Ants-Based Routing in Mobile Ad-Hoc Networks

Diplomarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von:
David Jörg

2004

Leiter der Arbeit:
Prof. Dr. Torsten Braun

Forschungsgruppe Rechnernetze und Verteilte Systeme (RVS)
Institut für Informatik und angewandte Mathematik

Abstract

A Mobile Ad-Hoc Network (MANET) is a collection of wireless mobile nodes forming a temporary network without using centralized access points, infrastructure, or centralized administration. To establish a data transmission between two nodes, typically multiple hops are required due to the limited transmission range.

In this paper we present the Ants-Based Mobile Routing Architecture (AMRA), a novel two-layered approach, which combines position-based routing, topology abstraction and swarm-based intelligence. We make use of the Mobile Ants-Based Routing algorithm (MABR) to construct an abstracted view of the network and to take routing decisions on top of the abstracted topology. Packet forwarding at the lower layer is performed by a straight packet forwarding protocol. We have targeted AMRA to large mobile ad-hoc networks with irregular node distribution. The QualNet network simulation software was used in order to conduct a basic performance comparison and an evaluation of the newly developed protocol. Our findings attest the high potential of the AMRA architecture. Particularly, compared to GPSR, it achieves to substantially reduce both hop count and trip time in a complex network topology.

Acknowledgments

This work has been carried out at the research group of Computer Networks and Distributed Systems (RVS), Institute of Computer Science and Applied Mathematics (IAM), University of Bern, 2004.

First of all, I owe my gratitude to Professor Dr. Torsten Braun who allowed me to realize my diploma in his research group. Many thanks go to Marc Heissenbüttel who assisted me throughout this thesis, introducing me to the subject area, providing me with continuous support, and many useful suggestions.

I wish to thank all my friends and colleagues who proposed so many valuable hints and contributed to the success of this thesis. I am specially thankful to Emanuel Huber for the many hours he spent on correcting and proof-reading my thesis. Thanks also go to the guys at QualNet support who quickly attended my questions and helped solving the one or other problem.

I am deeply grateful to my family for their love and support. Specially I would like to thank to my parents, Urs and Elsbeth Jörg, for supporting my education, their financial support making my studies actually possible.

Finally, I wish to express my loving thanks to my girlfriend Therese Huber.

Bern, December 2004
David Jörg

Contents

List of Figures

1 Introduction

In the last couple of years, the use of wireless mobile networks has strongly evolved, mainly through the increasing popularity of the internet and computer networks in general. To people it becomes more and more important to be flexible, independent from wires and workspace, but to be connected to the network, nevertheless. Different protocols have been developed and proposed for such purposes, but many of them still rely on cabled infrastructure such as wireless base stations, for instance. In those networks the routing between sender and receiver is not accomplished by the wireless network but still by the cabled components.

These days, the focus of research is being set on new technologies and routing protocols, which no longer require base stations, fixed routers, or any other infrastructure: Mobile Ad-Hoc Networks. In a mobile ad-hoc network all nodes may move randomly and are connecting dynamically to each other. This new type of network largely widens the operational area of computer networks. They may be used in areas with little or no communication infrastructure: think of emergency searches, rescue operations, places where people wish to quickly share information, like meetings, or avoidance of rear end collisions, if cars participate in such a network.

Even though, a handful of protocols for mobile ad-hoc networks already exist, only few of them are really usable in larger computer networks. Those protocols, such as Ad-Hoc On-Demand Distance Vector (AODV) Routing for instance, often rely on flooding a route request packet through the network, as soon as a node is willing to transmit data. The flooding is continued until the destination has been reached, an intermediate node knows a valid route to the destination, or until every node in the network has received the request. Upon reaching the destination the node is sending a route reply packet backwards the same way the route request came from. Every intermediate node is now setting up a forward route entry in its routing table and therefore establishes the desired route. For instance, other variations of protocols use caching, source routing, or regularly broadcast control packets in order to update the network topology. All these protocols are working better or worse in smaller networks. However, there are heavy scalability issues. As the network size grows, or the mobility of the nodes becomes higher, the usable bandwidth of the network strongly diminishes under the increasing load of the signaling traffic. This not only leads to congestion, but also causes long delays and finally high packet loss. Furthermore, in some protocols all nodes need to maintain a routing entry for every other node in the network – large routing tables, high memory usage and deterioration of the performance are the results.

Hence it is not astonishing that recently several proposals addressing the scalability problem have arised. One idea is to use geographical information for the routing process, supposing a node knows its own and the recipient's location. The own location can easily be determined by the use of the Global Positioning System (GPS), if a GPS receiver is embedded into the computer. For the position of the destination an appropriate location service is queried.

In Greedy Perimeter Stateless Routing (GPSR) for example, broadcasting of hello packets is only performed to detect the local neighborship, but no control packets are being flooded through the whole network. When a data packet needs to be routed, it is being sent to the neighboring node closest to the destination. Even though the scalability of this protocol is greatly improved, holes in the network topology (such as lakes or mountains for example) may cause problems and long unneeded detours of packets. Also, link capacity, link quality and delay are not taken into consideration in this approach.

In this document, we investigate the performance of a new approach: a protocol based on swarm intelligence and geographical routing. Artificial ants (mobile network agents) are being transmitted through the network to measure the current network state, to update the routing tables depending on the collected information, and to deposit an artificial pheromone on the route taken. We also introduce an abstraction from the dynamic, irregular topology of a mobile ad-hoc network to obtain a topology with logical routers and logical links. In the first part of chapter 2, a general introduction to mobile ad-hoc networks is presented. In the second part, two geographical routing protocols are introduced, including Greedy Perimeter Stateless Routing and the Terminode Routing Protocol. Chapter 3 provides a detailed explanation of swarm intelligence based routing and the Ants-Based Mobile Routing Architecture (AMRA), which is the main topic of this thesis. We follow up our explanation with a presentation on the network simulation software QualNet (chapter 4), and a detailed description of the AMRA implementation for this specific simulator (chapter 5). The simulation setup for different scenarios is presented in chapter 6, while chapter 7 discusses the various results. Not only the different metrics are being analyzed, but also a performance comparison between different configurations is being presented. The final chapter summarizes our findings with a number of conclusions based on the simulation results while also giving an outlook on possible future work.

2 Mobile Ad-Hoc Networks

As alluded to in the introduction, a *Mobile Ad-Hoc Network (MANET)* consists of a random collection of wireless mobile devices that cannot rely on centralized or organized connectivity. The range of MANETs varies from small, rather static topologies, to large, mobile and highly dynamic networks. To establish a data transmission between two nodes multiple hops are typically required due to the limited transmission range of a single node. Therefore, the network is required to perform self-configuration by means of the cooperation of mobile devices: all nodes operate as routers and need to be capable of discovering and maintaining routes, and to propagate packets accordingly. The movement of mobile nodes requires the employment of quite complex routing algorithms, as routes are not stable and need to be updated continuously.

This chapter describes several concepts concerning the operation of that kind of routing protocols. The first part explains three well established traditional MANET routing protocols (AODV, DSR, ZRP), the latter part presents newer aspects of MANET routing introducing position-based routing protocols (GPSR, Terminode Routing).

2.1 Traditional Routing Protocols for Mobile Ad-Hoc Networks

Routing protocols for mobile ad-hoc networks are classified into three different groups: *proactive*, *reactive* and *hybrid* protocols. The first protocols developed derive from static networks and require periodic advertisement and global dissemination of connectivity information for correct operation, which leads to frequent system-wide broadcasts. These so called proactive protocols are *table driven*, which indicates that they maintain a routing entry for every possible destination in the network. In Destination-Sequenced Distance Vector Routing (DSDV)[1], for instance, every mobile node in the network holds a routing table where it lists all possible destinations and the corresponding hop counts to them. The protocol transmits full dump and incremental control packets in order to update the route information in the entire network. This potentially large amount of network traffic strongly limits the use of this protocol to small ad-hoc networks.

A reactive protocol establishes a route only when needed, that is, if a node is willing to transmit data and is not aware of a route to the destination. Most often, reactive protocols rely on the transmission of route request and route reply messages, which are needed to establish and maintain the routes. Routes are only recorded as long as they are valid and expire after some idle time. On-demand route establishment leads to a drastic reduction of control traffic required for routing. At AODV and DSR we present two examples of reactive routing protocols.

Finally, hybrid protocols combine proactive and reactive aspects. Mostly, proactive mechanisms are used in the local neighborhood of a node to establish routes within a limited radius. Thus, broadcasting of routes through the entire network is avoided. Routing between distant nodes is still performed by on-demand routing. Hybrid routing is illustrated by means of the ZRP protocol.

[1]A detailed description of DSDV can be found in [1].

2.1.1 Ad-Hoc On-Demand Distance Vector Routing

The Ad-Hoc On-Demand Distance Vector routing protocol (AODV) described in [2] is an improvement of the Destination-Sequenced Distance Vector routing protocol (DSDV). The advantage of AODV is that it tries to minimize the number of required broadcasts. It creates the routes on a on-demand basis, as opposed to maintain a complete list of routes for each destination. Therefore, the authors of AODV are classifying it as a *pure on-demand route acquisition system* [3].

Path Discovery Process

When trying to send a message to a destination node without knowing an active route[2] to it, the sending node initiates a path discovery process. A *route request message (RREQ)* is broadcasted to all neighbors, which continue to broadcast the message to their neighbors and so on. The forwarding process is continued until the destination node is reached or until an intermediate node knows a 'fresh enough' route to the destination (see figure 2.1(a)).

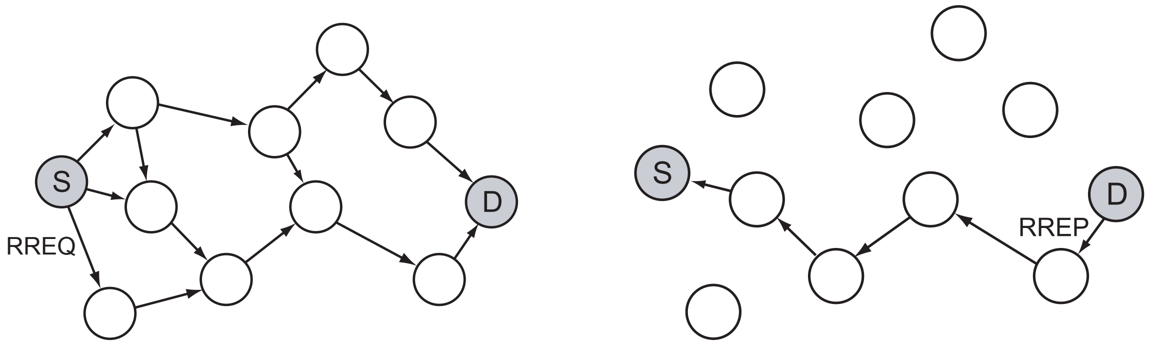
To ensure loop-free and most recent route information every node maintains two counters: *sequence number* and *broadcast_id*. The broadcast_id and the address of the source node uniquely identify a RREQ message. broadcast_id is incremented for every RREQ initiated by the source node. An intermediate node can receive multiple copies of the same route request broadcast from various neighbors. In this case – if a node has already received a RREQ with the same source address and broadcast_id – it will discard the packet without broadcasting it furthermore. When an intermediate node forwards the RREQ message, it records the address of the neighbor, which it received the first copy of the broadcast packet from. This way, the *reverse path* from all nodes back to the source is being built automatically. The RREQ packet contains two sequence numbers: the source sequence number and the last destination sequence number known by the source. The source sequence number is used to maintain 'freshness' information about the reverse route to the source while the destination sequence number specifies the actuality a route to the destination must have before being accepted by the source.

As soon as the route request broadcast reaches the destination or an intermediate node with a fresh enough route, the node responds by sending a unicast *route reply packet (RREP)* back to the node which it received the RREQ from (figure 2.1(b)). Actually, the packet is sent back reverse the path that has been built during broadcast forwarding. A route is considered fresh enough, if the intermediate node's route to the destination node has a destination sequence number, which is equal or greater than the one contained in the RREQ packet. As the RREP is sent back to the source, every intermediate node along this path adds a forward route entry to its routing table. The forward route is set active for some time indicated by a route timer entry. The default value is 3000 milliseconds, as referred in the AODV RFC. If a route is no longer used it is deleted after expiry of the route timer. Since the RREP packet is always sent back the reverse path established by the routing request, AODV only supports symmetric links.

Maintaining Routes

If the source node moves, it is able to send a new RREQ packet in order to find a new route to the destination. If an intermediate node along the forward path moves, its upstream neighbor notices the move and sends a *link failure notification* message to each of its active upstream neighbors to inform

[2]A route is considered active if it has an entry in the routing table that is marked as valid. Active routes expire after a certain amount of time or on occurrence of a link failure. Only active routes can be used to forward data packets.



(a) Source node S initiates the path discovery process. (b) A RREP packet is sent back to the source.

Figure 2.1: AODV Path Discovery Process.

them of the erasure of that part of the route. The link failure notification is forwarded as long as the source node has not been reached. After having noticed the failure the source node may reinitiate the route discovery protocol. Optionally a mobile node may perform local connectivity maintenance by periodically broadcasting hello messages[3].

2.1.2 Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol [4] is an on-demand routing protocol based on source routing. In the source routing technique a sender determines the exact sequence of nodes, which to propagate a packet through. The list of intermediate nodes for routing is explicitly contained in the packet's header.

In DSR every mobile node in the network needs to maintain a *route cache* where it keeps source routes that it has learned. When wanting to send a packet to some other host, the initiating host first checks its route cache for a source route to the destination. In case of finding a route the sender propagates the packet that way. Otherwise the source node initiates the route discovery process. Route discovery and route maintenance are the two major parts of the DSR protocol.

Route Discovery

For route discovery the source node starts by broadcasting a route request packet that may be received by all neighbor nodes within its wireless transmission range. The route request contains the address of the destination host, referred to as the *target* of the route discovery, the source's address, a *route record* field and a unique identification number. At the end, the source host receives a route reply packet containing a list of network nodes, which it should propagate the packets through, provided that the route discovery process was successful. A simple example is illustrated in figure 2.2.

During the route discovery the route record field is used to accumulate the sequence of hops already taken. First of all, the sender initiates the route record as a list with a single element containing itself. The next neighbor node appends itself to the list and so on. Each route request packet also contains a unique identification number called *request_id*. request_id is a simple counter, which is increased whenever a new route request packet is being sent by the source node. Thus, every route request packet can be uniquely identified through its initiator's address and request_id. When a host receives

[3]An AODV Hello message is defined as a RREP packet with a time to live (TTL) field of 1. Furthermore the node may send a unicast RREP or an ICMP Echo Request message to the next hop for local route maintenance.

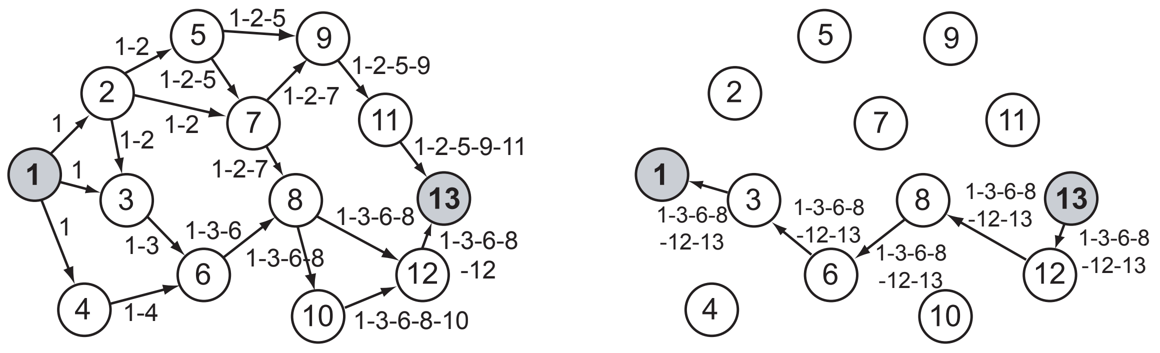
a route request packet, it is important to process the request in the order described below. This way we make sure no loops will occur during the broadcasting of the packets.

1. If the pair ⟨ source node address, request_id ⟩ is found in the list of recent route requests, the packet is discarded.

2. If the host's address is already listed in the request's route record, the packet is also discarded. This ensures removal of later copies of the same request that may arrive by using a loop.

3. If the destination address in the route request matches the host's address, the route record field contains the entire list of nodes, which have been passed through in order to reach the destination from the source node. A *route reply* packet is sent back to the source node containing a copy of this route.

4. Otherwise, add this host's address to the route record field of the route request packet and re-broadcast the packet.



(a) Building of the Route Record. (b) Propagation of the Route Reply.

Figure 2.2: DSR Route Discovery Process.

A route reply is sent back either if the request packet reaches the destination node itself or if the request reaches an intermediate node, which has an active route[4] to the destination in its route cache. The route record field in the request packet indicates the sequence of hops taken. If the generating node for the route reply is the destination node, it only takes the route record field of the route request and puts it into the route reply. If the responding node is an intermediate node, it appends the cached route to the route record and then generates the route reply.

Sending back route replies can be accomplished in two different manners: DSR may use symmetric links, but it is not required to. In the case of symmetric links the node generating the route reply simply uses the reverse route of the route record. When using unidirectional (asymmetric) links the node needs to initiate its own route discovery process and piggyback the route reply on the new route request.

Route Maintenance

Route maintenance can be accomplished by two different processes:

[4]An active route towards a destination has a route cache table entry that is marked as valid. Only active routes can be used to forward data packets.

- Hop-by-hop acknowledgment at the data link layer
- End-to-end acknowledgments

Hop-by-hop acknowledgment at the data link layer allows an early detection and retransmission of lost or corrupt packets. If the data link layer determines a fatal transmission error (for example, because the maximum number of retransmissions is exceeded), a *route error* packet is being sent back to the sender of the packet. The route error packet contains two parts of information: The address of the node detecting the error and the host's address, which it was trying to transmit the packet to. Whenever a node receives a route error packet, the hop in error is removed from the route cache and all routes containing this hop are truncated at that point.

End-to-end acknowledgment may be used, if wireless transmission between two hosts does not work equally well in both directions. As long as a route exists, by which the two end hosts are able to communicate, route maintenance is possible. There may be different routes in both directions. In this case, replies or acknowledgments on the application or transport layer may be used to indicate the status of the route from one host to the other. However, with end-to-end acknowledgment it is not possible to find out the hop, which has been in error.

2.1.3 Zone Routing Protocol

In a mobile ad-hoc network it can be assumed that most of the communication takes place between nodes close to each other. The Zone Routing Protocol (ZRP) described in [5] takes advantage of this fact and divides the entire network into overlapping zones of variable size. It uses proactive protocols for finding zone neighbors (instantly sending *hello* messages) as well as reactive protocols for routing purposes between different zones (a route is only established if needed). Each node may define its own zone size, whereby the zone size is defined as number of hops to the zone perimeter. For instance, the zone size may depend on signal strength, available power, reliability of different nodes etc. While ZRP is not a very distinct protocol, it provides a framework for other protocols.

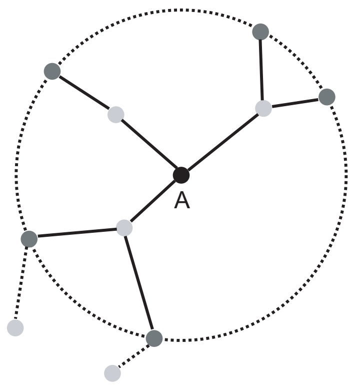


Figure 2.3: ZRP - Routing Zone of Node A, $\rho = 2$.

First of all, a node needs to discover its neighborhood in order to be able to build a zone and to determine the perimeter nodes. In figure 2.3, all perimeter nodes are printed in dark gray color – they build the border of A's zone with radius $\rho = 2$. The detection process is usually accomplished by using the *Neighbor Discovery Protocol* (NDP). Every node periodically sends some *hello* messages to its neighbors. If it receives an answer, a point-to-point connection to this node exists. Nodes may be selected by different criteria, be it signal strength, radio frequency, delay etc. The discovery messages are repeated from time to time to keep the map of the neighbors updated.

The routing processes inside a zone are performed by the *Intrazone Routing Protocol* (IARP). This protocol is responsible for determining the routes to the peripheral nodes of a zone. It is generally a proactive protocol. An other type of protocol is used for the communication between different zones. It is called *Interzone Routing Protocol* (IERP) and is only responsible for routing between peripheral zones. A third protocol, the *Bordercast Resolution Protocol* (BRP) is used to optimize the routing process between perimeter nodes. Thus, it is not necessary to flood all peripheral nodes what makes queries become more efficient. Below, the three protocols are described in more detail.

Intrazone Routing Protocol

The IARP protocol is used by a node to communicate with the other interior nodes of its zone. An important goal is to support unidirectional links, but not only symmetric links. It occurs very often, that a node A may send data to a node B, but node B cannot reach node A due to interference or low transmission power for example. IARP is limited to the size of the zone ρ. The periodically broadcasted route discovery packets will be initialized with a Time To Live (TTL) field set to $\rho - 1$. Every node, which forwards the packet will now decrease this field by one until the perimeter is reached. In this case, the TTL field is 0 and the packet will be discarded. This makes sure that an IARP route request will never be forwarded out of a node's zone.

As already mentioned, IARP is a proactive, table-driven protocol for the local neighborhood may change rapidly, and changes in the local topology are likely to have a bigger impact on a nodes routing behavior than a change on the other end of the network [6]. Proactive, table-driven routing delivers a fast, efficient search of routes to local hosts. Local routes are immediately available. Therefore, every node periodically needs to update the routing information inside the zone. Additionally, local route optimization is performed. This includes the following actions:

- Removal of redundant routes
- Shortening of routes, if a node can be reached with a smaller number of hops
- Detecting of link failures and bypassing them trough multiple local hops

Interzone Routing Protocol

The Interzone Routing Protocol is used to communicate between nodes of different zones. It is a reactive protocol and the route discovery process is only initiated on demand. This makes route finding slower, but the delay can be minimized by use of the Bordercast Resolution Protocol. IERP takes advantage of the fact that IARP knows the local configuration of a zone. So a query is not submitted to all local nodes, but only to a node's peripheral nodes. Furthermore, a node does not send a query back to the nodes the request came from, even if they are peripheral nodes.

Bordercast Resolution Protocol

The Bordercast Resolution Protocol is rather a packet delivery service than a full featured routing protocol. It is used to send routing requests generated by IERP directly to peripheral nodes to increase efficiency. BRP takes advantage of the local map from IARP and creates a bordercast tree of it. The BRP employs special query control mechanisms to steer route requests away from areas of the network that have already been covered by the query. The use of this concept makes it much faster than flooding packets from node to node.

A detailed description of the Bordercast Resolution Protocol, including its implementation is described in [7].

2.2 Position-Based Routing in Mobile Ad-Hoc Networks

Traditional MANET protocols all suffer from flooding certain control packets through the network in order to establish or maintain routes. This behavior consumes considerable bandwidth, causes packet delay and packet drops due to congested queues. Moreover, those protocols are hardly scalable for the routing overhead increases extremely with expanding network size, especially in high mobility scenarios.

A novel approach, known as *position-based* or *geographic routing protocols*, makes use of node positions to get rid of the packet flooding mechanisms. The idea is based on forwarding data packets in direction of the destination's location, wherever applicable, i.e. provided that a node obtained its own and the recipient's location. The own position is determined by an embedded low power GPS receiver, an appropriate location service is required to learn the destination's position. Proposals for location services are described in various papers: Distance Routing Effect Algorithm for Mobility [8], Quorum-Based Location Service [9], Grid Location Service [10, 11, 12], and Homezone [13, 14].

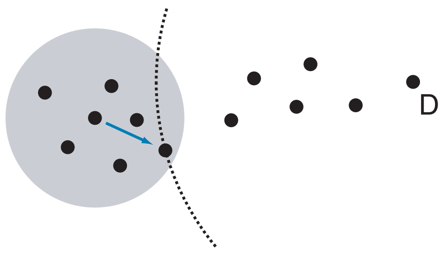
In the following we are describing two position-based routing protocols – Greedy Perimeter Stateless Routing (GPSR) [15] and Terminode Routing [16] – both used in this project to conduct performance comparisons.

2.2.1 Greedy Perimeter Stateless Routing

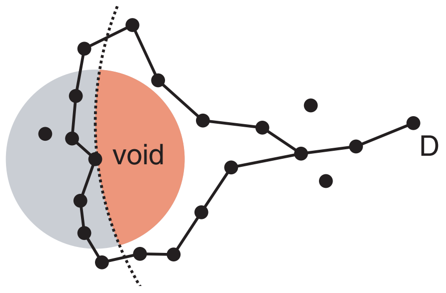
The Greedy Perimeter Stateless Routing (GPSR) algorithm, also known as Greedy-Face-Greedy (GFG), distinguishes two methods of forwarding packets: *greedy forwarding* is used whenever possible, *perimeter forwarding* refers to some 'backup mode forwarding', which is employed when greedy forwarding fails.

GPSR assumes all packets are marked with the destination's coordinates, what usually is performed by the originator. So, if an intermediate node receives a packet while being aware of its neighbors in the transmission range, it can determine the neighbor closest to the destination and forward the packet accordingly. Similarly, the forwarding is continued, the packet following trails towards closer geographic regions until the destination node is reached. Figure 2.4(a) gives an example of this so called greedy forwarding process. The gray shaded circle represents the transmission range of the intermediate node relaying the packet. The dotted circle around the destination illustrates that no closer node can be found in the transmission range. Therefore, the packet is being forwarded as shown by the blue arrow.

A simple beaconing algorithm allows a node to locate its neighbors: at regular time intervals every node initiates a small beacon to the broadcast address containing its own IP address and position. All beacons are tagged with a TTL of one to ensure that they are dropped at the neighboring nodes. Every node maintains a neighbor table and updates the neighbors' positions, or it adds new neighbors upon reception of a beacon. If no more beacons are arriving from a neighbor for some timeout period, the neighbor is considered dead or having left transmission range, and therefore is deleted from the neighbor table. Even though the beaconing is performed only locally, it still represents proactive routing, costing network bandwidth and battery. In order to reduce those costs the transmitting node's position is also piggybacked on each data packet. Furthermore, all network interfaces run in promiscuous mode to get notice of all packets sent within the transmission range. Thus, if a node transmits a data packet, the neighboring nodes can extract the position information from that packet. Therefore, the transmitting node can reset the beacon interval.



(a) In greedy forwarding, a packet is relayed to the neighbor closest to the destination.



(b) Greedy forwarding failure: no neighbor in transmission range is closer to the destination.

Figure 2.4: Greedy Forwarding in GPSR.

Although greedy forwarding works very well in most cases, it may fail under certain topology conditions, that is, if a *local maximum* is encountered. As illustrated in figure 2.4(b), this occurs whenever a packet arrives at a node but the node does not have a closer neighbor to the destination. At a hole in the geographic distribution of the nodes, such as a lake for instance, relaying the packet closer to the destination is not applicable. In this case, GPSR tries to deliver the packet using perimeter mode, a method using planarized graphs and the right-hand rule.

A graph is called *planar* if there are no edges crossing each other. Wireless network planar subgraphs are utilized to route around the perimeter of a hole until greedy forwarding can be continued. In this document, we present two well-known different planar subgraphs already used in different subject areas: the *Relative Neighborhood Graph (RNG)* and *Gabriel Graph (GG)* [17, 18].

In order to establish a planar RNG or GG graph, edges need to be removed from the initial network graph without disconnecting the graph. The removal of too many edges is leading to a segmentation of the network. The definition and construction of the RNG and GG graphs is depicted in figure 2.5 and performed as follows:

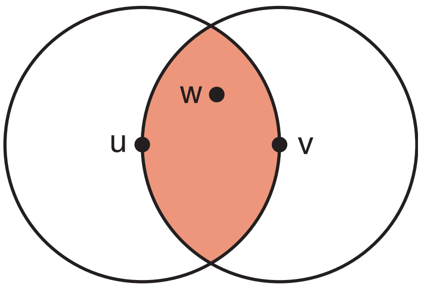
RNG Provided two given nodes u and v and an edge (u, v) of the initial network graph. (u, v) is retained if its distance, $d(u, v)$, is less than or equal to the distance between every other vertex w, and whichever of u and v is farther from w. In figure 2.5(a), the red zone must be empty of any vertices w for (u, v) to be included in the RNG graph. A mathematical representation is indicated below:

$$\forall w \neq u, v : d(u, v) \leq max[d(u, w), d(v, w)]$$

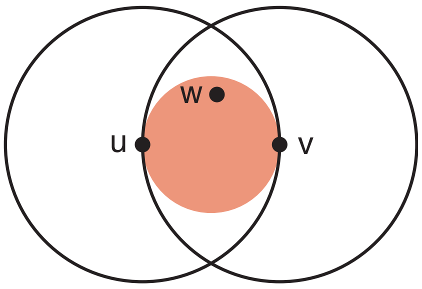
GG The definition of the gabriel graph shown in figure 2.5(b) is very similar: an edge (u, v) is included in the gabriel graph provided there is no other vertex w within the red circle with diameter is $\overline{uv}$, or:

$$\forall w \neq u, v : d^2(u, v) < [d^2(u, w) + d^2(v, w)]$$

Once the relative neighborhood or gabriel graph is constructed, we make use the *right-hand rule* for traversing, what we illustrate in figure 2.6(a). Assuming a packet arriving at node x from node y, the next edge traversed is the next one counterclockwise about x from edge (x, y). Node z is reached.



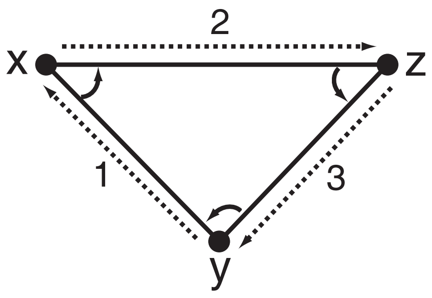
(a) The RNG graph. For edge (u, v) to be included in the planar graph, the red zone must not contain any nodes w.



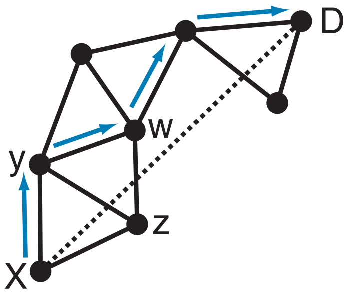
(b) The GG graph. For edge (u, v) to be included in the planar graph, the red circle must not contain any nodes w.

Figure 2.5: Relative Neighborhood Graph (RNG) and Gabriel Graph (GG).

Again, in order to reach our next edge we search counterclockwise from (x, z), and so on. The interior of the triangle in the example is traversed in clockwise order, that is ($y \rightarrow x \rightarrow z \rightarrow y$). Generally, the right-hand rule always traverses the interior of a closed polygonal region (a *face*) in clockwise order.



(a) Right-hand rule example.



(b) Perimeter mode forwarding example.

Figure 2.6: Right-hand Rule and Perimeter Forwarding.

Bringing all together, perimeter forwarding is summarized in figure 2.6(b). If a packet enters perimeter mode at node X towards destination D, GPSR forwards it on progressively closer faces of the planar graph, each of which is crossed by the line $\overline{XD}$. The right-hand rule is applied on each face in order to reach an edge that crosses line $\overline{XD}$. After encountering that edge, the packet moves to the adjacent face crossed by $\overline{XD}$. In our example, X determines y as the next node by the right-hand rule ((X, y) is the first edge counterclockwise from $\overline{XD}$). y borders the edge (y, z) that intersects $\overline{XD}$. There, packet forwarding occurs along the next face bordering the edge (y, z), which according to the right-hand rule is (y, w). The process is continuing that same way until destination D is reached.

When entering perimeter mode, the node location at that point, L_p, is recorded in the GPSR packet header. Perimeter node forwarding is continued until locating a node closer to the destination than L_p. As soon as a node is found, the packet is returned to greedy mode.

In a static network, GPSR ensures correct delivery of packets from any source to an arbitrary destination. However, in mobile ad-hoc networks, especially under high mobility, the algorithm is not safe from loops. Loops may happen during perimeter mode forwarding, as specified in the following example (see figure 2.7).

We presume a packet in perimeter mode already when it arrives at node 1. Recall that the packet follows the current face in clockwise order. Thus, node 2 (neighbors 1, 3 and 4) relays the packet to node 3 according to the right-hand rule. Subsequently, node 3, whose neighbors are 2 and 4, delivers the packet to node 4, again because of the right-hand rule (see figure 2.7(a)). Note that node 5 was not yet appearing in the neighbor table for whatever reason (dead or too far away), but is about to arrive and to announce its presence. So, node 4, with neighbors 3, 6, and recently node 5, computes a planar graph and applies the right-hand rule again, hence selecting node 5 (figure 2.7(b)). Finally, node 5, which now contains 2 and 4 in its neighbor table, forwards to node 2, node 2 to node 3, and so on. If node 5 stops moving, the packet will endlessly loop until the TTL expires.

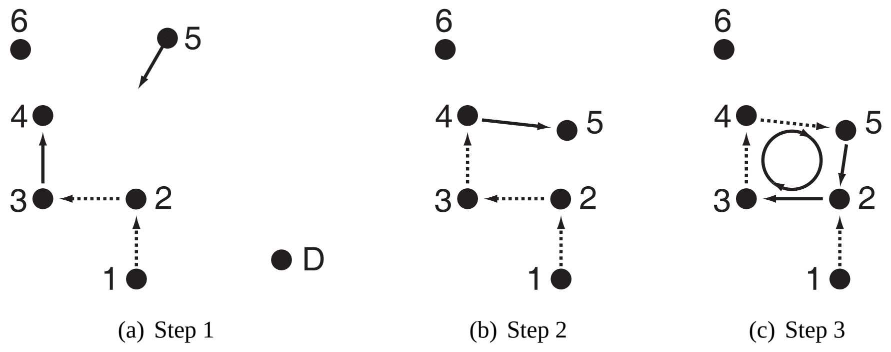


Figure 2.7: Example of loops that may show up in perimeter mode under mobility.

The initial planar face contains edges $((1,2),(2,3),(3,4))$. The arrival of node 5 causes the topology to alter and the creation of a new face. This face now consists of nodes $(2,3,4,5)$ and does not intersect the line between the position where the packet entered into the perimeter mode and the destination. The packet loops, trapped in the newly created face. The looping continues either until the face reopens due to topology changes, or until it moves to cross the line to the destination. The packet therefore enters a new face.

GPSR was initially proposed under a different name: Greedy-Face-Greedy (GFG) algorithm [19]. The later enhancement, GPSR, suggests to make use of the IEEE 802.11 medium access control feedback. The algorithm was also tested in a mobile network [15]. In the GPSR approach, MAC layer failure feedback to the routing protocol helps to figure out unreachable nodes before the neighbor dead interval expires. Thus, if a packet exceeds its maximum number of retransmit retries, the particular node entry is immediately deleted from the neighbor table.

2.2.2 Terminode Routing Protocol

The *Terminode Routing Protocol* described in [16, 20, 21] bears its name from mobile nodes acting as network nodes and terminals at the same time. It takes inspiration from the Zone Routing Protocol and combines two different algorithms for local and remote routing: the proactive *Terminode Local Routing (TLR)* and the reactive *Terminode Remote Routing (TRR)* respectively. Terminode Routing is a position-based routing protocol. It requires a GPS receiver embedded in every mobile node and presumes the availability of a location service. Every node is assigned with a *location-dependent address (LDA)*, which specifies a triplet of longitude, latitude and altitude, and a fixed *End-system Unique Identifier (EUI)*.

Terminode Local Routing

Similar to the Intrazone Routing Protocol in ZRP (see section 2.1.3), the Terminode Local Routing (TLR) proactively maintains tables about nodes within the local neighborship. A destination D is called *TLR-reachable*, if an arbitrary node X is able to reach it by means of the TLR protocol, signifying that D is located in the vicinity of X. The TLR-reachable area, also referred to as 'local radius', is specified in number of hops from the current node. The paper indicated suggests considering two hop neighborship for TLR (radius $\rho = 2$).

Locations are updated by virtue of periodically broadcasted HELLO messages, which contain EUI and LDA of the announcing node as well as the EUIs of all its one-hop neighbors. Based on the information received, every node builds its routing table of the two-hop neighbors and uses a simple two-hop link-state routing protocol to deliver packets to the TLR-reachable area by use of the EUI. Note that the LDA is not required by local routing decisions.

Terminode Remote Routing

The basics of TRR's functions are somewhat comparable to Greedy Perimeter Stateless Routing. It makes use of a default greedy packet forwarding method, called *Geodesic Packet Forwarding (GPF)*, which always tries to deliver packets closer toward the destination, roughly in a straight line (see figure 2.4(a)). The LDA of the destination is recorded in the packet header and serves as a geographic reference point where the packet is relayed to. As soon as a node identifies the destination to be TLR-reachable, it sets the TLR-bit in the packet header and switches to TLR routing mode. It is to be considered that a packet handed to TLR never can be reverted to TRR. It rather would be dropped. If due to a hole in the network topology greedy forwarding fails, GPF uses perimeter mode forwarding to route the packet around the problematic area (see section 2.2.1). Like GPSR, GPF is not robust against loops in perimeter mode what results in long and complex paths if node distributions contain big gaps. Therefore, if greedy forwarding is not possible, TRR utilizes the *Anchored Geodesic Packet Forwarding (AGPF)* algorithm.

The AGPF forwarding method is about a form of geographic source routing along *anchored paths*. A path consists of multiple fixed geographic points, called *anchors*. At initiation of a data packet the originator determines the sequence of anchors to follow. It attaches them to the packet header. The discovery of an anchored path is summarized later. Figure 2.8 gives an example of AGPF packet delivery. It takes place as follows: Originator S initiates a packet for destination node D and defines anchors A_1 and A_2 for the path to be taken. S forwards the packet applying GPF greedy forwarding in direction to the first anchor point. Every intermediate node checks if A_1 geographically lies in its transmission range. If so, A_1 is removed from the stack of anchors in the packet header. The packet then is directed to A_2. The process continues until all anchors have been removed, and the packet finally is addressed to the location of the destination. Again, GPF is utilized until the packet can be handed to TLR.

Restricted Local Flooding

Due to inaccurate location information provided by the location service, long queuing times, or high mobility, the destination node may be situated outside the vicinity of LDA_D at arriving of the packet. Thus, the destination never is TLR-reachable. As a consequence, the packet circles around the LDA_D location until the TTL field expires, wasting resources and locally congesting the network. In order to cope with this effect, an intermediate node tries to inhibit those undesired loops by checking if the

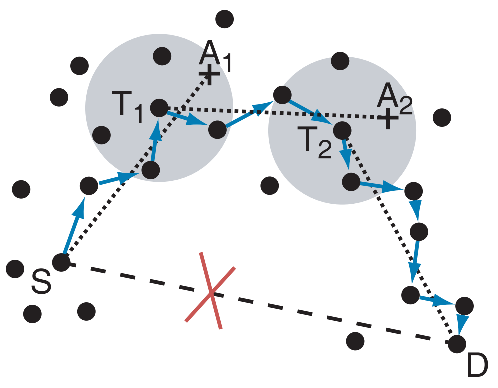


Figure 2.8: AGPF Packet Forwarding in Terminode Routing

expected destination location LDA_D falls into its transmission range, however the destination node is not TLR-reachable.

In this case, the article introduces two possible approaches: (a) limiting lifetime of packets to a value of three ($TTL_{new} = min(3, TTL)$), or (b) using the *Restricted Local Flooding (RLF)* method. RLF is a limited local flooding technique targeted on finding the destination around the estimated LDA_D location. Therefore, six copies of the original packet are flooded to six different geographic regions around the sending node, as depicted in figure 2.9. The LDA field of every packet is assigned with the appropriate new destination location (that is $X_1 \ldots X_6$), yet the EUI remains unchanged. The flooding process is restricted by limiting the search to two transmission ranges, and by setting a small TTL value of four for instance. If the destination remains non-TLR-reachable, the packet is discarded.

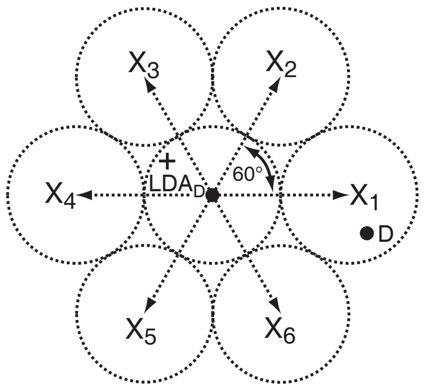


Figure 2.9: Restricted Local Flooding (RLF) Algorithm

Anchored Path Discovery

Two different path discovery methods are used at the source nodes to establish an anchored path towards the destination: *Friend Assisted Path Discovery (FAPD)*, or *Geographic Maps-based Path Discovery (GMPD)*. FAPD relies on the concept of small world graphs [22]. Every node maintains a list of other ‘friend’ terminodes, which it maintains paths to. Further, a node queries its friends in order to find a anchored path to the destination. Thus, a high cooperation between nodes is required for FAPD correctly operating.

GMPD requires a geographical view of the network to be at disposal for every terminode. It assumes the presence of areas with high node density, which are referred to as towns, and interconnections between these towns. Every mobile node maintains a list of existing towns and interconnections and thus is able to create anchored paths in accordance to the network topology. Besides the summarized network topology map, no cooperation of the nodes is needed for the establishment of anchored paths. A detailed explanation of FAPD and GMPD is presented in [20].

2.3 Discussion

Traditional routing algorithms for mobile ad-hoc networks all suffer from limited scalability. Scalability is a very important factor, as it determines if a protocol will function or fail when the number of mobile users increases. The process of route request flooding generates a considerable routing overhead, which accumulates and congests networks of larger size. Routes also break more often with increasing mobility. Thus, the source node is required to reinitiate the path discovery process causing even higher network load. DSR manages to reduce the overhead by aggressive route caching. However, this behavior effects the often use of stale routes, which leads to frequent packet retransmission and high delay times. ZRP bordercasts route requests only between distant zones. Yet, quite often, a new path discovery process must be performed due to frequent route failures in high mobility scenarios.

Position-based routing protocols achieve a substantial reduction of the routing overhead by the use of additional location information. Route requests are no longer needed since the approximate direction of the destination node can be determined. GPSR only broadcasts control packets locally in order to detect the neighboring nodes. The disadvantage of this protocol is the lack of local routing information. The algorithms is completely memoryless and does not profit from previous route experience. Thus, although better routes to the destination exist, no path improvement is achieved over time. This is reflected in the high hop counts and euclidean distances when GPSR encounters the perimeter mode. The Terminode Remote Routing protocol still relies on route requests to discover paths to distant friends. The GMPD path discovery approach is not realistic for it requires knowledge of the network topology. Usually this is not the case.

Our new proposal described later in chapter 3.3 deals with both problems: it reduces the routing overhead (a) by the use of location information, (b) by transmitting unicast network exploration agents whose frequency may be varied according to the network state, (c) by memorization of experienced trip times and hop counts for subsequent routing decisions, and (d) by utilization of normal data packets to assess the conditions within the network.

3 Swarm Intelligence Based Routing

The idea of ant algorithms derives from real swarms of certain insects, which develop a form of ‘swarm intelligence’ and solve complex problems through cooperation. One autonomous ant itself obeys primitive instincts and only performs limited, specialized tasks. However, the colony at large shows a global intelligent behavior.

This chapter explains the function of ant algorithms and the porting to computer networks at a very basic level, followed by a few sections about important other work done in this subject area, both for wired and non-wired networks. Finally, we give a detailed characterization of the algorithm used for this project. The facts about the implementation are discussed later in chapter 5.

3.1 Basic Ant Algorithm

Ants utilize a specialized form of communication, which is called *stigmergy* [23] in biology and means indirect communication through the environment. Every ant constantly deposes pheromone on the trail taken, a messenger substance related to hormones, which other ants are able to sense as they move along their path. Ants are attracted by the pheromone and tend to follow trails with higher pheromone concentrations. This causes a self-accelerated reaction, more and more ants following the same trail laying even more pheromone. This autocatalytic effect has many advantages, for it allows path optimization within the colony for instance.

Suppose two different trails between nest and food in the colony, as illustrated in figure 3.1. Two ants leaving the nest at the same time and looking for food autonomously decide, which trail to take at the intersection. More, we assume that both ants take a different path at the beginning and reach the food after certain time. This means, that the ant taking the detour reaches the food as well as the first intersection later when moving back to the nest. So, the double amount of pheromone is on the shorter path, influencing subsequent ants to primarily follow this path and not to take the detour.

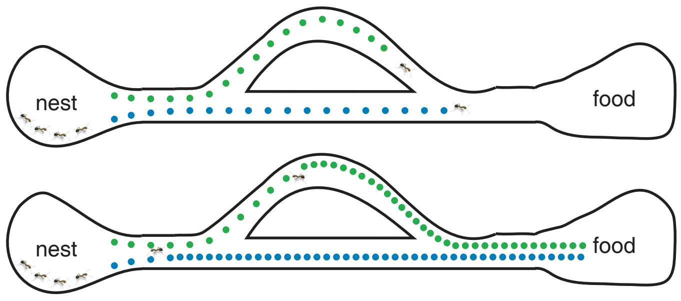


Figure 3.1: Ants take the shortest path between nest and food.

The adaption of ant algorithms to computer networks behaves as follows: virtual ants, so called *mobile routing agents*, are transmitted across the network and independently measure the current network state, such as packet delay or hop count for instance. The stored information is then evaluated by the receiving node and used to lay an artificial amount of pheromone to reinforce that specific link. The pheromones are often expressed by a routing table holding probability information for every outgoing link. Since real pheromones evaporate after certain time, also some decay function is required at the nodes holding the routing tables.

An approach based on swarm intelligence is very promising compared to other routing protocols, as it shows many advantages in terms of scalability, fault tolerance, adaptiveness, autonomy and parallelism. The number of agents transmitted can easily be adjusted to the network size. Agents can also be killed or reproduced dynamically based on the state of the network. In addition, new links, link breaks and congestions can rapidly be detected and new paths be reinforced by the autocatalytic effect described before. Finally, all agents operate independently, autonomous and also parallel. Therefore, the loss of one agent doesn't affect the network function nor needs the flow of agents to be controlled by any human.

3.2 Related Work

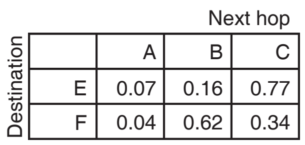
Already a number of swarm-based routing protocols exist, some proposed for fixed, wired networks (AntNet [24], ABC [25]), other approaches also address mobile ad-hoc networks (ARA [26], Termite [27]). An outline of the most established protocols is depicted in the following sections.

3.2.1 AntNet - Distributed Stigmergetic Control for Communications Networks

AntNet is an algorithm conceived for fixed, wired networks, which derives features from parallel replicated Monte Carlo systems [28], previous work on artificial ant colonies techniques [29, 30, 31, 32], and telephone network routing [25, 33]. The idea in AntNet is to use two different network exploration agents (*forward* and *backward ants*), which collect information about delay, congestion status and the followed path in the network. Forward ants are emitted at regular time intervals from each node to a randomly selected destination. This transmission occurs asynchronously and concurrently with the data traffic. As soon as a forward ant arrives at the destination, a backward ant moves back to the source node reverse the path taken by the forward ant. The subdivision in forward and backward ants has the following reasons: Forward ants are just employed for data aggregation of trip times and node numbers of the path taken without performing any routing table updates at the nodes. The backward ants get their information from the forward ants and use it to achieve routing updates at the nodes.

Every node in the network maintains two structures, which the agents cooperate with and concurrently read and write to:

Routing Table The routing table includes one line for each known destination in the network and one column for each outgoing link. For example, a node with three neighbors in a 50 node network has a table with 49 rows and three columns. Thus, for each destination D and for each neighbor N, a value P_{dn} is stored, which expresses the probability of taking this link for destination D. Therefore, the probabilities of one row sum to 1. An example of an AntNet routing table is presented in figure 3.2.



		Next hop		
Destination		A	B	C
	E	0.07	0.16	0.77
	F	0.04	0.62	0.34

Figure 3.2: AntNet Routing Table.

Forward ants and data packets use these probabilities in different ways in order to reach the destination: forward ants determine the next hop randomly among their neighbors, while also considering their probabilities registered in the routing table. In our example, seven percent of the forward ants will be transmitted to node A, 16 percent to node B, and 77 percent to node C in order to reach destination E. Remember that the backward ants simply follow the opposite path from the forward ants. In contrary, data packets always choose the link with the highest probability for the next hop.

Local Traffic Statistics This array holds statistic data for the state of the network, as experienced by the local node. It stores mean values and variances based on the agents' measured delays, and also a moving observation window of trip times used to estimate the best trip time value. Best results for mean and variance were observed by the following exponential model, where $o_{k \to d}$ represents the new determined delay from node K to destination D.

$$\begin{aligned} \mu_d &= \mu_d + \eta(o_{k \to d} - \mu_d) \\ \sigma_d^2 &= \sigma_d^2 + \eta((o_{k \to d} - \mu_d)^2 - \sigma_d^2) \end{aligned}$$

The last step is to update the routing table when a backward ant arrives. The actions applied include the *positive reinforcement* (increment) of probability P_{df} (the probability of choosing neighbor F, where the forward ant initially arrived from) and the decrement of all other neighbors' probabilities P_{dn}. For this purpose, a reinforcement value r is calculated for the new trip time under consideration of the current routing table and the local traffic statistics. $r \in (0, 1]$ is a dimensionless measure for the goodness of the newly experienced delay. For instance, a poor delay has different meanings in a low traffic or a heavy loaded network. There are several proposals of calculating such goodness values described in the AntNet paper [24]. The computation of this value is left out, as it is not required for the understanding of AntNet's operation and also shows to be mathematically complex.

The value P_{df} is increased and the probabilities P_{dn} decreased as follows so that the sum of all probabilities will remain 1.

$$\begin{aligned} P_{df} &= P_{df} + r(1 - P_{df}) \\ P_{dn} &= P_{dn} - rP_{dn} \end{aligned}$$

One should note that every discovered path receives a positive reinforcement regardless of the trip time noticed. Therefore, not only the computed value r is relevant for the reinforcement algorithm, but also the ant arrival rate.

The performance evaluations were conducted using a completely new implementation of a discrete event simulator in C++. The comparison with 6 state-of-the-art routing protocols in various test networks showed a superior performance in almost any scenario. Remarkably new in our approach are the local structures containing the current network state to better assess the trip times. Previous algorithms only used the information carried by the ants to achieve the reinforcement.

3.2.2 ABC - Ant-Based Control

Ant-Based Control is another stigmergy-based ant algorithm designed for telephone networks. It shares many similarities with AntNet, but also incorporates certain differences. The basic principle relies on mobile routing agents, which randomly explore the network and update the routing tables according to the current network state. The routing table, storing probabilities instead of pheromone concentrations, is exactly the same as in AntNet. Also probability balanced randomness of the ants' path selection is employed in order to favor the detection of new routes. One important difference applies to the use of the routing agents: ABC only uses a single class of ants (i.e. forward ants), which are initiated at regular time intervals from every source to a randomly chosen destination. After arriving at a node they immediately update the routing table entries for their source node, meaning that the pheromone pointing to the previous node is increased. It is important to see that only the backward path is influenced, and just packets traveling towards the ant's source profit from that route update. The second difference pertains to the calculation of the routing table update. The probabilities are reinforced by a simple formula without considering any goodness value or something like that. Therefore, no local traffic statistics are required to be recorded at the nodes. In the probability update equations below, Δp represents the probability increase, P_{df} respectively P_{dn} the probabilities for the positive/negative reinforced links:

$$\begin{aligned} P_{df} &= \frac{P_{df} + \Delta p}{1 + \Delta p} \\ P_{dn} &= \frac{P_{dn}}{1 + \Delta p} \end{aligned}$$

Two new aspects introduced in the ABC paper include the aging and delaying of ants. Thus, the authors intend to primarily encourage the ants to find short paths, and to secondly prevent the agents from visiting heavily congested nodes. The first goal is accomplished by reducing the Δp value, which reinforces the routing table, gradually over time. It is assumed that the age of the ants corresponds to the number of hops taken. Therefore, agents transmitted along shorter trails have a higher impact on the routing updates. The second objective is obtained by adding an artificial delay to the ants when passing congested nodes. This additional delay is related to the degree of congestion and tries to normalize the situation by the following effects:

- Delaying the ant forwarding from a congested node leads to a reduced flow rate of agents to its neighbors, which thereby prevents the ants from positively reinforcing the path towards the congested node. This allows other ants arriving from different nodes to find alternative routes around the obstacle and to reinforce them.
- The additional delay also increases the age of the ants when they arrive at the neighboring node. Therefore, the Δp value turns out smaller and has less impact on the updated probability value.

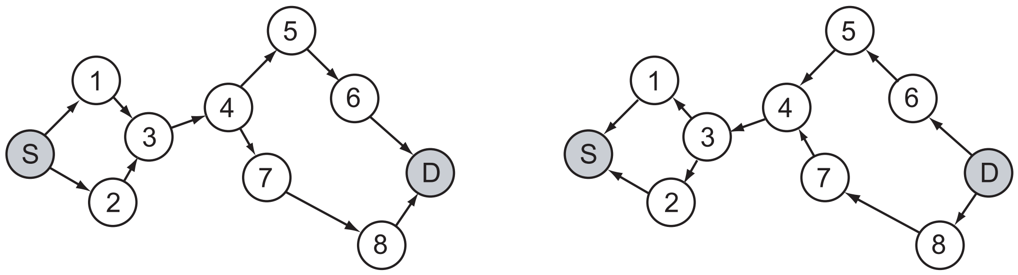
The equations below indicate a proposal for the aging and delay calculation, as described in the ABC paper. s denotes to the spare capacity of the current node. The delay is expressed in discrete time steps.

$$\Delta p = \frac{0.08}{age} + 0.005 \qquad delay = \lfloor 80 \cdot e^{-0.075 \cdot s} \rfloor$$

3.2.3 ARA - The Ant-Colony Based Routing Algorithm for MANETs

In contrary to the algorithms for cabled networks presented before *ARA* proposes a detailed swarm intelligence based routing scheme for Mobile Ad-Hoc Networks. It uses both forward and backward ants to explore new trails in the network. However, unlike the algorithms presented before the network agents are only transmitted on demand and are flooded through the entire network in a similar process as AODV. Also, the routing table does not contain probabilities; it stores pheromone concentrations, which are transformed into probabilities in a second step. The succeeding three stages are distinguished in the paper:

Route Discovery Phase The route discovery phase is suitable for the discovery of new routes. It is initiated by the sender, as soon as it is willing to transmit data to some destination. A small forward ant packet is flooded to all nodes in the network, until the destination is reached. Duplicate packets are identified by the use of a sequence number and are removed by the system. The destination node gathers the information from the forward ant and starts to send a backward ant to the source. The backward ant not only follows the path with the shortest trip time, but every path detected by the forward ant, as illustrated in figure 3.3. The data transmission can take place from the reception of the backward ant by the source. As an improvement to the algorithm, delivery of forward and backward ants to priorized queues is recommended.



(a) A forward ant is flooded from the source to the destination.

(b) The backward ant takes the reverse path of the forward ant.

Figure 3.3: Route Discovery Phase in ARA.

Route Maintenance The route maintenance phase is responsible for updating the routing information during the communication and starts as soon as the data transmission takes place. This is accomplished by the utilization of regular data packets, which lay down a fixed amount of pheromone and therefore reinforce the corresponding path. The algorithm updates both the forward and the backward path, which is explained in the example below (see figure 3.4). Assume a packet being sent from node 3 to node 4, bound for destination D. Thus, node 3 updates the forward path towards the destination (i.e. node 4) with an additional amount $\Delta\varphi$ of pheromone; node 4 likewise updates the backward path towards the source (i.e. node 3) with the same amount. In the ARA paper $\Delta\varphi = 0.1$ is presumed.

Destination	Next hop	Pheromone
S	1	φ_1
S	2	φ_2
D	4	$\varphi_3+\Delta\varphi$

(a) Routing table of node 3.

Destination	Next hop	Pheromone
S	3	$\varphi_4+\Delta\varphi$
D	5	φ_5
D	7	φ_6

(b) Routing table of node 4.

Figure 3.4: ARA routing table after packet transition from node 3 to 4.

The ARA proposal also allows for the evaporation of pheromones. Every second the pheromone concentration is decreased by a multiplicative factor.

$$\varphi_{i,j} = (1-q)\cdot\varphi_{i,j} \qquad q \in (0,1]$$

ARA uses a probabilistic routing model for the data packets, which means that packets are distributed among the neighbors according to the probabilities for the corresponding links. In fact, probabilities do not differ from the pheromone concentrations, as they are computed proportionally. In the formula below, N_i denotes to the neighboring nodes.

$$p_{i,j} = \frac{\varphi_{i,j}}{\sum_{k \in N_i} \varphi_{i,k}}, \qquad j \in N_i, \sum_{k \in N_i} p_{i,k} = 1$$

Route maintenance carried out in the manner described is not safe from loops. ARA achieves loop freedom by making use of the sequence numbers already mentioned. If a node receives a duplicate packet, it sets the `DUPLICATE_ERROR` flag and returns the packet to the previous node, which in turn erases the link to the node reporting the loop.

Route Failure Handling The ARA implementation in the paper assumes the presence of an IEEE 802.11 conforming MAC layer. This allows ARA to detect a link failure by the missing acknowledgment on the MAC layer and to deactivate that link by resetting the pheromone concentration to 0. Then, the routing table is checked for different links towards the destination and the packet gets relayed accordingly. If no further route exists the current node notifies the sender about the route failure, which thereby initiates a new route discovery process.

3.2.4 Termite - Emergent Ad-Hoc Networking

As the name implies, the *Termite* routing algorithm is modeled on the behavior of real termites building a termite hill. Like ants, termites act independently and communicate indirectly through the environment by the use of pheromones they are attracted to. In the beginning of the construction phase, termites move randomly, picking up a sand grain and laying it down at a random location. If they encounter another grain, the carried pebble is deposited at the same location and assigned with a small amount of pheromone. If enough pebbles are accumulated and one random small hill reaches a critical mass of pheromone, almost all termites are attracted to that heap and continue building their nest. The process is induced by negative and positive feedback: pheromones evaporate quickly from small piles, and larger piles are awarded with additional termites, which thereby further increase the pheromone concentration.

Even though, Termite uses probabilistic routing, the routing tables store pheromone concentrations, which are transformed into probabilities later. The approach presented in the paper proposes a fixed additive pheromone increase and an exponential decrease for the packet source, both within boundaries to avoid extreme pheromone differences.

$$\begin{aligned} \varphi_{dn} &= \varphi_{dn} + \Delta\varphi \\ \varphi_{dn} &= \varphi_{dn} \cdot e^{-\tau} \end{aligned}$$

The following values have been selected in the paper: $\Delta\varphi = 1$, $\tau = 0.105$, *decay period* $=$ 1 second, *initial pheromone* $= 1$, *pheromone ceiling* $= 10000$, *pheromone floor* $= 0.1$. If a packet arrives from an unknown source, a row is created in the routing table for the new destination. If this newly discovered node happens to be neighbor, in addition to the row also an additional column is appended. The specified initial pheromone value is assigned to the new entries. The concentration must not fall below the pheromone floor, even if no traffic is sent by the source for some time. This procedure helps to detect idle nodes easily.

If a packets needs to be routed, it is sent randomly based on pheromone values pointing to the neighboring nodes. Regardless of the concentration, packets are never relayed back to the last hop. To obtain probabilities for the routing decision, the transformation function below is applied to the pheromone entries. The constants K and F are used to tune the routing decision of Termite. K affects the sensitivity of probabilities to small pheromone concentrations, F amplifies or attenuates pheromone differences.

$$p_{dn} = \frac{(\varphi_{dn} + K)^F}{\sum_{i=1}^{N}(\varphi_{di} + K)^F}, \qquad K = 0, F = 2$$

The Termite routing scheme suggests four different protocol packets. A *route request* is initiated as soon as a node is not aware of the path to the destination. Ignoring the pheromones, the route request follows a random path through the network until a node is reached containing pheromone for the destination (i.e. no further move to the destination is performed). Optionally, multiple route requests may be triggered to speed up the path discovery. A *route reply* packet confirms the new route to the initiator of the request. The sender address of the route reply is spoofed, i.e. the packet appears to arrive from the searched destination. The delivery of the route reply is achieved through the usual probabilistic routing process. *Hello packets* are suitable to explore the local neighborhood if a node becomes isolated. As soon as a neighbor hears and replies to the hello the node will stop to broadcast immediately. Finally, *seed packets* proactively distribute a node's pheromone throughout the network. They may be useful when a node newly enters the network and wants to announce its presence. Similar to route requests, the information is spread by a random walk.

3.2.5 Discussion

AntNet and ABC both show robust and reliable performance in smaller, fixed wired networks. Their development was not intended for the use in large and highly dynamic mobile ad-hoc networks. ARA and Termite were tested in small-sized mobile ad-hoc networks where they operate very well. However, the route discovery phase still relies on packet flooding. ARA makes use of an AODV-like approach in order to find a valid path to the destination. In Termite, the route request ant follows a random walk through the network until a node is containing pheromone for the destination. The latter scheme reduces the routing overhead, although it may lead to long and absurd paths. The scalability of all protocols described is strongly limited by the ant flooding mechanisms. Furthermore, all protocols store a routing entry for every single destination node in the network. This causes a heavy increase of routing table size in large mobile ad-hoc networks.

The AMRA routing architecture presented in the section below tries to improve scalability as follows:

- Location information is utilized to avoid the flooding of ants. Ant packets are simply transmitted in the direction of the destination node. Planar graph traversal methods similar to GPSR/GFG are employed to route ants around problem areas.
- Data packets are marked with additional information about the network state to reduce the necessity of additional ants.
- The network area is grouped into zones of different size. The edge length of the zones increase with the distance from the actual node. A routing table entry is not maintained for every sole node, but only for each zone. Thus, the memory requirement of the routing table can be drastically reduced.

3.3 Ants-Based Mobile Routing Architecture

The routing protocol we implemented in this project is unique for it tries to combine geographic routing, topology abstraction and swarm intelligence in a two-layered approach. The algorithm, which we explain below principally follows the description in [34], however with certain changes and extensions.

We assume that nodes situated close to each other essentially share the same routing information. Further, mobility changes in the local neighborhood have a larger impact on the routing decision than a node move at the other end of the network. The *Topology Abstraction* pays attention to those two circumstances and introduces a simplified network topology by the use of *logical*[1] *routers* and *logical links*. A logical router represents a cluster of mobile nodes situated close together, principally holding the same routing information. The interconnection between logical routers is supposed to be an approximately straight line over possibly multiple hops and is referred to as logical link. The ants-based routing by *MABR* (Mobile Ants-Based Routing) is now achieved on top of this abstract topology. As a whole, we employ two different routing protocols to attain our goal: MABR determines the next logical router for a destination by means of pheromone concentrations. The forwarding process along the logical links is accomplished by a straight packet forwarding protocol (StPF). One point to note is that the two protocols operate at different layers: StPF is not aware of the abstraction performed. It just forwards the packets based on the geographical coordinates of the next logical router, which it received by MABR. In our approach, Greedy Perimeter Stateless Routing (GPSR) is employed for straight packet forwarding (see section 2.2.1). An overview of the architecture is depicted in figure 3.5. The red line indicates a link with high delays, which causes MABR to favor the upper logical links.

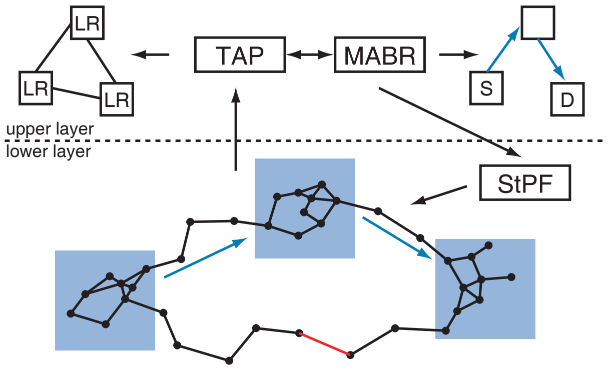


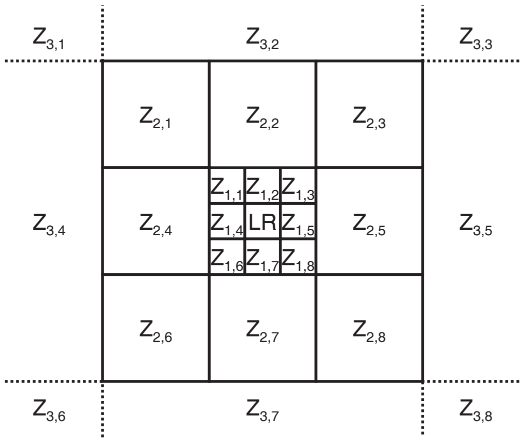
Figure 3.5: Overview of the AMRA Architecture.

3.3.1 Topology Abstraction

In order to obtain the abstraction briefly described, the plane is divided into geographical areas of equal size with all nodes within the area belonging to the same logical router. For simplicity reasons, we just use squares in this project, unlike in cellular networks where often regular hexagons are utilized. Logical routers are consistently arranged covering the entire area, each assigned with an unique ID that can be converted into coordinates. Thus, depending on the current location, every node is part of one specific logical router. If it crosses the borders of the current router, it automatically becomes a member of the new logical router.

[1]The term ‘logical’ is used to express paths, links, etc. in the upper layer.

Additionally, every logical router groups other logical routers into different zones $Z_{i,j}$, as shown in figure 3.6(a). The zone size varies with the current distance from the center router, i.e. farther zones are of substantial larger size. This reflects the importance of node moves near or far from the current node for the routing task. A node move of 500 meters in the local neighborhood may turn out in an entirely different routing direction. In contrast, the same motion of a node far away only marginally affects the direction. In figure 3.6(a) the innermost ring of zones $Z_{1,j}$ is composed of only one logical router each. In the next ring of zones $Z_{2,j}$, yet nine logical routers are bundled whereas $Z_{3,j}$ clusters 81 logical routers. It is important to notice that the view of zones is always relative. Hence every logical router resides in the center of his own zone model.



(a) Zones relative to specific logical router.

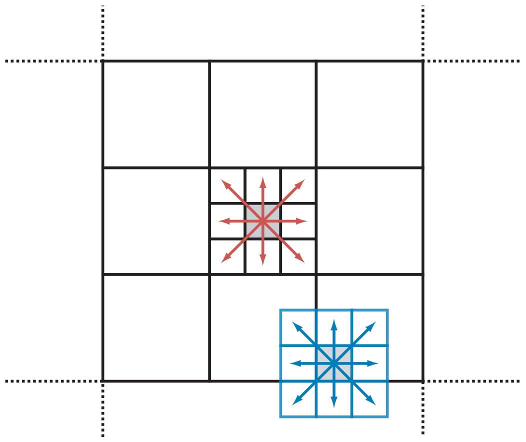

(b) Logical links of two different logical routers.

Figure 3.6: Logical Routers and Logical Links in the AMRA Architecture.

Logical links are established between the center logical router and adjacent zones. Every logical router maintains eight logical links. This is different from the approach proposed in [34], where logical links can be situated between the center logical router and any arbitrary zone. Thereby we expect a smaller routing table (only eight outgoing logical links) and a more consistent routing: packets can follow a straight line more easily when using shorter zone hops. Otherwise, when employing long logical links, greedy forwarding might not be possible. Figure 3.6(b) illustrates the logical links for two different logical routers and also points out the relative view of each logical router.

The second distinction from the paper denoted addresses the routing and link-cost tables. The article suggests redundant, distributed tables, which all nodes within the same logical router are sharing. Considering the large routing overhead for synchronizing and replicating those tables, we favor a passive method for the route and link-cost table update. In our approach, every mobile node maintains its own tables initialized with start values in the beginning. The nodes operate in promiscuous mode to overhear all packets sent and received within the transmission range. Therefore, they are able to update pheromone concentrations and experienced network state based on the packets of neighboring nodes as well. With an adequate selection of the logical router size (e.g. two times the transmission range), we expect homogeneous routing and link-cost tables at all nodes within a logical router.

3.3.2 Mobile Ants-Based Routing

In the following, we describe the two data structures maintained by every node:

Routing Table The routing table is set up on top of the abstracted topology. Unlike rows of node entries used in traditional routing tables, rows of zone entries are recorded for every known destination zone. Columns of zones are added for every outgoing logical link. Every destination zone owns eight logical links for each adjacent zone. We also call them *relay zones*. Since we use a probabilistic routing model, the pheromones are directly transformed into probabilities, which we store in the routing table. When encountering packets from an unknown destination zone, an additional row is added to the table and the probabilities for all eight outgoing logical links initialized with 12.5%. In the following a small extract of a possible MABR routing table is represented.

Destination	Relay zone / Logical link							
	$Z_{1,1}$	$Z_{1,2}$	$Z_{1,3}$	$Z_{1,4}$	$Z_{1,5}$	$Z_{1,6}$	$Z_{1,7}$	$Z_{1,8}$
$Z_{1,5}$	0%	0%	0.8%	0%	99%	0%	0%	0.2%
$Z_{3,2}$	61%	37%	2%	0%	0%	0%	0%	0%
$Z_{2,8}$	0%	0%	0%	0%	9%	0%	0%	91%

Link-Cost Table The link-cost table retains information about link quality from different destination zones. For instance, mean values and variance of end-to-end delay and hop count can be measured. When a packet arrives, this allows to compute some goodness value, which may be used for stronger or weaker reinforcement of the relay zone.

Besides the data traffic additional control packets are implemented to explore new paths in the network and to reinforce the links with pheromone. Forward ants are transmitted from a mobile node to some randomly chosen logical router in the network, either regularly or based on activity. In the second case nodes perpetually listen for forward ants initiated by other nodes within the same logical router. A forward ant is only initiated, if no packet from different nodes within that router was listened to during the last time interval. Forward ants store information such as packet type, sender coordinates, address of the destination logical router, coordinates of the last hop in order to detect logical router changes, coordinates of the last logical router to reinforce the correct logical link, a TTL value, a sequence number to uniquely identify packets, a timestamp about the initiation of the packet, just to enumerate the most important ones. These values are required for routing and updating the two tables. Forward ants are routed by use of the GPSR algorithm, disregarding the pheromones. If perimeter mode is encountered, ants are transmitted by the right- or left-hand rule randomly, depending on the choice of the initiating node. On occurrence of a forward ant MABR updates the pheromone concentrations of the backward path towards the source at every intermediate node, which processes or overhears the packet.

If the forward ant arrives at the destination, it updates the routing table's probabilities and is dropped afterwards. The initiating node may optionally request a backward ant to update the forward path toward the destination. This may be of interest, if no packet transmission has been seen from that logical router for long time. A backward ant stores the same information as a forward ant but additionally includes the network delay experienced by the forward ant as well as the first logical router where the forward ant had been sent through. Backward ants follow the same path as data packets, which is described later. In this project we decided not to use backward ants, as we expect enough bidirectional traffic in the network. Forward ants sent from the opposite direction can effect the same behavior as backward ants do. However, the functionality of backward ants is basically implemented in the protocol.

Data packets serve a dual purpose: Primarily they carry the data from the source to the destination. Secondly we also use them as ‘data ants’ to maintain existing paths. Therefore, an additional header added behind the IP header stores relevant information for packet delivery and pheromone updates, such as packet type, original IP protocol, source address and coordinates, destination address and coordinates, sequence number, TTL value, last logical router address and position, and a timestamp of the sending time. In order to add that header and to ensure correct packet handling we need to adapt the IP header and change the IP protocol to MABR. The original packet will be reassembled at the destination node after updating the pheromones.

Data packets and backward ants are randomly transmitted to the next adjacent zone based on the probabilities marked in the routing table. Upon the selection of the next zone, the packet is handed to GPSR. It is transmitted toward a previously defined anchor point within the next zone. Depending on the route direction, this anchor always refers to the farthest point within the next zone relative to the local node (see figure 3.7). Therefore, the destination coordinates in the GPSR header need to be adapted whenever a new zone is selected. As soon as the first node situated in the new zone processes the packet, MABR determines the next relay zone again, and so on.

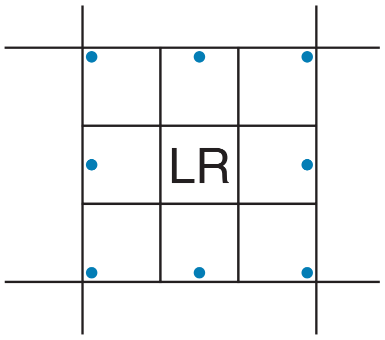
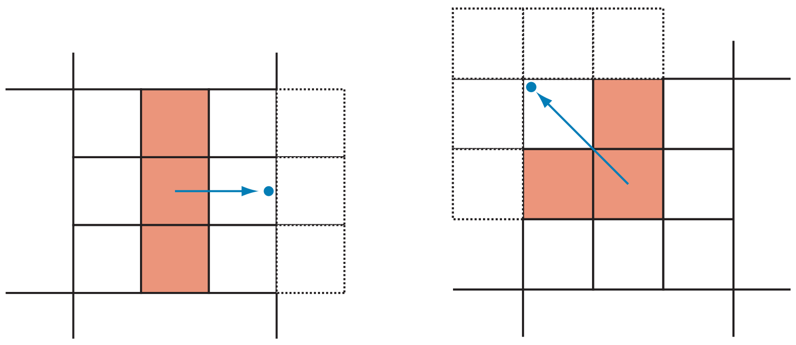


Figure 3.7: Packets are always addressed to the farthest point within the neighboring zone.

Furthermore, to avoid local loops and unreliable links, certain restrictions apply when determining the next zone hop. A data packet or backward ant may not be relayed to a zone whose probability falls below a defined threshold value. Also, three zones in direction of the provenience of the packet are locked for the next relay. Figure 3.8 illustrates two examples of zone locking. If a packet is relayed to zone $Z_{1,5}$ on the right for instance, zones $Z_{1,1}$, $Z_{1,4}$ and $Z_{1,6}$ on the left may not be used for the next relay. Among strange pheromone concentrations, it may happen that no suitable logical link can be found for packet relay. In this case GPSR forwarding toward the final destination’s coordinates is being used until a node provides a reasonable value for a next zone relay.

Supposing the transmission of a data packet toward zone $Z_{3,2}$, the following steps take place:

- The initiating node queries the location service, adds a MABR header to the data packet, and stores the destination coordinates into the recently added header.
- Based on the destination location learned the unique identifier of the destination zone ($Z_{3,2}$) is computed.
- The relay zone is determined according to the corresponding row in the routing table ($Z_{3,2}$, see page 26): The values of relay zones $Z_{1,3}$ to $Z_{1,8}$ fall below the predefined threshold of 20%. Therefore, these zones are not considered for routing. A concidence experiment now



(a) Packet forwarded to the right; three neighboring zones in the left are locked for the next zone relay.

(b) Packet forwarded to the upper left; three neighboring zones in the lower right are blocked.

Figure 3.8: Zone Locking in MABR when Relaying Data Packets.

determines the routing decision between relay zones $Z_{1,1}$ (61%) and $Z_{1,2}$ (37%), according to the route probabilities indicated. We suppose the selection of the logical link toward zone $Z_{1,2}$, although it is less probable.

- The current node determines the farthest point within zone $Z_{1,2}$, adds a GPSR header holding this anchor point, and hands the packet over to GPSR.

What finally remains is the positive and negative reinforcement of the links. A seen table is required for this purpose, which prevents packets from updating routing and link-cost tables several times. This might happen due to promiscuous listening when the same packet is not only heard once. Packets are uniquely identified by the sender address and sequence number and are held in the seen table for 30 seconds.

First of all, an update to the link-cost table is performed according to the new delay $o_{k\to d}$ experienced by the packet overheard or received. This is done in a similar way to the local traffic statistics in AntNet by using the formula below. η has a nominal value of 0.1, which specifies a moving window of about 50 data samples considered for the mean value and variance[2]. The same equations could be used for other metrics such as hop count as well.

$$\begin{aligned} \mu_d &= \mu_d + \eta(o_{k\to d} - \mu_d) \\ \sigma_d^2 &= \sigma_d^2 + \eta((o_{k\to d} - \mu_d)^2 - \sigma_d^2) \end{aligned}$$

Concerning the probability update in the routing table a goodness value is calculated by means of the network delay already experienced: the better the delay, the better the goodness value. A maximal goodness of 0.8 can be reached if the new delay undershoots one third of the mean value.

$$r' = \begin{cases} \frac{\mu_d}{3o_{k\to d}} & \text{if } \frac{\mu_d}{3o_{k\to d}} < 0.8 \\ 0.8 & \text{else} \end{cases}$$

[2]The weight of the t_i-th sample used to estimate the value of μ_d after j samplings, with $j > i$, is $\eta(1-\eta)^{j-i}$. In this way, for example, if $\eta = 0.1$, approximately only the latest 50 observations will really influence the estimate.

Using the computed goodness the probability P_{df} of the logical link where the packet arrived from is positively reinforced. All other logical links P_{dn} obtain a negative reinforcement. The formula is computed satisfying the condition that the sum of all probabilities for a destination zone remains 1.

$$\begin{aligned} r_+ &= (1 - P_{df}) \cdot (r')^2 \\ r_- &= P_{dn} \cdot (r')^2 \\ P_{df} &= P_{df} + r_+ \\ P_{dn} &= P_{dn} - r_- \end{aligned}$$

Another thing we want to model is pheromone evaporation, what is the same as an adjustment of probabilities towards 12.5%[3]. While moving a mobile node eventually reaches a different logical router whose members most likely are maintaining different probabilities. The direction to distant nodes will still be accurate but there may be substancial changes in the local neighborhood. Therefore, if a node changes the logical router, the probability balance function below is executed and recalculates all probabilites based on the distance from the center logical router. Bad links (probability below 12.5%) will benefit from a higher probability, good links will suffer from a degradation. $P_k^{i,j}$ refers to the probability of logical link k (that is relay zone $Z_{1,k}$) for destination zone $Z_{i,j}$.

$$P_k^{i,j} = P_k^{i,j} + \frac{0.125 - P_k^{i,j}}{3^i}$$

For destinations near the center logical router (for instance zone ring $i = 1$) the fraction returns a relatively large adjustment whereas the probability correction rapidly decreases for more distant zones.

Summarizing the functionality of the AMRA algorithm figure 3.9 depicts a simplified overview in form of a flow diagram. On the left hand side, the various events that may occur are indicated followed by the appropriate event handling. Technical details about the implementation are being discussed later in chapter 5.

3.3.3 Packets and Headers

The MABR protocol makes use of three different protocol packets, which are indicated by the MABR IP protocol number. Two bits inside the packet structures, the MABR header, respectively, are reserved to specify the type of packet. Thereby the MABR event dispatcher is able to identify the packet type and to hand it to the appropriate event handler for processing afterwards.

Forward ants are used to explore new paths in the network. They measure the current network state by means of trip times, hop count or euclidean distance traveled for instance. They are initiated toward randomly selected geographical locations, i.e. to random logical routers. Therefore, no particular final destination node can be specified in the IP header. In our implementation we chose to assign the IP address of the next hop to the IP header's destination field at both the source and every intermediate node.

Backward ants Backward ants serve the purpose to inform the originating node about the information collected by the forward ant. Since they carry additional data from the forward ants they

[3] 100% divided by eight outgoing links

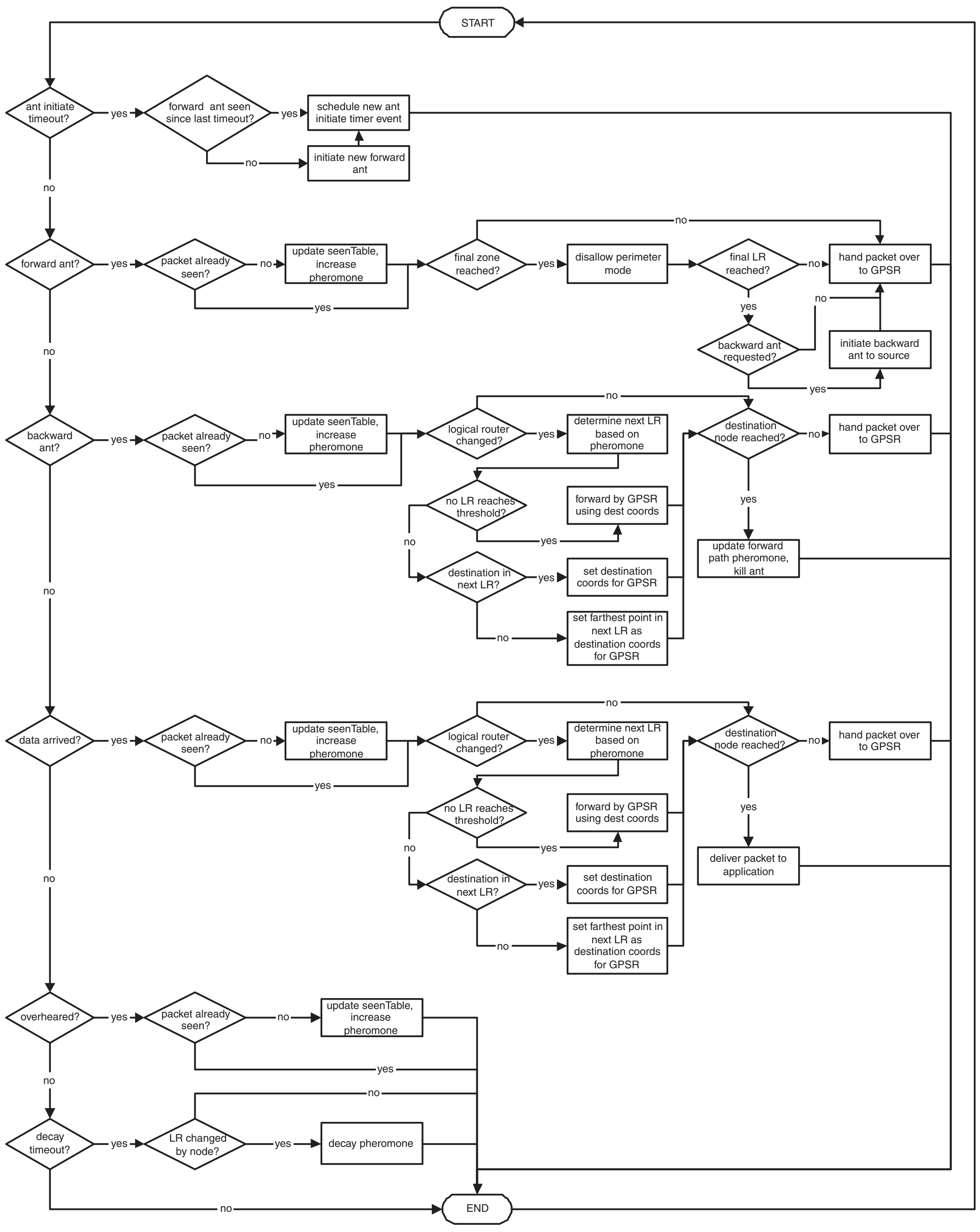

START
ant initiate timeout?
yes
forward ant seen since last timeout?
yes
schedule new ant initiate timer event
no
initiate new forward ant
no
forward ant?
yes
packet already seen?
no
update seenTable, increase pheromone
yes
final zone reached?
yes
disallow perimeter mode
final LR reached?
no
hand packet over to GPSR
yes
backward ant requested?
no
yes
initiate backward ant to source
no
backward ant?
yes
packet already seen?
no
update seenTable, increase pheromone
yes
logical router changed?
yes
determine next LR based on pheromone
no LR reaches threshold?
yes
forward by GPSR using dest coords
no
destination in next LR?
yes
set destination coords for GPSR
no
set farthest point in next LR as destination coords for GPSR
destination node reached?
no
hand packet over to GPSR
yes
update forward path pheromone, kill ant
no
data arrived?
yes
packet already seen?
no
update seenTable, increase pheromone
yes
logical router changed?
yes
determine next LR based on pheromone
no LR reaches threshold?
yes
forward by GPSR using dest coords
no
destination in next LR?
yes
set destination coords for GPSR
no
set farthest point in next LR as destination coords for GPSR
destination node reached?
no
hand packet over to GPSR
yes
deliver packet to application
no
overheard?
yes
packet already seen?
no
update seenTable, increase pheromone
yes
no
decay timeout?
yes
LR changed by node?
yes
decay pheromone
no
no
END

Figure 3.9: Simplified Flow Diagram of the AMRA Algorithm

are able to update both forward and backward paths at the same time. The destination node extracts the experienced trip time and hop count as well as the first logical router from the forward ant, and puts it into the backward ant. The first logical router determines the logical link used in order to transmit the forward ant, which needs to be reinforced. The backward ant acts as a common MABR control packet.

Data ants encapsulate data packets (e.g. TCP or UDP) sent across the network. As soon as the MABR router function encounters a new data packet, an additional MABR header needs to be added, and the packet must be transformed into a data ant. MABR adds its header behind the IP header, stores the original IP header protocol field into the new header, and pastes the MABR IP protocol number in the IP header. Values like the location of the source, the location of the final destination or the initiation timestamp are included in the new header. After having determined the next relay zone MABR adds a GPSR header between IP and MABR header. Among other entires, this header contains the coordinates of the next zone where the packet is determined to travel to. Further, the GPSR header stores the original IP protocol (now MABR) and stores the GPSR IP protocol number in the IP header. Subsequently, the packet is handed over to the GPSR routing protocol. Figure 3.10 shows the header fields of a data ant with its arrangement of IP, GPSR, MABR and UDP header. Having reached the destination, GPSR and MABR header are successively stripped away. The packet is then reassembled and finally becomes as a standard UDP packet.

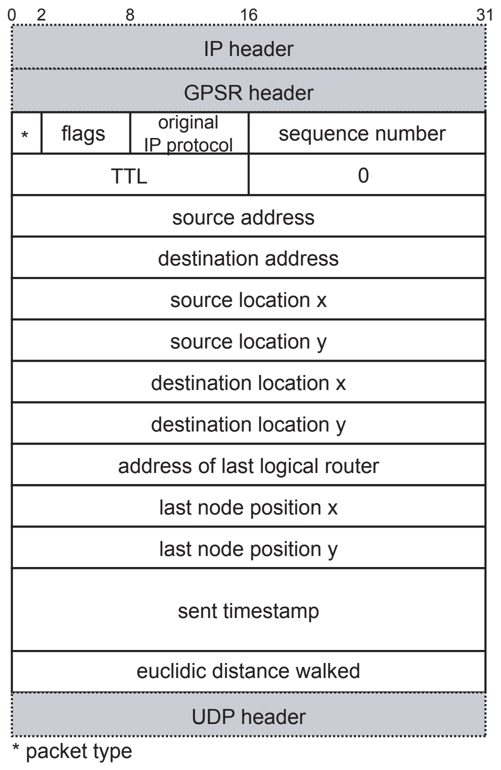


Figure 3.10: MABR Header for a UDP Data Packet.

A detailed specification of the MABR packet and header structures is given in appendix, page 73.

4 Network Simulation Software: QualNet

This chapter gives an idea of the network simulation software used in this project. Besides a general overview on QualNet, a in-depth description of routing protocol implementation, packets and event handling is presented. At the end we introduce how to use the simulator to set up a specific simulation experiment.

4.1 Overview

Wanting to perform wireless network simulation, an appropriate simulation software needs to be selected first. These days quite many discrete event simulators have become available. They include for instance *GloMoSim* [35, 36], a simulator developed by the Parallel Computing Laboratory, University of California, freely available for academic use, *QualNet* [37], a commercial simulation suite derived from GloMoSim, *ns-2* [38], developed by the Information Sciences Institute (ISI) of the University of Southern California and the VINT project [39], also freely available, and *OPNET* [40], for commercial use.

At the beginning of this project our choice fell on GloMoSim. Although there has not been any further development since December 2000 many people are still using it for it can be freely installed for academic use and it runs on different computer platforms. However, the decisive factor to choose GloMoSim were already existing implementations for GPSR, Terminode Routing and the Restricted Random Waypoint Mobility Model (see section 6.2.3) to conduct performance comparisons with the AMRA architecture. Unfortunately there is a number of deficiencies, which forced us to change to another simulation software during the project: GloMoSim uses a proprietary compiler (Parsec [36]), which makes it difficult to debug and track down memory errors by using specialized tools. Further, the documentation was sparely spread and no support was available while the mailing list often did not help. The most convenient solution was to move to QualNet for its many similarities to GloMoSim. It compiles using a standard C/C++ compiler, support is directly provided by the manufacturer, and last but not least this simulation software has already been used at university.

QualNet is a commercial modeling tool for wireless and wired networks developed by Scalable Network Technologies (SNT). The QualNet Developer suite consists of five different tools:

QualNet Simulator The discrete event simulator itself. SNT claims it to be "extremely scalable, accommodating high-fidelity models of networks of tens of thousands of nodes. QualNet makes good use of computational resources and models large-scale networks with heavy traffic and mobility in reasonable simulation times."

QualNet Animator A graphical user interface for intuitive experiment setup and animation tool. It consists of a standalone Java tool, which runs the simulator in a separate process while communicating with it through sockets and standard I/O streams.

QualNet Designer A finite state machine tool for custom protocol development. It allows simplified design of new routing protocols and faster deployment into the network simulation software.

QualNet Analyzer A statistical graphing tool for evaluating the metrics. It visually represents the statistics text file with the data collected during a QualNet experiment.

QualNet Tracer A standalone packet-level visualization tool for viewing the packets going up and down the protocol stack, and across the network. It reads packet data from an optional trace file generated during the simulation.

QualNet Importer A topology and a device importer to embed real network topology and traffic information from existing networks into a QualNet model. The data is collected through the use of SNMP.

The simulator is fully implemented in C/C++ while the graphical toolkit is programmed in Java. QualNet's design provides a layered architecture (physical, MAC, network, transport and higher layers) and utilizes standard APIs to interact with other layers. In this project we only used the simulator part for the graphical user interface devours a considerable amount of system resources and slows down the simulation process. The experiments were executed using the batch mode and the according configuration files.

4.2 Events, Packets and Messages

In a discrete event simulator the simulation is performed at specific points of time when an event occurs. There is no constant time flow. Such a simulator is based on an event scheduler, which contains any event that needs to be processed and stepped through. The simulation time is increased in discrete steps to the time of the actual event whenever an event occurs. Processing an event may also produce some new events; those are inserted into the event queue.

Every protocol in QualNet is implemented as a final state machine, which only changes its state on the occurrence of an event. Examples for such events are the arrival of data packets or a periodic timer event for advertisement of routing information to neighboring nodes. The simulator uses a stack model with each protocol inserted at one or more layers of the stack. At each layer, every protocol operates its own finite state machine. In order to enable communication between different layers protocols are required to create and schedule corresponding events to the target layer. In summary every protocol may either create events to change its own state or send an event to another protocol, which is handled at a different layer in the stack.

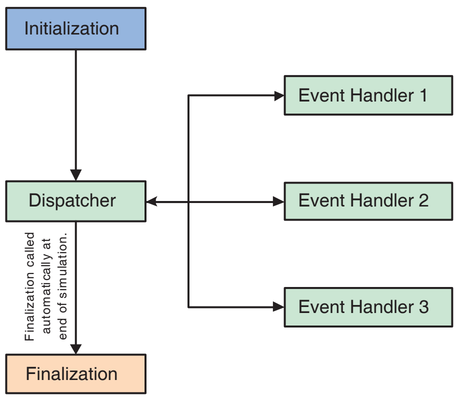


Figure 4.1: QualNet Event Handling.

A very simplified form of a protocol is presented in figure 4.1, in state diagram form. Every protocol starts with an initialization function, which reads external input and configures the protocol. The handling is then passed over to an event dispatcher. At the occurrence of an event for that layer QualNet Simulator first determines the event's protocol and hands it to the dispatcher for that protocol. The event dispatcher now checks for the type of event and calls the appropriate event handler for processing it. At the end of the simulation, a finalization function is called for every protocol. It prints out the collected statistics. The event that brings the simulator into finalize state is being generated automatically at the end of the simulation.

In QualNet all events are characterized by so called *messages*, a data structure that holds the information about the corresponding event. A message can hold two varying types of events, which are handled differently by the simulator.

Packet events represent data packets holding virtual or real data being sent across the network. When a node wants to transmit a packet to an adjacent layer in the protocol stack it schedules an event to that layer by sending an appropriate message. The arrival of the data packet is then simulated by the packet event occurrence at that designated layer. Usually, when a node transmits data to some other node, the packet is handed down through the protocol stack on the sending node. Then it is transmitted across the network and finally handed up on the receiving node. As well as in the real network architecture, header information is being added at each layer when sending down the stack, and header information is being removed when sending up the stack. Every layer in the network stack has its own responsibility for adding/stripping headers and for sending packets to its adjacent layer. QualNet Simulator provides layer specific API functions for message processing and passing, which is illustrated in figure 4.2.

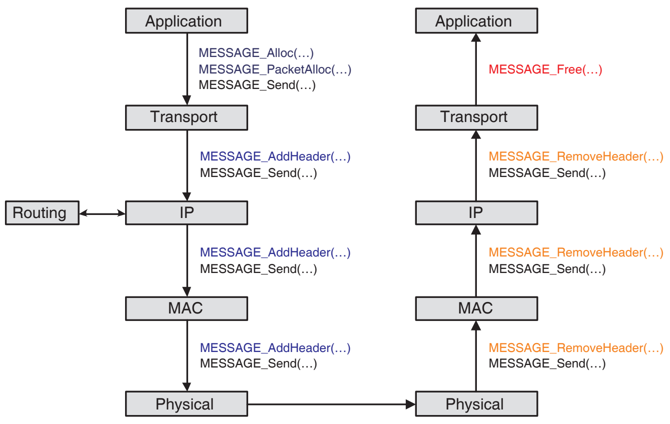


Figure 4.2: QualNet - Life Cycle of a Packet.

The QualNet message API also allows to save some extra information about a packet. This optional amount of data is not included in the payload and is ignored in the propagation and delay calculations.

Timer events are used by a protocol to schedule an 'alarm' for a future time. For instance, if a route entry expires after a certain time, the protocol may schedule a route expire event for that time. As soon as this event occurs, the appropriate event handler function is being called and the route entry is removed from the routing table. Periodic events may be implemented by resetting the timer event after processing it.

So, when implementing a new routing protocol, the developer has to write event dispatcher functions for the many messages respectively events that might turn up at that layer.

4.3 Implementing a Routing Protocol

This section provides a short insight into the implementation and integration of a routing protocol in QualNet. It describes the most important protocol functions, how they interact among each other, and how they communicate with the IP layer.

First of all, a unique protocol number for the routing protocol and the corresponding IP protocol needs to be defined in `include/network.h` respectively in `network/ip.h`. The latter is required, as the routing protocol will probably transmit control packets to establish and maintain routes. Thus, in order to guarantee handover of control packets to the routing protocol, the IP protocol number must differ from that of usual data (e.g. TCP or UDP).

As already stated, a routing protocol's core functions include initializing, event dispatching/handling, routing and finalizing. Figure 4.3 shows an excerpt of the most important QualNet functions used for protocol interaction. The yellow highlighted functions refer to MAC layer functions, the blue ones to the IP layer, and the green methods indicate our newly developed routing protocol. The new protocol is stored in a separate new file, such as `network/newprot.c` for instance.

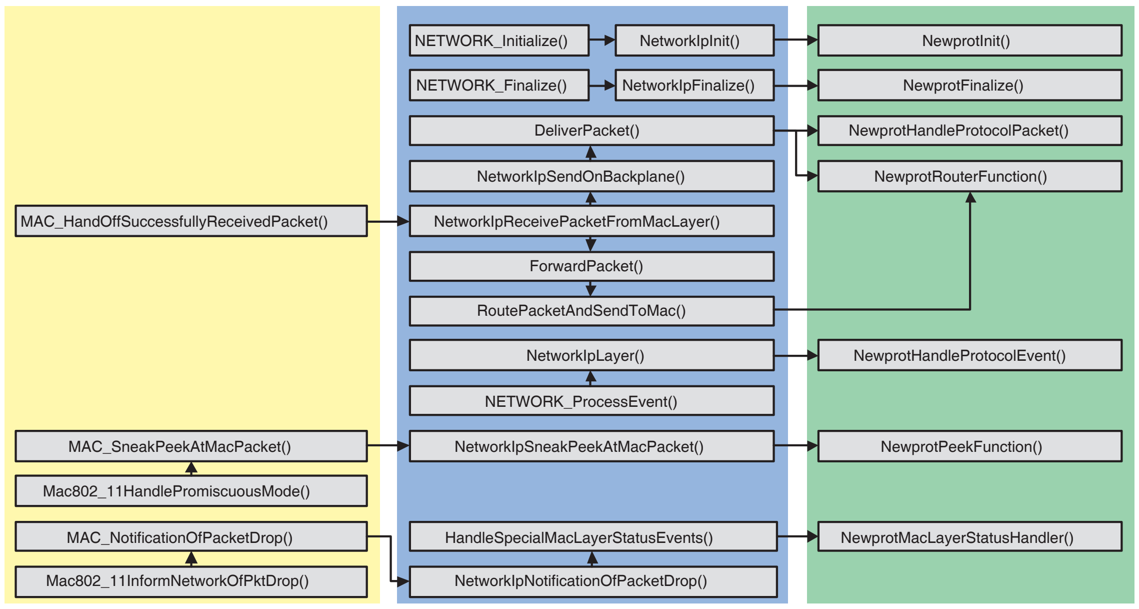

NETWORK_Initialize()
NetworkIpInit()
NewprotInit()
NETWORK_Finalize()
NetworkIpFinalize()
NewprotFinalize()
DeliverPacket()
NewprotHandleProtocolPacket()
NetworkIpSendOnBackplane()
NewprotRouterFunction()
MAC_HandOffSuccessfullyReceivedPacket()
NetworkIpReceivePacketFromMacLayer()
ForwardPacket()
RoutePacketAndSendToMac()
NetworkIpLayer()
NewprotHandleProtocolEvent()
NETWORK_ProcessEvent()
MAC_SneakPeekAtMacPacket()
NetworkIpSneakPeekAtMacPacket()
NewprotPeekFunction()
Mac802_11HandlePromiscuousMode()
MAC_NotificationOfPacketDrop()
HandleSpecialMacLayerStatusEvents()
NewprotMacLayerStatusHandler()
Mac802_11InformNetworkOfPktDrop()
NetworkIpNotificationOfPacketDrop()

Figure 4.3: QualNet - Function Overview of a Routing Protocol.

Initialization At simulation start, every node calls the `NewprotInit` routine to allocate and initialize model specific data. This includes memory reservation for the local routing variables, reading external input from files to configure the protocol, initialization of the routing table and statistical values, registering the router function, MAC layer status handler function, and the method to handle promiscuous mode.

```
void NewprotInit(Node *node, NewprotData **newprotPtr, const NodeInput *nodeInput,
    int interfaceIndex) {
    NewprotData *newprot = (NewprotData *) MEM_malloc (sizeof(NewprotData));
    (*newprotPtr) = newprot;

    (...)

    NewprotInitOptions(node, nodeInput);
    NewprotInitStats(node);
    NewprotInitRoutingTable(node);

    NetworkIpSetMacLayerStatusEventHandlerFunction(node,
     &NewprotMacLayerStatusHandler, DEFAULT_INTERFACE);

    NetworkIpSetPromiscuousMessagePeekFunction(node, &NewprotPeekFunction,
        DEFAULT_INTERFACE);

    NetworkIpSetRouterFunction(node, &NewprotRouterFunction, DEFAULT_INTERFACE);
}
```

Finalization is automatically performed at the end of the simulation and serves the purpose to output all statistical data collected throughout the simulation process. It first obtains a pointer to the local variable space and then reports the values mentioned. Those can be defined by the user and may include number of data initiated, number of data transmitted, number of data received, number of route requests sent, and so on.

```
void NewprotFinalize(Node *node) {
    NewprotData* newprot =
        (NewprotData *) NetworkIpGetRoutingProtocol(node, ROUTING_PROTOCOL_NEWPROT);
    char buf[MAX_STRING_LENGTH];
    sprintf(buf, "Number of Data Received = %d", newprot->stats.numDataRcvd);
    IO_PrintStat(node, "NETWORK", "NEWPROT", ANY_DEST, -1, buf);
}
```

Router function The router function registered during protocol initialization represents the core method for the routing process. Every time a packet arrives at a node, it is handed to this function to either determine the next hop towards the destination, or to carry out final tasks before passing the data to the application. It therefore needs access to the local variable space of the routing protocol (e.g. the routing table) and to the IP header of the packet in transition. Usually, control packets are handled by different functions and are therefore sorted out. Finally, if the destination has not yet been reached and the next hop is determined, the packet again is handed down to the MAC layer for further transmission.

```
void NewprotRouterFunction(Node *node, Message *msg, NodeAddress destAddr,
    NodeAddress previousNode, BOOL *packetWasRouted) {
    NewprotData* newprot =
        (NewprotData *) NetworkIpGetRoutingProtocol(node, ROUTING_PROTOCOL_NEWPROT);
    IpHeaderType *ipHeader = (IpHeaderType *) MESSAGE_ReturnPacket(msg);

    if (ipHeader->ip_p == IPPROTO_NEWPROT) {
        return;  // do not route control packets
    }

    if (NetworkIpIsMyIP(node, destAddr)) {
        *packetWasRouted = FALSE;
        return;
    } else {
        *packetWasRouted = TRUE;
    }

    (...)

    NetworkIpSendPacketToMacLayer(node, msg, DEFAULT_INTERFACE, nextHop);
}
```

Protocol packet and event handling Two methods, `NewprotHandleProtocolPacket` and `NewprotHandleProtocolEvent`, are needed to handle protocol packets and protocol events. Upon arrival of a Newprot control packet (e.g. route request) for that specific node, `NewprotHandleProtocolPacket` first determines the type of the packet arrived and defines further steps for packet handling, such as updating routing tables and sending back a route reply. Note that a packet is only handled by this routine, if it is addressed explicitly to this node or if it was sent to the broadcast address. Otherwise, `RoutePacketAndSendToMac` directly hands it to the router function (see figure 4.3).

`NewprotHandleProtocolEvent` processes special protocol events such as alarms or regular tasks like route expiry. All the related messages are defined in `include/api.h` through a unique number. Thus, the method first determines the type of event and then handles it accordingly, which may also include the creation of new future events. The original event message has to be killed at the end of event handling.

Promiscuous listening Some routing protocols make use of promiscuous mode in order to detect neighbors within transmission range for instance. In order to overhear all packets sent from and to other nodes, the `NewprotPeekFunction` needs to be implemented and registered during protocol initialization. Furthermore, promiscuous mode needs to be turned on in the main configuration file.

MAC failure handling `NewprotMacLayerStatusHandler` acts as a callback function from the MAC layer. It enables the routing protocol to get notice from link breaks at the lower layer, to delete the faulty link from the routing table and to retransmit the packet to a different node. Otherwise, in an IEEE 802.11 compliant MAC layer for instance, the packet would just be dropped after seven failed retransmissions. As well as the peek function and the router function, `NewprotMacLayerStatusHandler` needs to be registered during protocol initialization.

The last step to take is to enable the IP layer to recognize the Newprot protocol. Therefore, in the `network/ip.c` file, function calls to the initialize, finalize, packet and event handling methods must be inserted in the appropriate methods. Also, the header file of the newly developed protocol must be included in `ip.c`.

4.4 Setting up a Simulation Experiment

Besides simulation setup through QualNet Animator, which suffers from bad performance, an experiment can also be described and carried out by the specification of various simple and text-based configuration files. In the following paragraphs we present the most important settings that have been used and changed for the simulation process.

mysim.config In a clean QualNet install `bin/default.config` refers to the main configuration file, by which most of the simulator settings can be controlled. In the following, we operate on a copy of this file (`bin/mysim.config`[1]), which we customize for our purposes. It supports a huge amount of different options for every network layer: network topology, antenna and transmission parameters, frequencies, medium access control, routing models, and so on. By far most of the

[1]mysim.config is just used as an example in this document, virtual any other different file name may be chosen, e.g. aodv_1000n_highmobility.config.

parameters used in this paper were default values from QualNet. The description in this section is limited to customized values used for this project.

The first section specifies general simulation properties for a specific experiment:

```
EXPERIMENT-NAME mysim
SIMULATION-TIME   1080S
SEED   24597
COORDINATE-SYSTEM    CARTESIAN
TERRAIN-DIMENSIONS   (1500, 300)
```

- The experiment name is used for generation of the output files. Output is written to the following two files:
  - `<EXPERIMENT-NAME>.stat`, for collected statistics about the simulation.
  - `<EXPERIMENT-NAME>.trace`, for the packet-level trace file.
- Simulation time specifies the duration of the entire simulation. The following different units are supported: NS (nanoseconds), US (microseconds), MS (milliseconds), S (seconds), M (minutes), H (hours) and D (days).
- The seed value is needed for the initialization of the random number generator, which is used in several models, e.g. for random node placement or random mobility models.
- Finally the simulation area is defined by using the Cartesian coordinate system, the terrain dimensions are indicated in meters.

Furthermore the network topology needs to be defined: Every node in QualNet is addressed by a unique node identifier. In the example below, a subnet with 50 nodes is generated using 16 bits for the subnet mask. As an example, N16-0 indicates the address 0.0.0.0/16, while N8-192.168.0.0 means 192.168/24. Node placement is initialized using a custom node placement file, which is read by the simulator at startup. Additionally, we define a random waypoint mobility model (see section 6.2.1) using a pause time of 30 seconds, a minimum and maximum speed of 1 respectively 20 meters per second. Position granularity indicates how far a node may move in meters without needing to recalculate all parameters.

```
SUBNET N16-0 { 1 thru 50 }
NODE-PLACEMENT        FILE
NODE-PLACEMENT-FILE   ./mysim.nodes

MOBILITY                RANDOM-WAYPOINT
MOBILITY-WP-PAUSE       30S
MOBILITY-WP-MIN-SPEED   1
MOBILITY-WP-MAX-SPEED   20
MOBILITY-POSITION-GRANULARITY   5.0
```

Finally, a physical transmission model, transmission power, a routing protocol, the application and statistical parameters need to be defined. The application layer parameters for the simulation are stored in a separate file, which is described later. Statistics are collected by every different network layer and are stored to the `mysim.stat` file, which can best be viewed with QualNet Analyzer or post-processed with a Perl script. Packet Tracing is turned off by default, because it typically requires huge amounts of disk space. However, in order to determine the routing overhead and other custom metrics, packet tracing must be turned on.

```
PROPAGATION-PATHLOSS-MODEL  TWO-RAY
PHY-MODEL PHY802.11b
PHY802.11b-TX-POWER-DBPSK         7.874
PHY802.11b-TX-POWER-DQPSK         7.874
PHY802.11b-TX-POWER-CCK-6         7.874
PHY802.11b-TX-POWER-CCK11         7.874
IP-FORWARDING YES
ROUTING-PROTOCOL AODV
APP-CONFIG-FILE ./mysim.app

APPLICATION-STATISTICS                  YES
TCP-STATISTICS                          YES
UDP-STATISTICS                          YES
ROUTING-STATISTICS                      YES
NETWORK-LAYER-STATISTICS                YES
QUEUE-STATISTICS                        YES
MAC-LAYER-STATISTICS                    YES
PHY-LAYER-STATISTICS                    YES
```

mysim.nodes In the `mysim.nodes` file the placement of the nodes inside the simulation area is specified. Each line defines a new node record. The parameters must be given in the following order:

- First the unique node identifier is indicated.
- The second parameter is for consistency with the mobility trace format, it is always set to 0.
- Next, the x, y and z coordinate of the node are defined in brackets (in meters).
- Optionally, the orientation of the node using the last two floating point parameters may be defined.

```
# NODE-PLACEMENT-FILE
# Format: nodeId 0 (x, y, z) [azimuth elevation]
1 0 (187.5, 150.0, 0.0) 0.0 0.0
2 0 (1312.5, 150.0, 0.0) 90.0 0.0
3 0 (451.446533203125, 265.402221679688, 0.0)
```

mysim.app QualNet offers different models to produce the data expected to flow across a network – they are called *traffic generators*. Models of representative applications (such as web browsing, file transfer, Telnet) can be used, but there also exist different theoretical or trace-based models. We give a short description of each available model and limit the description of the file format to CBR, because only CBR traffic was used in the experiment setup.

- **FTP** represents the File Transfer Protocol server and client.
- **FTP/Generic** represents a more configurable model of the File Transfer Protocol server and client. The size of the items sent is not determined by network traces, but specified by the user instead.
- **HTTP** represents a single-thread web-browser and a set of web-servers the browser will connect to. The model considers 'think time' between client requests, and varying numbers of pages, items per page, and size of items, in the server responses. The client also utilizes variable lengths of sessions, during which it makes requests to the same server for a given number of pages.
- **Telnet** represents the clear text terminal server and client. The typing rate and sizes of server responses are taken from distributions created from network traces.

- **LOOKUP** is an unreliable query/response application that can be used to simulate applications such as DNS lookup or ping.
- **TRAFFIC-GEN** simulates a random-distribution based traffic generator.
- **TRAFFIC-TRACE** simulates a trace file-based traffic generator.
- **CBR** stands for **c**onstant-**b**it-**r**ate traffic. It is generally used to simulate multimedia traffic, or to fill in background traffic to affect the performance of other applications being analyzed.

  The file format is as follows:
  `CBR <src> <dest> <items_to_send> <item_size> <interval> <start time> <end time>`

  The example below indicates that node 1 will send 64 byte packets to node 2. The 0 for <items_to_send> specifies that the number of items is unlimited – items will be sent until <end time>. Four packets will be generated per second (every 0.25 seconds), starting from 180 simulation seconds, and ending at 890 seconds.

  ```
  CBR 1 2 0 64 0.25S 180S 890S
  ```

- **MCBR** stands for Multicast CBR (constant-bit-rate) traffic. It is similar to CBR, but sets a multicast address as the destination and requires multicast protocols to be running.
- **VBR** represents **v**ariable-**b**it-**r**ate traffic.
- **VoIP** is available for voice over IP applications. It requires configuration of the H323 settings.

To start the simulation process, the QualNet executable is invoked with the main configuration file as an argument: `qualnet mysim.config`.

4.5 Statistics

As mentioned in the previous sections every simulation writes detailed statistical output to a file (e.g. `mysim.stat`). The output is principally structured by node identifier and network layer, every line holding one specific value, as shown in the following:

$$< node\ id >, < interface\ IP\ address >, < instance\ id >, < layer >, < protocol >, < value >$$

During the finalization phase of an experiment, the `IO_PrintStat` function is responsible for statistic file generation. The unique node identifier is mandatory, all other fields can be optionally supplied or left empty. Concerning the queuing for instance a network interface may have several different queues, which are identified by the interface address and an instance id. However, depending on the number of network interfaces, it may only exist one instance for physical and MAC layer. The example below illustrates a cut-out of a statistics file for node number 182, showing physical, MAC and part of the network layer. The network layer is subdivided into the routing protocol (GPSR), IP protocol and FIFO queuing.

```
182,          , [0], Physical,    802.11,Signals transmitted = 3762
182,          , [0], Physical,    802.11,Signals received and forwarded to MAC = 10845
182,          , [0], Physical,    802.11,Signals locked on by PHY = 10899
182,          , [0], Physical,    802.11,Signals received but with errors = 54
182,          , [0], Physical,    802.11,Energy consumption (in mWhr) = 75.068
```

```
182,          , [0],      MAC, 802.11MAC,Packets from network = 966
182,          , [0],      MAC, 802.11MAC,UNICAST packets sent to channel = 920
182,          , [0],      MAC, 802.11MAC,BROADCAST packets sent to channel = 46
182,          , [0],      MAC, 802.11MAC,UNICAST packets received clearly = 919
182,          , [0],      MAC, 802.11MAC,BROADCAST packets received clearly = 3454
182,          , [0],      MAC, 802.11DCF,Unicasts sent = 920
182,          , [0],      MAC, 802.11DCF,Broadcasts sent = 46
182,          , [0],      MAC, 802.11DCF,Unicasts received = 919
182,          , [0],      MAC, 802.11DCF,Broadcasts received = 3454
182,          , [0],      MAC, 802.11DCF,CTS packets sent = 946
182,          , [0],      MAC, 802.11DCF,RTS packets sent = 923
182,          , [0],      MAC, 802.11DCF,ACK packets sent = 926
182,          , [0],      MAC, 802.11DCF,RTS retransmissions due to timeout = 2
182,          , [0],      MAC, 802.11DCF,Packet retransmissions due to ACK timeout = 1
182,          , [0],      MAC, 802.11DCF,Packet drops due to retransmission limit = 0
182,          ,    ,  NETWORK,      GPSR,Number of Data Originated = 920
182,          ,    ,  NETWORK,      GPSR,Number of Data Received = 919
182,          ,    ,  NETWORK,      GPSR,Total Number of Hops= 75335
182,          ,    ,  NETWORK,      GPSR,TotalPerimLen = 67783
182,          ,    ,  NETWORK,      GPSR,AvgPerimLen = 73.757345
182,          ,    ,  NETWORK,      GPSR,AvgHopCnt = 81.974973
182,          ,    ,  NETWORK,      GPSR,Number of HELLO Sent = 46
182,          ,    ,  Network,        IP,ipInReceives = 4373
182,          ,    ,  Network,        IP,ipInHdrErrors = 0
182,          ,    ,  Network,        IP,ipInForwardDatagrams = 0
182,          ,    ,  Network,        IP,ipInDelivers = 4373
182,          ,    ,  Network,        IP,ipOutRequests = 966
182, 0.0.0.182, [0],  Network,      FIFO,Total Packets Queued = 46
182, 0.0.0.182, [0],  Network,      FIFO,Total Packets Dequeued = 46
182, 0.0.0.182, [0],  Network,      FIFO,Total Packets Dropped = 0
182, 0.0.0.182, [0],  Network,      FIFO,Average Queue Length (bytes) = 51.729691
182, 0.0.0.182, [0],  Network,      FIFO,Average Time In Queue = 0.000000000
182, 0.0.0.182, [0],  Network,      FIFO,Longest Time in Queue = 0.000000000
182, 0.0.0.182, [0],  Network,      FIFO,Peak Queue Size (bytes) = 52
182, 0.0.0.182, [1],  Network,      FIFO,Total Packets Queued = 920
182, 0.0.0.182, [1],  Network,      FIFO,Total Packets Dequeued = 920
182, 0.0.0.182, [1],  Network,      FIFO,Total Packets Dropped = 0
182, 0.0.0.182, [1],  Network,      FIFO,Average Queue Length (bytes) = 309.396667
182, 0.0.0.182, [1],  Network,      FIFO,Average Time In Queue = 0.000000000
182, 0.0.0.182, [1],  Network,      FIFO,Longest Time in Queue = 0.000000000
182, 0.0.0.182, [1],  Network,      FIFO,Peak Queue Size (bytes) = 404
```

Thanks to the standardized file format, which is used in any simulation experiment, the statistics can easily be post-processed with a Perl script for instance.

5 Implementation in QualNet

The following sections characterize the AMRA implementation for the QualNet network simulation software. At the time the project was carried out version 3.6.1 was available and used to perform the implementation and to conduct simulation experiments. At first we point to some requirements of the desired solution followed by technical details regarding the embedding to the simulator. There we demonstrate how the interaction of the two routing protocols has been accomplished. Also, we present the default settings chosen to run simulations. A short abstract of new and modified files as well as remarks concerning the Terminode protocol porting are also included.

5.1 Requirements

AMRA requires the routing layer to be split in an upper and a lower layer as presented earlier in figure 3.5 on page 24. On the lower layer, the GPSR protocol is used for packet forwarding to designated logical routers. MABR operates on the upper layer and performs topology abstraction and takes routing decisions for packets. Neither GloMoSim nor QualNet have been developed in order to support interaction of multiple routing protocols at the same time. QualNet provides only a single pointer for holding the data structure of the routing protocol. Thus, it was quite a critical task to divide the network layer into two sublayers without needing to modify major parts of the network simulation software. The new solution to be implemented had to satisfy the following aspects:

- GPSR was supposed to work both standalone and in cooperation with MABR. If the AMRA routing architecture is selected in the main configuration file, GPSR has to be initialized by AMRA.
- Extensive changes to GPSR must be avoided in order to make adaption fast, easy and not to break the original protocol's operation in a standalone experiment.
- The network layer of QualNet ought to remain unchanged.
- Exchangeability of GPSR with other sublayer routing protocols should be achieved relatively straightforward without needing to adapt large parts of the AMRA code.

5.2 Subdivision of the Routing Layer and Interaction with GPSR

A solution using function pointers was considered in order to attain the goals described above. MABR instead of QualNet stores the data structure for the lower routing layer, i.e. GPSR in our case. During its initialization AMRA calls a special GPSR sublayer initialization function, which generally performs the same tasks as the standard startup function but additionally returns function pointers to relevant GPSR methods such as the router function, promiscuous peek function, MAC layer status handler, and finalization.

Most methods of a routing protocol require to obtain a pointer to the routing protocol's structure. This is done by using the `NetworkIpGetRoutingProtocol` function. In the two-layered approach we implement, a call returns a null pointer if a GPSR method tries to get the local variable space. This is the consequence of MABR being the main registered routing protocol in QualNet. The subsequent explicit typecast to a GPSR structure causes the program to crash. Therefore, we require some context switching every time a GPSR method tries to get its own data structure. This is achieved by calling a function pointer at the beginning of a GPSR method, which replaces the MABR pointer by the GPSR pointer in the QualNet routing structure. Hence, GPSR now is able to obtain its own local variable space as if it was the only routing protocol. In a GPSR standalone experiment the function pointer is assigned to a no-operation method. At the end of the GPSR method `RoutingDeceiveSublayerStop` resets the settings to the initial state. The use of a function pointer is very convenient for it allows to quickly exchange GPSR with any other lower layer routing protocol. During initialization the pointer can simply be loaded with the appropriate method for the specific lower layer protocol. The short code extract below illustrates context switching in a GPSR function.

```
extern void *(*sublayerRoutingStartFakeFunctionPtr)(
    Node *node,
    NetworkRoutingProtocolType *nrpt,
    int interfaceIndex);

void GpsrHandleProtocolEvent(Node *node, Message *msg)
{
    // switch context in QualNet to the lower layer routing protocol and save
    // upper layer routing protocol to revert at end of the method
    NetworkRoutingProtocolType mainRoutingMode;
    void *mainRouting = sublayerRoutingStartFakeFunctionPtr(node, &mainRoutingMode,
        DEFAULT_INTERFACE);
    // obtain pointer to the local data structure of GPSR
    GpsrData* gpsr = (GpsrData *) NetworkIpGetRoutingProtocol(node, ROUTING_PROTOCOL_GPSR);

    // protocol code here

    // revert local variable space to the upper protocol
    RoutingDeceiveSublayerStop(node, mainRouting, mainRoutingMode, DEFAULT_INTERFACE);
}
```

Concerning protocol interaction MABR always receives packets first, then processes them, and finally hands them over to GPSR if needed. The required GPSR methods can be invoked by MABR using the function pointers assigned during protocol initialization.

5.3 Settings

AMRA supports various configurations, which are customized by altering the main configuration file. This section describes the settings used in our simulation experiments. A description of the operation performed by AMRA is given in section 3.3.

Forward Ant Forwarding If greedy forwarding fails forward ants determine their next hop according to the GPSR right-hand or the left-hand rule. The left-hand rule works similarly as the right-hand rule: the next edge traversed is the next one clockwise (instead of counterclockwise) about the current node from the preceding edge. In order to detect as much paths as possible forward ants utilize right- and left-hand forwarding randomly, as selected by their initiator. A good path found using the right-hand rule can turn out quite bad when using the left-hand rule for instance. Note that backward ants follow the path determined by the pheromones.

Logical Router Size Logical routers in the form of squares are spread over the entire simulation area. A unique identifier is assigned to every logical router which also allows to compute the router's coordinates. Mobile nodes within a logical router are supposed to contain similar routing information. For our simulations we considered a logical router size of two times the transmission range, i.e. 500 meters.

Minimum Probability Threshold As explained in the AMRA section, logical links may not be used for relaying if the corresponding link probability falls below a defined threshold. This eliminates possible relays of packets to wrong directions or to very unreliable logical links. We found a minimum probability value of 20% to be sufficient.

Reinforcement Metric Quite many metrics could be used for the pheromone reinforcement: end-to-end delay, hop count, euclidean path followed, jitter experienced, or a combination among them for instance. Since QualNet supports delay calculations we utilized the end-to-end delay experienced by the packet in our experiments.

Ant Initiation Ants can be initiated regularly or based on activity. In the latter case, a forward ant is only generated if no other ant transmission from the current logical router has been listened to during the last time interval. Also, in order to prevent nodes from sending ants all at the same time and thus congesting queues, ants are delayed using a random jitter interval. In this work activity-based ant initiation was performed using a sending interval of 5 seconds and a jitter interval of 2.5 seconds.

Loop Detection No loop detection scheme was implemented for our simulations. However, if a forward ant passes the same node for the third time, the ant is dropped to avoid the creation of strange pheromone trails.

The table below summarizes AMRA protocol settings in the main configuration file:

Setting in mysim.config	**Description**
`ROUTING-PROTOCOL MABR`	Sets the MABR routing protocol, i.e. the AMRA routing architecture.
`MABR-SUBLAYER GPSR`	Sets the sublayer routing protocol. Currently only `GPSR` is supported.
`MABR-GPSR-ANT-ROUTING MIXED`	Determines the forwarding method for forward ants `RIGHTHAND`, `LEFTHAND`, or `MIXED` is supported.
`MABR-SEED 345`	Sets a random seed for the MABR algorithm.
`MABR-TAP-SIZE 500`	Sets the edge length of a logical router (in meters).
`MABR-MIN-PROB-THRESHOLD 0.2`	Sets the minimum threshold for a logical link to be considered for packet forwarding.
`MABR-TAP-PROBABILITY-METRIC DELAY`	Selects the metric used for the pheromone update. `DELAY` or `HOPCOUNT` are supported.
`MABR-ANT-MODE ACTIVITY-BASED`	Sets the ant initiate mode, which can be `REGULARLY` or `ACTIVITY-BASED`
`MABR-ANT-INTERVAL 5S`	Sets the interval to initiate ants.
`MABR-ANT-INITIATE-JITTER 2.5S`	Sets an ant jitter interval to make sure not all nodes transmit ants at the same time.
`MABR-FIRST-ANT-AFTER 0S`	The first ant is initiated after `MABR-FIRST-ANT-AFTER`. Useful for mobility to reach its steady state.
`MABR-LOOP-DETECTION NONE`	Loop detection is experimental and drops a packet if it is seen twice by the same node. `NONE` and `STRICT` can be entered.
`MABR-ENHANCED-STAT-INTERVAL 30S`	Sets an interval for enhanced statistics. Thus, every 30 seconds, mean delay, hop count, and euclidean distance are printed to the statistics file.

5.4 Remarks Concerning Porting of Terminode Routing

The initial presentation and implementation of the Terminode routing protocol makes use of the GloMoSim network simulator [16]. In order to provide a fair comparison of all protocols only Friend Assisted Path Discovery (FAPD) has been considered for our simulations. While the GMPD approach knows about the entire network, AMRA and GPSR are performing routing only based on the location of the final destination. Thus, the latter protocols would be discriminated.

Based on the original GloMoSim Terminode FAPD code we performed a porting to the QualNet network simulation software. Although no code review has been performed we experienced certain problems, which affected protocol operation or even caused the program to crash. Using the IBM Rational® Purify software many memory related errors showed up and could finally be fixed. We encountered uninitialized or free memory read or written, array bounds read or written, and other more. One or two copy-paste errors have been found and are fixed now.

A major flaw applies to the use of the extra information field, which stores additional data about the packet in transition (see section 4.1). This amount of data is not contained in the packet's payload and therefore is not considered in the propagation and delay calculation. However, the version of Terminode Routing we initially received makes use of this field in order to store routing relevant information such as the destination location. This structure, called `RoutingGeoDsrPacketInfo`, shows a considerable size of 736 bytes. For correct implementation it must be included in an additional header. In concern of QualNet porting we corrected this issue by moving the structure to an additional Terminode header. Another problem is given by the anchored path, which only has been implemented as a pointer and thus is treated as 4 bytes. We did not develop a solution to this problem. We would have liked to perform a detailed code review in order to guarantee correct operation of the Terminode routing protocol. In view of the limited time available to carry out this thesis we focused on further observations of the AMRA architecture.

Due to the limitations described above, simulation results are to be handled with care. In particular we have not been able to reproduce the results described in the various Terminode routing papers.

5.5 Customization of the Network Simulator

The following section lists file modifications and additions we used for embedding the new routing protocols and the restricted random waypoint mobility model. As already mentioned this work has been carried out on version 3.6.1 of the QualNet network simulation software.

Modification of Existing Files

addons/seq/node.h The node structure needed to be altered to add a `currentCity` identifier, which is required for restricted random waypoint mobility. Thus, a node can determine the city it had visited last.

bin/default.config The default configuration file has been provided with additional options for AMRA, GPSR and Terminode routing as well as for the restricted random waypoint mobility model and for restricted node placement.

include/api.h This file features new QualNet message types used in AMRA, GPSR and Terminode routing.

include/network.h Unique identifiers for the new routing protocol types were defined in `network.h`.

mobility/mobility.c Depending on the information provided by the main configuration file, a function call to the restricted random waypoint mobility model is needed to be added.

mobility/node_distribution.c A new method called `DistributeNodesRestrictedToCities` has been created. This function places mobile nodes randomly to cities according to the default city configuration file.

network/ip.c/.h This file experienced a number of changes: First of all, initialization, finalization, packet and event handler have been invoked for the three new routing protocols. We introduced an enhancement on MAC layer feedback what makes packets returning to the routing layer after seven failed retransmissions. Finally, if a GPSR protocol packet is encountered, which at the same time turns out as a MABR forward ant, it is directly handed over to the `MabrHandleForwardAnt` method rather than to the GPSR protocol stack.

New Files

The following files have been added to QualNet in order to conduct performance comparisons between AMRA, GPSR and Terminode routing:

bin/default.cities Defines cities for a specific experiment using restricted random waypoint mobility or restricted node placement.

bin/default.citylinks Defines highways between cities in case of restricted random waypoint mobility.

mobility/mobility_restricted_waypoint.c/.h A new implementation of the Restricted Random Waypoint Mobility model.

network/gpsr.c/.h The implementation of the GPSR routing protocol, which was initially ported from GloMoSim. Various enhancements to support the two-layered AMRA routing approach had to be included.

network/mabr.c/.h The new implementation of the MABR routing protocol.

network/sublayerrouting_handling.c/.h Provides procedures for the handling of the subdivided routing layer. Using these functions the context switching described in section 5.2 is performed.

network/tap.c/.h Provides functions used by MABR to perform topology abstraction and to maintain the routing table.

network/terminodes_fapd.c/.h The implementation of the Terminode routing protocol, which has been ported from GloMoSim.

Obviously all new files had to be included into the various Makefiles, which are to be found in the `main` directory.

6 Simulation Setup

Network simulations represent an important instrument for analyzing the operation of a routing protocol under varying conditions. In this chapter we focus on different simulation test scenarios we used to conduct basic performance comparisons. We first point to a few mobility models implemented and used during simulation, then we describe the various setups defined and finally present the metrics measured.

6.1 Overview

Since there are no previous implementations of the AMRA architecture, our major objective was to use the newly developed protocol to perform a basic performance study against two other position-based routing protocols, GPSR and Terminode Routing. It is commonly known that routing protocols for mobile ad-hoc networks sensitively react on mobility. Thus, besides a well-established standard scenario in small mobile network, we also examined various settings in a larger network, which includes utilization of a different mobility model, the use of static and dynamic mobile ad-hoc networks, and the reaction of the protocol on radical topology changes. Due to limitations of processing power, experiments have been limited to a maximum network size of 500 nodes.

6.2 Mobility Models

6.2.1 Random Waypoint Mobility Model

The *Random Waypoint Mobility Model* is by far the most often used model. It was first used by Johnson and Maltz in the evaluation of Dynamic Source Routing [4], and has been later refined by the same research group [41].

In this model a mobile node moves from its current location to a randomly chosen new location within the simulation area. It uses a random speed uniformly distributed between $[v_{min}, v_{max}]$. v_{min} refers to the minimum speed of the simulation, v_{max} to the maximum. The Random Waypoint Mobility Model includes *pause times* whenever a new direction and speed is selected. As soon as a mobile node arrives at the new destination, it pauses for a selected time period (pause time) before starting traveling again.

This model is often simplified by using a uniformly distributed speed between $(0, v_{max}]$. In this case the average velocity is assumed as $\frac{v_{max}}{2}$. Further, the average speed is expected to be constant during the simulation process what is not the case. The research work presented in [42] examines and mathematically proves that the average speed constantly decreases and would eventually reach zero when using this model. The mobile nodes become more and more ‘stuck’, traveling long distances at low speeds. Results obtained by such a model obviously will not be reliable. A simple solution suggested is to set a positive minimum speed. For instance, when setting a minimum speed of 1 m/s, a steady-state average speed settles after some time at a positive value.

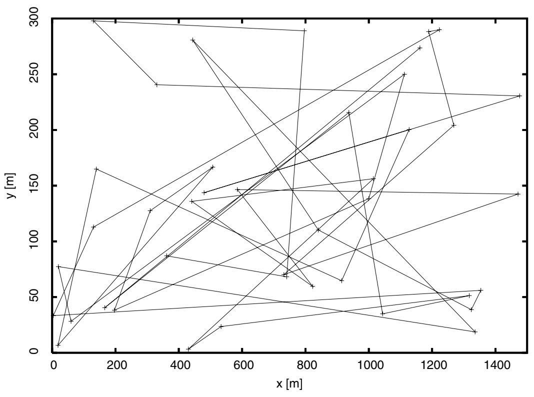


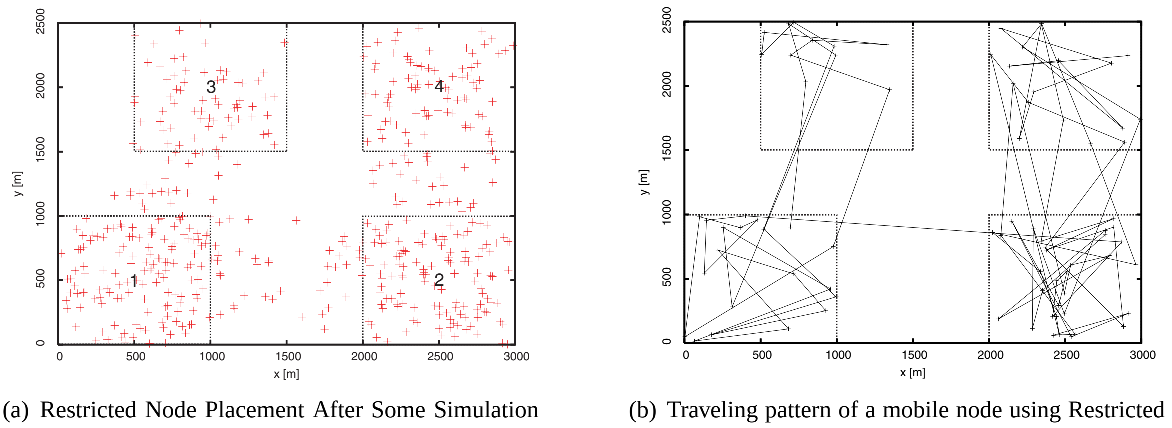
Figure 6.1: Traveling pattern of a mobile node using Random Waypoint Mobility.

6.2.2 Random Walk Mobility Model

Like the Random Waypoint Mobility Model, the *Random Walk Mobility Model* is a simple model based on random directions and speeds. However, no pause times are included between changes of direction and speed.

6.2.3 Restricted Random Waypoint Mobility Model

The *Restricted Random Waypoint Mobility Model* inherits aspects of random directions, speeds and pause times from the Random Waypoint Mobility Model. Additionally, it includes certain area restrictions. It aims to model behavior of nodes located in agglomerations of high node density, called cities, and on highways between cities. In real life, people often move within relatively small geographical areas but rarely travel long distances to other cities. Therefore we believe that this model better adapts to a real wide-area mobile ad-hoc network than the random waypoint mobility model does.



(a) Restricted Node Placement After Some Simulation Time, 3000m x 2500m, 500 nodes

(b) Traveling pattern of a mobile node using Restricted Random Waypoint Mobility

Figure 6.2: Restricted Random Waypoint Mobility Model

Initially, by applying *Restricted Node Placement* nodes are randomly distributed to several cities, which may be defined by the user. In the actual implementation both rectangular and circular cities are supported. Figure 6.2(a) depicts an example of restricted node placement for four rectangular cities.

Node movement within cities is very similar to the random waypoint mobility model. Therefore, minimum/maximum city speed and a city pause time need to be defined. As soon as a node arrives at its previously selected geographic destination in a city, it can either remain (i.e. walk to another destination in the same city), or choose a location within an other city, where the current city maintains a highway to. In order to support faster velocities during city changes, different speed boundaries can be assessed. Also, an alternate pause time may be applied after a city transition. The city change ratio is expressed by a city remain probability parameter, which is used in a coincidence experiment. It determines if we stay inside the current city or if a walk to another one needs to be done.

Moreover, one special type of node is distinguished in this model, called *commuter node*. Using only 'normal' nodes our first experiments showed the delivery ratio suffering from connectivity losses unless a very small city remain probability is assumed. A commuter node's major task is to maintain connectivity between cities by frequent movement on highways. Although commuters use same speeds as other nodes depending on location, a different city remain probability and pause time is defined to make them remain on highways. Upon arrival at the designated city location, a commuter only pauses for a short time and then most probably selects an other city. To get an idea of the model, figure 6.2(b) shows the track of a mobile node using restricted random waypoint mobility.

Finally, the model also features the dynamic creation and erasure of cities and highways and therefore allows to test a routing protocol's reaction on drastic changes of the network topology. Cities and highways as well as their creation and erasure are specified in two additional configuration files. An example of the files can be found in the appendix. The relevant settings entered into the main configuration file of QualNet are summarized in the table below.

Setting in mysim.config	Description
`NODE-PLACEMENT RESTRICTED`	Sets the restricted node placement.
`MOBILITY RESTRICTED-RANDOM-WAYPOINT`	Sets the restricted random waypoint mobility.
`CITY-PLACEMENT-FILE ./default.cities`	Sets the path to the city placement file.
`LINK-CONF-FILE ./default.citylinks`	Sets the path to the link configuration file, which specifies the highways.
`MOBILITY-RWP-PAUSE 120S`	Sets the pause time after a city change.
`MOBILITY-RWP-MIN-SPEED 10`	Sets the minimum speed for mobility between cities.
`MOBILITY-RWP-MAX-SPEED 30`	Sets the maximum speed for mobility between cities.
`MOBILITY-RWP-CITY-PAUSE 120S`	Sets the pause time after a location change within a city.
`MOBILITY-RWP-MIN-CITY-SPEED 1`	Sets the minimum speed for mobility within a city.
`MOBILITY-RWP-MAX-CITY-SPEED 15`	Sets the maximum speed for mobility within a city.
`MOBILITY-RWP-CITY-REMAIN-PROB 0.8`	Sets the probability for a node to remain within a city.
`MOBILITY-RWP-NUM-COMMUTERS 300`	Sets the number for commuter nodes, which are needed to maintain connectivity and predominately change cities.
`MOBILITY-RWP-COMMUTER-PAUSE 1S`	Sets the pause time for commuter nodes.
`MOBILITY-RWP-COMMUTER-CITY-REMAIN-PROB 0.1`	Sets the probability for a commuter node to remain within a city.

6.3 Simulation Scenarios

6.3.1 Evaluation in a Small Network with Uniform Node Distribution

Our first experiments followed a widely used simulation scheme for smaller mobile ad-hoc networks. They were targeted on a basic test of the AMRA architecture as well as on proving our algorithm not to be much worse than GPSR or Terminode Routing in this small network. The evaluations are based on a flat, rectangular area of 1500m $\times$ 300m and 50 randomly distributed wireless mobile nodes. Simulation time was 900 seconds. Node movement was reproduced by use of the Random Waypoint Mobility Model, using a minimum speed of 1 meter per second, a maximum speed of 20 meters per second, and pause times of 0, 30, 60, 120, 300, 600 and 900 seconds. The rate of mobility is characterized by the pause time: a pause time of 0 seconds signifies continuous motion, a pause time of 900 seconds corresponds to no motion. Five simulation runs have been performed for every setting using different seed values in order to generate varying movement patterns. Constant bit rate (CBR) sources, sending four 64-byte packets per second, ensured the generation of network traffic. The first packet was initiated at 120. This allows the mobility model to reach its steady state. We stopped sending data at 890 seconds to concede enough time to final packets to reach their destination. We were using both unidirectional and bidirectional traffic by use of 1 respectively 2, and 20 sources.

6.3.2 Evaluation in a Large Network with Restricted Random Waypoint Mobility Model

Next, we were interested in the protocol's behavior in a larger and more complex environment. Therefore, we introduced the Restricted Random Waypoint Mobility model, which simulated 500 nodes, four cities, and three highways on a rectangular field of 3000m $\times$ 2500m. The setup of cities and highways is equal to the one depicted in figure 6.2(a) on page 50. In this scenario, GPSR was obliged to use perimeter mode forwarding every so often due to the hole in the network topology. As a consequence of long processing times we only chose one fixed pause time of 120 seconds. A rather high mobility was configured: city speed varied between 1 and 15 meters per second, minimum and maximum speed on highways had been set to 10 respectively 30 meters per second. In order to obtain different node placements and movement patterns three simulations with varying random seeds for every protocol had been planned. The traffic model included 10 CBR sources that we used in a unidirectional and a bidirectional experiment.

6.3.3 Evaluation in a Large Static Network with Restricted Node Distribution

As presented later in section 7.2 results of the previous simulations appeared very disappointing for all protocols. Thus, to test AMRA in a more complex topology, and to guarantee the delivery of GPSR packets, we chose a static network whose node distribution is similar to the one in the previous experiment (see figure 6.2(a)). Three different node placements were used to gain sufficient simulation results. The routing protocol's task was to deliver packets from one source located in the center of city 3 to the center of city 4, and vice versa in the bidirectional case. Since the protocol's behavior is not affected by mobility in these scenarios, and to save simulation time, we shortened the simulation time to 300 seconds. Four data packets per second were initiated from 60 to 290 seconds of simulation time.

6.3.4 Evaluation of Radical Topology Changes in a Large Static Network

In a next phase, the static experiments above were refined in order to determine the influence of radical topology changes. In two different setups, an additional highway between city 3 and 4 (see figure 6.2(a)) was inserted or removed at 240 seconds of simulation time. For the first scenario, 31 arbitrary nodes were randomly relocated to the area between the two cities thus creating new possible paths. On concern of the highway removal, nodes were relocated between city 3 and 4 at simulation startup and returned to their original location at 240 seconds afterwards. Therefore, at the same time the highway gets immediately removed. We still used 1 respectively 2 sources located in the centers of cities 3 and 4 sending to the adjacent city. Simultaneously, four packets per second were sent from 60 to 290 seconds of simulation time.

6.4 Metrics

The following four metrics have been chosen for comparison of the protocols:

Packet delivery ratio Packet delivery ratio is calculated by dividing the number of packets received by the destination through the number of packets originated by the application layer of the source (i.e. CBR source). It specifies the packet loss rate, which limits the maximum throughput of the network. The better the delivery ratio, the more complete and correct the routing protocol is.

End-to-end delay End-to-end delay indicates how long it took for a packet to travel from the CBR source to application layer of the destination. It represents the average data delay an application or a user experiences when transmitting data.

Hop count Hop count is the number of hops a packet took until reaching its destination.

Euclidean distance The euclidean distance denotes to the length of the path a packet had been traveling until reaching the destination.

7 Performance Evaluation and Results

This chapter demonstrates the results of about 500 simulations runs, which were performed in different scenario settings. Besides a brief discussion of the diagrams on a per-parameter basis, we also provide a more detailed analysis for every major simulation scenario.

7.1 Small Network Experiment

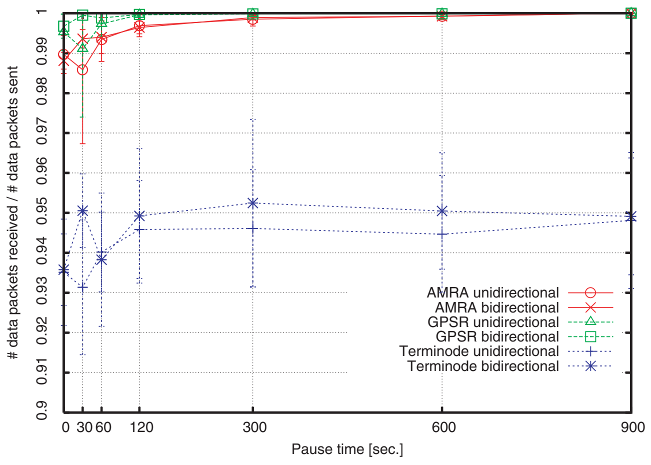
Right from start the AMRA architecture was designated to show weaker performance in this scenario than GPSR for instance. With the random node distribution specified, GPSR is able to forward a packet using greedy mode almost always, thus forwarding packets in a straight line toward the final destination. The overhead of ants and fluctuating pheromone concentrations in AMRA leads to weaker efficiency.

However, the AMRA simulation results show a very robust performance in this scenario. Figure 7.1 depicts the various metrics measured, containing simulations with 1 and 2 sources on the left hand side, and simulations with 20 sources on the right hand side. Concerning the delivery ratio showed in figure 7.1(a) and 7.1(b), AMRA always delivers more than 90% of the packets, even with 20 sources and no mobility pause times. AMRA drops a few more packets than GPSR, especially in the simulation runs with 20 sources and bidirectional traffic. First of all this is due to wrong decisions made by the probabilistic routing for the pheromone concentrations may vary. Secondly, AMRA always forwards packets to geographic relay points (see figure 3.7). Thus, under higher network load, nodes located next to these points get congested and drop packets. Our finding is affirmed by the higher delays showed in figure 7.2(d), which are caused by the queuing. As the mobility decreases (pause time toward 900 seconds) the routes turn out more and more consitent, and the delivery ratio increases to almost 1 in both the GPSR as well as the AMRA graph. Figure 7.2(e) and 7.2(f) present the average hop count experienced by the data packets: AMRA and GPSR both show very similar forms of the average hop counts. Due to the transmission toward the relay points AMRA exhibits a slight higher average hop count. Unfortunately, the Terminode routing protocol (FAPD) is far apart the other protocols. The statistics indicate many packet drops caused by congested queues, what might be a sign of loops. As we are not able to rule out an implementation error, a detailed analysis of the routing protocol needed to be done in order to assess the cause of weak performance.

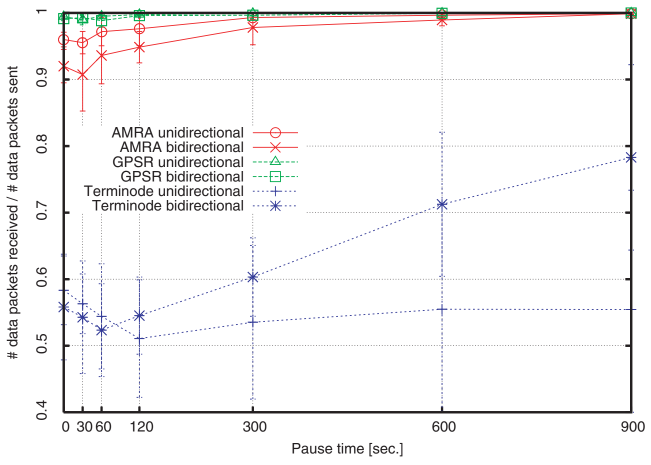
Summarizing the small network scenario we can state that AMRA keeps up very well with GPSR in terms of delivery ratio, end-to-end delay and average hop count. It needs to be reminded that GPSR was able to deliver all packets straightforward without needing to switch to perimeter mode. As one aspect of AMRA's design includes to provide a better solution for the hop consuming perimeter mode, it was not able to show its advantages in this scenario.

7.2 Large Network with Restricted Random Waypoint Mobility

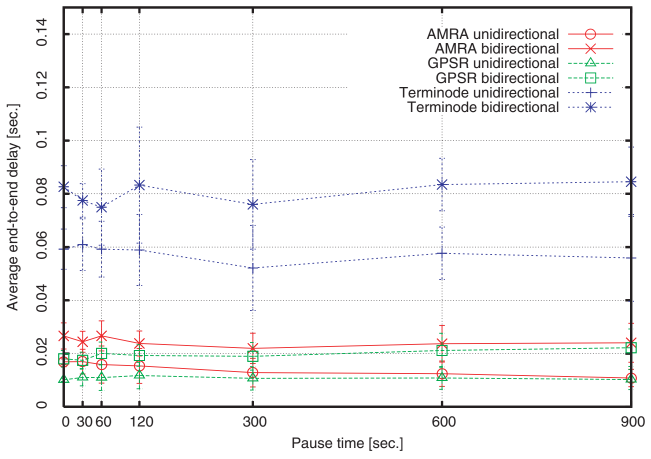
The first AMRA experiments carried out in a large network of 500 nodes using Restricted Random Waypoint Mobility were very disappointing. Even when varying logical router size, frequency of ant



(a) Packet Delivery Ratio, 1500m x 300m, 1 resp. 2 sources.

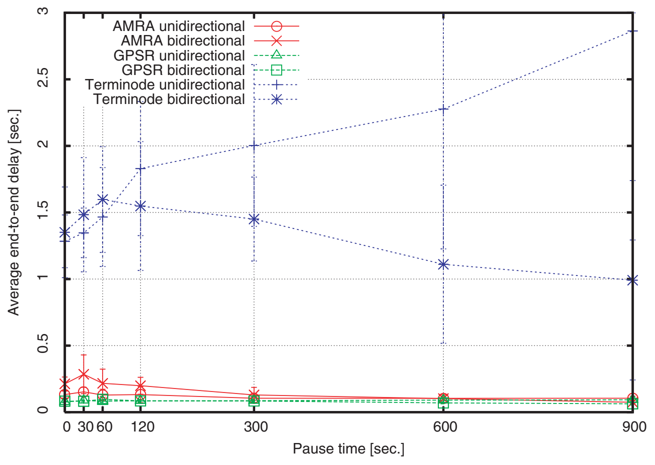


(b) Packet Delivery Ratio, 1500m x 300m, 20 sources.

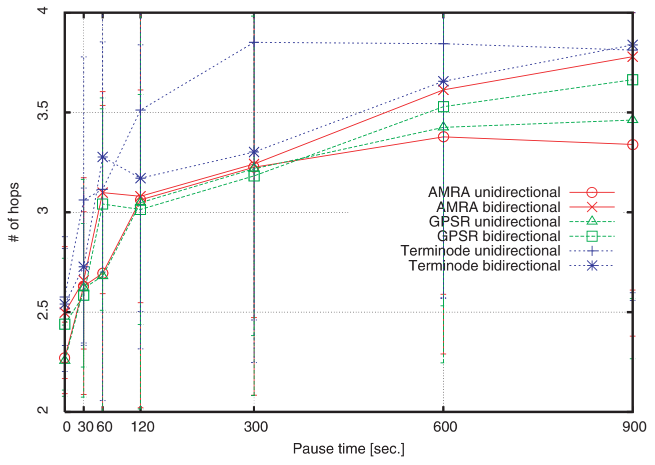


(c) End-To-End Delay, 1500m x 300m, 1 resp. 2 sources.

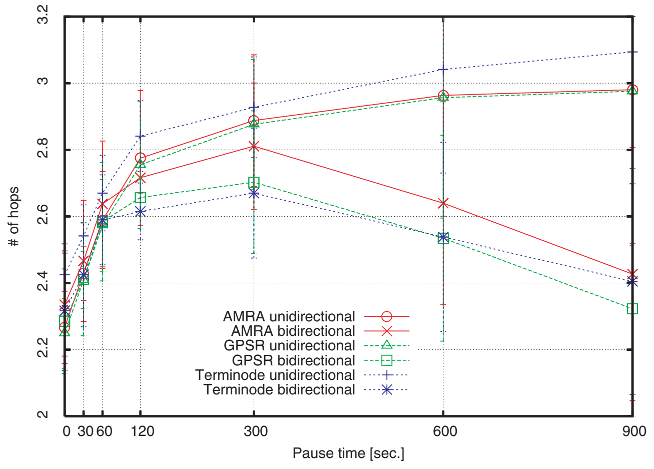
Figure 7.1: AMRA Performance Results in a Small Network with Random Node Distribution.



(d) End-To-End Delay, 1500m x 300m, 20 sources.



(e) Average Hop Count, 1500m x 300m, 1 resp. 2 sources.



(f) Average Hop Count, 1500m x 300m, 20 sources.

Figure 7.1: AMRA Performance Results in a Small Network with Random Node Distribution.

initiation, or the window of collected traffic statistics, no delivery ratio above 10% could be achieved. In some experiments the delivery ratio yet decreased to 2%. The analysis of the experiments' statistic files showed high packet losses due to congested queues and TTL expiry. Jittering the forward ants instead of initiating them all at the same time did not diminish the effect described. After not having found any errors in the AMRA implementation we planned to start the same simulations using the GPSR algorithm. Surprisingly, only approximately 15% of the initiated packets arrived at the destination. The same experiment with restricted node placement and without mobility turned out with a delivery ratio of 100%.

Figure 7.2 illustrates the cause of our findings: In a static network GPSR guarantees the reliable delivery of all data packets (figure 7.2(a)). However, in the particular high mobility scenario we implemented, lots of loops appear. The problem of loops has already been mentioned in the GPSR section on page 12. Sometimes a packet can recover from a loop and still be delivered to the destination node (figure 7.2(b)). Unfortunately, other packets remain caught in the loop or just step to the next loop until the TTL value expires (figure 7.2(c)). As a consequence, the corresponding nodes' queues get congested and start to drop packets. Additionally, due to the high queuing delay now experienced, the destination node moves away farther from the location initially determined whereby the packet can no longer be delivered to the destination.

In our scenario loops mostly occur on highways. With a speed of 10 to 30 meters per second, mobility in these areas is considerable high – specially no static nodes are encountered. Furthermore, the directions of nodes are not equally distributed: nodes rather move in the one or other direction on highways.

Our results show that GPSR is not a suitable routing protocol to use in complex high-mobility environments. Since AMRA relies on the correct function of the lower layer routing protocol it is not able to improve the simulation results. Almost all packets are lost due to curious pheromone concentrations caused by the loops.

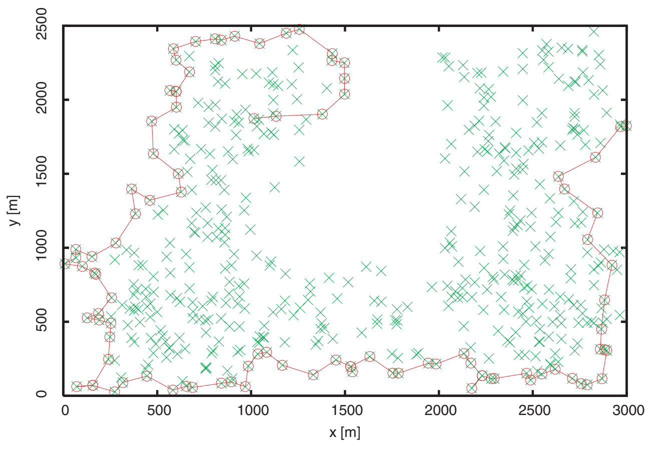
7.3 Large Static Network with Restricted Node Placement

The large static network used in this scenario features the same settings as the previous experiment although it omits mobility. Figure 7.3 presents the simulation results for 1 respectively 2 sources. 'AMRA prob' refers to the usual probabilistic AMRA route selection; 'AMRA max' only considers the logical link with the highest probability. Due to packet transmission from the center of city 3 to the center of city 4 (see figure 6.2(a)) GPSR was obliged to use perimeter mode forwarding.

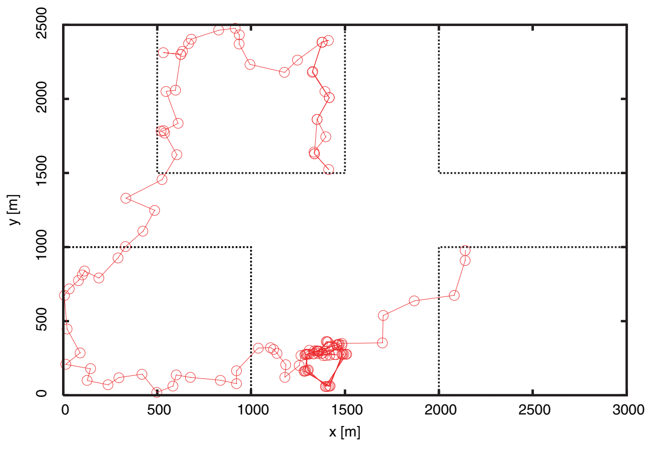
It strikes that AMRA clearly outperforms all other routing protocols in terms of average hop count, end-to-end delay, and average euclidean distance. Compared to GPSR the hop count of AMRA undershoots half the GPSR value (figure 7.3(a)). Also, we can assess substantial shorter trip times (figure 7.3(b)) and a shortened euclidean path (figure 7.3(c)). Remember that GPSR always forwards packets using short hops in perimeter mode. Thus the proportions of hop count difference and euclidean path difference do not match: GPSR is taking many small hops while AMRA is using fewer larger hops. Note that all protocols profit from the reverse path in the bidirectional scenario which is much easier and shorter to follow according to the right-hand rule.

The Terminode routing protocol fails to transmit nodes from city 3 to city 4. However, it reaches a delivery ratio of 50% in the bidirectional experiment as a result of the sucessful arrival of packets from the opposite direction. Through the sole use of this simple path a low hop count as well as a low euclidean distance can be achieved.

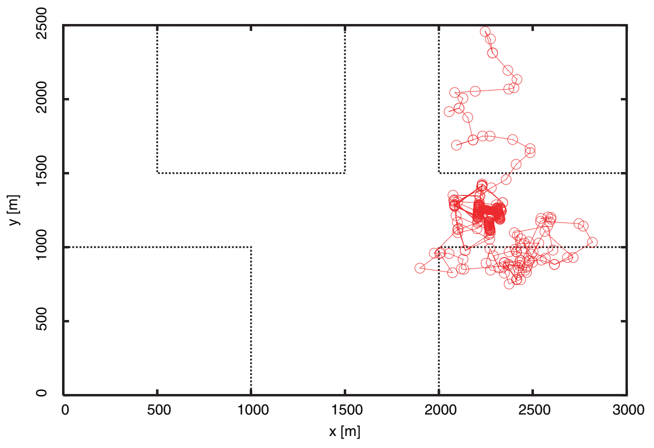
We were also interested in the behavior of AMRA if only the most probable logical link is consid-



(a) GPSR guarantees packet delivery in a static network: example of perimeter mode forwarding.

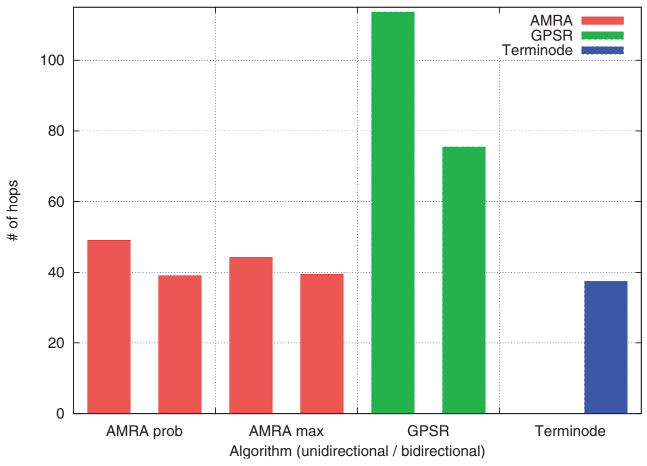


(b) Due to the high mobility on highways GPSR is no longer safe from loops and suffers from high packet losses.

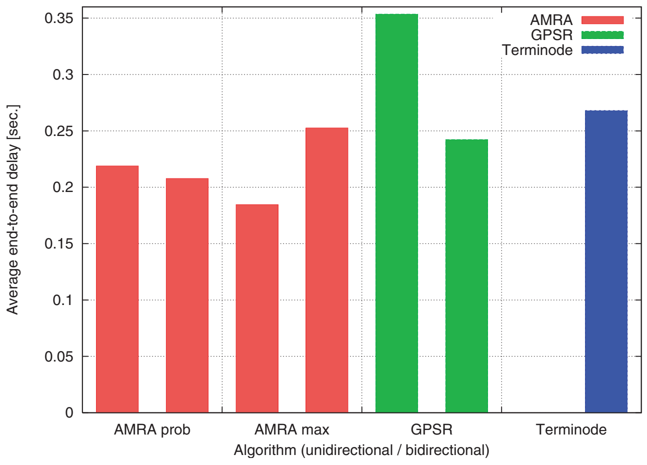


(c) Another example of a GPSR loop where the protocol fails to recover.

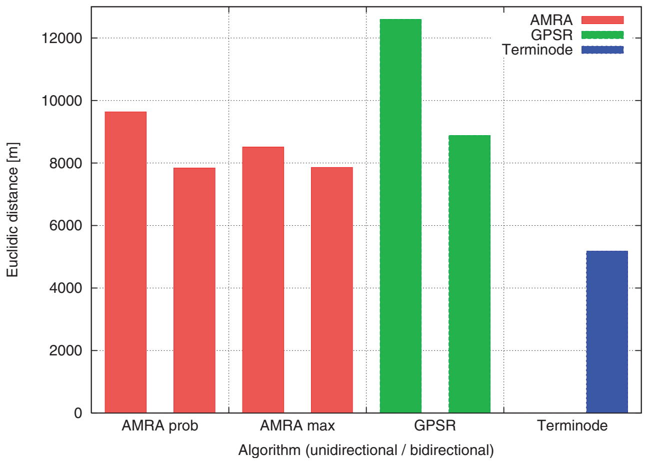
Figure 7.2: Loop problems that arise in GPSR in high mobility scenarios.



(a) Average Hop Count, 3000m x 2500m, static, 1 resp. 2 sources.

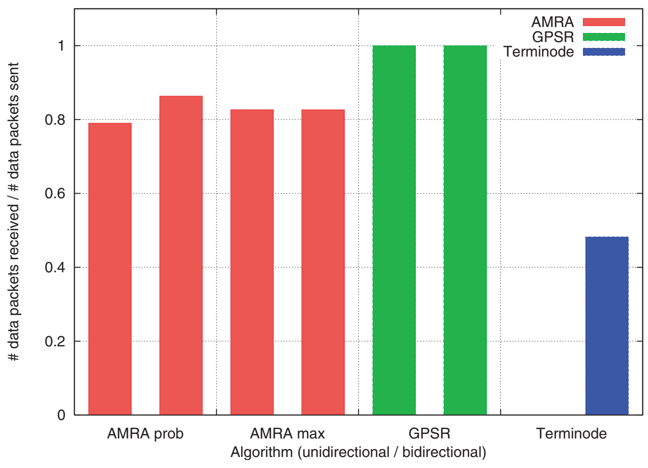


(b) Average End-To-End Delay, 3000m x 2500m, static, 1 resp. 2 sources.



(c) Average Euclidean Distance, 3000m x 2500m, static, 1 resp. 2 sources.

Figure 7.3: AMRA Performance Results in a Static Large Network with Restricted Node Distribution.



(d) Packet Delivery Ratio, 3000m x 2500m, static, 1 resp. 2 sources.

Figure 7.3: AMRA Performance Results in a Static Large Network with Restricted Node Distribution.

ered to relay a packet (AMRA max). Our findings show that hop count and euclidean distance can be reduced once more or at least remain at an equal level. In the case of bidirectional traffic packet paths are aggregated to identical geographical regions. This effect is even boosted by the use of the same logical routers all the time. Therefore, the queuing time as well as the trip time increases (see figure 7.3(b)).

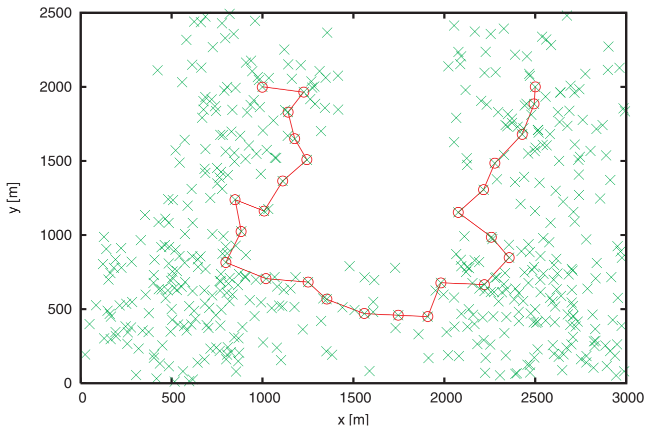
Figure 7.4(d) shows a worse delivery ratio of AMRA compared to GPSR. Regardless of the scenario used, the delivery ratio amounts to about 80%. Further analysis of the statistics file showed a number of packet drops due to TTL expiry as well as a quite high variance of hop count. Subsequent plots of different packet paths indicated the problem of loops that occur in AMRA due to inaccurate pheromone concentrations and due to the interaction with GPSR. Although AMRA explores good paths (figure 7.4(a)), routes are occasionally fluctuating and looping. For instance, in figure 7.4(c), the packet is correctly handed down on the left hand side according to the route probabilities. Between cities 1 and 2 it then encounters a node which is not able to provide a valid logical router (e.g. due to zone locking or due to low probabilities). Thus, the packet is handed to GPSR, entering perimeter mode and following all the path back to the initial city. The loops experienced effect a lower delivery ratio and high hop count fluctuations. Another example of a loop is presented in figure 7.4(b).

To sum up, this scenario demonstrates the high potential of the AMRA architecture in order to avoid perimeter mode forwarding and to deliver packets on fewer hops in shorter time. Although loops of that size are very difficult to detect, the problem needs to be solved in order to achieve higher delivery ratios.

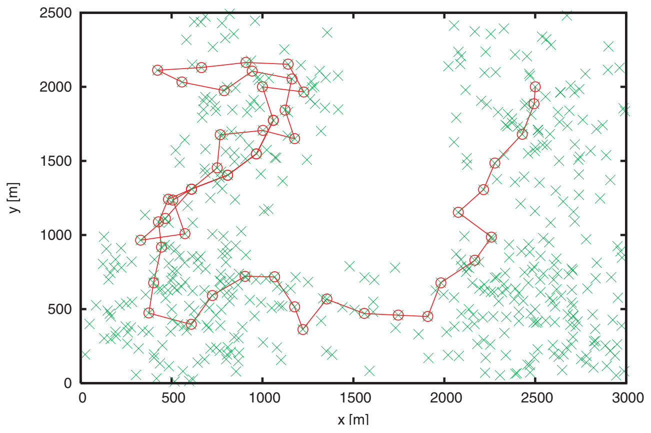
7.4 Radical Topology Changes in a Large Static Network

The last performance test applied to radical changes in the network topology. The two upper graphs – 7.5(a) and 7.5(b) – illustrate the creation of an additional highway at 240 seconds, the lower graphs – 7.5(c) and 7.6(d) – refer to the removal of the same highway. The hop count is plotted as a function of simulation time; every line refers to a specific simulation run.

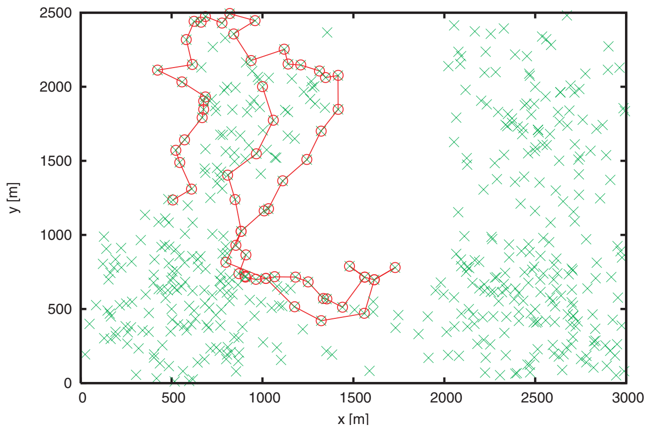
At first glance the high hop count variation during the AMRA experiments can be realized. Also,



(a) AMRA finds a good path from the source to the destination mostly.

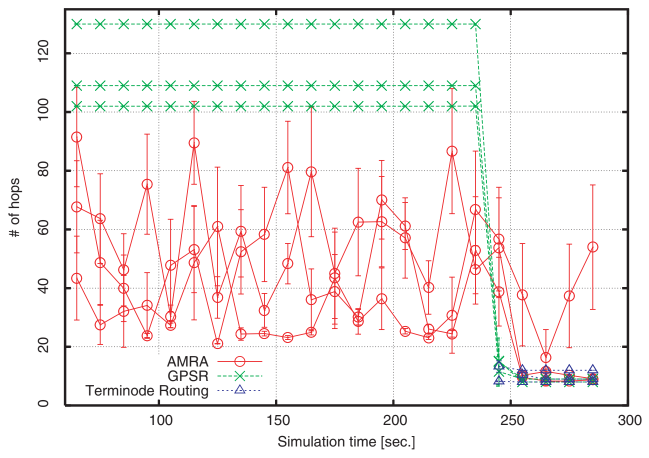


(b) Due to the interaction with GPSR, large loops may occur, which effect a lower delivery ratio and high hop count fluctuations.

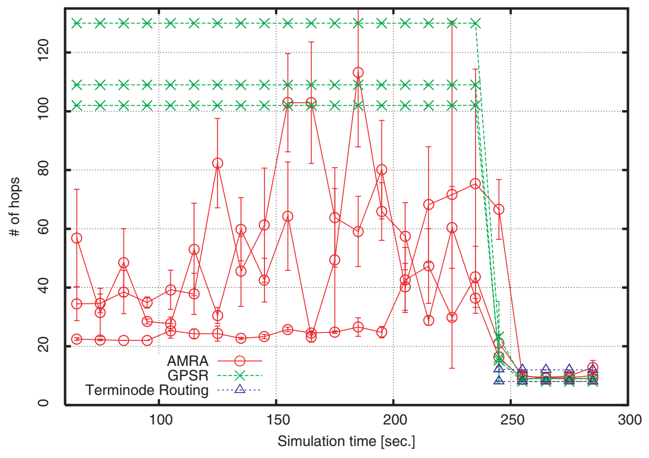


(c) Another example of loops in a AMRA route plot.

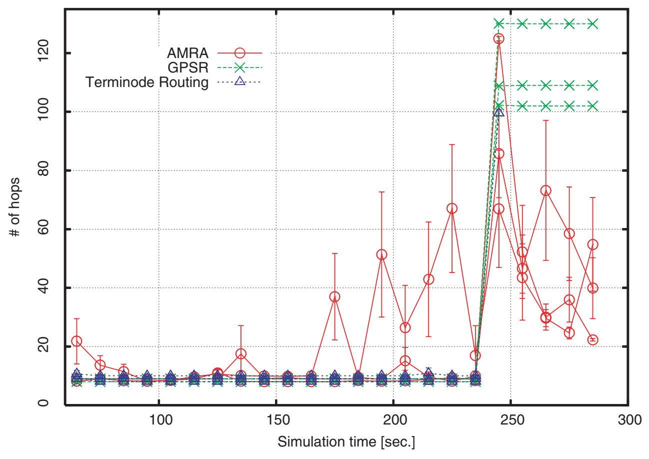
Figure 7.4: AMRA Route Path Examples.



(a) Hop count, 3000m x 2500m, static, 1 unidirectional source, link insertion after 240 seconds.

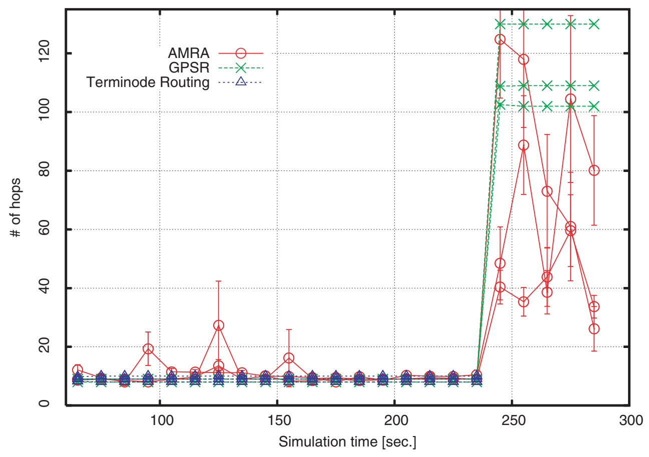


(b) Hop count, 3000m x 2500m, static, 2 bidirectional sources, link insertion after 240 seconds.



(c) Hop count, 3000m x 2500m, static, 1 unidirectional source, link removal after 240 seconds.

Figure 7.5: Reaction of the AMRA architecture to major topology changes.



(d) Hop count, 3000m x 2500m, static, 2 bidirectional sources, link removal after 240 seconds.

Figure 7.5: Reaction of the AMRA architecture to major topology changes.

AMRA typically reaches the destination in fewer hops than GPSR does. The routes fluctuate quite highly due to both the changing concentrations of pheromones and the probabilistic routing decisions. Even after insertion of the new highway some packets still follow the old path in one unidirectional experiment (figure 7.5(a)). Most likely no forward ant had yet announced the presence of the new path to the originating node. In the bidirectional scenario (figure 7.5(b)), the new highway is quickly detected and used to propagate subsequent data packets. GPSR quickly notices the new neighbors through the use of broadcast packets and immediately forwards packets along the new trail. AMRA requires more time for pheromone reinforcement to the new path but finally reaches comparable hop counts. Terminode Routing fails to propagate packets from city 3 to city 4 as stated earlier.

Even though the second scenario features a short path at the beginning, some AMRA packets still follow the longer trail (figure 7.5(c) and 7.6(d)). This effect is a direct consequence of the probabilistic routing performed. The forwarding of packets is split among all outgoing links according to their probabilities. As soon as the highway disappears at 240 seconds GPSR performs an immediate transition to the longer path by eliminating failing nodes from the neighbor tables. AMRA first experiences rather high hop counts until the 'backup path' is properly reinforced. Except to Terminode Routing all protocols manage to successfully establish the new path. AMRA shows the route fluctuation behavior already experienced.

8 Summary and Outlook

The performance evaluations carried out in this thesis have attested the high potential of the Ants-Based Mobile Routing Architecture. Particularly, compared to GPSR, it achieves to substantially reduce both hop count and trip time in a complex network topology. Further, it performs very well in a small mobile ad-hoc network using the random waypoint mobility model. Our investigations also highlight the protocol's ability to sufficiently react on radical topology changes.

The simulation results and the discussion presented in the previous chapter allows us to arrive at the following conclusions about the AMRA architecture:

1. **GPSR is not a suitable routing protocol for the lower layer.** As described in section 7.2 GPSR lacks the robustness against loops specially with increasing mobility. Looping forward ants consequently lead to erroneous pheromone distributions, which prevent AMRA from delivering data packets successfully. One possible solution is to replace GPSR with a more sophisticated routing protocol, which is proved to be stable in high mobility environments. For instance, the Beacon-Less Routing Algorithm (BLR) [43] claims to perform well in highly dynamic networks and to be robust against topology changes.

2. **Relaying toward fixed geographic points is critical.** On page 27 we described the forwarding of packets to specific fixed points within a logical router according to the destination's location. This behavior effects an aggregation of the packet flow toward a number of small geographic regions. Nodes located around such a relay point suffer from processing a substantial amount of packets that derive from their logical router. Thus, if the network load increases, this leads to long queuing times and packet drops.

3. **Interaction with GPSR requires some loop detection.** Figure 7.4(c) has very well illustrated the emergence of loops caused by the MABR protocol. Loops preferably occur due to the alternating use of the two routing protocols and due to inaccurate probabilities. Their detection represents a quite complex task as they cover large geographic regions. In order to avoid these loops a loop detection mechanism needs to be introduced in the AMRA architecture. For instance, if a node fails to provide a logical router for the next hop, the number of routing retries performed by the lower layer routing protocol might be limited. Further, a better arrangement of pheromones effects more accurate route decisions and thus prevents loops.

4. **Reinforcement needs to be reviewed.** The AMRA algorithm currently shows quite high path fluctuations, which essentially appear on long and complex trails. As we described in section 7.4, these occur both in unidirectional and bidirectional traffic models. Unstable pheromone values quickly loosing valuable path information may be the cause of such behavior. An analysis of the routing probability values as a function of simulation time would be an interesting subject for further investigations. If the probabilities turn out to be unsteady a reconsideration of the current reinforcement method is required.

A companion thesis [44] covering the same routing architecture was performed using a simple, Java-based network simulator. In contrary to QualNet this simulator does not provide detailed quantitative simulation results since it sets several aspects aside: physical, MAC layer, propagation and queuing characteristics are not considered in the simulations. Therefore, an analysis of experienced trip times cannot be performed, i.e. packets arrive at their destination without delay. Also, no node movement is occurring during packet transition. This eliminates the appearance of GPSR loops similarly as in our static scenarios. The simplicity of this simulator is reflecting in a remarkable simulation performance, which allows to evaluate the behavior of a plenty of mobile nodes in relative short times. The focus of the thesis mentioned was set on the examination of AMRA in large-scale mobile ad-hoc networks in the sense of a proof-of-concept. Thanks to the fast simulation times, a variety of parameters were tested and adjusted, such as the number of ants initiated, logical router size and reinforcement parameters. The AMRA results of both this work and the companion thesis point in the same direction: Compared to GPSR, hop counts and euclidean distances can remarkably be decreased in complex network topologies. Also, the network is able to adequately react on link changes after short time. We were not able to reproduce the same stable routes and high delivery ratios. This might be a cause of the simplified simulation model in the Java-based simulator, but needs further verification.

Unfortunately, we were not able to evaluate all aspects of the AMRA routing architecture in this thesis. The simulation experiments require considerable amounts of processing time limiting the scope of this work to a basic performance comparison. In addition, AMRA can profit from future improvements. The following list contains numerous aspects which require further investigations.

- In our simulations, an activity-based ant initiation scheme with a time interval of five seconds was presumed. It would be interesting to assess the influence of different ant initiation rates on the routing performance.
- The optimal size of logical routers should be verified through additional simulations. The influence of different clustering techniques in order to group nodes to logical routers might be considered as well. In cellular networks regular hexagons are utilized quite often.
- The use of backward ants was not included in our simulations. However, they may be useful if a node is not aware of a forward path toward the destination. Our proposal suggests to initiate a forward ant to the unknown region on-demand and to request a backward ant in order to learn about the new route.
- The current implementation of AMRA initiates forward ants to randomly selected logical routers, even including empty regions. As soon as no node lying closer to the destination can be determined, the packet loops around the empty area until the TTL value expires. This costs bandwidth, energy and falsifies the route probabilities. An improvement addressing to this problem would also be of interest.
- Logical routers are thought to contain nodes essentially sharing the same routing information. At the time being, no synchronization of routing tables between the nodes inside a logical router is performed. The development of a distributed routing table is a very challenging task for it needs to cope with nodes leaving the area or node failures. Further, the advantage of more accurate routing tables needs to be weighed up against the increasing routing overhead caused by the synchronization messages.
- AMRA does currently not implement any mechanism to deal with position inaccuracies. If the destination node can not be found nearby the expected location a data packet is simply dropped. The Terminode routing protocol suggests to use Restricted Local Flooding (RLF) in

order to solve this problem (see page 13). Probably it would be a reasonable idea to add this functionality to AMRA as well.

Glossary

802.11 802.11 is an evolving family of specifications for wireless local area networks (WLANs) developed by a working group of the Institute of Electrical and Electronics Engineers (IEEE).

ABC (Ant-Based Control) Ant-Based Control is a swarm-based routing algorithm designed for telephone networks. In order to explore the network ABC uses a single class of ants, which are initiated at regular time intervals from every source to a randomly chosen destination.

AGPF (Anchored Geodesic Packet Forwarding) A greedy-based forwarding technique used in the Terminode routing protocol, which uses fixed geographic points (anchors) to propagate packets.

AMRA (Ants-Based Mobile Routing Architecture) A two-layered swarm intelligence based routing architecture which includes topology abstraction and position-based routing. It makes use of the MABR protocol for routing decisions on the upper layer and of a straight packet forwarding protocol on the lower layer.

AODV (Ad-Hoc On-Demand Distance Vector Routing) A reactive routing protocol using route request and route reply packets to discover new routes. Route requests are flooded through the entire network.

API (Application Programming Interface) An API allows software developers to access the functionality of prebuilt software modules. An API defines data structures and subroutine calls.

ARA (The Ant-Colony Based Routing Algorithm) ARA proposes a detailed swarm intelligence based routing scheme for mobile ad-hoc networks. Ants are only initiated on demand and are flooded through the entire network in a similar process as AODV.

CBR (Constant Bit Rate) A CBR source transmits UDP packets across the network using a constant bit rate. In many simulation experiments 64-byte packets are initiated in regular time intervals.

DCF (Distributed Coordination Function) The Distributed Coordination Function forms the basis of standard CSMA/CA (Carrier Sensing Multiple Access / Collision Avoidance) access within an 802.11 wireless network. It first checks to see if the radio link is free before transmitting and to avoid contention, initiates a random backoff.

DSR (Dynamic Source Routing) A reactive routing protocol designed for smaller mobile ad-hoc networks, which is based on source routing. DSR makes use of route request and route reply packets to determine unknown routes and additionally employs rather aggressive route caching methods.

EUI (End-system Unique Identifier) A number to uniquely identify a network host.

FAPD (Friend Assisted Path Discovery) A method used to discover anchored paths toward a destination node, which is employed in Terminode Routing. Thus, a node maintains a list of friends queried in order to find an anchored path to the destination. FAPD relies on the concept of small world graphs [22].

FIFO (First In First Out) FIFO is a term used to describe routing buffers or queues. It states that packets are treated in the order that they are received by the router respectively the queue.

GFG (Greedy Face Greedy) A position-based routing protocol combining greedy and perimeter mode forwarding. A node always tries to relay a packet to the neighbor, which lies closest to the destination. If greedy forwarding fails GFG achieves to forward the packet around the perimeter by the use of the right-hand rule as well as planar graph traversal methods.

GMPD (Geographic Maps-based Path Discovery) Another method for discovering anchored paths used in Terminode Routing. In this algorithm every mobile node receives a map of the entire network topology and thus determines anchor points.

GPF (Geodesic Packet Forwarding) A forwarding algorithm employed in Terminode Routing, which is similar to GFG or GPSR.

GPS (Global Positioning System) GPS comprises of multiple satellites all of which orbit the earth twice a day. Users with a GPS receiver use timing information from the satellites in order to triangulate their position on the earth surface.

GPSR (Greedy Perimeter Stateless Routing) A position-based routing protocol similar to GFG. Some MAC layer feedback enhancements were added in this approach in order to detect wrong neighbor table entries more quickly.

IEEE (Institute of Electrical and Electronics Engineers) The Institute of Electrical and Electronics Engineers was formed in the late 1800s as an organization of technical professionals. Nowadays, it is very active in the development of technical specifications and standards, including many of the technologies used to build the infrastructure of the internet.

IARP (Intrazone Routing Protocol) A simple proactive link-state routing protocol required in the Zone Routing Protocol to perform routing tasks within a local zone. Often two hop neighborship is considered for a zone.

IERP (Interzone Routing Protocol) Using the reactive IERP protocol, the Zone Routing Protocol achieves routing between distant zones.

IP (Internet Protocol) Defines network layer packets and procedures used to move datagrams from host to host; currently, Version 4 (IPv4) is standard.

Java A high-level, object oriented programming language developed by Sun Microsystems. It runs on top of a virtual machine and is highly portable to almost any computer platform therefore.

LDA (Location-Dependent Address) The location-dependent address in Terminode Routing consists of a triplet of longitude, latitude and altitude.

LL (Logical Link) An abstraction introduced in the AMRA architecture, which connects a logical router with its adjacent zones. Packets tend to follow logical links in a roughly straight line, possibly over multiple hops.

LR (Logical Router) A topology abstraction required by the MABR routing algorithm. Therefore, mobile nodes located in a square geographical region are grouped together and build a logical router.

MABR (Mobile Ants-based Routing) An ants-based routing protocol performing routing decisions on top of an abstracted network topology. MABR requires a straight packet forwarding protocol at the lower layer to relay the data packets.

MAC layer (Media Access Control) In the OSI model of communication, the Media Access Control layer is one of two sublayers of the Data Link layer. MAC handles access to a shared medium such as Ethernet for instance.

MANET (Mobile Ad-Hoc Network) A collection of wireless mobile nodes forming a temporary network without using any centralized access point, infrastructure, or centralized administration. To establish a data transmission between two nodes, typically multiple hops are required due to the limited transmission range of the nodes.

Perl Perl is a script programming language that is similar in syntax to the C language. It is an interpreted language and provides powerful methods for list processing, making it suitable to post-process the output of simulation experiments.

QualNet® is a discrete event simulator developed by Scalable Network Technologies. The QualNet Developer Suite consists of several tools, which allow to build models and simulate wired and wireless networks.

RLF (Restricted Local Flooding) A local flooding technique proposed in Terminode Routing, which transmits copies of a packet to six geographical regions.

RREP (Route Reply) A route reply packet informs the initiator about the recently detected new route.

RREQ (Route Request) A route request packet is initiated in order to detect a route to a destination.

SNMP (Simple Network Management Protocol) SNMP is a protocol used to perform network management and allows the monitoring of network devices and their functions.

SNT (Scalable Network Technologies) The developer of QualNet® network simulation software.

Terminode Routing A hybrid routing protocol relying on anchored paths to route data packets. It makes use of → *TLR* to perform local routing, and of → *TRR* for remote routing.

TLR (Terminode Local Routing) A simple two hop link-state routing protocol required in Terminode Routing.

TRR (Terminode Remote Routing) A routing protocol used in Terminode Routing to deliver packets to distant nodes. See → *AGPF*.

TTL (Time To Live) The TTL value specifies the lifetime of a data packet. At each hop the counter is decreased by one. If it reaches zero, the packet is dropped.

WLAN (Wireless Local Area Network) A generic term covering multiple technologies providing local area networking via a radio link. Examples of WLAN technologies include 802.11a, 802.11b, Bluetooth and DECT (Digital Enhanced Cordless Telecommunications).

ZRP (Zone Routing Protocol) ZRP is a hybrid routing protocol distinguishing between routing within limited zones (proactive) and routing among those zones (reactive).

Appendix

Structure of an AMRA Forward Ant

```
typedef struct
{
    /* the packet type 0..3 (2 bits) */
    uint8_t pktType : 2;
    /* we may request an optional backward ant */
    uint8_t requestBackwardAnt : 1;
    /* a flag if the backward ant has already been sent */
    uint8_t backwardAntSent : 1;
    /* unused */
    uint8_t unused : 4;
    NodeAddress srcAddr;
    LOG_RT_ADDR destRtAddr;
    uint16_t seqNumber;
    uint16_t ttl;
    /* the coordinates of the source node, required to determine source zone */
    float srcpos_x;
    float srcpos_y;
    /* the first logical router we relayed to, required for backward ant */
    LOG_RT_ADDR firstLogRtId;
    /* the last logical router that was passed, required for pheromone update */
    LOG_RT_ADDR lastLogRtId;
    float lastLogRtPos_x;
    float lastLogRtPos_y;
    /* timestamp when the packet was initiated */
    clocktype srcSentTimestamp;
} MABR_FWD_ANT_Packet;
```

Structure of an AMRA Backward Ant

```
typedef struct
{
    /* the packet type 0..3 (2 bits) */
    uint8_t pktType : 2;
    /* flag that no convenient logical router zone was available for
     * relay and the packet was sent only using GPSR */
    uint8_t noRelayZoneAvailable : 1;
    /* unused */
    uint8_t unused : 5;
    NodeAddress srcAddr;
    NodeAddress destAddr;
    uint16_t seqNumber;
    uint16_t ttl;
    float srcpos_x;
    float srcpos_y;
    float destpos_x;
    float destpos_y;
    LOG_RT_ADDR lastLogRtId;
    float lastLogRtPos_x;
    float lastLogRtPos_y;
    /* timestamp when the packet was initiated */
    clocktype srcSentTimestamp;
    clocktype fwdAntDelay;
    uint16_t fwdAntHopCount;
    LOG_RT_ADDR fwdAntFirstRelay;
} MABR_BWD_ANT_Packet;
```

Structure of the MABR Header for Data Packets

```
typedef struct
{
    /* the packet type 0..3 (2 bits) */
    uint8_t pktType : 2;
    /* flag that no convenient logical router zone was available for
     * relay and the packet was sent only using GPSR */
    uint8_t noRelayZoneAvailable : 1;
    /* unused */
    uint8_t unused: 5;
    unsigned char originalIpProtocol;
    NodeAddress srcAddr;
    NodeAddress destAddr;
    uint16_t seqNumber;
    uint16_t ttl;
    float srcpos_x;
    float srcpos_y;
    float destpos_x;
    float destpos_y;
    LOG_RT_ADDR lastLogRtId;
    float lastLogRtPos_x;
    float lastLogRtPos_y;
    clocktype srcSentTimestamp;
    /* variables requires for statistic purposes */
    float euclDist;
} MABR_DATA_Packet;
```

Structure for the Local Variable Space of AMRA

```
typedef struct struct_network_mabr_str
{
    /* seed for the TAP/MABR algorithm */
    unsigned short seed[3];
    /* route table */
    MABR_RT routeTable;
    /* struct for statistics */
    MABR_Stats stats;
    /* table for ants already seen */
    MABR_ANT_ST antSeenTable;
    /* the current sequence number of the node for protocol packets*/
    int seqNumber;
    /* pointer to TAP which stores information about topology abstraction */
    MABR_TAP *tap;
    MABR_ANT_INITIATE_MODE antInitiateMode;
    MABR_LOOP_DETECTION_MODE loopDetectionMode;
    clocktype antInitiateInterval;
    clocktype antInitiateJitter;
    clocktype firstAntAfter;
    clocktype antOverhearTime;
    LOG_RT_ADDR currentLogicalRouter;
    /* information about the lower layer routing protocol */
    NetworkRoutingProtocolType sublayerRoutingMode;
    void *sublayerRouting;
    RouterFunctionType sublayerRouterFunction;
    FinalizeFunctionType sublayerFinalizeFunction;
    MacLayerStatusEventHandlerFunctionType sublayerMacLayerStatusEventHandlerFunction;
    PromiscuousMessagePeekFunctionType sublayerPromiscuousMessagePeekFunction;
    unsigned int gpsrAntRoutingMode;
    clocktype enhancedStatsInterval;
} MabrData;
```

City Placement File for Restricted Random Waypoint Mobility

```
#
# CITY-PLACEMENT-FILE
# (only used for Restricted Random Waypoint Mobility)
#
# Format (retcangular city):
# cityId x_min y_min width height create_time erase_time
#
# Format (circular city):
# cityId x_center y_center radius 0 create_time erase_time
#
# IMPORTANT: Cities get removed with all corresponding links, but
# cities get created without any links. Please use the link definition
# file for creating and removing links
#

1    0    0 1000 1000 0S 900S
2 2000    0 1000 1000 0S 900S
3  500 1500 1000 1000 0S 900S
4 2000 1500 1000 1000 0S 900S
```

Link Configuration File for Restricted Random Waypoint Mobility

```
#
# LINK-CONF-FILE
# (only used for Restricted Random Waypoint Mobility)
#
# Format (link insert):
# INS <link from> <link to> <create time>
#
# Format (link delete):
# DEL <link from> <link to> <erase time>
#
# <link from> and <link to> refers to a cityId (see CITY-PLACEMENT-FILE)
#
# IMPORTANT: This file corresponds to the CITY-PLACEMENT-FILE. Cities
# get removed with all corresponding links, but cities get created
# without any links. To initialize cities with links, please use 0S as
# the create time.
#

INS 1 2 0S
INS 2 1 0S
INS 1 3 0S
INS 3 1 0S
INS 2 4 0S
INS 4 2 0S
INS 3 4 720S
INS 4 3 720S
```

Bibliography

[1] C. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” in *Proceedings of the ACM SIGCOMM’94 Conference on Communications Architectures, Protocols and Applications*, London, UK, August 1994, pp. 234–244.

[2] C. Perkins, E. Belding-Royer, and S. Das, “RFC 3561: Ad hoc on-demand distance vector (AODV) routing,” July 2003, category: experimental. [Online]. Available: ftp://ftp.isi.edu/in-notes/rfc3561.txt

[3] C. E. Perkins and E. M. Royer, “Ad-hoc on demand distance vector routing,” in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’99)*, vol. 3, New Orleans, LA, USA, February 1999, pp. 90–100.

[4] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353, ch. 5, pp. 153–181.

[5] Z. Haas, M. Pearlman, and P. Samar, “The zone routing protocol (ZRP) for ad hoc networks,” July 2002, IETF Internet Draft, draft-ietf-manet-zone-zrp-04.txt.

[6] J. Schaumann, “Analysis of the zone routing protocol,” December 2002. [Online]. Available: http://www.netmeister.org/misc/zrp/zrp.pdf

[7] Z. J. Haas, M. R. Pearlman, and P. Samar, “The bordercast resolution protocol (BRP) for ad hoc networks,” Internet Engineering Task Force (IETF), July 2002, IETF Internet Draft, draft-ietf-manet-zone-brp-02.txt. [Online]. Available: http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-brp-02.txt

[8] S. Basagni, I. Chlamatac, V. Syrotiuk, and B. Woodward, “A distance routing effect algorithm for mobility (DREAM),” in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Dallas, TX, USA, 1998, pp. 76–84.

[9] Z. J. Haas and B. Liang, “Ad hoc mobility management with uniform quorum systems,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 228–240, April 1999.

[10] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, “A scalable location service for geographic ad hoc routing,” in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, USA, 2000, pp. 120–130.

[11] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. S. J. De Couto, “Carnet: A scalable ad hoc wireless network system,” in *Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System*, Kolding, Denmark, September 2000.

[12] "The grid ad hoc networking project homepage," 2004. [Online]. Available: http://www.pdos.lcs.mit.edu/grid/

[13] S. Giordano and M. Hamdi, "Mobility management: The virtual home region," Institute for Computer Communications and Applications (ICA), Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland, Tech. Rep., March 2000. [Online]. Available: http://icwww.epfl.ch/publications/documents/IC_TECH_REPORT_199937.pdf

[14] I. Stojmenovic, "Home agent based location update and destination search schemes in ad hoc wireless networks," Computer Science, SITE, Univerity of Ottawa, Canada, Tech. Rep., September 1999.

[15] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, USA, August 2000, pp. 243–254.

[16] L. Blažević, "Scalable routing protocols with applications to mobility," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2002.

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

[18] G. T. Toussaint, "The relative neighborhood graph of a finite planar set," *Pattern Recognition*, vol. 12, no. 4, pp. 261 – 268, 1980.

[19] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, no. 6, pp. 609–616, 2001.

[20] L. Blažević, S. Giordano, and J.-Y. L. Boudec, "Self organized terminode routing," *Cluster Computing*, vol. 5, no. 2, pp. 205–218, 2002.

[21] L. Blažević, J.-Y. L. Boudec, and S. Giordano, "A scalable routing scheme for self-organized terminode network," in *Proceedings of Communication Networks and Distributed systems modeling and Simulation conference (CNDS)*, San Antonio, Texas, USA, January 2002.

[22] D. J. Watts, *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, October 1999.

[23] P. P. Grassé, "La theorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs," pp. 41–83, 1959.

[24] G. D. Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, 1998.

[25] R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz, "Ant-based load balancing in telecommunications networks," *Adaptive Behavior*, no. 5, pp. 169–207, 1996.

[26] M. Günes, U. Sorges, and I. Bouazizi, "ARA – the ant-colony based routing algorithm for manets," in *Proceedings of the ICPP Workshop on Ad Hoc Networks (IWAHN 2002)*. IEEE Computer Society Press, 2002.

[27] M. Roth and S. Wicker, "Termite: Emergent ad-hoc networking," in *Second Mediterranean Workshop on Ad-Hoc Networks (MED-HOC NET 2003)*, Mahdia (Tunisia), 2003.

[28] S. Streltsov and P. Vakiki, "Variance reduction algorithms for parallel replicated simulation of uniformized markov chains," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 6, pp. 159–180, 1996.

[29] M. Dorigo, V. Maniezzo, and A. Colorni, “Positive feedback as a search strategy,” Dipartimento di Elettronica e Informatica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.

[30] M. Dorigo, “Optimization, learning and natural algorithms,” Ph.D. dissertation, Dipartimento di Elettronica e Informatica, Politecnico di Milano, Italy, 1992.

[31] M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[32] M. Doringo and L. M. Gambardella, “Ant colony system: A cooperative learning approach to the travelling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[33] R. Schoonderwoerd, O. E. Holland, and J. L. Bruten, “Ant-like agents for load balancing in telecommunications networks,” in *Proceedings of the First International Conference on Autonomous Agents*. ACM Press, 1997, pp. 209–216.

[34] M. Heissenbüttel and T. Braun, “Ants-based routing in large scale mobile ad-hoc networks,” in *Kommunikation in verteilten Systemen (KiVS03)*, Leipzig, Germany, February 2003.

[35] UCLA Parallel Computing Laboratory, University of California, “GloMoSim Scalable Mobile Network Simulator,” Software Package, December 2000. [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/

[36] ——, “Parsec, Parallel Simulation Environment for Complex Systems,” Software Package, October 1998. [Online]. Available: http://pcl.cs.ucla.edu/projects/parsec/

[37] Scalable Network Techologies, “Qualnet simulator,” Software Package, 2003. [Online]. Available: http://www.qualnet.com

[38] Information Sciences Institute, University of Southern California (USC), “NS-2 network simulator,” Software Package, 2004. [Online]. Available: http://www.isi.edu/nsnam/ns/

[39] “The Virtual Internet Testbed (VINT) Project,” USC/ISI, Xerox PARC, LBNL, and UC Berkeley, 1997. [Online]. Available: http://www.isi.edu/nsnam/vint/

[40] OPNET Technologies, Inc., “OPNET network simulator,” Software Package, 2004. [Online]. Available: http://www.opnet.com/

[41] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” in *Proceeedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM’98)*, October 1998, pp. 85–97.

[42] J. Yoon, M. Liu, and B. Noble, “Random waypoint considered harmful,” in *Proceedings of INFOCOM*, 2003.

[43] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, “BLR: Beacon-less routing algorithm for mobile ad-hoc networks,” *Elsevier’s Computer Communications Journal*, vol. 27, no. 11, pp. 1076–1086, July 2004.

[44] T. Huber, “Ant-based mobile routing architecture in large-scale mobile ad-hoc networks,” Master’s thesis, Institute of Computer Science and Applied Mathematics (IAM), University of Bern, Switzerland, 2004.