate cell is taken into account when setting the TTL value for a request packet. It does not make sense to have a packet with a high TTL for a level-1 candidate cell because the packet should travel only a few hops to this cell. If it has to travel far, it is more likely that the cell does not contain any node and the packet can be forwarded to the next level. The TTL value is set using the formula

$$TTL = 30 \cdot l \;,\; l : \; level\ of\ RC$$

.

Cellcast request A cellcast request sent to a cell c must reach all nodes within c. We decided to achieve this by choosing an appropriate cell size (see page 35).



(a) level 1 (b) level 2 (c) level 3

Figure 4.2: Requests answered on different hierarchy levels depending on positions of request sender (A) and request target (B). The shaded squares are the RCs following Figure 4.1.

Therefore, a cellcast request in our implementation of HLS is a 1-hop broadcast message. Any node within radio range receiving this request which has location information about the target prepares an answer packet. To avoid collisions of answer packets, a simple timer based contention is used. A node which receives an answer packet for a cellcast request discards its own answer packet.

With this cellcast request mechanism it is quite likely that nodes which are not within the correct cell c receive the request and send an answer because c covers only about 15.8 percent of the area in which the cellcast request can be received. This does not disturb the functioning, in fact it increases the success probability.

Artificial home perimeter Early simulations with a basic implementation showed that it is difficult to find a location server if the destination candidate cell is empty especially in low-density scenarios. This is no problem on lower levels because the request is just forwarded to the candidate cell on the next level. If the candidate cell is on the highest level and no location server can be found, the request fails. We therefore considered different mechanisms to prevent failures at top-level candidate cells. Possible solutions might be:

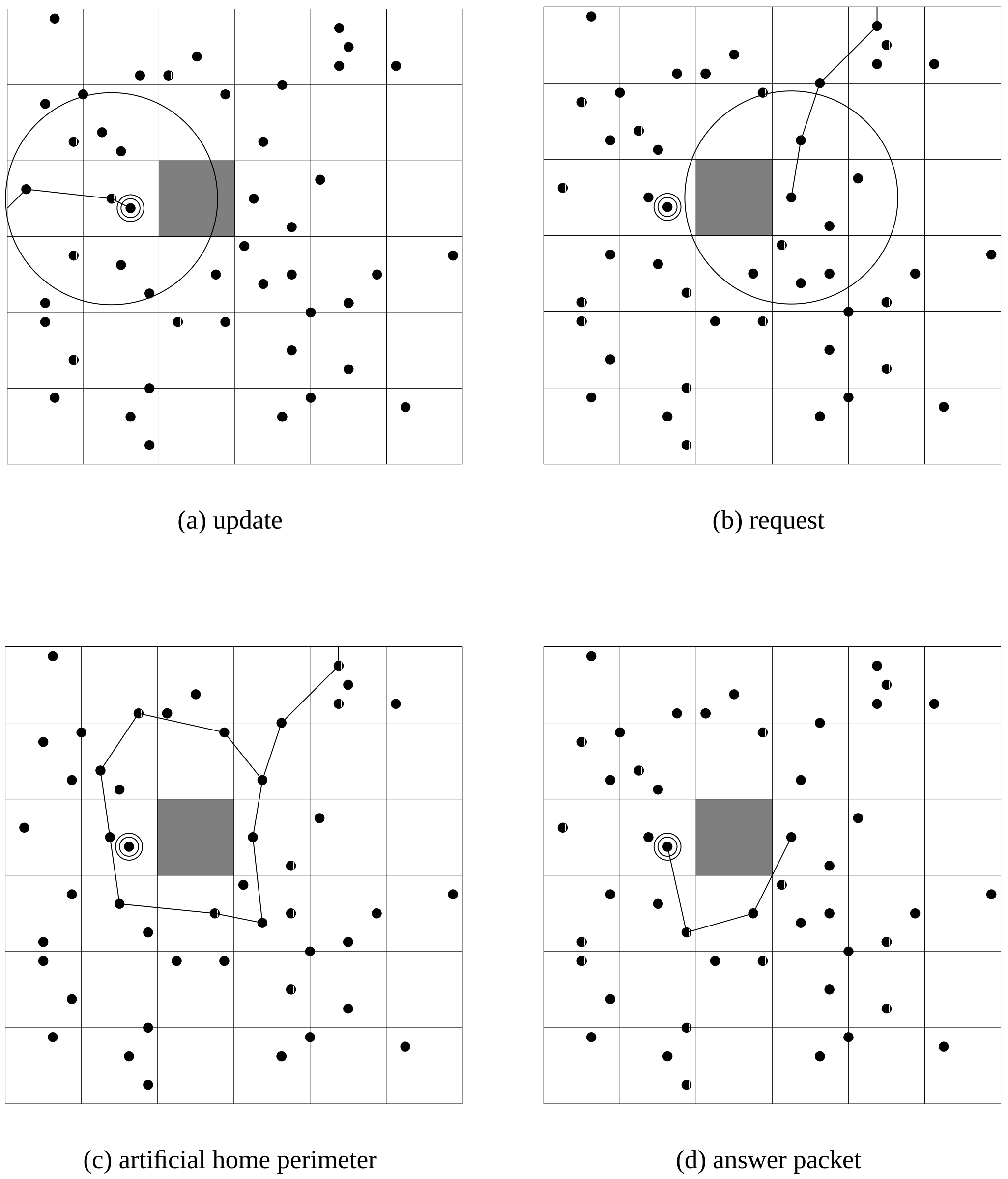
- Flooding / geocast : the requested area is extended to the neighbor cells and the request is flooded or geocasted to these cells.

- Home perimeter : one of the side effects of GPSR perimeter mode can be used as described in [RKY$^+$02] to let the request packet be forwarded on a perimeter around the target cell. One of the nodes on the perimeter should be location server and can therefore answer the request.

The home perimeter solution seems to be more suitable for the task because it generates less packets than the first alternative. However, there is one major drawback: the HLS implementation would be limited to the use with GPSR in perimeter mode as routing protocol. Thus, we decided to implement our own method to route around an empty cell in order to find a location server. We call this method *artificial home perimeter* in the remainder. One should keep in mind that this is only done for the top-level RC.

The basic idea of the artificial home perimeter request is sending the packets successively to each of the neighboring cells of the top-level RC. The request is transmitted as unicast packet. To allow nodes which do not directly participate in forwarding the packet to also receive the request, we run the network interface in promiscuous mode. The promiscuous mode disables address filtering and allows all nodes within radio range of a transmission to process the packet no matter if they are the next hop of it or not. Any node which receives the request and possesses location information produces an answer packet. As with the cellcast request, we use a timer based contention to minimize collisions.

The functioning of an artificial home perimeter is shown in Figure 4.3. Whenever an update packet cannot reach any node within the target cell, it is stored at the best available node (Figure 4.3(a), the position server is the node marked with two circles). If a request is sent to the respective cell, the location server is not in the correct cell and thus may be unreachable (Figure 4.3(b)). In this case, the request is routed counterclockwise to the neighboring cells of the RC (Figure 4.3(c)). The location server in this example is not directly involved in forwarding the request but nevertheless receives it because of the promiscuous mode. It sends an answer to the sender of the artificial home perimeter request (Figure 4.3(d)) which on its part can forward the request to the requested node.


(a) update

(b) request

(c) artificial home perimeter

(d) answer packet

Figure 4.3: The artificial home perimeter mechanism

4.1.2 Unimplemented Features

In the following all features which potentially improve the performance of HLS but have not yet been implemented for the evaluation are listed.

Cells As mentioned in Section 3.1.1, cells should be hexagonal to most optimally cover the area. Besides, the hash algorithm used to compute the responsible cells is very simple with no optimizations applied.

Updates

- To avoid overloading the network and to reduce collisions, the concept of time division multiplexing (TDM) could be used. The nodes are divided into a number of groups depending on their node id. Each group is assigned a time slot, a recurrent time interval. A node only sends updates in the time slot of its group.
- The application of the indirect location scheme presented in Section 3.2 may significantly reduce traffic and enable HLS to scale to large networks.
- To avoid intensive proactive update traffic and, nevertheless, to make use of the GPSR mechanism to recover from local maxima, the TTL value of update and handover packets could be manipulated. If one of those packets enters perimeter mode, the TTL of this packet is set to a low value depending on the scenario and the position of the forwarding node to avoid pathological cases. This TTL manipulation detains the packet from traveling many hops on a perimeter while it can recover from a local maximum. If the packet is forwarded more often than the manipulated TTL allows, the node where the TTL expires becomes location server. If the packet is switched to greedy mode again, the old TTL can be restored, decremented by the number of hops taken in perimeter mode.

Requests

- In the current implementation, a request is forwarded from candidate cell to candidate cell with ascending levels. It is possible that a node is closer to a candidate cell of a higher level than to that of level one or two. In this case, the request should be sent directly to the higher level candidate cell where the chances to find a location server for the requested node are much higher.

- Instead of dropping reply packets or request packets that have already met a location server, nodes shall retry to transmit those packets after a few seconds.
- The answers to cellcast requests and artificial home perimeter packets are currently scheduled on a random basis. It would be better to schedule them according to the timestamp of the information to be transmitted. Hereby, the principle: ”information of lowest age is the most accurate and should thus be transmitted first” enhances efficiency.
- The answer to a request does not necessarily have to be generated by the requested node. If the information is still up-to-date and the distance between a location server and a request sender is short, a location server could also generate the answer. Hence, traffic is reduced.

Reacting to node characteristics The performance of HLS could be improved by adapting its implementation to node speeds. Every node could register its current maximum speed. It is likely that other nodes also move with similar speeds, thus a lot of parameters (e.g. the maximum age of a cache entry) can be adapted with this information.

If the nodes transmit their actual speed together with the location information, movement prediction and parameter modulation are possible. This may greatly improve HLS’ performance (and that of any other location service).

4.1.3 HLS Packet Types and Sizes

To allow detailed bandwidth measurements, the size of each packet is traced in detail. Table 4.1 states the sizes of the basic data types for the packet header. These basic values are valid for both HLS and GLS.

The subsequent Table 4.2 states for each packet type the header sizes used in the Hierarchical Location Service.

DATA TYPE	SIZE IN BYTES	LOCSERVICE SPECIFIC
Node identifier	4	no
Location coordinate	3	no
Timestamp	2	no
TTL	1	no
Position (x and y coordinate)	6	no
Node information (ID, timestamp, position)	12	no
Grid	2	GLS
Order	1	GLS
Square (grid + order)	3	GLS
Cell	2	HLS
Level	1	HLS
Flags	1	HLS

Table 4.1: Basic field sizes used in bandwidth calculations

PACKET TYPE	HEADER FIELDS All packets contain node information about the sender, TTL and flags. Additional fields are listed below.	SIZE IN BYTES
Update	target cell	16
Request	target cell, target level, target node	21
Reply	target node information	26
Handover	target cell, (number of infos to handover) · (node information)	$16 + n \cdot 12$
Cellcast Request	target cell, target node	20
Cellcast Reply	cellcast source node information, target node information	38
Artificial Home Perimeter Request	base cell, actual target cell, ID of requested node	22

Table 4.2: HLS header sizes by packet type

4.2 GLS and ns-2

The concepts of the Grid Location Service have been introduced in Section 2.3.5. The implementation used for the following evaluations is a version developed at the University of Mannheim, Lehrstuhl für Praktische Informatik IV [PI4], and adapted to work together with GPSR. This can influence the simulation results as stated in [KFHM02]:

> Since the original authors used a greedy routing scheme based on two-hop neighborhood and grid-based forwarding, specific to this location service, while we pair it with the standard GPSR scheme, it is very likely that our scheme does not represent the maximum performance of GLS.

The implementation uses the following features and optimizations:

- Smallest grid: 250 m
- TTL 64 for each packet
- Update distance: 250 m
- Always update location information in packets passing by
- Send updates if
  1. node is crossing the border of a grid
  2. node has covered the update distance since the last update
  3. a time interval equal to $2\frac{u}{s}$ has passed; u : update distance, s : current node speed
- Aggressive caching. All location information which is found in a packet to be forwarded is stored in the location database of the forwarding node.

4.2.1 GLS Packet Types and Sizes

Table 4.3 contains the sizes of the GLS specific headers grouped by packet types. These values are based on the basic sizes defined in Table 4.1.

PACKET TYPE	HEADER FIELDS All packets contain node information about sender and target and a TTL field. Additional fields are listed below.	SIZE IN BYTES
Update	square, timestamp	29
Request	requested node ID	29
Reply	-	25

Table 4.3: GLS header sizes by packet type

4.3 IEEE 802.11

IEEE 802.11 [IEE99] is an official standard for wireless LANs, belonging to the group of 802.x protocols[2]. It mainly specifies the physical and the medium access control layer. The interface to upper layers is the same as that of other members in the 802.x protocol group.

In our simulations, we used the ns-2 version 2.1b9 implementation of IEEE 802.11 with an additional bug fix which was ported to ns-2 version 2.1b8a (for details see [Käs03]).

4.4 Null MAC

The *Null MAC* was developed in the course of the fleetnet project [FN]. It simulates an ideal implementation of the layers 1 and 2 of the ISO/OSI-model. A packet transmitted via the Null MAC reaches the target node if this node is in radio range. The Null MAC does not produce any collisions because it uses peer-to-peer connections instead of a shared medium. Nevertheless, the bandwidth is limited. A more detailed description can be found in [Käs03].

[2]Other members are Token Ring or Ethernet.

Chapter 5

Simulation and Evaluation

This chapter contains a description of the simulation setup used to evaluate GLS and HLS. Based on the results obtained through these simulations, the two algorithms are compared. Finally, a detailed analysis of the HLS simulations is presented.

5.1 Simulation Parameters

In RFC 2501 [CM99], a discussion about ”routing protocol performance issues and evaluation considerations” for MANETs can be found. It specifies the parameters presented in Table 5.1 to be varied for the *network context*, in the following called *scenario*.

PARAMETER	EXPLICATION
network size	the total number of nodes in the network
connectivity	the number of neighbors = density
topological rate of change	change speed of topology
link capacity	available bandwidth
fraction of unidirectional links	-
traffic patterns	non-uniform traffic, burst traffic, etc.
mobility	movement / mobility model
fraction of sleeping nodes	-

Table 5.1: Network context parameters as presented in RFC 2501

In our simulations, we focused on the influence of network size, connectivity and topological rate of change on the location services' performance. The nodes in all simulations move according to the *modified random direction mobility model* [Käs03][1]. In this model, a node choses direction and speed and moves in this direction for a random time period. It then halts for a predefined period. After the pause, it selects other direction, speed and time values and so on until the simulation is finished. Whenever the node hits a boundary of the simulation area, it is reflected at the boundary and continues to move until the time period expires.

For all simulations, we use the following terminology. A *query* is a location lookup for a node T launched by a node S which is handed over to the location service interface of S. It can be answered either locally from the location cache of S, we speak of a *cache lookup* or a *local lookup* in this case, or by sending a *request*. A request packet is forwarded via a location server to the target node T. This node generates a reply packet which is sent to S. Therefore, a request is always triggered by a query.

5.1.1 Basic Simulations

For the basic simulations, we use scenarios of size 2×2 kilometers. These scenarios are populated with 100, 200, 300 or 400 nodes which corresponds to a node density of 25, 50, 75 or 100 nodes per square kilometer. The nodes select their movement speed out of the intervals $[0, 10]$, $[0, 30]$ or $[0, 50]$ m/s and do not pause between changing directions. This results in an average node speed of approximately 5, 15 and 25 m/s. Table 5.2 contains the parameters used for the basic simulations.

As a consequence, 12 different scenarios were simulated. For each scenario an average of 10 runs was taken. For each run, 1200 test-queries were launched, resulting in 12000 queries for each scenario. The first query was launched 15 seconds after the beginning of a simulation, the last one 4 seconds before the end. For all basic simulations, we used IEEE 802.11 with a bandwidth of 2 MBit/s as protocol for the wireless communication and GPSR as routing protocol. The

[1]This mobility model is a variant of the *random waypoint model* [JM96]. Similar mobility models have been proposed in [BCSW98, BZ03].

number of nodes	100, 200, 300, 400
node density per square kilometer	25, 50, 75, 100
max speed (m/s)	10, 30, 50
simulation time (seconds)	300
requests per node	12, 6, 4, 3
requests per simulation run	1200
number of runs	10

Table 5.2: Simulation parameters for the basic simulations

only traffic within the network consists of packets sent by the location service and those produced by GPSR. In order to be able to compare the two location services, a request packet always has to reach the target node where a reply packet is generated.

5.1.2 Large Scale Simulations

The Hierarchical Location Service is designed to scale to very large networks with a big geographical extension and a lot of participating nodes. Therefore, we extended the size of the network. Due to limited computational capacities, we used the Null MAC instead of MAC 802.11 as a model for the lower network layers and ran only 5 runs with 150 seconds instead of 10 runs with 300 seconds as in the basic simulations.

For the large scale simulations, a fixed node density of 75 nodes per square kilometer was used to guarantee sufficient greedy connectivity, i.e. nodes are dense enough so that the majority of all packets can reach their destination with pure greedy forwarding. The geographical extension of the network is 2×2, 3×3, 4×4 and 5×5 kilometers, resulting in 300, 675, 1200 and 1875 participating nodes. The nodes move with the same speed as in the basic simulations. In order to prevent the network from being overloaded, each node sends two requests in the course of a simulation run. This corresponds to the network load produced in the 300 node scenario of the basic simulation. The parameters of the large scale simulations are presented in Table 5.3.

number of nodes	300, 675, 1200, 1875
node density per square kilometer	75
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

Table 5.3: Simulation parameters for the large scale simulations

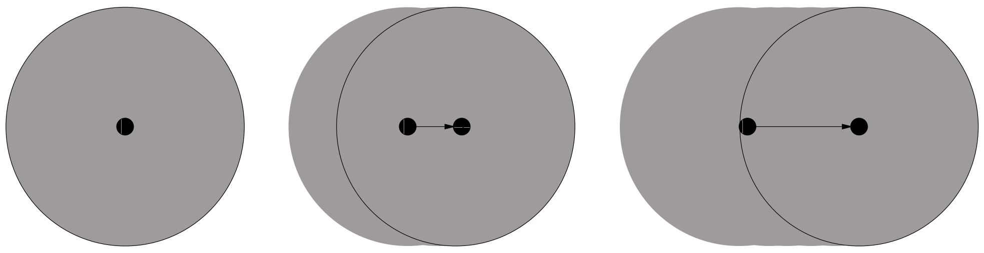
PROPERTY	GLS	HLS
distributed operation	x	x
loop-freedom	TTL	TTL
demand-based operation	-	-
proactive-operation	x	x
security	-	-
sleep period operation	-	-
unidirectional link support	-	-

Table 5.4: Comparison of GLS and HLS based on the criteria given in RFC 2501.

5.2 Evaluation Parameters

5.2.1 Qualitative Properties

RFC 2501 also specifies ”desirable qualitative properties for MANET routing protocols”: distributed operation, loop-freedom, demand-based operation, proactive-operation, security, sleep period operation and unidirectional link support. With a location service being an indispensable service for geographic routing, these criteria can also be applied to them. The result of the comparison between GLS and HLS is shown in Table 5.4. It is obvious that both location services own the same qualitative properties according to RFC 2501, we therefore do not go into detail here.



(a) immobile node (b) node covers 100 m (c) node covers 250 m

Figure 5.1: The movement effect for nodes with a radio range of 250 meters.

5.2.2 Quantitative Properties

Besides the qualitative properties in Section 5.2.1, RFC 2501 names quantitative properties for the routing protocol evaluation: end-to-end data throughput and delay, route acquisition time, percentage out-of-order delivery and efficiency.

In order to get reliable values for these properties, simulations with data traffic should be performed. We nevertheless tried to adhere to the suggested parameters as much as possible. We therefore decided to evaluate query success rate and bandwidth consumption in the comparison between GLS and HLS which are efficiency parameters. In the detailed analysis of HLS, route acquisition time is evaluated in combination with some HLS specific parameters. These specific parameters will be explained when necessary.

5.3 Movement Effect

The *movement effect* occurs with increasing speed if nodes store the position information obtained from beacons and updates for a certain amount of time, e.g. 10 seconds. The faster these nodes move, the more area is covered by their radios in these 10 seconds, and the more information can be collected. This is shown in Figure 5.1.

Assuming that the other nodes have not covered a distance greater than 250 meters, these nodes are still within radio range of the location from which they have sent the beacon in the presented cases. Whenever the node in Figure 5.1(c) looks in its cache for a node's position, the probability of success is nearly twice as high as that of the node in Figure 5.1(a).

5.4 Caching Effect

The main difference between the GLS and HLS implementation lies in the level of caching which has been used. The Grid Location Service implementation is an already optimized version using aggressive caching (see Section 4.2). The effect of this is presented in Figure 5.2. There, the percentage of queries which could be answered without sending a request packet is shown for both location services. HLS updates are only forwarded in GPSR greedy mode (see Section 4.1.1), therefore only one graph is shown. GLS update packets may use perimeter mode. For this reason, we show two curves.

A node using the HLS implementation has only knowledge of the positions of nodes from which beacons are received plus the ones for which it is the location server. This can be seen by comparing Figures 5.2(a) - 5.2(d) to Figure 5.2(e). About 5 percent of all queries have nodes within radio range as target, i.e. the source - destination distance is lower or equal to 250 meters. The number of queries which could be answered from the location database of a node is just slightly higher than those 5 percent at low speed for HLS. The increasing number of local lookups at higher speeds, approximately 4 percent independent of the number of nodes, are a consequence of the above described movement effect.

If using the GLS implementation as location service, a node caches all available information. Apart from the information about nodes for which it is location server, it stores source and target information from all packets passing by. Investigating the fraction of cache lookups of GLS presented in Figure 5.2, it can be concluded that each node has knowledge about the positions of 20 - 50 percent of all nodes due to this caching. The higher cache lookup rate with perimeter routing results from the perimeters a packet has to take. Especially in sparse scenarios an update packet has to be forwarded over a high percentage of available nodes to reach its target. Therefore, nodes have more location information in their caches as we observe in Figure 5.2(a). The effect decreases with higher node density because perimeters tend to be shorter and updates only visit a few additional nodes.

The aggressive caching used in the GLS implementation improves the results in two ways:

1. The fraction of local lookups is much higher. Thus, less requests have to be sent which decreases network load and packet collisions. Furthermore, a local lookup always produces a result whereas a request can get lost, be unable to find a location server and so on.

2. Nodes can better update the location information about the target in packets forwarded by them which increases the delivery ratio.

On the other hand, the accuracy of cached location information decreases over time which in turn may decrease the success rate of location queries. As stated in [MWH01]:

> The face-2 algorithm and the perimeter routing of GPSR are currently the most advanced recovery strategies. The only drawback of the current greedy approaches is that the position of the destination needs to be known with an accuracy of a one-hop transmission range, otherwise the packets cannot be delivered.

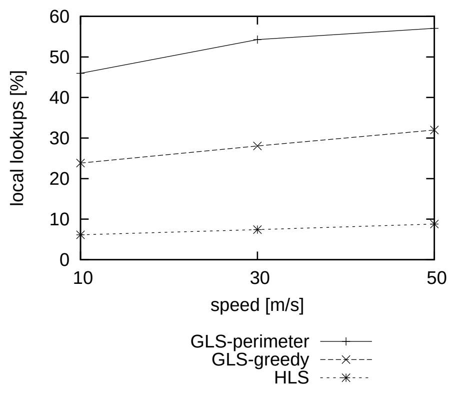
Therefore, cache lookups with an accuracy less than the radio range (250 meters) are classified as lookup failures. Especially at higher node speeds, the number of cache lookup failures increases. When regarding the simulation results of GLS, these effects which work in opposite directions should always be kept in mind.

5.5 Greedy Simulations

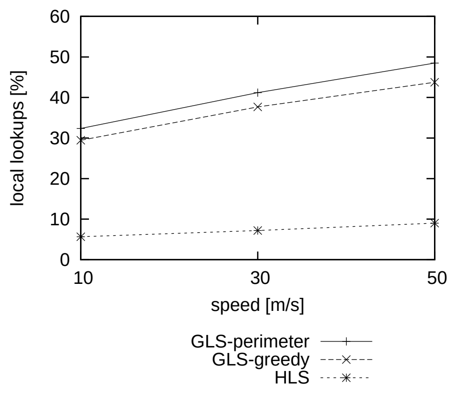
In all simulations of this section, GPSR with greedy forwarding (no perimeter mode) was used. The scenarios which have been simulated are the basic scenarios presented in Section 5.1.

5.5.1 Success Rate

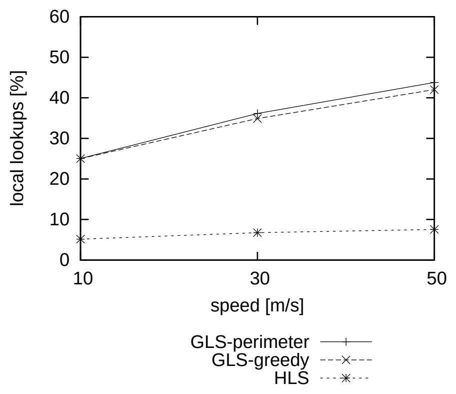
The *success rate*, i.e. the percentage of queries which have been successfully answered by the location service, is shown in Figure 5.3. The labels have the



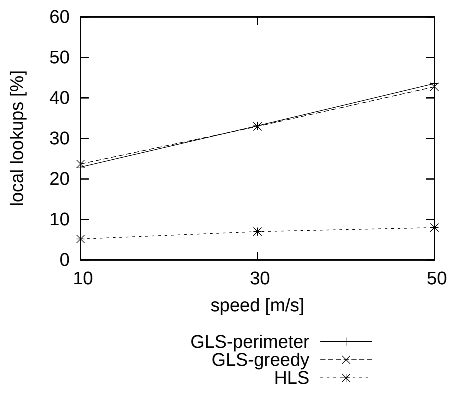
(a) 100 Nodes



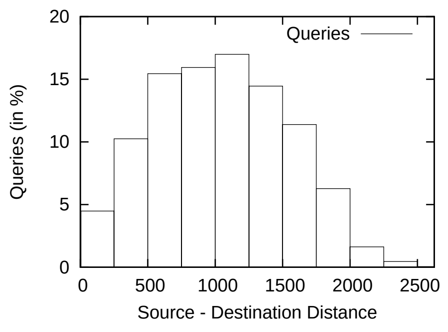
(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes



(e) Query distance

Figure 5.2: Caching effect and request distance

following meaning:

GLS-max The percentage of queries for which GLS produced a result, i.e. all queries minus the request failures. This value is only given as a benchmark for the maximum possible success rate of our GLS implementation and also includes the cache lookup failures.

GLS The percentage of queries for which GLS produced a correct result, i.e. the successful cache lookups and the requests for which a reply was received.

GLS-req The percentage of request packets send by GLS for which a reply was received.

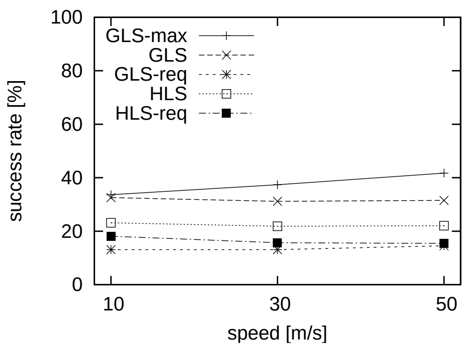
HLS The percentage of queries for which HLS produced a correct result, i.e. the successful cache lookups and the requests for which a reply was received.

HLS-req The percentage of request packets sent by HLS for which a reply was received.

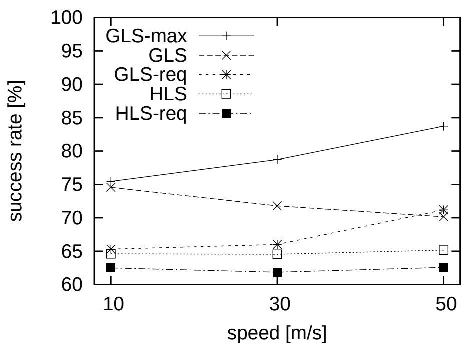
The values of interest that should be compared to get an impression of the location services' performance are HLS versus GLS and HLS-req versus GLS-req. The first pair shows the *overall success rate*, i.e. the percentage of queries which could be answered correctly by the location service. The second pair visualizes the *request success rate*, i.e. the capability of each location service to run a full location request including other nodes: find a location server, forward the request to the target node and return the reply packet to the sender.

In the scenario with 100 nodes (Figure 5.3(a)), the request success rates of both location services lie between 10 and 20 percent. This is caused by the lack of greedy connectivity in this sparse scenario. Only with heavy use of caching, the location services achieve overall success rates of around 30 percent. The success rates ameliorate in the 200 node scenarios presented in Figure 5.3(b). GLS performs better here both with respect to overall success rate and request success rate. The node density is still too low for HLS to correctly deliver updates and to identify location servers with pure greedy routing.

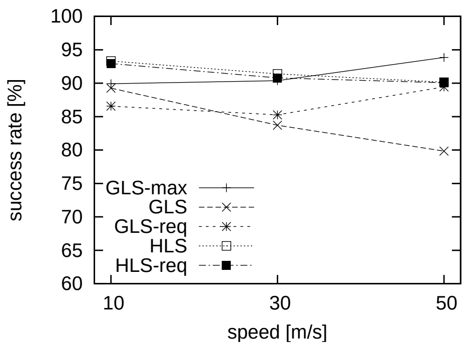
In the 300 node scenario of Figure 5.3(c), nodes are dense enough to allow a correct functioning of the HLS mechanism. Both for overall success rate and



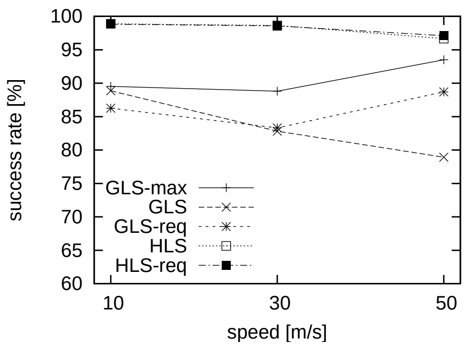
(a) 100 Nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.3: Success rates of HLS and GLS with greedy forwarding.

request success rate, HLS achieves better results than GLS. This further improves in the 400 node scenario (Figure 5.3(d)). We observe that the Grid Location Service does not profit from the higher node density in our simulations. All curves for GLS have only changed slightly compared to the 300 node scenario.

Regarding Figure 5.3, one can see that HLS needs a certain minimum node density to work and that success rates improves with growing node density. In our simulation with greedy forwarding, this minimum is 75 nodes per square kilometer which corresponds to the 300 node scenario. There are two reasons for this. On the one hand, the higher the node density, the higher the greedy connectivity. On the other hand, the selection of location servers is also directly influenced. The higher the node density, the lower the probability that a target cell for an update or request is empty.

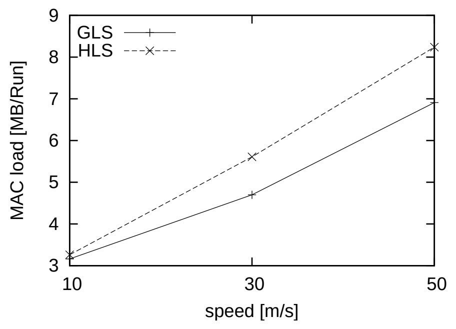
5.5.2 Bandwidth

Another significant value is the bandwidth consumed by each location service to achieve the results mentioned above. The different packet sizes used for the bandwidth measurement are the ones specified in Section 4.1.3 for HLS and in Section 4.2.1 for GLS. The results of these measurements are presented in Figure 5.4.

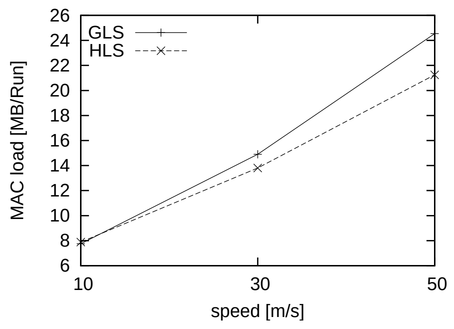
As expected, HLS consumes less bandwidth than GLS and the consumption grows with a lower gradient as speed increases. The only exception is the 100 node scenario. There are multiple effects which create this behavior:

- A lot of handover packets are necessary due to low node density.
- HLS sends more requests than GLS because of the low local lookup rate.
- Each node produces a time-triggered update every 9 seconds for the RC on the top-level without considering the current node speed. In the GLS implementation, the update interval depends among others on the speed as presented on page 46. It therefore produces less update packets at low speeds.

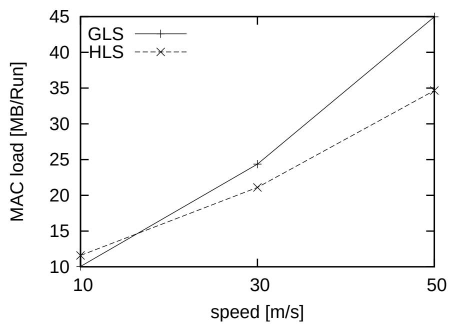
In order to get a better feeling of the influence of updates on bandwidth consumption, we analyzed the absolute number of update packets sent and forwarded



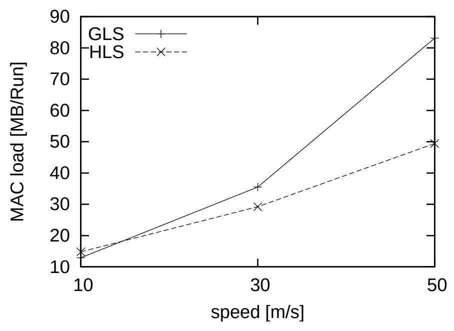
(a) 100 Nodes



(b) 200 Nodes

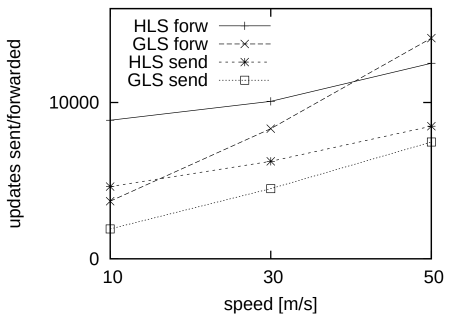


(c) 300 Nodes

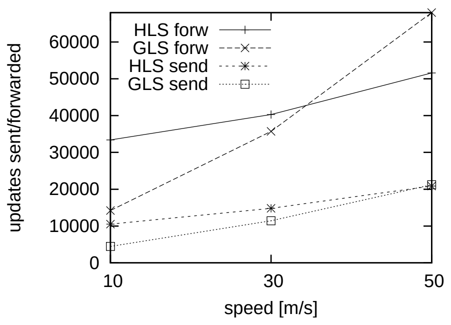


(d) 400 Nodes

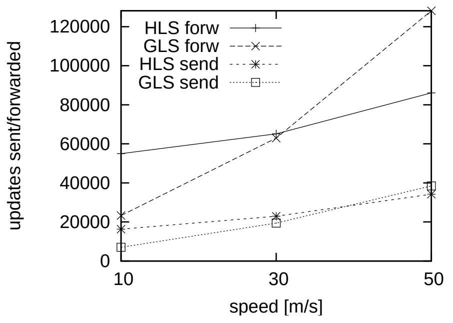
Figure 5.4: MAC load of GLS and HLS in the greedy simulations



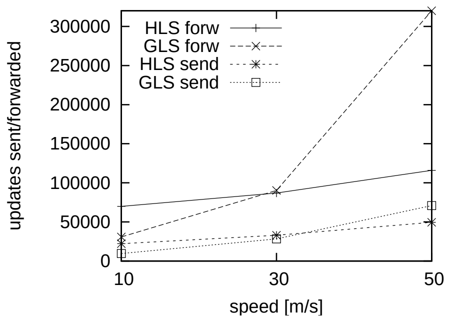
(a) 100 nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.5: Number of updates sent or forwarded by HLS and GLS in the greedy simulations

by each of the location services. The results are presented in Figure 5.5. At low speeds, HLS sends, and therefore also forwards, more updates than GLS for all node densities. This is mainly due to the time-triggered updates with the 9 second interval which produce a high basic load. As speed increases, the number of updates send by HLS grows slower than that of GLS.

At all node densities, the absolute number of forwards of the Grid Location Service increases faster than those of HLS with growing speed. Furthermore, the average number of hops per update packet, which can be calculated by dividing the number of forwards through the number of sends, increases for GLS with rising speed while it stays nearly the same for HLS. A possible explanation for this effect could be the update algorithm used by GLS. Forwarding decisions for update packets are based on location information about other nodes. The precision of this information sinks with growing speed. This may result in detours and therefore in an increasing number of forwards.

With the Hierarchical Location Service, the average number of hops per update packet does not depend on node speed. This results from the purely position-based method of identifying location servers. HLS delivers its updates to cells with fixed positions. The target position for an update packet is therefore always known with absolute precision independent of movement speed or positions of other nodes.

5.6 Perimeter Simulations

Motivated by the low success rates of both location services in sparse scenarios, we ran simulations with the same scenarios as in Section 5.1 using GPSR with activated perimeter mode.

5.6.1 Success Rate

The success rates achieved by the location services in these simulations are presented in Figure 5.6. We use the same definitions as in Section 5.5.1. As expected, both HLS and GLS benefit from the perimeter mode.

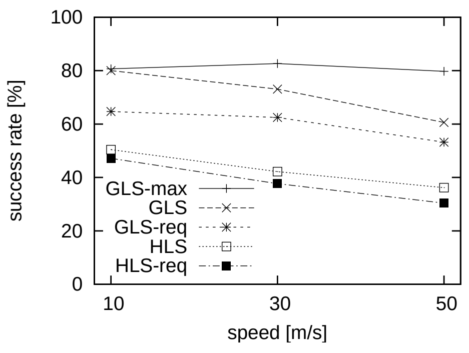
Especially GLS achieves reasonable overall success rates for all scenarios while HLS' success rate is low for the sparse scenarios (Figure 5.6(a) and 5.6(b)). Partly, those results are a consequence of the aggressive caching used in the GLS implementation in combination with perimeter forwarding.

A request does not have to reach an active location server for the target node to be processed successfully. It is sufficient if it reaches one of the nodes that has stored location information in its cache. From Section 5.4 we know that each node using the GLS implementation has knowledge about the positions of 20 - 50 percent of all other nodes. It can therefore be expected that a request packet which tries to find a location server meets a node having cached the target location. According to the DREAM forwarding principle (see Section 2.3.1), the position information does not need to be very precise. Nodes forwarding the request can update the location of the target when the request approaches the target node's position. Although requests retrieve cached location information in the greedy simulations, a lot of request packets have to be dropped here because of missing greedy connectivity. Perimeter forwarding improves the delivery ratio of these packets. Our HLS implementation avoids caching as much as possible and does therefore not profit from this effect as much as GLS.

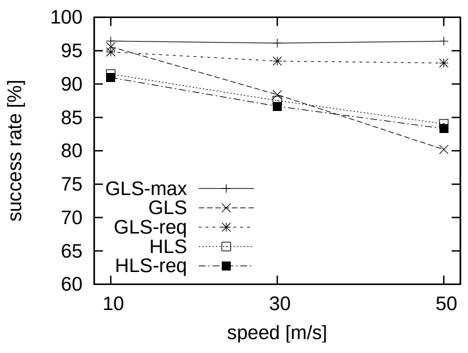
The request success rate for GLS in the 200 node scenario (Figure 5.6(b)) is always around 94 percent while the overall success rate decreases with growing node speed, mainly due to cache lookup errors. The Hierarchical Location Service performs well in this scenario, both overall and request success rate are between 83 percent and 91 percent through all speeds. Nevertheless, HLS shows still problems finding location servers.

This improves in the scenario with 300 nodes presented in Figure 5.6(c). Nodes are now dense enough to enable the HLS concept to work. HLS outperforms GLS both with respect to overall and request success rate.

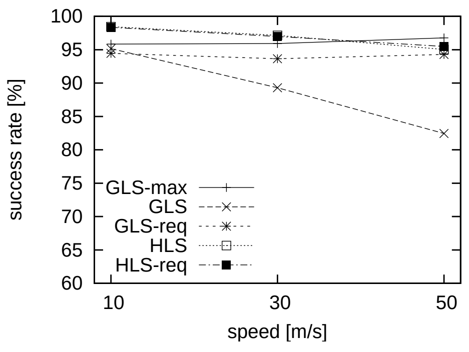
It is interesting to compare the GLS curves for the 200, 300 and 400 node scenarios. All curves are very similar in shape and magnitude for these scenarios. Furthermore, all curves correspond to those in the greedy simulations. The only exception is the 400 node, 50 m/s scenario. Here, we encounter the *IFQ drop*



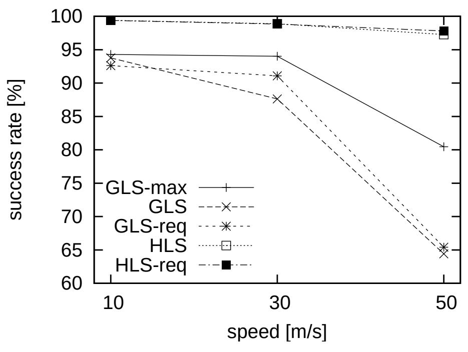
(a) 100 Nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.6: Success rates of HLS and GLS with perimeter forwarding.

effect already described in [KHFM02a][2]. GLS produces a lot of update packets resulting in high network load and, finally, in network congestion. Packets have to be dropped by the interface queue between the link layer and the MAC layer due to queue overflow.

In contrast to the greedy simulations, the minimum node density required for HLS to work well shrinks when using perimeter mode. A node density of 50 nodes per square kilometer, this corresponds to the 200 node scenario, seems to be sufficient.

[2]The IFQ drop effect already occurs in the 300 node simulations of this paper; this is due to the fact that radios with lower bandwidth and another mobility model was used. For details see [KHFM02a, Käs03].

5.6.2 Bandwidth

The bandwidth measurement as presented in Figure 5.7 is not completely fair for sparse scenarios because the GLS implementation allows packets of all types to use perimeter mode. HLS does not profit much from update packets traveling on long perimeters around a void. In fact, this can disturb the functioning of HLS because it is very likely that the target cell for an update is empty. Thus, HLS update and handover packets are not allowed to use perimeter mode. While this can lead to a lower success rate, it also avoids heavy MAC load. This explains the much higher MAC load of GLS in the scenario shown in Figure 5.7(a). In the other scenarios, greedy connectivity is higher and perimeters are in general not very long. The influence of the different treatment of updates therefore decreases with increasing node density. The decreasing influence can be seen by comparing the updates sent and forwarded in the greedy simulations (see Figure 5.5) to those of the perimeter simulations (Figure 5.8). The curves in Figure 5.8(a) are dominated by the GLS update forwards. The domination decreases with growing node density. In the 300 node scenario the graphs for the greedy and perimeter simulations are very similar (Figure 5.5(c) and Figure 5.8(c)).

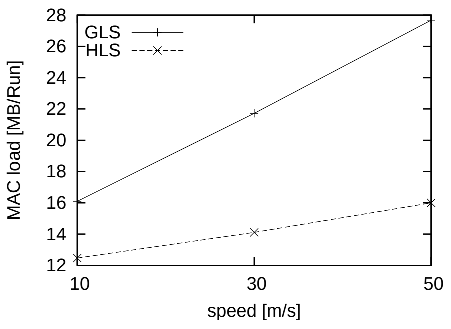
In Figure 5.7, we observe the same curve shapes as in the bandwidth measurements of the greedy simulations. Consequently, the Hierarchical Location Service in general consumes less bandwidth and scales better than GLS with increasing node speed.

5.7 Null MAC Simulations

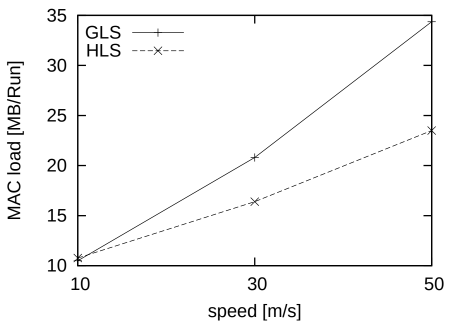
In this section, we present the results of the large scale scenarios. All of these were simulated using the Null MAC and the activated perimeter mode.

5.7.1 Reference Simulations

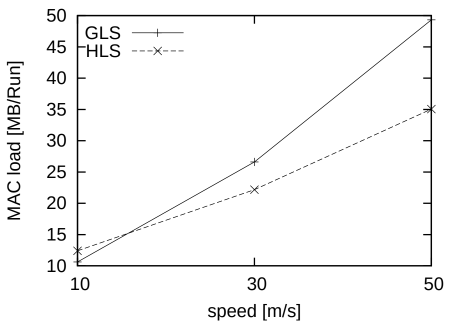
To become an impression of the influence of the different MAC layers, we simulated the scenarios of Section 5.6 using the Null MAC. The success rates of both location services are presented in Figure 5.9.



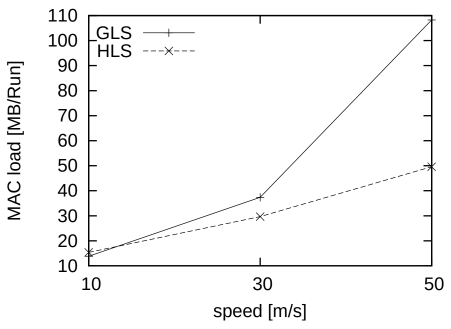
(a) 100 Nodes



(b) 200 Nodes

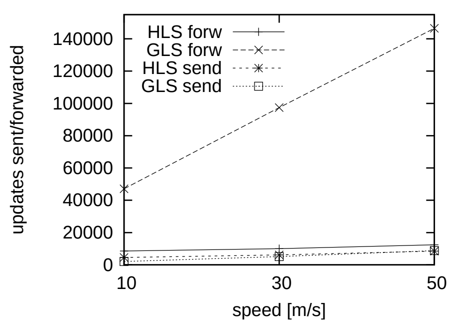


(c) 300 Nodes

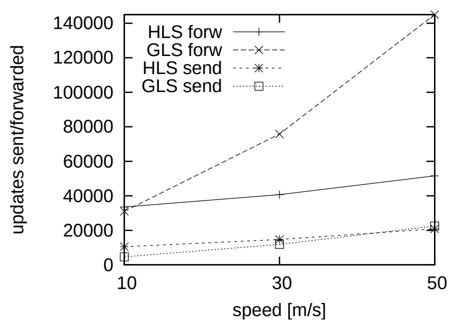


(d) 400 Nodes

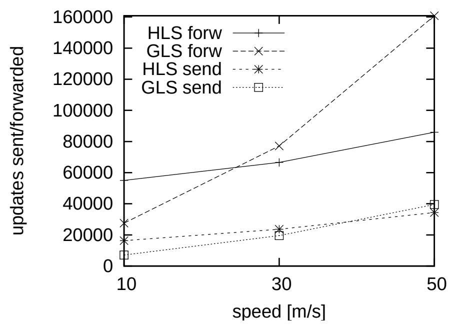
Figure 5.7: MAC load of GLS and HLS in the perimeter simulations



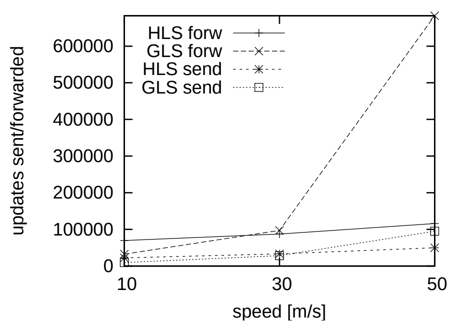
(a) 100 nodes



(b) 200 Nodes

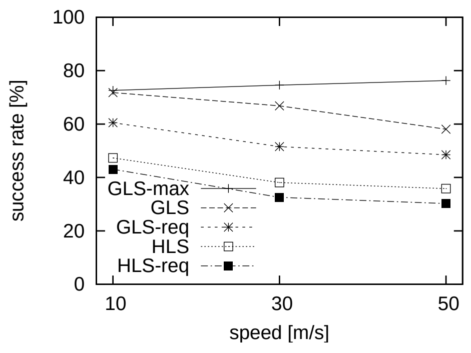


(c) 300 Nodes

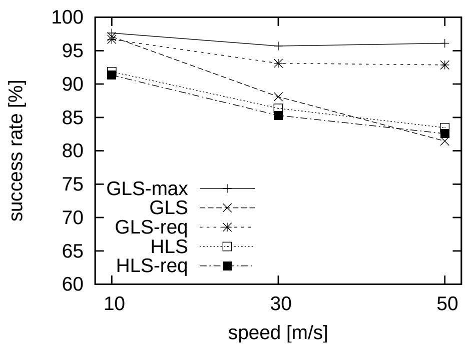


(d) 400 Nodes

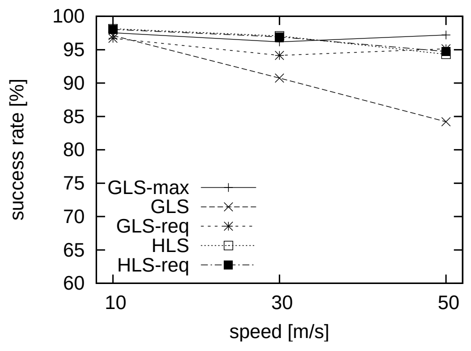
Figure 5.8: Number of updates sent or forwarded by HLS and GLS in the perimeter simulations



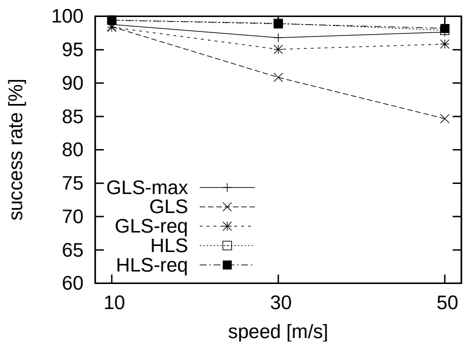
(a) 100 Nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.9: Success rates of HLS and GLS with perimeter forwarding and the Null MAC.

The results of HLS are slightly better than those of the simulations with IEEE 802.11 as MAC layer (see Figure 5.6). Here, the Null MAC mainly eliminates the errors produced by collisions. GLS profits much more from the usage of the Null MAC. In the 300 and 400 node scenarios, it suffered from collisions and overload. These effects do not occur here, hence GLS shows enhanced results.

5.7.2 Large Scale Simulations

The simulations presented in the following are those run according to the specifications in Section 5.1.2. We used the same basic cell / square sizes for HLS and

GLS as with the previous simulations. Due to the larger simulation area, more hierarchy levels are necessary (see Table 5.5). HLS uses four hierarchy levels in the 3×3 and 4×4 kilometer scenarios (size of level-4 region = $24 \cdot 176$ meters = 4224 meters), in the 5×5 kilometer scenarios, five levels (size of a level-5 region = $48 \cdot 176$ meters = 8448 meters) are used. The Grid Location Service uses a maximum order of four for the 3×3 and 4×4 kilometer scenarios.

Figure 5.10 shows the success rates achieved by both location services. Figure 5.10(d) only contains the success rates for the Hierarchical Location Service. The GLS simulations needed too much memory in this scenario and could therefore not be simulated. In all of the simulations, HLS' success rate stays almost constant for all area sizes, it varies only slightly. In contrast to this, the success rate of the Grid Location Service drops with increasing simulation area size. We can only speculate about the reasons for this descent and cannot be sure whether it is an effect of the implementation or the algorithm.

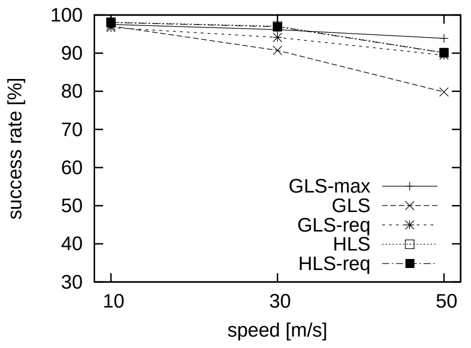
A possible explanation might be the routing scheme for updates and requests. The update and request packets rely on other nodes' location servers such that they are forwarded to the correct node. Inconsistencies in location information may lead to detours or even routing loops (see [KHFM02a]). Following 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 [LJD+00]. These siblings 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.

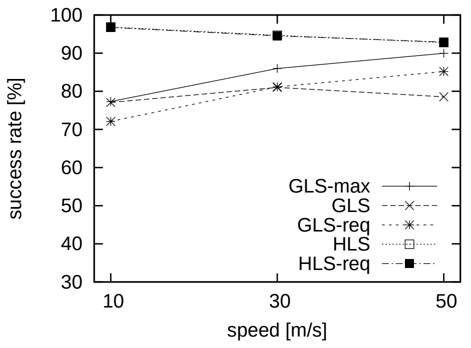
Nevertheless, there are several effects of wireless communication which degrade HLS' performance and that of any other location service with growing speed and network size. These effects like lost neighbors or packet collisions decrease the probability for a packet to reach the next hop and sum up with the number of hops that a packet has to take (this will be discussed in more detail in Section 5.8.2).

Area size	GLS, size and order of biggest square	HLS, size and level of biggest region
basic square/cell size	250 meters	176 meters
2×2 kilometers	1000×1000, order 3	2112×2112, level 3
3×3 kilometers	2000×2000, order 4	4224×4224, level 4
4×4 kilometers	2000×2000, order 4	4224×4224, level 4
5×5 kilometers	not simulated	8448×8448, level 5

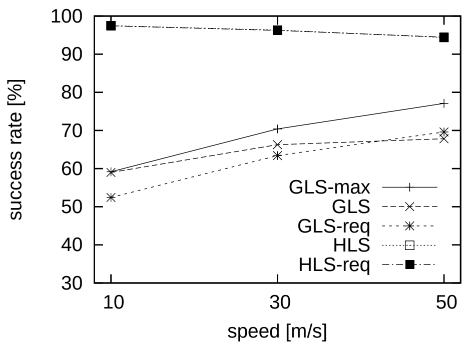
Table 5.5: Area partitioning in the large scale simulations for GLS and HLS



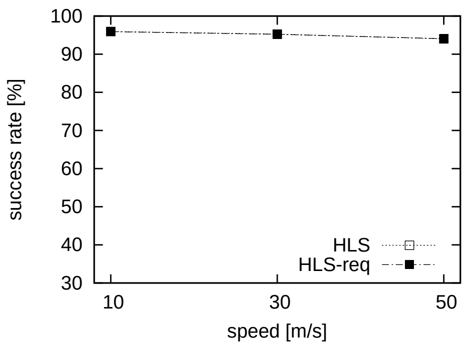
(a) 2×2 kilometers, 300 Nodes



(b) 3×3 kilometers, 675 Nodes



(c) 4×4 kilometers, 1200 Nodes



(d) 5×5 kilometers, 1875 Nodes

Figure 5.10: Success rates of HLS and GLS in the large scale simulations

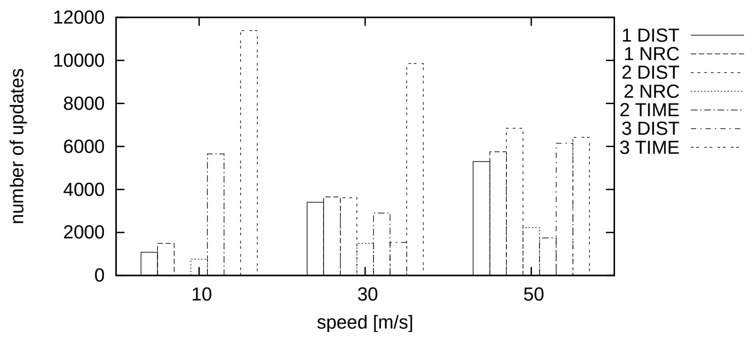


Figure 5.11: Update reasons for the 400 node scenario

5.8 Analysis of HLS

This section contains a detailed analysis of HLS' performance. Whenever the sending of updates is evaluated, we only show the graphs for the greedy simulations because update packets are not forwarded in perimeter mode (see Section 4.1.1).

5.8.1 Updates

Figure 5.11 shows the reasons why HLS sends updates to its location servers in the 400 node scenario[3]. The numbers in front of the marks of the bars stand for the different levels in the hierarchy. The abbreviations stand for the subsequent update reasons:

DIST The update was sent because the node has covered a distance greater than 250 meters and cannot be reached at its last known position.

NRC The node has changed the region. As a result, it has to send the update to a new responsible cell.

TIME The update was sent because the maximum time between two updates has expired.

[3]The graphs for the scenarios with other node densities look similar except for the absolute numbers of updates.

The NRC-updates for level three are not shown because they only occur once per node and simulation run. The high basic update load we encountered in our simulations is a result of the timeout we have implemented for position information and explained in Figure 5.11: by far most updates are time-triggered in the 10 m/s scenario. Other update reasons contribute much less to the traffic here. The number of time-triggered updates diminishes with higher node speeds because more distance-triggered updates are sent. Whenever a node sends an update, the timer for the time-triggered updates is reset.

It is interesting to observe the growth rate of the distance-triggered updates in contrast to the updates which have to be sent due to a region change. While the two values for the level-1 RCs grow with the same factor, the gap between the two values opens for higher levels. That is a result of the direct location scheme. The indirect location scheme makes that kind of updates unnecessary.

5.8.2 Update Delivery Ratio

HLS transmits its location updates to cells and request the position information from nodes in these cells. An information can only be found in a cell if it was correctly delivered beforehand. Therefore, we examined the *update delivery ratio*, i.e. the percentage of updates which have arrived in the correct cell[4]. The result is shown in Figure 5.12. The bars marked with ”fr” show the ”false receive” updates (i.e. the updates that have been received by a node which is not in the correct cell, e.g. when the routing finds no better node). The bars marked with ”r” show the correctly received updates (i.e. updates which have been received by a node that is in the correct cell).

We only show the update delivery ratio for varying node densities. The values do not depend on the node speed because of the HLS update mechanism: updates are sent to cells which have fixed positions. There is no location inaccuracy in target information.

The graph reveals two effects: the update delivery ratio decreases with growing hierarchy level and increases with higher node density. The dependence on

[4]If the update does not reach the correct cell, it is stored at the node where this is detected. The delivery of the update is retried with a handover packet after 2 seconds.

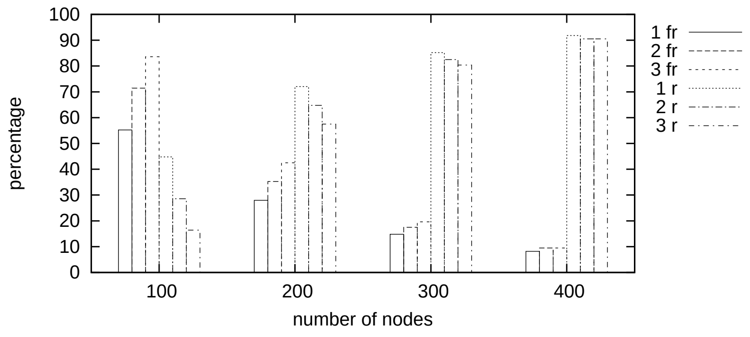


Figure 5.12: How updates are received

node density is apparent. With growing node density, greedy connectivity increases and also the probability to find a node in the target cell.

The update delivery ratio decreases with growing hierarchy level because the average distance between update sender and responsible cell increases. If a packet has a probability of $p \in [0,1)$ to be forwarded on the next hop, the probability to reach the target decreases with the number of hops (h) the packet has to take in total:

$$p^h > p^{h+1} \text{ for } p \in [0,1), h > 0$$

In general, the 300 and 400 node scenarios with a node density of over 75 nodes per square kilometer and the use of greedy forwarding of update packets achieved update delivery ratios of 80 percent and more for the first three hierarchy levels. Networks with a larger geographical extension require more hierarchy levels. As the update delivery ratio decreases for growing hierarchy levels, a minimum node density of 75 nodes per square kilometer seems necessary to achieve a sufficient ratio. Another method to accomplish a sufficient update delivery ratio is the usage of an appropriate greedy forwarding failure recovery technique like the GPSR perimeter mode.

5.8.3 Handover

Handover packets are the basic mechanism used to keep location information in the responsible cell if the node designated to be location server for this information leaves the cell. Therefore, we were interested in the distance between handover sender and target cell at the moment of sending. Figure 5.13 shows this distance. One should keep in mind that position information in update packets which could not be forwarded to the correct cell is delivered by handover packets, too. For this reason, higher handover distances occur when the position information could not be delivered in the first run, compare Figure 5.12. If node speed is higher, more handovers are sent due the actual reason for which the handover mechanism was implemented. This explains the spread of the percentage for increasing speeds. All handovers in the area from 150 to 300 meters are sent from the neighbor cells of the RC, therefore the first bar represents the handovers from these neighbors.

The graphic shows that the handover mechanism accomplishes two tasks: it forwards the location information which could not yet be delivered plus it keeps this information in the RC in case the location server leaves the RC.

5.8.4 Failure Analysis

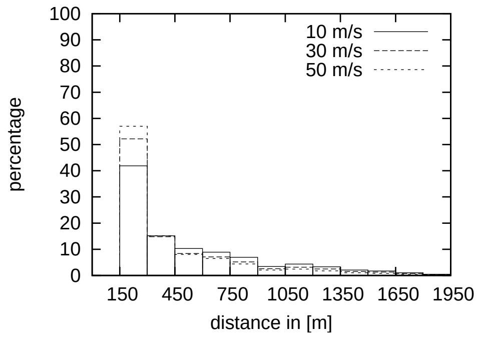
To achieve a better understanding of HLS, we analyzed in detail why requests failed. Figure 5.14 shows how a location query is treated by the distributed HLS algorithm. To distinguish the final states, each state is marked with a label. These final states consist of failure and success cases, the former have the subsequent meaning:

HTO Home perimeter timeout. The request has been sent to all candidate cells and after that, a home perimeter has been sent unsuccessfully. No location server was found.

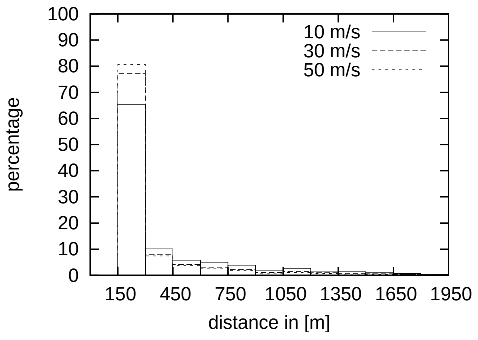
HTO after H_sa Home perimeter timeout after home perimeter send answer. A home perimeter packet was sent and a location server found, but the reply of the location server did not reach the home perimeter sender. This case is abbreviated as H_sa in the following.

HAS Home perimeter already sent. The request could be forwarded to a location server by using the home perimeter mechanism but there was no route to the target.

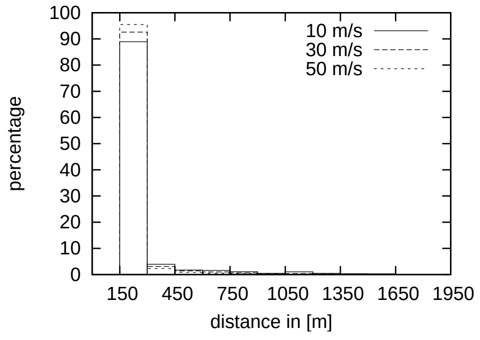
Reply drop No route for a reply packet could be found.



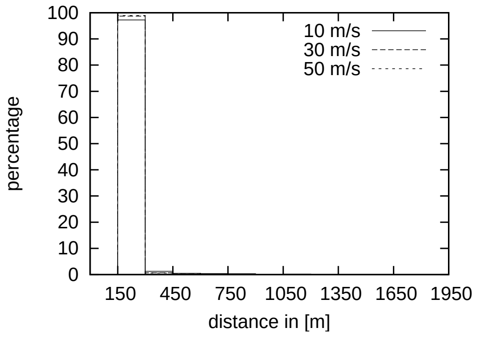
(a) 100 Nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.13: Handover send distance

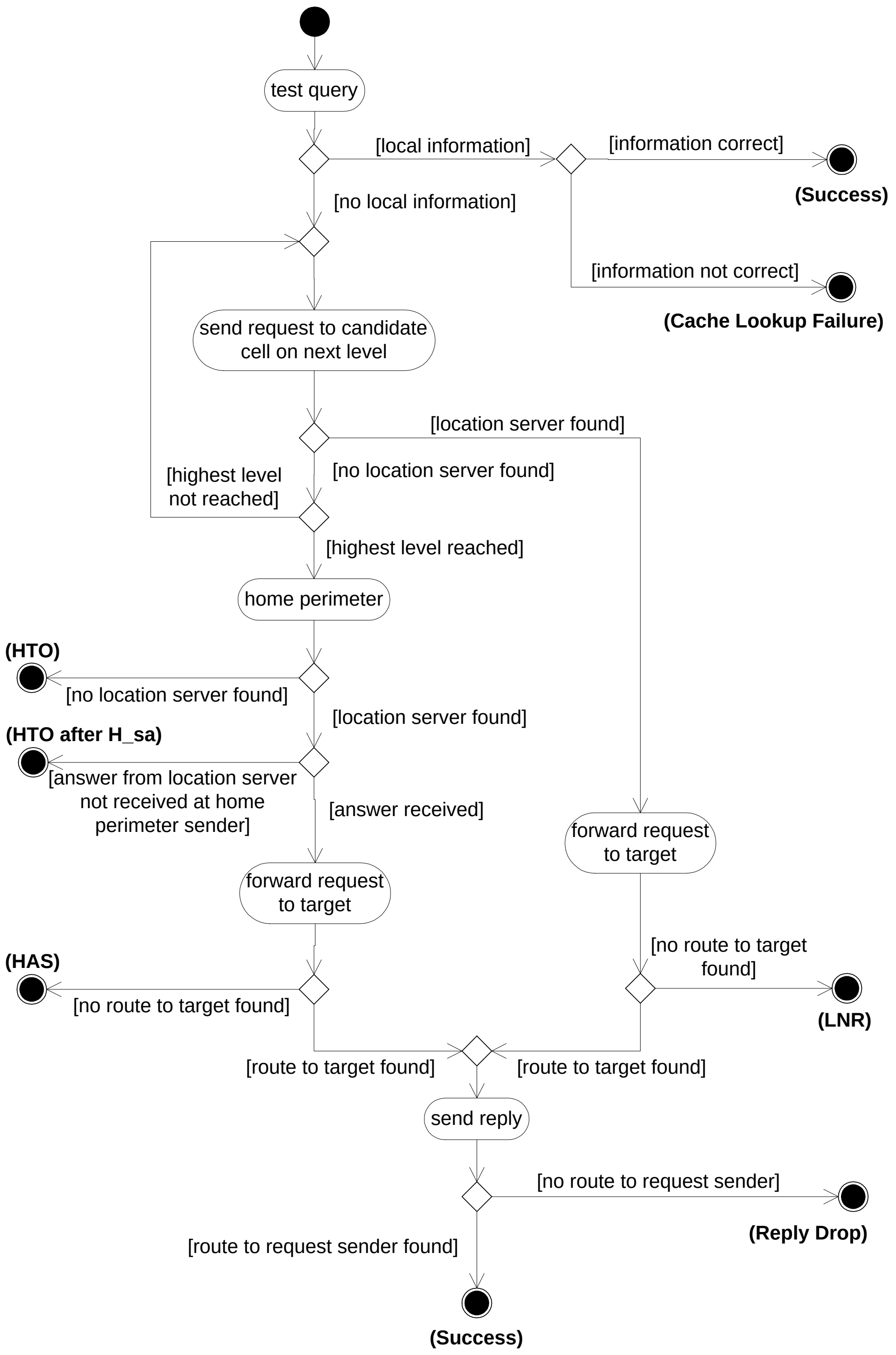

test query
[local information]
[information correct]
(Success)
[no local information]
[information not correct]
(Cache Lookup Failure)
send request to candidate cell on next level
[location server found]
[highest level not reached]
[no location server found]
[highest level reached]
home perimeter
(HTO)
[no location server found]
[location server found]
(HTO after H_sa)
[answer from location server not received at home perimeter sender]
[answer received]
forward request to target
forward request to target
(HAS)
[no route to target found]
[no route to target found]
(LNR)
[route to target found]
[route to target found]
send reply
[no route to request sender]
(Reply Drop)
[route to request sender found]
(Success)

Figure 5.14: Treatment of a location query by HLS

LNR Located but no route. The request could be forwarded to a location server without using the home perimeter mechanism but there was no route to the target.

Cache lookup failure The query could be answered from the location cache but the position information had a deviation of more than 250 meters. The cache lookup error in all scenarios is below 1 percent of all queries for HLS. Therefore it is not listed in our failure case evaluation.

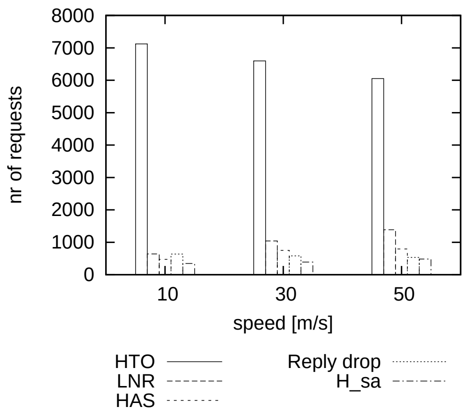
The Figures 5.15 and 5.16 show the reasons for failure in the greedy and perimeter simulations. The y-axis contains the absolute number of queries which failed out of the 12000 test-queries which have been launched (see Section 5.1.1). Presenting relative numbers would distort the fact that in dense scenarios only few queries fail. This should be kept in mind when regarding the graphs. While 3000 failures for 12000 request is a significant number, 15 failures are insignificant.

The reason for low success rates of HLS in the 100 node scenarios of the greedy simulations are obvious (Figure 5.15(a)): around 50 percent of all queries fail because no location server can be found. The low node density impedes the HLS algorithm from efficient functioning. From Section 5.8.2 we know that only few updates actually arrive at the target cell. Therefore requests cannot find a location server.

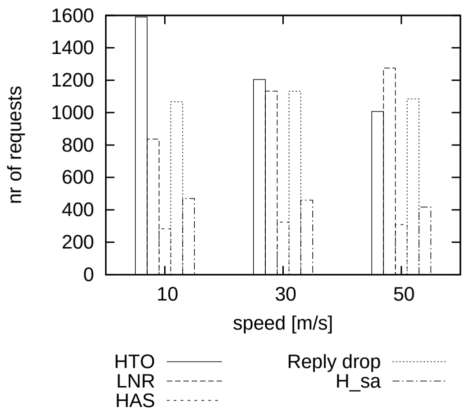
In general, two trends are immanent. The number of requests which cannot find a location server decreases with higher speeds. Secondly, the number of requests that find a location server but cannot be delivered to the target increases. Both effects are interconnected and result from the movement effect described above.

The picture becomes clearer when investigating Figure 5.16 which shows the failure reasons for perimeter simulations. While the ”no location server found” reasons dominates the 100 node scenario, it only negligibly affects the other scenarios. Requests mainly fail because no route could be found, either for requests which have already met a location server or for replies.

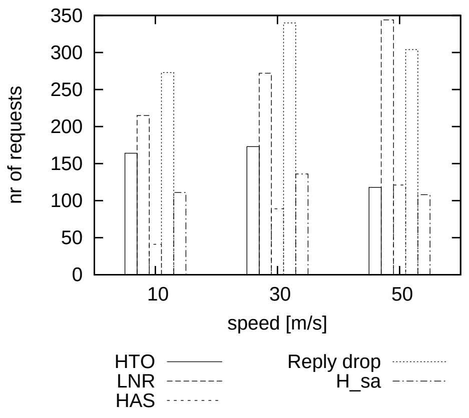
If a request fails with the ”LNR” or ”HAS” reason, a location server has been found but the request could not be delivered to the target. This can happen because of two reasons: the position information is not exact enough or the routing was not



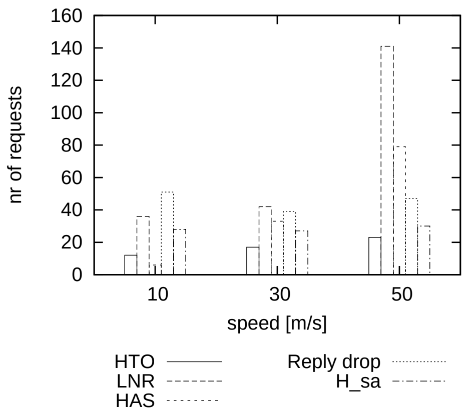
(a) 100 Nodes



(b) 200 Nodes

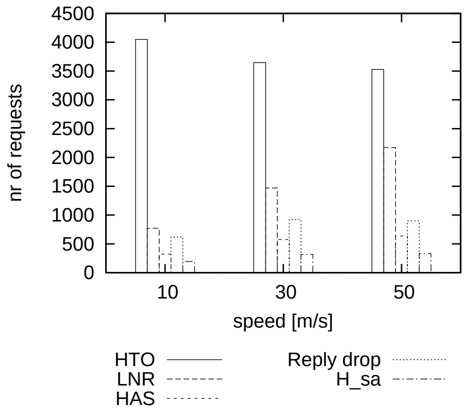


(c) 300 Nodes

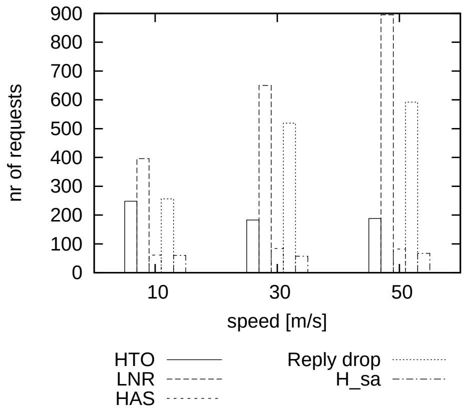


(d) 400 Nodes

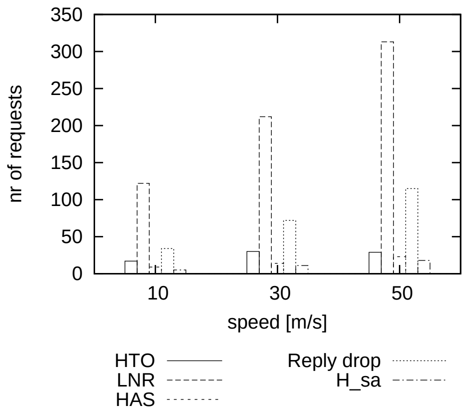
Figure 5.15: Failure reasons in greedy simulations



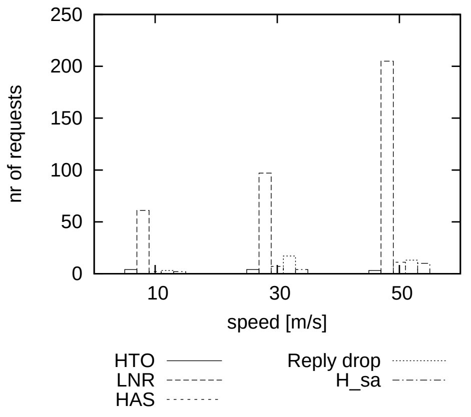
(a) 100 Nodes



(b) 200 Nodes

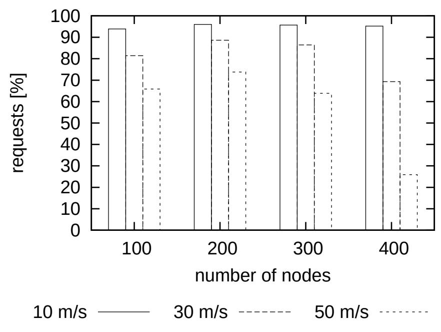


(c) 300 Nodes

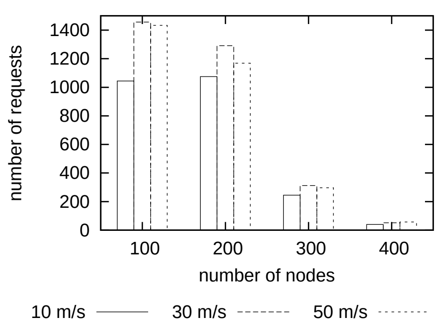


(d) 400 Nodes

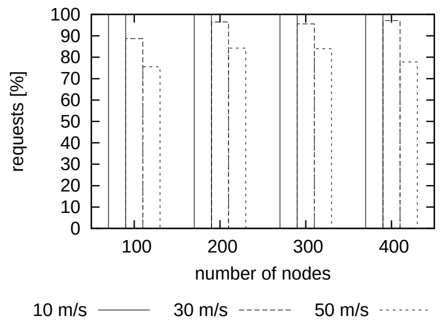
Figure 5.16: Failure reasons in perimeter simulations



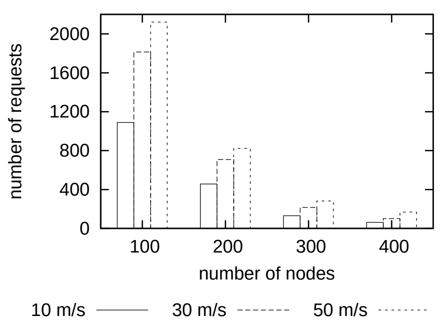
(a) greedy simulations, percent



(b) greedy simulations, absolute



(c) perimeter simulations, percent



(d) perimeter simulations, absolute

Figure 5.17: Requests which are dropped although the position information about the target is still precise enough

able to find a route to the target although the information is still exact, e.g. because of TTL expiration, network partitioning, loops on perimeters, lost neighbors, etc. Since we want to evaluate the performance of the location service, we traced the deviation of the location information when a packet was dropped with the ”LNR” or ”HAS” reason. The requests which have been dropped although the position information had a deviation of less than 250 meters are presented in Figure 5.17. The graphs on the left-hand side show the percentage, the graphs on the right the absolute numbers.

For both greedy and perimeter simulations, the percentage of drops with exact information decreases with growing speed: while a position of a node that moves with 10 m/s is correct for 25 seconds, it is correct only for 10 seconds at

25 m/s. Nevertheless the majority of positions were known precisely enough for the perimeter runs.

With increasing speed, the movement effect leads to a better distribution of location information and more requests find a location server. Less requests get dropped due to ”no location server found” reasons and more request get dropped after a location server is found. As a consequence, also the number of requests which get dropped with exact information rises as presented in Figure 5.17(b) and 5.17(d).

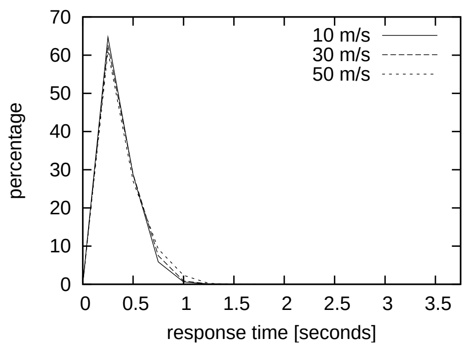
5.8.5 Response Time

The time between sending a request and reception of a reply is called *response time*. Figure 5.18 presents the response time of HLS in perimeter simulations for different scenarios. The graphs for greedy scenarios vary only slightly and are therefore not shown here. Each point at a time coordinate x in the chart stands for the interval $(x-0.25, x]$. To be able to judge the share of the location request process, the average reply age is shown in Figure 5.19. This is the time difference between sending the reply and its reception. It directly indicates how long it takes a packet to be forwarded from the target of the request to the source.

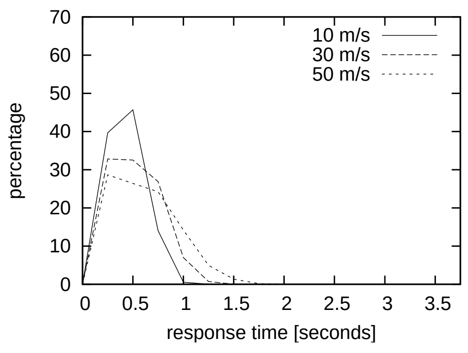
The response time is composed of:

- the time it takes a packet to be forwarded between the target candidate cells.
- the timeout for cellcasts if the cellcast fails (0.3 seconds)
- the time it takes the reply to reach the source

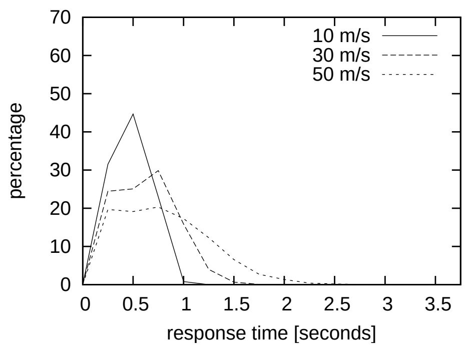
The shapes of the curves are results of the communication load on the network. This load increases with node density and speed because more updates are sent. A higher load produces more collisions resulting in retransmissions and in unwanted delays. For all scenarios, the response time for the majority of all requests remains below 2 seconds. Even in the 400 node, 50 m/s scenario, the total number of requests having a response time of more than 2 seconds is less than 6 percent. By comparing the response time to the average reply age, it is obvious that the response time scales well with the time it takes a packet to be forwarded between request sender and request target.



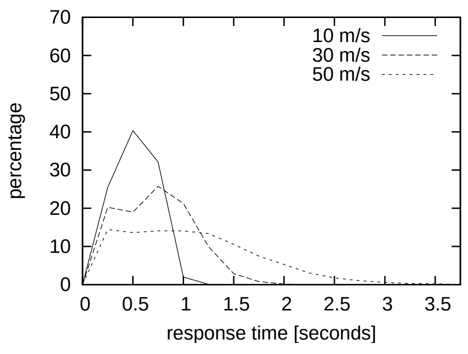
(a) 100 Nodes



(b) 200 Nodes



(c) 300 Nodes



(d) 400 Nodes

Figure 5.18: HLS Response time

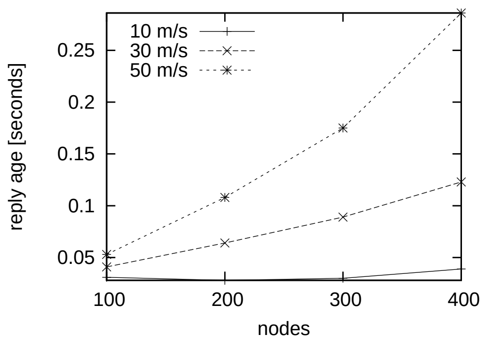


Figure 5.19: Average reply age

Level	Probability that $F(t,l,pos_S) = F(t,l,pos_T)$	Percentage of requests that find a location server at that level
l = 1	$9/144 = 6.25\%$	$9/144 = 6.25\%$
l = 2	$36/144 = 25\%$	$(36-9)/144 = 18.75\%$
l = 3	$144/144 = 100\%$	$(144-36)/144 = 75\%$

Table 5.6: On which level a request should find a location server in the 2×2 kilometer scenarios.

5.8.6 Location Servers

In this section, we evaluate on which level a request found a location server for the first time. Analytically speaking, a node should find a location server for another node in the RC of the region on the lowest level in which both nodes reside. With the cell building scheme used and two random node positions pos_S for a node S and pos_T for a node T, the probabilities for a request of S to find a location server of T are as presented in Table 5.6.

To verify the analytical results, we did some simulations with a program which has been developed for this purpose. The tool does the following:

1. Generate pos_S and pos_T randomly, the positions must be within the boundaries of the simulated area, i.e. 2×2 kilometers.

2. Calculate the level l for which $F(t,l,pos_S) = F(T,l,pos_T)$, i.e. the level on which the hash function computes the same responsible cell. For that purpose, the same class as in the ns-2 simulations has been used.

Additionally, the tool has a switch to test the influence of beacons: when source and destination are closer than 250 meters, S can receive beacons from T and knows the location of the target without sending a request. These ”lookups” are classified as being done on level ”0”.

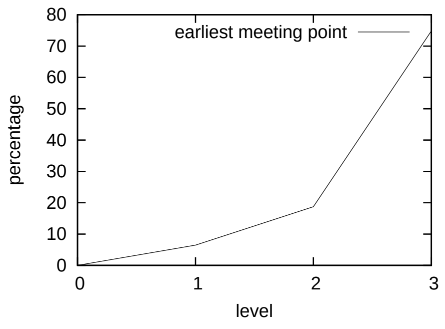
The results of these simulations are presented in Figure 5.20. Figure 5.20(a) shows on which level the responsible cells are equal for the case without attention to any beacons. Figure 5.20(b) shows the results when the beacons are taken into consideration. The graphs correspond to our analysis above.

The probability to find a location server in a responsible cell of a region is therefore equal to the probability that two randomly generated positions fall in the same region. This is especially interesting when these results are compared to the real localizations which are presented in Figure 5.21[5].

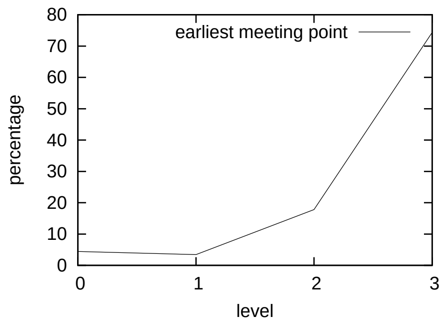
While the graphs for the greedy simulations in Figure 5.21(a) and 5.21(b) look similar to those of the analysis presented above (Figure 5.20), they differ a lot for the perimeter simulations in Figure 5.21(c) and 5.21(d). In the 100 nodes case of the latter, most of the requests find a location server on the first level. This is due to low node density. As already mentioned in Section 5.8.2, the probability to reach the target decreases with the number of hops a packet has to take. Only few requests reach the higher level RCs, and therefore, they cannot be answered on that level.

The influence of node speed in all graphs is a consequence of the movement effect. The probability to find a location server for the requested node in the first level RC rises with the speed of the nodes. More requests find a location server on the first level, less requests need to be forwarded to higher level RCs. The percentage nearly doubles from 10 to 50 m/s in Figure 5.21(d).

[5]We only show the 100 and 400 node scenarios because the scenarios with 200, 300 and 400 nodes produced similar results.

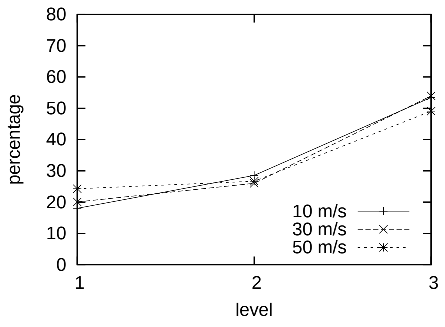


(a) without attention to beacons

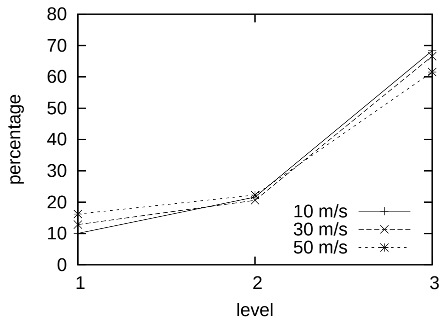


(b) with attention to beacons

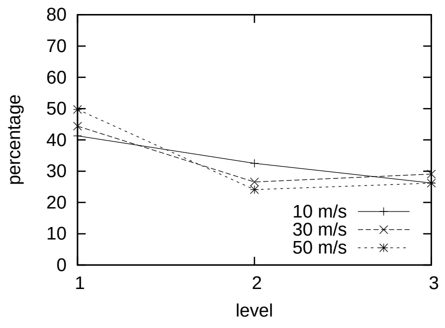
Figure 5.20: The result of the hierarchy simulation



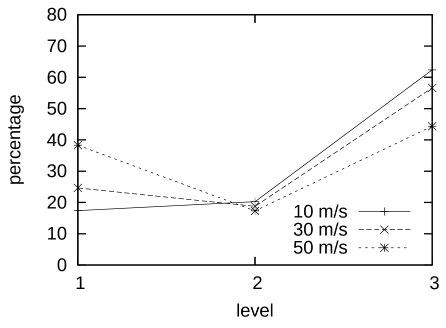
(a) 100 Nodes, greedy



(b) 400 Nodes, greedy



(c) 100 Nodes, perimeter



(d) 400 Nodes, perimeter

Figure 5.21: The levels in the hierarchy on which a location server was found

Chapter 6

Conclusion

6.1 Summary

This thesis presented a new location service for mobile ad-hoc networks, the Hierarchical Location Service. We gave a general description of the HLS algorithm and showed some possible extensions. The Hierarchical Location Service uses a hierarchical structuring of the MANET's area to calculate the position of location servers for other nodes. With this structure, the density of location servers for a node depends on the distance to this node. The communication complexity of a request therefore depends on the distance between request sender and request target.

To achieve the structure above mentioned, the area of the ad-hoc network is divided into cells which are grouped in regions of different hierarchy levels. By using a hash function, a node computes a set of these cells which host its location information. The hash function is only based on node id and current position. Hence, it only needs local knowledge. The Hierarchical Location Service does not select special nodes to become location servers, it selects geographical regions to contain a location server. Therefore, an arbitrary node within the cell stores the location information. If this node leaves the cell, the information is handed over to another node within the cell. A request for a node is sent to a possibly different set of cells which is also calculated with the hash function mentioned above. The design of the function guarantees that the intersection of these sets is non-empty.

Therefore, the request should meet a location server.

The Hierarchical Location Service was compared to the Grid Location Service based on simulations of a variety of scenarios with GPSR as routing protocol. We found that HLS in our simulations needs a certain minimum node density between 50 and 75 nodes per square kilometer to work well. In scenarios with a node density lower than this minimum, the Grid Location Service achieved better query success rates. In all scenarios with a node density equal to or above this minimum, HLS achieved higher success rates than GLS.

In scenarios with a low node density, requests mainly fail because no location server can be found. If node density is higher, HLS is able to find a location server in most cases. Here, the few request failures are mainly caused by the routing. Furthermore, we discovered that the Hierarchical Location Service in general consumes less bandwidth than the Grid Location Service and that HLS' performance is less affected by high node speeds.

We have shown that the Hierarchical Location Service 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[1].

To conclude the results of our simulations, the Hierarchical Location Service scales better than the Grid Location Service with respect to node density, speed and size of the area in which the MANET is deployed while it achieves comparable or better success rates.

6.2 Directions of Future Research

The research about the Hierarchical Location Service is by far not complete. The location service in its base version needs to be studied further. Future research might be dedicated to:

Improved caching Using aggressive caching might improve HLS' results espe-

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

cially for low node densities. Caching can be subdivided into different levels as described in the following:

- Cache all location information which is available right at a node, i.e. only packets which are routed by this node.
- Cache the information ”overheard” in the promiscuous transmission mode, i.e. all packets that are received on the MAC layer.

Data traffic Until now, all simulations were carried out only simulating request traffic, i.e. no real data traffic was sent. Studying HLS’ behavior maybe in contrast to that of GLS in presence of data packets should provide more interesting results.

Very large scenarios HLS is designed to scale to large networks, but memory and computation power limitations impeded the simulation of networks with a size of more than 5×5 kilometers at a node density of 75 nodes per square kilometer. However, HLS should be simulated in larger scenarios to determine the limits of this new approach. This can be either achieved by using more powerful simulation hardware, writing a customized simulator or using a higher level of abstraction to avoid the simulation of every single data packet.

Optimization of the hierarchical structure A question which has not been addressed yet is the ideal number of hierarchy levels and their structure. In our simulations, we used small-sized cells and an arbitrary number of hierarchy levels. The determination of the ideal cell structure and hierarchy is subject to further research.

Some of the possible starting points for improvements of the Hierarchical Location Service might be the extensions mentioned in Section 3.5 and the not yet implemented features of Section 4.1.2. Besides, there are a lot of possible improvements to look into in order to change the basic HLS algorithm:

Hitchhike mode for updates Particularly the location updates are sent very frequently and the size of the packets is small. As a consequence, there happen to be a lot of collisions. A possible measure against those collisions might be the piggybacking of location updates on other packets. Update packets do not have to arrive at the target area in a few milliseconds, also a delay of some seconds is tolerable. Thus, location updates can ”hitchhike” on other packets: whenever a packet passes the node where the update is actually located and leaves this node on the same hop as the update packet would do, the location information can be piggybacked on the passing packet. In contrast to pure piggybacking, the update information can use a different packet for every hop. Only if the update cannot be forwarded to its target in a predefined timespan, extra update packets have to be send.

Nondeterministic cell scheme Currently, the cells are calculated strictly by a geometric subdivision of the ad-hoc network area. While this works well for a scenario with random node movement and no movement limitations, it will not work for a scenario with environmental restrictions. When using HLS e.g. in a car based scenario, it does not make much sense to place cells on lakes, buildings, mountains and the like. Furthermore, handover and update traffic could be reduced if the cells are optimized for the layout of lanes on streets: if when a node leaves a cell, it tries to handover the information to another node moving towards the opposite direction. The new location server enters the cell and holds the information until it leaves the cell on the other side where the information is handed over to another node. Update packets in the above mentioned ”hitchhike” mode could even stay with a node if this node will enter the target cell in a few seconds.

Adaptive cell scheme In order to work properly, location services like GLS, HLS, GRSS or Homezone need to be preconfigured with the area of the ad-hoc network. Participating nodes need to know the size and form of the area in which the network should be used. It would be interesting to evaluate HLS in scenarios without a predefined area. For this purpose, HLS has to be modified. Possible modifications are described in the following:

- The structure of the tree is built generically in a bottom-up way. The cells are all fixed, they can be for example computed based on a modulo operation on coordinates. In the next steps, the regions are built. Nodes can mark their update packets with a flag indicating that they want to be informed if the RC was reached by the update. After having send some updates, the node should be able to determine the form and extend of the ad-hoc networks' area with enough detail to determine which hierarchy levels do exist.

- An algorithm to detect the border of the network is employed. Either some interior nodes periodically send "border discovery packets" that travel in east-west or north-south direction, or nodes try to determine if they belong to the border of the network. The information acquired by these mechanism is then flooded to the network.

It depends on the scenario if the border discovery mechanism has to be executed periodically or just once at the startup op the network.

Home perimeter The home perimeter method proposed by Karp ([RKY+02]) may improve the identification of HLS location servers in sparse scenarios and may make the information more stable. This seems to be a promising approach particularly in connection with the indirect location scheme proposed in Section 3.2. RCs on higher levels should be updated with the lowest frequency possible to avoid traffic. If an algorithm such as the home perimeter method can guarantee to a high degree that no information is lost, the updates only have to be sent when required by the algorithm. Thus, updates to prevent loss on higher levels will not be necessary any more.

Omnipotent routing During our simulations, it became obvious that HLS interacted in a harmful way with GPSR. HLS sends its packets to cells, not to nodes. Thus, whenever a packet is dropped by the routing, it is not clear if the packet was dropped because the target cell was empty or due to any other reason. To avoid this interaction, another point on the future research list is the implementation of an idealized, omnipotent routing. This routing protocol uses global information for its routing decisions and therefore avoids packet drops due to inconsistencies. It selects all nodes which are in radio range of the target position, calculates paths to these nodes and forwards the packet on the link which belongs to the shortest of those paths[2]. This decouples the location service from effects caused by the routing while it is still possible to use an arbitrary MAC layer.

Compress position information Methods like the Bloom Filter technology [Blo70] could be used to compress the position information in responsible cells and handover packets. With the indirect location scheme, RCs on higher levels contain information similar to the summaries generated by GRSS. Therefore, the same data reduction methods can be applied.

Analytical estimation of HLS scalability From the large scale simulation in Section 5.7.2 we know that HLS scales well with the size of the simulation area and the number of participating nodes. It would be interesting to do an analytical estimation of the scalability of HLS and try to find a theoretical boundary which limits this scalability.

[2]In order to achieve this, it uses global knowledge to determine the nodes in radio range.

Appendix A

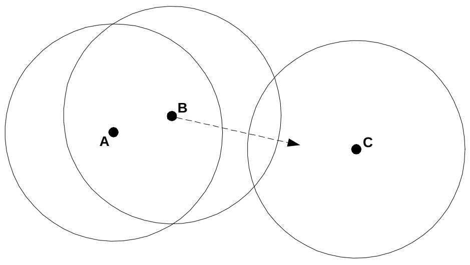
Implementation

A.1 GPSR Error

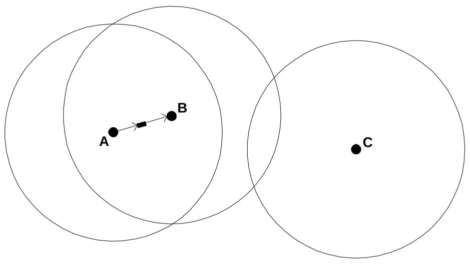
During the simulations for the diploma thesis, a not yet described error case of GPSR in mobile scenarios occurred. In the following, the error case will be described in the smallest scenario possible. The relevance of this error for a real-world implementation of GPSR will be analyzed thereafter.

In our example, node A wants to send a packet to node C. As shown in Figure A.1(a), node A and B are in each others radio range, C is out of the radio range of both B and A. Node B moves in C's direction but does not enter its radio range during the communication. A now sends a packet destined to C (A.1(b)). The packet reaches B where it is switched to perimeter mode because no better greedy neighbor can be found. It is forwarded to A and once again forwarded to B (A.1(c)). In the time it took to transmit the packet, B has moved a bit in the direction of C. The distance between B and C is smaller than the distance between the point at which the packet entered perimeter mode, B's former position. Therefore, the packet is switched to greedy mode. Since no better greedy node with respect to C can be found, the packet is once again forwarded in perimeter mode. It reaches A and is returned to B and so on (A.1(d)). The packet has entered a routing loop from which it can not recover with the original GPSR algorithm.

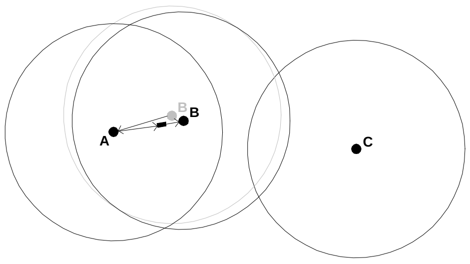
While this problem would not occur in the scenario described above in a real-world implementation due to the limited spatial resolution of the positioning ser-



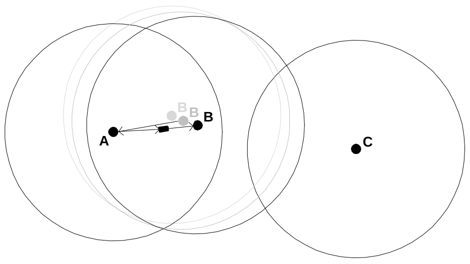
(a) B moves towards C



(b) A sends a packet with target C in greedy mode



(c) B forwards the packet in perimeter mode and moves on



(d) The packet has entered a loop between A and B

Figure A.1: The GPSR error case in the smallest scenario possible

vice (e.g. GPS), now consider a scenario with some more nodes. Node A is connected with some other nodes which have at least the same distance to C as A, but none of those is closer to C than B. The packet would loop and return to B after some hops. Depending on the number of hops, at least 0.5 - 1 seconds could have passed. During this timespan, it is quite likely that node B has covered a distance significant enough to be noticed by the positioning service. The packet loops until the TTL of the packet expires (or C could be reached).

The problem can be solved partially by including the node id in the decision whether to switch between perimeter and greedy node. On the other hand it is likely, e.g. in highly dynamic scenarios, that a link exists between the former partitions due to node movement after the loop of the packet.

A.2 Tracing

Besides the standard ns-2 trace information (see [Käs03, NS2]) we have added our own traces to get a better insight into HLS. These traces track important events for HLS and always start with the prefix ”HLS_”.

HLS_H_s Handover send: a handover packet is sent.

HLS_H_rr Handover real receive: a handover packet is received in the correct cell.

HLS_H_fr Handover forced receive : a handover packet is received in a wrong cell (because the routing could not find a way to the correct cell).

HLS_CC_sr CELLCAST send request: send a cellcast, a broadcast to reach every node in the actual cell.

HLS_CC_sa CELLCAST send answer: send an answer to a cellcast.

HLS_CC_da CELLCAST destroy answer: destroy the scheduled answer because an answer from another node was received.

HLS_CC_ra CELLCAST receive answer: the sender of the cellcast receives an answer.

HLS_CC_to CELLCAST timeout: no answer for a cellcast was received.

HLS_UPD_s Update send: send a position update to a RC.

HLS_UD_r Update receive: receive an update in the correct cell.

HLS_UD_fr Update forced receive: receive an update in the wrong cell (because the routing could not find a way to the correct cell).

HLS_REQ_s Request send: send a request.

HLS_REQ_a Request answer: answer a request, this can only be done by the target of the request.

HLS_REQ_u Request unforwardable: no route to the target candidate cell.

HLS_REQ_l Request located: a node has information about the target of the request and updates the target position.

HLS_REQ_n Request next (level): the request is forwarded to the candidate cell on the next hierarchy level.

HLS_REQ_d Request drop: a request had to be dropped.

HLS_CIC_s Circle cast send: send a home perimeter (in the implementation called ”circle cast”).

HLS_CIC_d Circle cast drop: drop a home perimeter packet.

HLS_CIC_f Circle cast next cell: send the home perimeter packet to the next cell on the list.

HLS_CL Cache Lookup: the node has information about the target of a query, it therefore returns the value found in the cache.

HLS_CUpd Cache update: indicates evaluation of piggybacked location information during a communication connection.

HLS_REP_d Reply drop: no route to requesting node.

HLS_REP_r Reply receive: the sender of a request receives a reply packet from the target node.

A request normally produces more than one packet, at least a request and a reply. Therefore each request gets a unique request id, composed of the node id of the request sender and a consecutive number which is contained in any HLS packet except the updates. This facilitates the assignment of a packet or event to a request. If we have a ns-2 HLS trace file named ”hls_trace.txt.gz”, the command[1]

```
zcat hls_trace.txt.gz | grep ’10_0’
```

prints all events and packets connected with the request id ”10_0”, i.e. the first request send by node 10 during the actual simulation run.
With the command

```
zcat hls_trace.txt.gz | grep ’HLS_.*10_0’
```

all special HLS event traces are printed, an example output might look like this:

```
HLS_REQ_s 39.049339824507 (10_0) 10 ->59 <60 88.00 968.00 (1)>
HLS_REQ_f 39.049896576668 (10_0) 76 ->...->59
HLS_REQ_f 39.050897224732 (10_0) 90 ->...->59
HLS_REQ_f 39.051881872788 (10_0) 76 ->...->59
HLS_REQ_f 39.052886070695 (10_0) 79 ->...->59
HLS_REQ_f 39.054290450897 (10_0) 93 ->...->59
HLS_REQ_l 39.055514937856 (10_0) 12 ->59 [19.16698 90.50 21.67] <48 60 (1)>
HLS_REQ_f 39.056335423196 (10_0) 54 ->...->59
HLS_REQ_f 39.109430985297 (10_0) 36 ->...->59
HLS_REQ_l 39.110491763076 (10_0) 56 ->59 [38.20465 37.07 255.29] <25 60 (1)>
HLS_REQ_l 39.171253741906 (10_0) 38 ->59 [38.77669 38.42 261.34] <12 60 (1)>
HLS_REQ_a 39.172273877499 (10_0) 59 -> 10
HLS_REP_r 39.268470817289 (10_0) 10 <-59 [39.1723 39.36 265.54] {1.04}
```

Explanation: Node 10 sends a request for node 59 to cell 60 on level 1 (REQ_s). The request is forwarded via node 76, 90, 76, 79 and 93 (REQ_f) until it reaches node 12 which has location information about node 59 (REQ_l). It forwards (REQ_f) the request via node 54, 36, 56 and 38 (the two last nodes have more exact position information about node 59 and therefore update the position in the request packet header (REQ_l)) until it reaches node 59. This node generates a reply (REQ_a) which is received by node 10 (REP_r).

[1]We assume that the reader is familiar with linux and bash.

A.3 The Visualizer

The visualizer has been developed as a test environment for the cellbuilder, the class which implements the hash function to calculate the responsible cells. The functionality of the visualizer has then been extended, it can now be used as a visual debugging tool for HLS. Therefore, it parses at startup the special HLS entries (see Section A.2) in a ns-2 trace file. The actual version can only analyze static scenarios.

Figure A.2 shows a screenshot of the visualizer. The scrollable screen in the center of the window contains the current scenario. Cells are quadratic and painted in different shades of gray in the picture. Already parsed requests can be found in the panel on the left hand side, they get painted by clicking on them. Above the center screen is the command line. The subsequent commands are available for the command line, *italic* pieces stand for command arguments, ()-brackets mark optional arguments.

cell *id* The cell with the given id will be highlighted.

rc *node level* Highlights all responsible cells for node on the given level.

hierarchy *node1 [node2]* Highlights the hierarchy for node1 seen from the position of node1, or, if also the second argument is given, from the position of node2.

cls Cleaning the screen.

zoom *factor* Sets the zoom to the given factor. Default is 1.0 . E.g. if the factor is 0.5, the screen is displayed with half the size. The zoom factor is absolute (this means to get back to default, you have to type ”zoom 1.0”). It may be necessary to do a ”cls” after zooming.

bc *node* Shows the radio range (bc = broadcast) of the given node.

request *requestId* Shows the request with the given id (which must be in the format node_number, e.g 17_0). The trace file will be completely parsed for the request. It may be better to uses the already parsed requests in the scrollbar on the left side).

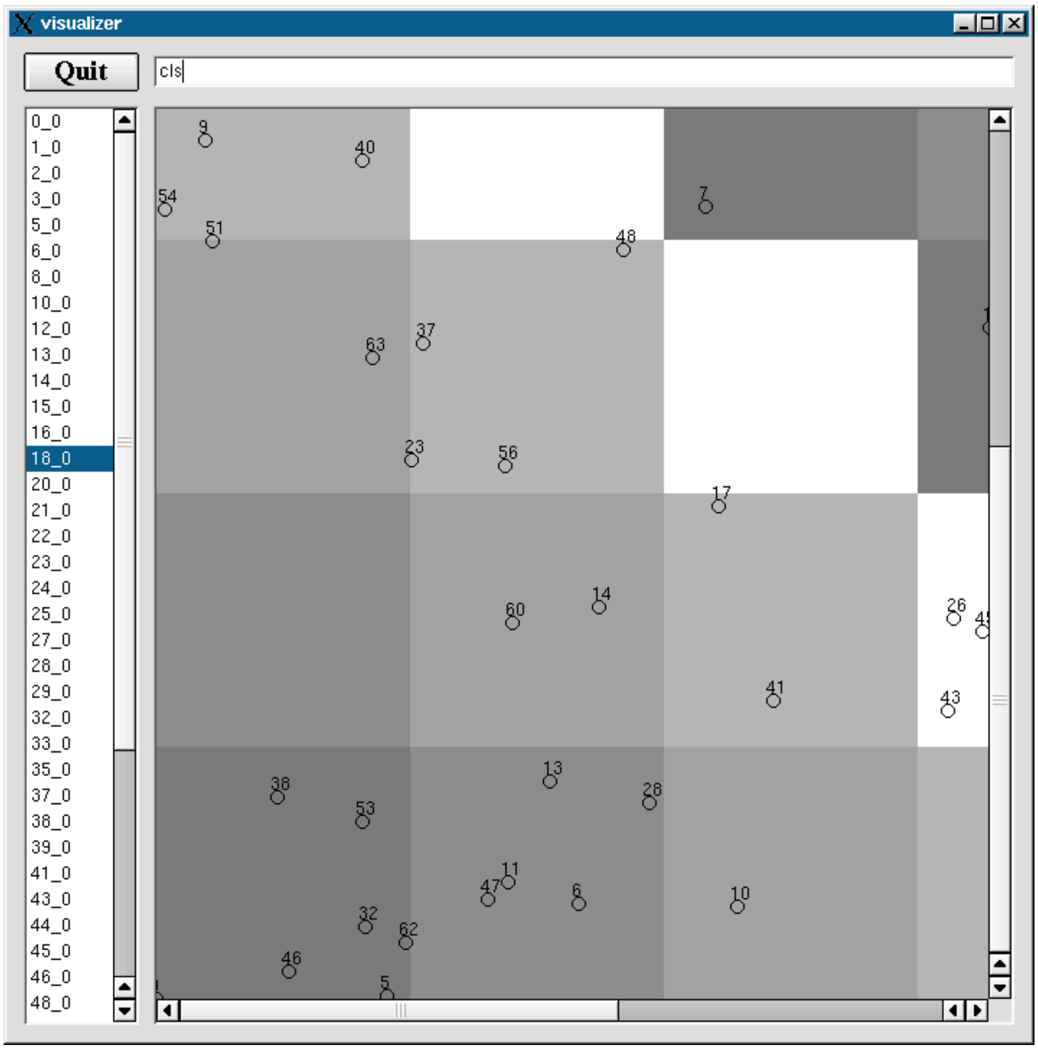


Figure A.2: Screenshot Visualizer

poss *node* Highlights the active position servers of node.

screenshot *filename* Saves the current screen under the given filename (which must end with .png because the file is in png-format) to the file system.

nb *cellid* Marks all neighbor cells of the cell with the ID cellid.

help Display a help screen with the available commands.

Appendix B

Tools

In this chapter we present evaluate.pl, a tool which has been used for the evaluation of the HLS ns-2 simulations. A detailed description of the other tools used for the evaluation and simulation, run.tcl, scengen and trafgen, can be found in [Käs03].

B.1 Evaluate.pl

Evaluate.pl is a perl script which evaluates the trace file produced by a ns-2 simulation run. The version which we used for the evaluation of the Hierarchical Location Service is an adapted variant of the script developed for the fleetnet project ([FN]) at the University of Mannheim, Lehrstuhl für Praktische Informatik IV [PI4].

The following listing contains an example output of this perl script together with short explications.

General statistics Information about the simulation setup.

```
Statistics:
----------
5 GPSR Runs evaluated (Mac/802_11)
300 Nodes in an Area of 2000x2000 sqm for 300.000 secs
Nodes moved at max. 10 m/s (Average:   5.250 m/s)
----------
```

Overview Number of test queries, cache lookups, request, replies and the age of the location information.

```
Queries          : 6000
Cache Lookups    : 304 min age 0, max age 9.89029610, avg age 1.64769762
Requests send    : 5696
Replies received: 5304 min age 0, max age 0.442737950, avg age 0.029708578
```

Request details Total number of requests dropped, number of requests dropped divided by level and drop reason. A description of the reasons can be found in Section 5.8.4.

```
Requests dropped: 262
       1 LNR : 4
       2 LNR : 8
       3 HAS : 22
       3 HTO : 144
       3 LNR : 84
```

Reply details Total number of replies dropped, number of replies dropped divided by drop reasons: IFQ = overflow in the interface queue between the link layer and the MAC layer, TTL = TTL expiration, CBK = MAC callback, NRTE = no route.

```
Replies dropped : 128
IFQ drops       : 0 (req 0, reply 0)
TTL drops       : 4 (req 0, reply 4)
CBK drops       : 0 (req 0, reply 0)
NRTE drops      : 128 (req 0, reply 124)
```

Failure rate Percentage of queries which could not be answered correctly. Calculated with the formula:
failure rate $=$ queries $-$ successful cache lookups $-$ replies.

```
failure rate    : 6.53 %
```

Request drops with exact information How much of the request dropped had exact enough location information about the target node. Divided by the method how a location server was found. ”Info still exact enough” = found location server in a cell; ”HAS info still exact” = found location server after an artificial home perimeter.

```
Info still exact enough           : 96
HAS info still exact              : 15
```

Script check Lists requests for which no drop or answer event has been detected, test-queries without a cache lookup or a request send (this may happen if the node already launched a query for the target, can happen sometimes due to the randomly generated requests) and replies which have been sent but for which no drop or receive event is detected.

```
Unanswered requests :
test-queries without cl or req_s :
Missing replies: 0
```

Response time See Section 5.8.5.

```
Request Travel Time :
0 s: 31.50 %
0.25 s: 45.34 %
0.5 s: 22.38 %
0.75 s: 0.74 %
1 s: 0.04 %
```

Location servers The parameters of the cell in which a request found a location server. Same cell = target RC of the request, Not same cell = another than the target RC. Also see Section 5.8.6.

```
Request first locate:
Same cell
Level 1 : 12.39 %
Level 2 : 11.74 %
Level 3 : 51.42 %
Not Same cell
Level 1 : 2.29 %
Level 2 : 4.25 %
Level 3 : 17.92 %
```

Artificial home perimeter How much artificial home perimeter (CIC) packets have been sent and dropped with respect to the position of the cell for which it was necessary: cells with 3 neighbors are corner cells, cells with 5 neighbors are cells at the border of the simulation area, the rest are inner cells.

```
CIC results
total cics : 273
Cell with 3 neighbors: 13
Cell with 5 neighbors: 120
Cell with 7 neighbors: 140

CIC drops with respect to number of neighbors
Cell with 3 neighbors: Dropping 7 requests
Cell with 5 neighbors: Dropping 86 requests
Cell with 7 neighbors: Dropping 81 requests
```

Update statistics Reasons for which updates have been sent together with the level (see Section 5.8.1) and how updates have been received (see Section 5.8.2).

```
Update statistics :
send:
1 : 9743
1 DIST : 4127
1 NWRC : 5616
2 : 8024
2 DIST : 5120
2 NWRC : 2904
3 : 44334
3 DIST : 10
3 NWRC : 1490
3 TIME : 42834

received :
1 fr : 1392 (14.29 %)
1 r  : 8351 (85.71 %)
2 fr : 1427 (17.78 %)
2 r  : 6599 (82.24 %)
3 fr : 8082 (18.23 %)
3 r  : 36250 (81.77 %)
```

Query distance Shows the distribution of the distance between sender and target of a query.

```
Query Source - Destination Distance :
0 to 250 m : 3.97 %
250 to 500 m : 11.52 %
500 to 750 m : 15.25 %
750 to 1000 m : 17.27 %
1000 to 1250 m : 17.7 %
1250 to 1500 m : 15.28 %
1500 to 1750 m : 10.53 %
1750 to 2000 m : 6.03 %
2000 to 2250 m : 1.98 %
2250 to 2500 m : 0.43 %
2500 to 2750 m : 0.03 %
```

Cache lookup deviation Shows the distance between the current position of a node and the position which was found in the local location cache.

```
Cache Lookup Deviation :
0 to 50 m : 95.39 %
50 to 100 m : 4.61 %
Entries above radio range: 0 %
This corresponds to 0.00 percent of all queries
```

Handover statistics The number of handover packets which have been sent and received. ”Real receive” means reception in the correct cell. ”Forced receive” means reception in another cell, this also includes all handovers which have not been transmitted by the handover sender. A handover packets may contain information about more than one node, therefore the number of entries per handover packet together with the number of packets is listed.

```
Handovers:
Send           : 33151
Real receive   : 9439
Forced receive : 23712
Handover Send Distance :
0 to 150 m : 0 %
150 to 300 m : 88.93 %
300 to 450 m : 3.92 %
450 to 600 m : 1.73 %
600 to 750 m : 1.55 %
750 to 900 m : 1.09 %
900 to 1050 m : 0.46 %
1050 to 1200 m : 1.06 %
1200 to 1350 m : 0.47 %
1350 to 1500 m : 0.3 %
1500 to 1650 m : 0.27 %
1650 to 1800 m : 0.1 %
1800 to 1950 m : 0.02 %
1950 to 2100 m : 0.06 %
2100 to 2250 m : 0.02 %
2250 to 2400 m : 0.02 %
2400 to 2550 m : 0.01 %

Handovers number of infos per Packet
1      : 8606
10      : 27
11      : 8
12      : 1
13      : 3
2      : 8195
3      : 7845
4      : 4917
5      : 2225
6      : 800
7      : 329
8      : 138
9      : 57
Average of 2.67 infos per handover

------ HLS statistics end --------------------------------------
```

Packet statistics Gives information about the number of packets which have been sent, forwarded, received and dropped grouped by packet type.

```
Packet Statistics:

                      Sent       Forw       Recv       Drop
       RTR/GPSR
              BEACON   147293          0    1855483          0
       RTR/HLS
              CCREPLY     431         74        498         15
              CCREQ      1777          0      22255          0
              CIREQU       48        303        341          0
              HNDOVER    2599        619       3179          0
              REPLY      1182       5572       6351         25
              REQUEST    3019      12303      14461          0
              UPDATE    13067      49737      58949          0
```

Bandwidth Consumption Gives information about the MAC load and the number of packet sends and forwards per run separated by MAC and routing layer.

```
Bandwidth Consumption (kB/Run):

MAC    : 23151.96 kB/Run (360132.20 s/f)
  GPSR :  9061.97 kB/Run (147293.00 s/f)
  HLS  : 10969.34 kB/Run (128746.00 s/f)
  MAC  :  3120.65 kB/Run (84093.20 s/f)
RTR    :  4711.01 kB/Run (238031.20 s/f)
  GPSR :  1582.25 kB/Run (147293.20 s/f)
  HLS  :  3128.76 kB/Run (90738.00 s/f)
```

Appendix C

Ideas

This chapter contains ideas for further developments. These ideas have not been tested or evaluated, they should be seen as a sketch.

C.1 Cryptography

It might be interesting to apply public key cryptography on the problem of location obfuscation as presented in the following:

- A node encrypts its location by using its own private key and spreads this encrypted information with the location service in the network. When another node requests the position of a node, the input to the function is the encrypted information and the position of the source. The function is designed in a way to return a direction in which the packet has to be forwarded. In order to prevent exact triangulation of a node's position, a kind of ”position jitter” is introduced by the node.

- Web of thrust: locations only can be resolved if different nodes work together, the encrypted information has to be treated by different nodes possessing different parts of the key.

- We assume that the following conditions are true:

  1. It can be guaranteed that a calculation for which an actual location is necessary only can be done in the nearby of that location. In other

words: when the node which wants to do the calculation is asked for its coordinates, these coordinates can be trusted.

2. Location information can be encrypted in a fashion that only next hop calculation is possible: the input for the function are the encrypted coordinates and the actual coordinates of the node, the output is an approximate direction (with a certain ”location jitter”, i.e. artificial inaccuracy). The packet has to be forwarded in this direction.

If 1. and 2. are combined, the location of a node is obfuscated. Its position can still be triangulated, but it is not accurate and the attacker has to physically move to do this.

- Anonymous communication. When a node initiates a connection, it does not have to write its true id to the outgoing packets. It is sufficient to generate a connection id. Packets are routed with position information, the node id is irrelevant for the routing decision. Only in the neighborhood of a node, the assignment connection id - node id needs to be known. The node can communicate with others while being anonymous.

Bibliography

[BCSW98] S. Basagni, I. Chlamatac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (dream). In *Proc. of the 4th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MOBICOM) '98*, pages 76–84, Dallas, TX, USA, 1998.

[BGL00] Ljubica Blažević, Silvia Giordano, and Jean-Yves LeBoudec. Self-Organizing Wide-Area Routing. In *Proceedings of SCI 2000/ISAS 2000*, Orlando, July 2000.

[BGL01] Ljubica Blažević, Silvia Giordano, and Jean-Yves LeBoudec. Self Organized Routing in Wide Area Mobile Ad-hoc Networks. In *Proceedings of IEEE Symposium on Ad-Hoc Wireless Networks (Globecom 2001)*, San Antonio, Texas, November 2001.

[Blo70] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[BMJ$^+$98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the fourth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '98)*, pages 85–97, Dallas, Texas, October 1998.

[BMSU99] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge 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.

[BZ03] Nikhil Bansal and Liu Zhen. Capacity, Delay and Mobility in Wireless Ad-Hoc Networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, San Francisco, CA, March 2003.

[CHH02] Srdjan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad hoc networks. *Cluster Computing Journal*, 5(2), 2002.

[CJL$^+$01] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Quayyum, and Laurent Viennot. Optimized Link State Routing Protocol. Internet Draft, draft-ietf-manet-olsr-05.txt, work in progress, October 2001.

[CM99] Scott Corson and Joseph Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. IETF, RFC 2501, January 1999.

[CMU] The cmu monarch wireless and mobility extensions to ns-2. http://-www.monarch.cs.ċmu.edu/cmu-ns.html.

[FN] The FleetNet project. http://www.fleetnet.de.

[GAL] The Galileo project. http://europa.eu.int/comm/dgs/-energy_transport/galileo/index_en.htm.

[GH99] Silvia Giordano and Maher Hamdi. Mobility Management: The Virtual Home Region. Technical Report SSC/1999/037, EPFL-ICA, October 1999.

[GLO] The GLONASS project. http://www.rssi.ru/SFCSIC/english.html.

[Gro] Bluetooth Special Interest Group. The bluetooth specification. http://www.bluetooth.com.

[GS69] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[HGBV01] J. P. Hubaux, Th. Gross, J. Y. Le Boudec, and M. Vetterli. Towards self-organized mobile ad hoc networks: the terminodes project. *IEEE Communications Magazine*, January 2001.

[HH01] Zygmunt J. Haas and Marc R. Haas. The Performance of Query Control Schemes for the Zone Routing Protocol. *IEEE/ACM Transactions on Networking (TON)*, 9(4):427–438, August 2001.

[HL99] Zygmunt J. Haas and Ben Liang. Ad Hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking (TON)*, 7(2):228–240, April 1999.

[Hsi01] Pai-Hsiang Hsiao. Geographical Region Summary Service for Geographical Routing. *Mobile Computing and Communications Review*, 5(4):25–39, October 2001.

[IEE99] ANSI/IEEE. *International Standard ISO/IEC 8802-11*, 1999 edition, 1999. http://www.ieee802.org/11/.

[JM96] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

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

[KFHM02] Michael Käsemann, Holger Füßler, Hannes Hartenstein, and Martin Mauve. A Reactive Location Service for Mobile Ad Hoc Networks. Technical Report TR-02-014, Department of Computer Science, University of Mannheim, November 2002.

[KHFM02a] Michael Käsemann, Hannes Hartenstein, Holger Füßler, and Martin Mauve. A Simulation Study of a Location Service for Position-

Based Routing in Mobile Ad Hoc Networks. Technical Report TR-07-002, Department of Computer Science, University of Mannheim, 2002.

[KHFM02b] Michael Käsemann, Hannes Hartenstein, Holger Füßler, and Martin 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.

[KK00] Brad 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.

[Käs03] Michael Käsemann. Beaconless position-based routing for mobile ad-hoc networks. Master's thesis, Universität Mannheim, 2003.

[KV98] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the fourth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '98)*, pages 66–75, Dallas, Texas, October 1998.

[LJD$^+$00] Jinyang Li, John Jannotti, Douglas S. J. DeCouto, David R. Karger, and Robert 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.

[MWH01] Martin Mauve, Jörg Widmer, and Hannes Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network*, 15(6):30–39, November/December 2001.

[NI97] Julio C. Navas and Tomasz Imielinski. GeoCast – Geographic Addressing and Routing. In *Proceedings of the third annual*

ACM/IEEE International Conference on Mobile computing and networking (MobiCom '97), pages 66–76, Budapest, Hungary, September 1997.

[nis] National institue of standards and technology. http://www.nist.gov.

[NS2] The ns-2 network simulator. http://www.isi.edu/nsnam/ns/.

[OTBL01] Richard G. Ogier, Fred L. Templin, Bhargav Bellur, and Mark G. Lewis. Topology Broadcast Based on Reverse-Path Forwarding (TBRPF). Internet Draft, draft-ietf-manet-tbrpf-03.txt, work in progress, November 2001.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV). In *Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM '94)*, London, United Kingdom, August 1994.

[PC97] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1997)*, pages 1405–1413, Kobe, Japan, April 1997.

[PI4] Lehrstuhl für Praktische Informatik IV, Universität Mannheim. http://www.informatik.uni-mannheim.de/informatik/pi4.

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, New Orleans, LA, February 1999.

[RKY$^+$02] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott 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.

[RT99] Elizabeth M. Royer and Chai-Keong Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[Sto99] Ivan Stojmenovic. Home agent based location update and destination search schemes in ad hoc wireless networks. Technical Report TR-99-10, Computer Science, SITE, University of Ottawa, September 1999.

[Tou80] Godfried Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

Index

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mannheim, den 27.März 2003 Wolfgang Kieß