Experimental evaluation of flooding in ad-hoc networks

Bachelor-thesis
by

Andreas Tarp

from
Düsseldorf

Lehrstuhl für Rechnernetze
Prof. Dr. Martin Mauve
Universtität Düsseldorf

Dezember 2004

Advisor:
Dipl. Wirtsch.-Inf. Wolfgang Kieß

Acknowledgments

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

First of all, I would like to thank Wolfgang Kieß for his support, ideas, valuable comments and the interesting discussions.

I would like to thank Thorsten Ingenrieth for helping me with his language skills and Stephan Zalewski for GPS measurements and installing Linux on the iPAQs.

Furthermore I would like to thank Ulrike Tarp, Thomas Schmidt, Andreas Drexler, Monika Kottwitz, Thorsten Ingenrieth, Britta Winterschlade, Stephan Zalewski, Christian Lochert, Markus Koegel, Thomas Ogilvie, Alfonso Cervantes and Andreas Barthels for looking out for the iPAQs during our experiments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conventional networks (cable-based or wireless) depend on an available infrastructure. The components of such a conventional network can be divided into end systems and administrative systems such as routers, switches, access points etc. A different approach is used by ad-hoc networks. Nodes work as end systems and routers at the same time. In contrast to conventional networks, they are able to operate in a scenario where no infrastructure is available such as disaster areas, car to car communication and battlefields.

There are two different kinds of ad-hoc networks: Static and mobile ad-hoc networks (*MANETs*). Unlike in static networks node movement is allowed in MANETs. The device movement in MANETs leads to a more complex behaviour of the network in comparison to a static ad-hoc network.

As the understanding of the less complex case is a prerequisite for understanding a more complex case we first concentrate on static ad-hoc networks, which only be called "ad-hoc networks" in the following paragraphs.

1.1 Problem Statement

A great number of routing algorithms for ad-hoc networks have been tested in simulations [14, 8]. To this day there are only a few field tests with real end systems [17, 11]. In contrast to simulations with hundreds or thousands of nodes, field tests using a simple protocol such as flooding have shown that in reality the

behaviour of an ad-hoc network can be very complex and different to simulations. The authors of [12] present an evaluation of flooding in a wireless sensor network with up to 169 nodes arranged in a dense grid. Unlike the expected behaviour that packets spread steadily hop by hop away from the original source until every node of the network has received the packet, they found out that not all nodes receive all packets. Another important aspect of their research is that the existence of so called *backward links*, which means that a node receives a packet from a node further away from the original source, and so called *long links*, which is the term for the fact that the length of the link is longer than the expected radio range, was discovered.

The intention of this thesis is to obtain experience with field tests on our hardware and to evaluate our results. In addition we want to know, if the results of the above mentioned sensor network are similar to the behaviour of our own hardware.

1.2 Contribution

In this paper we present an experimental evaluation of flooding in a static ad-hoc network consisting of seven to ten nodes. We introduce the underlying software and a strategy to distribute important status information to all participating nodes. Our experiments show that 802.11 wireless networks behave similar to the sensor network described in [12]. We point out, that the expenditure of time needed to perform our experiments is very high, because of the high demand of man power. The ratio between the time actually needed for flooding and spend man-hours is 1:68.75.

1.3 Structure

We present our experimental platform in Chapter 2. In Chapter 3 we give insights into the parts of the software which we used and had implemented before. A description of our flooding experiments and a detailed analysis of the captured data is presented in chapter 4. Chapter 5 contains a conclusion.

Chapter 2

The Test System

In this chapter we present our experimental platform. It contains the hardware and the software employed.

2.1 Hardware

Our test system consists of ten HP iPAQ 5500 PDA's with integrated 802.11b [7] WLAN and a 400MHz XScale CPU. The relevant technical specifications are given in Table 2.1:

CPU	400MHz
ROM	48MB
RAM	128 MB
WLAN	802.11b

Table 2.1: Technical specifications

802.11b:

The 802.11b standard uses the unlicensed 2.4 GHz band. This band is also used by e.g. *Bluetooth* and microwaves. To avoid packet collisions 802.11 uses *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). CSMA/CA verifies that the channel is not in use before sending packets. If the channel is in use, the

device pauses for a randomly chosen small interval and then tries again. Full particulars can be found in [7].

2.2 Software

The underlying Operating System of the IPAQs is a *Familiar Linux* distribution [1] which was adapted to our needs. Our flooding algorithm is written in Click [15], a modular router developed by the Massachusetts Institute of Technology (MIT).

2.2.1 The Click Modular Router Project

A Click router consists of so called *elements* which are connected to build the router. Every element is responsible for exactly one job. There are elements to reduce the *time to live (TTL)* field, to calculate *CRC* checksums or to queue packets. Elements are written in C++ and therefore it is possible to add new functions.

The connection of the elements takes place in a so called *router configuration*. It is written in a simple language as the following easy example of a click router in Figure 2.1 shows. In the example incoming packets are caught from one network device, then the Ethernet Header of IP packets are modified and in the last step the packets are sent out to another device. All other packets are discarded:

To control the flow of packets inside the router the elements have multiple in- and outputs. This structure enables elements to send packets according to requirements like source or destination on different ways through the router configuration. Elements with their in- and outputs are connected in the following way:

```
element1[a] -> [b]element2[c] -> [d]element3
```

`element1`, `element2` and `element3` represent three different elements. `a` and `c` are output ports of their elements, `b` and `d` are input ports. Port numbers start with 0. It is allowed to leave out the declaration of the first ([0]) in- or output of an element as done in the above example click configuration for

```
class::Classifier(12/0800, -);

FromDevice(eth0)
-> class
-> Strip(14)
-> EtherEncap(0x0800, 00:02:8A:B7:C0:AA, FF:FF:FF:FF:FF:FF)
-> Queue
-> ToDevice(eth1);

class[1]
-> Discard;
```

Figure 2.1: Example click configuration

`element1`. An arrow "->" is used to connect two elements and their in- and outputs respectively with each other.

The first line of our example (Figure 2.1) contains a declaration of a new alias `class` for the element `Classifier(12/0800, -)`. The alias `class` can be used instead of `Classifier(12/0800, -)` for the rest of the configuration. The `Classifier` element sends different packets to different outputs depending on a filter expression. A list separated by comma contains the expressions: `(expression0, expression1, expression2, ...)`. Packets complying `expression0` are send to output `[0]`, packets complying `expression1` to output `[1]` and so on. In this case `12/0800` as `expression0` represents IP packets. `0800` is the entry in the type field of the ethernet header reserved for IP packets. `12` is the position of this type field in a packet (the 12th byte). `Expression1` is a special symbol "-" representing all incoming packets that are not covered yet. So all IP packets are sent to output "[0]", every non IP packet to output "[1]".

The element `FromDevice` with the option `eth0` grabs incoming packets from ethernet device `eth0` and passes them to the next element, in our case the `Classifier` element represented by the alias `class`.

`Strip` with `(14)` as option deletes the first 14 bytes of the packet to remove the ethernet header having a length of 14 bytes.

After this, `EtherEncap` writes a new ethernet header containing the type of packet (`0x0800` represents IP packets), source (`00:02:8A:B7:C0:AA`) and destination (`FF:FF:FF:FF:FF:FF`) MAC address at the beginning of the before `stripped` packet.

Then the packet is passed onto a first-in-first-out (FIFO) `Queue` before it is sent out on ethernet device `eth1` by the `ToDevice` element.

The last two lines contain a connection between output `"[1]"` of `class` (as an alias for the `Classifier` element) and a `Discard` element. As all non-IP packets are passed onto output "[1]", this is used to discard all non-IP packets arriving at `eth0`.

2.2.2 Trace Tools

To log all data relevant for later analysis we need a tool which is able to capture every packet including all headers from a network device. The Linux tool *tcpdump*[5] is used for this purpose. It stores the captured packets in a binary file format that uses only a small amount of disk space, which is limited on mobile devices.

We use different packets for flooding and sending status information. Their characteristics for identification are stored in the first eight bytes of the payload. Therefore all necessary information to definitely identify a packet is available in the header and the first eight bytes of the payload. This setting reduced the amount of data to capture to the first 50 byte of a packet (42 byte header length, 8 byte payload). *Tcpdump* is started in the following way:

```
tcpdump -s 50 -w filename.cap
```

- "-s 50" logs only the first 50 byte of each packet
- "-w filename.cap" captures data into the declared file

2.2.3 Analyse tools

To convert the binary data stream created by *tcpdump* into ASCII text files we use Tethereal [10], a command line version of the network analyser Ethereal [9].

These text files containing the information captured by *tcpdump* are analysed by a small number of *perl* [4] *scripts* which were implemented for this purpose. Their task is to analyse the raw data and to write the results to disk. These results are processed by *gnuplot* [2] to visualise them and to make them easier to interpret.

Chapter 3

Implementation

This chapter provides an insight into the two self programmed click elements we use, into the layout of our three click configurations and into the way we chose to automate running more than one test consecutively.

3.1 Flooding

Flooding is one of the simplest protocols to spread data in ad-hoc networks. Every node rebroadcasts every packet it receives exactly once. Duplicate packets are filtered to avoid infinitive looping and to limit network traffic. So every packet sent out somewhere in the network spreads hop by hop until it has reached all nodes. As a result of packet collisions or packet loss caused by the surrounding of the network, nodes could be prevented from receiving a packet.

Flooding in ad-hoc networks is usually used to find routes between end systems like in DSR [13] or AODV [19]

Algorithm 1 presents a schematic overview of the operational sequence of flooding.

Algorithm 1: Flooding:

```
if (packet received for the first time) then {
   store packet ID;
   decrease time to live field (TTL);
```

```
   change source field in ethernet header to own ID;
   rebroadcast packet;
   }
else {discard packet;}
```

3.2 Operational sequence of an experiment

It is well-known that in wireless networks link quality between nodes varies [16, 17] and the rate of packet losses is contrary to cable based networks not negligible. Furthermore even in static wireless networks link quality can change as time goes on, because of physical disturbers like cars or people moving within the test environment without participating the actual experiments. To be able to explain the ratio of packet losses caused by bad links in our flooding tests, we need to measure the link quality of every existing link in our environment undisturbed by packet collisions or other internal network effects. If for example a node has only bad links to all neighbours even in such a measurement, this also influences flooding in a negative way.

To achieve this measurement requirement we have to run a series of measurements which are liable to the following regulations:

- to avoid packet collisions only one device is allowed to send packets at a time
- one after another every device sends out some measurement packets
- every node logs all incoming measurement packets

We flooded packets with different size and sending rate. These flooding packets differ from measurement packets in payload. To obtain statistically more secured data tests with every combination of packet size and sending rate were performed several times. This procedure is very time consuming and therefore link quality is likely to change during our experiments. Because of this it is necessary to run link measurements after each flooding cycle.

To be able to measure link quality between two flooding cycles the devices have to switch between two operational modes:

- **Flooding mode:** Nodes in flooding mode respond to flooding packets as described in Algorithm 1.

- **Measurement mode:** Devices in measurement mode send measurement packets at a time, when no other node is sending. Afterwards the node waits for a calculated time (still capturing all incoming packets by *tcpdump*) until all other nodes finished sending measurement packets. Then *tcpdump* is restarted to log the next flooding/measurement cycle.

The basic idea to combine these two modes is to use a domino effect to activate measurement mode at all nodes one after another. Nodes are placed in a way that a functional route through the whole network exists containing every node exactly once. A example for such a node placement is shown in Figure 3.1.

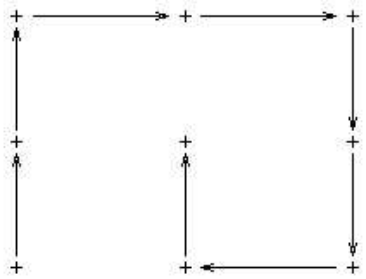


Figure 3.1: Example node placement

Following this route it is possible that each node activates measurement mode at the subsequent node in the route by sending his own measurement packets. Due to the source information stored in the ethernet header of incoming packets it is no problem to differ between all nodes in the network. This way a node can determine the time to switch to measurement mode by looking out for measurement packets of his predecessor. In measurement mode a node

- waits at first for a calculated time to make sure his predecessor finished sending (the interval depends on the amount of measurement packets sent by the node)

- sends his own measurement packets

- waits again until all other nodes have finished sending in measurement mode (the interval depends on the number of nodes in the network)

To check success of the measurement only the last node of the route mentioned above has to be checked. If it has changed state, it is guaranteed that all nodes have sent out their measurement packets because of the domino effect the last device only changed state if all others have changed state before. Therefore all nodes are ready for the next flooding cycle. If not, one link has failed (for example because of physical changes in the surrounding) and the nodes have to be checked sequentially to find the broken link. Then the measurement has to be activated again at this node and will continue until all nodes send their measurement packets.

Advantages of this operational sequence:

- It makes it possible to spread information all over a network ensuring that every node has received it (in our case the measuring packets)

- Only the device sending the original packets and the last device of the measurement route need to be supervised by technical instructed assistants. If measuring fails it can be detected at the last node

Disadvantages:

- It is not possible to randomly place nodes because of the requirement of a special working route containing every node in a set order

- Node placement needs more time

In our experiments it was no restriction to place nodes in this order, so that this disadvantage was not relevant for us but it needs more time than random node placement.

To realise the above described solution we use a Linux shell script. It ensures that the *click* configurations and *tcpdump* are started and that the device pauses after sending measurement packets.

3.3 Click elements

Most click elements necessary for our configuration were provided by the latest click release (version 1.4.1). For our experiment we had to write two additional elements. One to filter duplicate packets, to prevent infinitive looping during flooding and one to to stop the running click router from inside the router.

3.3.1 IpDupeFilter - the duplicate filtering element

The *IpDupeFilter* element is designed to filter duplicates of incoming packets on the basis of IP sequence numbers. The basic idea of the element is

- to save a 32 bit packet identification of the last x packets for every source IP
- to discard packets if their identification was saved before

The actual value for x is passed as an option to the element by the click configuration file. The 32 bit packet identification (ID) consists of the following values stored in the IP header of each packet:

- the 16 bit sequence number field
- the 3 bit fragmentation flags field
- the 13 bit fragmentation offset field

The values of fragmentation flags and offset are not necessary for our experiments but stored in case of using *IpDupeFilter* in networks with potentially fragmented packets.

As the sequence number field contains only 16 bit there are only $2^{16} = 65536$ different sequence numbers. To avoid the problem of ID collisions if a source sends more than 2^{16} packets and therefore the sequence numbers wrap around to before used ones the packets ID cache only contains the ID's of packets not older than 30 seconds because we assume that ID's do not wrap around in this time.

New packets are sent to the first output. If a packet is identified as a duplicate, it is passed to the second output of the element, which should be connected to a discard element.

3.3.2 Exit - an element to stop a running click router

To switch between flooding and measurement mode we need to stop the running click router if a special measurement packet[1] is received by the node as described in Section 3.2. The detection of these packets is performed by the *Classifier* element, which is described in Section 2.2.1. These packets are passed to our self written *Exit* element. This element calls a function that immediately stops the click router. So it is possible to stop the click router by sending a special packet to the node.

3.4 Click configurations

We developed three different click configurations:

1. The packetgenerator configuration that generates and sends the flooding packets. It runs on a single node during our experiments.

2. The flooding configuration that performs the flooding on all nodes but the original flooding packet source. Nodes do not generate own packets, they only rebroadcast incoming packets.

3. The measurement packetgenerator configuration that generates and sends measurement packets. It runs on all nodes.

3.4.1 The flooding configuration

The flooding configuration checks if a packet is a flooding or a measurement packet by using the *Classifier* (Section 2.2.1) element.

It ensures that for flooding packets

- the incoming IP packet is valid (*CheckIPHeader*)[2]

- duplicates are filtered (*IpDupeFilter*, Section 3.3.1)

[1] every node sends out more than one measurement packet. The minimum number we used was 20. The probability to miss all of them is small

[2] click element names in brackets

- the time to live (TTL) field is decreased as long as it is not expired (*DecIPTTL*)

Once the packet has passed the above mentioned inspections it gets a new ethernet header and is sent out again.

Measurement packets are discarded or passed to the *Exit* (Section 3.3.2) element that stops the click configuration depending on their source as described in Section 3.2.

3.4.2 The packetgenerator configuration

This configuration generates raw data, encapsulates it with IP and ethernet Header and passes it onto the network device. The number of packets to send, the sending frequency and the size of the packets is adjustable. The configuration quits after sending all scheduled packets. The first few bytes of the generated raw data (the packets payload) mark these packets as flooding packets. The *Classifier* element in the flooding configuration (Section 3.4.1) checks these bytes. This configuration runs on only one node and is the original source of all packets, that are flooded by all other nodes.

3.4.3 The measurement packetgenerator configuration

This configuration is related to the packetgenerator configuration (Section 3.4.2). The difference can only be found in payload. The first few bytes are different to the bytes used in the packetgenerator configuration. This configuration runs on all nodes to send measurement packets as described in Section 3.2.

Chapter 4

Experiments and Evaluation

In this chapter we present a description of the flooding experiment setup and an detailed analysis of the captured data.

4.1 Initial setup

In the early stages of our work we planned to conduct our experiments in the open country, to avoid interferences. The concept was to

- be as undisturbed as possible by other transmitters like wireless networks etc.
- get direct line of sight between all nodes
- place nodes in a grid
- place immediate neighbours in radio range with low packet losses
- place second-grade neighbours out of each others range

To estimate the dimension of the required open country we measured the radio range of our iPAQs by sending ping-packets from one device to another and estimating the loss rate by counting the lost packets. We observed a interesting behaviour: On direct line of sight there were very different rates of reception every few metres. The greatest open country in the vicinity had a maximum dimension

of about 300 metres. Over this distance nearly 100% of the packets sent out by one node were received by the other one. In the absence of a bigger field we measured the reception rates for greater distances in another non planar surrounding: Within the distance of about 950 metres there were areas with great packet loss up to 100 % and areas with nearly perfect receptions. Sometimes these different areas were very close. Even at the maximum distance of 950 metres we got a very low packet loss of only about 20%. This radio range might be a result of the surrounding, because one of the two nodes was placed between two small embankments which potentially were able to concentrate radio waves. An experiment to solve the question of the maximum radio range in a open country with a proper dimension is a subject of future work.

Even for a small grid of 3x3 nodes it was impossible to find a field with the required dimension in the surrounding area of the university. As the research with more nodes is planned for the future which would need even more space we abandoned these type of experimental setup.

4.2 Location

Following events might affect a running experiment:

- moving cars, people, and animals are able to change general link quality
- buildings with running WLAN networks can cause packet collisions
- outside influences like humidity or atmospheric pressure can change sending and reception characteristics

As these events are hard to control or even uncontrollable, we decided to perform the experiments in a realistic surrounding. We have chosen an in use university parking area. Cars, trees and bushes offer a radio shadow so that it is possible to place nodes in a comparatively small area without reaching all nodes directly.

4.3 Experiment one

This experiment was carried out to verify the operability of our test system outside the laboratory and to collect a first amount of data. It consisted of seven devices layed out in an area of about 165m x 90m.

4.3.1 Configuration

Experimental parameters

We used a packet length of 800 bytes and a sending frequency of 50 packets per second. That results in a data rate of about 40 kilobytes per second. We sent out 10000 flooding packets and 500 measurement packets per cycle. This setup was repeated five times. All packets flooded within our network were originally sent out by the device called *Homer*

Node placement

Nodes were placed in two rows consisting of 3 respectively 4 devices as shown in Figure 4.1 and Table 4.1. The aspect ratio between latitude and longitude used in Figure 4.1 amounts to 1.6:1 because of the following calculation:

Average earth radius [6]:

$$r_{earth} = 6371.3km$$

Earth circumference:

$$c_{earth} = 2 \cdot pi \cdot r_{earth} \approx 40032km$$

Distance between two lines of latitude (360 all together):

$$d_{latitude} = \frac{c_{earth}}{360} = \frac{40032km}{360} \approx 111.2km$$

Distance between two lines of longitude (at a latitude of 51.187°):

$$d_{longitude(at51.187^\circ)} = \frac{cos(51.187^\circ) \cdot r_{earth} \cdot 2pi}{360} \approx 69.7km$$

Ratio between latitude and longitude:

$$\frac{d_{latitude}}{d_{longitude(at51.187^\circ)}} = \frac{111.2km}{69.7km} \approx 1.6$$

This ratio is used for every Figure with plotted GPS coordinates.

	latitude	longitude	distance to *Homer*
Homer	51.187900	6.795250	0 m
Marge	51.186983	6.795633	105.33 m
Lisa	51.186433	6.795617	165.01 m
Bart	51.186450	6.794867	163.32 m
Maggie	51.186817	6.794733	125.61 m
Apu	51.187300	6.794467	86.13 m
Burns	51.187500	6.794550	65.97 m

Table 4.1: GPS coordinates of experiment one, distances to *Homer*

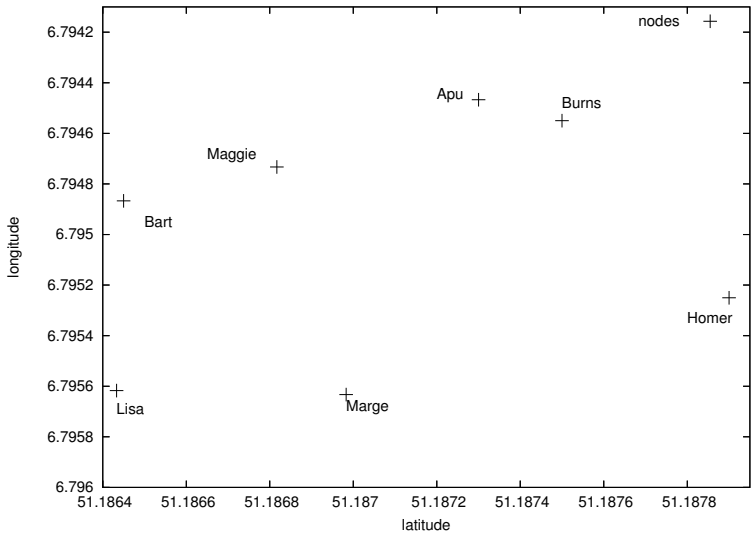


Figure 4.1: Node placement for experiment one, the size of the area was about 165 m x 90 m

4.3.2 Analysis

Reception rate

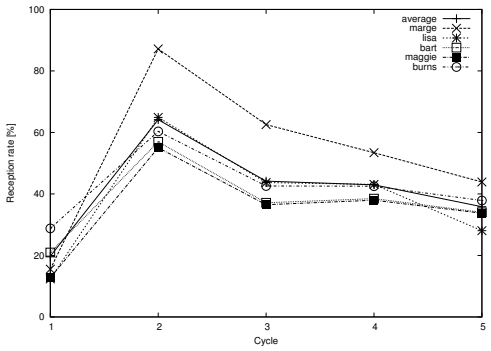


Figure 4.2: Reception rate

The reception rate of a node is the percentage of the original packets sent out by *Homer* received by the node. As shown in Figure 4.2 the average rate of packet reception per node was relative small over the whole network. In average less than 50 % of the original flooded packets were received by the nodes. Figure 4.3 presents an explanation of this behaviour:

- only three nodes (*Marge*, *Apu* and *Burns*) were able to receive packets directly from the original source *Homer*
- most links between the nodes had loss rates of more than 50%
- no node received packets in a relevant amount from more than 3 other nodes, most of them only from one or two nodes

All these points together explain that high packet loss is measured from the first hop on and that these packet losses were not compensated by receiving missed packets through other routes.

Therefore we decided to change the way of node placement for the next experiments. To obtain better rates of packet reception throughout the network we needed more links with lower loss rates between the nodes. So nodes had to be placed in a more dense topology.

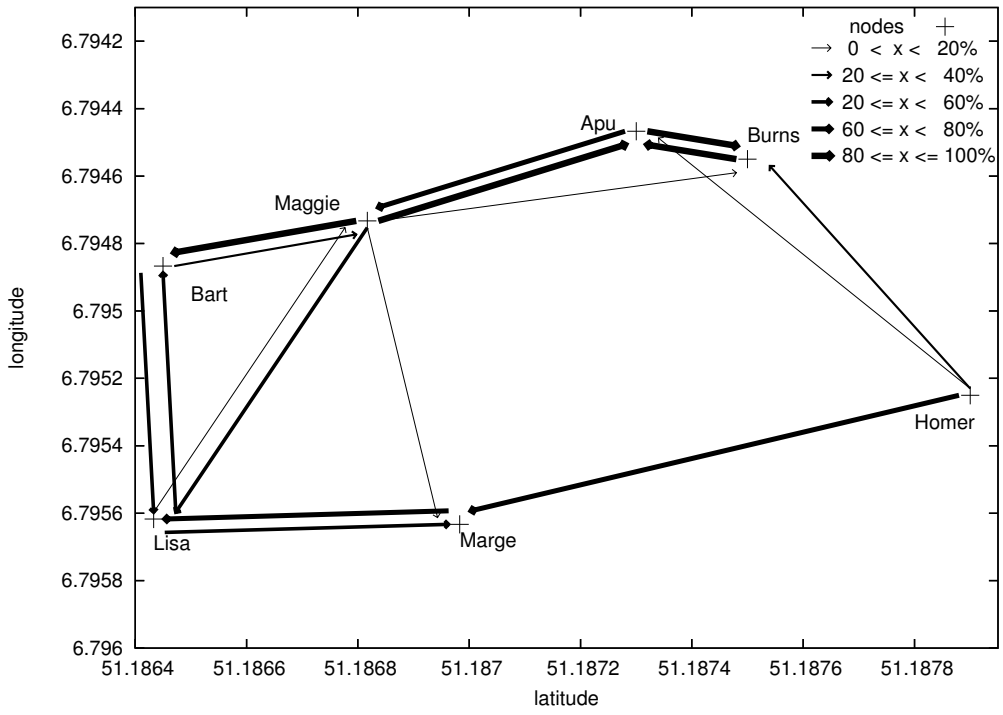


Figure 4.3: Link quality, cycle four

4.4 Experiment two

The second experiment consisted of 10 nodes. It was performed at the same experimental site as experiment one (Section 4.3).

4.4.1 Configuration

Experimental parameters

For this experiment we choose packet lengths of 60 and 200 bytes. For each length we performed three cycles with two packets per second and three cycles with 20 packets per second. As the batteries of our iPAQs only last about two hours and device placement takes a while we only had about 6 minutes for each cycle, which limits the amount of packets sent out. At each cycle with

- 2 packets per second, 250 flooding and 20 measurement packets were send.
- 20 packets per second, 2500 flooding and 200 measurement packets were sent.

The smallest possible size of an IP packet is 34 bytes. It is a packet with a 14 bytes ethernet header, a 20 bytes IP header and no payload. We chose a size of 60 bytes because this size roughly complies with the size of a small packet containing route searching information in *DSR* (14 bytes ethernet header, 20 bytes IP header, 4 bytes DSR header, 4 bytes DSR route request header, 4 bytes per hop). To evaluate the influence of the packet size, we chose a second size of 200 bytes.

The resulting data rates of the flooded packets can be found in Table 4.2. They are very low while using these small packets and sending rates of only 2 and 20 packets per seconds[1]. But in general flooding is used to broadcast status information or route requests/announcements and in our opinion in most cases not suitable to transfer huge amounts of data, because of its huge demand of bandwidth throughout the whole network. Therefore this seems to be reasonable data rates for flooding.

[1] about 0,1 to 4 kb/s compared with a maximum transfer rate between two nodes of about 475kb/s tested with iperf[3] and also with our click packetgenerator configuration (Section 3.4.2)

Node placement

In the first experiment (Section 4.3) we found out that bad links especially at the first hop after the sender, cause high loss of packets within the whole network. Therefore we placed devices differently way for the second experiment. To create more routes with lower loss rates throughout the network from the starting hops on we placed more nodes in radio range of the original sender and ensured that the starting links had a very good link quality. Figure 4.4 and Table 4.3 present the node placement.

At first we performed three cycles of our *exp-200/2* (Table 4.2) experiment. The following three cycles were run with the setup of the *exp-200/20* experiment. Device *Guest2* failed during these cycles because of an empty battery. This device had a central position within our network (Figure 4.5). So some good working links were broken and thus the network topology changed fundamentally. Consequently the results of cycles with running node *Guest2* and cycles without *Guest2* can not be compared without caution.

4.4.2 Analysis

Asymmetric links

The link quality between two nodes can be different for each direction. This is shown in Figure 4.5(a). The links between *Moe* and *Apu* are a good example. *Apu* received 80% of the packets broadcasted by *Moe*, but *Moe* received not a single packet by *Apu*.

A possible explanation might be that every node has different direct surroundings and therefore different sources of interference.

Changing link quality

The link quality between two nodes can vary to a huge extent over a period of time. Figure 4.5(a) in comparison with Figure 4.5(b) presents this behavior. For example the link from *Apu* to *Maggie* was nearly perfect at cycle one, but less than 20% of the packets were received by *Maggie* at cycle three. About one third of the links in our experiments changed more than 30 percent during measurments. The

frequency	packet size	data rate	label
2/s	60 byte	0.117 kb/s	exp-60/2
2/s	200 byte	0.391 kb/s	exp-200/2
20/s	60 byte	1.172 kb/s	exp-60/20
20/s	200 byte	3.906 kb/s	exp-200/20

Table 4.2: Data rates experiment two

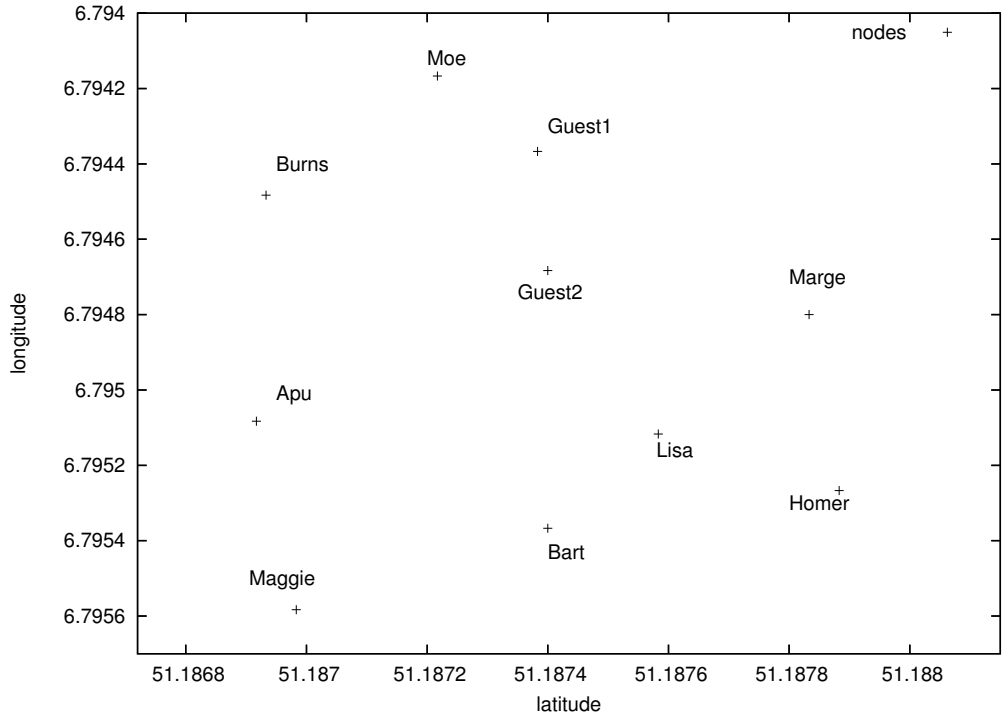
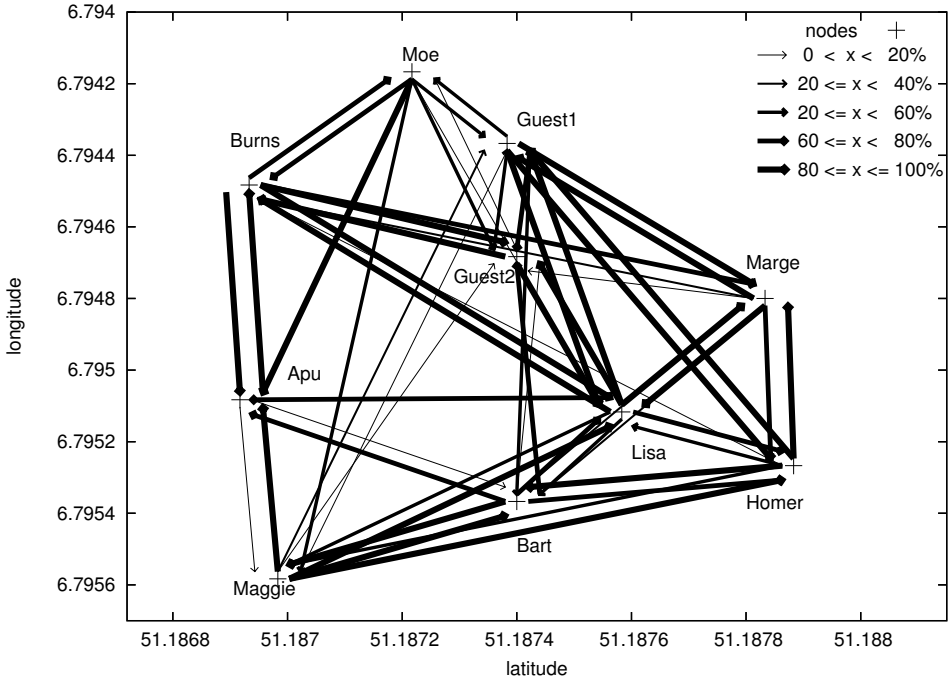
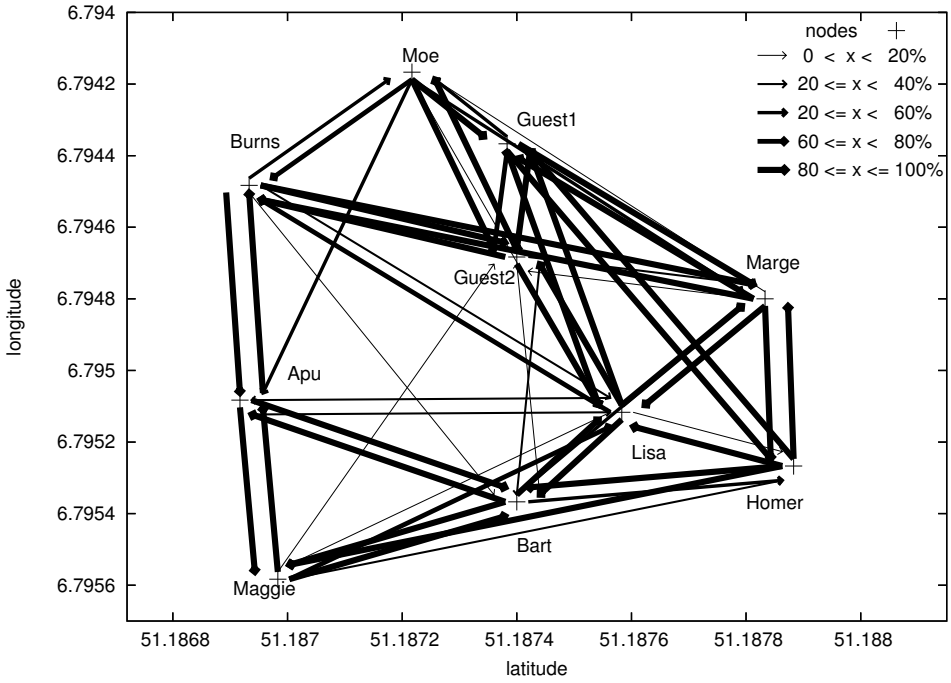


Figure 4.4: Node placement for experiment two, the size of the area was about 108 m x 86 m



(a) *exp-200/2* cycle three



(b) *exp-200/2* cycle one

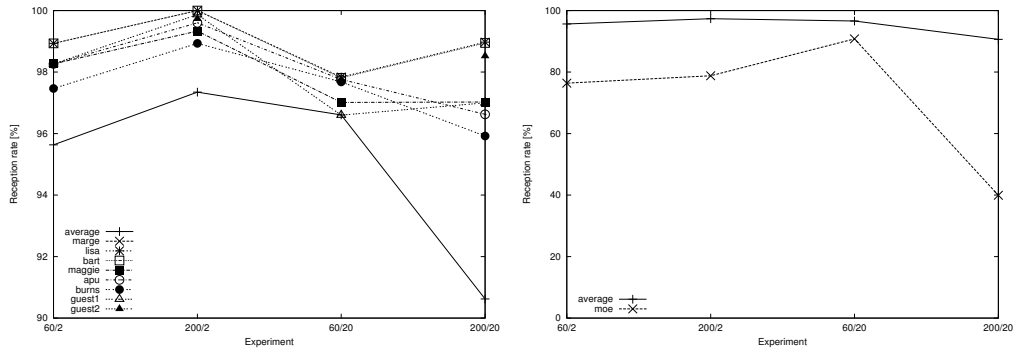
Figure 4.5: Link Quality

	latitude	longitude	distance to *Homer*
Homer	51.187883	6.795267	0 m
Marge	51.187833	6.794800	33.0 m
Lisa	51.187583	6.795117	34.93 m
Bart	51.187400	6.795367	54.12 m
Maggie	51.186983	6.795583	102.4 m
Apu	51.186917	6.795083	108.1 m
Burns	51.186933	6.794483	118.85 m
Moe	51.187217	6.794167	106.52 m
Guest1	51.187383	6.794367	83.76 m
Guest2	51.187400	6.794683	67.34 m

Table 4.3: GPS coordinates of experiment two, distances to *homer*

changing link quality can also be explained by the changing surroundings such as cars passing by.

Reception rate



(a) (b)

Figure 4.6: Percentage of received packets

Figure 4.6(b) shows the average percentage of packet reception over all nodes and all three cycles per data rate. It can be seen that the reception rate is higher than 90% for all data rates.

As shown in Figure 4.6(a) most nodes feature a reception rate between 95% and 100%. Only *Moe* has a significant smaller value (Figure 4.6(b)). This can be explained with Figure 4.5. In contrast to the other nodes *Moe* only has links to a small number of nodes, which also have a relatively bad quality.

Hop analysis

Figure 4.7 shows the ratio of the number of hops packets needed to reach each node. As expected nodes farther away from the original source *Homer* present higher hop counts.

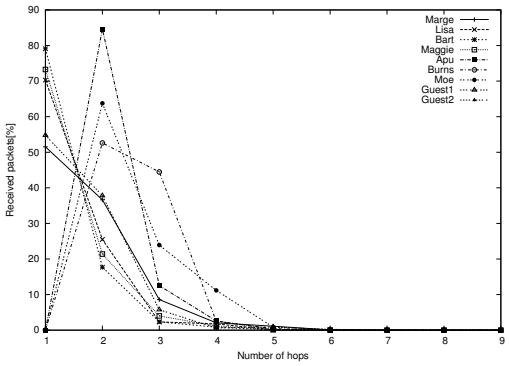
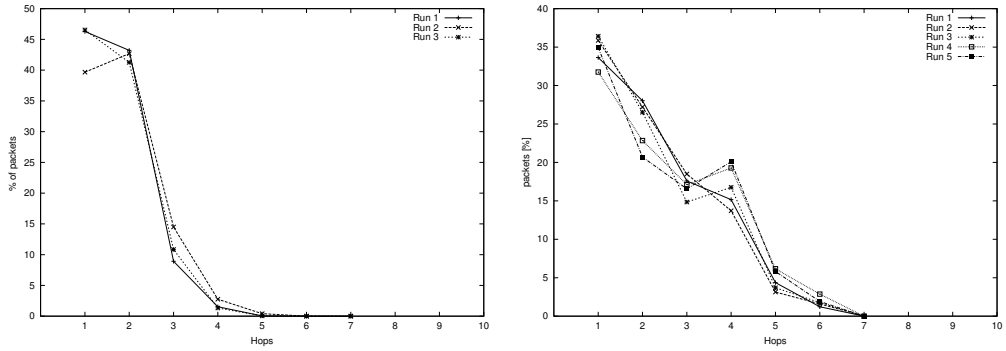


Figure 4.7: Number of hops without duplicates (*exp-200/20*, cycle three)



(a) *exp-200/2* (b) *exp-800/5*

Figure 4.8: Packets minimum hop count including duplicates

Figure 4.5 and 4.3 show that in theory every node can be reached from *Homer* over at most two hops and three respectively. Figure 4.8 shows the minimum hop count for each packet and each node. It can be seen that some packets used significantly longer routes. The maximum number of hops a packet used in our experiments with nine nodes was seven and in experiment exp-800/50 it was six. This can be problematic if flooding should be limited as it is done in the expanding ring search technique used as an optimization for AODV [18]. If for example flooding was limited to three hops in our experiment exp-800/50 (Figure 4.8(b)) up to 30% of the packets received without limited flooding could have been lost. The influence of less traffic and therefore less collisions throughout the network could slightly counteract the higher loss rates. To analyse this behaviour in networks with more nodes is a project of future work.

Packet Loss

A noticeable effect in our setup was that *Lisa* is a good indicator for packet reception. In more than 90% of transfers no other node has received a packet missed by Lisa. And in the majority of all other cases only very few nodes received it.

As a result of the fact that *Lisa* presents good reception statistics to most other nodes (Figure 4.5), it was very unlikely that *Lisa* missed a packet received by most of the other nodes.

Long links

The authors of [12] define a *Long link* as follows: "Link that is significantly longer than expected".
An example is shown in Figure 4.9 (link between *Homer* and *Burns*). This points out, that the radio range of a node is highly variable.

Backward links

The authors of [12] define a *Backward link* as follows: "Link in which the recipient of the flood is closer from the original source than the transmitter".
An example of this behaviour is shown in Figure 4.10(b) between the nodes *Bart* and *lisa*. An explanation for this behaviour is that a node reached by a *Backward*

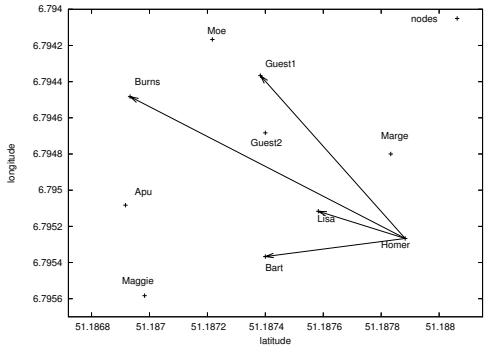
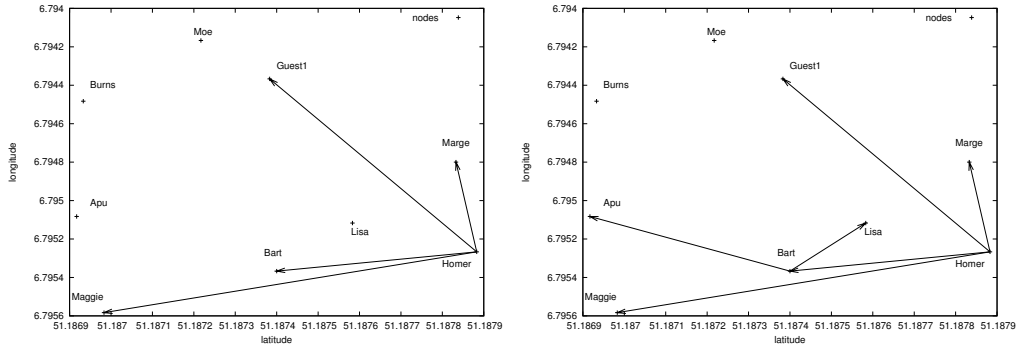
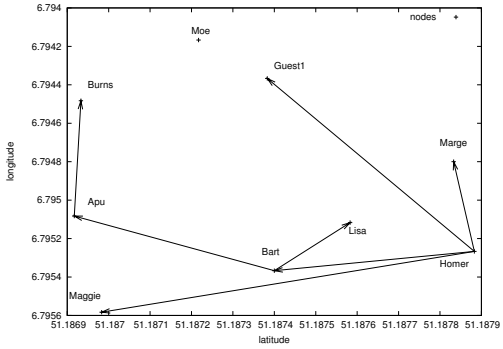


Figure 4.9: Long link, *exp-60/20* cycle three, sequence number 96, hop one



(a) first hop

(b) second hop



(c) third (final) hop

Figure 4.10: Propagation of a single packet, experiment *exp-60/2*, cycle one

link has missed the packet from nodes geographically closer to the original source because of collisions or packet loss but nodes further away received the packet. These nodes rebroadcast it and their broadcast can be received by the node missed before by a backward link.

Straggler

The authors of [12] define a *Straggler* as follows: "Node that misses a transmission, even though it would be expected to receive a packet with high probability". Figure 4.10(c) presents an example for a *Straggler* in our experiments. The node *Moe* missed the flooded packet in contrast to his direct neighbours *Burns* and *Guest1*.

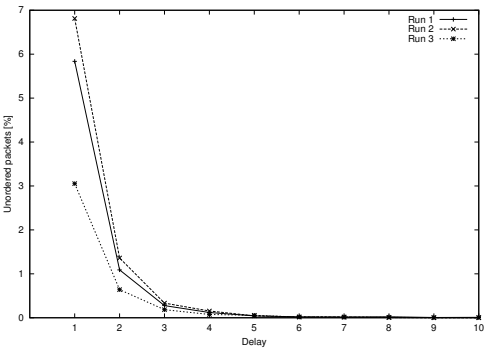
Packet order

We define a packet as unordered if a packet with sequence number n is received after one or more packets with a sequence number $n+x$ for $x \geq 1$. The unorder level of a packet is defined as the maximum of x. To analyse the packet order we examined all incoming packets including duplicates.

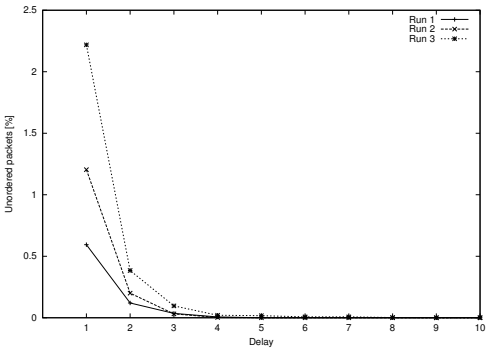
At the experiments with a sending frequency of two packets per second we discovered that all packets were received in order.

For a better comparability we present the packet order results of experiment one (Section 4.3) together with the results of the experiments with 20 packets per second. As shown in Figure 4.12 up to 14% of the packets were unordered per node. The experiment with the highest unorder levels was experiment *exp-60/20* 4.12(a). The experiment with packet size 200 bytes (4.12(b))had a significant lower rate of unorder (Run one of *exp-200/20* experiment must not be compared with the other runs or data rates, because as mentioned in Section 4.4.1 node *Guest2* failed after run one and therefore the results are not comparable). A noticeable effect is that the graphs of individual nodes have a similar shape and do hardly overlap, but the variance between the runs is high. It seems to be that a unorder near the original source leads to an unorder at distant nodes.

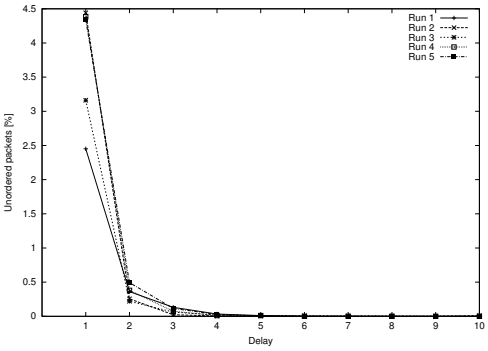
The unorder level of our experiment can be found in Figure 4.11. It presents the delay of all incoming packets at all nodes of the network per run and data



(a) exp-60/20

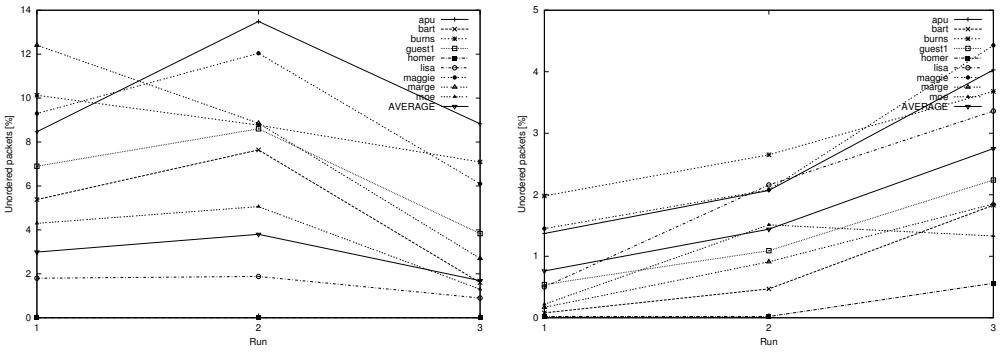


(b) exp-200/20



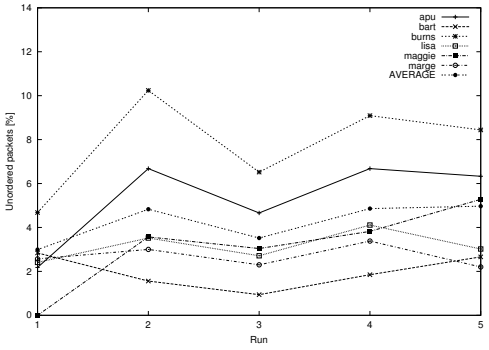
(c) exp-800/50, experiment one (Section 4.3)

Figure 4.11: Level of unorder



(a) exp-60/20

(b) exp-200/20



(c) exp-800/50, experiment one (Section 4.3)

Figure 4.12: Percentage of unordered packets

rate. Most delayed packets are only one packet delayed, but a few feature a high unorder level. At experiment *exp-200/20* (Figure 4.11(b)) we examined a maximum unorder level of eight, which equals about 0.4 seconds. At a packet size of 60 bytes (Figure 4.11(a)), the maximum unorder level was 18 (about 0.9 seconds) and in experiment exp-800/50 (Figure 4.11(c)) we discovered a unorder level of 56, which equals about 1.1 second.

4.5 Required manpower

During our experiments we discovered that a great deal of manpower is required. As an example, we present the time required for experiment two (Section 4.4):

- briefing of helpers, about 30 minutes
- node placement, about 30 min
- 12 cycles of flooding/measurement, each about 6 min (all together about 72 min)
  - flooding, about 2 minutes (24 minutes)
  - measurement, about 4 minutes (48 minutes)
- measurement of nodes' GPS coordinates, node removement, debriefing, about 20 min

All together this makes 150 minutes that are needed to for only 24 minutes of flooding. This is a ratio of 6.25:1

As we performed our experiments on a public area, every node needed to be supervised by a helper. According to our experience it was necessary to have one additional technical instructed helper for roughly every then nodes to be able to react on device problems. Therefore 1.1 x helpers are needed in a network with n devices. For our experiment with ten devices this meant the following calculation of flooding output per man-hour:

As mentioned above eleven helpers are needed for ten nodes . Each of them is required for 150 minutes. All together there were $11 \cdot 150 = 1650$ man-minutes

= 27.5 man-hours needed for only 24 minutes of flooding. This means that one minute flooding required 68.75 minutes working time.

In future experiments with more nodes the expenditure of time for link quality measurement and node placement will increase proportional to node quantity and therefore the ratio will be even more disadvantageous (e.g. for a 20 node network the time required for measurement would be about 8 minutes per cycle and the time needed for node placement about one hour. GPS coordinate measurement and node removement will also increase a little).

To avoid this problem and to be independent from battery run-time we tried to place the nodes in several offices at the university. We encountered the following difficulties:

- access to a lot offices is needed to avoid the problem of node supervising
- because of a limited amount of accessible offices and bad radio reception characteristics in buildings it is very difficult to place nodes
- nodes placed in used offices were moved without our knowledge (even a movement of only a few centimetres can lead to a useless link in this environment), so some necessary links were down
- very time-consuming troubleshooting because we had no direct access to some offices

We were not able to place nodes as described in Section 3.2 because most links between the nodes were not functional. Therefore it was not possible to capture usable data and we abandoned the plan to place the devices in the university building.

4.6 Reproducibility

We assume that our results are not exactly reproducible, although we measured the exact GPS coordinates of every participating node. This is first of all a result of the changing topology on the used parking area. Moving cars can largely affect link statistics, as it is shown in Section 4.4.2.

Chapter 5

Conclusion

Our experiments prove the results of [12] that even a simple protocol like flooding has a very complex behaviour. We encountered backward links, long links, asymmetric links, changing link quality, delayed packets and Stragglers. As a result of our experiments flooding seems to be working with a high success rate if nodes are placed in a way that each node is in good radio range to a couple of other nodes (Section 4.4) and also the used data rates are very small. If nodes are in radio range of only a small number of other nodes and in addition the links between these nodes have a bad quality, flooding has only a very low success rate as presented in Section 4.3.

For future experiments with even more nodes the required man power will be a problem. As we pointed out in Section 4.5 there is a need for about 1.1 helpers per participating device. Also the ratio between invested man hours and actual flooding time will be even worse in networks with more nodes than our ratio of 68.75:1

Bibliography

[1] *Familiar Linux*. http://www.handhelds.org.

[2] *gnuplot: a command-line driven interactive datafile and function plotting utility*. http://www.gnuplot.info.

[3] *Iperf: a tool to measure maximum bandwidth*. http://dast.nlanr.net/Projects/Iperf.

[4] *Perl*. http://www.perl.com.

[5] *tcpdump: a tool for network monitoring, protocol debugging and data acquisition*. http://www.tcpdump.org.

[6] Wikipedia: Earth radius. http://en.wikipedia.org/wiki/Earth_radius.

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

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

[9] Gerald Combs. *Ethereal: The world's most popular network protocol analyzer*. http://www.ethereal.com.

[10] Gerald Combs. *Tethereal: network protocol analyzer*. http://www.ethereal.com/docs/man-pages/tethereal.1.html.

[11] S. Desilva and S. Das. Experimental evaluation of a wireless ad hoc network. In *Proceedings of the 9th Int. Conf. on Computer Communications and Networks (IC3N)*, Las Vegas, October 2000.

[12] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report UCLACSD-TR 02-0013, Computer Science Department, UCLA, 2002.

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

[14] 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.

[15] Eddie Kohler, Robert Morris, Benjie Chen, and John Jannotti. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[16] D. Maltz, J. Broch, and D. Johnson. Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed. In *Proceedings of WCNC 2000*, September 2000.

[17] David A. Maltz, Josh Broch, and David B. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, 1999.

[18] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF RFC 3561 (Experimental), July 2003.

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

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmitteln benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, den 14.Dezember 2004 Andreas Tarp