Routing in Ad Hoc Networks with mobile Users
Master Thesis

MA 2006-02

Bernhard Distl

Supervisor: Prof. Dr. Bernhard Plattner
Advisors: Vincent Lenders, Dr. Martin May

Computer Engineering and Networks Laboratory (TIK)
Department of Information Technology and Electrical Engineering
ETH - Swiss Federal Institute of Technology Zürich

2nd June 2006

Abstract

Routing in ad hoc networks is a challenging task. Recent discoveries show that transmission errors and errors caused by moving nodes have different characteristics. A probability model is available that differentiates transmission errors from errors caused by moving nodes. This thesis researches the possible use a routing protocol could make of this knowledge.

Several ways of applying the probabilistic model to a real protocol are presented. The presented applications are possible for routing protocols in general. We propose an application in a field-based routing protocol.

One approach is implemented as an extension of a field-based routing algorithm. The extension is evaluated with practical measurements. The practical measurements were conducted with handhelds (HP iPAQ Hx2410) running Linux.

The evaluation results show that the application of the probablility model improves the performance of the implemented algorithm. Further improvements can be made by adjusting the implementation parameters based on the probability model.

Zusammenfassung

Routing in mobilen Ad-hoc Netzwerken ist eine vielseitige Herausforderung. Neue Erkenntnisse zeigen, dass bertragungsfehler und Fehler durch mobile Benutzer unterschiedliche Eigenschaften haben. Es ist ein Wahrscheinlichkeitsmodell das diese zwei Fehlerarten unterscheiden kann verfügbar. Diese Arbeit untersucht praktische Anwendungsmöglichkeiten dieses Modells.

Verschiedene Möglichkeiten zur Anwendung des Wahrscheinlichkeitsmodells in einem realen Protokoll werden präsentiert. Die Vorschläge sind allgemein auf Routingprotokolle anwendbar. Wir schlagen die Anwendung in einem Field-based Routing vor.

Ein Ansatz wird als Erweiterung eines Feed-based Routing Protokolls implementiert. Diese Implementierung wird praktisch evaluiert. Die Messungen werden dabei mit Handhelds (HP iPAQ Hx2410) unter Linux durchgefhrt.

Die Messergebnisse zeigen, dass die Performance des Protokolls durch die Anwendung des Wahrscheinlichkeitsmodells verbessert wird. Weiteres Verbesserungspotential liegt in der Feinabstimmung der Entscheidungsparameter der Implementation.

I would like to thank Prof. Dr. Bernhard Plattner for the opportunity to do my master thesis at the TIK. It was a very interesting and challenging task assignment and I continued my previous semester thesis to some extent.
I also would like to thank Vincent Lenders and Dr. Martin May who gave a lot of support throughout my work on the thesis. I felt very comfortable in the team and enjoyed the atmosphere.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter explains the motivation of this master thesis. A summary of the work done is presented and the structure of the rest of this report is explained.

1.1 Motivation

Nowadays, wireless networks are a well established form of mobile communication. In the last time, a new form of wireless communication has seen a noticeable growth. This form is called mobile ad hoc networking.

Mobile ad hoc networks (MANETs) deliver wireless communication to devices without the need of a preinstalled infrastructure. In WLAN[1] technology this means communication without the use of an access point.

To support this way of communication, a new field-based routing algorithm was developed at TIK [2]. This routing protocol was designed and implemented on a i386 notebook running a Linux operating system.

Conclusions from another work at TIK [1] show different characteristics for link lifetimes, depending on the source of failure. The results implied that two major link failure sources can be distinguished. Transmission errors usually lead to link breakage after only a short amount of time. Disappearing nodes due to the mobility of users show a significantly different behavior concerning their link lifetime.

Goal of this master thesis is to combine the newly developed routing protocol [2] and the results from [1] to optimize the existing routing algorithm. Optimizations could be made considering different factors:

- Available bandwith usage

[1]IEEE 802.11a,b,g,...

- Route stability
- Faster alternative route detection

Several optimization approaches can be proposed. One of the proposals will be implemented and tested against the original version of the protocol.

1.2 Summary

New discoveries considering link failure sources and link lifetime in mobile ad hoc networks are to be used in a existing routing algorithm.

In the first part of this work, an existing routing algorithm [2] is optimized by incorporating the results of a real user mobility study [1]. Therefore, related work is studied and several optimization variants are proposed. One of the proposed optimizations is then implemented.

To test the optimized algorithm with real user mobility, it is necessary to port the routing code to a HP iPAQ Hx 2410 handheld computer.

The second part of the work is the porting of the code to the iPAQ and the test of the algorithm. For the requirement of real user mobility, a scenario with 5 iPAQs where at least 1 is ported by a real user is set up for measurements.

The results of the optimized algorithm show a better performance than the original version. Depending of the amount and ratio of transmission and mobility errors the performance gain is not constant but visible in most cases.

1.3 Structure of the Thesis

The first part of this thesis describes the theoretical part of the work. In chapter 2, an overview of related work can be found. Three other ad hoc routing algorithms are described and it is concluded that no other routing algorithm differentiates between transmission errors and errors caused by node mobility. In chapter 3 some WLAN concepts and facts are presented that the work in this thesis is based on.

In chapter 4 the suggested possible applications of the probability model in a routing protocol are presented. Most of the suggestions aim at routing protocols in general. One is focused on field-based routing.

The second part documents the application of the conclusions of the first part. In chapter 5 the implementation of one application of the probability model is described.

Chapter 6 describes the three chosen algorithm policies that will be evaluated in the testbed setup. Expectations of the performance are stated. The chosen evaluation conditions and preparatory measurements can be found in chapter 7.

The testbed setup and practical evaluation results are presented in chapter 8 and in chapter 9 the conclusion and suggested future work is given.

Chapter 2

Related work

This chapter covers the related work and theoretical basics of this work. The first section gives a short view on routing algorithm classes followed by an algorithm overview and three algorithms that are explained in a little more detail. After having examined the related work, the conclusion shows that no other protocol uses different link failure sources.

Section 2.4 describes the field-based routing protocol that is used in the second part of this work and in the last section the probability model that is used as a basis for the suggested improvements is shown.

As this work deals with optimization in mobile ad hoc network routing, it was desireable to have an overview over what techniques are used in other similar concepts. This chapter gives a summary of the protocols and techniques that are employed by other projects in the same field.

2.1 Routing Algorithm Classes

As the field of mobile ad hoc routing is quite popular, a lot of routing algorithms have been proposed in this field. This section gives an introduction on what techniques are commonly used and how they have been adapted to fit the requirements in mobile ad hoc networks.

In mobile ad hoc environments, every node is able to act as client, server or router. Its role can and usually will vary from connection to connection, as mobile nodes can offer services, request services or route remote requests that they can not satisfy. Therefore, routing in mobile ad hoc networks is distributed among all present nodes, so every node can be considered as a router.

2.1.1 Classification

Routing algorithms for mobile ad hoc networks can be classified according to different criteria. One of the possibilities is the classification by the way they gather and store routing data. This leads to the following classes:

Table driven routing protocols store the information about possible routes through the network in tables. This is the most classic approach.

On demand protocols try to discover routes only if they are needed. They do not keep a map of the entire network, but only routing information that was actually needed.

Geographical routing takes its idea from the intuitive "send it into the right direction" approach. By definition these protocols use geographic information often obtained by GPS[1]. This information is not easy to obtain in a mobile, often indoor environment with critical power constrains.

Power-aware routing can be an extension of any other routing technique, where power consumption of every link or transmission can be taken into account.

Each of those classes has its own advantages and disadvantages. The most important classes for mobile ad hoc routing are described in the following subsections.

2.1.2 Table Driven Routing

As the name says, this class of routing protocols store their routing information in tables. Routes in the network are then identified by the consultation of the internal routing table.

The routing information is, in many cases, collected via the periodic exchange of special routing information packets. These packets contain the senders address as well as its routing table (sometimes in condensed form).

By periodically exchanging this information, every node creates its proper map of the network. This topology information is then in turn used to determine a route to the requested destination.

The frequency of these exchange messages is a crucial point in ad hoc networks, as it determines the speed at that topology changes are propagated. An interval that is too big can create unnecessary disruptions in communications and therefore render the network next to useless.

[1]Global Positioning System

As every routing information packet has to be transmitted it creates network traffic (often called routing overhead). Periodic exchange of routing information then creates permanent network traffic that reduces the available bandwith. Too fast exchange of such information can consume a considerable amount of bandwith leaving only a small fraction for actual connections.

2.1.3 On Demand Routing

On demand routing is based on the strategy that routes are only determined if necessary. Thus, no topology information is stored in the nodes, and there are no periodic routing information exchanges.

A route is determined in a process called route discovery when it is needed. The node then sends out a packet calling for the destination, and the destination answers. Every intermediate node adds its address, so when the packet returns to the source, a complete route is identified.

The node then usually caches the information for further use (as any intermediate node may do), but does not propagate it any further into the network.

The advantage over periodic exchange of routing information is the saved bandwith, as there are only (hopefully) sporadic route discoveries. The full bandwith is available for the required connection.

When the number of route discoveries is high, the bandwith consumption can get very high[2]. This results in a slower network connection and longer response times. Longer response times for the first packet are also unavoidable, as the route discovery process takes some time. To reduce this delay, intermediate nodes can answer the route request if they have the requested information cached. The delay issue therefore is minimized but still remains.

2.2 Algorithm Overview

In this section several existing ad hoc routing algorithms are presented. This aims at identifying common approaches and features that are often implemented.

For a start a list of algorithms is given. Algorithms that might bear features that are interesting to this project are then discussed one after another.

DSDV Destination Sequence Distance Vector [25]

WRP Wireless Routing Protocol [23]

CBRP Cluster Based Routing Protocol [12]

[2]Usually a flooding technique is used to propagate the route discoveries.

AODV Ad Hoc On-demand Distance Vector [11]

DSR Dynamic Source Routing [19]

TORA Temporally Ordered Routing Algorithm [27]

ABR Associativity Based Routing [9]

CEDAR Core Extraction Distributed Ad-hoc Routing [21]

OLSR Optimized Link State Routing [15]

TBRPF Temporary Broadcast Reverse Path Forwarding [17]

In the following subsections, three of the above protocols are discussed. Two on demand routing protocols with very different metrics and one table driven protocol were chosen.

2.2.1 Associativity Based Routing

Associativity based routing was developed in 1996 at the cambridge university and patented in the US in 1999. The main concept in this routing protocol is associativity. It aims at selecting more stable and thus more long-lived links in the network.

The metric to calculate the associativity value can be based on various elements like: link delay, signal strength, power life, route relaying load, period of presence or spatial and temporal characteristics. The actual calculation of the associativity ticks is not given in the description.

Routes in this protocol are determined on demand, saving bandwith in phases where no new routes are needed.

The routing process consists of 3 phases: discovery, reconstruction and deletion. Routes are discovered by broadcast and replies where all intermediate nodes update the associativity value in the packet payload according to the link metric the have for the next hop.

2.2.2 Ad-hoc On-demand Distance Vector

AODV is another example of an on demand routing protocol. It discovers its routes in a similar way than the ABR protocol does. The source broadcast a route request. Intermediate nodes create reverse path information in their routing tables and rebroadcast the route request.

The destination answers and the first reply can be used to establish a route. If two answers arrive at the source it may select the route with fewer hops or shorter delay.

If a node disappears or a link becomes unusable, the routing process has to be started over. Routes that are no longer needed are automatically deleted from the nodes caches.

2.2.3 Optimized Link State Routing

OLSR is a proactive table driven routing protocol for ad hoc networks. It optimizes the traditional link state routing algorithm for application in ad hoc networks.

The protocol periodically exchanges hello messages with its neighbors and deducts the network topography from this information. Broken links are detected by means of missing hello packets.

One concept to reduce routing overhead in this protocol is the use of multipoint relays. Multipoint relays are nodes that are automatically selected by the protocol. Their task is to distribute broadcast messages throughout the network without having every node sending a broadcast message. The multipoint relays are determined in a way that every node in the network has a multipoint relay as a direct neighbor. The protocol therefore aims to find the shortest paths in the network.

As additional features the protocol can support sleep mode operation, though this extension is not defined in the RFC 3626[14]. This would add power awareness to the protocol, supporting longer times of operation of mobile devices.

2.3 Conclusion

All of the regarded routing protocols have optimizations in regard of their deployment in mobile ad hoc networks. However, none of them take different link failure causes into account.

Protocols that aim at the shortest path may, in some circumstances, tend to select long distance links. Long distance links may suffer greatly in link quality by a small change to the nodes location. As with long distance links, the nodes operate on the edge of their transmission range, the link is likely to break from a small position change. This effect is also called the edge effect.

Power aware features are, if they are planned at all, only considered as protocol extensions. Even though this feature is highly desireable, this seems to be good practice as not every device may deliver the necessary information. In magnetic routing, power awareness can be incorporated in the metric if need be, but is not a requirement in itself.

As for the scalability of network size, on-demand routing often seems to be the design of choice for mobile ad hoc networks. The advantage of hello protocols is, that there is a predefined amount of routing overhead that one can calculate. Quality of Service aware applications / routing can benefit from that fact. Route discoveries of on-demand protocols may lead to spontaneous high bandwith requirements that interfere with some QoS constraints.

None of the studied protocols pays attention to the fact that links can be influenced by different sources of failure (transmission error of mobility). The impact on the routing performance of different link failure sources is to be determined in this project. The next chapter lays out different approaches to use this knowledge to improve the routing algorithm.

2.4 Field based Routing

Field based routing [2] is an approach that is focused on provided services more than individual servers. The information that is used to forward data over the network is collected from periodic broadcasts.

2.4.1 Routing Table Generation

Like port numbers in TCP the term service is used in field-based routing to identify a connection endpoint. The main difference is, that unlike in IP the node address that provides a service is not necessarily known by the initiator. Every service type has an associated capacity value that designates the "quality" of the service. The quality can be defined as needed e.g. available bandwith, available storage,. . .

Every node that provides a service send out periodic service advertisements by broadcast. These advertisements are forwarded throughout the network. This way the nodes establish a table of routing information that points to the best next hop for each service. The metric calculation to determine the "best next hop" is based on the service type capacity and can additionally use the hop count or other information.

The second mechanism to build the routing table are the neighbor exchange messages. Every nodes periodically broadcasts the information of his routing table to its neighbors.

With this information a potential field is created where every node knows the potential of its neighbors for every service type. Data for a service type is forwarded to the highest known potential.

2.4.2 Forwarding Data

A node that wants to access a specific service type composes a service query and consults its routing table for the best next hop. A reply service type that is unique in the network is created in case that an answer is needed.

Every intermediate node that receives a service query forwards the query over its next best hop and adds the reply service type if present to its routing table.

A node that receives a query for a service type that is provides itself will fulfill the query and answer to the reply service type if necessary. By the unique reply service type it is guaranteed that the reply is routed to the node that sent the query.

2.4.3 Implementation Notes

The field-based routing approach was implemented in [3] as MAgNETic routing. The potential calculation there determines the potential from the capacity and the hop count. Therefore, a service query will be forwarded over the fewest hops possible if there is only on service instance for a service type available.

2.5 Probability Model

In the work presented in [1] a new probability model is presented. This model accounts for the fact that errors on a link can be caused by either transmission errors like multipath fading or nodes that move out of transmission range. Errors that come from multipath fading, interference or related sources are summarized under the term transmission errors. The other source for errors, the ones that are caused by nodes moving out of transmission range are referred to as mobility errors.

In figure 2.5 the characteristics of the two error sources are plotted. One can see that in the first phase of the link the probability that an error is a transmission error is higher than that it is a mobility error. After a certain time the two characteristics intersect and afterwards the mobility errors become more and more dominant.

Basically the link lifetime can be divided into at least two phases. Phase 1 where transmission errors dominate and phase 2 where mobility errors dominate. One or more intermediate phases could be introduced to attribute to the degree of difference between the two curves.

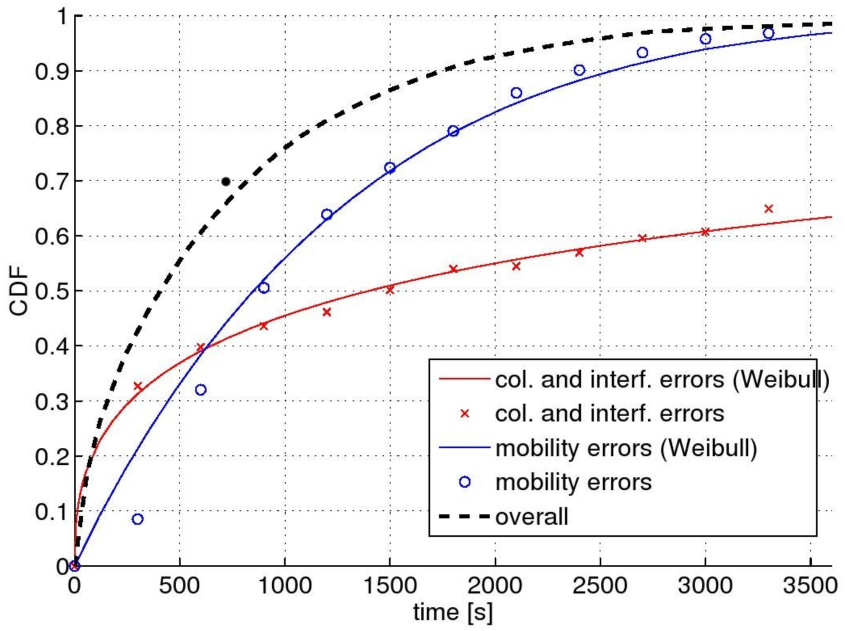

Figure 2.1: CDFs for mobility and transmission errors

This conclusion leads to the question if this fact can be used in a routing protocol. In chapter 4 some applications to a routing protocol will be suggested.

Chapter 3

WLAN Facts

This chapter describes some of the information of wireless LAN that this work is based on. It is not a complete introduction but aims only at giving the necessary facts that are needed to understand this work. A very good technical introduction to WLANs can be found in [28].

In the first section the possible types of WLAN networks and their use in this work are described. The second section points out the acknowledgments of frames on the MAC layer in WLAN networks. This mechanism is a central part of one of the possible modifications that are presented in chapter 4.

3.1 Types of Networks

In IEEE 802.11 wireless LANs [6], two different types of networks are distinguished. Infrastructure networks are deployed with access points as central unit. The access point has the possibility to coordinates data traffic between the wireless stations. Access points also are the link between the wired backbone and the wireless network. Nodes that are connected to the wireless network can adjust their transfer speed themselves or the accesspoint can make this adjustment. Usually the transfer speed is decreased if the signal to noise ratio drops. This increases the chance for a correct reception of the frames.

The second type of networks that the standard defines are ad hoc networks. In contrast to the infrastructure networks they are designed to be deployed without any supporting hardware like access points and without access to a backbone network. Ad hoc networks can be created for example by a couple of handhelds with WLAN cards. They can use the network to share data (like contacts or documents). The network exists only as long as the handheld users want to communicate.

From a technical point of view, ad hoc networks are much more limited in functionality than infrastructure network. The transmission speed for ad hoc networks is usually limited to 2 Mbit/s and broadcasts within ad hoc networks are distributed at the fixed rate of 1 Mbit/s.

3.2 MAC Layer Acknowledgments

MAC layer acknowledgments are a mechanism that is introduced in wireless LAN as the probability for a transmission error is tremendously higher than in a wired network. A frame that is transmitted over a wireless network is acknowledged immediately after reception by the receiving station. The transmission of the frame and its acknowledgments is considered as an atomic operation. If any of the two is missing the entire operation is failed.

The standard has set retry limits for two different packet sizes. For short packets the retry limit is 4 and for long packets the maximum retry limit is 7. This means that after 4 tries for short packets the wireless adapter reports an error and discards the packet. It then is up to the overlying application to correct the lost packet.

Chapter 4

Possible Modifications

Having studied other relevant work in this area, it is clear that none of the other protocols uses the concept of differentiated failure probabilities for mobility and transmission errors. It thus has to be decided, in what way this knowledge can be used for the optimization of a mobile ad hoc routing protocol.

In this chapter, several proposals are presented on how the conclusions of the preceding work can be used to improve routing performance. The ideas that are laid out are in the first section independent of any specific routing protocol. Nevertheless, the suggestions in section 4.2 aim at the improvement of a field-based routing protocol and thus are sometimes oriented on proactive table driven protocol design.

4.1 Generic Modifications

4.1.1 Adaptive MAC–Layer Retransmissions

The approach aims at a cross layer functionality, in a way that route lifetime information exerts an influence on the number of retransmissions a mac layer tries to do before a mac error is reported.

Every link that is maintained in the routing table has its associated lifetime. As long as the link is younger than a certain threshold, the mac layer does its standard procedure as it is usually optimized to correct transmission failures as good as possible. The threshold value is to be determined by the assumed characteristics of the link and mobility failure probabilities.

If the link lifetime passes the threshold value, the probability of a link failure due to mobility becomes the precedent effect. The idea now is to minimize the time it takes to detect that a node has moved out of range. Normally, the mac layer would do its predefined number of retransmission. By configuration changes the link layer could be configured in a way such that it does fewer retransmission retries before an error is reported.

The beneficial effect is the smaller delay that is caused by the link breakage detection. This enables the routing algorithm (which has to be notified of the transmission failure) to react more quickly and determine a new route to the destination.

The number of retransmissions can be gradually decreased until a predefined minimum value is received, as the possibility of transmission errors requires some attention. Even if a link is relatively old the possibility of a transmission error due to fading or other effect still is of concern and has to be dealt with.

One drawback in this approach is, that the mac layer is not aware of different connections going over the same interface. So if there are multiple connections going on, it is impossible to set the retry count individually for a single connection. This might lead to increased failure reports for younger links. This errors have then to be dealt with on the routing layer by initiating another transmission for the same route based on the lifetime of the used link.

Another realization of this idea can be to permanently reduce the number of mac layer retransmissions and deal with retransmission counts on routing level by repeating transmissions more often for younger links and fewer for older links.

This approach might also not be portable to other wireless transmission systems as for example bluetooth or infrared networks.

4.1.2 Routing Table Maintenance

The maintenance of routing table entries can also be based on the fact that after a certain time, the probability of a link breaking due to mobility becomes significantly higher than link loss due to transmission errors.

So could routes that exist only for a short time yet be maintained, even in the case of a reported mac layer error. This is to account for possible fading events or other (temporary) failure sources. A tolerance number can be introduced to set an error count limit at which the link is removed from the routing table. With increasing age of a link the tolerance value can be decreased. This leads to a faster removal of broken old links than erroneous new links. This attributes to the conclusion that old links usually break due to mobility rather than transmission errors.

This idea can also be combined with the previous proposal from section 4.1.1. This combination would allow the protocol not only to quickly identify broken links due to mobility but also to maintain a more accurate and up to date routing information table.

4.1.3 Extended Failure Source Model

From a practical point of view, another issue arises regarding mobile wireless devices. Most of the devices that will be participating in mobile ad hoc networks are user driven devices such as PDAs.

Use pattern of PDAs show a fundamentally different characteristic than any other mobile device in use. Not only for the battery capacity they remain switched of or in a form of suspend mode most of the time. Duty cycles of PDAs range in the region from 30 seconds to several 10 minutes. This has implications on link and route lifetimes as well.

This fact can also be exploited by the algorithm in the form of a off-duty broadcast before suspending. Any node that is being suspended could broadcast this information to let others know that they have to select another route. This way a lot of unnecessary connection interruptions can be avoided.

Drawback is, that the routing algorithm has in some form to be notified that the device is going to suspend mode. The resulting increased complexity might still be worth the effort as presumably a significant number of connection interuptions can be avoided by this technique.

4.1.4 Acknowledgment Mechanism

An acknowledgment mechanism could be introduced in the routing algorithm that serves the purpose of mobility link loss detection. As in field-based routing service advertisements and neighbor exchanges are sent using a broadcast mechanism, the introduction of acknowledgments is limited to actual service queries and replies. Therefore, it is only possible to benefit from a faster link loss detection for nodes that are engaged in data transfer. As the data for each service is usually forwarded to one host[1], it is probable that the mobility link loss can be detected by missing acknowledgments from the forwarding node.

The main purpose of this acknowledgment mechanism is to facilitate mobility link detection and not detection of a single lost data packet. This task is left to the upper layers to fulfill. As the probability for a link breaking due to mobility rises over time, the acknowledgment rate can be adjusted accordingly. The mechanism can be based on two things:

- Number of transmitted packets

[1]and thus actively concentrating data traffic to this host

- Timeout value

An acknowledgment based on the number of transmitted / received packets can be implemented with a varying number of packets necessary for an acknowledgment. With the use of a newly detected link the number of packets required for an acknowledgment is set to a reasonably high predefined number. As the link ages this number is gradually reduced until a minimum value is reached. A tolerance that accounts for packets lost due to interference or other effect can be incorporated. Using a timeout value for the acknowledgment mechanism bases on the same idea as packet counting, but uses a timer that triggers the transmission of an acknowledgment. The timeout value starts at a predefined time and is gradually reduced as link lifetime grows until a minimum value is reached. The receiver can initiate the acknowledgment timer with the reception of the first data packet. With the reception of the first acknowledgment by the sender, it can track the periodic appearance of acknowledgments.

The timer version of the acknowledgment mechanism seems to offer a higher reliability for link loss detection. A constant transmit rate for packets is not guaranteed and so a link might have been lost for quite some time before enough packets that should trigger the acknowledgment are sent.

4.2 Field Based Routing specific Modifications

4.2.1 Service Advertisement Revocation

If a node operates according to the proposal 4.1.1, another possibility is revealed if not only link states but also route states are considered.

An intermediate link in a route consisting of more than one hop can break due to mobility. In this case, the intermediate node which is maintaining that link, can inform the source node of that fact. The source node can then decide on the action to be taken. Either it knows a second route to the destination that does not use the broken link, or it has to initiate another route discovery process.

In terms of field-based routing, there would need to be a shortcut to propagate the new node weights. If upon the event of a mobility broken link, a new node weight calculation can be propagated, a faster adoption to the topography change could be achieved.

A new action subtype can be implemented to fulfill this requirements. Field based routing nodes use service announcements to provide services to other nodes. If an intermediate node detects a broken link to a service instance, it could initiate a new node weight calculation and propagate its new potential for this service before the next neighbor exchange is due.

Routing information in field-based routing is determined in a two step process:

1. Calculate service potentials in each node (build potential fields). This calculation is based on all received service advertisement messages.
2. Determine forwarding node for each service. For each service, the neighboring node with the highest service potential is selected as destination node for the respective service. This decision is based on the neighbor exchange packets.

To gain faster adaption times in case of mobility link failure, this two steps can be exploited. The procedure consists of three steps which are described in the following subsections.

Mobility Link Loss Detection

A node has to detect a link loss due to mobility of one of its neighbors. The detection criterion is based on the observations made in previous work which yields a time threshold value (that has to be predetermined or computed somehow) after that transmission errors are most likely due to mobility. This decision can be based on three different factors (or a combination of them):

1. Missing service advertisement message from a neighboring node.
2. Missing neighbor exchange message
3. MAC layer transmission error

For 1 and 2 a counter for every neighboring node is needed to detect a missing neighbor exchange or service advertisement message. To make use of factor 3 it is necessary to maintain link lifetime data for each link.

A MAC layer error occurring at a time later than the threshold value is a strong indicator for link loss, as well as a missing neighbor exchange after this threshold value is.

Missing service advertisement messages alone do not necessarily indicate a broken link, as it is possible that a specific service is no longer available (e.g. the service has been deactivated). In this case, a form of probing could be implemented to check wether the node is still reachable or not. This probing can be based on passive information such as a neighbor exchange from the node in question or any other received packet / acknowledgment that the node in question transmitted. Active probing could also be performed to make sure that the node is missing. This requires another neighbor probe packet type that has to contain the UID in question. This packet is only sent by the detecting node and not forwarded throughout the network.

Potential Field Recalculation

When a link loss is detected by the criteria of the previous subsection, the potential field will be recalculated. The main performance improvement can be gained out of the fact that this recalculation happens immediately (by the detection of the broken link) and not delayed by the service advertisement timeout.

The recalculation of the potential field needs to be initiated by a special "service revocation packet" that is broadcasted by the node detecting the link loss. The service revocation packet should also be broadcasted by any node that disables one of its services.

In the case of the service revocation packet being broadcasted by the node that detected the link loss, special care has to be taken that the missing node might still be reachable by other nodes in the network. The service revocation packet therefore has to carry not only the UID and service type but also the last received sequence number. This causes other nodes to update their potential field only if they have not received a service ad from that node carrying a higher sequence number. As the service revocation packet is transmitted after the timeout this should prevent any conflicting service indications.

Routing Information Update

After the new potential values are calculated, changes of forwarding nodes might be necessary. This routing information update can be achieved by transmitting neighbor exchange messages shortly after having recalculated the potential values.

A short delay between the recalculation and the neighbor exchange messages allows the service revocation packets to travel throughout the network and take effect before routing information is updated.

A special flag could be used to indicate that the neighbor exchange packet is sent out-of-plan to update the routing information. As usually only a part of the network will be touched by the topography change, this flag can be used to limit the transmission range of such out-of-plan neighbor exchange packets. They could only be propagated over one or two hops and then discarded, leaving the propagation of the update to the rest of the network to the standard update routines.

Chapter 5

Implementation

In this chapter, the implementation of the chosen approach is described. It consists of two major parts that work together to achieve the desired functionality. The first part is the notification in the case of transmission errors and the second part deals with the proper reaction to such an event.

The implementation is made in such a way that the reaction to transmission errors can be configured in one central point where the link lifetime and the number past retries can be used to decide the proper action.

5.1 MAC Error Notification

To be able to react to MAC errors, a way is needed to get knowledge of transmission failures. Various possibilities have been considered and the use of a Linux provided mechanism was chosen. In Linux, a mechanism for sending messages from one entity[1] to another exists. This inter-process communication is similar to networking in its concept.

The netlink API provides communication endpoints called netlink sockets that can be used very similar to networking sockets. With a configured netlink socket it is possible to send messages to any other process that is listening on a corresponding socket. Very much like network sockets, netlink sockets can single-cast or broadcast packets. So it is possible to distribute messages to a group of processes as well as to one single process only.

Netlink messages are passed asynchronously, so the message is put on a message queue before it is delivered. This way, the sending application is not blocked by the receiving process.

[1]that can either be a kernel module, the kernel or any running process.

5.1.1 Generation Of Netlink Messages In The Driver

To be able to notify a process of the occurrence of a MAC error, the device driver needs to open a netlink socket and send a message in case of an error. The Orinoco module that is included in the Linux kernel and used in this project is already capable of sending such messages.

In the case of a transmission error, the hardware generates an interupt and sets a status register that the frame has exceeded its maximum number of retries before an acknowledgment was received. The driver checks for this condition and broadcasts a netlink message to inform any interested process about this event. By the way the transmission is handled by the device, the packet data is lost and only the destination hardware address of the packet that caused the error is passed to the application.

The fact that the driver uses a netlink broadcast led to the consideration to use a separate netlink socket to pass the MAC error information to MAgNETic instead of receiving the broadcast message. From an application point of view the messages received by a broadcast socket implies additional procession load on the application. It has to decide wether the message was generated by the device driver or by any other kernel subsystem that also broadcasts this type of netlink message. For the sake of performance in the driver, the additional processing was outsourced to the application and so the existing broadcast mechanism could be used in the driver.

5.1.2 Reception Of Netlink Messages in MAgNETic

In MAgNETic, the first task was to create and configure a netlink socket in the application that listens for the type of netlink broadcast messages generated by the driver. Upon the reception of a message, the protocol has to check wether it is concerned or not. This is done by extracting the destination MAC address from the netlink message (if present) and check the neighbor list for a corresponding entry. Further processing is only done if a matching neighbor can be found.

The reception of a netlink message again work very similar to the reception of a standard network packet. The use of a broadcast distribution mechanism restrict the communication to be one way from the driver to the application. This implied a necessity to buffer the sent packets in the protocol itself and initiate retransmissions from the protocol and not the driver.

5.2 Implementation Details

This section describes the implementation concepts that were used to incorporate the desired functionality into the existing MAgNETic implemention.

At first an appropriate location within the protocol sources had to be identified. By the nature of the task, that has to deal with asynchronous events in parallel to the rest of the protocol functions, it was chosen to implement the modification as a separate thread. The original implementation already runs in several threads and the implementation as another thread gave the most flexible approach as only small changes to the rest of the code had to be made. The new thread is called netlink thread and handles the reception of error messages and the proper reaction.

The second part of the modification had to allow for selective retransmissions. Bound by the fact that the hardware only provides information about the erroneous destination address (the packet is already discarded at that time) the protocol has to store the sent packets to have them available for possible retransmissions. This part of the modification were to be made in the other threads of the program and could not be done separately in the netlink thread.

5.2.1 Buffering Outgoing Packets

As the basic concept for this implementation is centered around retransmission, all packets need to be stored in memory for a short timeframe. This is, because the wireless adapter that does the sending of the packet does not return the entire packet but only the destination address in case of an error. So, the implementation needs to store the outgoing packets itself.

The mechanism for keeping outgoing packets in store for possible retransmissions was implemented as a list of sent packets that is filled upon the transmission of a packet and cleared continuously by a timer function. The list may be accessed by different functions simultaneously, so it is protected by a mutex lock mechanism that was implemented similar to the style existing in the original implementation.

Filling The Packet Buffer

The packet sending mechanism of the original implementation had to be modified directly to accommodate the buffering mechanism. This was done because the buffering is done in one place only, the `send_packet` function and is not needed in any other part of the program.

The packet buffer is organized as single linked list with struct packet_buffer type elements representing the packets. An element of the type packet_buffer contains the following items:

- payload_buffer (unsigned char *)
- type (unsigned short)
- length (unsigned short)
- dest_addr (unsigned char *)
- retry_count (unsigned short)
- lifetime (struct timeval)
- next (struct packet_buffer *)

In the function send_packet() a packet buffer element is created before the actual packet is passed to the networking stack.

With the implementation as a single linked list, packets can be added in a very efficient way as no list traversals are needed to add a packet.

As only unicast packets are acknowledged on the link layer, broadcast packets are not added to the packet buffer as there would never be a transmission error for one of them.

The memory is allocated to hold exactly the amount of data of the current packet. The send method had to be modified further to pass an argument for the number of retransmissions that have already been done for this packet. This makes the buffer maintenance easier and faster. The details of the retransmission and buffer handling are explained later in this chapter.

Packet Buffer Cleanup

As all sent packets are added to the packet buffer, a way to delete old items from the buffer is necessary. The original implementation uses a timer thread to perform various repeating task. This fact was used to create an additional timer entry that calls the packet buffer cleanup function every couple of seconds (The exact time can be configured in defines.h).

The time a packets lives in the packet buffer can be configured in the file defines.h. A separate timer was added to the existing timers and the cleanup interval can also be configured. The timer calls the function packet_buffer_cleanup() that removes all packets that are older than the configured value from the buffer.

It then releases the memory and returns. The access to the packet buffer list is controlled by a mutex that has to be locked before the buffer is modified.

The size of the packet buffer largely depends on the time the packets are stored and the maximum data rate of the network interface. A packet store time of 2 seconds and a maximum measured packet rate of around 15 - 20 packets per second do not arise the need for a faster packet buffer access or cleanup. The buffer implementation could be changed to a hash table if necessary[2].

5.2.2 MAC Error Handling

The MAC error and retransmission handling is a task that runs completely independent of the rest of the protocol functions. This is why we chose to implement it as a new thread. The entire netlink_thread was added to the original codebase. In this way, the modifications to the rest of the codebase could be minimized, keeping the structure as clean as possible.

The netlink thread waits for netlink message from the driver. If it receives a message, it first checks for the presence of a known mac address from the message and the neighbor list. Unknown MAC addresses are ignored.

In the next step, the packet(s) for the reported destination address are extracted and removed from the packet_buffer. (If they are retransmitted they are added to the packet_buffer again by the send method.)

As only the destination MAC is reported by the driver, it is not possible to tell which packet (in the case that multiple packets to the destination mac are in the buffer) caused the error. All packets are scanned for their retransmission count. The maximum retransmission count is then passed as one argument to the central decision making function called `do_retry()`.

The second argument for that function is the link lifetime. This information is extracted from the neighbor list. The neighbor list contains all known neighbors. If a new neighbor is discovered it is added to the neighbor list with a timestamp. The link lifetime is calculated from the current time and the timestamp in the neighbor list.

Depending on the return value of the `do_retry` function, a retransmission is initiated or not. If the packets are retransmitted, the retry count for all packets is increased and they are resent by passing them to the `send_packet` method.

In the case that no retransmission is done, all packets are discarded and the neighbor is removed from the neighbor list.

[2]e.g. in the case of higher tansmission speeds.

Chapter 6

Considered Implementation Versions

This chapter describes the two protocol versions in their respective behavior that will be subject to examination. The common starting point for both versions is, that the underlying mac layer reports transmission errors to the routing layer. This reporting only affects unicast packets, as broadcast packets like neighbor exchanges or service advertisements are not acknowledged on the mac layer. In the following sections, the reaction of the protocol on the reported error in the three versions is outlined. In the last section some expectations regarding the performance under different conditions and corner cases are summarized.

6.1 Original version

The intended reaction of the original version of the protocol is to discard a neighbor as soon as a transmission error to it occurred. This behavior was only outlined, but not implemented in [3]. During the modification implementation in this work, it is now possible to get this behavior. Therefore, as soon as a mac error is reported, the corresponding neighbor and its related entries are removed from the routing tables. New forwarding devices are then calculated as necessary.

6.2 Modified version

The modified version of the protocol accounts for different mac failure sources. A reported error can either be due to mobility (out of range) or a transmission error. In the work done in [1] a probability model is presented. Based on this probability model, selective retransmissions are done by the protocol if a mac error is detected. The main variable in the decision wether a packet will be retransmitted or not is the link lifetime.

In this version, three link lifetime phases are considered. Those link ages can be called:

- new
- transitional
- old

For every link age, a different number of (maximum) retransmissions can be set.

6.3 Maximum Retries

To complete the picture of possible reactions of the protocol, a third variant is considered in the test, the maximum retry version. In contrast to the modified version, this version ignores the link lifetime and always does a predefined number of retries. This is in order to differentiate the possible beneficial effect of the modified version that considers the link lifetime from the simple approach that more retries are always better. This should result in a slower response time if the link breaks due to mobility after a certain time.

6.4 Expectations and corner cases

We expect an average better performance for the modified version of the protocol, as it should adapt faster to a change in the topology and equally avoiding to many and to early changes. There are however some cases where the performance on a single mac error will be worse than the original version. First there is a list of corner cases, and afterwards the considerations to each case are mentioned.

- early mobility broken link
- lots of transmission errors to one neighbor
- very high data rates

In the case of a link that breaks very early due to mobility, the original version of the protocol has the fastest reaction time possible. It immediately abandons the route and tries to find a new one. The modified version is going to lose some time as it will retransmit the packet several times before removing the neighbor from its routing tables. Such cases will happen, according to the probability model they will be rare though.

In the case where a neighbor within range moves to a spot with strong local interference (e.g. strong multipath fading) and stays there, it might require many retransmissions and thus slowing down network performance over this link. The original version will switch to another route more quickly, but might suffer from frequent route changes, as some neighbor exchanges from might be received from the weak node.

Very high data rates could be a problem for both versions as all sent unicast packets need to be temporarily stored to be able to resend them later. With very high data rates the current implementation might not be efficient enough to handle high data rate sending and many mac errors at the same time. Memory usage might also become more important in high bandwith application, as all sent packets are stored in memory for a short timeframe.

For standard scenarios, the modified implementation of the routing protocol is expected to behave better than the original one. This is due to the fact that transmission errors receive a better treatment, and so better (shorter or less costly) routes are preserved if possible.

Chapter 7

Evaluation Concepts and Parameters

In this chapter, all considerations and definitions that were made regarding the performance comparison of the two protocol versions are specified. To make the results reproducible and reliable it is necessary to have defined measurement setups that do not favor any of the versions by setting certain parameters favorably for any of the versions.

The first section outlines the measurement concepts. In this section the way how the data was collected and the transmission and mobility error scenarios are described.

The second and third section show the preparatory work that was necessary to meet the measurement conditions that are presented in the first section. Especially, the way the transmission errors were generated is shown.

7.1 Measurement concepts

The goal of the measurements is to compare the performance of the original implementation of the MAgNETic routing protocol to the modified version that was developed and implemented during the earlier phases of this thesis. Both protocol versions will be run in the same scenarios and the performance is analyzed by the produced log files and a packet capture log from a wireless monitoring station.

7.1.1 Measurement conditions

To be able to compare the measurements between the two protocol versions, a couple of testbeds with predefined measurement conditions are defined. The main interest lies on how the protocol deals with errors that occurred during data transmission (that might be either caused by mobility or transmission errors).

Basically, errors that are due to mobility represent a final condition for a link as it is broken. To see how the two versions deal with transmission errors, it is more interesting to see the impact of various degrees of background noise (that can be one cause of transmission errors) on the routing performance.

In terms of location, one testbed setup is defined. The basic layout can be seen in figure 8.1. Three measurement conditions within the same scenario are then produced:

- low background noise (few transmission errors)
- medium background noise (frequent transmission errors)
- high background noise (lots of transmission errors)

By moving one node in the network, the adaption of the protocol to the topology changes is examined. The movement patterns of the node are deduced from the previous work made in [5].

7.2 Measurement preparation

The key factor in the series of measurements is the level of background noise, and therewith the number of transmission errors that occur. Three background noise levels were mentioned in the previous section. Before the measurements can be performed the actual impact of background noise on transmission errors has to be determined.

Transmission errors can be due to many different causes, mainly fading and background noise. In this test setup, transmission errors should be mainly caused by background noise to have a reproducible scenario. A preparatory measurement was necessary to adjust the background noise to cause the desired level of transmission errors.

The preparatory measurement was conducted with 2 iPAQs and one wireless monitoring station as well as a notebook and access point for background traffic generation. One notebook in an access point network was used as background noise source. A continuous packet sending mechanism was used to control the background traffic to different packet ratios. The background traffic was created in a neighboring channel infrastructure network in the same room.

The two iPAQs were the sending and receiving node. No routing over multiple hops was performed, as the impact on one link only was of interest.

The senders log of reported transmission errors in combination with the log of the wireless monitoring station, a background noise to transmission error ratio was found. This ratio is plotted in figure 7.3.1.

7.3 Measurement parameters

In order to set different MAC error rates for the practical evaluation, a series of preparatory measurements were conducted to determine a useable set of parameters for the background noise. Several parameters for the background noise were adjusted during the measurement series:

- transmission packet rate
- background packet size
- background packet rate

The error rate was measured for a link between two iPAQs with the compact flash WLAN card that was used throughout this work. For the measurements and Ad Hoc network between the two iPAQs was created on channel 11 and an infrastructure network (AP and 1 notebook) in the same room served as background noise simulation. The interfering network was set up on channel 10 and a packet generator on the notebook was used to generate continuous traffic.

7.3.1 Background Noise Results

The results from the background noise measurements are presented here. They served to select background noise levels for the practical evaluation. It was soon clear that small background packets did not produce the desired results. For the 20 packets per second case, both results (for small and large background packet sizes) are shown.

The corresponding graphics show the transmission error rate for every measurement.

- 4 packets per second, large background packets 7.3.1
- 20 packets per second, small background packets 7.3.1
- 20 packets per second, large background packets 7.3.1

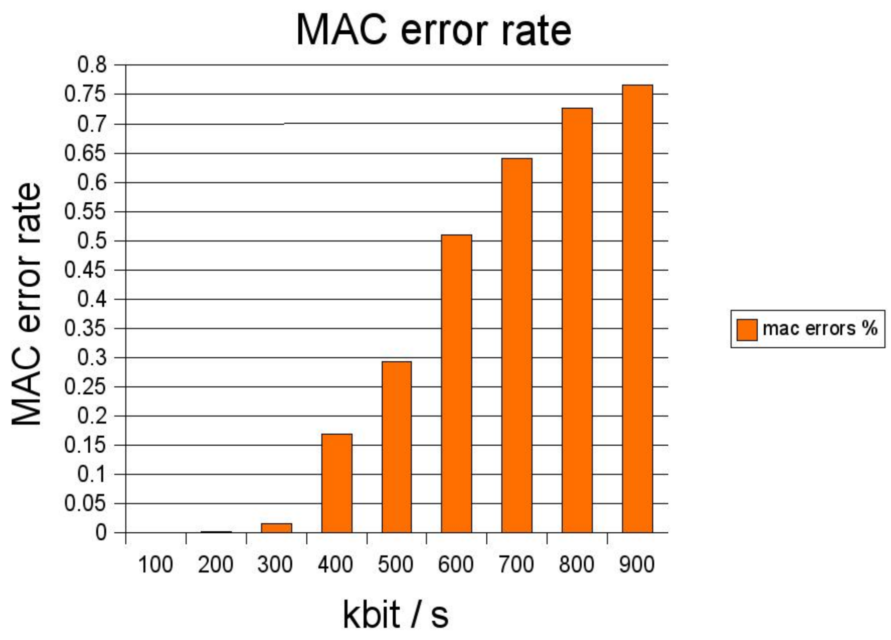

Figure 7.1: MAC error rate for background traffic (4 packets p. sec. data traffic)

- 30 packets per second (maximum rate, large background packets 7.3.1

In figure 7.3.1 the occurrence of transmission errors begins at a background traffic rate of 300 kbit/s and then shows a continuous increase. As the practical evaluation will show a higher protocol packet rate this ratio was not used for determining the background traffic rates.

Figure 7.3.1 shows a different behavior. The MAC error ratio is very low for low packet rates and then instantly jumps to a very high value. With this settings it is not possible to control the generated MAC error rate as desired.

The ratio plotted in figure 7.3.1 meets the requirements and evaluation protocol data rate best. The MAC error rate begins to increase very early and gradually increases until the maximum is reached. The higher data rates show a fluctuation that was caused by the packet generator in combination with the automatic transmission rate adjustment issued by the access point.

The MAC error rates shown in figure 7.3.1 are the rates for the maximum protocol (ad hoc link) speed. The rates are measured with protocol traffic at maximum speed in the ad hoc network. This scenario has more protocol traffic than will be present in the practical evaluation.

For the practical evaluation a background packet size of 1292 bytes per

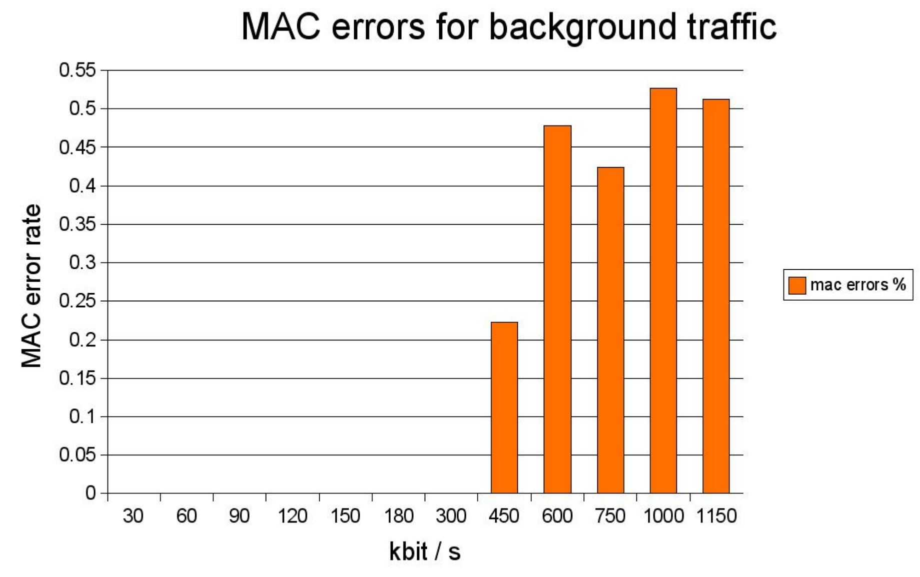

Figure 7.2: MAC error rate for background traffic (20 packets p. sec. data traffic, small background packets)

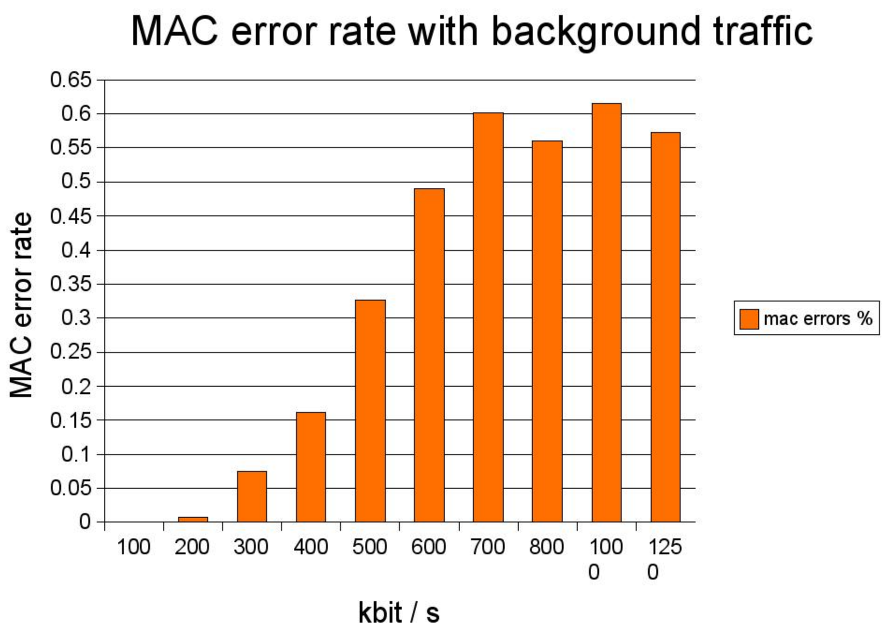

Figure 7.3: MAC error rate for background traffic (20 packets p. sec. data traffic)

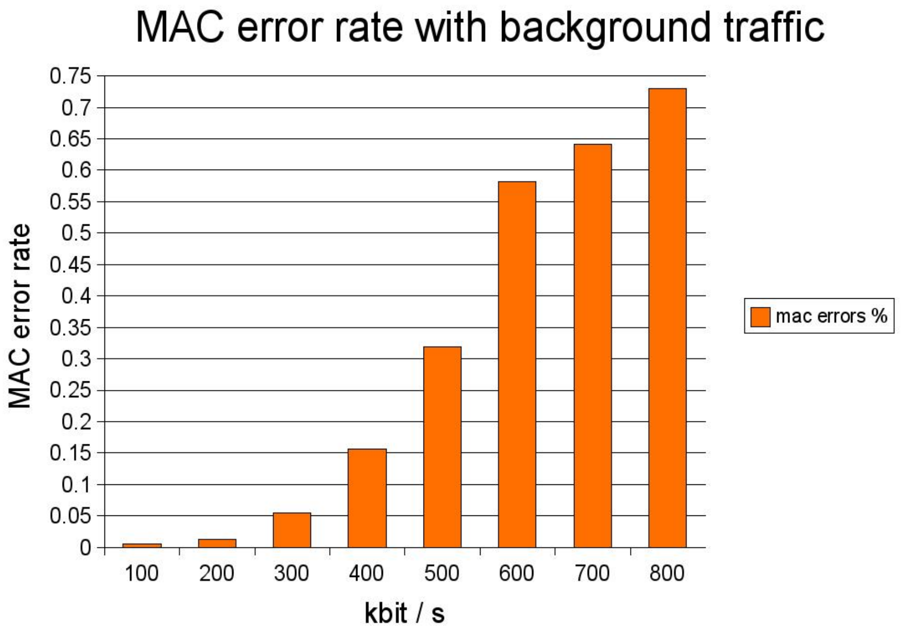

Figure 7.4: MAC error rate for background traffic (30 packets p. sec. data traffic)

packet was selected. The chosen MAC error rates are

- 1% (at a background traffic rate of 200 kbit/s)
- 32% (at a background traffic rate of 500 kbit/s)
- 60% (at a background traffic rate of 700 kbit/s)

Chapter 8

Practical evaluation

In this chapter the practical evaluation of the different MAgNETic routing versions is described. An overview over the setup and configuration is given as well as the individual measurement results are shown.

8.1 Configuration

The measurement was conducted using 5 iPAQs running Linux. The iPAQs used a compact flash wireless card that works with the Orinoco kernel module from the Linux kernel.

- 5 x HP iPAQ Hx 2410
- Pretec Compact Flash WLAN
- Ad Hoc network on channel 11 with 2 Mbit transmission rate
- Infrastructure network co-located on channel 10
- Notebook in infrastructure network
- Packet generator for background noise generation from notebook

The practical evaluation was done in a setup shown in 8.1. The two possible routes were constructed using the MAC filter facility provided by magnetic.

The testbed consist of 5 stations with one (station 3) configured as "server" providing a specific service type. Another station (station 1) is requesting this service type and sending out continuous service queries. Normally all queries would be routed from 1 via 2 to 3 and the reply the same way back. Station 2 was considered the mobile node and moved around as someone would in a 3 room

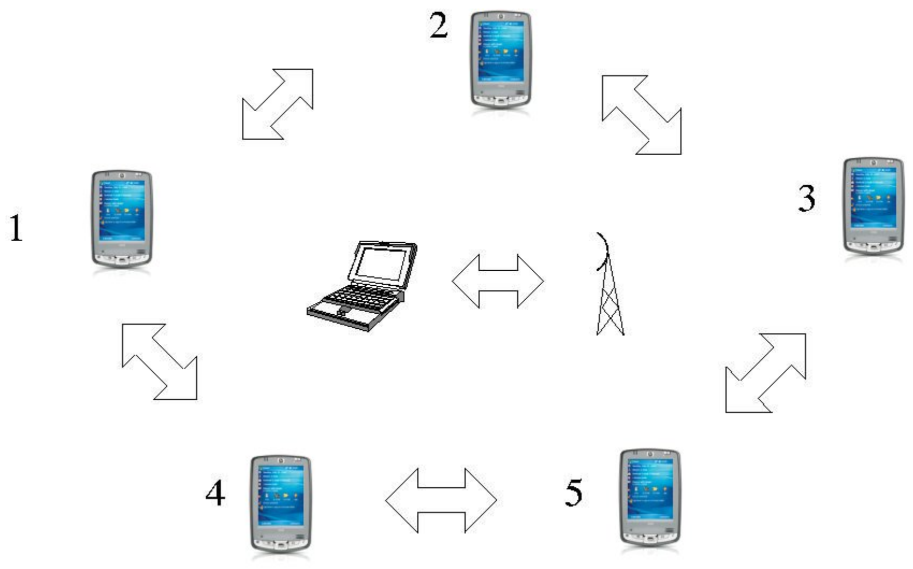

Figure 8.1: Testbed setup

area, in fact it was carried around in a 3 room flat during the measurements. The movements were not planned but consisted of the daily routine and other things that occur in a normal apartment. The alternative route over stations 4 and 5 was only to be taken if station 2 disappeared for some reason[1]

As one condition for the measurements was a controlled rate of transmission errors, the distance between the iPAQs was not chosen too large to avoid transmission errors that were caused by long distance links. The iPAQs were positioned in a distance of about 3 to 5 meters between each other.

The transmission error rate should also be equal for every link, so the interfering network was constructed by placing the access point and traffic generating notebook in the center of the setup ab about equal distances from all iPAQs.

The measurements were conducted over at least 5000 service queries at a rate of 2 queries per second. This gave a measurement duration of about 41 minutes per measurement.

8.2 Measurements

With every protocol version, 4 measurement configurations were planned.

- no background noise

[1]like moving out of range or being blocked by transmission errors

- low background noise
- medium background noise
- high background noise

The measurement without background noise was to verify the correct functionality and to prove the absence of transmission errors for the no background noise case.

As the measurements continued, the implementation turned out to be not as reliable as expected. For the medium background noise case, where a packet failure rate per link was 0.32 almost the number of received replies dropped to a level where a connection would almost be unusable. Instead of the expected 21% received packets, the received packets ratio lay significantly below 10%. Due to this fact, the high background noise case was not feasible for measurements.

8.2.1 Static Setup

The first series of measurements was done without mobility of the nodes. This measurement series served as basis to show the reaction of the two fixed policy implementations (original version and always retry version) in a pure transmission error setup.

The results of this measurements give the best case results that the implementation can achieve. The new variable retry version can in the best case come very close to the results of this measurements if its reaction on mobility errors is optimal.

8.2.2 Mobility Setup

The second series of measurements were conducted with mobility. The testbed setup remains the same but node number two was ported by a real user. The most interesting aspect of this measurements is the added presence of mobility errors and the reaction of the different versions on them.

The protocol will in case of a detected mobility error switch to the link over nodes 4 and 5. This longer path then results in a longer delay.

The real user mobility also results in a certain amount of time where the node number two is out of range that is not equal for all measurements. This difference could be a source of uncertainty in the evaluation results. It was observed that the absence times were in the same order of magnitude for every measurements.

8.2.3 Metrics

To be able to compare the results of the different implementation versions, two measurement metrics were agreed on. The two metrics are:

- end to end delay
- received packet ratio

The two metrics show different characteristics of the performance of the implementations. The received packet ratio reflects the amount of time where no connection to the service providing node was available. This could be either caused by lost packets or by the removal of the service type from the routing table as a consequence of a MAC error.

The second metric, the end to end delay, shows the performance within the received packets. This value is influenced by the number of mac retransmissions that are done before the correct reception of the packet and by the length of the route the packet takes on its way.

To evaluate the performance it is necessary to take both values into account. Otherwise for example, a version that receives only very few answers but those answers came very fast would be better rated than a version where a substantial amount of answers but with a slightly longer delay because of more MAC retransmissions are received.

So, the optimal version should show the best combination of received packet ratio and end to end delay.

8.3 Results

8.3.1 No mobility and no background noise

A reference measurement without mobility and without background traffic was performed to get the best possible performance. As no transmission errors occurred, there was no difference between the three different algorithms. The measurement was performed with the same setup as for the other tests (see figure 8.1). The measured average delay without mobility and transmission errors was 10.26 ms.

8.3.2 Static setup with transmission errors

A static setup measurement was also conducted to evaluate the two reference algorithms (original and maximum retries version). The beneficial effect of the

	original version	maximum retry
1% MAC errors	17.19 ms	15.99 ms
30% MAC errors	19.81 ms	35.26 ms

Table 8.1: average delay without mobility

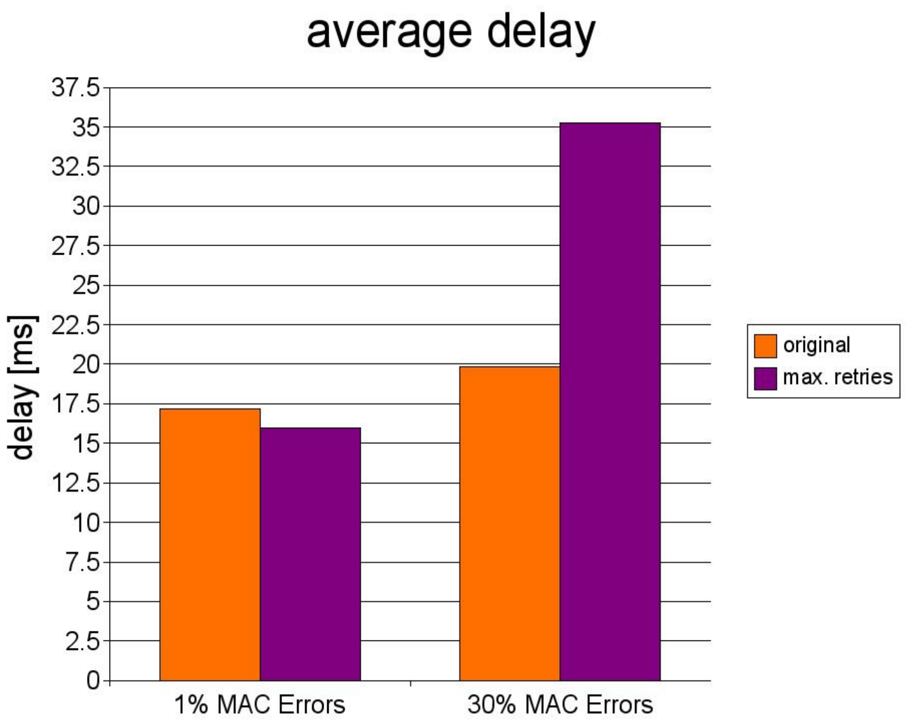

Figure 8.2: Average delay in static setup

variable retries mechanism is only expected to be visible in setups that include mobility. Thus, the results of this algorithm would not be conclusive.

The two algorithm were tested under the same conditions as in the later setup with mobility. The background traffic rates were set to generate the same amount of MAC errors as in the mobility scenarios (1% and 30% MAC errors) and no node was moved during the measurement. The results are shown in figure 8.3.2. In the table 8.3.2 the exact values are given.

8.3.3 Measurements with mobility

The average delay was chosen as an indicator of the performance of the different versions. As the measurement conditions were equal but not identical these figures indicate a trend and are not to be taken as absolute ranking. The

	original version	maximum retry	variable retry
1% MAC errors	16.313 ms	20.071 ms	16.404 ms
30% MAC errors	28.868 ms	24.432 ms	26.684 ms

Table 8.2: average delay with mobility

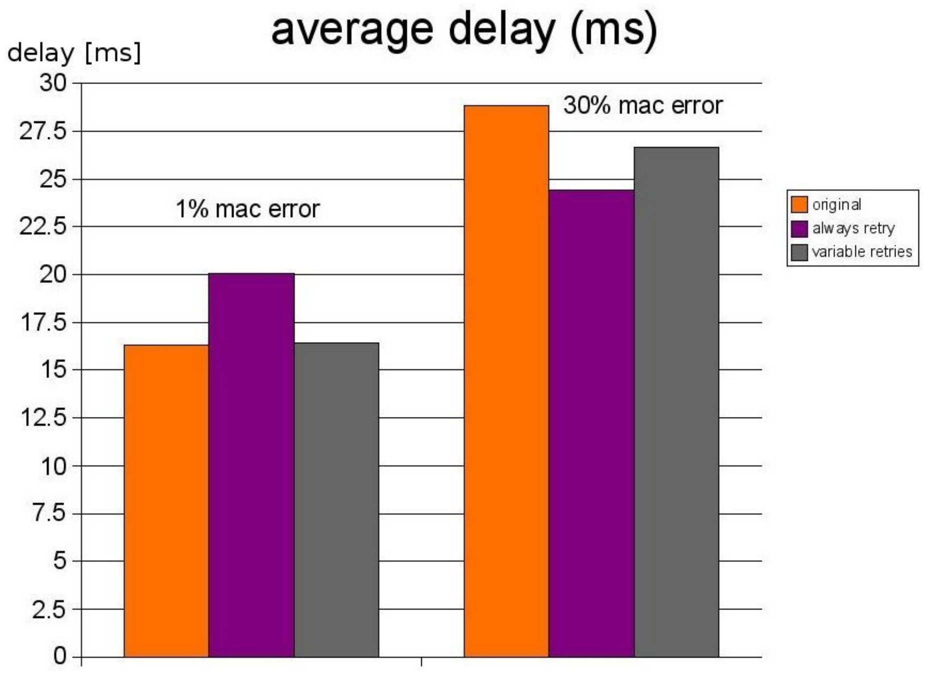

Figure 8.3: Average delay in setup with mobility

amount of time the main link was broken due to mobility or transmission errors was about but not exactly the same for all measurements.

The average delay was calculated from all received replies during the measurement period. Therefore this numbers do not show the actual count of received packets.which should also be considered for a performance comparison. These numbers show that the received packet ratio, as can be expected, is worst for the original version. This difference is mostly visible for the medium background noise measurement. In the low background noise case, almost all packets were received with every protocol version.

In figure 8.3.3 the average delay for the three versions in the mobility setup can be seen. The exact numbers are also shown in table 8.3.3. It is visible that the variable retry version has an average delay that lies between the original version and the maximum retry version. In the low background noise measure-

	original version	maximum retry	variable retry
1% MAC errors	99.77	99.53	99.52
30% MAC errors	2.84	85	12.93

Table 8.3: % received packets with mobility

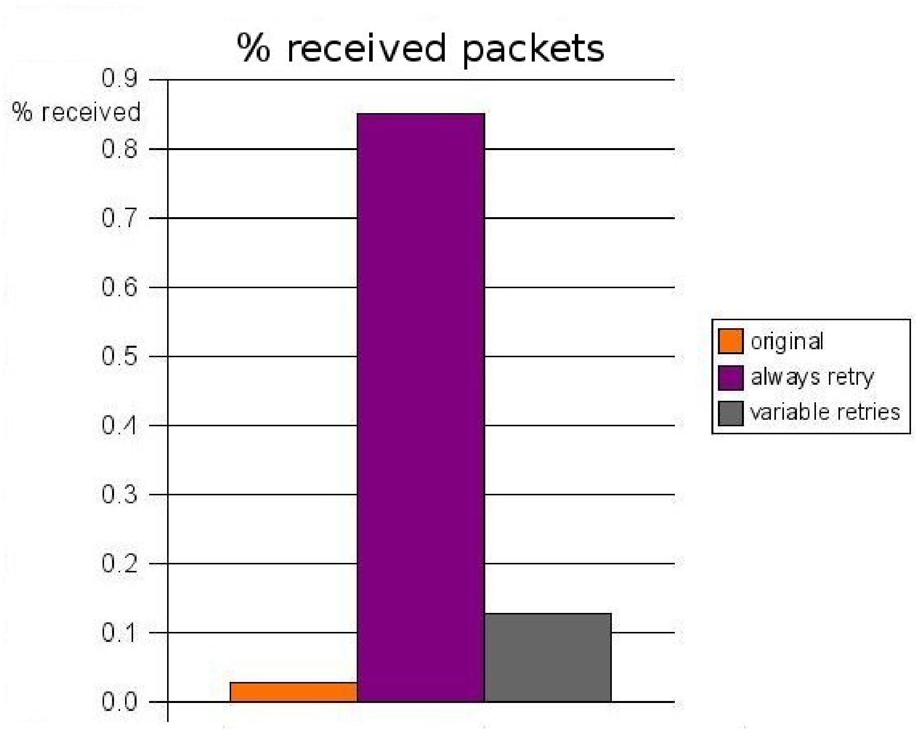

Figure 8.4: Average delay in setup with mobility

ment the variable retry mechanism is very close to the good performance of the original version. For high background noise the variable retries implementation is visibly better than the original version but there still seems to be room for optimizations regarding to the performance of the always retry version. The same is true for the received packet rate as can be seen in figure 8.3.3 and table 8.3.3.

In the medium background noise case the implementation (or the protocol) are already on the edge to be unusable, but the number of received packets is worst for the original version, followed by the variable retry version. The best result in this case is achieved by the maximum retry version. This is to be expected in an environment where there are more transmission errors than successful transmissions.

The results from the medium background noise case showed that there was no use in further increasing the background noise level. It was surprising that the received packet rate was as low as observed. Calculated from the single link error rate, a packet reception ratio of about 20% could be expected. The actual rates were somewhere between 4% and 10%. This might be due to lost service advertisements or neighbour exchanges. These messages are distributed by broadcast and hence do not produce MAC errors (as they are by definition not acknowledged). The removal of a neighbour and the loss of a service ad can cause the protocol to (correctly) remove the service type from its routing table. If now the following service ads and/or neighbour exchanges are also lost, the service remains unavailable and queries are dropped.

The purpose of the practical evaluation was to test the protocol under "real world conditions" and not in a simulator. A lot of knowledge about the inner workings and standards conformity of real products was gathered during the implementation and testing as well as the measurement phase. As a consequence the results presented here might be influenced by the hardware in use and the testing conditions. Care was taken to make the results reproducible and comparable. The results presented here are a hint on the useability of the proposed modification and its possible performance compared to simpler approaches and not to be taken as absolute values before more measurements in different setups have been performed.

8.4 Encountered problems

During the measurement phase, a couple of problems were discovered and had to be solved.

First the original implementation experienced random lockups at completely random times during measurements. This turned out to be caused by a not easily traceable inter thread deadlock that was caused by unmanaged access to a global variable. With this fixed the implementation was running a lot more stable but still got "killed" a few times and medium background noise conditions. The cause of this was not discovered any more (as the situation could not be easily reproduced and happened only a few times).

Battery lifetime was an issue to a certain extent. Running on batteries, the iPAQ would last for about 2 or 3 measurements before the battery was depleted completely. All iPAQs except the mobile node were connected to a power supply during the measurements. The charging mechanism does not yet work very efficiently and so the mobile node was connected to a power supply during stationary phases of the measurement[2].

Sometimes a MAC error for every packet was reported. This seemed to be cause by a notebook that was associated with the ad hoc network and that had a wired connection active at the same time but only if the ad hoc network was not created by the notebook. To resolve this issue the notebook was configured to create the wireless ad hoc network and the other iPAQs then joined the network. The notebook in the ad hoc network was used to monitor the other iPAQs and distribute the different versions of the MAgNETic implementation and gather the log files over the network. This helped to reduce the number of reboots that were necessary for the iPAQs which in turn helped to save a lot of time between the measurements. The iPAQs were controlled via a serial connection or a ssh connection. Files were copied using the scp protocol.

The packet generator did from time to time not keep the requested data rate. This lead to inconsistent measurement conditions that might affect the presented results to a certain extent. The fluctuations of the packet rate were caused by the automatic transmission rate selection feature in wireless LANs. The access point instructed the notebook running the packet generator to use a lower data rate. This lower data rate was too low to keep the packet generator producing enough traffic. The problem could be fixed by manually changing the rate setting on the packet generator computer.

[2]As if a user would place the device in its cradle while he is working.

Chapter 9

Conclusion

During this work a lot of knowledge of the existing approaches to ad hoc wireless communications was gained. The new idea to incorporate link lifetime and the distinction between mobility and transmission errors was presented. From this idea, several practical application cases were deduced and presented. One promising approach was then implemented into the existing MAgNETic implementation.

During the implementation phase, a lot of information about the inner workings of wireless LANs and the standards conformity of some real products was gained. This lead to some changes in the implementation and showed that not everything that is possible according to a standard is doable with a real world product. Obviously the manufacturers make their own decisions and assumptions about the use of their product and only create their products standards compliant as far as needed for them to work.

The practical evaluations purpose was to test three different retry mechanism implementations under "real world conditions". This means measurement setups with real devices and interferences. During this phase very much practical experience could be gathered and the understanding of the consequence of transmission errors for the protocol were better understood.

In an effort to make the iPAQs integrated wireless LAN device work, the Linux pcmcia handling was explored. Unfortunately, writing a device driver for the integrated wireless chipset could not be completed. The lack of technical documentation and support by HP prevented the further progress of the Linux kernel module development.

The practical evaluation results show, that the application of the probability model to the field-based routing algorithm really improve the performance of the algorithm. The performance gain is not yes optimized but it well visible in the results.

The modifications that were implied to make the retransmission mechanism possible imposed a far greater load on the program than the actual decision making function and the handling of the retransmissions. For any other protocol or platform that already has the possibility to do retries the additional modification for variable retry handling is worth the performance gain.

9.1 Contributions

This work made contributions in various fields:

- Summary of existing ad hoc networking protocols. In the theoretical part, existing protocol designs were studied and investigated for the idea of using different failure sources (transmission and mobility). No other protocol that uses this aspect could be found.
- Proposals for the practical use of the new failure probability model. The step from the theory to practical application of the probability model was made. The theoretical value and its application and existing practical difficulties or restrictions had to be weighted off against each other. The proposed approaches aimed at different layers and different aspects of the communication layer model to show that the idea could be applied to different parts of the communication process.
- One implementation proposal was chosen and implemented into the existing MAgNETic routing protocol.
- During the implementation the original idea of the MAgNETic MAC error handling was realized.
- The implementation now allows for arbitrary decision making functions on the basis of link lifetime and retry counting by simply replacing one function.
- A practical evaluation was planned and conducted. The practical evaluations purpose was to test different retry handling algorithms under "real world conditions" and not in a simulator. A lot of factors can influence a real measurement and the results are not absolute but give a good starting point to judge the usability of the various algorithms
- As a preparatory work to the practical evaluation, a relationship between background noise and transmission errors was established. By generating traffic in a geographically co-located wireless network (on a different

channel) the desired ratio of transmission errors can be set by selecting the appropriate background traffic.

- The iPAQ root file system was updated to the latest kernel version.

9.2 Future Work

To gather more data and judge the impact of the variable retransmission algorithm more reliably, it is desireable to make more measurements. The algorithm should be tested in various scenarios that change several parameters. As the expected beneficial effect increases with the amount of mobility, a scenario with high mobility would be of interest for measuring. Additionally, the following parameters could be adjusted:

- Node count
- Number of service providers
- Hop count
- Packet sizes
- Number of nodes requesting a service

Bibliography

[1] Vincent Lenders, Joerg Wagner and Martin May. Analyzing the Impact of Mobility in Ad Hoc Networks. 2nd ACM/Sigmobile Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN), Florence, Italy, May 2006.

[2] V. Lenders and B. Plattner. MAgNETic Routing: A Content-Based Routing Approach for Ad Hoc Networks. TIK-Report Nr. 184, ETH Zurich, Switzerland, December 2003.

[3] Florian Süss. Service Discovery and Routing in Mobile Ad Hoc Networks. Master thesis, ETH Zurich, Switzerland, September 2005.

[4] Bernhard Distl. Ad Hoc Communication with Handhelds. SA-2005-18, TIK, ETH Zurich, Switzerland, July 2005.

[5] Jrg Wagner. Analysis of Dynamics in Mobile Ad Hoc Networks. Technical Report SA-2005-18, TIK, ETH Zurich, Switzerland, September 2005.

[6] ISO/IEC 8802-11: 1999. IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. http://standards.ieee.org/getieee802/802.11.html

[7] SMF Design Team, IETF Manet Working group, Simplified Multicast Forwarding for MANET, http://www.ietf.org/internet-drafts/draft-ietf-manet-smf-01.txt

[8] Associativity based routing, http://www.comp.brad.ac.uk/ sbu-ruha1/abr.htm

[9] Toh, C.-K.. "Associativity-Based Routing for Ad-Hoc Networks". Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems, March 1997, Vol. 4, No. 2, pages 103-139.

[10] Ad-hoc on demand distance vector, http://moment.cs.ucsb.edu/AODV/aodv.html

[11] C. E. Perkins, ”Ad-hoc on-demand distance vector routing,”in MILCOM ’97 panel on Ad Hoc Networks, Nov. 1997. http://citeseer.ist.psu.edu/perkins02adhoc.html

[12] Cluster based routing protocol, http://www.comp.nus.edu.sg/ tayyc/cbrp/

[13] Optimized link state routing, http://menetou.inria.fr/olsr/

[14] OLSR RFC 3626, http://menetou.inria.fr/olsr/rfc3626.txt

[15] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, ”Optimized link state routing protocol for ad hoc networks,”Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001. http://citeseer.ist.psu.edu/jacquet01optimized.html

[16] Topology broadcast based on reverse path forwarding, http://tbrpf.erg.sri.com/

[17] R. Ogier et al., ”Topology Dissemination Based on ReversePath Forwarding (TBRPF),”Request for Comments 3684, February 2004. http://citeseer.ist.psu.edu/ogier04topology.html

[18] Dynamic source routing, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt

[19] Johnson, D. and Maltz, D. (1996). ”Dynamic source routing in ad hoc wireless networks”. in Mobile Computing (ed. T. Imielinski and H. Korth), Kluwer Academic Publishers, Dordrecht, The Netherlands. http://citeseer.ist.psu.edu/johnson96dynamic.html

[20] Core extraction distributed ad-hoc routing algorithm, http://timely.crhc.uiuc.edu/Projects/cedar/cedar.html

[21] Sivakumar, R. Sinha, P. Bharghavan, V. CEDAR: a core-extraction distributed ad hoc routing algorithm. IEEE Journal on selected Areas in Communications, Aug 1999 Volume 17 Issue: 8 pages 1454-1465

[22] Wireless routing protocol, http://wiki.uni.lu/secan-lab/Wireless+Routing+Protocol.html

[23] S. Murthy and J.J. Garcia-Luna-Aceves. ”An Efficient Routing Protocol for Wireless Networks”. ACM Mobile Networks and Applications Journal, Special issue on Routing in Mobile Communication Networks, Vol. 1, No. 2, 1996. http://citeseer.ist.psu.edu/murthy96efficient.html

[24] Destination sequence distance vector, http://decision.csl.uiuc.edu/ wireless/dsdv/

[25] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In Proc. of the ACM SIGCOMM, October 1994. http://citeseer.ist.psu.edu/article/perkins94highly.html

[26] temporally ordered routing algorithm, http://wiki.uni.lu/secan-lab/Temporally-Ordered+Routing+Algorithm.html

[27] V. D. Park and M. S. Corson, ”A highly adaptive distributed routing algorithm for mobile wireless networks,”in IEEE Infocom, 1997. http://citeseer.ist.psu.edu/park97highly.html

[28] Matthew Gast. 802.11 Wireless Networks: The Definitive Guide, Second Edition. O'Reilly Media, Inc., ISBN: 0596100523

Appendix A

Schedule

A.1 Milestones

November 5th, 2005	Thesis start, assignment of tasks
February 3rd, 2006	Intermediate presentation, preliminary report version
May 5th, 2006	Final presentation, submission

A.2 Week tasks

Week 45, 2005	Set up the workspace, install programs, create schedule.
Week 46, 2005	Look for related work that is also differentiating transmission failures and mobility
Week 47, 2005	Port original MAgNETic code to the iPAQ
Week 48, 2005	Define a set of possible mechanisms to improve the performance of MAgNETic routing, or any other routing protocol.
Week 49, 2005	Determine Linux network driver capabilities to modify layer 2 parameters dynamically / Investigation of the iPAQs built-in wireless interface and a Linux device driver for it
Week 50, 2005	Start implementation of the most promising optimization mechanism
Week 51, 2005	Implementation of the most promising optimization mechanism

Week 52, 2005	Vacation
Week 1, 2006	Vacation
Week 2, 2006	Implementation of the most promising optimization mechanism
Week 3, 2006	Implementation of the most promising optimization mechanism
Week 4, 2006	Implementation of the most promising optimization mechanism
Week 5, 2006	Implementation of the most promising optimization mechanism
Week 6, 2006	Implementation of the most promising optimization mechanism
Week 7, 2006	Implementation of the most promising optimization mechanism
Week 8, 2006	Implementation of the most promising optimization mechanism
Week 9, 2006	Finish modification implementation
Week 10, 2006	Define a set of metrics to assess the performance of the routing protocol
Week 11, 2006	Test of the modified algorithm
Week 12, 2006	Test of the modified algorithm
Week 13, 2006	Test of the modified algorithm
Week 14, 2006	Compare results of modified algorithm to the performance of the original code
Week 15, 2006	Compare results of modified algorithm to the performance of the original code
Week 16, 2005	Report writing
Week 17, 2005	Report writing
Week 18, 2005	Presentation

Appendix B

Workspace Setup

B.1 Used Software

To conduct the developpment work a suitable program suite and storage layout was created. The following programs were used throughout the work:

- Latex for writing the documentation
- Subversion for archiving documentation and sourcecode
- Eclipse 3.1.x for C programming and Subversion repository access
- Openembedded for cross compiling
- Debugging was done using gdb and gdbserver