Understanding the Loss Behavior of Position-Based Greedy Routing in Vehicular Highway Scenarios

Studienarbeit
Von

Roland Krüger
(rkrueger@rumms.uni-mannheim.de)
aus
Dresden

vorgelegt am
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Fakultät für Mathematik und Informatik
Universität Mannheim

Oktober 2005

Betreuer: Dipl. Wirtsch.-Inf. Holger Füßler
Dipl. Wirtsch.-Inf. Thomas King

Contents

1 Introduction

In the past years, the importance of wireless devices and radio communication technologies has increasingly grown. A broad availability of inexpensive equipment for wireless communication has raised demand for the development and study of suitable routing protocols that enable freely moving network nodes to exchange data with each other via radio communication in a loose network. Such nodes may be represented by cars equipped with radio devices or people using cell phones or handheld devices.

One such routing protocol is referred to as Position-Based Routing (PBR). In PBR, routing decisions are based on maximizing the progress each single data packet can make at each hop towards its destination. This approach is also termed *greedy forwarding* since forwarding nodes will pass data packets to that particular neighboring node that is geographically nearest to the destination.

Great effort has been put into enhancing this plain greedy forwarding process with optimization schemes that improve the overall performance of the protocol. These optimizations have a significant impact on the protocol's behavior. The present work aims at investigating the Position-Based Routing protocol together with its optimization schemes. Prior to that, an introduction of the involved techniques and algorithms of the protocol will be given. The analysis undertaken in this work is facilitated by way of simulating with the ns-2 network simulator. The main question that is intended to be answered here is about the influence that each single optimization scheme exerts on the overall performance of the protocol. In order to examine this question, the protocol is simulated with the optimizations alternately switched on and off. Additionally, the statistical properties of the greedy forwarding process will be analyzed and interpreted.

The remainder of the work is structured as follows. Section 2 will give an explanatory introduction to the protocols and mechanisms that are employed for the research done for this work. It deals with position-based routing, location services and the supporting algorithms involved in these techniques. After having illustrated the details of the PBR protocol, a more thorough problem description of the object of investigation is given in Section 3. Section 4 deals with the creation and configuration of the simulation scenarios as well as with the adaptions that were made to the network simulator's code. In Section 5, the results from the simulations are presented. For each seperate feature that influences the protocol's behavior, the respective effects are described and pictured with corresponding graphs. Finally, Section 6 provides a concluding overview of the work and recapitulates the findings made so far.

2 Position-Based Routing

2.1 Mobile ad-hoc networks

A mobile ad-hoc network (MANET) is a network of freely moving autonomous nodes which are able to communicate with each other through wireless radio devices. Due to the arbitrary movement patterns of the network's participants, links between nodes usually are of transient nature, i.e. they may break unexpectedly. Additional nodes may join the network or can be removed from it at any time. Thus, a mobile ad-hoc network is characterized by a constantly changing network topology.

Another characteristic of MANETs is the role of the individual nodes. They can act both as hosts and as network routers. Indeed, the task of routing network traffic is the most crucial part they are responsible for. Unlike wired networks, there are no special routers that are solely dedicated to routing network traffic.

Wireless networking equipment has recently become increasingly cheaper. Today's large dissemination of wireless devices, such as cell phones, hand-helds and notebooks, allow for a widespread adoption of ad-hoc networks providing inexpensive communication applications without the need for cost-intensive infrastructures.

There is an abundance of conceivable fields of application for MANETs. One of them is the use in road traffic where it can be used for exchanging information about road condition, traffic congestions or danger areas.

The focus of this work is on bidirectional traffic patterns as can be observed on a typical German highway. Each individual car here represents a member of the simulated MANET.

2.2 Routing strategies

In a typical mobile ad-hoc network, nodes are moving around at different speeds and in different directions. In doing so, links to formerly neighboring nodes may break while connections to new neighbors are established. In such an environment, routing is very different from routing schemes usually employed in static networks such as the Internet. The fluctuating properties of a mobile ad-hoc network must be allowed for with dedicated routing schemes.

Two categories of routing procedures in MANETs can be generally distinguished: *position-based routing* and *topology-based routing* [5]. In topology-based routing, information about existing links between nodes is used for making routing decisions. This is done by first finding a valid route to a destination node prior to sending the first data packet. Protocols from this category are very vulnerable when used in highly dynamic environments. The probability of a route to break becomes higher the longer an established route and the higher the relative speed of the involved nodes is.

In contrast, position-based routing is an approach that merely uses knowledge about the geographic positions of the participating nodes. It is not necessary to first discover a valid route to the destination node. Rather, a target node's physical position must be inquired if it is not already known. Such inquiries are facilitated with a location service (see Section 2.4). After having found out the destination node's position, a number of approaches can be applied to deliver a data packet to this node.

The main advantages over topology-based routing are that position-based routing is stateless and that it is more stable in an environment with a higher topology change rate. Only information about the current neighbors has to be kept by a router, instead of keeping track of entire routes. Furthermore, in a network whose topology changes very rapidly, position-based routing is a more promising strategy because logically maintained routes tend to break more easily, as already noted earlier.

2.3 Beaconing

2.3.1 Beacon packets

Position-based routing protocols require information about other nodes' geographical position which should be as precise as possible. Knowledge about its own position can be obtained by a node quite easily through technologies like the *Global Positioning System* (GPS). It is then necessary to share that information with the rest of the network so that routing decisions can be based on it.

Nodes learn about their neighbors, i.e. nodes located in their direct transmission range, by means of *beacons.* A beacon is an empty data packet containing the current geographic position of the beacon's sender. It is periodically broadcast within a given time interval which is referred to as the *beacon interval.* Upon receiving such a packet, a node adds an entry for the respective sender to its internal neighbor table. This entry is only then removed after a timer times out without being reset by overhearing a new beacon by the same node. Hence, every node knows about each of its neighbors with a temporal resolution determined by the beacon interval and the time-out. It is easy to see that the larger the beacon interval is, the higher becomes the bias between a neighbor's actual position and its position as it is recorded in the neighbor table entries of its peers.

2.3.2 Implicit beaconing

Implicit beaconing is an optimization scheme that helps keep the neighbor information up-to-date. Besides of sending beacons to announce their positions, network nodes provide each forwarded data packet's header with their own positional information. To enable the evaluation of these so-called implicit beacons by all neighbors, the network interfaces of all participating nodes are operating in promiscuous mode. I.e., bypassing MAC address filtering, each node overhears all packets that are on the transmission medium even if this node is not the registered addressee. Thus, all packets other than real beacons can be considered as piggybacked implicit beacons that improve the up-to-dateness of the overhearing nodes' neighbor table entries.

Aside from that, implicit beaconing is also a means to reduce network traffic. For each piggybacked beacon that is transmitted by a node, this node's internal beacon timer can be reset. Thus, it is not necessary for a node to send any real beacons as long as it forwards data packets at a higher or equal rate than is determined by the beacon interval.

2.4 Location services

In order to being able to communicate with non-neighboring nodes, each participant of a MANET registers his positional information with an installed location service. This service later takes care of disseminating this information into the network as needed. There exist different approaches that describe how this can be done. One of them can be termed as a *some-for-some* approach [6]. Here, each participating node is routing agent and location server at the same time, i.e. there are no dedicated servers for that task. Position requests from other nodes are served by those location servers that have actual knowledge about the targeted node. Each node can provide its own position, for instance.

Other strategies include stand-alone servers that serve all location requests (*some-for-all*) or hierarchies of regions that each are managed by local location servers that are responsible for one particular level of hierarchy.

In the some-for-some approach, when a node needs to find out the position of another node, it poses a position request to the location service. This results in the request being flooded through the network until it is received by a location server that has knowledge about the inquired node. A reply can then be easily sent back to the inquirer via the reverse route that has been established by the request packet on its way through the network. This is done through a unicast connection and by using the same routing protocol that is also employed for delivering common data packets.

2.5 Greedy forwarding

After having acquired a destination node's geographical position, data packets can be sent there. One possibility to do so is *greedy forwarding.* Using this scheme, a packet is always forwarded to the node that is positioned geographically nearest to the destination node. Each data packet that is sent carries the destination node's physical position in its header. After having received such a packet, a forwarding node has to make a new routing deci-

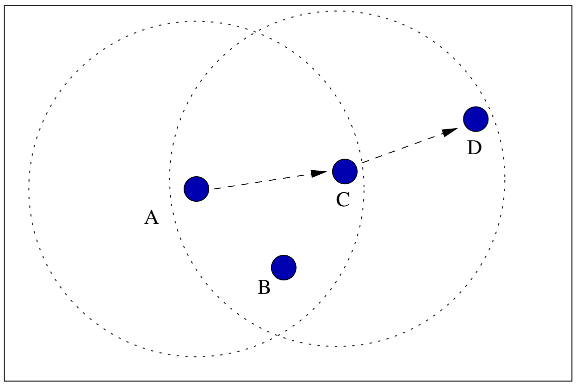
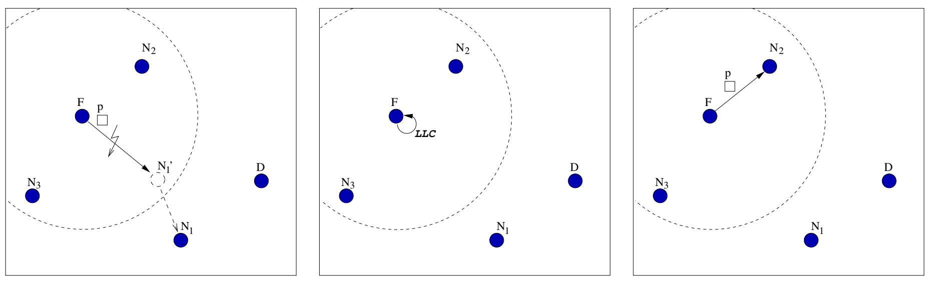


Figure 1: Greedy forwarding: Node A forwards to node C since C is nearer to the destination D than node B. The dotted circles indicate A's and C's radio ranges.

sion. First, it is checked by this node if there's an entry for the destination node in its local neighbor table. If so, the packet can be directly delivered there. If on the other hand the destination node is not within one-hop reach, it is tried to forward the packet to a suitable neighbor. The forwarding node will scan through its neigbor table, pick out a node that ensures the most progress towards the destination, and marks that node as the next hop. If no next hop can be found or the packet cannot be successfully forwarded, the packet is dropped.

2.6 Recovery strategies

The forwarding of a packet may fail due to different reasons. First, it is conceivable that the forwarding node turns out to be located at a local maximum, i.e. of all nodes within transmission range the forwarding node itself is closest to the destination. In such a case, greedy forwarding fails because it cannot find a route even if one exists. A mechanism to avoid packet drops caused by reaching local maxima is to temporarily interrupt greedy forwarding in such a case and find a detour around the maximum. This mechanism, which is termed *perimeter mode*, is implemented in the Greedy Perimeter Stateless Routing Protocol (GPSR) [3] which will here not be discussed any further. The problem of reaching local maxima is of no issue in the highway scenarios examined in this work. This is due to the fact that data packets will be sent one-dimensionally either up or down the road. If a packet can't be forwarded anymore it is because the destination node is located in another partition of the net and can't be reached whatsoever.



(a) (b) (c)

Figure 2: Lost link callback feature

2.6.1 Lost link callback

Packet forwarding can also fail on the MAC layer. This happens when a next hop has been chosen that is no longer reachable. Mobile nodes only learn about their direct neighbors upon receiving a beacon. In the meantime between two successive beacons, it can happen that two neighboring nodes move outside of each other's transmission range. This means that the respective neighbor table entries have become *stale*, i.e. they are no longer valid. They will only be purged from the neighbor table after a time-out. If such an entry is selected as the next hop, the packet's forwarding will fail since the next hop is no longer reachable. A recovery strategy to circumvent this problem is named by the term *lost link callback*. Here, the MAC layer invokes a dedicated callback funktion in the routing layer after having unsuccessfully attempted to send a packet. This callback function now removes the stale entry from the neighbor table and chooses another suitable next hop from the purged table. If necessary, this procedure is repeated until no further next hop candidate can be found or the maximum number of callbacks has been made. Only then the packet to be sent is considered undeliverable and will be dropped.

For an example of this feature, consider Figure 2. In Figure 2(a), node F intends to forward a packet p, which is heading towards its destination D, to node N_1. According to F's neighbor table, N_1 is expected to be located at position N_1' and is thus nearest to D from its point of view. N_1 is therefore chosen as the next hop. In reality, however, node N_1 in the meantime has moved towards a position that is outside of F's radio range. Hence, packet p cannot be successfully delivered. This is noticed by F's MAC layer, since it doesn't receive any acknowledgement. F's MAC layer then invokes the lost link callback function in the routing layer to notify it of the failed packet forwarding attempt (hinted in Figure 2(b)). Node F can now choose another next hop to try to send p there, as depicted in Figure 2(c).

3 Detailed problem description

An analytical inspection of the PBR protocol suggests that the protocol's performance does exclusively depend on the process of forwarding packets between two intermediate nodes. This can be verified by looking at the elemental routing decisions which are performed by every forwarding node. As a packet travels through the network, at each hop a new routing decision based solely on the information the routing node currently holds is made. Since the entire procedure of greedy forwarding is only dependent on the nodes directly involved in the individual forwarding processes, one can state that the only parameter that determines whether a packet will successfully be delivered is the loss probability p_i, with $0 \leq p_i \leq 1$. Such a p_i specifies the probability that a packet is dropped at the ith hop for any reason whatsoever. Since the success of a single-hop transmission is independent from all previous transmissions, a single loss probability p can be assumed to be valid for every forwarding process.

A completely analytical approach now yields the following probability function $S(n)$ for a packet to be successfully delivered from source to destination, provided that each forwarding process can be seen as an independent random experiment:

$$S(n) = (1-p)^n \tag{1}$$

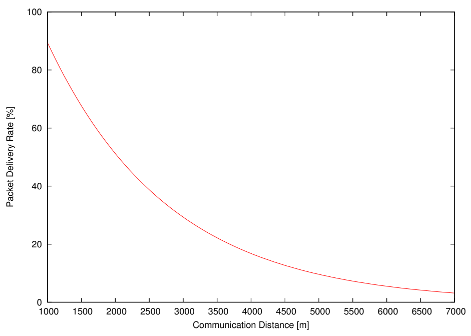
Here n denotes the number of hops that a packet has to take on the way to its destination. At the same time $S(n)$ can be interpreted as an estimate for the packet delivery ratio (PDR). This value refers to the ratio between sent and received data packets.

If the loss probability n is larger than zero, one can expect a convex PDR graph as exemplarily depicted in Figure 3(a) in an idealized way. Simulations of the greedy forwarding protocol, however, have produced rather dissatisfactory results in that the described effect couldn't be properly discerned. Figure 3(b) shows the PDR graph for a simulation of the PBR protocol with three different beacon intervals and a location service installed.

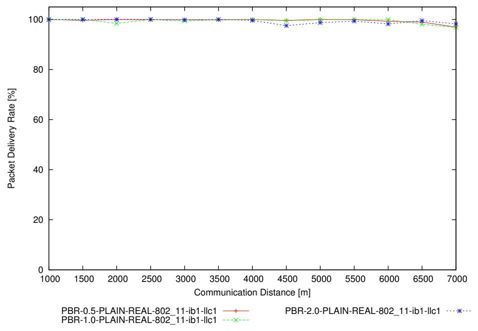
There are some factors that can be considered relevant for this behavior. Since Equation (1) relates to the pure greedy forwarding scheme, the additional optimizations implicit beaconing and lost link callback could be responsible for the graph's presentation.

Another conceivable influential factor can be seen in the installed location service. The most critical influence that the usage of a realistic location service has can be found in the delay that stems from the time-consuming process of requesting information about other nodes' positions.

In this work, a description of the effects the protocol's optimization schemes have is provided. This is done by isolating the different factors from each other and deriving the influence the single items exert on the protocol's behavior from appropriate simulations. For this, the two optimization



(a)



(b)

Figure 3: Estimated and obtained PDR curve progression

schemes are made seperately able to be switched on and off. Furthermore, the location service will be replaced by simulator knowledge, i.e. every node will learn about the positions of other nodes right away and with no delay.

4 Implementation

4.1 The ns-2 network simulator

The simulations for solving the problem at hand are performed using the ns-2 network simulator [7]. Ns-2 is supported by DARPA through the VINT (Virtual InterNetwork Testbed) project. Additional contributors such as the CMU Monarch projects or Sun Microsystems are providing substantial parts to the code.

The ns-2 network simulator is a deterministic and discrete event simulator. It is entirely composed of C++ and OTcl classes and provides an extensible interface needed by implementers to add their own code. A full-fledged scripting engine allows an easy setup and execution of arbitrary simulation scenarios. For this, an OTcl interpreter serves as a frontend for installing and controlling simulation scripts.

The output of a simulation run is an ASCII trace file with one-line records for every interesting incident that may happen during a simulation. Such incidents can be events like node movements, packet sending, receiving, and forwarding, packet drops and other miscellaneous information. Such a trace file can later be analyzed and summarized by specific scripts and visualized in proper graphs.

A simulation setup generally consists of two parts: a movement pattern and several accompanying communication patterns. Both pattern types are given in the form of OTcl scripts that tell the simulator what to do. As the name suggests, movement patterns contain all information about the node movements. Here, the nodes are placed on their initial position and are later moved along a virtual road. As the nodes leave or enter the simulation area they are switched on and off accordingly.

Communication patterns, on the other hand, set up all the necessary data that is needed for two nodes to get in contact with each other. Here, the communication facilities of every pair that has been chosen from the movement data set are initialized. This includes the setup of routing and ping agents as well as the definition of the data link's parameters.

Normally, for one scenario there's one movement pattern together with several communication patterns. The latter usually differ only in the average distance that is between any two communication partners and in the individual nodes that are meant to get in touch with each other.

In Section 4.2.3, a description of the patterns that were used for this work can be found.

4.2 Scenario generation with the `hwgui` tool

The `hwgui` tool was initially developed as a visualization tool for highway scenarios [8]. The tool is written in Java using the Swing GUI library. A command line mode was later added for scenario data file handling purposes.

In GUI mode, the user can open and display a uni- or bidirectional movement pattern. In addition to a graphical representation of the network nodes on a virtual highway, several general and time-dependent statistical values are shown.

If used in command line mode, one can process scenario files that have to be in the original DaimlerChrysler format (see Section 4.2.1 for an explanation). The two main functions of that mode are the generation of movement and communication patterns readable by ns-2 and the calculation of a set of statistics from movement patterns [4].

4.2.1 Origin of the movement patterns

To obtain real world comparable results, the simulation of common highway traffic is needed which is required to as much as possible come close to traffic patterns that are observable on ordinary highways. For this reason the output of DaimlerChrysler's driver behavior simulation tool `FARSI` is taken as the basis from which all needed movement patterns are derived. `FARSI` is a simulator that generates traffic patterns as observable on a typical German highway. To accomplish the goal of approximating reality as precisely as possible, `FARSI` takes into account a large number of fine-grained variables and parameters. These include driver behavioral models and technical aspects of the simulated vehicles. The resulting movement patterns then reflect realistic lane usages, speeds, distances and driver behavior [2].

The simulator's concrete output describes a short period in time (usually several minutes) of vehicular movement in only one direction. Such output comes in the form of a tabular text file where each column represents the trajectory of one vehicle. Starting from there, it is now necessary to convert such a `FARSI`-specific scenario text file into a format that is processable for ns-2. This is accomplished with the `hwgui` tool.

4.2.2 Creation of bidirectional movement scenarios for ns-2

Prior to converting a scenario into a Tcl script which can be ultimately understood by ns-2, some preparations have first to be done with the original DaimlerChrysler scenario files (abbreviated to 'DCX-files' in the following).

As stated above, the original DCX-files only provide unidirectional traffic patterns. To obtain the required bidirectional scenarios, the onward traffic has to be explicitly added by way of combining two unidirectional scenarios. This is accomplished by inverting one scenario and adding it to another one as the opposite direction. Before that is done, each original unidirectional scenario file is divided into ten cuts of equal duration such that each of those cuts contains a slice with a total duration of sixty seconds. In another step, two of these cuts are combined into one bidirectional scenario file. In order to avoid singularities stemming from the same pattern pasted onto itself,

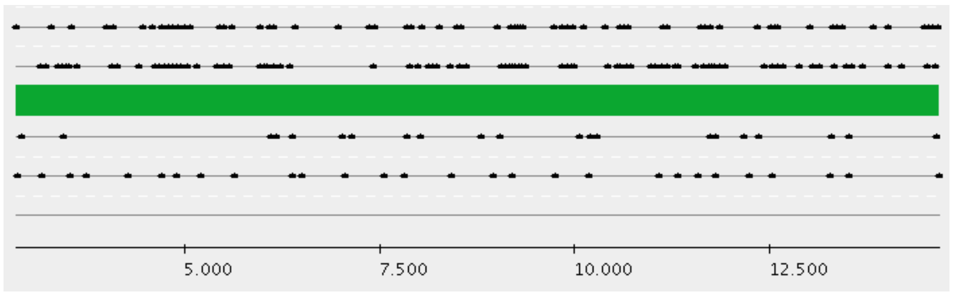


Figure 4: Screenshot of a bidirectional highway scenario with two lanes per direction displayed in `hwgui`. Black dots indicate node positions.

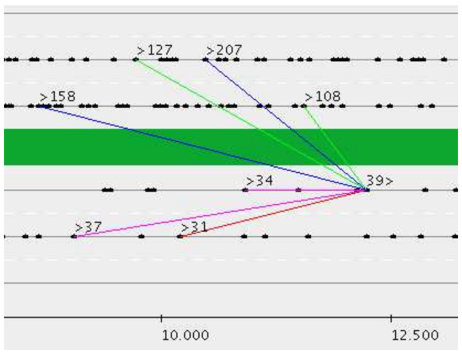


Figure 5: Screenshot of several communication pairs in the `hwgui` tool

the procedure of combining the cuts is done in such a way that the first cut is combined with the last, the second cut is combined with the last but one, and so on.

A set of five bidirectional movement patterns is thereby gained from one original DCX-file. These are used in a next step as a basis for generating communication patterns, i.e. nodes that will start a communication process with each other in the simulations are chosen randomly. Figure 4 depicts an examplary screenshot of `hwgui`'s highway section.

4.2.3 Communication patterns

For the simulations, a number of movement and communication patterns has been created with the `hwgui` tool. The movement patterns differ in two parameters: the node density per lane and kilometer and the number of lanes per direction. The resulting patterns emulate typical bidirectional traffic patterns on a common German highway. For each one of these patterns, random communication partners are chosen in a specific manner referred to as *single source scheme*.

In a communication pattern, a number of equally distributed node pairs are randomly selected and set up. The communication triggered between two nodes during a simulation run will be a stream of ping packets. Other than usual for a ping stream, the receiving nodes are set up in such a way, that a received ping packet will not be answered with an echo. This is done in order to avoid any effects caused by the optimization schemes of the routing protocol to emerge.

A set of communication patterns has been generated with the single source scheme. The different patterns vary in an increasing communication distance, i.e. the average distance between any two partnering nodes during the communication process. This distance starts with 1000 meters and goes up to seven kilometers in steps of 500 meters. With a transmission radius of 500 meters it is thus guaranteed that a packet is forwarded at least once by an intermediate node. The procedure for finding and setting up communication partners is as follows. First the source nodes for every communication stream are selected. This set of nodes stays fixed for each communication distance. Then the destination nodes are chosen in such a way that two preconditions hold. First, while sending packets, it is guaranteed that the sender and receiver will have a distance between them that falls in the interval $[\Delta_C - 500m, \Delta_C]$ with Δ_C denoting the communication distance. Second, all destination nodes have to be located either ahead of or behind their respective source node during the whole communication time. In order to obtain comparable results, it is important that each packet, that is sent from the same source node, will be routed preferably via the same intermediate nodes. This is assured by the scheme described above and by having a single source node start its communication process at the same time for each communication distance. Local discontinuities will then occur systematically rather than randomly so that they have a comparable effect over the simulations of the different communication distances.

As an example, Figure 5 shows a screenshot from the `hwgui` tool, that depicts a single source node and its respective partners from a set of communication patterns. The node with number '39' here serves as source node with the other nodes being the individual destinations.

It should be remarked that the different communication patterns are not simulated at the same time as the screenshot would suggest. Rather, each pattern is simulated individually, so that they don't interfere with each other. Nevertheless, when simulating the seperate communication patterns, a single source node will always start its sending process at the same time instance. This is hinted in the screenshot by the colored lines, that stand for an ongoing communication link.

As a side note it should be stated that every communication pair is chosen in such a manner that both sender and receiver are located in the same partition of the network. Thus, it is assured that a packet drop can be attributed to a shortcoming of the routing protocol and not to a non-existing path to the destination node.

4.3 Adjustments made to the network code

In order to separate the effects of the two optimization techniques implicit beaconing and lost link callback from each other, a switch has been added to the code of the PBR implementation to enable and disable the two features individually. The features can thus be managed through the OTcl interface and switched on or off in the OTcl control scripts as needed.

When having deactivated the implicit beaconing feature, the update of the neighbor table is bypassed upon receipt of a normal, non-beacon packet. So only real beacons will be evaluated.

When switching off the lost link callback feature, the selection of another suitable next hop is avoided in the callback function. So if the MAC layer detects the failure of delivering a packet, the data packet is dropped at once.

4.4 Evaluation scripts

One objective of this work is to give an empirical estimation of the per-hop packet loss probability p. Therefore the existing script for evaluating simulation runs was enhanced with code that will support this task.

In the evaluation step, a number of diagrams is output that depict several interesting aspects of a simulation. Such are, for instance, packet delivery ratio graphs, graphs with the number of packet collisions, or ping delay graphs. There have now been added three new diagrams that are explained in the following.

4.4.1 Drop distance histograms

The first new type of diagram shows a histogram of the drop counts with respect to the distance between the sender and the place where the drops occurred. This histogram comes with two different distance metrics. In the first type, the horizontal axis is divided into bins with a width of fifty meters. A dropped packet falls into that specific bin which contains the distance between the packet's source node and the node where the drop took place.

In addition to that, another kind of metric is used. That type of drop distance histogram uses a hop metric. Here, the drop count with respect to the number of hops the lost packets have travelled so far is shown.

For the creation of these histograms, only the simulations with the largest communication distance, i.e. 7000 meters for this work, are considered. Additional information is put into the histograms by displaying the average as a vertical bar and the standard deviation of the data.

Drop distance histograms are helpful for interpreting the obtained simulation data. If the individual forwarding processes are indeed independent from each other with one globally valid loss probability, it can be expected that all drops are occurring equally distributed over the entire communication range beginning at the sender and extending to the destination node. This means for the drop distance histogram that is subdivided into fifty meters bins that each bin is approximately filled with the same amount of drops.

4.4.2 Loss probability graph

This graph shows the average probability p that a packet drop occurs between any two nodes.

As already mentioned in Section 3, the characteristics of the position-based routing scheme suggest that the probability of a packet loss between two nodes is independent of the overall route length between sender and receiver. To confirm this, the observed packet loss probability is calculated and displayed in a graph as follows.

The process of forwarding a packet between two nodes can be interpreted as a random experiment, or more specifically, a Bernoulli trial. The forwarding of a packet either succeeds or fails. The entire operation of sending a packet from a source to a destination can thus be seen as a Bernoulli process where each forwarding event is a Bernoulli trial. In order to compute p, the total number of forwarding attempts together with the number of failures, i.e. packet drops, has to be counted. p can then be calculated as

$$p = \frac{\sum packet\ losses}{\sum single\ hop\ transmissions} \tag{2}$$

Here, the number of packet losses corresponds to the total drop count of a simulation. A single hop transmission is defined as an attempt to forward a packet to another node, regardless if it was successful or not.

The overall drop count is calculated as the number of sent packets minus the number of successfully received packets. Here, duplicate packets that have emerged and later been dropped during a simulation should actually be added, too. Since it turned out to be infeasible to identify them, they are ignored in this calculation. So, if there are any losses of this type, they distort the calculation of p a bit. But it has turned out that they have only marginal weight, if any. Duplicate packets are discussed in more detail in Section 5.2.1.

The loss probability graphs now show the development of p over the increasing communication distance.

4.4.3 Graph of estimated PDR

Based on p, estimated PDR graphs show the outline of the PDR provided that the assumption of the independence of the forwarding processes applies.

These graphs are generated by taking the packet loss probability as empirically determined with Equation (2) and using it together with Equation (1) to obtain the fraction of the number of sent packets that is estimated to be successfully delivered.

As stated before in Section 4.4.2, routing a packet from a sending node to a destination node is a Bernoulli process if the assumption made above holds true. For such a process, the probability that an event A will take place exactly m times if an independent random process is repeated n times is given by

$$P(A_n) = \binom{n}{m} \theta^m (1-\theta)^{n-m} \tag{3}$$

Here, θ is defined as the probability that the event A will occur if the random experiment is conducted only once [1].

If applied to the routing process of a data packet, θ can be interpreted as p, and event A can be defined as the occurrence of a packet drop. The problem of determining the probability that a packet is successfully delivered from end to end can thus be restated as follows: What is the probability that event A (a packet drop) does never occur when forwarding a packet n times, i.e. when transporting it via a route of n hops?

Applied with Equation (3), the problem yields

$$P = \binom{n}{0} p^0 (1-p)^n \tag{4}$$

which resolves to Equation (1).

For the generation of the graph the problem how to determine n arose. This was solved by taking the empirically observed average route length of each successfully delivered packet as an estimate for n.

In addition to the estimated PDR graph, another curve is drawn into these diagrams for comparison. In contrast to the graph described above, this curve shows an *idealized* PDR graph. While for the estimated PDR curves the fluctuating ps are taken for the calculation of each sampling point as they can be read off of the loss probability graph, the idealized PDR graphs are calculated with only one fixed loss probability. This one is determined by averaging all observed ps starting with a communication distance of 2500 meters. The lower communication distances are omitted as there are made only too few executions of the random experiment, i.e. there are too little single hop transmissions to yield a significant loss probability.

Since Equation (1) is calculated on a per-hop basis, a conversion had to be applied in order to adapt the route length n to the x-axis scaled in meters. To achieve this, it is calculated how much meters are covered by one hop on average. This mean hop distance is then used to compute the correct position on the x-axis according to the average route length for the different communication distances.

5 Simulations and results

In order to investigate the question at hand as described in Section 3, the influences of the different factors that affect the protocol's behavior are separately isolated and examined by means of running simulations with the ns-2 network simulator.

As already stated previously, three sources were identified that can be taken into account as a potential bias for the protocol's behavior: the installed location service, implicit beaconing and the lost link callback feature. Therefore, a simulation setup has been arranged with the latter two features alternately switched on and off and the location service replaced by simulator knowledge.

5.1 Simulation setup

A set of different movement scenarios has then been used for the simulations. All of these have a total duration of sixty seconds. The number of lanes per direction is two and three lanes, respectively. The node density per direction is a combination from a set of 2, 6, and 11 nodes per kilometer.

The communication patterns have been set up with varying parameters. In each scenario there are several pairs of communicating nodes that will start their sending process one after the other, so that no two pairs are communicating concurrently. The average distance between them during the sending process ranges from 1000 meters through 7000 meters with a stepsize of 500 meters, each step size forming a separate communication pattern that is simulated individually.

Two types of data traffic has been separately set up. The first type is constituted by single-packet data streams, i.e. between each communication pair only one ping packet is exchanged. To obtain a large sampling of forwarding processes, an amount of fifty communication pairs per scenario has been chosen for this type. For the second type, every data stream consists of thirty ping packets emitted with a sending rate of eight packets per second by a total number of ten communication pairs.

All setups were simulated with a beacon interval of 0.5, 1.0, and 2.0 seconds, respectively.

Every simulation setup is repeated ten times with the global random seed changed each time. This is done for obtaining relatively stable results with respect to the statistical distribution.

Furthermore, another minor setting was done for the simulations, namely adjusting the MAC retry count to one. This will have the MAC layer attempt the transmission of a packet only once. If that transmission fails the first time, the MAC layer will do no further effort but immediately informs the routing layer of the failure through a lost link callback.

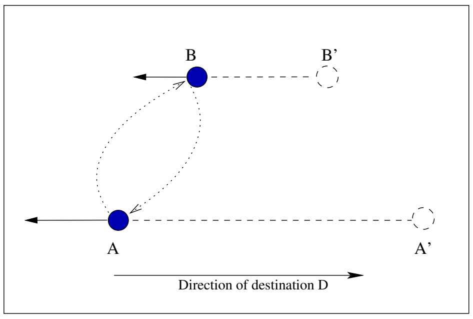


Figure 6: Emergence of routing loops

5.2 Observations

First of all, some general observations have been made throughout the simulations. These are outlined in the next sections.

5.2.1 Packet duplicates

A first observation in the simulation data can be done of packet duplicates. The emergence of duplicates can occur in all cases when the lost link callback feature is active. Duplicates can easily be identified by the destination nodes and will be silently dropped by them. A packet is duplicated when the following exemplary situation takes place. Node A intends to forward a packet to node B. This will cause the MAC-layer to transmit that packet over the channel to the next hop and to hereafter wait for an acknowledgement of reception. Node B then correctly receives the packet and sends the mandatory ACK-packet back to A. If this ACK-packet is now lost during transmission so that A doesn't receive it in time, A's routing layer lost link callback is invoked. This will have A retransmit the data packet. Such will happen each time an ACK-packet is lost and a subsequent lost link callback is issued. These unintentionally created packets will add to overall network traffic and must be kept in mind when making statements about the networks characteristics.

This occurrence will only happen if the lost link callback is enabled. Otherwise, if no ACK was received, the routing layer will drop the data packet under all circumstances without trying to retransmit it.

5.2.2 Routing loops

Routing loops are a phenomenon that can take place under certain conditions when the implicit beaconing feature is turned off. A loop can occur as exemplified in Figure 6.

If implicit beaconing is deactivated, information about neighbor positions can only be disseminated by explicit beacon packets. In Figure 6 node A wants to forward a packet to node B, as the destination is located in the east. Both nodes drive to the left during which A is overtaking B. The actual node positions are marked with A and B, respectively. However, the view on the network by the nodes according to their neighbor tables is a different one. The last time that A and B have received a beacon from each other they were both placed at the positions marked by A' and B'. Only these positions are relevant for them for making routing decisions. If now node A has to route a packet to destination D it will forward it to B, since it is aware of its own position and knows that B' is nearer to D. B in turn will directly send the packet back to A, because this node thinks that A is at position A' and thus nearer to D. Hence, a routing loop has emerged, and the packet is sent back and forth between A and B until the time to live counter drops to zero causing the packet to be discarded. Of course, situations other than the one depicted in Figure 6 are imaginable where routing loops can occur with implicit beaconing disabled.

In contrast, if implicit beaconing was enabled, the forwarded packet would have piggybacked the actual position of node A so that B could first have updated its neighbor table. The packet then wouldn't have been sent back, since B would have known that it is nearer to the destination node than A.

The occurrence of routing loops can be visualized with the drop distance histograms. In Figure 7 it can be seen that the majority of the data packets is dropped after 64 hops. Since the time to live (TTL) counter for each packet is initially set to 64, this indicates that these packets are discarded because the TTL counter has dropped to zero, which can only happen in case of a routing loop.

5.2.3 Effects of the location service

As already mentioned in Section 3, the deployed location service adds a significant delay to the delivery times of all data packets. This effect is induced by the rather time-consuming process of requesting the positional information of other nodes as described in Section 2.4. After this information has been obtained, the routing process of the data packets towards the destination can begin. It has to be borne in mind that the destination node keeps changing its position after having answered a location request. Thus, the discrepancy between the destination node's current position and the position that is carried to the requesting node in the location request's reply grows larger over time. If the delay of the data packets becomes too large, however, it may happen that the destination node has moved too far away from the position it occupied while answering the location request. The last-hop forwarding node in charge of finally delivering the data packets then

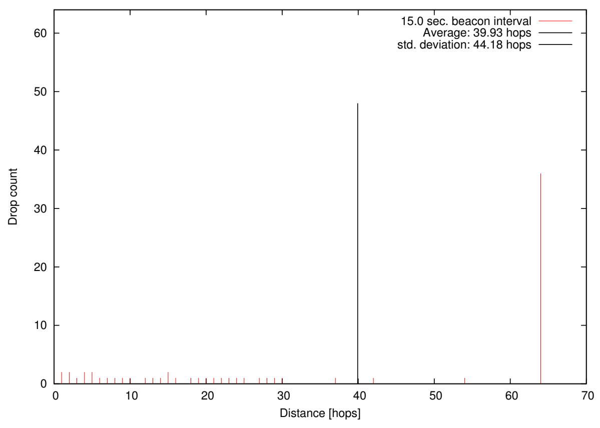


Figure 7: Drop distance histogram for a beacon interval of 15.0 seconds

doesn't find the destination node in its transmission range and is compelled to drop these packets. Requesting the destination node's position anew in such a case is not intended by the protocol. Therefore, the larger the delay and the higher the average node velocity is the larger becomes the probability of this sort of packet drops.

Another effect of the location service can be observed together with the implicit beaconing feature. Since a location request is accompanied by a lot of data traffic, such as flooding the request and its answering, many nodes are able to overhear that traffic and can thus update their neighbor tables with the most current positional information of the involved nodes. Location requests therefore facilitate the subsequent routing process. This is all the more significant the greater the distance between the requesting and the destination node is, and thus the larger the area affected by a location request becomes.

5.3 Simulation results

For the illustration of the simulation results, the simulation's output of a scenario with two lanes per direction and a node density of six nodes per lane in either direction is used here as a representative.

5.3.1 Enabling implicit beaconing and lost link callback

When simulating with implicit beaconing (henceforth abbreviated to IB) and lost link callback (LLC) enabled and with the location service replaced by simulator knowledge, it is noticeable that with the beacon intervals set to 0.5, 1.0, and 2.0 almost all packets reach their destination, so that the PDR is very close to or exactly 100% all the time. In Figure 8 the PDR

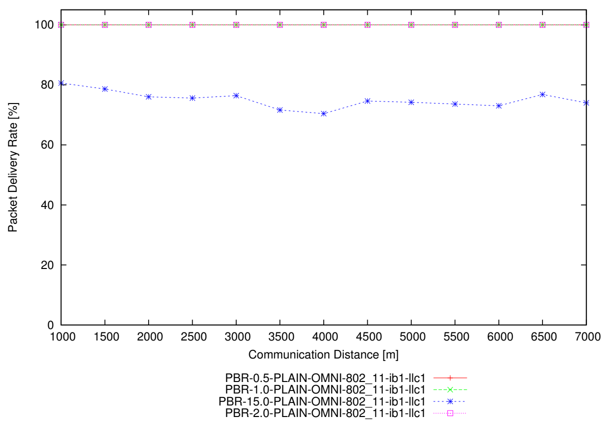


Figure 8: PDR with IB and LLC enabled

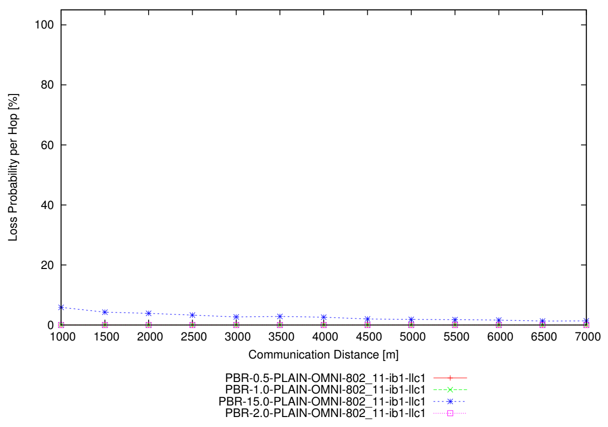


Figure 9: Loss probability graph with IB and LLC enabled

graph for this situation is shown. In order to discern any interesting effects, a larger beacon interval of 15.0 seconds therefore has also been simulated for comparison. Of course, such a large beacon interval imposes a number of problems on the interpretation of the simulation's data, since it is not very realistic. For now, these problems shall be put aside as some interesting effects can still be observed with such a large beacon interval.

As can be expected, the loss probability for this case is close to or equal zero. Figure 9 shows exactly this.

Taking this empirically determined loss probability and calculating the estimated PDR graph results in the graph depicted in Figure 11(a). In this sort of graphs, the red line represents the idealized PDR. Here, it can be clearly seen that for a beacon interval of 15.0 seconds a single constant loss

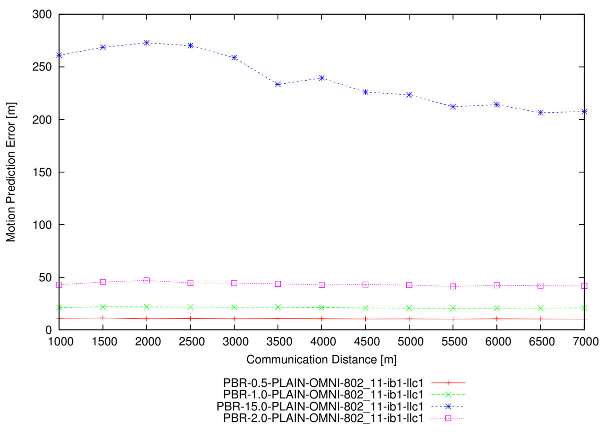


Figure 10: Motion prediction error

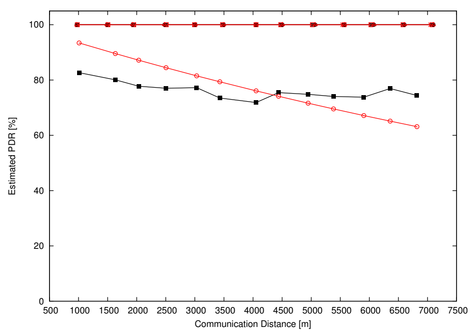
probability cannot be assumed for every communication distance. Rather, as shown in Figure 9, the packet loss probability is decreasing with an increasing communication distance. Therefore, the estimated PDR curve is almost horizontal while the idealized PDR curve as decribed in Section 4.4.3 is sloping downwards.

A hint why this could be the case gives the motion prediction error as shown in Figure 10. The motion prediction error is defined as the average distance between the actual positions of the next hop nodes and their presumed positions as recorded in the forwarding nodes' neighbor tables. For example, the distance between the positions marked by N_1' and N_1 in Figure 2(a) is included in the motion prediction error. For the case with a 15.0 seconds beacon interval this type of error constantly decreases with the increasing communication distance (see Figure 10). This means, in effect, that the routing decisions become better over time.

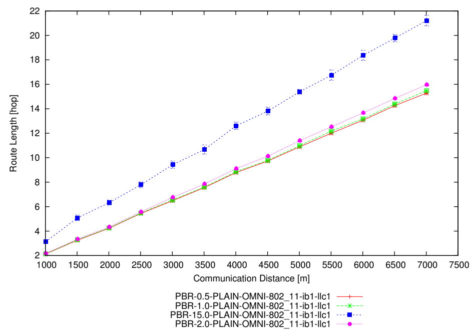
This effect can be attributed to the IB feature. With IB turned on, a piggybacked beacon that accompanies a data packet through the network will update the neighbor tables of all nodes that overhear it along the way. Thus, the routing process for a packet that afterwards travels along a route of thus updated nodes benefits from the preceding implicit beacon.

The larger the beacon interval is, the more important are implicit beacons, since they are the only means of updating the neighbor tables for the long time span between two ordinary beacons. In the situation depicted in Figure 10, this effect becomes the more striking, the larger the communication distance is. This is due to the fact that with a larger communication distance, the area that is affected by the piggybacked beacons becomes larger, too. So, the chance of routing over nodes with more up-to-date information also grows larger.

The mean route length that is taken for n in Equation (1) is depicted

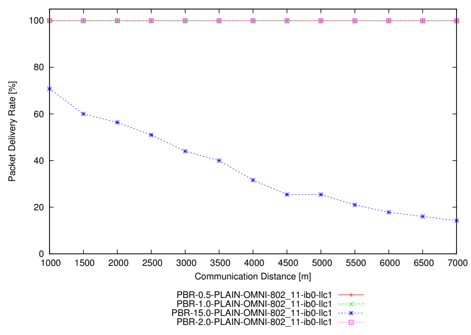


(a) Estimated and idealized PDR

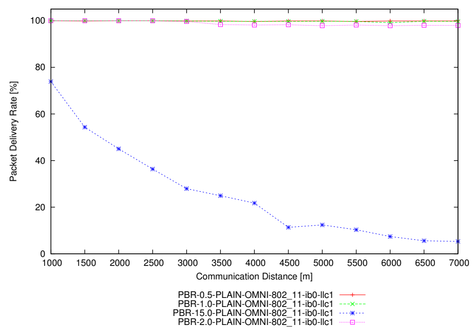


(b) Average route length

Figure 11: Both features turned on



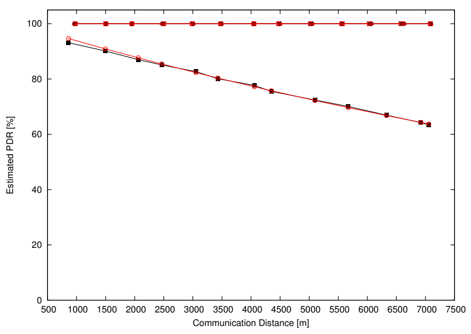
(a) 6 nodes per km and lane



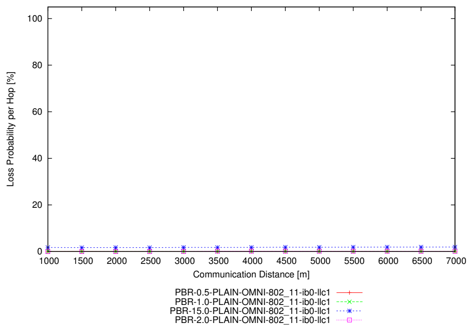
(b) 2 and 6 nodes per km and lane

Figure 12: PDR curves with IB disabled

in Figure 11(b). This type of graph shows the average number of hops that every sucessfully delivered packet has taken on its way. Here, it is noteworthy that the larger the beacon interval becomes, the longer the average route lengths get. This is due to the fact that with an increasing beacon interval the information in the network nodes become more and more apt to be stale and unreliable. Routing decisions thus get increasingly unprecise. A node may even be inadvertently routed backwards. Such a negative progress can happen when a next hop node's position in reality is behind the forwarding node, whereas its entry in the forwarding node's neighbor table says that it is ahead of it. This is what takes place in a routing loop (see Section 5.2.2), for example.



(a)



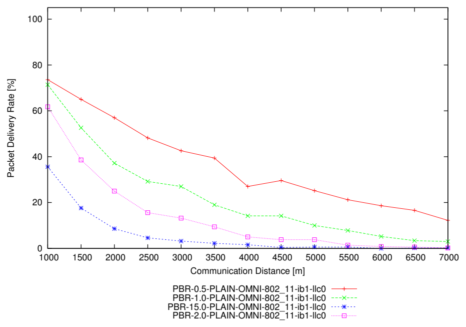
(b)

Figure 13: Estimated PDR and loss probability with IB disabled

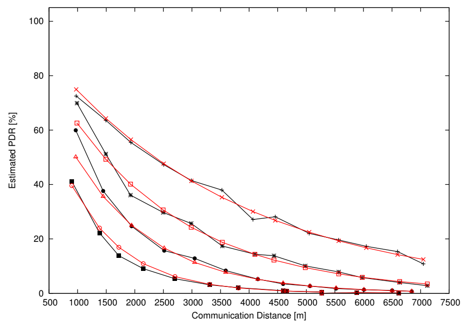
5.3.2 Disabling implicit beaconing

The next step in isolating the PBR optimization features is to only switch off IB. The PDR graph for the simulations with IB disabled shows Figure 12(a). Here, it is not possible to descry any significant effect that the missing IB has, as the delivery rate is continuously at 100% for the 0.5, 1.0, and 2.0 seconds beacon intervals. For comparison, another PDR graph is shown in Figure 12(b) from a scenario with a lower node density which is in either direction 2 and 6 nodes per kilometer, respectively. There the PDR slightly deteriorates. This is explicable by the fact that with a higher node density, a node's neighbor table is filled with more entries than with a sparse density. Thus, the routing algorithm has a higher chance of finding a reachable neighbor with its trial and error scheme.

In the case at hand, a shortcoming of the method of estimating the loss probability as described in Section 4.4.2 becomes apparent. In Figure 13(b) it can be clearly seen that the estimation for the loss probability for a beacon interval of 15.0 seconds is far too small. This faulty estimation results in the estimated PDR graph shown in Figure 13(a), which is too optimistic since the observed PDR curve (see Figure 12(a)) runs much lower. An inspection of the drop distance histogram offers an explanation for that. Figure 7 shows the drop distances in hop metrics for that case. The great amount of routing loops, identifiable by the 64-hop drops, account for a large number of registered single-hop transmissions which are then responsible for the distortion in the calculation of the loss probability. Single-hop transmissions appearing in a routing loop shouldn't be counted in Equation (2), since these do not relate to the random experiment of forwarding data packets.



(a) Observed PDR graph



(b) Estimated PDR

Figure 14: Lost link callback switched off

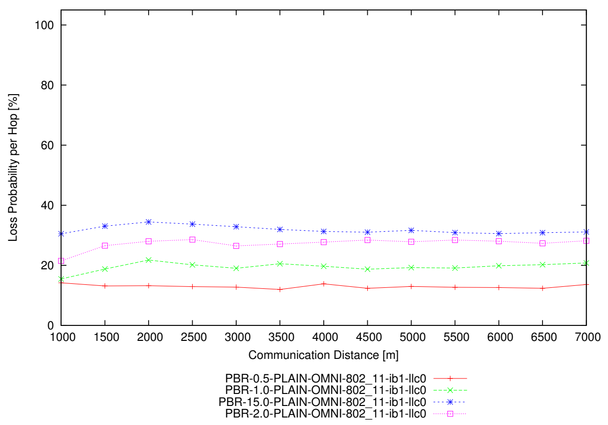
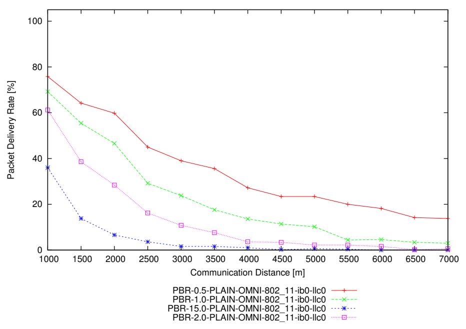
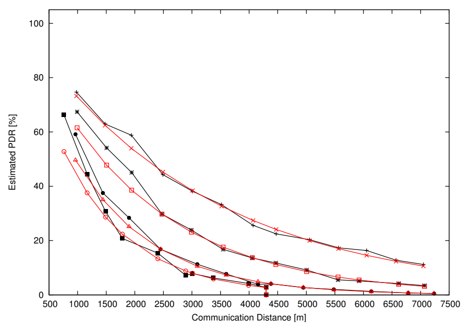


Figure 15: Loss probability with only LLC switched off



(a) Observed PDR graph



(b) Estimated PDR

Figure 16: Both features switched off

5.3.3 Disabling lost link callback

The by far largest effect on the behavior of position-based routing is exerted by the LLC feature. With LLC switched off, the PDR curve attains the expected convex curvature as described in Section 3. Figure 14(a) depicts this. Also, the loss probability remains comparatively constant regardless of how large the communication distance becomes (see Figure 15). This fact supports the assumption that the greedy forwarding processes are independent from each other and are governed by a single, globally valid loss probability. If Formula (1) is applied to these probabilities, an estimated PDR graph is obtained as in Figure 14(b). Since the loss probability's variance is small for all communication distances, the idealized PDR curves nestle closely to the estimated curves.

Comparison of Figure 14(a) and Figure 14(b) implies that the loss probability as empirically determined with Formula (2) is here a good estimate for p.

In Figure 14(a) it can be observed that the beacon interval directly correlates with the packet delivery ratio. This is due to the better and more timely information the nodes have about their neighbors when the position updates are spread more frequently.

5.3.4 Disabling both implicit beaconing and lost link callback

If in addition to LLC the IB feature is turned off, too, no significant variation can be made out in the graphs. In Figure 16(a), Figure 16(b), and Figure 17 it can be seen that the curves are almost identical to those of the previous paragraph. This can be explained by the fact that in this simulation setup only one ping packet is exchanged between sender and receiver. Implicit beaconing can only have a tangible effect if more than a few packets travel through the network. The first packet will disseminate the most

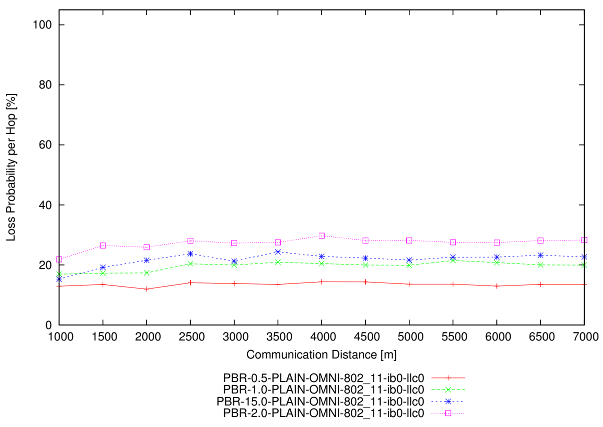


Figure 17: Loss probability with both features switched off

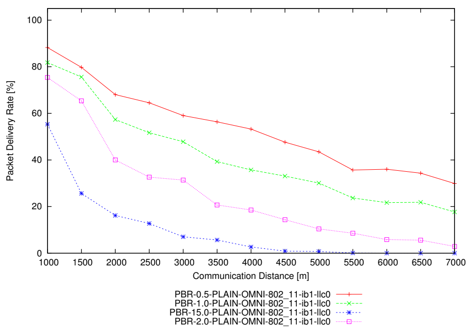
current positions of the forwarding nodes among all neighboring nodes on its way through the network. For example, an echo reply packet could take advantage of this positional information update, since the routing decisions made for it can be based on very recent information. IB thus has a positive effect on the PDR because it virtually increases the beacon interval of the forwarding nodes to the packet sending rate in the area that is traversed by data packets. In addition, it favors succeeding packets that are routed through the same area, regardless whether these originate from the same source.

A more palpable impression of the effects IB has can be observed when looking at the graphs of simulations in which each communication pair exchanges an amount of thirty ping packets that are answered with echo packets. Then the information updates caused by travelling data packets increasingly facilitate the overall routing process the larger the area becomes that is affected by IB. See Figure 18(a) and Figure 18(b) for the PDR of the case with 30 ping packets. These show that the routing results are slightly better with IB switched on.

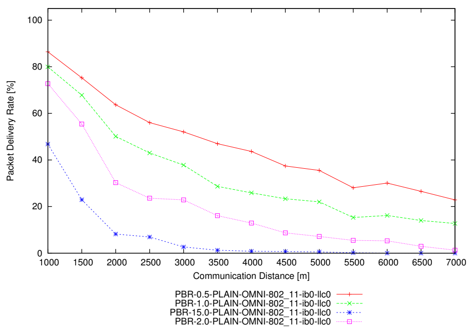
Figure 19 shows the drop distance histogram for a beacon interval of 0.5 seconds. There it can be seen that the drops are distributed approximately equally over the whole range of 7000 meters. This meets the expectations as stated in Section 4.4.1.

Once again, however, the loss probability's estimation for a beacon interval of 15.0 seconds is too small. Inspection of the drop distance histogram once more reveals the occurrence of routing loops (see Figure 20(a)).

Figure 20(b) indicates another problem of the loss probability's calculation. With a large beacon interval such as 15.0 seconds and both protocol optimizations switched off, it becomes very hard to transport any data packets to the destination at all. As can be seen in Figure 20(b), most of the



(a) with IB enabled



(b) with IB disabled

Figure 18: PDR graphs for data traffic consisting of 30 ping/echo pairs and LLC disabled

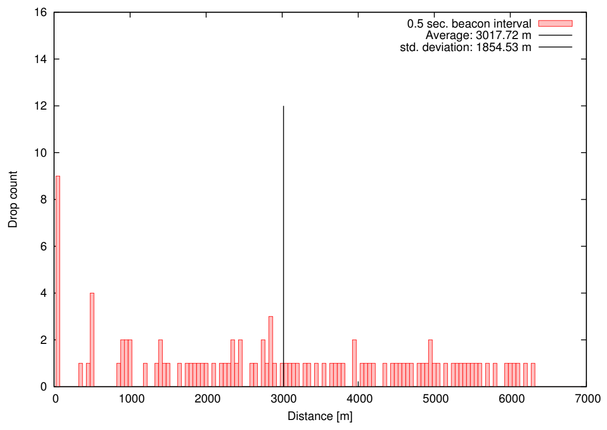
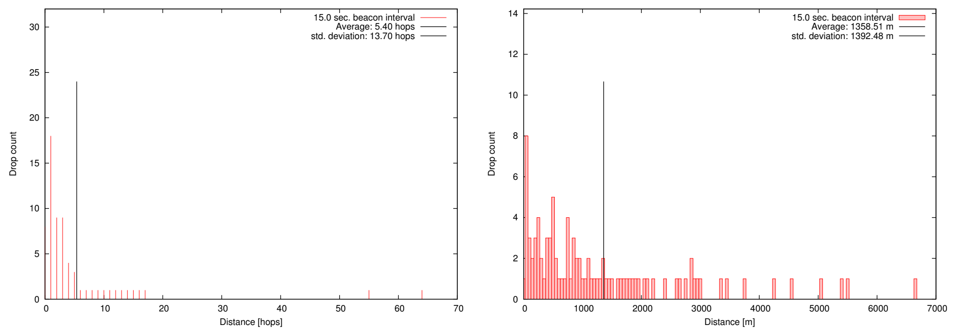


Figure 19: Drop distance histogram for a beacon interval of 0.5 seconds



(a) Drops per hop

(b) Drops per 50 meters

Figure 20: Drop distance histograms for a beacon interval of 15.0 seconds and both optimizations switched off

packets are lost very early. With that, the number of executed Bernoulli trials decreases, since packets that are lost early cannot later on contribute to the random experiment anymore. This leads to a decreased random sampling which distorts the overall results. Simulations as they are conducted for this work therefore prove to be inappropriate to fully examine the problem at hand. A solution to this problem is suggested in the next section.

6 Conclusions and future work

In this work an examination of the loss behavior of the Position-Based Routing protocol was presented. By separately isolating the individual optimization techniques implicit beaconing and lost link callback, the effects of these optimizations on the protocol's behavior were exposed and described. It was shown that the by far greatest impact on the protocol's performance is evoked by the lost link callback feature. Switching it off results in a distinct drop of the packet delivery rate. In addition, it was demonstrated that as expected a globally valid single loss probability which affects the overall probability of a successful packet delivery can be assumed when the protocol is stripped off of its optimization features. Evidence was provided for the presumption that the basic greedy forwarding routing process is in fact a Bernoulli process. Furthermore, some observations could be made when disabling the implicit beaconing feature. First of all, without implicit beaconing it becomes possible for data packets to get entrapped in routing loops. Secondly, implicit beacons basically adjust the beacon interval to the packet sending rate in the area that is traversed by them. Thus, subsequent forwarding processes can take advantage of the more up-to-date neighbor tables, so that there is a higher chance for them to successfully transmit data packets.

The beacon interval that is chosen for running a simulation has been proven to be the major influencing factor for the protocol's performance. The larger that interval is set, the more the overall delivery rates deteriorate. It has been observed that with an increasing beacon interval the motion prediction error increases, too. This directly results in longer average route lengths with the chance of forwarding packets with negative progress growing larger accordingly.

Finally, after examining the drop distance histograms, it became apparent that the determination of the global loss probability by means of simulating real world scenarios with ns-2 is insufficient. Since a packet that is dropped early on its traversal through the network can't contribute to the random experiment of forwarding network traffic any longer, the results become increasingly unreliable if much of the overall amount of data packets is lost prematurely. A future approach to tackle this problem can be conceived by changing the implementation of the random experiment. Instead of simulating traffic scenarios as done in this work, one could, for instance, only count the number of drops without actually discarding the dropped packets. Thus, each data packet would invariably reach its destination yet will be possibly counted once or more often as dropped.

A more natural way of determining the loss probability could also be done by not setting up any communication partners in ns-2 at all. By extending the `hwgui` tool appropriately, it is conceivable to randomly choose any two neighboring nodes and check the probability of a successful packet forwarding process at a particular time instance between them. This could then be repeated and aggregated over a large number of randomly chosen neighbors. Since the forwarding processes are independent from each other and do not depend on the chosen communication distance either, this proceeding is admissible.

References

[1] Oskar Anderson, Werner Popp, and Manfred Schaffranek. *Schätzen und Testen*. Springer, Berlin, 1997.

[2] Holger Füßler, Martin Mauve, Hannes Hartenstein, Michael Käsemann, and Dieter Vollmer. A Comparison of Routing Strategies for Vehicular Ad Hoc Networks. Technical Report TR-02-003, Department of Computer Science, University of Mannheim, July 2002.

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

[4] Roland Krüger, Holger Füßler, Marc Torrent-Moreno, Hannes Hartenstein, and Wolfgang Effelsberg. Statistical Analysis of the FleetNet Highway Movement Patterns. Technical Report TR-2005-004, Department of Mathematics and Computer Science, University of Mannheim, 2005.

[5] Martin Mauve, Hannes Hartenstein, Holger Füßler, Jörg Widmer, and Wolfgang Effelsberg. Positionsbasiertes Routing für die Kommunikation zwischen Fahrzeugen. *it + ti*, 44(5):278–286, October 2002.

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

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

[8] HWGUI project page.
http://www.informatik.uni-mannheim.de/lib/projects/hwgui/.