Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Prof. Dr. Roger Wattenhofer
Office ETZ G63
Gloriastrasse 35
ETH Zurich
CH-8092 Zurich
Switzerland

Semester Thesis: Simulating Media Access Control Protocols in Wireless Ad-Hoc Networks

by

Frederic Dreher

Supervisor: Olga Goussevskaia (golga@tik.ee.ethz.ch)

Contents

Chapter 1

Introduction

The first MAC protocol designed for wireless networks was Aloha. As opposed to prior point-to- point computer communications, Aloha used a shared medium for transmission. Because of its very simple mechanism, where each node just sends data with probability p, Aloha is widely studied and serves as a basis for many of the MAC protocols being deployed nowadays in wired and wireless networks, such as Ethernet and IEEE 802.11. Keeping the random access spirit of Aloha, a variety of MAC layer protocols have been proposed for wireless ad-hoc networks. Many of these protocols use some variation of CSMA (Carrier Sense Multiple Access), which improves the utilization of the channel by allowing nodes to first listen to the medium and then decide whether to transmit or not. The main challenge of all these protocols is to deal with the hidden terminal problem, which makes it tricky to take preventive measures against collisions, since two sending nodes may not be within mutual reach of each other and therefore not be able to sense that the other is transmitting to the same receiver, for example.

1.1 Interference Model

In this thesis we aim to analyse, through simulations, the performances of MAC protocols for wireless ad-hoc networks in the SINR (Signal-to-Interference-plus-Noise-Ratio) interference model are compared. In the SINR model, the energy of a signal fades with the distance to the power of the path-loss exponent α. If the signal strength received by a device divided by the interfering strength of competitor transmitters is above some threshold β, the receiver can decode the message, otherwise it cannot.

$$\frac{\frac{P_x}{d_{xy}^{\alpha}}}{N + I_y} \geq \beta \tag{1.1}$$

,
Where P_x is the power level of the transmission, d_{xy} is the distance between nodes x and y, $\alpha > 2$ is the path loss exponent, which depends on external conditions of the medium, $\beta \geq 1$ denotes the minimum signal to interference ratio requires for a message to be successfully received, N is the ambient noise, and I_y is the total amount of interference experienced by ther receiver y. This interference, which is caused by all simultaneously transmitting nodes in the network is defined as follows:

$$I_y = \sum_{v \epsilon \backslash Vx} \frac{P_v}{d_{vy}^{\alpha}} \tag{1.2}$$

We assume here that all nodes transmit with the same power level.

1.2 Simulation Framework

We use a network simulator developed by the Distributed Computing Group, called Sinalgo [11]. Sinalgo is a simulation framework for testing and validating network algorithms. Unlike most other network simulators, which spend most time simulating the different layers of the network stack, Sinalgo focuses on the verification of network algorithms, and abstracts from the underlying layers: It offers a message passing view of the network, which captures well the view of actual network devices. Sinalgo was designed, but is not limited to simulate wireless networks.

1.3 Extension to Framework

We developed a tool, called simtest, in order to get reliable results and to automate the execution of many simulation calls and produce graphical representation. This extension runs several simulations in batch mode and generates graphics from its output. It provides an easy interface to specify input variables and output parameters to generate the graphics using gnuplot [12]. The program runs several simulations (according to specification) and generates the graphics from the mean vales to reduce the variance. Furthermore it calculates and plots variances for all outputparameters and stores the rawdata separately to review the Sinalgo simulation output. With some configuration simtest makes it possible to analyse or compare one or more protocols with a single command.

1.4 MAC Protocols

The Media Access Control (MAC) data communication protocol sub-layer is a part of the data link layer specified in the seven-layer OSI model . It provides channel access control mechanisms that makes it possible for several terminals or network nodes to communicate within a network. The MAC protocol handles the multiuser medium access. Since in a radio network the air is the transmission medium, a MAC protocol for wireless networks determines its access rules for transmission.

1.5 Thesis Overview

This thesis is organized as follows. In Chapter 2 the Sinalgo simulator and the simtest program are introduced. In Chapter 3 we describe the protocols analysed in Chapter 4 and their implementation in Sinalgo. In Chapter 4 we present all simulation results, and in Chapter 5 we examine the results and analyse the protocols.

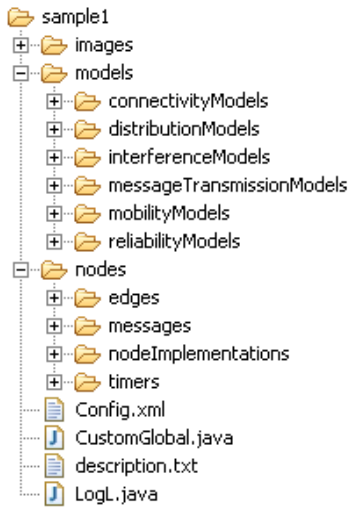
Chapter 2

Simulation Framework

In this chapter, the simulation framework and its extension are described.

2.1 Sinalgo

Sinalgo is a Java based wireless network protocol simulator. The synchronous time-slot environment is chosen to set the protocols up. These so called rounds are comparable to a global timer in a network set ub by a base station or the nodes.



Sinalgo Project Structure

The Sinalgo simulator provides many dummy templates in the default project. To set up your own project you can implement, as depicted besides, the models, such as interference, distribution, connectivity, mobility, reliability, and the message transmission model, and the implementation of the edges, messages, nodes, and timers to customize a project. Furthermore each project has its own configuration and description file. The CustomGlobal class holds global variables for the project and is an appropriate location to log the statistics for simtest.

The Sinalgo default project and base files provide tools for logging, a random number and distributions generator, a print tool to generate EPS or PDF files from the gui, and a statistics class to sample data.

Key to a protocol implementation is the node and the message implementation. Messages usually can be kept simple, header and data, but the node implementation determines the global behaviour of the protocol.

The default models used here are:

- Connectivity Model: Unit Disk Graph (UDG) connectivity which is a purely geometric connectivity model: Two nodes are within communication range if their mutual distance is below a given threshold, the maximal transmission radius. By modifying this radius the number of neighbours to each node (sending or transmitting unit) is influenced.
- Interference Model: Signal to Noise Interference Ratio (SINR)
- Distribution Model: Random Distribution Model, the nodes are placed randomly on the simulation area

- Static, no mobility model is used

2.2 *simtest*

This extension to the Sinalgo simulator framework runs several simulation calls and gathers information about these by running one scenario more times and modifying one or more parameters sequentially. From these results it can then generate statistics and graphics. The parameters and sequences have to be specified in an XML input file. It contains information about the paths to execute Sinalgo, log directory, the command to execute gnuplot, and so on, parameters to call Sinalgo, one running variable parameter to which the graphical output will refer to, and the labels to find the information in the log file of the simulator. For more information about the use, refer to Appendix A.1. In this section a brief description about the usage is given, the mode of operation and the usefulness of the program is showen.

2.2.1 Use of *simtest*

Basically, the use of this program is to test protocols fully automatically. With the configuration of Sinalgo according to simtest specification, one or more projects can be tested for all kinds of scenarios. You can run several instances of the same Sinalgo configuration to reduce the variance and proceed through several Sinalgo configurations without gathering the information about the single results by hand e.g., you can run simulations changing the number of nodes for two projects going through a sequence of increasing message length, running each configuration 50 times to compare two algorithms. This time consuming process can be launched with a single command and it produces graphical output and tables with the mean values and variances. As a result, the process which is extensive by hand can simply be run overnight.

2.2.2 Mode of Operation

The program proceeds in following five steps:

- Getting input from XML file
- Computing Sinalgo Arguments
- Executing the commands one by one to
- After each execution gather the specified parameters and write them to data files
- Draw graphics from the data files created

Table 2.1 lists the class and their basic functionality.

2.2.3 Inputparameters

The simtest input file must reside in the simtest root folder and be named `simtestParameter.xml`. Table 2.2 shows the values which have to be specified.

class	brief description
Main	base class calling all methods, warning and error handling
Configuration	class holding global variables, such as total number of simulations, number of simulation calls per configuration, etc., and the interface for other methods to set the input parameters and the mentioned variables
Parameters	the base class for parsing through the input file and Sinalgo log file, it distributes these values to other classes and contains the interface to draw the graphics
Executor	a class that executes Java system calls i.e., executes the Sinalgo simulations
Output	creates, copies, or recreates data, log, gnuplot script files, writes to the data files
Outputpair	holds the information about the label pairs, which will generate one graphic each and are used to parse the Sinalgo log file
DataSeries	a statistics class to compute mean values and variances

Table 2.1: Class list and a brief description of each

inputparameters	
paths	execution path, path to Sinalgo log file, simtest log path, raw data log path
gnuplot command	the gnuplot command line command may be different for different systems
OS	operating system, Windows or Linux
file modification tag	this tag determines if the data output files in the log path are modified rerunning all simulations or just the graphics are drawn from it, useful for automatic plot, if you changed the data (erased or added a data block) or want to create another outputformat
output format	can be either PNG or EPS, PNG is colored, EPS is more readable included in documents
No of simulation calls per configuration	the amount of sinalgo calls to reduce variance
Sinalgo parameters	node name (containing project name) No of nodes to create, No of rounds to run before termination, models (distribution, interference, connectivity), and the rest has to be introduced as overwrite statement
runnning parameter	also a Sinalgo parameter, but due to this sequence the graphical output is adjusted
parameter analysis parameters	plotpairs with an X tag and a Y tag to retrieve information from the Sinalgo log file and to determine what graphics to draw

Table 2.2: *simtest* Input Parameters

2.2.4 Sinalgo Adaption

To achieve a full automation, simtest has to be able to process the Sinalgo log file. It has to be in XML format and look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document>
<simtest>
<!--*****************SIMTESTDATA*********************-->
<UDGrMax value="300"/>
<!--**************SIMTESTDATA************************-->
</simtest>
</Document>
```

and must be named according to the input path to log file, given in the simtest input file. Disable the Sinalgo log to time folder option in the configuration file so that simtest finds the log file after each simulation in the same place.

The UDGrMax-tag is an example tag to show the format of an entry. Specify this tag (`UDGrMax`) in the simtest input file as a tag for plotting and simtest will find it in the Sinalgo log file and add an entry in the output file. Template functions to create this file structure are within the simtest package.

Chapter 3

Protocols

The protocols studied in this thesis are presented in this chapter. Their implementation in the simulation framework can not always be like reality. Hence, there are some restrictions in implementation. The protocols are described in general and then the implementation is discussed in another subsection.

3.1 Aloha

Aloha was the first MAC protocol designed for wireless networks. As opposed to prior point-to-point computer communications, Aloha uses a shared medium for transmission.

3.1.1 Concept

The first version of the protocol was basic:

- If there is data to send, send the data
- If the message collides with another transmission, wait random time and try resending

The critical aspect is: what is random? The quality of the backoff scheme chosen significantly influences the efficiency of the protocol and the predictability of its behavior. Pure Aloha had a maximum throughput of about 18.4%.

An improvement to the original Aloha protocol was Slotted Aloha, which introduced discrete timeslots and increased the maximum throughput to 36.8%. A station can send only at the beginning of a timeslot, and thus collisions are reduced. It uses a DATA-ACK Sequence for transferring data.

3.1.2 Implementation

The Slotted Aloha protocol fits to the concept of rounds in the Sinalgo simulator and was therefore implemented. The random waiting time concept is emulated by using a sending probability with which each node will send its data. This sending probability is the most influencing factor for the Aloha performance. In our simulations we decided to set our sending probability as follows:

$$P_{send} = \frac{1}{N_{noOfNeighbors} * k} \tag{3.1}$$

Where k is the sending probability coefficient, P_{send} is the probability a node sends a message in this round, $N_{avgNoOfNeighbors}$ is the number of neighbours a node has. This way the sending probability is adjusted according to the density of the network.

3.2 MACAW

MACAW was first introduced in 1994 [2] as an improvement to Multiple Access with Collision Avoidance (MACA) protocol and alternative to CSMA based algorithms. In MACAW the RTS-CTS handshake was incorporated for the first time.

3.2.1 Concept

In MACAW each node has a Network Allocation Vector (NAV) which makes the node back off. MACAW uses RTS-CTS-DS-DATA-ACK frame sequence for transferring data to solve the hidden terminal problem, where two senders are out of range and may cause a collision at another node which is within the range of both. If a node A wants to send a message to node B it sets a random timer (implemented as sending probability) and if no overhearing makes A back off it broadcasts an RTS (Ready To Send) message containing B's ID and the data message length. Upon reception of a RTS, B broadcasts immediately a CTS (Clear To Send) message containing also the data message length. This exchange done, A broadcasts a DS (Data Send) message containing the data message length and starts sending the data. When the data transmission is complete, B acknowledges this with an ACK message. If an ACK message gets lost and B receives an RTS for the same packet it acknowledged already, it sends directly the ACK message.

Every node overhearing an RTS sets its NAV and backs off for the amount of time a successful RTS-CTS exchange needs. It will then probably overhear a DS message and back off the time the data and the following ACK have to be sent. This is called virtual carrier sensing. A node overhearing a CTS backs off immediately the amount of time DS-DATA-ACK sequence would need.

3.2.2 Implementation

In the framework the minimum message length is one round, this is for RTS, CTS, DS and ACK, the data message can be longer. For instance a node backs off two rounds upon overhearing a RTS message, one to send back the CTS, another to overhear the DS. In every case of waiting (back off, wait for CTS, wait for ACK, ...) a timer is set regarding the estimated time the e.g., ACK arrives. In case a timer runs out the node goes back to contend state and sets a random timer. In two cases the timer is not random, when a node awaits a CTS and the timer runs out it increases the backoff by 1.5 and in case of successful transmission, upon reception of an ACK, it decreases by one. This backoff algorithm is called MILD back off and was introduced in [2] as an improvement to the binary exponential backoff (BEB), which we implemented also in MACAW to compare the backoff algorithms. In the base-station-cell model there is also a copy scheme, in case a node switches to another cell and adapts its backoff, but here this is not considered (no mobility).

3.3 IEEE 802.11 DCF

The Distributed Coordination Function (DCF) [5] is the IEEE 802.11 standard and so practically applied in many networks and by many users and was first introduced 1997. It is interesting to evaluate its performance in ad-hoc networks.

3.3.1 Concept

DCF uses CSMA/CA in a form of noise listening and a RTS-CTS-ACK sequence. Once a node senses the noise below a certain threshold and/or no carrier signal it sets a random timer and then sends its RTS if the channel is still considered free. A node receiving a RTS destined for itself sends immediately

a CTS and waits for data message. This data message is sent without delay by the requesting node upon the CTS reception. A successful transmission is notified by an ACK message. All messages contain the whole transmission time needed e.g., a RTS contains RTS-CTS-DATA-ACK time units, a CTS only DATA-ACK. Any message overhearing node backs off the time pending to end of transmission. Like the MACAW node it sets a NAV. In case a CTS is lost the nodes use Binary Exponential Backoff (BEB) (increase: double) which is reset to two on successful transmission (reception of an ACK).

The main difference to MACAW is the transmission sequence (without DS message) and the CSMA. MACAW does not sense the medium before sending. Furthermore the backoff algorithm differs from the original MACAW, which uses the MILD backoff algorithm instead of the BEB, and a node in DCF backs off directly the whole transmission length upon overhearing a RTS message, where a node in MACAW waits just the time until a DS message arrives and backs off the whole transmission length only upon a successful handshake.

3.3.2 Implementation

This protocol is ported in the same manner as MACAW to the Sinalgo framework considering the messages and the inbox handling. The messages contain more information about the time left for a successful transmission, and the DS message is not present.

The sensing algorithm is adapted to the SINR model, if the noise sensed in the node's area is below a certain threshold the medium is considered as free to transmit. The node runtime-sequence is put in a random order such that a node with a great ID is not always blocked by the low ID nodes' messages in the air, whose interference it would sense.

3.4 Location Enhancement to IEEE 802.11 DCF

Location Enhancement to DCF (LED) [3] is an enhancement to DCF and an interesting approach to increase the throughput of DCF by using a less pessimistic backoff algorithm. The main enhancement is the adding of a block containing location information about sender and receiver to every message , if possible.

3.4.1 Concept

This protocol (LED) is like DCF but has the location enhancement. Therefore a node sends its position and the noise sensed at sending time within each message. Now every RTS contains the senders position and the receiver can add its location information to the message to complete the block (GPS or another positioning method can be used for this). An overhearing node calculates its own influence to the nodes attempting to build up a connection using a propagation model. During the blocking decision making process, a non- receiver station (denoted as station i) of the frame calculates if its own transmissions will cause enough interference to interrupt the data delivery to which the just received frame belongs. The station needs to calculate the power level of its own transmission at both the source, denoted as P_i^s , and the destination, denoted as P_i^d , of the ongoing data delivery using an appropriate propagation model. [3] assumes a free space omni-directional propagation channel model [4]. This is a model in which many channels, especially outdoor channels, have been found to fit in practice. In this propagation model, the received signal power, P_r, is calculated as follows:

$$P_r = \left\{ \begin{array}{ll} \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 D^2 L} & D \leq D_{cross} \\ \frac{P_t G_t G_r h_t^2 h_r^2}{DL} & D > D_{cross} \end{array} \right\} \tag{3.2}$$

where P_t is the transmission power, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, D is the separation between the transmitter and the receiver, h_t is the transmitter elevation, h_r is the

receiver elevation, L is the system loss factor not related to propagation (≥ 1), λ is the wavelength in meters, and D_{cross} is calculated as $D_{cross} = (4\pi h_r h_t)/\lambda$. The first sub-model of this equation is called the Friis free-space propagation model and only used when the distance between the transmitter and the receiver is small. The second sub-model is called the two-ray ground reflection model and used when the distance is large. The station also needs to calculate the received power level of the destination station's transmission at the source, denoted as P_d^s , and that of the source transmission measured at the destination, P_d^s . If ($P_s^d > \beta P_i^s$) and ($P_s^d > \beta P_i^d$), the station should not block its own transmissions. Otherwise, it should block its transmissions. In the case that the communication parameters of either the source or the destination are unknown, the assessing station assumes the worst and blocks its own transmission. The noise at sending time may have changed at receiving time, but the transmission delay of a message is not so long that the actual noise taken in consideration when making the blocking decision differs significantly.

3.4.2 Implementation

The algorithm to evaluate a node's influence in the Sinalgo implementation we follow the SINR model. Instead of the Friis and Two-Ray model the node computes its influence over the distance following the equation shown below. We use constant transmit powers and therefore the power of a transmission depends only on the distance.

$$P_i^s = \frac{P}{d_{(i,s)}^\alpha} \tag{3.3}$$

$$P_i^d = \frac{P}{d_{(i,d)}^\alpha} \tag{3.4}$$

$$P_s^d = \frac{P}{d_{(s,d)}^\alpha} \tag{3.5}$$

Where $d_{(A,B)}$ is the distance from node A to node B and α is taken from the SINR interference model. The noise at sending time is denoted as N_{ST}^s at the connection source and N_{ST}^d at the receiving node in the connection.

The blocking condition is $P_s^d > \beta(P_i^s + N_{ST}^s)$ and $P_s^d > \beta(P_i^d + N_{ST}^d)$.

Chapter 4

Results

In this chapter we present the results, the chapter is devided into two sections, one treating a single protocol for itself, and another comparing the protocols for different densities, message lengths, and topologies.

To analyse the protocols and their behaviour we picked out five marking characteristics.

- DATAThroughput, in frames, the average amount of received data frames in 100 rounds per node
- DATAReceptionEfficiency, the index for data throughput, the data frame to data send attempts ratio
- CollisionIndex, messages which did not reach their destination node , including control messages (this is not the inverted DATAReceptionEfficiency, but closely related), in relation to the send attempts
- TimeUntilSuccessPerMessage, is the average time (in rounds) a node needed to transmit successfully a complete message
- FairnessIndex, according to Jain's fairness index:

$$F = \frac{(\sum_{i=1}^{N} \gamma_i)^2}{N \sum_{i=1}^{N} \gamma_i^2} \tag{4.1}$$

  Where N is the number of connections and γ_i is the number of received packets for connection i

The numbers which belong to the lines are accorded to the parameters in the title, starting from the left.

UDG/rMax	average number of neighbours
150	6
200	10
250	16
300	22
350	28
400	35

Table 4.1: Key to the density analysis

Simulation Area	1000 x 1000
Connectivity Model	UDG: $rMax = 200$
Interference Model	SINR: $\alpha = 3, \beta = 1$
Geometric Node Collection	$rMax = 400$
Message	Length = 5 (Aloha:1)
Distribution Model	Uniform Random Distribution
Number Of Nodes	100

These are the default values and are changed due to the aim of the analysis

Table 4.2: Default Simulation Setup

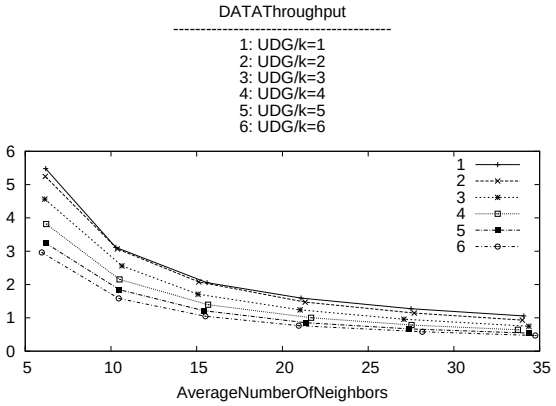


Figure 4.1: Aloha: Data Throughput

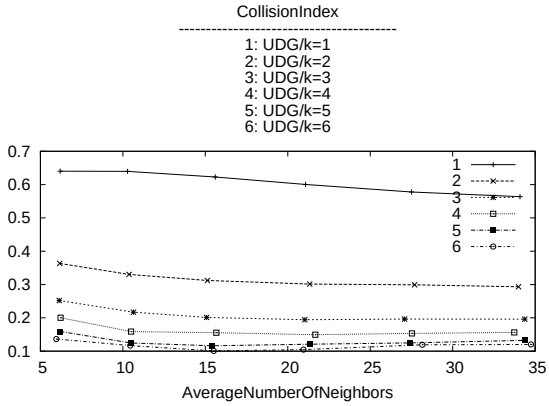


Figure 4.2: Aloha: Collision Index

4.1 Aloha Performance

The simulations for the Aloha Protocol examine the influence of the sending probability coefficient k. The graphics show the performance for different k's in dependence of the density given in average number of neighbors. It shows that the performance has a certain breakpoint.

Figure 4.1 shows that the data Throughput is decreasing with the increasing density and greater k. Line 1 for k equal 1 has the highest throughput.

The collision index in Figure 4.2 shows the collision ratio. The ration of messages not received by their destination node decreases with a greater k.

For k greater than one the fairness index is quite stable and remains in a range of 0.1 for every density as seen in Figure 4.3. For k equal to one the protocol is less fair.

The time until success for each message (Figure 4.4) is in the range of approx. 3 rounds from a k greater than 2, performance is less for smaller k.

Finally, the data reception efficiency increases the smaller k is, Figure 4.5.

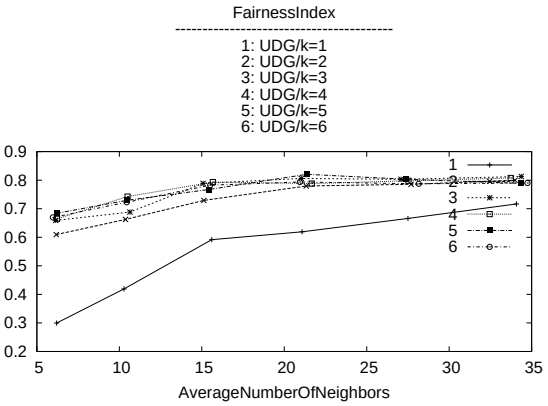


Figure 4.3: Aloha: Fairness Index

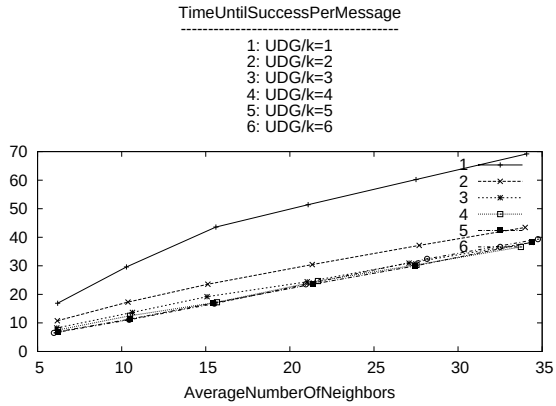


Figure 4.4: Aloha: Time Until Success Per Message

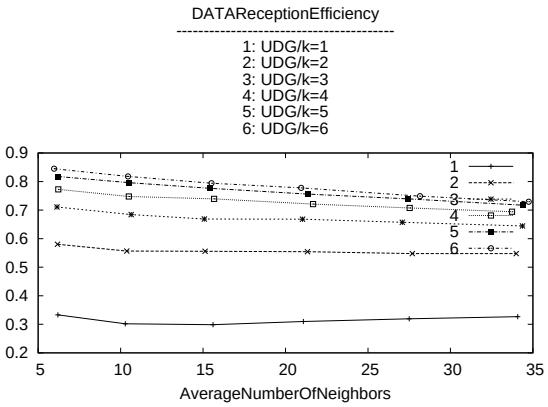


Figure 4.5: Aloha: Data Reception Efficiency

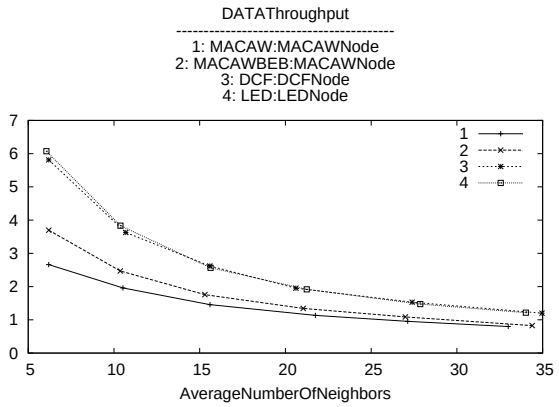


Figure 4.6: Density Analysis: DATA Throughput

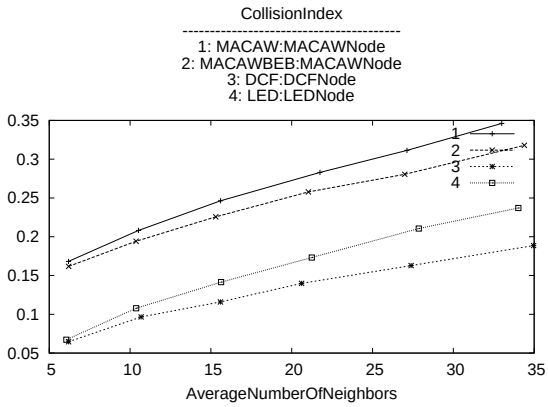


Figure 4.7: Density Analysis: Collision Index

4.2 Comparing Protocols

This section examines the protocols in comparison of to other. Hence, the focus is layed not only on the performance change but also on the influence of the parts a protocol consists of, such as the backoff algorithms, the transmission sequences, and the CSMA algorithm usage. All graphics show the results for two MACAW protocols (one designed with a MILD Backoff algorithm (standard), the other using BEB), DCF, and LED protocol. LED is DCF based and both use BEB and CSMA.

Therefore the impact of the backoff algorithm can be studied looking at the MACAW results. To see the effect of using CSMA, the DCF and the MACAW protocol behaviour will be examined.

The standard MACAW protocol, in further text called MACAW, the MACAW protocol using BEB instead of MILD backoff algorithm, denoted MACAWBEB, the DCF original protocol, denoted as DCF, and the resulting plot for the location enhancement to DCF protocol, short: LED, are analyzed.

4.2.1 Density Analysis

The density analysis is made using a message length of five frames, that makes the data five times longer as any control message. That means that there is a 20% overhead. The graphics are organized such that to the left with an average number of neighbors of 5 are the results for a less dense network. To the right edge the density increases up to 35 neighbors in average.

Figure 4.6 shows the throughput of data in frames per node and 100 rounds. DCF and LED achieve quite the same throughput, from 6 maximum to 1 frame in average. Below, MACAWBEB has 1 frame more throughput but converges towards MACAW for higher density.

The collision index in Figure 4.7 shows that a higher density brings more collisions to all. This performance is similar to Figure 4.6, MACAW shows worst performance and so on. LED produces more collisions than DCF, the throughput was the same.

Figure 4.8 shows the fairness index and here the different backoff algorithms separate the performance of the protocols. With a fairness index of average 0.5, which decreases only slightly, the MACAW leads. All protocols perform a quite stable fairness for different densities. Second is DCF surrounding the average af 0.4 with a slight downwards trend. LED starts with the same value for low density, takes a 0.1 step towards unfairness and then covaries parallelly to the original DCF. Last, with an average of 0.3, MACAWBEB acts similar to MACAW, slightly worse.

The fairest algorithm has also the highest rate of time to succcessful transmission of a whole message. The time to success per message in Figure 4.9 shows that MACAW time to success increases faster than the others. From 100 rounds per message to 250 in average. The others increase by 80-100 rounds from

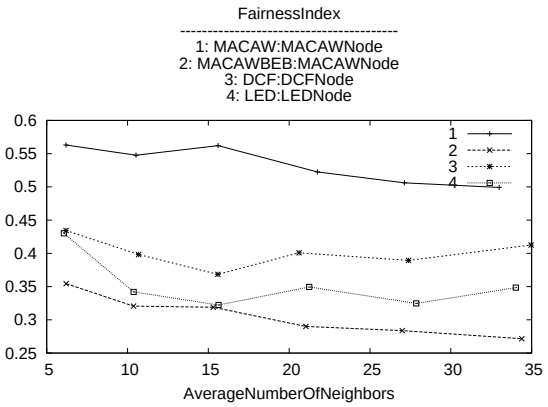


Figure 4.8: Density Analysis: Fairness Index

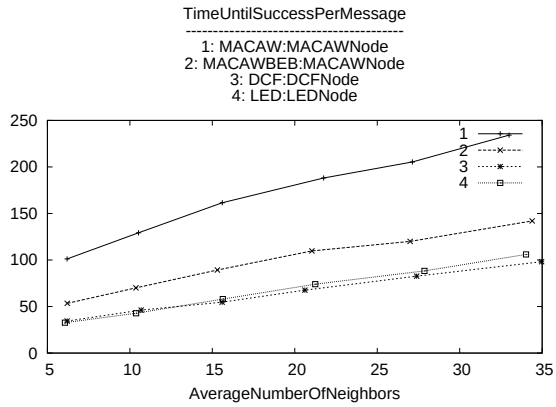


Figure 4.9: Density Analysis: Time Until Success Per Message

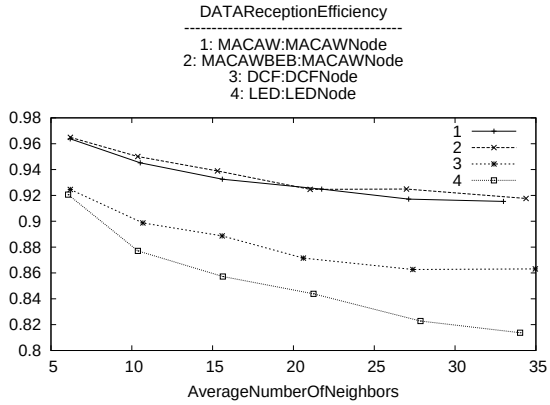


Figure 4.10: Density Analysis: DATA Reception Efficiency

lowest density to highest density. MACAWBEB starts around 50 rounds and DCF and LED have the same performance and start around 30 rounds per message for low density.

The reliablity of the protocols shown in Figure 4.10 points out clearly that the MACAW protocols have a better reception efficiency regarding only the data frames. With a reception rate of 0.96 decreasing with higher density to 0.92 both MACAW protocols, independent of the backoff algorithm used, show the same performance. DCF and LED start at the same point for low density, 0.92. Where in LED the data experiences more collisions the denser the network is. For an average of 35 neighbors per node DCF has a 0.86, LED a 0.8 data reception efficiency.

The impact of the backoff algorithm is significant for the performance. BEB produces less collisions than MILD backoff algorithm, has a significantly less time to success per message, and a little more throughput, but MILD is way fairer than all BEB based protocols.

The influence of the transmission sequence used and the inbuilt of CSMA in the protocol can be seen in Figures 4.6, 4.7, and 4.10, higher throughput and less collisions combined with a worse data reception efficiency. Hence, the collision index is caused by colliding control messages in the MACAW transmission sequence.

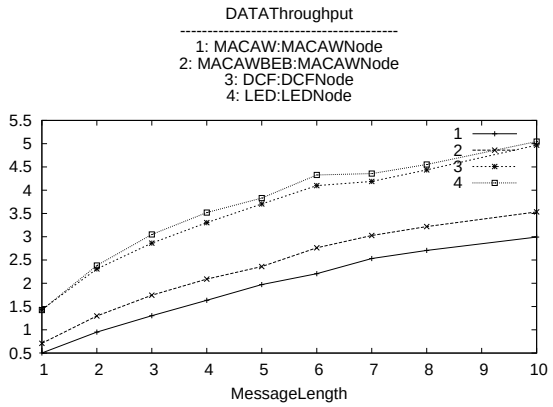


Figure 4.11: Message Length Analysis: DATA Throughput

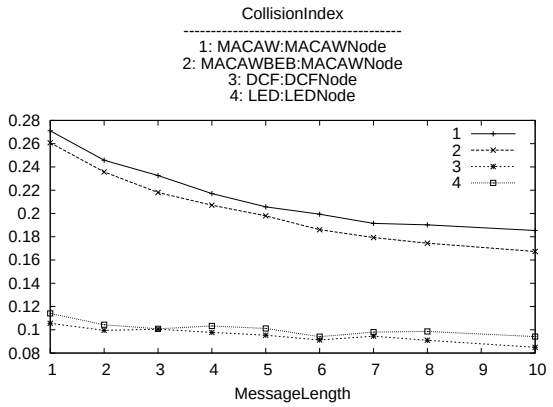


Figure 4.12: Message Length Analysis: Collision Index

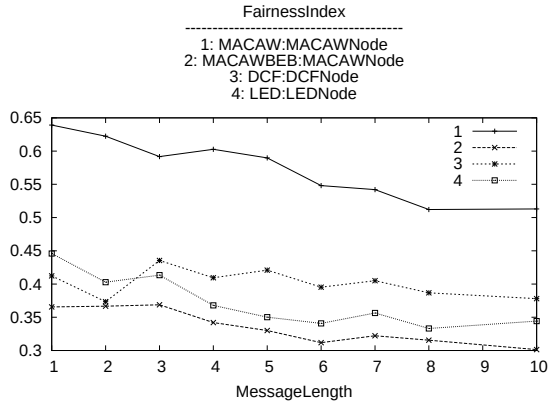


Figure 4.13: Message Length Analysis: Fairness Index

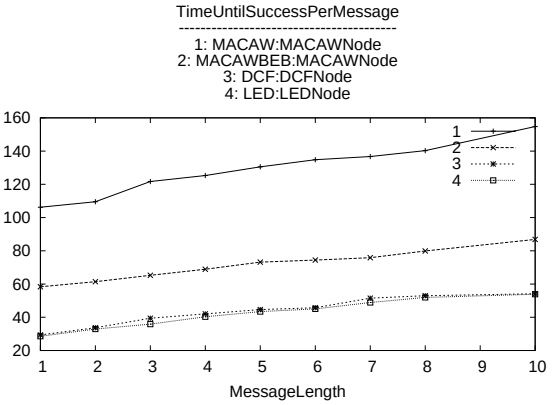


Figure 4.14: Message Length Analysis: Time Until Success Per Message

4.2.2 Message Length Analysis

The network behaviour for longer messages changes. Since the overhead shrinks and the reliability of the transmission gets more weight regarding the performance. The message length simulated varies from 1 to 10 frames per message. All nodes use the same message length.

Figure 4.11 shows the data Throughput (throughput). All protocols at least triple their data frame throughput for a message length from 1 frame to 10 frames lentgh, where the DCF based protocols have with 1.5 to 5 frames per node and 100 rounds , in average, the double of the MACAW based ones. With an increasing message length the difference shrinks. MACAWBEB has 40% more throughput than MACAW.

The collision index, depicted in Figure 4.12 shows that the DCF based protocols behave the same, decreasing a little (around 4%) in the region of 10% colliding messages. The MACAW based protocols start off at 27% collisions and decrease strongly with the shrinking overhead (in relation to the message length). For a message length of 10 frames only 18% collisions occurr. The MACAW using BEB performs a little better ($\sim$5%).

Regarding the fairness, Figure 4.13 shows that the fairness decreases with a longer message, for the protocols using BEB only a little in a region from 0.3 to 0.45, where the DCF protocol is the fairest. Special is the behaviour of DCF for small message length, it shows an irregularity, its fairness index is below the one of LED and crosses its line for a message length of 3 frames. MACAW using MILD

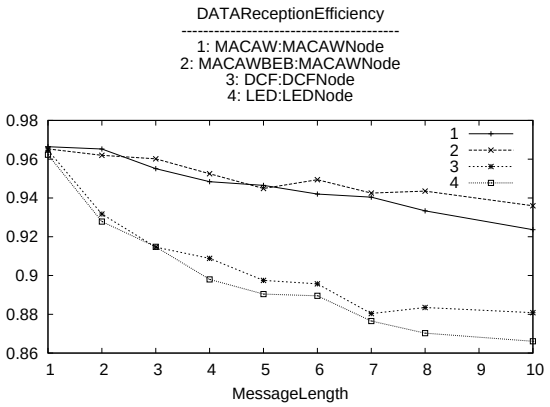


Figure 4.15: Message Length Analysis: DATA Reception Efficiency

backoff algorithm protocol starts at 0.65 and decreases stronger under the influence of a longer message. Its performance decreases to 0.5 fairness index and is the fairest for all message lengths.

The time a message needs for a successful transmission is shown in Figure 4.14. As expected, the time to successful transmission increases with the message length. For MACAW from 110 rounds average to 150 rounds, MACAWBEB needs in average only half of the time (60 to 90 rounds average). The DCF based protocols perform the same, starting from 30 rounds average to 50 for a message length of 10 frames.

All protocols perform the same data reception efficiency for a short message (one frame), as shown in Figure 4.15. MACAW and MACAWBEB keep their performance which decreases only to 93% for a 10 frames message length, where the DCF based protocols' performance goes down to 87% arrived messages. LED performs a little worse. The differences between the single protocols with the same base are more obvious with the increasing message length.

The behaviour of the protocols are devided by the backoff algorithm used regarding the time to success and the fairness, and otherwise by the base (MACAW or DCF). For longer messages, more data flows but the reception efficiency decreases. Less collisions occur overall with less overhead for reliable protocols regarding the data frames.

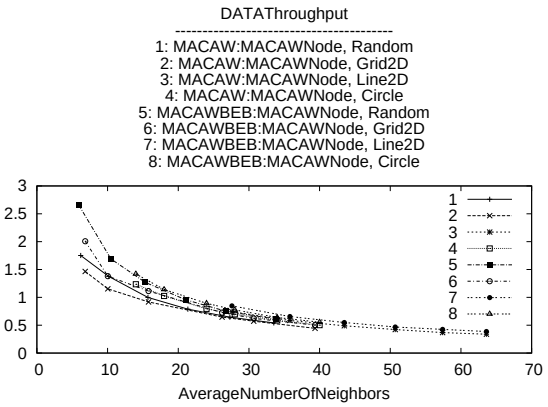


Figure 4.16: Topology Analysis: DATA Throughput, MACAW

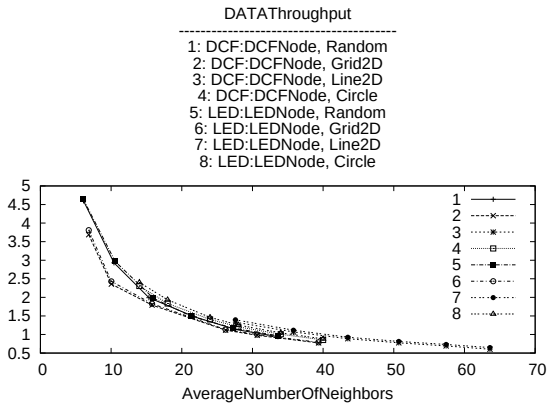


Figure 4.17: Topology Analysis: DATA Throughput

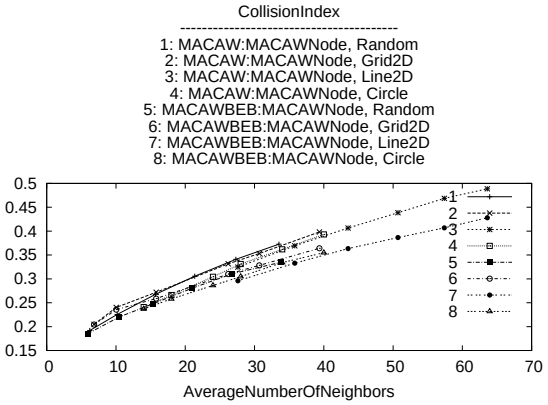


Figure 4.18: Topology Analysis: Collision Index, MACAW

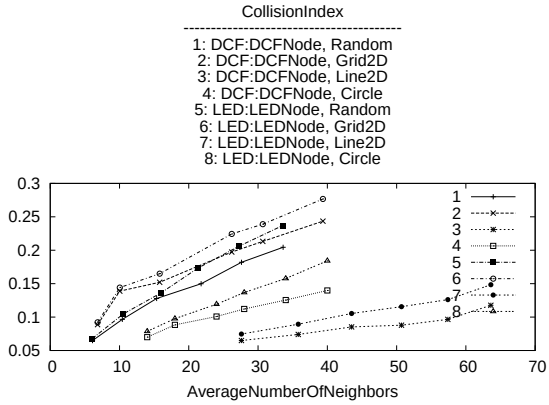


Figure 4.19: Topology Analysis: Collision Index

4.2.3 Topology Analysis

Protocols act differently in different topologies. Here the performance of the protocols in 4 scenarios is shown:

- uniform random distribution
- a grid over the whole simulation area which give the same distance to every neighbor in the horizontal or vertical direction
- a line from top to bottom of the simulation area
- a circle

To obtain readable graphics the protocols are split into two sections, one containing MACAW and MACAW using BEB, the other DCF and LED. The variability of the average number of neighbors can be explained by each topology. Most neighbors are in reach when all nodes are on one line.

Figure 4.16 and Figure 4.17 show that the perfomance regarding the data throughput depends mostly on the density but does not vary much for the different topologies. The DCF based Algorithms have a higher data Throughput but for an average of 65 neighbors per node the performance is approximately the same for both sections.

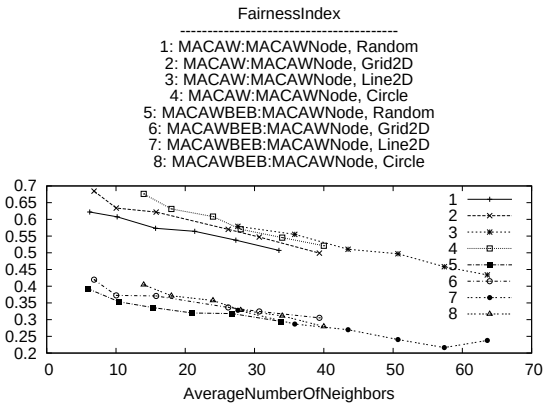


Figure 4.20: Topology Analysis: Fairness Index, MACAW

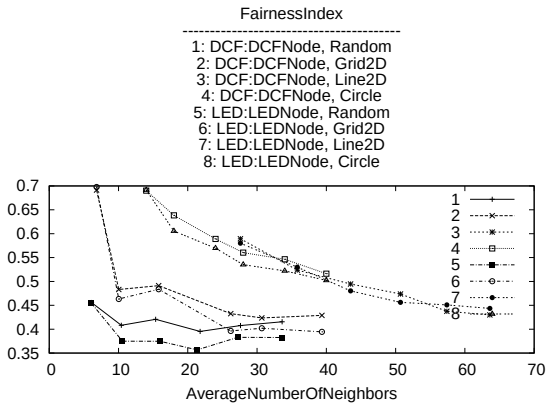


Figure 4.21: Topology Analysis: Fairness Index

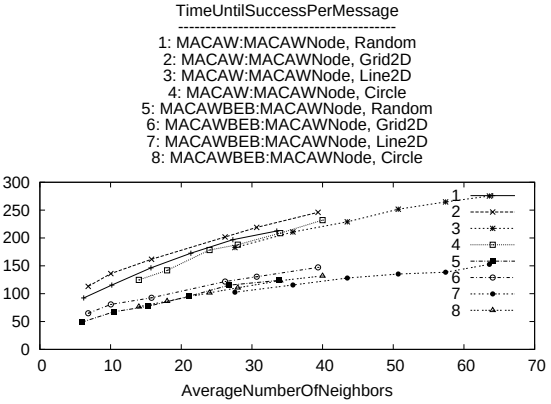


Figure 4.22: Topology Analysis: Time Until Success Per Message, MACAW

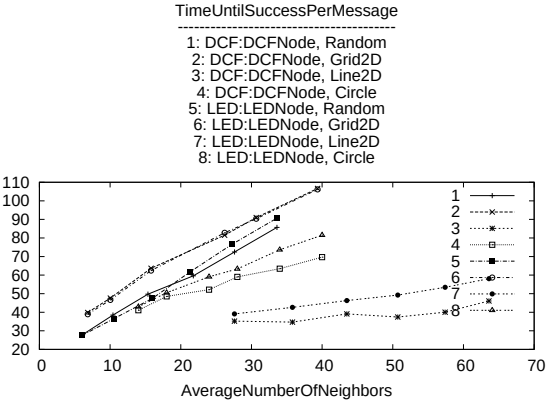


Figure 4.23: Topology Analysis: Time Until Success Per Message

The collision index for the MACAW based protocols in Figure 4.18 is stable with a changing topology. MACAWBEB produces always a little less collisions. The DCF based protocols (Figure 4.19) have the performance already seen in the density analysis for the random distribution, produce more collisions for a grid distribution, and with a more aligned distribution scenario like a circle and a line less collisions. With the line distribution the DCF based protocols have a collision index of only 0.1 for an average number of neighbors of over 60. The LED protocol simulation results for every scenario in a worse collision index than the DCF.

In Figure 4.20 the fairness index performance results are only devided by the backoff algorithm used, not significantly by the topology of the network. MACAW has a higher fairness index, around 0.6 decreasing with the density. MACAWBEB has the same performance in the region around 0.3. Significant in Figure 4.21 is the drastic decrease from the low density region for a grid distribution. The behaviour of the DCF based protocols regarding the fairness index decrease stronger with the higher density than in Figure 4.20. This lines show the best performance in these graphics.

Figure 4.22 shows the average time until successful transmission of a whole message. For both MACAW Protocols the time until success increases for higher density networks. The MACAW protocol using MILD backoff algorithm performance increases more with the higher density and is in average 170 rounds. MACAWBEB performance is in the region of 50 to 110 rounds. Both are stable for all topologies. In Figure 4.23 the results vary with the topology. Worst is the performance for a grid topology, then the random distribution and the grid distribution topology. Line distribution performance separates itself by an almost constant very good performance around 50 rounds per message even for a high density.

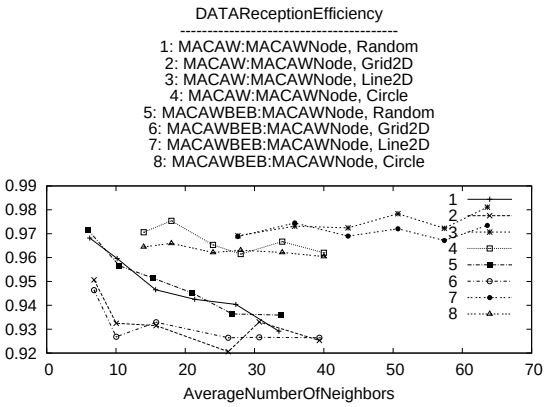


Figure 4.24: Topology Analysis: DATA Reception Efficiency, MACAW

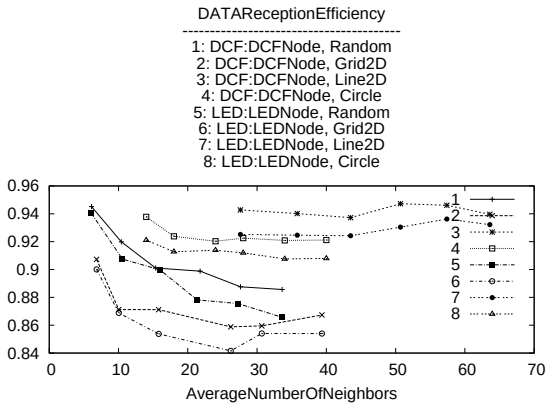


Figure 4.25: Topology Analysis: DATA Reception Efficiency

In Figure 4.24 the MACAW based protocols show no stability for different topologies. The range of variation is from 0.92 to 0.98 i.e., very good data reception efficiency. Circle and line scenario performance are stable with the density, random scenario performance decreases strongly and linear with the density, and grid topology performance decreases in the low desity region and is then quite stable over the density, perform worst.

Figure 4.25 has a range of data reception efficiency from 0.84 to 0.96. Also here the line and circle distributions result in a stability over the density, the others decrease as described above for Figure 4.24.

This Analysis shows the behaviour of the protocols for different topologies. The impact of the topologies is greater for the DCF based protocols, which increase their performance for a circle and line topology. Less interference from more than two sides causes these protocols to work better. Since the circle distribution is not entirely cross connected its performance reaches the one of the line distribution. There is no influence of the topology to the impact of the backoff scheme, both MACAW based protocols are stable for all topologies. One exception here is the data reception efficiency where the MACAW protocols show a behaviour similar to the DCF based ones. Remarkable is the stable data throughput, Figure 4.16 and Figure 4.17 although other performance parameters differ so much.

LED performs always a little worse than DCF for all shown results.

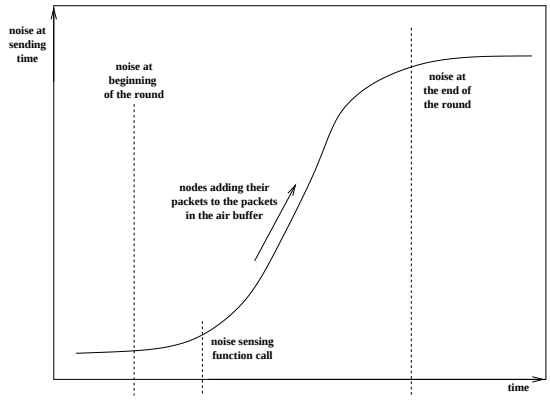
Chapter 5

Conclusion

This chapter discusses the conclusions the simulation results allow. The first section describes the conclusions about the Sinalgo simulator and the influence of it. The second section describes the conclusions for the different protocols in general and the following the specialties of the different analysis comparing the protocols.

5.1 Sinalgo

We used the synchronous mode in Sinalgo to run our simulations. Hence, Sinalgo updates each node sequentially which poses some irregularities in comparison to the real world. The way of carrier sensing we implemented goes through the packets-in-the-air buffer, where packets are added when a node wants to send a message. This buffer is always set to zero pre round, so the first node using the CSMA function will hear no noise at all, the last node called senses the most.



We randomzed the sequence in which the nodes are called to grant fairness as far as possible. As depicted in the graphic besides it may happen that a node at sensing time senses very low noise at its position since its function call is at the beginning of a round and the noise increases drastically during the round that the next round the same node can not decode a incoming message, because it is desturbed by messages added later to the buffer i.e., nodes that are in a sending process and do not use the CSMA before sending or nodes in reach not blocking their own transmission based on the LED function.

The CSMA functionality can so far not be implemented a hundred percent accurately in the Sinalgo simulator.

Furthermore we used the UDG connectivity model, which makes it impossible to overhear a message for a node out of reach, even if there is very little noise. This restriction does not allow the LED mechanism to work properly.

5.2 Protocols

In this section the protocols' performance in general are described and some issues are taken out and explained in detail.

5.2.1 Aloha

The performance of Aloha protocol is strongly dependent on the sending probability used by each node. A higher density results in less data throughput for all sending probability coefficients (k). The fairness index and the time until success per message increase with the density, the other performance indicators are almost stable. The change of k changes all performance indicators. For k equal one the sending probability is very high and a lot of nodes attempt to send messages in high frequencies. This frequency lowers, but the probability of a collision increases, with an increasing density. Therefore the data reception efficiency and the collision index is stable. With low k and high density more data is send and more data collides, the time until success rises in value, all messages need more time and the protocol becomes fairer, but the throughput decreases.

A k between two or three deliveres the best performance since in terms of data throughput the value falls with an increasing k and all others improve in big steps until two (fairness, time to success) or three (data reception efficiency and collision index) and then stay in that region. The throughput means to keep the value as low as possible and the other values mean to keep it up. The balance is found with a value of two or three.

5.2.2 MACAW

The MACAW protocol, independent of the backoff scheme, shows always more collisions than the DCF based protocols, but a higher data reception efficiency. The collision index must be caused by control messages. It takes long for a node to establish a connection but then the reliability of the connection is high. This effect is caused by the transmission sequence used. Upon overhearing an RTS packet the node only backs off until there is a DS package, otherwise it does not set the NAV to the value of a full message transmission time. Shorter NAV periods per connection cause more control message traffic and so more collisions. The fact that only virtual carrier sensing is applied (no CSMA) enhances the probability of a control message collision.

The MACAW protocol is cautious and takes more time to establich a connection, this can be seen in all graphics showing the time to success per message. This behaviour of the protocol is reflected in the data throughput.

The fairness of the protocols is mostly determined by the backoff scheme used. The MILD backoff algorithm has much longer times to success per message but is more fair (fairness index: $\sim$0.55) than the BEB (fairness index: $\sim$0.35). Also the collision index and the throughput suffer from the fairness. To achieve fairness the protocol must forfeite other performance values.

5.2.3 DCF

The DCF protocol shows best performance regarding the data flux, collision index, and time until success per message. The data reception efficiency is always over 80% and the fairness index is in the region of 0.4 for all simulations. This protocol has a good data throughput and performs not bad in all scenarios. It is a protocol suitable for multiple purposes.

5.2.4 LED

The location enhancement to DCF performs always a little wors than the original DCF. As explained in the first section the sequential call of the nodes' sensing function and the usage of the UDG connectivity model causes the LED concept not to work. The idea of local enhancement is to reconsider the blocking decision due to the position information. This is only useful and causes improvement in low noise regions where a CTS or RTS message is overheard and the node does not block its own transmission because the node knows that its influence at the connecting nodes is not significant. This scenario does not happen with the UDG model and so the LED protocol does not bring any improvement.

But why is it performing worse then? It may happen that a connecting node senses very low noise at sending time even if there is much noise because it is called at the beginning of the round. Since the noise at sending time matters in the LED protocol an overhearing node will then decide not to block its own transmission due to the SINR interference model used to calculate the influence over the distance. The noise level then experienced at the connecting node is much higher than estimated and there occurs a collision.

The LED protocol has no chance to establish its advantage but is evendisturbed in its functionality and so performs always a little worse than DCF.

5.3 Comparing Protocols

In this section the impact of certain parameters, such as the density, the message length, and the topology of the network are examined in general terms and comparing the protocols simulated.

5.3.1 Density Analysis

The density analysis shows clearly that the performance of all protocols gets worse the denser the network is. Collisions increase, throughput, and the reliability decrease, fairness remains stable. In a denser network it is difficult to make a protocol work properly since due to the high number of nodes in reach it is more coincidential and so more difficult to control.

5.3.2 Message Length Analysis

With a longer message length the throughput improves for all protocols, less collisions occur. Also the fairness decreases because longer messages make nodes wait longer and the traffic is less fair. Longer messages have a greater impact on the collisions for MACAW. Longer messages mean that there is less overhead and since the high collision rate for MACAW was caused mostly by the control messages it decreases.

For DCF the reliability of a data transfer suffers with an increasing message length. The DCF protocol uses no DS messages to indicate a successful handshake and nodes overhearing an RTS message back off for a whole message transmission length in vain if the CTS or DATA transmission is disturbed. The longer waiting times make the moment when a node which waited in vain and becomes active again more random. If then the CSMA functionality is not granted it may interrupt an ongoing data transmission.

5.3.3 Topology Analysis

For all topologies simulated the throughput of all protocols is the same and dependent on the density. The MACAW performance indicators are stable for all topologies, except the reliability, this improves

the same as DCF. It improves due to less interfering actions by other nodes transmitting, explained in further text.

The DCF performance indicators increase all the more linear a topology is. With a line topology the performance does decrease very little even for a high density. This improvement of performance can be explained by less unexpected interference. If interference comes only from two sides, as given in a line topology it can better be estimated and so the sensing algorithm has more grip, even though its functionality is not a hundred percent accurate.

Furthermore in a linear topology the backoff algorithms work more appropriate due to the less random behaviour caused by the symmetry of a line or a circle.

The higher fairness with the linearity has to be seen like the following. When one connection is established, many nodes wait due to the high density and nodes just out of reach are in contend state (ready to transmit) and only hold back by either the CSMA or the random time to wait before transmission. Like this some kind of alternating behaviour comes up and the nodes able to send a messagecan do this quite sequentially. So few, but many different nodes can send their messages and the fairness increases. Because of the higher density and more efficient backoff schemes for linear networks only few , but well organized, traffic takes place. So the performance is better but the throughput is stable.

Bibliography

[1] O.Goussevskaia, T.Moscibroda, R.Wattenhofer, "Competitive and Fair Distributed Media Access in Wireless Networks," Zurich, CH, 2006

[2] V.Bharghavan, A.Demers, S.Shenker, L.Zhang, "MACAW: A Media Access Protocol for Wireless LAN's," London, UK, 1994

[3] T.Nadeem, L.Ji, A.Agrawala, J.Agre, "Location Enhancement to IEEE 802.11 DCF," IEEE INFOCOM 2005, Miami, USA, 2005

[4] J.G.Proakis, "Digital Communications," 4th edition, p. 270, McGraw-Hill, New York, USA, 2000

[5] The IEEE 802.11 Distributed Coordination Function

[6] G.Brar, D.M.Blough, P.Santi, "Computationally Efficient Scheduling with the Physical Interference Model for Throughput Improvement in Wireless Mesh Networks," MobiCom '06, Los Angeles, USA, 2006

[7] M.Ergen, P.Varaiya, "Delay Analysis of Distribution Coordination Function in IEEE 802.11," CA, USA, 2005

[8] C.Chaudet, D.Dhoutaut, I.G. Lassous, "Performance Issue with IEEE 802.11 in Ad Hoc Networking," IEEE Communications Magazine, 2005

[9] P.Chatzimisios, V.Vitsas, A.C.Boucouvalas, "Throughput and Delay Analysis of IEEE 802.11 protocol", UK, 2003

[10] A.S.Tanenbaum, "Computer Networks," Fourth Edition, Pearson Education, 2003

[11] The Sinalgo Project Homepage
`http://dcg.ethz.ch/projects/sinalgo/`

[12] Gnuplot, a portable command-line driven interactive plotting utility
`http://www.gnuplot.info/`

[13] D.M.Chiu, R.Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," Computer Networks and ISDN Systems, 1989

[14] R.Jain, "The Art of Computer Systems Performance Analysis," John Wiley and Sons, 1991

Appendix A

Tutorial *simtest*

A.1 Introduction

simtest is an extension to a framework which runs bunches of simulations with Sinalgo simulator to analyse Media Access Control (MAC) protocols and to generate automatically graphics to interpret the results. simtest runs a simulation and gathers afterwards the specified values to store them to data files from which it will draw graphics.

To achieve this settings and filestructures must be in a proper shape. The file with the simtest input settings must be named 'simtestParameters.xml' and reside in the simtest root folder. This tutorial helps you customize the framework to your specific needs. Refer to the README file in the simtest root folder for quick access. If you run Linux, put a copy of the 'runsinalgo' bashscript from the simtest root folder in the root folder of Sinalgo. Too much output to the console can cause a buffer overflow in the executor class.

For further information or questions about Sinalgo refer to the Sinalgo homepage.

A.2 Inputfile

Delivered with the simtest package there is a simtestParameters_TEMPLATE.xml file explaining itself so far as ready to start. For a deeper understanding the functions will be explained more in the following sections. A simtest parameter file can look as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Document>

<input>

<execpath value="/home/nick/Sinalgo/"/>
<sinalgologfile value="/home/nick/Sinalgo/logs/log.xml"/>
<logpath value="/home/nick/Sinalgo/SimData/"/>
<rawdatapath value="/home/nick/Sinalgo/SimData/RawData/"/>
<gnuplotCommand value="gnuplot"/>

<OS value="linux"/>

<fileModification value="yes"/>
```

```
<outputformat value="png"/>

<numberOfSimulationCallsPerConfiguration value="30"/>

<nodename value="2" value0="LED:LEDNode" value1="DCF:DCFNode"/>
<noOfNodes step="4" Min="100" Max="400"/>

<rounds step="0" Min="500" Max="1000"/>

<overwrite value="2" value0="SINR/alpha=2" value1="SINR/alpha=3"/>

<distributionModel value="1" value0="Random"/>
<connectivityModel value="1" value0="UDG"/>
<interferenceModel value="1" value0="SINR"/>

<runningParameter type="overwrite" value="7" value0="UDG/rMax=100"
 value1="UDG/rMax=150" value2="UDG/rMax=200" value3="UDG/rMax=250"
 value4="UDG/rMax=350" value5="UDG/rMax=450" value6="UDG/rMax=600"/>

</input>

<output>
<!--outputnames matter for logging and have to correlate with Sinalgo logs-->
<roundanalyse>

</roundanalyse>
<parameteranalyse>
<!--TEMPLATE:  <plotpair XValue="" YValue=""/>  -->
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="AverageReceivedMessagesPerRoundAndNode"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="NumberOfNodes"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="NumberOfReceivedMessagesOverall"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="MessageCollisionsOverAll"/>
<plotpair XValue="UDGrMax"
 YValue="AverageNumberOfNeighbors"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="ReceivedDATAMsgToDATAMsgBroadcastsRatio"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="FairnessIndex"/>
<plotpair XValue="AverageNumberOfNeighbors"
 YValue="AverageDATAMessageReceptionPerRoundAndNode"/>
</parameteranalyse>

</output>

</Document>
```

In this example 8 graphics with the averages of 30 simulation calls per 56 different configurations will be drawn. Each will contain 8 lines with 7 datapoints per line. Raw data will be stored in
`/home/nick/Sinalgo/SimData/RawData/`
and the graphics in
`/home/nick/Sinalgo/SimData/`.

inputparameters	
paths	Execution path, path to Sinalgo log file, simtest log path, raw data log path. Put here the correct paths, do not forget the slash at the end of a path declaration.
gnuplot command	The gnuplot command line command may be different for different systems, usually it is `gnuplot` for Linux systems and `pgnuplot` for Windows.
OS	Operating system, Windows or Linux.
file modification tag	This tag determines if the database in the log path is modified rerunning all simulations or just the graphics are drawn from it, useful for automatic plot, if you changed the database (erased or added a data block) or want to create another outputformat.
output format	Can be either PNG or EPS, PNG is colored, EPS is more readable included in documents, but in vertical format, use `\includegraphics[angle = -90]{imgName}` to put them into a document
No of simulation calls per configuration	The amount of similar sinalgo calls to minimize variance. The raw data will be stored in the folder specified in the general settings. The variance is always plotted besides the graphic, the same filename with an appended `_variance`.
Sinalgo parameters	Node name (containing project name) No of nodes to create, No of rounds to run before termination, models (distribution, interference, connectivity), and the rest has to be introduced as overwrite statement. The overwrite statement makes the simtest program flexible if you introduce new parameters.
runnning parameter	Also a Sinalgo parameter, but due to this sequence the graphical output is adjusted. The line break in the plot is according to this parameter. All other parameters are represented as different lines in the same plot.
parameteranalyse parameters	Plotpairs with an X tag and a Y tag to retrieve information from the Sinalgo log file and to determine which graphics to draw. The X value should covary with the running parameter, else the plots are unreadable. These tags have to be introduced in the Sinalgo log file as XML tags with the same String.

Table A.1: *simtest* Input Parameters

A.3 Configure Sinalgo Output

The Sinalgo log file must be named as written into the 'sinalgologfile' tag. XML format can be obtained by using the following instructions and functions:

```
Logging log = Logging.getLogger("log.xml");

public void preRun() {
// A method called at startup, before the first round is executed.
log.logln("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n <Document>\n<simtest>");
log.logln("<!--*****************SIMTESTDATA*********************-->");
}


public void onExit() {

// logXMLln("", +"");
logXMLln("recMsgOverall", ""+numberOfReceivedMsgOverAll);
logXMLln("noOfNodes",""+noOfNodes);
log.logln("<!--**************SIMTESTDATA************************-->");
log.logln("</simtest>\n</Document>");
}

public void logXMLln(String name, String value){
log.logln("<"+ name + " value=\""+value+"\"/>");
}

public void logXMLcomment(String comment){
log.logln("<!--"+comment+"-->");
}
```

The first line will generate log file called log.xml. The `PreRun()` method creates the necessary XML tags at the beginning and the last line of the `onExit()` function writes the closing tags. Its first three lines are an example of how to create two entries, `recMsgOverAll` and `noOfNodes` using the logXMLln method declared below. Introduce these methods to the CustomGlobal class of your project and these entry names can be used to generate the graphics.

All the tags used to generate the log file entries have the same name as in the simtestParameters.xml file under the output section.

A.4 Execution

You can execute simtest, like Sinalgo, either from the eclipse development environment or from command line interface. In eclipse the jdom.jar binaries must be included to the project, if not done on project generation. From command line change to the root folder and type
`java -cp "binaries/jdom.jar:src:bin" simtest.Main`
for Linux and
`java -cp "binaries/jdom.jar;src;bin" simtest.Main`
for Windows.

List of Tables

List of Figures