NXP/MPR - An Optimized Ad-Hoc Flooding Algorithm

I. Fliege, A. Geraldy

Technical Report 343/05

NXP/MPR - An Optimized Ad-Hoc Flooding Algorithm

I. Fliege, A. Geraldy

Computer Science Department, University of Kaiserslautern, Kaiserslautern, Germany
{fliege,geraldy}@informatik.uni-kl.de

Technical Report 343/05

Computer Science Department
University of Kaiserslautern
Postfach 3049
67653 Kaiserslautern
Germany

NXP/MPR –
An Optimized Ad-Hoc Flooding Algorithm

Ingmar Fliege, Alexander Geraldy
{fliege,geraldy}@informatik.uni-kl.de
Networked Systems Group
TU Kaiserslautern

2.11.2005

Abstract

In the report, we will present a selective flooding algorithm for a voice radio application in mobile ad hoc networks called NXP/MPR. NXP/MPR, the Neighbor Exchange Protocol with Multi Point Relay, aims at reducing the network load compared to basic flooding and thereby alleviating problems arising from bandwidth shortage. It does not only fit to the voice radio application, but can be used for a variety of broadcast problems. The voice radio application is chosen as a challenge due to the high usage of bandwidth and the restricted ressources in ad hoc networks.

We will describe the basic idea behind NXP/MPR, give some detailed inside into the engineering, and we will present performance simulations comparing NXP/MPR and basic flooding for several scenarios.

1 Introduction

Wireless mobile ad hoc networks are built of a set of network nodes without infrastructure, prior setup, and fixed positions. Due to the lack of dedicated routers every node has to be service provider (i.e. router) in addition to its own service use. So, these networks are based on a "public spirit", which builds the real network topology. Nevertheless, in these networks we have to cope with different new problems adding to those known of the wired networks domain. The wireless network medium and the movement of the nodes may lead to transient or permanent splitting of the network. The combination of the shared wireless medium and the high number of possible routers leads to problems, e.g. frequent collisions, the hidden or exposed nodes problem and bandwidth shortages. Therefor, special routing algorithms are needed to cope with the dynamic and frequent changes in topology (e.g. freshness of routing information and loop-freeness), the error-prone medium (e.g. loss tolerance), and all the restrictions of the wireless network.
In this report we will introduce a sophisticated, selective flooding algorithm,

i.e. a distributed algorithm for the distribution of information to all reachable network nodes. Flooding may be used for the dissemination of any kind of information, for example measured data, audio data in a voice radio application, or routing information, i.e. assisting any other routing algorithm. Flooding can be regarded as a very basic routing algorithm, which does not rely on addressing a single sender or a set of senders. In basic flooding, every node forwards a newly received message (at least) once, so the recursive forwarding leads to the information dissemination. In many cases, this will lead to redundant transmission, resulting on the one hand in a high network availability, but on the other hand in a high network load and broadcast storms [9] (i.e. the massive simultaneous forwarding by many different nodes, leading to a period of collisions and network break-down).
In the following report, we will outline the basic idea of NXP/MPR, a sophisticated, selective flooding algorithm, which reduces the network load compared to basic flooding. We will illustrate several aspects of the engineering phase and network simulations with NXP/MPR, and we will point out possible improvements and future uses.

2 Related Work

The special needs of mobile ad hoc networks have led to a huge number of communication protocols. Many of them belong to the class of routing protocols, broadcasting protocols or neighborhood protocols.

2.1 Routing Protocols

Routing protocols shall discover the route to a receiver (unicast routing) or to a set of receivers (multicast routing). This can either be done when the route is needed (reactive routing) or beforehand (proactive routing). To determine the route, the routing algorithm either needs full information about the network (e.g. link state routing) or aggregated information (e.g. distance vector routing). The result might be the next hop of the route (distributed routing) or the full route (source routing). Known routing protocols include DSR [3] (unicast, reactive, aggregated information, source routing), DSDV [10](unicast, proactive, aggregated, distributed), AODV [11](unicast, reactive, aggregated, distributed), and OLSR [4][1](unicast, proactive, partially link state, distributed). All these routing algorithms differ in their way to find new routes, to maintain these routes or to use them. Each of them can be proven to show good performance in certain different scenarios.
We should keep in mind that all of these routing protocols either use an internal or an external broadcasting or neighborhood protocol to exchange the protocol information. These components influence the overall performance of the routing algorithm.

2.2 Broadcasting Protocols

An overview of current broadcasting techniques can be found in [17] and [15]. According to Williams, broadcasting can be classified as follows: basic flooding, probabilistic flooding, area-based flooding, and neighbor knowledge methods. In contrast to basic flooding with its broadcast storm, probabilistic flooding aims at reducing the number of rebroadcasts stochastically and thereby reducing the overall network load. The broadcasting probability can be set statically (which is possibly inadequate) or based on the number of earlier rebroadcasts received by the node.

In area-based flooding, the forwarding node first estimates the area reached by its forwarding and not covered by previous forwards. If this area is too small, no forwarding is done. To measure the area resp. the distance between nodes, the global positioning system or a local distance estimation (e.g. round trip time or receive strength) is needed. The neighbor knowledge methods are based on a periodically exchanged neighbor information, which leads to a (more or less) up-to-date topology. This topology can be used to select reasonable forwarders.

The MPRel (Multipoint Relaying) [6] belongs to the group of neighbor knowledge methods. Any sender decides which direct neighbor should forward the packet. This decision is based on the knowledge of the 2-hop neighborhood and is communicated by **hello** packets. The decision is done in 2 steps. At first, the necessary forwarders are chosen (take a forwarder, iff it is the only forwarder to any node), then the neighbor is chosen which reaches most remaining 2-hop-neighbors. The decision is then announced in the next **hello** message.

The AHBP (ad hoc broadcast protocol) [17] differs from MPRel by multiplexing the decision with any already scheduled broadcast, reaching a quicker update of the forwarding set. Additionally, the decision process is slightly optimized, e.g. for networks with high mobility. Both protocols only use forwarders with bidirectional link between sender and forwarder.

2.3 Neighborhood Protocols

Neighborhood protocols can be classified as proactive or reactive protocols. Proactive protocols exchange neighborhood information periodically while reactive protocols only send information in reaction to certain events. The Neighbor Exchange Protocol (NXP, [8]) aims at optimizing the network load by sending **hello** messages only when necessary. In the meantime, the neighborhood is sustained by sending (short) **keepAlive** messages. These **keepAlive** messages contain a sequence number of the last **hello** message to recognize lost **hello** messages.

The neighbor Detection of OLSR [1] can be classified as neighborhood protocol, too. It uses periodic **hello** messages containing the neighborhood of the sender and the states of the links. The **hello** message is also used to identify unidirectional and bidirectional links and MPRs. The knowledge of the 2-hop neighborhood is used to compute the multipoint relays of OLSR.

3 NXP/MPR

3.1 Functional Requirements

We want to develop a specialized routing protocol for the distribution of audio data through a mobile ad hoc network. The following requirements are to be met:

Functionality: Our objective is to set up a network of mobile nodes, consisting of one talker, which wants to be heard by all participants in this network, even if the receiver is more than one hop away from the sender. For instance, a bicycle trainer wants to give commandos to his trainees and wants every member of his team to hear his commandos.

Efficiency: Because of the high bandwidth assumed by audio data, we must insist on the bandwidth efficiency of the flooding algorithm. The high bandwidth usage of basic flooding and the broadcast storm must be eliminated or at least significantly reduced. The prevention of loops is very important to economize the bandwidth and to sustain the operability of the network.

Adaptability: NXP/MPR must adapt to a immense number of topologies and to the dynamic topology formed by mobile nodes. If the network changes, splits, or reunites, NXP/MPR must quickly regain the control over the network in short time.

3.2 Key Idea

In optimal case messages are only forwarded if forwarding is needed to ensure that messages reach every node of the network. To analyze this, we would need global knowledge of the network in every node making this decision. If we were using centralized routing, we would have to gather the network information in one single node, and to disseminate the decision through the whole network. With distributed algorithms, we would need every node to gather the information and to come to a decision agreed upon with every other node. Both cases depend on a huge amount of fresh information about the network, leading to a high bandwidth usage.
We will weaken the optimal case by restricting the network information to two hops. NXP/MPR will therefor consist of two distinct protocols. First, the Neighbor Exchange Protocol (NXP, according to [8]), which acquires and distributes neighbor information within a region of two hops distance, and second the Multi Point Relay (MPR) protocol which decides about the set of forwarders and which messages to forward. The MPR protocol relies on the information acquired by the NXP and by combination we get a self-learning algorithm that selects automatically and dynamically the forwarders and adapts to the changing network topology.
We will now take a look at both protocols and how they cooperate.

3.3 The Neighbor Exchange Protocol (NXP)

The Neighbor Exchange Protocol [8] shall exchange information about the two-hop neighborhood of all nodes. Therefor, every node uses in the first step the NXP **hello** message to announce itself to its neighbors. In second step, the **hello** message carries not only the sender address but also the list of 1-hop neighbors of the sender. Every receiver of this **hello** stores a list of 1-hop neighbors and for each of them the corresponding list of 2-hop neighbors, all entries with a corresponding reachability estimations.
Instead of broadcasting this **hello** message frequently, we use small **diffHello** messages to announce small changes of the neighborhood and tiny **keepAlive** messages to keep unchanged neighbor lists alive. All messages contain sequence numbers, which are increased with every **hello** message. If any **diffHello** or **keepAlive** messages contains a different sequence number than the last **hello** message of the same neighbor, the inconsistency is recognized and handled by a "**poll/hello** " handshake.
Thus, new nodes are rapidly recognized by the interaction of these four messages and drifting nodes can be determined by timeouts. To improve the behavior of NXP in scenarios with transient or weak links, 1-hop or 2-hop neighbors are only accepted if they are determined to be *UP* using the following state machine: All new nodes enter state *HOLD*. If enough messages from this node are received without timeout, *UP* is reached. If timeouts occur, *HOLD* or *DOWN* are reached. By this neighbor estimation, we delay the use of new nodes and prevent the use of neighbors which are only transient. To use these *UP*, *HOLD*, and *DOWN* states, 2-hop neighbor information is only contained in **(diff)Hello**, if the 2-hop-neighbor is *UP*.
Our neighbor exchange protocol is based on NXP [8], but differs in an important point: Our specification includes **diffHello** messages, which transfer small changes efficiently and extend the message set of NXP.

3.4 MPR

The MPR protocol [6] shall determine the set of forwarding nodes within the network. Unlike some other Routing protocols, this forwarding node set is not unique but relative to one reference node. Every node computes its own forwarding set, containing only nodes within 1-hop distance. Then the node broadcasts its set to inform the neighbors about their roles. When a message is originated at the node or forwarded by this node, its neighbors contained in the forwarding set will forward this message. To prevent forwarding loops, we cache information about already forwarded messages and drop duplicates.

3.4.1 The Selection Algorithm

The forwarding group selection algorithm is based on the 2-hop neighborhood information of NXP and the decision algorithm of [6]. The selection algorithm runs as follows:

1. eliminate all neighbors not knowing this node (unidirectional link)
2. forwarding_group:={}, not_covered_nodes:=2-hop-neighborhood
3. mandatory forwarders: If there exists any node in not_covered_nodes, which is covered by only one neighbor, add this neighbor to forwarding_group, else goto step 5.
4. cleanup (delete all covered nodes from not_covered_nodes) and goto step 2
5. compute covered node set for every neighbor, add neighbor with largest covered node set, if equal choose one randomly.
6. cleanup
7. if not_covered_nodes is not empty goto step 5

3.4.2 Differences to MPRel and AHBP

Our version of MPR does not include the neighbor detection and exchange, but relies on the NXP protocol. Similar to NXP, our MPR uses a data message to send the MPR selection, a **poll** indicates any inconsistency and leads to an immediate **hello**. An **active** message serves as a periodic keep alive message. New MPR selections are not only sent piggy-back with NXP **hello** packages (periodically, efficient, but slow), but rather directly (fast, but inefficient). To increase the efficiency, all MPR messages are multiplexed with NXP messages and payload data when available. The efficiency is additionally improved by the **active** message. Whenever possible, the **hello** message is omitted and replaced by an **active** message.

4 Engineering of NXP/MPR

4.1 Objectives

Besides the functional description of NXP/MPR, the protocol engineering plays an important role. The main goals of protocol engineering are to handle the functional and the data complexity of the protocol, as well as to guarantee the quality and maintainability of the specification and the code. To achieve the traceability between specification, implementation (code) and simulations, we use one SDL specification as starting point, we use *Telelogic Tau* [16] to translate this specification into C-code and *ns*+sdl [5] to simulate generated code. That leads to several advantages: First, any problem encountered in productive or simulation environment can be traced back to the SDL specification. Second,

any changes in the SDL specification are transferred into code instantly. This development process has recently proved to be profitable in several SDL specifications [12, 14, 13].
Within this development process, we have applied SDL structuring constructs to handle the complexity of NXP/MPR, decomposing the system into Block Types/Blocks, Process Types/Processes and Procedures. The structure of the specification shall reflect the ideas and its logical structure.

4.2 Structure of NXP/MPR

In figure 1, we can see the structure of NXP/MPR. At the bottom, the interface to the MAC layer is shown, at the top, an application can be connected. We clearly see the partitioning into NXP on the left side and MPR on the right. The only direct connection is via Channel **SelNXP**, which transfers the neighbor tables from NXP to MPR. The junction between NXP and MPR is realized in **Mux**, which combines data from NXP, MPR and Application and sends the multiplexed data via the medium, and in **Demux**, which distributes the parts of an incoming message to the corresponding processes. Additionally, **CoDecNXP/MPR** (un)pack the SDL signals of the received messages.

This structure allows to change one part of NXP/MPR without affecting the other. We preserve a high level of abstraction during the specification, which helps to rise the quality of the specification and to prepare future changes.

4.3 Management of Neighbor Information

An important task of the specification is to manage the neighborhood information. We have to store each direct neighbor with its last sequence number, its (*UP/HOLD/DOWN*-) state, and a list of its direct neighbors. Especially the *UP/HOLD/DOWN* state machine (with timers and signals) for every direct neighbor requires to instantiate a Process Type NbrTable for every neighbor. The process Neighborhood has then the role of demultiplexing incoming signals to the corresponding NbrTable and to collect data from all **NbrTables** when generating **hello** messages or when sending all information to MPR. Hence, especially the process type NbrTable is very concise.
The same design principle was followed in the specification of MPR. The Process Type MPRfTable is instantiated for every neighbor and contains its selection status and a timer for timeouts.

4.4 Review of the NXP/MPR Specification

After the NXP/MPR specification was finished, a complete review of the specification was done. We will now discuss the advantages and disadvantages, some decisions taken and alternatives.

Structure: As already said, the structure is adequate to abstract between NXP and MPR. In addition, it is good practice to decompose into neighbor

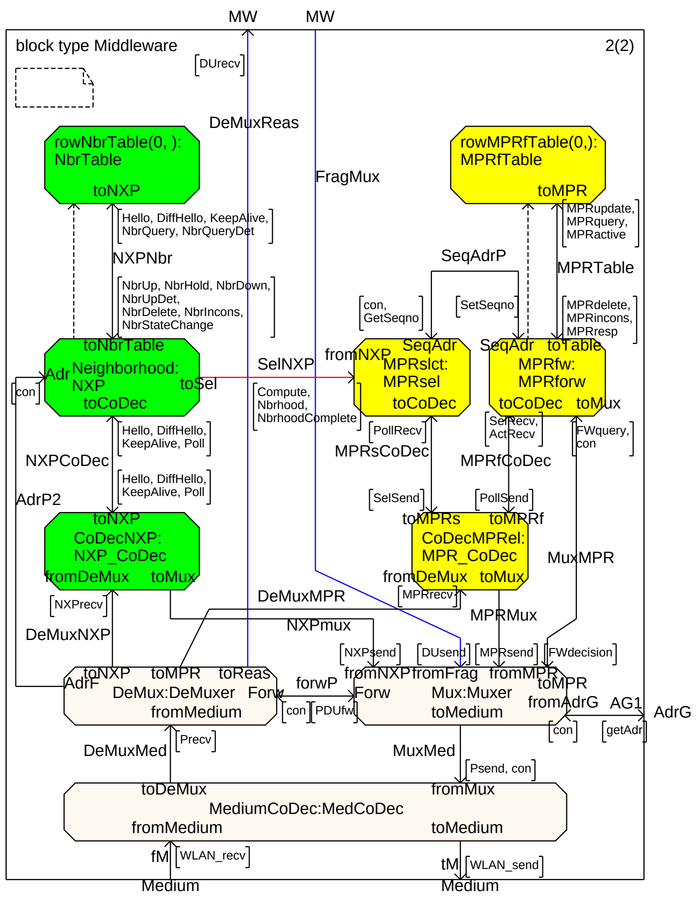

Figure 1: Structure of NXP/MPR

protocol and multi point relay, e.g. to reuse one of them in other protocols.

Data Management: An important decision during the specification was to introduce the multiply instantiated NbrTable and MPRfTable process types. This ended in very concise specifications of both process types, but caused additional complexity for the management of the instantiated processes. This complexity was hidden in the processes Neighborhood and MPRfw, which had do to extra hand shakes to collect data (lots of signals had do be sent internally), particularly several data structures existed in both the NbrTable and the Neighborhood process.

New Complexity Management: To simplify the structure of the NXP/MPR system, we eliminated the instantiated processes and created tables containing the neighbor information. To handle distinct timers for every neighbor, we used an SDL timer with parameters. Hence this timer can be set independently for every neighbor (timer parameter=neighbor address). When it expires, we get the node address for which it was started. So our neighborhood process becomes generic by using the corresponding node address in every input signal or timer signal. The need to exchange lots of data, to store data redundantly or to gather information from different processes was completely eliminated, resulting in an SDL specification with one third less in page count. Additionally, the (simulation) run time complexity was reduced significantly, too.

5 Simulations

We have simulated NXP/MPR for two reasons. First, we wanted to do a functional analysis to discover problems in the specification which lead to wrong behavior of the system. Second, we wanted to compare NXP/MPR with basic flooding or other routing protocols for a statistical evaluation (e.g. packet delivery ratio, delay, network load). The first type of simulation can be done stepwise using Telelogic Tau Simulator. The current state can be observed using the SDL or MSC tools of Telelogic Tau.
For the simulation of larger scenarios, we have used the *ns*+SDL simulator [5], an enhancement of the widely used ns2 network simulator. With this simulator, we have analyzed the behavior of NXP/MPR in networks of dozens or hundreds of nodes and over a quite long period of simulated time. Also, mobile nodes and network partitions have been simulated. The results of the simulation are written into log files, which contain functional traces (i.e., a detailed view on SDL transitions), traces of the simulator itself, as well as logs of the simulated system. The latter two allow for a network centered statistical evaluation of the System.

Algorithm	Basic Flooding	NXP/MPR
#Network nodes	5x5	5x5
#Bytes sent	15.020.547	1.060.154
#Packets sent	117.931	10.749
#payload packets sent	5.000	5.000
#payload packets received	4.517	4.800
#Duplicates received	82.365	0
Avg. payload loss [%]	5,89	0,007
#payload packets forwarded		
per node	4.517	0
per cent [%]	100	0

Table 1: 25 nodes, each in direct reach of each other

Performance Simulations We have simulated the behavior of NXP/MPR in two static scenarios. In the first scenario, the network is formed of 25 nodes positioned at the same place, so each of them is in direct reach of each other. In optimum, no message will be forwarded, since no receiver is out of reach of the originator of the message. In basic flooding, every node will send each message once, which will lead to a multiple of the network load.
As we can see in table 1 from the number *#payload packets forwarded*, NXP/MPR has recognized this simple topology and does not forward at all.
In a second simulation, we have tested the MPR selection algorithm. We simulated a network with 9 nodes in a line, where every node can reach 2 nodes left and 2 nodes right of it (if existent). As you can see in figure 2, the leftmost node (no. 1) is the data sender and its forwarding group consists only of node 3. Node 3 has selected the forwarding group $\{1, 5\}$. Node 1 recognizes the duplicate, and node 5 forwards, and so on. In the second example in figure 2, node 2 is the sender.

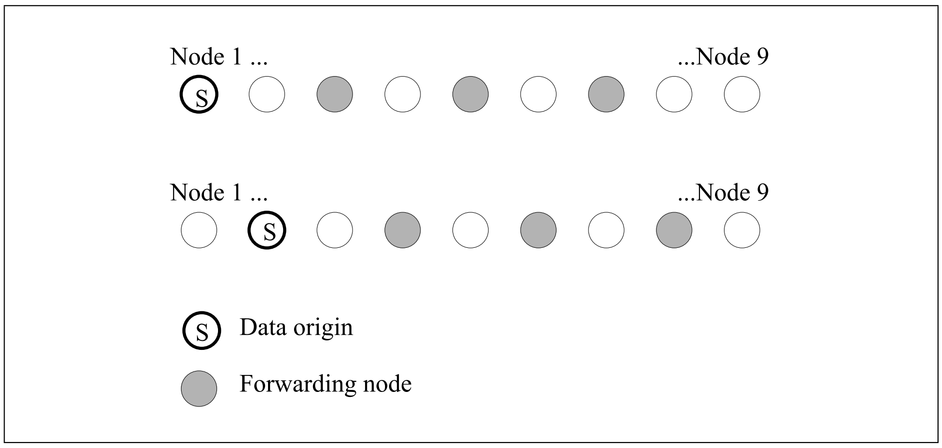

Figure 2: Forwarding nodes for 9 nodes example

Finally, we have compared basic flooding to NXP/MPR in a scenario with mobile nodes. We observes a network of 20 nodes with random motion and measure the overhead (forwarded packets) and the loss ratio.
As we can see in tabletab:table2, NXP/MPR uses much less bandwidth than Basic Flooding. We can see a much lower forward count, much less duplicates, but with the cost of a noticeable higher payload loss rate. However, in networks where the bandwidth matters, NXP/MPR can save a lot of load even with mobile nodes. To optimize the loss rate will be on future research.

6 Outlook

At the moment, works in different directions are in progress. First, the re-engineering of NXP/MPR will be concluded to improve the quality of the SDL

Algorithm	Basic Flooding	NXP/MPR
#Nodes	20	20
#Bytes sent	9.937.610	4.313.123
#Packets sent	78.023	31.392
#Payload packets sent	4.000	4.000
#Payload Packets received	3.701	3.499
Avg. payload loss [%]	2,6	7,91
#payload packets forwarded		
per node	3.701	1.033
per cent [%]	100	29,39
#Duplicates received	20.992	5.324

Table 2: 20 mobile nodes

specification. Second, NXP/MPR will be enhanced to handle gray zones [7] properly. Gray zones are a immanent problem of wireless networks. They describe the zone in which a node receives packets only with a probability significantly smaller than 100%. NXP/MPR is not able to handle this gray zone either in his neighbor table nor in the MPR decision and will not work satisfactorily. Additionally, the lack of redundancy leads to a higher loss rate when using NXP/MPR with gray zones. Here, implicit acknowledgments would help to recognize loss and to retransmit the packet.
Another work in progress addresses the use of NXP/MPR within routing protocols to optimize the broadcasting of routing information. The attempt to integrate NXP/MPR and AODV is described in [2].

References

[1] Clausen, T. ; Jacquet, P.: Optimized Link State Routing Protocol (RFC 3626). http://www.ietf.org/rfc/rfc3626.txt. Version: 2003.

[2] I. Fliege, A. Geraldy: Mikroprotokoll-basierte Komposition von Routingprotokollen am Beispiel von AODV und NXP/MPR (in german), Technical Report 344/05, Networked Systems Group, University of Kaiserslautern, 2005

[3] Johnson, David B. ; Maltz, David A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: Imielinski (Hrsg.) ; Korth (Hrsg.): Mobile Computing Bd. 353, 1996.

[4] Jacquet, P. ; Muhlethaler, P. ; Clausen, T. ; Laouiti, A. ; Qayyum, A.; Viennot, L.: Optimized Link State Routing Protocol for Ad Hoc Networks. In: Proceedings of IEEE INMIC, S. 62-68, Dezember 2001.

[5] Thomas Kuhn, Alexander Geraldy, Reinhard Gotzhein, Florian Rothl"ander. ns+SDL - The Network Simulator for SDL Systems. 12. SDL Forum, Grimstad, Norway. June 2005.

[6] Laouiti, A. ; Qayyum, A. ; Viennot, L.: Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In: 35th Annual Hawaii International Conference on System Sciences (HICSS2002), 2002.

[7] H. Lundgren, E. Nordstr"om, and C. Tschudin, Coping with communication gray zones in ieee 802.11b based ad hoc networks, Proceedings of 5th ACM International Workshop on Wireless Mobile Multimedia (WoWMoM02), 2002.

[8] Mosko, M. ; Garcia-Luna-Aceves, J.: A self-correcting neighbor protocol for mobile ad hoc wireless networks. In: ICCCN, S. 556-560, 2002.

[9] Ni, S. Y. ; Tseng, Y. C. ; Chen, Y. S. ; Sheu, J. P.: The Broadcast Storm Problem in MANETs. In: Proc. International Conference on Mobile Computing and Networking (MobiCOM), S. 151162, 1999.

[10] Perkins, Charles E. ; Bhagwat, Pravin: Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In: Proceedings of the SIGCOMM 94 Conference on Communications Architectures, Protocols and Applications, S. 234-244, August 1994.

[11] Perkins, C. ; Belding-Royer, E. ; Das, S.: Ad Hoc On Demand Distance Vector (AODV) Routing (RFC 3561). http://www.ietf.org/rfc/rfc3561.txt. Version: 2003.

[12] F. Rothl"ander: "DSDV/AODV Analyse, Spezifikation und Simulation", Diplomarbeit, University of Kaiserslautern, Fachbereich Informatik, 2005.

[13] D. Schneider: Auswahl, Adaption und Entwurf eines Routingverfahrens für Ad-Hoc-Netzwerke; diploma thesis, in german, University of Kaiserslautern, Computer Science Department, 2004.

[14] T. Sonntag: "Optimiertes Fluten am Beispiel einer Sprechfunkanwendung", Projektarbeit, University of Kaiserslautern, Fachbereich Informatik, 2005.

[15] Stojmenovic, I. ; Wu, J.: Broadcasting and Activity-Scheduling in Ad Hoc Networks. In: Mobile Ad Hoc Networking, S. Basagni et al., eds., IEEE Press, 2004.

[16] Telelogic TAU, Telelogic AB, Sweden, http://www.telelogic.com

[17] Williams, B. ; Camp, T.: Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), S. 194-205, 2002.