Evaluation of Routing Dependability in MANETs using a Topology Emulator

(MASTER THESIS BY GROUP 1039)

Group 1039

Morten N. Jensen
Anders Nickelsen

Submitted June 11th 2007

AALBORG UNIVERSITY

AALBORG UNIVERSITET

Institute of Electronic Systems, Department of Control Engineering
Fredrik Bajersvej 7
Telephone 96 35 87 00
Fax 98 15 17 39
http://www.ies.aau.dk

Title:

Evaluation of routing dependability in MANETs using a topology emulator

Theme:

Distributed Application Engineering

Semester:

9th and 10th

Project group:

1039

Authors:

Morten N. Jensen
Anders Nickelsen

Supervisors:

Erling V. Matthiesen
Hans-Peter Schwefel

Copies: 5

Report (pages): 125

Appendix (pages): 18

Total (pages): 152

Submitted: June 11th 2007

Enclosed CD-ROM contains source code and downloaded references.

Synopsis:

Performing experimental evaluation of wireless networking is prone to large differences between experiments as the non-deterministic nature of the environment may influence the results differently between tests. This fact has motivated the work described in this report, which constitutes the development and deployment of a topology emulation testbed for evaluation of routing dependability in mobile ad hoc networks. The context for routing evaluation is based on a use case where a highly dependable video conferencing application is used to provide medical assistance from a hospital to an ambulance in motion. This scenario entails ad hoc multi-hop routing from the ambulance to a gateway in the ad hoc network.

The topology emulator imposes link properties of wireless links onto wired links between end-nodes, based on detailed loss and delays models incorporating properties of WLAN 802.11a with OFDM physical layer and statistical fading models for the channel. Functional and performance requirements to the topology emulation testbed have been derived and the testbed has been implemented and verified towards these requirements.

A routing evaluation framework consisting of performance and dependability requirements has been defined based on the use case. Within this framework DSR and OLSR, representing the reactive and proactive families of ad hoc routing have been evaluated.

From the evaluation results it is concluded that the reactive DSR is a better choice for ad hoc routing in highly dynamic ad hoc scenarios, as the end-to-end reliability of DSR surpasses that of OLSR. However, more systematic evaluation of the influencing factors from mobile environments must be performed, to further substantiate this conclusion.

Contents

Preface

The work described in this report has been conducted on the 9th and 10th semester of the master program Distributed Application Engineering under the Department of Control Engineering at the Institute of Electronic Systems at Aalborg University, in the period September 4th 2006 to June 11th 2007.

On this basis, the process of developing and deploying a framework for evaluating routing dependability, has been documented.

The report is targeted at an audience with the same academic level as the project group.

Report organization

The report is divided into chapters describing the steps performed in the development of the testbed and the evaluation of the routing protocols.

Structure

Chapter 1 is the introduction to the concepts of dependable services and contains the motivation for this work. In Chapter 2, the scenario is presented in which the application used as a case is described. Chapter 3 describes an in-depth pre-analysis of the properties of the case analysis with focus on the ad hoc domain, the provisioning of dependable service and an analysis of methods feasible for evaluating dependable solutions. Chapter 4 concludes the pre-analysis with a problem statement, describing the main objectives of this project and specifying delimitations from the case. Requirements to the evaluation testbed are derived and described in Chapter 5 and in Chapter 6 the design of the entire topology emulator is described as well as implementation issues and the results of the implementation verification towards the functionality. In Chapter 7, the performance of the implemented topology emulator is verified towards the specified requirements. Chapter 8 and 9 contain the specification of a dependability evaluation framework and the evaluation of two routing protocols within this framework. Finally, Chapter 10 and 11 provides the conclusion of this work and outlook to future work.

Appendices

In supplement to the documentation, appendices have been produced in order to document in-depth specifications of certain parts of the development, modeling process and the system as well as detailed test-results. These are presented in Appendix A-F.

Reading guide

Throughout the report references such as [Alex Varshavsky, 2004] are made to entries in the bibliography contained in the back of the report. In addition to the report, a CD-ROM is enclosed containing simulation traces, MATLAB scripts, source code and copies of downloaded references from the bibliography.
Besides references of books and technical reports, the bibliography also contains references to documents downloaded from the Internet. A such entry contains both the URL from which the file can be downloaded and the location of a downloaded file on the enclosed CD-ROM.

Morten N. Jensen

Anders Nickelsen

Chapter 1

Introduction

This chapter describes the background of providing dependable services. This background motivates for researching dependable services with special focus on mobile ad hoc environments. Based on that, the initial problem to be investigated is outlined. This problem is derived from the overall problem domain of mobile wireless ad hoc networks.

1.1 Background

A challenge in service provisioning is that of ensuring *dependability*. Dependability of a service means that it is one that can justifiably be trusted. This is an important aspect perceived by the user of the delivered service, as the user requires a high degree of trustworthiness in a service.

As wireless devices have become commercial off the shelf products, new environments emerge for service provisioning. The environments are either built based on an infrastructure or arise ad hoc between devices. These environments, that are typically characterized as *unreliable*, pose numerous new threats to the dependability of the services delivered. Properties such as availability of both devices and links and the number and types of faults that occur during service provisioning entail refinement of the existing service provisioning mechanisms. For instance, links are externally interfered causing packet-losses more frequently. Also, issues such as security on the link level must be considered. Furthermore, the ongoing development of the environments regarding mobility of the devices introduce even more complex problems to be handled. As devices move around, dynamics are introduced on their inter-connections. Resilience toward e.g. rapidly changing topologies must be considered from a service point of view to ensure service dependability in the mobile ad hoc environment.

HIDENETS [HIDENETS, 2006b] is a project investigating challenges in service provisioning in highly dynamic network environments. The use cases used in this project involve car-to-car scenarios containing both ad hoc and infrastructure domains. Providing services in such domains requires special focus on dependability, as some services are safety critical. This means that the result of a failing service can be catastrophic. Thus fault tolerance aspects of dependability are of particular interest to the project.
Several layers of services are considered in the HIDENETS project as depicted in Figure 1.1 on the facing page; communication layer services are provided to the middle-

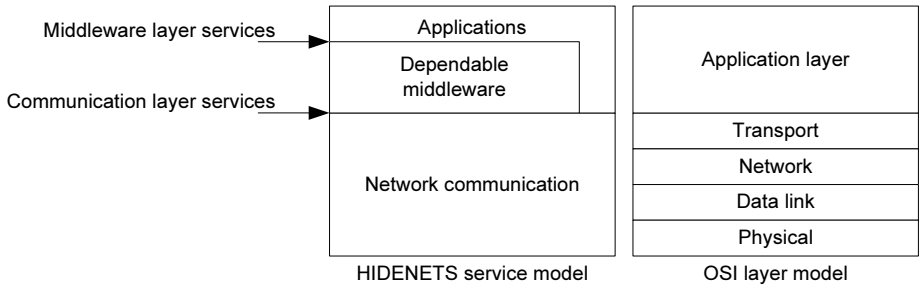

Figure 1.1: HIDENETS model of where services are provided, i.e. from the communication layers and also from a middleware layer to the applications.

ware layers in which middleware layer services are provided to the application layer. Examples of communication services are radio resource management, medium access control, network routing and transportation. Examples of middleware services are session control, naming, profile management and cooperative backup.

As with all other development, testing and evaluation of the models and the solutions are mandatory tasks. However, when it comes to testing solutions for mobile ad hoc networks several issues emerge. Often the environments of these services are very complex, making a completely analytical evaluation intractable. Similarly, as the solutions often include numerous entities and require complex mobility patterns, the effort put into performing a purely experimental evaluation often exceeds the gain derived from it. Also, difficulties in ensuring repeatability of the experiments are dominant in experimental setups. In-between lies simulation and emulation. Often, complete simulation is tedious, as re-implementation of all functionality is required by the simulator. Therefore, several suggestions have been developed of using combinations of emulation and simulation for evaluation of (mobile) ad hoc network solutions [Ke et al., 2000][Zheng and Ni, 2002].

1.2 Initial problem description

This project focuses on investigating highly dependable services in mobile ad hoc networks. The main focus is on communication layer services as these are basic features necessary to attain high dependability. Considering the importance of test and evaluation of solutions, the aim is to develop an evaluation framework for existing and future solutions. To do this, several objectives of the following pre-analysis have been specified.

Specification of a use case The use cases in the HIDENETS context contains a high degree of topology dynamics. One of these use cases, providing challenges for communication services, is selected as a foundation for the requirements to the communication layer services.

Examine requirements to the dependability of services To derive such dependability requirements, an analysis of the applications and services residing in the scenario is performed. Services exist that each are dependable to some extent and it is investigated how these behave in the context of a highly dynamic scenarios.

Analysis of ad hoc environment As the dynamics of the scenarios are caused by the properties of the ad hoc environment, these properties are analysed to establish a view of the underlying factors creating the dynamics.

Identify evaluation methods It is investigated which approach is appropriate in the context of evaluating dependability solutions. Based on the chosen approach an evaluation framework is developed, which is suitable for testing both communication and middleware services. This framework should also be usable for evaluating upper layer applications utilizing the services. A such framework could incorporate a network emulator emulating the dynamics, e.g. mobility and link properties, of the environment.

Chapter 2

Scenario

This chapter covers the use case selected to form a platform for the analysis of dependable solutions in ad hoc networks. The use case is selected from the HIDENETS D1.1 [HIDENETS, 2006b], which this chapter is based on. However, only selected parts of the use case are used to form the basis for the analysis. The applications associated to the use case are described along with the services needed to support these. The purpose is to provide provide the requirements set by the different applications.

2.1 Use case

The use case described in the following is a subset of the use case *Car accident (incl. distributed black-box)*, [HIDENETS, 2006b, p. 52].

The setting for the overall use case is an accident on a road where a number of people have been injured. In this setting a range of applications shall be utilized using ad hoc and infrastructure IP based networks.

After the accident an ambulance arrives at the scene. Some of the people involved in the accident are seriously injured and medical expertise is needed, which is not present at the crash site. To accommodate this, applications are used that make it possible to stream voice and video to a nearby hospital along with the patients blood pressure, pulse and other vitals. These applications are used at the crash site and on the way to the hospital and are referred to as *Access to medical expertise (multimedia)* applications and consists of video conference, streaming and interactive data applications. The functionalities provided by these applications are utilized and supported by a number of *actors*. The actors are listed below along with their role or purpose:

End user: Person using one or more applications.

Terminal: Interface through which applications may be utilized.

Gateways: To support the communication between hospital and ambulance a gateways are needed which connect the ad hoc and infrastructure domains. Any end user can provide a gateway e.g. a PDA which is both UMTS and Wi-Fi enabled, can provide a link between the Wi-Fi, potentially ad hoc, domain and the UMTS domain. There may also be stationary gateways, however, it is assumed that the majority of gateways will be provided by end users.

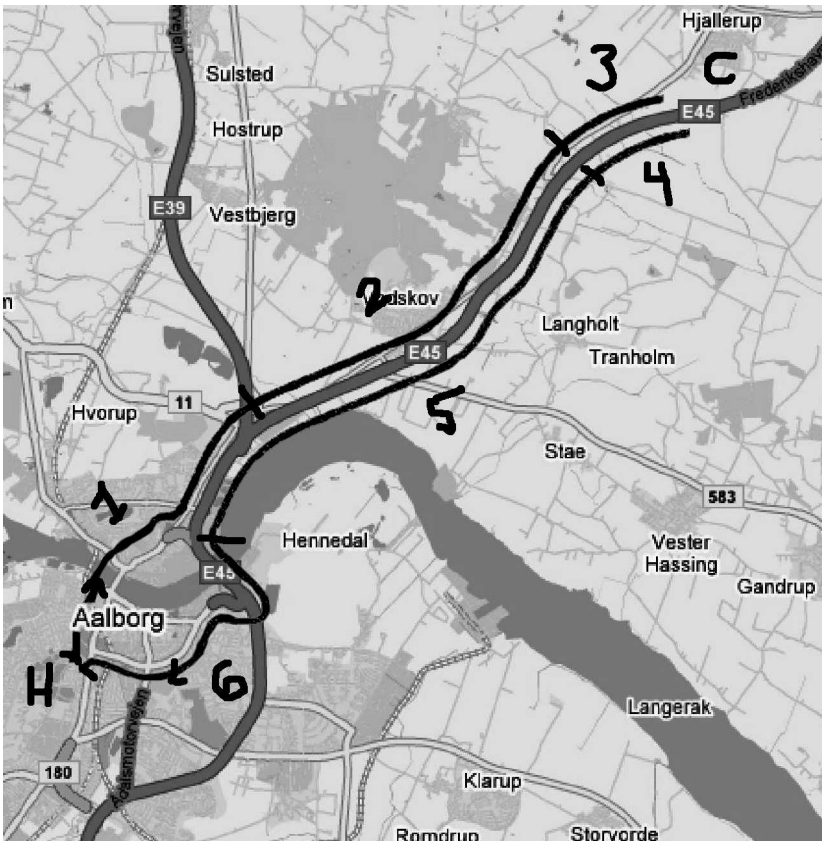

Figure 2.1: Map depicting the route which the ambulance follows to and from the crash site and the different segments which the route is divided into.

In order to render the roles of the different actors visible and to investigate possible issues a scenario is described in the following. The scenario is played out in Aalborg (Denmark) and constitutes a map, a movement pattern for the ambulance and a number of actions performed by the actors of the scenario. The map and the route which the ambulance traverses can be seen in Figure 2.1 on the preceding page. This route is divided into a number of segments. For each segment the following, relative, parameters are be specified:

Traffic density: Describes the traffic density (ρ) in terms of cars per kilometre (cpk) road, including all lanes and oncoming traffic.

Average speed: The speed of the ambulance in kilometres per hour (km/h). This parameter is to some extent correlated with the traffic density.

Gateway connectivity: Describes the amount of available gateways.

Applications: The applications used in the specified segment.

Network domain: Specifies the available network technologies. For this scenario, three different technologies are used; IEEE 802.11p[1], GSM and UMTS.

The values for the different parameters can be seen in Table 2.1 on page 13.

Segment 1: The ambulance leaves the hospital, in the center of Aalborg, heading for the crash site. The environment is urban with many tall buildings, a high gateway connectivity and a high car density entailing a low speed and a high network load. When leaving the hospital, the ambulance receives the names of the injured and pulls their medical records using an interactive data application. A terminal, which supports all the wireless technologies of the scenario and has a standard 17" LCD screen, is used for the interaction. UMTS is used to download the medical records. These are defined to have an average size of 2MegaByte (MB).

Segment 2: As the ambulance leaves the city center and enters the freeway, the environment changes to a more open terrain and the car density decreases lowering the network load and the gateway connectivity, and making it possible for the ambulance to increase the speed. In reality the change in parameters will be smooth, however, to decrease the complexity of the scenario the change occurs immediately when moving from one segment to the other. This is the case for transitions between all segments.

Segment 3: Getting closer to the crash site the car density increases once more, thereby also increasing the amount of gateways available and the network load. Furthermore, the speed of the ambulance is decreased.

Segment 4: Having picked up an injured person the ambulance heads back to the hospital. There is still a high car density in the area around the crash site which yields a high gateway connectivity and a medium network load. The paramedics in the ambulance request remote assistance in the form of a video conference with medical experts at the hospital. Furthermore, the patients vitals are monitored at the hospital using a streaming data application. Both applications use UMTS to establish communication with the hospital.

[1] IEEE 802.11p is a draft standard projected to be used for the facilitation of communication in highly mobile Mobile Ad hoc NETworks (MANETs)

Segment 5: A few kilometres from the crash site the traffic density drops, along with the gateway connectivity and network load, and the ambulance increases it's speed. Both the video conference and the streaming data application are still in use.

Segment 6: When the ambulance reaches the city centre the traffic density increases, and so does the gateway connectivity and network load, and the ambulance slows down. Furthermore, the UMTS network in this area is congested making it difficult to meet the requirements of the video streaming application. This forces the ambulance to establish a connection using IEEE IEEE 802.11p. The amount of available gateways supporting IEEE 802.11p is relatively low making it necessary to route the data through multi-hop ad hoc network.

2.2 Applications

The applications needed to provide communication between ambulance and hospital are be parametrized in the following along with the communication level requirements that must be fulfilled to support the applications. The parameters in the succeeding descriptions are taken from [HIDENETS, 2006b].

2.2.1 Video conference

The purpose of this application is to provide two-way real-time voice and video between terminals. In the use case described in the previous section, this includes maintaining communication while moving from the crash site to the hospital possibly switching between different access points or relaying data through several hops in the ad hoc domain to reach an infrastructure gateway. The video conference application has the following communication level requirements:

Throughput: The throughput should be as high as possible, since high quality images should make it easier for the remote medical experts to make assessments. Data rates in the range of 16 - 384 kb/s are normal for video conference applications, however, even higher data rates are likely to be necessary for this scenario.

Communication delay: The delay affects the patient care directly, hence, the requirements to this is high. For end-to-end communication the delay should be below 150 ms.

Delay jitter: For real time video streaming applications jitter is a critical factor, since high jitter might distort the communication, decreasing the quality of the patient care significantly. Hence, the jitter should be as low as possible and not exceed 50 ms.

Data integrity / packet error ratio: The effects of these factors strongly depend on the video and audio codecs used. These often provide different levels of error protection/correction. Since human perception is somewhat tolerant with respect to erroneous information packet errors are acceptable to some extent.

Packet loss: This requirement is affected by the audio and video codecs ability to handle errors. However this is still a strict requirement since the quality of the signal should be as high as possible. Loss of packets may disrupt video and audio making these difficult to interpret by the end users.

Segment	Traffic density	Average speed	Gateway connectivity	Application(s)	Network domain(s)	Network load
1	200 cpk	40 km/h	20 - 40	Interactive data	GSM, UMTS, IEEE 802.11p	High
2	50 cpk	130 km/h	5 - 10	None	GSM, UMTS, IEEE 802.11p	Low
3	150 cpk	30 km/h	10 - 20	None	GSM, UMTS, IEEE 802.11p	Medium
4	150 cpk	30 km/h	10 - 20	Video conference, Streaming data	GSM, UMTS, IEEE 802.11p	Medium
5	50 cpk	130 km/h	5 - 10	Video conference, Streaming data	GSM, UMTS, IEEE 802.11p	Low
6	200 cpk	40 km/h	0 - 5	Video conference, Streaming data	GSM, IEEE 802.11p	High

Table 2.1: Table showing the different parameters for each segment of the scenario.

2.2.2 Streaming data

In the use case this application shall be used to stream the patients vitals and the ambulances position to the hospital. Here correctness of data is the most important parameter. The streaming data application has the following requirements:

Throughput: The throughput depends on the nature of the information to be sent. In the scenario, none of the information requires a high throughput and this parameter is considered of less importance.

Communication delay: As with throughput the delay is not a important factor, however, a high delay will increase the response time to changes in the patients vitals.

Delay jitter: This parameter can make it harder to interpret the data stream, such as the heart rate, correctly, provided this issue is not handled by upper layers. Hence, jitter should be regarded as a critical parameter.

Data integrity / packet error ratio: Correctness of data is very important so the requirements to this should be high. But as the data is updated regularly damaged packets may be discarded without degrading the quality of the application significantly. The received packets should, nevertheless, be in the correct sequence in order to give the right impression of the patients vitals.

Packet loss: Since data is updated regularly packet loss should not be an issue, but should still be kept at a minimum to ensure that the data is as new as possible.

2.2.3 Interactive data

This application shall provide the possibility for a user to interact with other users or servers. In the context of the use case the interactive data application shall be used to pull the information on the injured at the crash prior to arrival. The Interactive data application has the following requirements:

Throughput: This parameter depends on the use of the application. It is expected that the size of a patients medical records is in the magnitude of megabytes. Hence, this application could set high requirements to the throughput depending on how fast the information is needed.

Communication delay: Delay is not critical in the case of the request for medical records. This parameter should however be kept at a minimum to increase the user experience when requesting the records.

Jitter: Jitter is not relevant for this application, since it has no significant impact on the user experience nor the transfer of the medical records.

Data integrity / packet error ratio: Both parameters are handled by TCP. This could entail some issues with regards to throughput, since TCP treats a packet error as a lost packet. This triggers TCP's flow control which decreases the throughput i.e. the paramedics might not receive the medical records in time.

Packet loss: This parameter is critical due to the impact on *Data integrity / packet error ratio*.

In order to meet the requirements set by the applications above, e.g. by making them dependable, numerous communication level services are needed which ensure that the requirements are met. The succeeding chapter will explore this issue.

Chapter 3

Preanalysis

The preanalysis investigates relevant aspects of the scenario specified in the previous chapter. Starting from the ad hoc environment, the effects of the physical properties of the wireless medium and of general issues in wireless networks are investigated, by describing the properties and issues together with their impact on the wireless communication. Another part of the physical environment is node mobility, hence, mobility models suitable for the scenario are evaluated by relating them to the mobility which is likely to be experienced in the scenario. Having examined the physical environment, the concepts of dependability and dependable services in this environment are investigated by relating them to the scenario and determining their impact. Finally, methods for testing dependable solutions are examined by evaluating different approaches. The purpose of this chapter is to provide a rudimentary overview of the concept of dependable solutions in MANETs, making it possible to identify interesting problems and through these define the scope of this project.

3.1 Ad hoc environment

This section investigates the physical environment in which the dependable applications reside. Starting from the physical properties of the scenario, a number of different environments are identified along with their effect on the signal propagation and thereby the operation of the applications. Having identified the different environments the section explores the concept of mobility models and covers a range of different models suitable for the scenario. The purpose is to determine the complexity of the environment of the applications along with appropriate mobility models making it possible to portray movement in a realistic manner with respect to the scenario.

3.1.1 Physical properties

Compared to traditional wired networks, wireless networks are less reliable. This is due to the physical properties of the wireless medium. These properties ultimately introduce a significant increase in the probability of bit errors, leading to lost frames and packets. In scenarios where a high level of dependability is desired, the unreliable wireless channel might make it difficult to meet the requirements need to ensure dependability. Furthermore, the mobility of the nodes has an impact on the properties of

the wireless channel and introduce a dynamic topology where it is difficult to ensure peer to peer connectivity, which further decreases the reliability.

Starting from general problems in MANETs, this section relates these to the environments of the system from the use case (see Chapter 2 on page 9). Two environments, urban and freeway, are derived from the use case. Both have different properties affecting the propagation and quality of the wireless signal. These properties varies depending on the technology used for communication. IEEE 802.11p is not deployed nor widely studied yet, IEEE 802.11a is to model 802.11p, since it is the one used within HIDENETS for this purpose [HIDENETS, 2006a] and since it is very similar to 802.11p [Zang et al., 2005]. UMTS and GSM are not considered further in the project, as the scope of the scenarios are concentrated on the ad hoc domains.

Urban environment
The urban environment is characterized by:

- Potential high traffic density.
- Potential high network load.
- Many buildings.
- Speed limits at 50 km/h and below.
- Many gateways.

Freeway environment
The freeway environment is characterized by:

- Potential medium/high traffic density.
- Potential high network load.
- Few buildings.
- Speed limits at 130 km/h and below.
- Few gateways, depends on the traffic density.

This section lists and investigates a number of general issues and characteristics of MANETs, starting from the characteristics of the wireless channel, and relates them to the environments mentioned previously.

The power if a signal received through a wireless channel varies as a function of space, frequency and time. This generally referred to as fading. Fading may be subdivided into small- and large-scale fading, each containing different factors which affect the wireless channel:

Large-scale fading
Large-scale fading covers the factors which cause the signal to gradually fade as a function of the distance. The two main contributors to this kind of fading are shadowing and path-loss:

- *Pathloss* expresses the decrease in signal strength caused by the propagation of the radio wave in an obstacle free environment. The impact of pathloss is a weaker signal i.e. the Signal to Noise Ratio (SNR) is decreased leading to higher bit error probabilities. This is an unavoidable characteristic for both the urban and freeway environment.

- *Shadowing* describes the degradation of a wireless signal caused by objects between sender and receiver refracting or diffracting the signal. Shadowing is expected to have the largest impact in the urban environment due to the large amount of obstacles. However, research shows that objects such as cars can reduce the signal quality significantly when near the sender or receiver if the antennas are not sufficiently elevated [Gaertner and Cahill, 2004]. Consequently a large vehicle such as a truck or a bus might be able to obstruct the communication between passenger cars. Hence, shadowing should be regarded in the freeway environment.

- Another property which affects the signal in the wireless channel is the *non-isotropic radio propagation* properties of most antennas, entailing that the signal strength varies if receiver and transmitter rotate with respect to each other [Stuedi et al., 2005]. This is an unavoidable characteristic for both environments. However, this effect should be of less importance for the urban environment since the signal propagation is likely to be affected by obstacles before the non-isotropic nature of the antenna becomes significantly pronounced to have an impact on the signal quality.

Small-scale fading

Small-scale fading, as opposed to large-scale fading, may cause large fluctuations in the signal strength with relatively little or no change in the distance between sender and receiver.

- *Multipath fading* occurs when a signal is reflected by one or more objects, entailing that the receiver can receive multiple copies of the same signal. The copies may have different attenuation, delay and phase shift potentially inducing destructive interference on the signal. This leads to fading and as a consequence of this a lower SNR ratio. This characteristic is relevant for the urban environment, since there are many obstacles that can reflect the signal. In the freeway environment little multipath fading is expected, however, reflections from the road and other cars might have an affect on the signal.

- *The Doppler effect* is described by the difference in wavelength observed at receiver and transmitter caused by movement of receiver and/or transmitter. This effect can cause the transmitted signal to be detected wrongfully at the receiver i.e. at a higher or lower frequency, depending on whether receiver and transmitter are moving away or toward each other. In the urban environment the Doppler effect can be neglected, since it has little effect on the wireless signal at velocities below 50 km/h [Gaertner and Cahill, 2004]. At velocities much higher than 50 km/h the Doppler effect can affect the signal. The modulation scheme used has a great impact on the signal distortion caused by the Doppler effect. Wireless cards using Orthogonal Frequency Division Multiplexing (OFDM), such as 802.11g, are sensitive to the Doppler effect, since OFDM uses a number of overlapping orthogonal sub-carries which are very vulnerable to Inter-Carrier Inter-

ference (ICI) when the Doppler effect is present. Conversely, cards using Direct Sequence Spread Spectrum (DSSS), e.g. 802.11b, are relatively unaffected by the Doppler effect at any velocity [Brownfield and Davis, 2005]. Entailing that the Doppler effect should be considered for the freeway environment depending on the modulation scheme, since 802.11p will be using OFDM it is likely that the Doppler effect will have an impact on wireless cards following the 802.11p specification.

3.1.2 General MANET problems

Beyond the physical characteristics of the wireless ad hoc environment potentially decreasing the quality of the communication channel, there are a number of issues on the layers above the physical that influence the wireless communication:

Terminals problems
On the MAC-layer in wireless networks there are two significant terminal problems; The exposed and the hidden terminal. A *hidden* terminal is a node which is unaware of an ongoing transmission in it's vicinity and whose attempt to transmit will disrupt the ongoing transmission. In Figure 3.1(a), A is attempting to transmit data to B, however, C is outside the range of A and is not aware of the transmission. Hence, C can unknowingly disrupt the communication from A to B. Different methods have been proposed to accommodate this problem. In 802.11 a random access scheme called Distributed Coordination Function (DCF) is used [IEEE, 2003], nonetheless, research indicates that this scheme is inefficient in multi-hop MANETs [Alex Varshavsky, 2004]. Furthermore, DCF introduces the *exposed* terminal problem, which is caused by the Request To Send (RTS) and Clear To Send (CTS) of the DCF. Here a node is prevented from transmitting due to the transmission of a nearby node e.g. In Figure 3.1(b), A sends an RTS to B. B does not reply with a CTS, since D in the meantime has sent an RTS to C and C has replied with a CTS, which has triggered B's random backoff mechanism entailing that B does not send a CTS to A. This leaves A with the wrong impression that B is unable to receive the transmission. Consequently the transmission from A to B is delayed unnecessarily. The terminal problems mentioned will be present in both the urban and freeway environment.

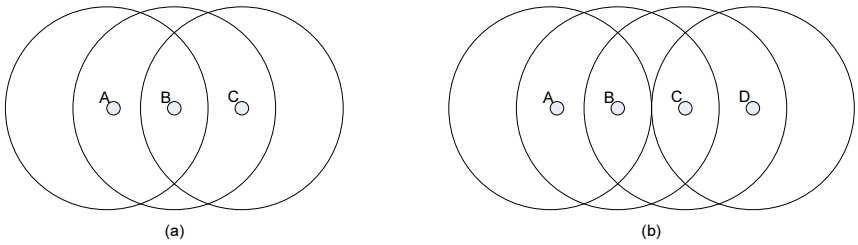

Figure 3.1: (a) the hidden terminal problem, (b) the exposed terminal problem.

Dynamic topology
The mobility of the MANETs nodes affect the network layer and consequently has a significant impact on the network topology which is highly dynamic. One of the main concerns is to guarantee a connection between sender and receiver and to keep

factors such as delay low enough to meet the requirements of the applications. In this context routing is of particular importance, since it should maintain the connection between sender and receiver and guarantee that delay requirements are not violated. A range of different routing techniques have been proposed for MANET routing [Johnson and Maltz, 1996] [Perkins and Royer, 1999] [Perkins and Bhagwat, 1994], and should be subject for further investigation if aspects relating to Quality of Service (QoS) and application requirements are to be investigated.

Security
In wireless networks security is important, since the wireless medium is easily accessible, and spans multiple layers. Hence, methods are needed to ensure that the end users can use the services accessible through the network at all times and without compromising data integrity and confidentiality. Hence, three requirements must be met to ensure the security of the communication of the end user:

- Integrity: To ensure that the transmitted data is not altered on the way to the receiver.
- Confidentiality: To ensure that the transmitted data is only read be sender and receiver.
- Availability: To ensure that services are available when needed.

Different methods have been developed to address the security aspects mentioned above.

Power consumption
One way to increase the security is to lower the transmitting power of the antenna decreasing the range of the wireless signal. This also has the benefit of lowering the power consumption of the wireless terminal. Power consumption can be an issue for small hand held terminals, since they have little battery capacity, making it necessary to save power in any way possible. Power consumption is not likely to be an issue for car-to-car communication, since cars carry a large battery capacity and power is generated whenever the engine is running.

Limited bandwidth
All the characteristics and issues mentioned can affect the bandwidth and in most cases limit it. Leaving the wireless network with a relatively limited bandwidth depending on how pronounced the issues and characteristics are.

3.1.3 Mobility modeling

As described in the previous section MANET topologies are highly dynamic due to the mobility of the individual nodes. The following investigates the mobility patterns which can be expected in the scenario and relates these to a range of mobility models. The purpose is to indicate which models might be suited to simulate the movement of nodes in a MANET.

Based on the properties of the urban and freeway environment described in Section 3.1.1 on page 15, the characteristics of the environments to be modelled is determined in order to establish grounds for evaluating the individual mobility models.

	Deterministic	Hybrid	Random
Macroscopic			
Microscopic		X	

Table 3.1: Table showing the mobility model classification, including the class to be used.

Urban: The urban environment is characterized by; a large number of obstacles in the form of buildings, stop and go traffic caused by traffic lights, a high traffic density, low velocities caused by traffic regulations and few lanes.

Freeway: In the freeway environment there are; few obstacles, the traffic flow is smooth, there is a medium traffic density, the velocity is relatively high and the number of lanes is large with respect to the urban environment.

The sought mobility model should be able to reflect the characteristics described above. Beyond this, physical restrictions imposed on the cars by temporal dependence of the velocity and the presence of other cars should be included. To do this the factors resulting in the characteristics described should be modelled i.e. the model should cover the following:

- Road capacity: The model shall take the number of lanes for the roads into account.
- Traffic density: It shall be possible to vary the number of nodes used in a scenario, thereby changing the traffic density.
- Traffic regulations: Speed limits shall be taken into account, hence, it should be possible to specify a maximum velocity for each length of road.
- Traffic lights: These play an important role in the mobility patterns of the urban environment i.e. the model shall include some representation of the effects of traffic lights.
- Temporal velocity dependence: The dependence between a cars previous and current velocity shall be included in order to make the cars accelerate and thereby their movement more realistic.
- Restrictions caused by other cars: The restrictions imposed on the individual car by traffic shall be modelled, entailing that cars must adjust their velocity to the car in front if it is not possible to overtake.

The concept of mobility modelling has been a subject of extensive research due to it's applicability in the development of cellular networks and WLANs and the deployment of road infrastructure, and thus a large variety of mobility models exist. The existing models can be subdivided based on the mobility parameters observed and by the degree of randomness in the mobility models (see Table 3.1).

Macroscopic mobility models are used to model mobility at a high level of abstraction e.g. fluid dynamics can be used to describe the traffic flow, hence, the individual node is not observed but an aggregate of several nodes. Conversely, microscopic mobility model each node separately. For both levels of mobility, the degree of randomness

can be categorized as; random, hybrid and deterministic. Deterministic models can e.g. be based on a trace from a real system i.e. the node movement is bound to predefined paths from the trace. In random models, the nodes can move in any direction within a bounded area. The intermediate step between deterministic and random is defined as a hybrid approach. Here each node moves in a random manner but the movement is restricted by some notion of infrastructure e.g. the area where the nodes may move is a map with roads and the nodes may only travel from one location to another using the roads.

The latter approach is the most relevant for modelling mobility in MANETs, since it incorporates the restrictions imposed on the nodes by the road infrastructure providing more realistic mobility patterns.

Beyond reflecting the mobility patterns realistically, the mobility model should also make it possible to reflect the relevant physical properties described in 3.1.1. Hence, metrics such as velocity and position for each individual node, should be modelled. This makes it possible to delimit the scope of the analysis to microscopic mobility models, since these reflect the movement of each individual node.

Conclusively the sought mobility model shall model each individual node and include maps reflecting the road infrastructure i.e. microscopic hybrid mobility models.

The following provides an overview of selected mobility models, that can be used to represent the movement patterns of nodes in MANETs.

Manhattan mobility models

[Bai et al., 2003] proposes a probabilistic map based approach to modeling mobility in MANETs, called the Manhattan mobility model. Here the map consists of a number of horizontal and vertical two-way streets restricting the nodes movement. When a node reaches an intersection the probability of turning left is 0.25, the probability of turning right is 0.25 and the probability of continuing in the same direction is 0.5. Nodes are bound to the lane they are moving on, hence, they cannot overtake. The velocity of each node is determined based on Formula (3.1) i.e. the velocity of each node varies with time but is dependent on the velocity if the previous time step. Nodes travelling in the same direction on the same street will adjust their velocity to the node ahead in order to maintain a predefined distance since they are unable overtake.

$$\left|\overline{V}_i(t+1)\right| = \left|\overline{V}_i(t)\right| + random() * |\overline{a}_i(t)| \tag{3.1}$$

The Manhattan model is relatively simple and incorporates the restrictions imposed on the nodes by road infrastructure. However, the simplicity of the model indicates that it is unable to simulate realistic movement, since the dynamics caused by e.g. traffic regulations, traffic lights and multiple lanes are omitted. This makes the model unsuitable to simulate movement in the urban environment. Due to the composition of the map used in the model, which resembles that of a city or urban environment, it is also not suitable for the freeway environment. [Bai et al., 2003] also proposes a model for a freeway environment where the city like map is replaced by one similar to a freeway. The nodes are still restricted to one lane and the dynamics caused by traffic regulations, traffic lights and multiple lanes are still disregarded. Though inadequate for modelling mobility in the final emulator, the Manhattan mobility models might be useful in early versions, since they are simple and easy to implement.

Graph based mobility models

Another approach that uses maps is the graph based mobility model presented in [Tian et al., 2002a]. This model is based on a graph, representing the restrictions imposed by the infrastructure, where the vertices are locations to which the nodes can travel and the edges are the roads they can travel along. The graph is fully connected meaning that there is a path to from any one vertex to all other vertices in the graph. The movement in this model is similar to that of the random walk, since each node is initialized at a random vertex and then randomly chooses another vertex to travel to. The shortest path is always selected. When a node reaches it's destination, it will wait a random time interval between t_{staymin} and t_{staymax} before choosing a new destination. The velocity of the node is randomly chosen for each destination between v_{min} and v_{max}.

This approach was developed to simulate pedestrians moving in some infrastructure, thus there are no restrictions on the capacity of each edge i.e. it is similar to a road with an infinite amount of lanes. Furthermore, traffic regulations are disregarded and nodes move at constant velocities.

Several approaches have been developed in order to increase the realism of the mobility models. [Choffnes and Bustamante, 2005] suggests an approach, called STreet RAndom Waypoint (STRAW), where the graph is based on real map data and traffic regulations are taken into account. Here the nodes can still travel from a location to some destination but speed limits, effects of other cars, traffic lights and traffic regulations are imposed on each node. This is also the case for the city section mobility model [Davies, 2000]. Both approaches lack the possibility for a node to overtake. An equivalent approach which incorporates the possibility of overtaking can be found in [Gorgorin, 2006].

None of the investigated approaches fulfil all the requirements derived at the beginning of this section alone. However, by combining their properties the requirements are met and yield the following approach, which is defined as a starting point for modelling mobility in the scenario, where:

- Road infrastructure is represented by vertices and edges.
- Vertices are locations represented by Cartesian coordinates.
- Edges are characterized by: Capacity (specified in lanes), length and speed limits.
- Nodes are represented by: Position, direction and velocity.
- Nodes obey speed limits with some variation.
- Nodes can overtake if there is sufficient space to do so.
- Nodes are initialized at randomly selected vertices.
- Nodes select destinations randomly and always the shortest path from origin to destination.
- Nodes wait a random time at each destination before selecting a new destination.
- In the urban environment traffic lights are added to all four way intersections.

This approach is treated further in the mobility model design of Section 6.3.2 on page 49.

3.1.4 Summary

This section has outlined the issues that can arise in MANETs in the urban and freeway environment. It was found that a broad range of physical characteristics of the wireless channel were present in both environments, however, some were more relevant depending on the environment e.g. multipath fading was more pronounced in the urban environment, whereas the Doppler effect only was relevant for the freeway environment.

Furthermore, microscopic hybrid mobility models suitable for both environments were investigated and a rudimentary graph based mobility model was defined as a starting point based on research performed in the field of mobility modelling.

3.2 Dependable services

In this section an analysis of requirements to provisioning of dependable applications is presented, including the underlying services needed to support dependable applications. The general concepts of dependability are examined. The dependability descriptions are derived from the HIDENETS framework in [HIDENETS, 2006b] combined with [Avizienis et al., 2004], which discusses dependability terminology. Moving from general to specific, this section presents the services needed in the scenario and their required dependability properties. The purpose of this investigation is to clarify the functionality and dependability requirements of each service necessary for the dependable applications.

3.2.1 Definition of dependability

Dependability of a service means that is can justifiably be trusted. To analyze what makes a service trustworthy, the dependability domain is divided into three major parts; *threats* towards successful service delivery, the *attributes* of the service success and *means* to prevent service failure. It is based on these parts that the individual services are evaluated in the further analysis.

Threats: Threats to service dependability are the faults that inevitably will occur and challenge the success of the service. A service *failure* occurs when the service does not live up to its specification or if the specification does not fulfil the functional requirements. Any such deviation from correct service is called an *error* and the supposed cause of the error is referred to as the *fault*.

Attributes: Several dependability attributes are assigned to a service when evaluated. The most important attributes are; *availability*, *reliability*, *safety*, *integrity* and *maintainability*. Combining some of these can create new ones such as the combination of availability and integrity, including *confidentiality*, results in the attribute *security* which often also is considered. For detailed definitions of the attributes, [HIDENETS, 2006b] and [Avizienis et al., 2004] should be consulted. Typically relative values are assigned the attributes for evaluation. However, due to the inevitable occurrence of faults in any service, the attributes should be interpreted in a probabilistic sense.

Means: Developing dependability is improving the dependability attributes of a service. For this several means are used: *Fault prevention*, *fault removal*, *fault forecasting* and *fault tolerance*. Prevention and removal are primarily concerns

of the development phase, ensuring the quality of the service specification and design. Forecasting is a means of a running service to avoid failing prior to a forecasted fault, typically based on probabilities of fault occurrence. Tolerance of a fault is the detection of and recovery from a fault occurring in the system.

3.2.2 Dependable services in the scenario

The applications described in the scenario all contain both middleware and communication layer services (cf. Figure 1.1 on page 7) which must be dependable for the applications to be trustworthy enough to be used. Dependable communication services are considered mandatory for the middleware services to provide dependability. Thus, the primary focus is to investigate the communication layer services, i.e. layer 2-4. The middleware layer service are referenced, when directly related to functionality of the communication services.

The domain of the services is delimited to concern the functionalities relevant to ad hoc communication. As the infrastructure domain is needed for the applications to exist, gateways between ad hoc and infrastructure domains are considered the boundary of the ad hoc domain.

The services are described in the same order as the service architecture of a communicating model is described. This way the services handling a shared wireless medium, the ability to do networking, transportation of data, session control, and cross-layer monitoring all are considered.

The services are common to all the scenario applications. It is the requirements to performance and dependability attributes of the services that differ in the application. Thus, the requirements are different depending on the applications considered in a scenario. Therefore, the services are described as general means for communication in the scenario, not regarding the requirements from any application at this point.

Shared wireless medium

Different radio technologies are used throughout the scenario in the ad hoc domain. WLAN, GSM and UMTS are mentioned as possibly available media, possibly in simultaneous combination. The existence of any available and reliable radio channel is mandatory for all the applications to succeed. Accessing and using these different media requires dependable servicing at the data link layer. Today, some dependability mechanisms exist at layer 2 in the form of fault tolerance. For instance, if frames are lost due to collision on the channel, the collision is detected and the frame is retransmitted. However, losing a radio channel is not tolerated at layer 2 today. Hence, this is a threat to the dependability of the link service.

To be able to benefit dependability-wise from possibly having several radios available, mechanisms for reliably utilizing access to multiple radios are needed. Moreover, several issues regarding connectivity exists within the scope of a single radio domain, previously described in the ad hoc environment analysis. Such dynamically changing environmental influences make the communication channel unreliable, posing more threats to the service dependability. For any networking services to rely on the channel in a dependable manner, these issues need to be handled.

Networking

For end-to-end communication with one end in the ad hoc domain dependable networking services such as routing, topology control, addressing and gateway management are

needed.

As the networking performed in the scenario is based on IP, generally applied routing algorithms such as Routing Information Protocol (RIP) [Malkin, 1994] or Open Shortest Path First (OSPF) [Moy, 1994] could be used. However, due to dynamic link changes, traditional routing algorithms for wired networks do not apply very well in the ad hoc domain [Jacquet and Laouiti, 1999]. Instead, routing algorithms for ad hoc networks are applied.

Ad hoc routing algorithms are divided into two major groups; proactive and reactive. Essentially, proactive routing algorithms maintain information for routing continuously no matter if there is traffic, whereas reactive routing algorithms find appropriate routes on-demand. Maintaining routes requires more control signals than rediscovering routes, which in turn takes longer time [Broch et al., 1998].

No matter the group, common steps for performing ad hoc routing exist. These are *neighbor discovery*, *information dissemination* - i.e. exchange of control messages, *shortest path calculation*. These steps need to be performed correctly in order for a path to be discovered successfully.

Control messaging on the communication network incorporates a problem of differentiation of traffic as control messages are mandatory to support the dynamics of the topology. However, heavy control communication between routers may overload the already limited bandwidth. Running bandwidth-sensitive applications may render such routing algorithms unusable. Contrary, time-sensitive applications may benefit from the instantly available, low-latency routes. In all, the control messages are evident for performing topology control. Failure of a routing algorithm means inability to maintain an end-to-end connection defined by performance requirements. Faults leading to such a failure are breakdown of links or routing loops which must be tolerated for any routing algorithm to be used by a dependable application.

Topology control emerges from the problem of maintaining a topology among routing nodes. With the increasing range of the radios, many nodes are potentially available for connection. This may not, however, be optimal in the case of moving cars. Position, speed and direction of the node can be used to help determine which nodes are fit for participating in the topology and perform topology control.

Maintaining address spaces of network nodes is also complicated in the ad hoc domain, especially on the border-line between ad hoc and infrastructure domain. With traditional IP, the static nature of the nodes in a topology makes static addressing possible, i.e. once a node is assigned an address it does not change. However, with the inherent node mobility in the ad hoc environment, static addressing is intractable to manage. Instead, methods such as mobile IP [Perkins, 2002] can be used. Mobile IP uses agents for maintaining references between the actual address of the node and the address known to the public. This enables continuous addressing of mobile nodes. However, this also poses as a threat to the pure ad hoc communication, if nodes are subject to change of address while being used.

In the same context, the aspect of *gateway maintenance* must also be considered, as loss of gateway is another existing threat to networking between ad hoc and infrastructure domains.

Gateways are used when connectivity from the ad hoc domain to an infrastructure domain is needed - as for instance it is the case with the video conference application. Here one peer is placed in the ad hoc domain and the other in the infrastructure domain. For peer-to-peer communication, gateways between the two domains must be accessible at all time. Such gateways can either be mobile or have fixed placements. Gateways based on UMTS are believed to be mobile as these are part of the moving

cars. WLAN access points are placed accessible from the road in cafees, gas stations or phone booths, working as inter-domain gateways. To ensure access to a gateway is a challenge, as a changing topology dynamically changes the connections in the network. Partitioning of the network may result in losing the connection to the current gateway. To remain connected, selecting a new gateway and establishing new routes are required tasks. Depending on the changes in the topology such tasks may be non-trivial.

Data transport

For transportation of data on layer 4, not many new protocols exist that are optimized for the ad hoc domain. Instead, attempts have been made to optimize the existing protocols for wired network, e.g. TCP and UDP, in order to make them perform satisfactory in the ad hoc domain. These are relevant considerations due to the diversity of the applications existing in the scenario. A video conferencing application traditionally uses UDP for streaming, while an application for interactive data would use TCP to ensure integrity of the transferred data. Optimization of these algorithms is needed as, for instance, the back-off algorithm of TCP is very unfit for considering packet loss due to frame loss induced by poor link conditions [Tanenbaum, 2002], which results in an unnecessary reduction of throughput. Throughput is the important parameter in the context of data transport and is required to behave reliably.
Failure of transport is typically not meeting application performance requirements such as throughput, delay and delay jitter. Faults leading to this are for instance packet loss, either due to poor or lost wireless connections or congestion or packet reordering in the network, or as described, the need for re-routing due to breakage of the route.

For the resilience aspects of transportation and data integrity, solutions such as SCTP [Stewart and Metz, 2001], have emerged. SCTP extends TCP and UDP by facilitating multi-stream *messaging*. *Messaging* means that instead of requiring each byte to be received in the right order as TCP does, SCTP uses *messages*, spanning more bytes. This enables an optional degradation of the ordering in which bytes are received making the protocol more resilient towards packet loss. Multi-stream means, that SCTP can handle multiple streams within the same peer association. This makes it possible to use different physical connections, e.g. both UMTS and WLAN, for increased availability and reliability. These properties of SCTP help enable resilient transportation solutions.

Fault detection and performance monitoring

To aid the diagnosis of a running system, services providing fault detection mechanisms are included in the service architecture. Such services may use cross-layer monitoring or work inside a specific layer diagnosing faults. Increasing the possibility for successful detection and diagnosis of faults in the system increases the reliability of the communication. This can be achieved it several ways - either using dedicated fault alarms generated in the system or performance measurements revealing abnormalities in the communication patterns. These measurements could be packet loss rate or wireless signal quality or a combination. Such measurements can be performed both actively, i.e. sending dedicated traffic as measuring probes, or passively, i.e. where the existing traffic is used for probing.

For the communication services examples of performance metrics at the link layer are capacity, frame loss and frame delay. At the network layer examples are throughput, packet loss and delay. At the transport layer, reordering of packets can be used.

For such services to raise the resilience of the system, the services themselves need to be highly tolerant of internal faults. An erroneous fault detection service may pro-

duce false alarms or miss occurring faults, which poses a threat to the dependability of services relying on the fault detection.

Lower level service middleware
On the borderline of communication services towards middleware layer services, there might exist some services managing aggregated node information for the middleware components. Such information could be profiles of power supply, bandwidth and media types. Also, the network task of *gateway and network selection* might be influenced from this level, e.g. decisions on base-station or network-technology may be controlled from this level. To do this, dependable information regarding the network(s) must be gathered from or made available to this level.

A QoS and differentiation manager can also be present in this layer. This manager handles differentiation based on QoS policies defined for different classes of both traffic and resilience. Resilience differentiation is providing different resilience support to traffic streams during faults. For instance, a voice call may be closed down after fail-over as a consequence of a web session having higher resilience requirements and hence is prioritized.

Finally, at this border-line middleware layer, session control can be performed. Session control tasks include establishing, maintaining and terminating session between multiple participating nodes. Session Initiation Protocol (SIP) [IETF, 2000] is among the methods to do session control.

Although these service are not purely communication service, they are described here as failure of these services might be induced as faults in the underlying services.

3.2.3 Summary

This analysis has investigated the aspects of dependable services by identifying the main components of dependability and applying them to the functional requirements of the services in the scenario. Regarding the applications derived from the scenario, i.e. video conference, streaming data and interactive data, the different supporting network services have been investigated with respect to dependability. In this investigation, *threats* to the dependability, *properties* of dependability and *means* for overcoming the threats have been identified for each service layer. Analyzing the threats to the dependability, the dynamics of the environment play an important role in the fault domain, mainly because it influences on all levels of the communication stack, from link to network. Also, it is revealed that although dependability means exist on several layers, it is far from every threat that can be overcome. Generally, the analysis reveals that dependability issues exist on each layer which must be faced before an entire dependable solution can be realized.

3.3 Evaluation methods of dependable solutions

Having analyzed the ad hoc environment and the functional requirements of the services needed in the scenario, this section investigates different methods of evaluating dependable services given the different operating contexts. Depending on the functionality subject of evaluation, several evaluation methods can be applied. In the context of mobile ad hoc networks, literature provides numerous descriptions and experiences on such evaluation methods. In the following, an analysis of evaluation methods is

presented. The purpose of this analysis is to enable selection of a subset of methods for later evaluation of a dependable solution. The general approaches to evaluation are *analytic*, *simulation*, *emulation* and finally *experimental setup*.

3.3.1 Analytic

The analytic approach entails development of models for all aspects of the problem domain. Complete analytical evaluations have been performed on e.g. link properties [Rajarman, 2002], network properties [Rajarman, 2002, Tian et al., 2002b] and routing performance [Krishnamachari et al., 2000] [Qayyum et al., 2000]. Overall these approaches have a specific functionality or context in focus. This means that the level of detail in the specific models of interest is very high, but the surrounding models may be very simple. For instance, including complex models of link properties when modelling network specific properties can make analytic solutions intractable. In the context of evaluating mobile ad hoc networks models, complex link properties as well as complex network properties may all have to be included to represent a realistic domain, i.e. generate a usable model. Inclusion of that many details, along with the fact that the ad hoc network in the car-to-car scenario tends to scale, e.g. in the number of partitions, nodes or connections, can render solutions hard to derive, if not numerically unsolvable [Qayyum et al., 2000, Krishnamachari et al., 2000].

3.3.2 Simulation

One way to deal with the intractability of theoretical models are to use computers to simulate them numerically. In general, simulation is a widely used evaluation method in the mobile ad hoc network community [Heidemann et al., 2001, Short et al., 1995]. Simulation permits as much detail as the combination of processing power and time permits. When developing simulation models, reuse of general models entails continuous development of the models, enhancing the model realism. In addition, simulation can be carried out in non-real-time. This allows for slowing down the simulation to monitor the steps of the process.

However, no reuse of real implemented functionality is possible in a simulation environment. This means, that all protocols and functions need to be re-implemented in order for them to be used during simulation. This can result in use of simplified models based on assumptions of functionalities not directly relevant of the evaluation. For instance, a wireless MAC protocol model might be replaced with an existing wired MAC protocol model, as the contribution from the wireless properties might be limited and out of scope. Indeed, this reduces the effort of re-implementation, but also decreases the accuracy of the simulation.
Also here, the previous described dynamics of the ad hoc environment requires the models to be realistic. As the application dependability properties are dependent on the dependability of each of the underlying functionalities, this entail an inclusion of potentially many, realistic simulation-implementations for successful evaluation.

The most popular tools used for performing simulations are ns-2
[McCanne and Floyd, 1995] and GloMoSim [Takai and Bajaj, 1999]. Both simulators use scripting languages to define the specific simulations. The features of the simulators are generally alike, and they both provide many detailed tools for evaluating ad hoc networks [Flynn et al., 2002]. Moreover, detailed models for popular technologies, e.g. WLAN 802.11, have been thoroughly tested and is widely used in the simulation

environments. This means, that simulation of some off-the-shelf technologies does not require re-implementation.

3.3.3 Emulation

Emulation of functionality is a hybrid between simulation and pure experimental setup, working as a means for reducing the effort of re-implementation of simulation. Allowing for some components of the real system to execute, an emulation environment facilitates the control and setting of some underlying parameters or functionality. For instance, the properties of a physical layer can be emulated using Field Programmable Gate Arrays (FPGAs) [Judd and Steenkiste, 2004]. The FPGAs are software configurable logic capable of emulating hardware, in this context wireless radio properties. MAC properties can be emulated to evaluate network properties evaluation [Zheng and Ni, 2002][Flynn et al., 2002]. Network properties can be emulated for higher layer functionality evaluation [Vahdat et al., 2002, Ke et al., 2000]. In all of these evaluations, traffic is generated by real systems. This way, complex traffic modelling and cumbersome component re-implementation is avoided and the focus can remain on the performance evaluation regarding the underlying parameters. However, in the case of having real systems as a part of the evaluation framework, real time requirements emerge. Combined with complex models to be emulated, it can be difficult to accomplish emulation in real-time. Also in this case, simplifications can be introduced to relieve the need of processing capability of the emulator, which in turn increases the usability of it.

3.3.4 Experimental setup

The experimental setup is presented as the final evaluation method. An experimental setup contains all the parts of the real system, possibly equipped with probing facilities to measure metrics for performance evaluation. Tests performed using experimental setups often provide the evaluation with a great level of detail and realism. However, in the context of experimental mobile ad hoc networks, the greatest challenge is the assurance of repeatability [Maltz et al., 1999]. Due to the complexity and diversity of the wireless environments combined it is hard to ensure execution under similar conditions for each test run. Also, when regarding mobility, the accuracy of the motion patterns is critical. Moreover, the need for personnel and hardware to setup up the complete experimental testbed might be impossible to acquire due to costs.

3.3.5 Summary

In this section different methods have been presented for evaluating applications running in the mobile ad hoc environment. The methods span between the two pure setups; pure analytic and pure experimental setup. In between, the simulation and emulation method have also been investigated. Overall, each method has pros and cons, depending on the context for it to be used in. Issues such as *computational complexity*, *implementation redundancy* and *repeatability* are all parameters of each method. Depending on the size and complexity of the evaluated solution, any of the methods can be applied. However, the size and complexity of the domain show to be proportional to the complexity of the evaluation methods meaning that a large and complex system is hard to evaluate using the most complex methods, which is why simplifications often are incorporated. Analytic models may be tied to evaluation of one specific solution

whereas a generic emulator or formal descriptions of an experimental setup may be used for several evaluations.

Chapter 4

Problem statement

The problem statement constitutes the conclusion of the preanalysis. In the following the main topics from the analysis are summarized. In addition, the main problem to be considered in this project is detailed. Moreover, the objectives in dealing with the main problem are listed and discussed along with a delimitation from topics which are out of scope of this project. The chapter is concluded by a brief description of the content of the remainder of the project.

4.1 Problem description

As wireless equipment has become commercial off-the shelf products, the domain in which it can be used is expanding. One of the new areas of wireless applications is car-to-car communication. Applications in this area can make great use of the mobility the wireless environment provides. However, many applications are mission critical meaning that a failure in the performance of the application may have catastrophic results. To prevent such failures, dependable services must be deployed to support the applications in becoming resilient. For an application to be dependable means to have a high level of availability, reliability, integrity, security and maintainability.
To meet requirements to these properties fault tolerance needs to be present in the services.

A scenario has been defined to investigate what threatens resilient applications to be used in car-to-car communication. The scenario contains a dynamic environment and several applications that must be highly dependable. The application properties are influenced and may be threatened by the properties of the environment and the services supporting it. An analysis of the influencing factors has been performed. Moreover, to ensure dependable solutions the development must employ thorough testing. Testing of dependable application can be carried out by several methods, all having pros and cons, which have been analyzed.

Environment influences
The influences from the environment are many and diverse. The analysis reveals influence on many of the layers needed for car-to-car communication. Properties such as *path-loss, non-isotropic radio propagation, shadowing, multipath fading* and the *Doppler effect* are present due to the wireless medium. Individually and in combination these make any communication channel based on a wireless medium suffer from

unreliable connections on the link layer. Also, the inherent mobility of the nodes make link connections highly unstable.
Node mobility combined with unreliable connections introduces a **limited bandwidth**, as an increasing part of the bandwidth is used for control messages to manage the network. When these properties propagate into the higher layers they result in issues such as a **dynamic topology**. An outside issue with the mobile entities is their limited power capacity. However, when considering cars as nodes, this can be neglected.

Different services

The services to be deployed in order for the applications to run are all affected by the properties of the environment. One common factor for the applications of the scenario is the communication channel that needs to be highly resilient towards the dynamic environment. The different services of the communication channel and the influences of the environment on these services have been analyzed. The analysis shows that although some level of fault tolerance is implemented on different layers, complete dependable solutions are hard to accomplish. The different layer services such as *medium access*, *topology control*, *network routing* and *data transport* all have some fault tolerance incorporated. Retransmissions occur when frames or packets are lost. New routes are discovered if the peer to peer path is lost. Yet, the high level of dynamics that reside in the environment still challenges the dependability of the services. Loss of frames is very frequent which also results in packet loss. When packets are lost, the throughput is degraded which can result in failure towards the performance specifications of the applications. Re-discovery of routes may increase the delay of the channel. In the end, these faults decrease the dependability of the application. To aid the fault tolerance mechanisms of the different layers in meeting the requirements, specific services such as multi-layer fault tolerance and performance monitoring should be applied.

Evaluation framework

The described properties of the environment and the services imply thorough testing of the solutions implementing the services. It has been analyzed which challenges that need to be overcome for a test-environment to be appropriate for evaluating dependable solutions. Analytic methods are very accurate on a small scale but can become very complex and intractable when scaled in parameters, which is often the case in ad hoc networks.
Instead simulation can be used, which provides a non-real-time test environment with much computational power for numerically solving complex models. However, here much implementation effort is needed, as the algorithms and mechanisms to be evaluated need to be re-implemented in the simulation environment.
To overcome this, an emulation environment can be used. Here, real implementations of the services and applications can be run in a controlled environment. This, however, requires the emulation setup to be able to run transparently to the services. One issue here is the real-time execution of the application which requires the emulator to calculate the emulated properties in real-time.
The final setup investigated is the experimental setup which provides the most realistic results. Here, an issue of repeatability arises, as external influence might be induced while testing, rendering a repetition of the experiment impossible. Also, the cost and effort to set up an experimental framework may exceed the benefit from the evaluation.

4.2 Main problem

A need for communication services exists in the ad hoc domain. Basically, this requires successful link connection, routing and transport of data. Successive dependence between these functionalities exists in the form that successful transport requires successful routing which in turn requires successful link connection. Improving dependability in these functionalities improves resilience in the overall application.
Ideally, fault tolerance prevents errors from propagating up the layers. However, this is not always the case. Optimal fault tolerance on the link layer, for instance, is still incapable of tolerating movement of nodes and as a consequence links still disconnect. This means that optimizing link level dependability can only increase the overall resilience in a limited way. The mobility issue must be handled at the routing layer. Succeeding in this means that an end-to-end connection is always present, with the term 'always' interpreted in a probabilistic manner. For streaming applications such as video conferencing this is sufficient for dependable communication and no additional fault tolerance is applied by the transport layer. As with the link layer, this means that optimization of routing dependability properties reduces dependability means on the upper layers. Moreover, as the routing layer must tolerate link disconnections, it will also handle non-optimal link fault tolerance, i.e. that the routing is capable of handling general link faults by abstracting them to connectivity-faults. This consideration emphasizes that optimizing routing would help compensate for the missing dependability of the underlying layers of the communication. Hence, a fundamental need for dependable ad hoc routing exists to support the dependable applications described in the scenario.

As identified in the analysis several algorithms exist today which provide ad hoc routing. However, to be used in the car-to-car context, a high level of availability and reliability need to be achieved in the algorithms. It will be investigated how these existing algorithms perform in a highly dynamic ad hoc setup in terms of availability and reliability. For evaluation of the routing algorithms, appropriate metrics need to be identified. The context of the evaluation is based on the applications described in the scenario. In segment 6 of the scenario, the UMTS gateway is not available by single-hop. Thus a gateway must be accessed via WLAN multi-hop to prevent application failure. This means that one peer is the application residing on a mobile node, and the other is the gateway either residing on another node or has a fixed location somewhere near the road.

To do proper evaluation, a testbed must be designed and set up. Experience shows that both the analytic and the simulation approach are subject to simplifications in the modelling. Moreover, the simulation requires the algorithms to be re-implemented in the simulation environment.
Despite the high level of realism in the experimental setup, the lack of reproducibility prevents thorough evaluation of the algorithms. A such framework should be used for field test and end-user testing, where it is more qualitative terms than quantitative terms in the evaluation. Finally, the emulation testbed remains. Being a hybrid, this facilitates deployment of complex signal propagation and mobility models as well as realistic application behaviour and performance requirements employing realistic traffic patterns in the emulated network. Utilizing these properties, a topology emulator is developed for evaluation testbed in this project.

As several routing algorithms exist, a generic framework for equal evaluation must

be developed, and an emulator provides this without requiring re-implementation of all the tested algorithms. For future use, the emulation testbed can be applied to applications running on top of the routing algorithms. This way the nodes can be subjected to a realistic, yet controllable, mobile ad hoc environment. A conceptual illustration of the emulation testbed is depicted in Figure 4.1.

Based on this conclusion of the pre-analysis the main problem to be addressed is as follows:

Evaluation of existing ad-hoc routing protocols with respect to dependability using a topology emulation testbed.

4.3 Objectives

Two major objectives of this project to research the specified problem domain are; 1) specification and development of a topology emulation testbed and 2) definition of a routing evaluation framework along with conducting routing evaluation. Details of the two are listed in the following.

4.3.1 Emulation testbed

- Specify topology emulator requirements with respect to functionality and performance
- Develop the topology emulator in the context of the requirements
  - Define topology emulator architecture design for the required functionality
  - Define mobility model
  - Define link property models
    - Loss models
    - Delay models
  - Implement models and emulator components in the topology emulator
- Verify the topology emulator with respect to functionality and performance requirements

4.3.2 Routing evaluation framework

- Perform analysis of existing routing protocols for evaluation
- Identify routing evaluation metrics in terms of performance and dependability
- Derive requirements for routing evaluation results in terms of the metrics
- Perform evaluation of routing protocols with respect to performance and dependability

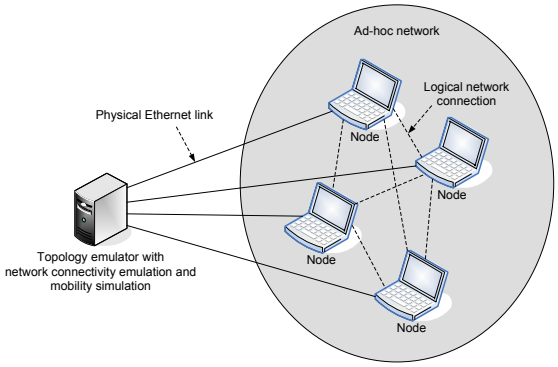

Figure 4.1: Conceptual illustration of the topology emulation testbed.

4.4 Delimitations

- **Only concern ad hoc node to gateway network environment**
  Topics such as gateway selection and topology control schemes are out the scope of routing evaluation in the ad hoc domain.

- **Only concern segment 6 of the scenario**
  Segment 6 provides a relatively dense environment with special focus on requiring multi-hop routing for the success of the dependable applications

- **Only availability and reliability of dependability**
  Safety, maintainability and security are secondary to the availability and reliability in terms of routing. They cannot be considered, if the routing is not available and reliable .

- **Only graph based mobility models**
  Out of the many mobility models researched in the pre-analysis, the graph based provide the most features in terms of the requirements set to a mobility model of the scenario.

4.5 Overall requirements to emulation testbed

This section describes the definition of the overall requirements to an emulation testbed as the concept depicted in Figure 4.1. In the remainder of this report, the following terms are used. The *testbed* is the entire emulation framework, incorporating *end-nodes* (laptops or PCs), and all equipment needed for the *topology emulator* to function correctly.

Scalable architecture: Several routing applications are needed to create experimentally interesting scenarios with multi-hop routing. As the testbed only emulates the topology of the wireless network and not application behavior, these application must exist and be connectable to the testbed. Such applications range from simple end-to-end *ping* operations to more complex distributed network applications dependent on routing, e.g. a distributed blackbox [HIDENETS, 2006b].

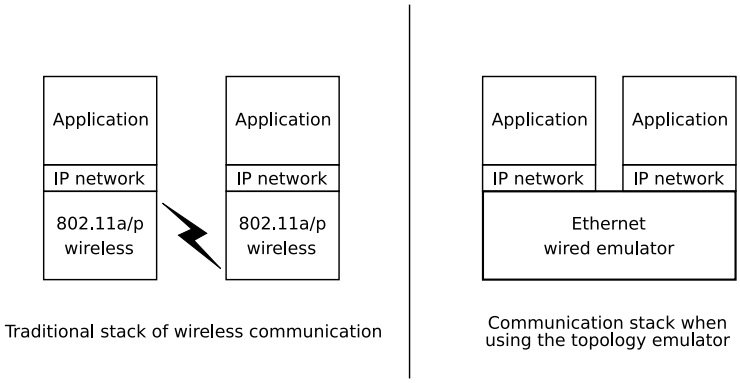

Figure 4.2: The concept of network emulation as seen from the communication stack. The figure illustrates what layers of the traditional wireless communication stack that are exchanged by the emulator for emulating a wireless network.

Typically, these reside on workstations or laptops equipped with wireless network adapters, while being developed. To facilitate testing of such applications it must be possible to connect these **end-nodes** to the emulator.

Transparent emulation: Facilitating the previous described scalability can be attained in several ways. One solution is to place the applications on the emulator by use of multiple virtual execution environments running the applications under test. Another would be to keep the applications on the end-nodes and attain emulation by redirecting all traffic through the emulator, for instance by replacing each wireless interface with a wire plugged into the emulator. Either way, the applications running on top of the emulated topology will not tolerate any deviance from running in a real environment. Any behavior change, for instance in medium access algorithms or from biased delays or bandwidth must be compensated for by the emulator, masking it from the applications under test. As illustrated in Figure 4.2, the emulator must transparently emulated the behavior of the link and physical layers.

Real-time emulation: Emulating topology for applications running in real-time requires the emulation to run in real-time. All packet handling and interface responses, e.g. link layer responses, must be carried out continuously and in proper time. This requirement derives strict deadlines for the emulation framework, and analysis of the range of time to work in must be performed to assess the possible facilities of the emulator.

Usability: To be able to use the system for emulation within a range of experiments, the system needs to be easily accessible and somewhat easily configurable. Basically this means, that changes needed to be performed on the end-nodes in order for the nodes to be connected/used with the emulator must be held at an absolute minimum. This way, even development machines for the end application and alike are more easily connected and the application area of the emulator is kept wide.

4.6 Outline

Having defined the objectives and delimitations of the work to be performed in this project, the report proceeds by describing the analysis performed to establish the requirements to the different aspects of the testbed. Subsequently the design and implementation which is derived based on these requirements, is covered. The resulting topology emulator implementation is hereafter evaluated, to determine it's applicability. Having established the capabilities of the testbed and thereby enabling the definition of scenarios which are supported, the methodology used, to define the metrics and the scenarios used for routing dependability evaluation, is described. This methodology is used to specify, execute and evaluate a number of experiments regarding routing dependability, which is described subsequent to the methodology. Finally, the achievements and the conclusions which may be drawn based on the work in the project are reported and potential future work is described.

Chapter 5

Testbed requirement analysis

This chapter presents an analysis of the requirements set forth to complete the objectives of this project. The requirements stated previously concern both the general evaluation of dependable routing and the specific performance requirements to the emulation testbed to be used for evaluation.

To be able to evaluate the dependability properties of the routing implementations, an emulation testbed is built. For this testbed to generate evaluation results, the testbed itself must be verified against the requirements set to it. In this section, an analysis of the requirements to the emulation testbed is described. Two major groups of requirements to the testbed exist (see Section 4.5); functional requirements and performance requirements. Functional requirements are requirements to *scalability*, *transparency* towards end nodes and *usability*. Performance requirements regard bounds on *packet delays*, *bandwidth reductions* and *packet losses*. As functionality and performance are combined, more requirements arise. For instance, requiring a scalable platform with support for multiple nodes calls for analysis of the performance behavior parametrized by the number of end nodes. Also, the traffic characteristics of several nodes may influence the performance of the testbed, and thus further substantiating functionality-dependent performance.

5.1 Functional requirements

The functional requirements cover architectural considerations necessary to develop a topology emulation testbed.

Transparency
As the testbed is to be used to emulate topologies in MANETs, network and link properties is emulated in the system. In a networking system, packets are passed on from the network layer to the data link layer on a sending end node. Then the packets are transmitted through a network link. On the receiving end node, the packets are delivered from the data link layer to the network layer. The emulator emulates this link transmission of the packets through and must thus comply with the interfaces specified between the data link layer and the network layer. This means that below the network layer, implementation changes facilitating emulation are tolerated, but on and above the network layer, deviances from ”normal” operation, i.e. when traffic is not subjected to

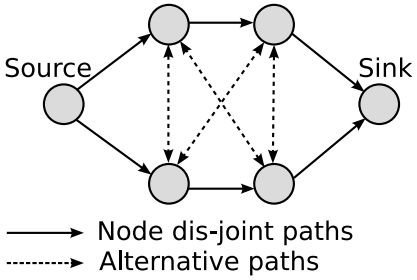

Figure 5.1: Example of topology generating interesting scenarios for routing evaluation. It must be possible to create several paths from source to sink (ideally node dis-joint) to evaluate routing in a dynamic environment.

emulation, are not tolerated.

Scalability

Scalability in the testbed covers several areas. First, the architecture must support connections of multiple nodes. To create interesting scenarios with regards to routing via separated routes, at least 6 nodes are deployed (cf. Figure 5.1), which must also be supported by the testbed. Second, scalability in the number of nodes results in a possibly scaled load on the entities in the system. In this sense, it must be ensured that processing power bottlenecks do not exist and that the system is capable of handling many nodes, possibly with different traffic characteristics.

Collision and broadcast domains

The spatial scale of the simulated environment may be too large for only one, connected radio domain to exist and independent partitions may be created. To emulate this, it must be taken into account that these domains are completely separate and independent, meaning that handling of parallelly received datagrams optimally must be performed on the same time.

Moreover, one characteristic of wireless networks is that everybody in a radio domain hear a wireless transmission, i.e. the nodes reside in a collision domain. In this domain, collisions of link layer datagrams, i.e. *frames*, may occur.

However, the emulator is to replace the layer 1 and layer 2 of communication, meaning that entire *packets* (and not frames) are received on the emulator from the end-nodes and packets are delivered from the emulator to the end-nodes. These packets potentially can consist of several packets, i.e. requiring several transmissions to be concluded. This means, that to be able to emulate a radio domain, prone to frame collisions, the emulator must be aware that nodes may send packets simultaneously and must be able to emulate drops of the packets due to frame losses, in turn, due to collisions in the radio domain. Also, as a consequence of the emulator working on packet level, frames cannot be broadcasted so all nodes in a radio domain can hear them. However, packet broadcast must still be supported, requiring the emulator to forward all broadcast packets to all nodes within a radio domain.

For the emulator to support multiple independent domains, it is necessary to separate packet broadcast domains. Without end nodes alteration, due to transparency, the emulator must be able to let a subset of all end nodes hear only a subset of all forwarded

packets.

5.2 Performance requirements

Besides providing functionality to enable satisfactory emulation, the performance of the topology emulation must also be verified. For instance, it must be verified that the delay imposed by the emulator is similar to that of a wireless channel. Also, the link layer bandwidth provided by the emulator must match that of the wireless channel. Optimally, all these parameters are modelled and simulated by the simulator only to be emulated by the emulator to ensure complete control and repeatability of the evaluation experiments. However, not all parameters are completely controllable from within the emulator. The simulations are all software-controlled, but to perform emulation, some hardware and operating system properties have influence. Hence, additional delay and bandwidth reductions must be accounted for in the testbed system as a whole to ensure satisfactory emulation.

Delay
Significant for the delay is that is must be similar to that of the wireless channel, thus the delay characteristics of a wireless channel must be established.

In the design chapter for the simulator, the models are described that are used for deriving the delay of the wireless channel. To ensure that the emulator is capable of emulating these delay, a comparison of the minimum possible wireless delay must be performed to the minimum attainable processing delay of the emulator.

Bandwidth
Regarding bandwidth, it is just as important to not reduce bandwidth as to imitate the delay. [Hadzi-Velkov and Spasenovski, 2003] considers bandwidth of the wireless channel, but unfortunately only for 802.11b, the 11Mbps version. The analysis shows a bandwidth-degradation of around 35% at frame transmission level in an *error-prone* channel in the best cases, calculated for 30 nodes. Comparing to experimental analyses, e.g. [Wijesinha et al., 2005] showing a reduction of 50% and increasing for 802.11g with DCF, justifies that not nearly 54 Mbps per link is needed from the emulating system. Using the lower value of 35% from [Hadzi-Velkov and Spasenovski, 2003] as reference for link bandwidth of the emulator seems adequate, also considering that the video stream applications from the scenario do not come anywhere near 35 Mbps ($54\ \text{Mbps} \cdot 65\%$).

Scalability
Depending on the number of processing power and units used in the emulator testbed, the number of connected end nodes will also have impact on the delay (and possibly bandwidth) performance. As for the functional requirements, the minimum number of nodes readily handled by the emulator is 6 nodes. The performance parameters are not tolerated to drop when using up to 6 nodes on the system, preferably more as the usability of the system for creating interesting scenarios is raised with more nodes.

Input traffic characteristics
As with the number of end nodes, also the traffic load generated by the end nodes has impact on the testbed performance. Traffic relayed in the system is inevitably queued

and it must be ensured, that the size and service of such queues are adjusted to handle different input traffic. Cases of different traffic patterns are Markov-Modulated Poisson process to generate bursty, in-dependent traffic. Furthermore, when considering the video-conference origin of the traffic in the scenario, more realistic models can be applied, e.g. model incorporating long range dependence.

5.3 Model requirements

The model requirements cover the aspects relevant to providing a model of the environment for the scenario. The model provides a transfer function converting the properties of the environment into network layer delay and packet loss. This entails that the main requirement to the model used in the testbed is to be capable of reflecting the different properties of the specified scenarios and their effect on packet loss and delay.
A scenario may be divided into the following parts:

Mobility model describes the node movement in the scenario. The mobility model is comprised of a model describing the movement of the individual nodes and a map restricting this movement.

Link technology specifies the technology used to provide link level communication, i.e. the physical and data link layer.

Channel model defines the physical conditions which the link technology is subject to, e.g. the background noise interfering with a wireless signal.

The model must be comprised of a number of sub-models, with generic interfaces, each representing a part of the scenario. Hereby a suite of models is provided which can be used to evaluate a range of scenarios, while the generic interfaces allows the replacement of the individual models in case e.g. another link technology is needed or another mobility model is needed. This increases the areas in which the testbed is applicable. The individual models has properties affecting the scenario, it should be possible to affect these properties through parameters in the models, enabling a thorough evaluation and increasing the usability of the testbed.

To further increase the usability of the testbed it should be possible to use trace files as input to the individual models. This contributes to the usability in two ways:

- Storing the results for the individual models in files makes it possible to replace the output of a model with that of another tool, e.g. a trace from an external mobility model, thereby saving resources spent on implementation.
- The files may also make it easier to debug applications after a scenario has been executed, since they contain a complete characterization of the properties of the scenario, making it easier to determine why an application failed.

The individual models used in the testbed should be as realistic as possible to provide an accurate representation of the properties of individual scenarios.

Chapter 6

Topology emulator design and implementation

Establishing the requirements to the testbed enables the subsequent design and implementation.

This chapter describes the topology emulator design including the development of the topology emulator architecture. The physical and logical network architecture and software components needed for the emulator are derived based on the requirements from previous chapters and selected parts of the implementation of the testbed are described. The purpose of this chapter is to provide an overview of the design and implementation of the testbed.

6.1 Network architecture

Given the functional requirements specified in section 5.1, a suitable physical and logical network architecture for the emulator can be developed. It is necessary to define the network technology to be used in the architecture. Emulating the properties of IEEE 802.11 requires that a medium is present on which desired properties can be imposed. This medium can be either virtual or physical depending on the implementation. Using a virtual medium entails that the emulation must be executed on a single machine, which must be very powerful if the architecture should be scalable. Hence, a physical medium is selected for the emulation. Since the testbed environment shall make it possible to reproduce test results, uncontrolled stochastic properties are undesirable in the physical architecture e.g. using a wireless channel for the physical links would have a significant negative impact on the reproducibility of test results. Consequently, a more reliable channel shall be used, for this purpose Fast Ethernet is selected, since it offers a relatively low Bit Error Rate (BER) and sufficient bandwidth to support emulation of 802.11.

6.1.1 Physical network architecture

Starting from the physical architecture the main concern is to ensure that the solution is scalable making it possible to test a wide variety of scenarios. For this purpose three different physical architectures are considered (see Figure 6.1):

Decentralized architecture: Figure 6.1a shows a decentralized architecture where the nodes are connected through a hub. This is a relatively scalable solution, since the only limitation on the number of nodes is the number of ports in the hub. However, when the nodes are connected directly through a hub, software must be present on each node which facilitates the emulation of the wireless channel and simulation of the node movement. This would require that the nodes have a common representation of the systems state e.g. node positions, scenario and ongoing transmissions, consequently adding control communication overhead to the network. Moreover, this solution conflicts with the requirement, that adding software to the nodes should be kept at a minimum.

Centralized architecture: The centralized approach (Figure 6.1b) connects all nodes through a emulator node which is responsible for the emulation and simulation. This removes the communication overhead mentioned above and increases the usability of the testbed, since no software installation is needed on the nodes to control the emulation and simulation. This approach is, however, suboptimal with respect to scalability since the emulator node must contain a Network Interface Card (NIC) for each node attached to the emulator.

Centralized architecture with switch: The last approach (Figure 6.1c) combines the two approaches mentioned above, replacing the hub of the decentralized approach with a switch, to provided a physical architecture which is both scalable and requires no alteration of the attached nodes. One drawback of this approach is that the traffic from all nodes is sent through one link between emulator node and switched, potentially creating a bottleneck in the network, since the Ethernet channels used from the nodes to the switch are 100Mbit/s channels, which does not correspond well to the 54Mbit/s theoretical maximum bandwidth of an IEEE 802.11g channel. Entailing that the emulator node only would be able to emulate one channel with maximum throughput. However, within the context of streaming applications, bandwidth utilization is not predicted to be as high as 54Mbit/s - the streams used are around 1Mbit/s. This argues, that the 100Mbit/s Ethernet channel can be used as a substitute for the 54Mbit/s WLAN channel, as the data stream will never overload the link. To further ensure that this solution is scalable the link could be upgraded to Gigabit Ethernet.

Based on the discussion above the *Centralized architecture with switch* is used. This means, that the *topology emulator* concept from here on is defined as the combination of two components; 1) an *emulator node* and a 2) a *switch*.

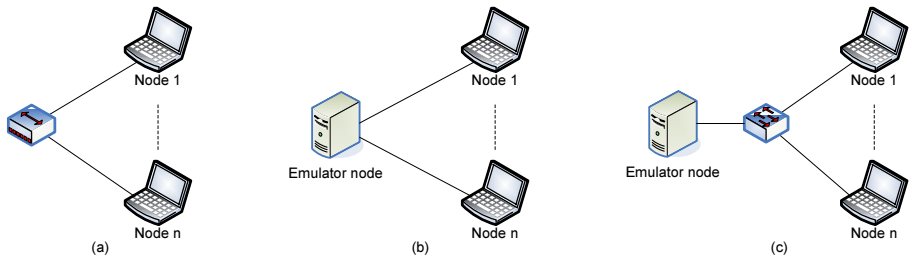

Figure 6.1: Different suggestions to the physical architecture of the testbed.

6.1.2 Logical network architecture

Having defined the physical architecture, a logical architecture is defined upon the physical to facilitate network topology emulation. The logical architecture must ensure that the statically wired network topology, including the emulator node, are completely transparent to the end-nodes. Still, the emulator node must have total control of connectivity between nodes and the ongoing traffic between them.

To revoke the node-to-node network communication facilities of the switched, physical architecture, the switch is set to tag traffic with 802.11q VLAN tags, i.e. a unique tag per port on the switch. This effectively prevents node-to-node communication, as no layer-2 broadcast domain exists in which ARP-requests are answered. The switch also supports *trunking* used to collect and share traffic between switches, facilitating VLAN domains across multiple switches. Connecting the emulator node to a trunking port reveals all communication over the switch to the emulator node.

To ensure transparency, the emulator node is implemented as a VLAN-aware bridge, as sketched in Figure 6.2. This means that the emulator node itself is invisible to the other nodes on the network, yet, traffic between the end-nodes is forwarded, which enables node-to-node communication of the nodes connected to the switch. Being VLAN-aware means that the emulator node bridges packets between all VLANs, recreating the layer-2 broadcast domain. Controlling these bridges between the VLANs means controlling node-to-node connectivity on the switched end-nodes. Trunking all packets through the bridge also enables packet loss and delay induction making it possible to emulate the unreliable nature of WLAN, since each packet can potentially be dropped or delayed from within the emulator node.

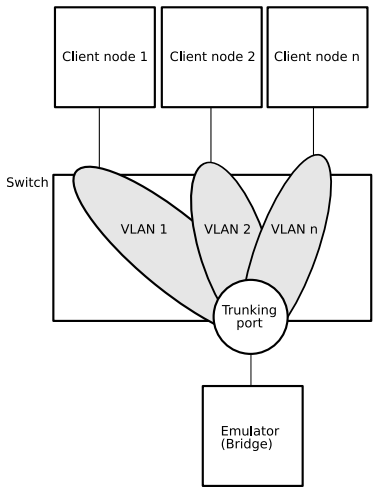

Figure 6.2: Prototype implementation of the emulator testbed.

6.2 Software architecture

The core functionality of the emulator node, mentioned in the previous section, is to impose the properties of a Mobile Ad hoc NETwork (MANET) on to the wired net-

work. This means examining each packet and imposing the MANET properties by either delaying or dropping it.

In order to achieve this, facilities are needed which enable the emulator to do this. These facilities may be divided into:

- *Simulation*: The process of determining the current state of the individual links in the network based on models representing the MANET.

- *Emulation*: Imposing the properties derived from simulation to the packets flowing through the emulator.

The emulation is facilitated in the `Emulator` component and simulation in the `Simulator` component.

Imposing delay and packet loss on the individual packets passing through the emulator intuitively suggests that simulation must be performed for each packet. Ideally simulation should run in real time providing link properties, for the emulator whenever a packet is received, based on the current state of the simulated network. To employ this approach some considerations must be made. In order to simulate in real-time the models used for the simulation must be computationally efficient implying that simple models must be used. Such a solution would limit the applicability of the testbed. Simulation may also be performed off-line making the emulation and simulation asynchronous. This can only be done under the condition that the parameters of the scenario which affect the packet loss and delay, e.g. network load, are determined and used in simulation prior to the execution of a test. Consequently, the positions of the individual nodes must also be simulated prior to test execution, entailing that the end-node applications can have no influence on the movement of the individual nodes.

To support both deterministic and stochastic emulation, the output of the simulation is distributions representing the delay and packet loss probability. Generally, stochastic emulation is preferable, since it is likely to produce more realistic results. For a deterministic model the distributions would only contain one fixed number. These distributions may vary over time according to the dynamics of the scenario.
The packet loss distribution is simplified to a Bernoulli distribution, i.e. a single number p representing the probability that a packet is lost. This information is provided for each link between the nodes of the scenario at each time step.
To make the delay distribution as generic as possible and further decrease the computational cost of emulation, the distribution is represented by a static two dimensional array where each row contains a delay CDF and each column represents the average delay for a fixed part of the probability mass from the CDF. Selecting different rows enables the use of different CDFs at different times, for instance at each time stamp if the row number is supplied for each link in each time step. The static delay table is depicted in Figure 6.4). For the delay model used in this project the row index would be the number of one hop neighbours (see Section 6.3.5 on page 57 for details), which is calculated for each time step.

Having determined the format of the different distributions the interface between the simulator and the emulator may be defined as the properties of the individual links for each time step.

These properties are stored in a file constituting the interface between the emulator and the simulator. The file format, for a scenario with two nodes, can be seen in Figure 6.3. Here each line specifies the parameters for individual links at each time step.

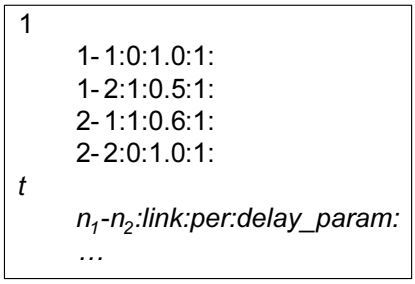

Figure 6.3: Example of the format for the link properties file at one time step in a scenario with two nodes. t is the time step, $n_1 - n_2$ is the link for which the properties apply, per the probability that a frame is dropped and $delay\_param$ the row number for the delay table.

	1	2	3	.	.	m
1	5	7	23	.	.	.
2	9	10	36	.	.	.
3	32	56	57	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	.	.	.	.	.	.

Figure 6.4: Static delay table used during emulation. The rows are the indices for the individual CDFs and the columns provide the values for the different CDFs with the resolution, m.

Defining the link properties for fixed intervals entails that some level of detail is lost. The loss of detail is governed by the time between each link update and the rate of change in properties between each update. This entails that the resolution on the simulation should be relative to the dynamics of the scenario.
The rate of change in properties is mainly governed by the cars movement. Thus the degree of change in the properties of the links, caused by the cars movement, should be considered in the implementation to determine an adequate resolution. The resolution should be high enough to ensure that the change in links properties for each step is relatively low. Consider an update rate of 1 update per second. Simulating two cars, moving at a speed of 100 km/h towards each other, means change their relative position to each other by 55 meters within each sample of 1 second. Within 55 meters many significant changes in the link properties occur (cf. section 3.1.1), why an update rate of 1 second is too slow for the emulator to work satisfactory. Considering a rate of 1 ms, the relative change is only 5.5cm, where changes are insignificant to the overall communication, why it is sufficient for satisfactory emulation.

Adopting the asynchronous approach with off-line simulation and on-line emulation makes it easy to replay a scenario, since the link properties for each time step are available in a file. This makes it easy to reproduce results or attain statistical significance through multiple executions of the same scenario.

Beyond emulation and simulation, facilities are necessary, that make it possible to display and configure a scenario and to log data gathered during scenario execution.

This leads to the components of Figure 6.5. The functionality of the individual components is sketched in the following. The description of the `Simulator` and `Emulator` components is brief, since they will be treated in detail later.

Simulator
This component is responsible for the execution of the scenario. Hence, it contains models capable of translating node positions into packet loss and delay for individual links, based on the scenario. The specifications for the scenario are configured through the `GUI`.

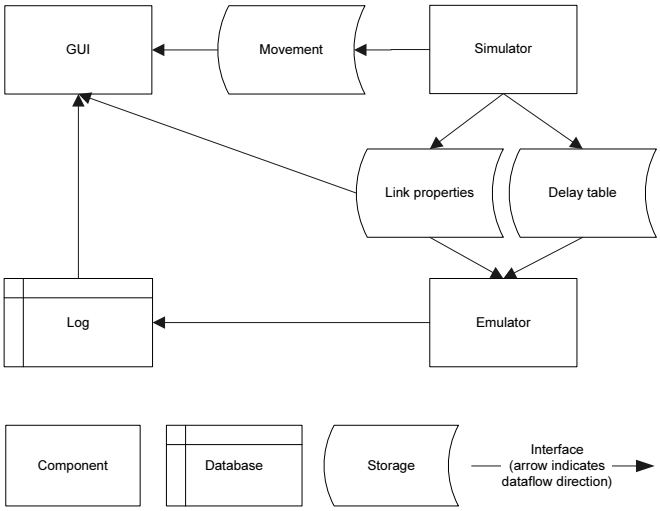

Figure 6.5: Component diagram of the emulator node.

Emulator
The `Emulator` is responsible for imposing the packet loss and delay, determined through the simulation, on the packets passing through the emulator node.

Graphic visualization
The GUI component provides the capability of configuring the information displayed on the topology emulator GUI. Moreover, it facilitates the configuration of all remaining components e.g. specifying the parameters for the `Simulator`, of the scenario to be simulated. Furthermore, the GUI visualizes performance metrics and the scenario which is executed, including node movement, possible links between nodes and utilized links. As the main task of the GUI is to provide a visualization of the scenario, a means for displaying a such scenario is developed. Overall, the GUI component is implemented in JAVA, as it provides many of the basic visualization schemes needed for creating a GUI. Also, Java is platform independent, yielding the GUI applicable on many platforms.
For visualization, a scenario consists of a map of waypoints connected by roads and cars driving these roads, communicating by network link. These two concepts, i.e. 1) the map and 2) the car topology, can both be abstracted into graph representations of nodes and edges. To visualize such graphs, *JUNG* [JUNG, 2007] and *prefuse* [Prefuse, 2007] have been investigated as graph visualization libraries for Java. *Prefuse* provides support for displaying multiple graphs in the same view, which *JUNG* does not. As it is inherent to display both the map and the car topology simultaneously, *prefuse* is chosen for visualization. Moreover, *prefuse* also supports labelling of nodes and edges in a graphs facilitating intuitive labelling of waypoints and mobile devices as they move during scenario execution.

Logging
The logging component provides facilities for storing data gathered from the network by the `Emulator` and information provided by the `Simulator` component e.g. node

positions and packet loss probabilities. The logging component records all packets entering the emulator with timestamps for later post-processing. Combined with the simulator output files previously described, all properties needed to reproduce a scenario, either for re-emulation or for re-visualization in the GUI are represented by the logger component. The logging of the packet can be performed either into files or into a database. As much of the logged data is logged for post-processing, e.g. for data-analyses of ongoing communication between nodes, a database is used. A database system provides convenient interfaces to searching, filtering and processing of logs, which can then be exported for data-processing in for instance MATLAB. For database implementation, several options exist. A few of the popular servers are MySQL [MySQL, 2007] and PostgreSQL [PostgreSQL, 2007]. As PostgreSQL provides the most features as well as being the best known, this is chosen for development of the logging component.

6.3 Simulator

This section covers the design of the simulator, focusing on the models used. First the overall design of the *Simulator* component is described and later the individual models used for simulation are described.

6.3.1 Design

As described earlier, the simulator is composed of a number of models translating the properties of a scenario into the packet loss and delay, which shall be imposed on the traffic flowing through the emulator node.

The model composition is based on the different components constituting a scenario, specified by the requirements to the model in Section 5.3.

In order to develop suitable models, it is necessary to establish the context of the models and define the predominant factors which they should reflect. These factors define the input and output of the individual models. Starting from the network layer delay and packet loss the following model composition is derived:

- *Data link layer model*: The network layer delay and packet loss is caused by delay and frame loss in the data link layer. The data link layer delay is dependent on the time it takes to send each frame, the mechanisms used to access the channel and provide the links between nodes and, in case error detection and correction is employed, the Frame Error Rate (FER). Frame errors are caused by bit errors, which occur in the physical layer, making the input to the data link model the BER, the transmission rate and configuration parameters for the mechanisms used in the data link layer i.e. the link technology configuration.

- *Physical layer model*: In the physical layer the BER is governed by the coding and modulation employed and the quality of the received signal. Entailing that the input to the physical layer should be the quality of the received signal. The quality may be specified in different ways, here the SNR is commonly used. However, different physical layer models rely on different representations of the signal quality. Therefore, the input to the physical layer model should be the Receive Signal Strength (RSS), leaving the conversion to signal quality to the model. Beyond the RSS, the physical layer configuration in terms of modulation scheme and coding should be configurable.

- *Channel model*: The RSS is determined by the degree of fading in the channel. Fading is dependent on the environment and the distance the signal must travel. Hence, the input to the channel model should be the link length and some representation of the environment. The way the environment is represented is dependent on the channel model and is not specified further, since no immediate generic representation can be established.

- *Mobility model*: The mobility model depends on the map which restricts node movement, hence, the input to the mobility model is the map specified through the scenario. The output of the mobility model should be the coordinates of the individual nodes, and not the link length as specified above, to achieve a more generic output making it possible to replace the mobility model with movement traces from external tools, e.g. ns2. This entails that a component is needed which transforms node positions to link lengths.

The models are, as described, related to each other but also need external configuration based on the scenario. The relations and the configurations is illustrated in Figure 6.6, which shows the individual models and their interfaces.

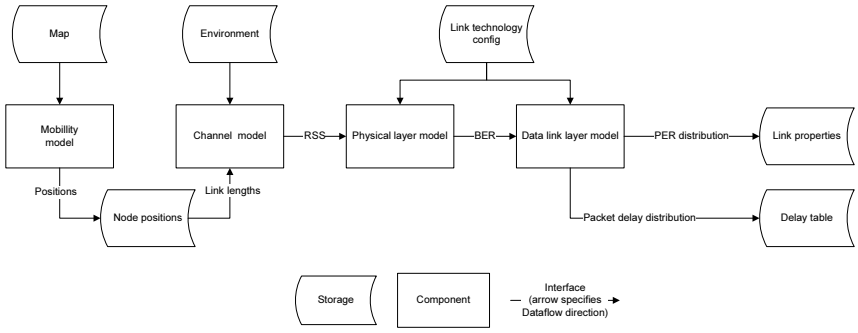

Figure 6.6: Components and interfaces of the `Simulator`. The simulation record consists of the delay table and the link properties

The following will, starting from the mobility model, cover the different models described above, relating them to the analysis performed in the preanalysis Chapter 3.1.1 on page 15.

6.3.2 Mobility model

Mobility modeling is treated in the preanalysis. Here a number of properties are derived, which should be reflected in a model characterizing the urban or the freeway setting. Moreover, a number of models are reviewed.
Including all the listed properties entails a complex mobility model which might only be feasible for large scenarios with 50 or 100 end-nodes, whereas the scale of the scenarios of this project is much smaller. Provided large scale mobility is needed tools are available, which provided many of the properties listed and outputs node positions e.g. SUMO [DLR, 2006] or UDel [Bohacek, 2004]. The following describes the mobility model implemented in the simulator.

Model

The model used is a graph based time discrete microscopic mobility model. The model is based on a graph representing the map to be used in the scenario. A map consists of:

- *Waypoints*, which are represented by vertices in the graph and may at any given time be: A starting point, a destination or an intermediate point which a node must pass through when moving from start to destination. A waypoint is characterized by it's coordinates.

- *Roads*, that connect the different waypoint and are represented by vertices in the graph. Each road is bi-directional and has infinite capacity. Roads are characterized by their speed limit and the waypoints they connect.

The map defines the restrictions imposed on the movement of the individual nodes by infrastructure. The paths which the nodes traverse in this infrastructure may be either deterministic or stochastic:

- *Stochastic*: When stochastic movement is employed a random waypoint model is used i.e. each node is initiated in a randomly selected starting point and assigned a random destination. Dijkstra's shortest path algorithm is used to determine the shortest path between start and destination [Dijkstra, 1959]. When a node reaches it's destination a new destination is randomly selected and a new path is computed.

- *Deterministic*: Using deterministic movement entails defining the paths of each individual node prior to the execution of the scenario. No interface is provided for this.

The above movement patterns are not mutually exclusive, hence, a scenario may be comprised of any combination of deterministic and stochastic paths.

The movement of the individual end-nodes, for each time step, is governed by a movement model, where each node accelerates until it reaches the speed limit of the current road.

The model, described above, provides a simple representation of the node mobility while accounting for most of the properties listed in section 3.1.3 on page 19. The properties not accounted for, are the temporal dependence between end-nodes, since roads have infinite capacity, and the dynamics caused by traffic regulations other than speed limits.

6.3.3 Channel model

The movement of the individual nodes, simulated using the model described in the previous section, causes the properties of the transmission paths between them to change. This change is governed by the properties of the wireless channel.

The wireless channel has been the subject of extensive research, since many systems rely on it e.g. GSM and WLAN. This entails that a wide range of models exist. This section provides a classification of the channel models, describing the features of the model classes and thereby providing a foundation for the choice of a feasible model. Afterwards the choice of model is justified, by describing a range of models and relating them to the scenario and the requirements to the system.

Model classification

Models of the wireless channel may be divided into two main groups, physical and statistical models, based on the modeling approach.

Physical: The physical approach, models the properties of the wireless channel by deriving a model based on the laws of physics. This approach can potentially yield very accurate results. However, accuracy is achieved through models with a high level of detail, entailing that e.g. the environment in terms of buildings and trees must be specified, which is the case for ray tracing models [Wang et al., 2004]. Accuracy may also entail cumbersome implementation and long execution times due to the complexity of the wireless channel. Simple models exist, such as the free-space path-loss model (see equation (6.1)).

Statistical: Opposite the physical approach, the statistical does not rely on accurately representing the different aspects which affect the wireless channel. The statistical model establishes an abstract representation of channel, by selecting an appropriate distribution and fitting the parameters for this distribution based on observations. This approach is often used when modeling radio propagation [Ghassemzadeh et al., 2002] [Chong et al., 2003] [Cerpa et al., 2005], since it provides a relatively accurate and computationally efficient alternative to physical approaches.

The accuracy of the statistical approach relies on the quality of the measurements, which the parameters of the model are based on. Hence, an accurate statistical model requires measurements from the environment to be modeled. These are difficult to obtain for e.g. a MANET. The physical approach relies on an accurate representation of the physical properties, which are also difficult to obtain depending on the complexity of the environment affecting the channel.
The following describes both physical and statistical models and relates them to the environments specified in the preanalysis.

Models

As depicted in Figure 6.6 the input to the channel model is the link length i.e. the distance between sender and receiver. This information shall be used to determine the contribution of the channel to delay and packet loss.

The propagation of the wireless signal introduces a delay, which depends on the distance the signal must travel, however, this delay is in the magnitude of a few microseconds and is therefore disregarded. Thus, the main contribution of the wireless channel is the attenuation and variation of the signal, leading to an increased probability of an erroneous decoding of the wireless signal. Hence, the output of the model is the received signal strength.

Signal attenuation and variation has been widely studied and a large variety of models exist for determining how a signal degrades with distance. Friis defines a physical model of the received signal strength as [Friis, 1946]:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{6.1}$$

Where G_t and G_r is the gain for the sender and receiver respectively, λ, the wavelength, L the system loss, e.g. loss in components of the receiver, P_t the transmitted

signal power and d the distance between sender and receiver. This model assumes that there are no objects between sender and receiver affecting the signal. Hence, it is not adequate for the urban environment, where there are many objects disrupting the signal propagation e.g. buildings and trees. More accurate physical models use ray tracing to determine the received signal. Here a representation of the environment is used to determine how the signal propagates yielding a high accuracy. The precision of the model is dominated by the accuracy of the representation of the environment. When a detailed model of the environment is not present, but a relatively high accuracy is desired, measurements may be performed to fit statistical models to the desired environment.

A statistical model, which encompasses signal attenuation, is the shadowing model. Shadowing can be characterized as a large-scale fading, since it is the attenuation of the signal caused by objects obstructing the propagation path between transmitter and receiver. [Rappaport, 2002] has developed a model which incorporates shadowing and outputs the path loss in dBm:

$$PL(d) = PL(d_0) + 10\beta log(\frac{d}{d_0}) + X_{dBm} \tag{6.2}$$

Here β is the path loss exponent determining the degree of attenuation, d_0 a reference distance and X_σ is a log-normal random variable with mean σ representing the variation in signal strength at a given distance. Different parameter settings are available for this model along with a description of the environments for which the parameter settings are suitable in [Rappaport, 2002]. This model does not include the effects of small-scale fading, however this may be remedied by extending X. For scenarios where a dominant ray, e.g. Line of Sight (LoS), between sender and receiver often is present, like the freeway environment, a Rician distribution is commonly used to model the signal variations caused by small-scale fading. In scenarios where there is no dominant ray, a Rayleigh distribution can be used (see [Proakis et al., 2001] for details on Rician and Rayleigh fading channels).

The shadowing model provides a good compromise between between complexity and availability, since it may incorporate large-scale fading, and by extension small-scale fading, along with a fairly simple statistical model, where parameters for different environments are available. Hence, it is selected as channel model for the urban and freeway scenario.

The parameters are specified as part of the scenario, hence, they do not change during scenario execution. This entails that the model does not account for the actual spatial and temporal variations of the signal caused by the movement of sender and receiver in the environment, but an average, since the coefficients σ and β are constant throughout the simulation. This simplification decreases the accuracy of the model, but avoids the need to specify a complete map of the environment and the resulting increased complexity caused by the updating of the models parameters.

Implementation

The shadowing model was implemented in *MATLAB*. In order to achieve the desired output, the RSS was determined by subtracting the path loss, determined by (6.2), from the transmitted power.

$$[P_r]_{dBm} = [P_t]_{dBm} - PL(d) \tag{6.3}$$

A simulation of the RSS using this model can be seen in Figure 6.7. Here $d_0 = 1$,

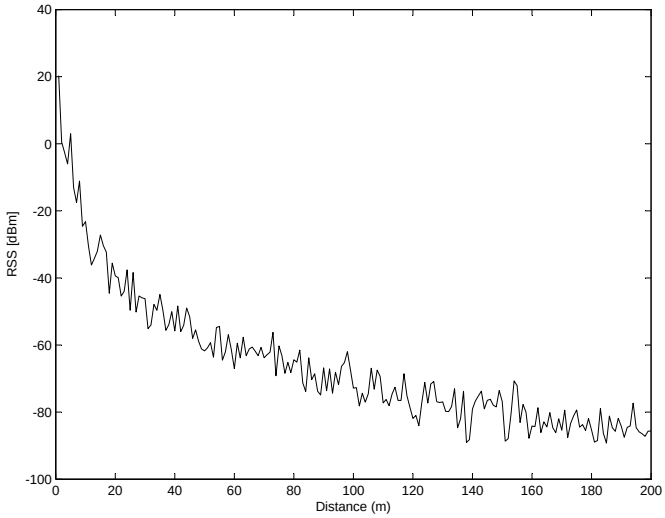

Figure 6.7: RSS simulation using the shadowing model.

Environment	β	σ
Free space	2	0
Urban	2.7 - 3.5	4 - 12

Table 6.1: Selected parameters for the shadowing model from [Rappaport, 2002].

$\beta = 2$ and $\sigma = 4$. As expected the signal fluctuates and decays with distance. Different parameter settings for the implemented model can be seen in Table 6.1. Small-scale fading was not included in the implemented model.

6.3.4 Physical layer model

The purpose of the physical layer is to detect the signal transmitted across the wireless channel, transforming the signal into a stream of bits. The delay at the PHY layer is comprised of the time each bit occupies the channel, and the contribution to packet loss is the rate of incorrectly detected bits, the BER. The delay is easily obtained, since it can be defined as the number of bits sent divided by the rate at which they are transmitted. This calculation is left to the data link model described later, since the knowledge of the amount of bits to be transmitted is available in the data link layer. The BER is more difficult to obtain, since it is dependent on the modulation scheme.

As specified in the preanalysis 802.11a is used to model the link technology used in the scenario.

IEEE 802.11a uses OFDM as modulation scheme. A commonly used approach for modeling OFDM, is detailed in [Mangold et al., 2004], where the RSS is used to compute the BER. This model is selected as the model for 802.11a in this project due to it's availability and since parameters are provided by [Zang et al., 2005], to configure the model for simulation of 802.11p.

Data transmitted using OFDM is demultiplexed into a number of sub-carriers each containing part of the data to be sent. The data of each sub-carrier is modulated in

OFDM symbols containing a number of bits. The number of bits contained in each symbol varies with the modulation scheme.

Mangold assumes that there is no Inter-Symbol Interference (ISI) and Inter-Carrier Interference (ICI), hence, that the channel delay spread, i.e. the maximum delay experienced between the reception of two identical signals, is below some threshold entailing that the symbols of the individual carriers do not overlap, and that the different sub-carriers do not interfere with each other. This leaves the background noise, other stations transmitting and the fading of the wireless channel as the only sources of disturbance.
These assumptions imply that the model is not accurate for channels subject to severe small-scale fading.

The model is based on the ratio of energy per symbol E_{av} to the energy of the background noise N_0, i.e. the SNR for each OFDM symbol. This is expressed as:

$$\frac{E_{av}}{N_0} = \frac{\alpha_g C}{N + \sum I} \tag{6.4}$$

Where C is the received signal strength, N is the background noise and $\sum I$ is the cumulated interference level i.e. interference from other stations transmitting. α_g is a power loss caused by the guard interval for each OFDM symbol, and is defined as $\alpha_g = \frac{T_b - T_g}{T_b}$ where $T_b = 4\mu s$ is the duration of an OFDM symbol and $T_g = 0.8\mu s$ is the duration of the guard interval. This interval is introduced to avoid the signal degradation introduced by ISI. ISI is caused by multiple receptions of the same signal with different delays. T_g assumed to be larger than the delay spread of the channel, eliminating ISI.

Equation (6.4) forms the basis for calculating the BER, the formulas used to obtain the BER from the symbol SNR are described in Appendix E.

Implementation
The model of [Mangold et al., 2004] was implemented in *MATLAB*. The bit error probability of this model for different modulation schemes can be seen in Figure 6.8. The results shown in the figure match the ones reported by both [Mangold et al., 2004] and [Chung and Goldsmith, 2001]. Implying that the model is correctly implemented.

6.3.5 Data link model

Moving from the physical layer to the data link layer, bit errors are translated into frame errors. The FER is correlated with the delay, since frame errors lead to retransmissions increasing the time from frame transmission to successful reception. The delay introduced in the link layer may vary depending on the mechanism used to access the channel. As IEEE 802.11a is used to model the physical layer the same standard is employed here as the starting point to model the mechanism of the data link layer.

In the following, first the channel access scheme used in IEEE 802.11a is described, to provide an understanding of the mechanisms and enabling a choice of model, second the choice of delay model is justified and the model is outlined. Afterwards the packet loss model is described. Lastly, relevant aspects of the model implementation is covered.

In ad hoc networks using IEEE 802.11a the DCF, which is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), is used to access the channel.

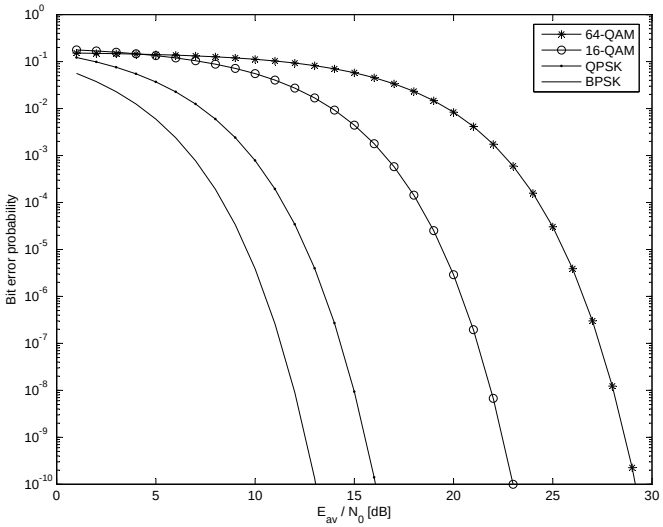

Figure 6.8: Correlation between symbol SNR and bit error probability for different modulation schemes of OFDM.

Here each station senses the carrier prior to initiating a frame transmission. Carrier sensing is performed at the PHY-layer by sensing the channel and at the MAC-layer by analysing frames sent by others nodes to determine the duration of the ongoing transmission. The duration of the current transmission is included in the header of selected frame types, allowing each node to maintain a Network Allocation Vector (NAV), indicating the time the channel is reserved by an ongoing transmission.

Two schemes are available to reserve the channel, RTS/CTS and basic access. The basic access scheme consists of a two way handshake with positive acknowledgement. When a frame is ready to be sent, the node immediately initiates the transmission, if the channel is sensed idle for a DCF Inter-Frame Space (DIFS) period. If this is not the case, the node draws a random backoff counter from a uniform distribution, called the contention window (CW). The backoff counter represents the time, discretized in time slots of size 9μ seconds for IEEE 802.11a, the node has to wait before attempting a transmission. The backoff counter is decremented each time the channel is sensed idle for one time slot. If the node experiences an unsuccessful frame transmission, the contention window is doubled and a new backoff counter is drawn. For multiple failed transmissions of the same frame, the contention window is doubled until it reaches some predefined maximum value, CW_{max}. A frame transmission using the basic access scheme is illustrated in Figure 6.9. Note that the intervals; Short Inter-Frame Space (SIFS), DIFS and PCF Inter Frame Space (PIFS) in the figure, are used to prioritise channel access between different frame types.

To accommodate for hidden terminals (described in the preanalysis), the basic access scheme is extended to a four way handshake. Here RTS and CTS messages are exchanged prior to data transmission, to increase the probability that all stations are aware of the ongoing transmission (see Figure 6.10).

When employing both RTS/CTS and the basic access schemes a maximum number of retransmissions is defined for all frames, entailing that if this limit is exceeded the frame is discarded. When below this threshold frames are delayed according to the contention for the channel and the retransmission of erroneous frames, when above,

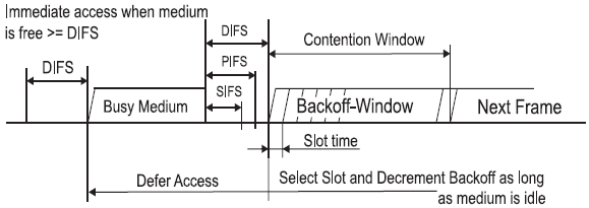

Figure 6.9: Transmission of a frame using the basic access scheme of the DCF [IEEE, 2003].

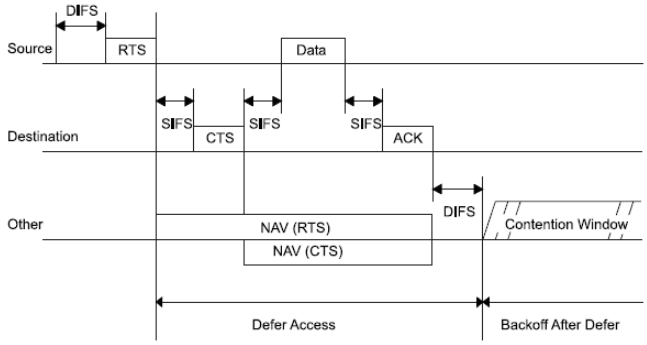

Figure 6.10: Transmission of a frame using the RTS/CTS access scheme of the DCF [IEEE, 2003].

frames are discarded. Frames may also be discarded if the rate of frames, arriving in the data link layer, exceeds the data link layer frame service time, since this may cause a buffer overflow.

Delay model

A large variety of models of the data link level delay exist, since it has been a popular research topic due to wide commercial application of IEEE 802.11. Many are based on the work performed by Bianchi in [Bianchi, 2000], where the saturation throughput is derived using a two-dimensional Markov model, assuming an error free channel. However, most work is focused on establishing means of the data link layer performance under saturated conditions, which is not suitable for the scenarios in this project. As specified in the requirement analysis the output of the data link model should be a delay distribution. Few studies have been performed where the delay distribution of the data link level delay has been derived, however, [Tickoo and Sikdar, 2004] and [Engelstad and Osterbo, 2006] establish this distribution. [Engelstad and Osterbo, 2006] focuses on delay bounds and delay percentiles, whereas [Tickoo and Sikdar, 2004] focuses on deriving a distribution for the delay experienced by a target node. The latter approach corresponds well to the concept of imposing delay on packets from individual nodes and is selected to model the delay.

The model of Tickoo is based on a discrete time G/G/1 queue. The model supports the use of arbitrary frame length and packet inter-arrival time distributions under finite, unsaturated load.

The foundation of the model is the frame collision probability, which constitutes the only notion of frame errors in the model. This probability is, in the model, only

dependent on the number of nodes contending for the channel and their packet arrival rate. This model leads to the following simplifications:

- Frame loss due to imperfect channel conditions is not taken into account, yielding a potentially lower delay than one would expect in a real experiment.
- Frame loss caused by the disturbances from hidden terminals is not included. This simplification is valid when the RTS/CTS access scheme is used, under the assumption that there are no asymmetric links. However, links are not symmetric when the shadowing model is employed.

These simplifications may be accommodated to some extent by expanding the collision probability to include the FER and by including disturbances from hidden terminals when determining the BER based on Equation (6.4).

Packet loss model

The BER determined in the physical layer model causes frame errors in the data link layer. However, the BER cannot be mapped directly to a FER, since error correcting decoding schemes are employed in the physical layer, entailing that the mapping is not simple [Pursley and Taipale, 1987]. However, [Jelitto and Truong, 2003] provides an approximation to the translation, where the BER is divided by a value, $b = 3.3$, before translating it into a FER, yielding:

$$FER \approx 1 - (1 - BER_{payload}/b)^{L_{frame}} \cdot (1 - BER_{plcp}/b)^{L_{plcp}}, \tag{6.5}$$

here L_{frame} is the length of the frame in bits and L_{plcp} the length of the physical layer preamble in bits. The physical layer preamble is always transmitted at the lowest bit rate, hence, the modulation scheme of the preamble and the frame may differ, which in turn, may results in different bit error rates, $BER_{payload}$ and BER_{plcp}.. This model assumes that an entire frame including the preamble is subject to the same SNR and that there are no errors in control frames, i.e. RTS, CTS and ACK frames are not subject to errors. The model may easily be extended to incorporate this, since an error in either of the control frames, of the access scheme used, causes the node to defer assuming the frame was lost. However, control frames are mostly transmitted using lower transmission rates, hence, they are less susceptible to errors.

Under the assumption that packets are not fragmented, the packet loss probability with transmission retry limit N can be expressed as, $PER = FER^N$. In case fragmentation is used and the loss of one fragment entails a packet loss $PER = 1 - (1 - FER^N)^F$, where F is the number of equally sized fragments. Packets may be fragmented either if they exceed the maximum allowed frame size or if schemes are employed which add a fragmentation threshold to increase the throughput.

This model assumes that no frames are lost due to buffer overflow in the data link layer. When simulating networks under saturated conditions this might lead to imprecisions, since the PER determined by the model in these situations probably will be too low when the frame errors caused by buffer overflow are omitted. This imprecision is unlikely to be experienced in the scenarios where the model is used in this project, since the network load for these scenarios is relatively low.

Implementation

The packet loss model was implemented according to the formulas listed in the previous section assuming no fragmentation. This assumption is made to decrease the

complexity of the model and to ensure consistency to the delay model which also assumes no fragmentation for simplicity. This simplification is unlikely to have a significant impact on the precision of the models when used for the scenarios, specified in later chapters, since the packet sizes in these scenarios do not exceed the fragmentation threshold of 802.11a.

The delay model was implemented according [Tickoo and Sikdar, 2004], for further details on the model refer to Appendix F. The main result from Tickoo is the packet service time at the individual nodes, expressed as a Probability Generating Function (PGF). This service time is expressed as the multiplication of the PGFs of the PMFs which comprise the different aspects of the service time.

- *l(n)* is the packet length PMF, which describes the time a successful packet transmission occupies the channel. For the RTS/CTS access scheme it is the time it takes to send RTS, CTS the frame and ACK.

- *bo(n)* constitutes the PMF for the number of backoff slots experienced by the node.

- *x(n)* comprises the PMF for the time where the backoff counter may not be decremented due to transmissions from other nodes occupying the channel.

To avoid transforming the PMFs into PGFs, the PMFs are convolved to obtain the service time distribution. This convolution is performed under the assumption that the PMFs are independent. Assuming that packet size is constant $l(n)$ simplifies to a scaling of the convolution product of $bo(n)$ and $x(n)$. However, $bo(n)$ and $x(n)$ are not completely independent since they both depend on the frame collision probability. The contribution to $x(n)$ from frames colliding is small compared to the contribution from frames which are successfully transmitted, since frames which collide only occupy the channel for the time it takes to send a RTS frame opposed to a successful transmission which occupies the channel for the time it takes to send RTS, CTS, the frame and ACK. Thus the main contribution to $x(n)$ are the successful transmissions, these are independent from the time spent in backoff. This indicates that the error introduced by assuming independence is relatively small and the precision lost due to the assumption is considered to be justified by the reduced complexity of the model. To verify the model the delay distribution was determined using the same parameter setting as used by Tickoo and comparing to his results.

The delay Cumulative Distribution Function (CDF) for a network with 10 nodes each generating 100 kbit/s CBR traffic using 1000 byte UDP packets, can be seen in Figure 6.11. The average delay is $4ms$, which seems to correspond to the delays reported by Tickoo based on the figures in [Tickoo and Sikdar, 2004]. The CDF has a stair shape, this is caused by the fact that the main contribution to the delay at the node is comprised of the synchronized successful transmission of other nodes, each contributing to the steps on the CDF.

To convert the CDF into a generic format readable by the simulator, it is transformed into an array where each field contains a part of the probability axis. The converted CDF, with a resolution corresponding to 0.001 of the probability axis for each field, is seen as the broken line in Figure 6.11. The figure indicates that the loss of precision caused by the conversion is small.

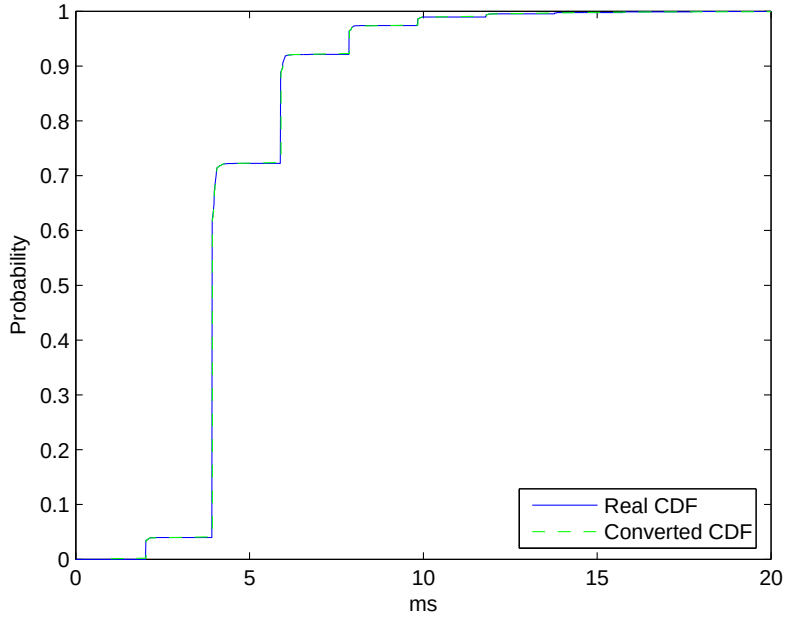

Figure 6.11: CDF and converted CDF, for a network consisting of 10 nodes, calculated based on the delay model.

6.4 Emulator

This section describes the design and performance verification of the emulator components contained in the emulator node of the testbed. Generally, functional requirements to the emulator are defined to guide the overall design. Sub-requirements are derived to guide design of the individual components. From these requirements, solutions for functionalities of each component are presented and capabilities and limitations of the chosen solution are evaluated.

Based on the output of the link property simulation, the job of the emulator software is to effectuate these properties onto links and ongoing traffic between end-nodes connected to the switch in the testbed. To be able to do this, the emulator must be able to 1) read from the simulation files, 2) control the bridges between the virtual LANs and last 3) intercept the ongoing traffic on the bridges. Reading simulation data from a file, controlling the VLAN bridges and handling packet delay requires a timer with a high resolution, which is describe prior to the remaining components.
In Figure 6.12 interactions between components maintaining the required tasks are illustrated. The solutions for the components maintaining each functionality requirement are presented in the following. The specific hardware platform on which the topology emulator is implemented is presented in Appendix A. In addition, the performance of each solution is verified, showing that it is implemented without affecting the ongoing traffic over the bridges more than requested by the simulated data.

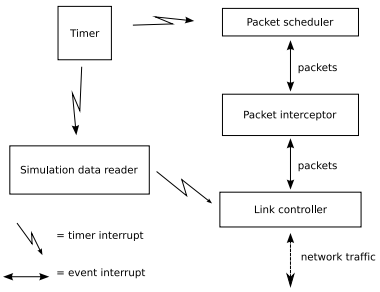

Figure 6.12: Components maintaining the tasks of the emulator software.

6.4.1 Timer

Using Linux as operating system on the emulator node, several ways of implementing a high resolution timer exist. This can be accomplished by using real-time capabilities of the Linux kernel itself (kernel version > 2.6.18 [Kernel, 2007], by using verified real-time extensions to Linux such as RTAI [RTAI, 2007a] or RTLinux [RTAI, 2007b] or self-contained real-time operating systems such as eCos [eCos, 2007] or QNX [QNX, 2007]. Throughout history, the real-time capabilities of the Linux kernel have been questionable, providing timer precision only down to 1ms inside the kernel. However, as of kernel 2.6.18, the real-time capabilities of the Linux kernel have improved [Kernel, 2007]. Now, the Linux kernel is pre-emptable, allowing for processes to have high priorities and thus controllable in a real-time manner. As using a regular Linux with a pre-emptable kernel, by far, is the easiest way to implement a reliable, high-precision timer, this approach is chosen for the timer in the emulator.

The timer ticks are supplied by a hardware clock called the Real Time Clock (RTC), which is installed on practically every kind of main-board. This clock can be programmed to report hardware interrupt with frequencies of 64-8192 Hz in multiples of 2. By giving the timer the highest priority in the system, only hardware interrupts, such as hard-drive operations, graphical adapter refreshes or network adapter interrupts, can preempt the reporting of timer ticks. Unfortunately, hardware interrupts are in the same pool of priority, and therefor interrupts from the network adapter cannot be given higher priority than the remaining types.
The only concern here is, that the emulator is placed in an environment with potentially high network activity, meaning that during high network load the timer may be delayed. The following verification shows that the precision of the timer can be considered reliable, though a bit slower than 8192 Hz and that network load does not influence the timer precision.

Figure 6.13 shows the probabilities of getting different timer period values. Table 6.2 summarizes these number into means and standard deviations depending on the network load on the emulator. From the table it is clear, that the timer runs precisely and independent of network load. From the figure it seen, that not all periods fall into the same bins. This is due to the timer being preempted by hardware interrupts. However, as seen from Figure 6.14, it is observed that a shorter period directly follow a longer period which means that the timer itself compensates for the hardware interruption. In addition, the standard deviations of the samples show that the delays in timer ticks are not significant enough to cause the timer to overflow. Calculating the double tick probability - i.e. the probability that a period encompasses two ticks - reveals that

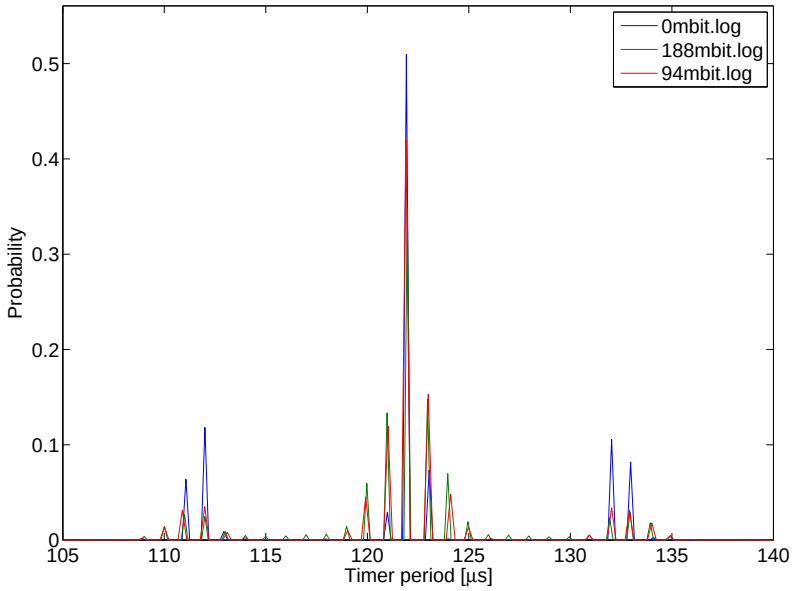

Figure 6.13: Probabilities of timer periods.

Load	Timer μ [μs]	Timer σ [μs]
0 Mbit	122.0770	6.4663
94 Mbit	122.0771	4.8035
188 Mbit	122.0767	4.7103

Table 6.2: Results from timer period benchmark, sample size = $8192 \cdot 30$.

$P(X_{period} \leq 244.2534) = 0$, no matter the load on the emulator. This means that the timer ticks once and only once per timer period, and is thus a reliable timer interrupting at a frequency of 8191.6Hz. The fact that it is not running at 8192Hz is important when used for sampling. A deviation of even nanoseconds in period has great impact when emulating for long periods of time as it accumulates an error in time effectively slowing down the time in the emulation.

6.4.2 Reading the simulation output

The simulation output data is loaded into and stored in the emulator by a reading process. The reading process can be implemented by several means including shared memory, files and databases. In this work, files are chosen as simulation output containers. This is primarily for portability, e.g. to support simulation data generated from external tools - such as *ns-2*. Also, when considering performance aspect, files are the faster approach; shared memory needs a lot of memory available (typically RAM) for future data, which must be pre-allocated and is thus unavailable to the packet handling part of the emulating process. Choosing a database has pros and cons; the database features

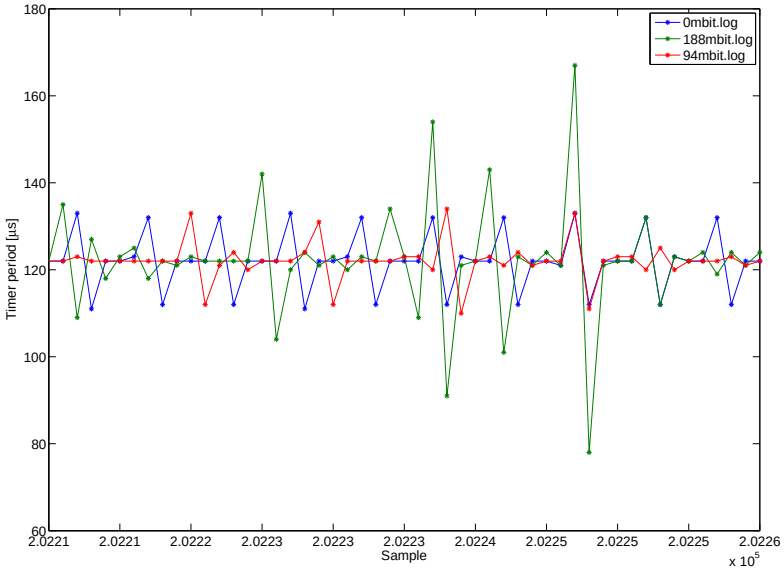

Figure 6.14: Timer compensation of hardware interrupts in timer periods. A longer period is directly followed by a shorter period.

a simple interface to data, optimizing storage- and search-capabilities, yet ensuring real-time access to the data is difficult as a database management system is typically a large program running in user-space. This means, that it would be pre-emptable by the reading process itself, possibly causing a dead-lock in the system.

All simulation data is stored in only one file, where each property sample is marked with time stamps as to when to apply the properties. When emulating, the data must be applied at the specified time stamp, in real-time preferably with high precision. The reading is performed by checking the timer a each tick and, when the time of a time stamp is reached, reading the properties specified with the time stamp. The fact, that the frequency of the timer is 8191.6Hz, with a period of approximately 122μs, introduces a jitter on the reading of the time stamp. The advancing timer and the time stamp does not necessarily equally match in value and thus a 'greater than or equal to'-comparison must be carried out. This comparison means that the reading of a time stamp has a jitter equal to the timer period, i.e. 122.077 μs. However, as the timer is reliable, this jitter is stationary and thus not a drift in the reading process. In conclusion, the jitter is accepted for the reading process.

6.4.3 Controlling VLAN bridges

The bridges between the VLANs created by the emulator represent two-way WLAN links between end-nodes. Emulating the properties of each node-to-node WLAN link means to control the properties of each bridge. This control is performed on multiple levels to attain the required functionality.

From the simulation, a set of parameters is produced to be imposed on the WLAN

links. This set consists of *link existence*, *packet error rate* and *packet delay distribution*. Significant of these parameters are that they apply to one-way links. This means that only controlling the (two-way) bridges is not sufficient as removing one VLAN bridge effectively removes two WLANs, which is not necessarily intended. Instead, control of the individual traffic flows across the bridges is required.
A ubiquitous tool for traffic flow control is *iptables*[Netfilter, 2007], which controls a set of packet forwarding rules in the network handling part of the Linux kernel, called *netfilter*. Although applicable for network traffic flow control, *iptables* is not applicable for traffic flow control in the emulator. End-nodes use ARP-requests to discovery network addresses of each other based on hardware-addresses. These frames do not reach the network layer on the emulator, and hence, cannot be controlled by *netfilter* rules from *iptables*. A consequence of this is that ARP-request are forwarded between node even though the links should appear to be non-existing. As an alternative, *ebtables* [Netfilter, 2007] is used, which controls data link layer rules in *netfilter* - i.e. the *iptables* of the data link layer. By using *ebtables*, asynchronous control of the bridged traffic flows is obtained by controlling each frame forwarded by the emulator, also including ARP-requests.

Using *ebtables* for controlling traffic flows is difficult in a real-time perspective. *Ebtables* is a user-space tool and timing of prototypes have indicated that executing an *ebtables* command in the normal system environment to update the netfilter rules takes more than 1ms and has a very poor precision. This means that it is very hard to determine when the new properties are actually applied in the kernel and effective on the forwarded traffic. Moreover, the fact that it is a user-space tool combined with the timer structure implemented in the emulator endangers the execution of the *ebtables* command, i.e. it could be starved by being preempted by the timer.
To solve this, a kernel module is implemented called *ebtables_emulation*, which is depicted in Figure 6.15. This module plugs into the *ebtables* part of *netfilter* in the kernel allowing it to examine every incoming frame before it leaves the kernel. Static rules to drop packets on individual VLAN flows are injected into *netfilter* using regular *ebtables*. This means that frames sent from, for instance, *vlan2* to *vlan3* has one rule to drop frames instead of forwarding them to *vlan3*. A rule for each combination of incoming and outgoing VLANs are represented in *netfilter*.
However, since it is not the intention to drop all packets on the emulator, an extension is added to the *ebtables* rules. The *ebtables_emulation* kernel module decides if a frame matches the rule set for it. As only one rule exists for a given frame, the frame is forwarded to the network level in the emulator if it does not match this rule. In the *ebtables_emulation* module, the VLAN tags of the frame are compared to an array holding link existences - if the link does not exists, a *match* verdict is given to netfilter, and subsequently the frame is dropped by way of the rule.

Updating the link existences in the *ebtables_emulation* module must be carried out as soon as they are read from the simulation data, as described previously. Implementing such a user-space to kernel-space inter-process communication has many pitfalls, as not much standard functionality is available in the kernel compared to user-space. Methods such as shared memory, queues using character devices or procfs, and netlink sockets are available for this. The simplest form of these communication methods is creating a file in the *procfs*, which is a virtual file-system accessible from both kernel and user-space. One of the major advantages of using *procfs* is the registration of callback functions within the kernel-module for reads and writes to the procfs-file. This way, busy-waits when reading from the file are eliminated, as the callback functions are called upon each read of the file and write to the file and precious processing power

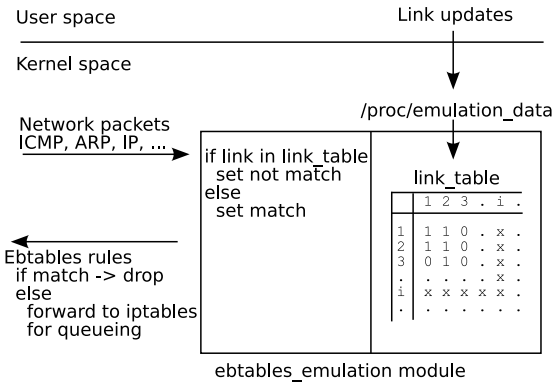

Figure 6.15: Design of the *ebtables_emulation* module. Packets from the network stack are injected into the module. If a link does not exists between the sender and receiver VLAN, they are declared matches to a DROP rule in *ebtables*.

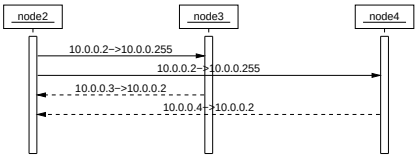

Figure 6.16: Test result of the broadcast domain functionality of the emulator.

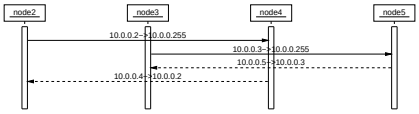

Figure 6.17: Test result of the partitioning functionality of the emulator.

is saved. The procfs-file enables the reading process to communicate periodic updates from the simulation data to the *ebtables_emulation* module, which in turn imposes link existence directly on forwarded frames - including ARP-requests.

Several series of tests have been carried out to verify functionality of the *ebtables_emulation* module implementation. In Appendix B, the bridging functionality of the module is verified. The results of the tests show, that the bridging is working as expected. In Figure 6.16, the result of a broadcast test is seen showing that although the nodes are separated into different VLANs by the switch, they appear to be within the same partition of the topology. Figure 6.17 shows the successful result of emulating two partitions, where node-pairs appear to be out of broadcast range between the two partitions, due to emulated missing logical links.

As well as the bridging functionality has bee tested, the ability to dynamically update the link properties is verified in Appendix B. The emulator is configuring to toggle the existence of a link between end-nodes between open and closed with a period of 1 second. By measuring the timestamps of packets getting through the open link on the receiving node, it is possible to calculate the time it takes to apply the properties, though, only within the precision of the packet rate used for measuring. The calculations show, that an excess time of $68\mu s$ is spent on average per 1 second period. However, as the precision of the measurement is $\pm$0.5ms, this excess time may as well be from sending or forwarding packets in the emulator rather than applying properties. Moreover, as the measurement is within the precision, it is concluded that

the *ebtables_emulation* module is capable of applying properties within a precision of 0.5ms, which is sufficient for applying the properties outputted by the simulation with a minimum resolution of 1ms.

6.4.4 Intercepting ongoing traffic and emulating

To emulate the properties of a link used for communication, the simulated properties must be imposed on the packets actually flowing between node. This requires each packet to be intercepted and treated according to the simulation. The interception of packets is accomplished using *iptables* and a feature of *netfilter* called *netfilter_queue*. Use of these tools enables enqueueing of packets in a user-space accessible queue by inserting a rule into *netfilter*. In the emulator, this rule encompasses every packet forwarded between VLANs, thus covering all end-node traffic. When a packet is enqueued, it is not forwarded until a *verdict* has been given - ruling either to *accept* or *drop* it. This queuing and verdict is exploited in the packet error and packet delay emulation.

As the simulation provides the *packet error rate*, the packet error emulation reduces to deciding whether to drop or accept a packet based on the packet error rate. To do this, a uniformly distribution random number from 0 to 1, U, is drawn and compared to the packet error rate given by the simulation. The packet drop decision rules is then

$$\text{if } U_{uniform} \begin{cases} < p_{\text{packet\_error}} & \text{then } \textit{drop} \text{ packet} \\ \geq p_{\text{packet\_error}} & \text{then } \textit{accept} \text{ packet} \end{cases}$$

If the packet is to be dropped, the drop-verdict is given directly back to *netfilter* and the next packet is examined.

If the packet is not dropped, a delay for the packet is determined based on the number of 1-hop neighbors of the sender and the packet is delayed accordingly before forwarded to the receiving node. This real-time delaying of the packet is a non-trivial task in Linux. Several ways of using sleep functions in Linux have been investigated, each yielding an overhead delay of more than 10ms, which is unacceptable. However, using the timer structure previously described, acceptable timer precision is attained to support real-time delay. This interrupt structure enables a packet list to be maintained by a *packet scheduler* process, in charge of scheduling packets and sending accept-verdicts to *netfilter* when appropriate. The packet list is implemented as a circular buffer of linked lists. The slots of the circular buffer corresponds to time slots of the timer, i.e. each slot represents 122.0770 μs. The linked lists in each time slot are implemented to support multiple packets to be scheduled for delivery in the same time slot. The buffer rotates once per timer tick. A packet is inserted in the linked list located in the time slot corresponding the delay offset from the current time slot. This slot is chosen using the expression $((\texttt{round}\,(t_{\text{delay},\mu\text{s}} \cdot f_{\text{timer}}))\ \ \texttt{mod}\ \ n_{\text{slots}})$ to the nearest integer and using that integer for slot placement. $t_{\text{delay},\mu\text{s}}$ is the simulated delay in microseconds, f_{timer} is the frequency of the timer, `round` is used for attaining an integer-index for the circular buffer and the modulus operation is to stay within the buffer when placing the packet.
Due to the granularity of the timer, delay jitter of up to 122.0770μs is created by the packet scheduler, which can be seen from Figure 6.18 and is evaluated in a subsequent section.
Accept-verdicts are given to *netfilter* for each packet residing in the linked list in the time slot corresponding to the current time.

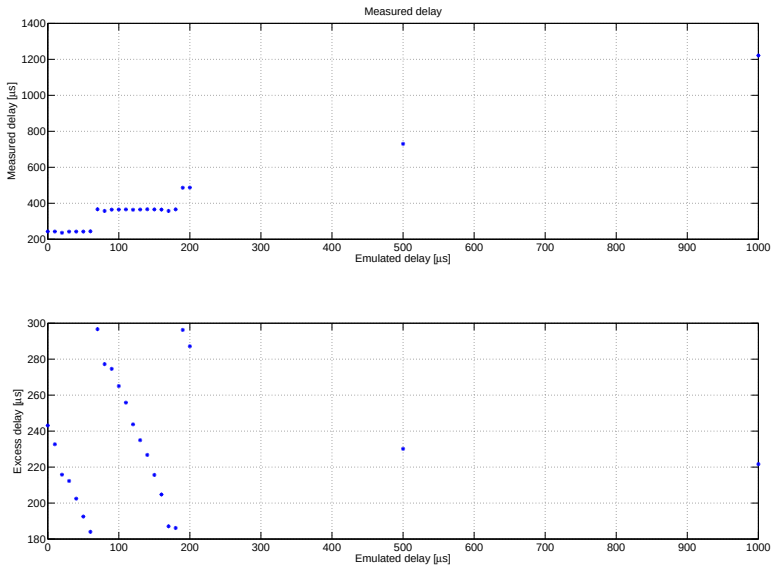

Figure 6.18: Delay precision of the scheduler. Upper figure is one-way delay compared to emulated delay and lower figure is delay-bias compared to emulated delay. The shifts in the upper figure and the slopes in the lower figure are due to scheduler granularity of approximately 122 μs.

To test that the emulator is capable of applying given values of packet loss and delay, different test have been performed, which are described in Appendix B. From these tests, it is concluded, that the implemented loss and delay implementations work as expected. The emulator is capable of emulating a given packet loss onto ongoing traffic and is capable of delaying not-dropped packets of ongoing traffic, though, still within the granularity of the scheduler. Concerning delay, a constant excess delay is observed, as a consequence of using the described network architecture with more links and more packet processing than in a direct connection. The magnitude and consequences of such a delay is investigated and discussed in the following chapter, describing the performance verification of the emulator.

Chapter 7

Testbed evaluation and discussion

Having described the design and implementation of the topology emulator, this chapter describes the experiments performed to determine the performance of the develop approach and thereby the boundaries which it may be applied within.

The service time of the topology emulator is examined and compared to the minimum delays to be emulated based on the delay model described previously. This comparison is performed in order to establish if the emulator is capable of emulating a wireless channel without introducing a delay bias. Further, the performance of the topology emulator is tested with regards to capacity, in terms of achievable packet rate, to determine the number of nodes which may be connected to the topology emulator and the amount of traffic the individual nodes may generate.

7.1 Packet delay evaluation

The physical network architecture of the topology emulator entails more transmissions per frame than on a wireless link. The topology emulator employs Fast Ethernet and Gigabit Ethernet on the wires, but still, the end-to-end delivery of one frame on a physical level requires 4 transmissions. More transmissions result in an increased delay of frame delivery. Moreover, as the emulator node bridges packets and emulates packet properties, all frames from a sending end-node are withheld on the emulator node until an entire packet is received, the packet is then examined by the emulator node, and all frames are afterwards delivered to the receiving end-node, given that the packet is not dropped on the emulator node. In addition to the delay induced by the increased number of transmissions, also queueing, processing and scheduling on the emulator node takes time and delays the packets. Hence, it is required to perform an evaluation of the amount of *service* delay induced by the topology emulator when forwarding traffic.

7.1.1 Service delay determination

On all links, wired as well as wireless, the delay of a transmission is dependent on the size of the packet. To investigate if the service delay of the topology emulator is

also dependent on the packet size, due to the queueing and processing, experiments are carried out consisting of transmitting packets with different sizes.

The experiments employ two end-nodes, one sender and one receiver. The sender *pings* the receiver 100 times and measures the round-trip time of the *ping* packet. The *pings* are repeated for packet sizes in the range 8-64000 bytes in intervals of 1000 bytes. 8 bytes is the lowest possible payload *ping* can send, and 64000 bytes is the maximum of the possible payload of an IP-packet. This experiment is performed for both a direct 100Mbit/s connection of the end-nodes and for the end-nodes connected via the emulator. This is to measure the reference delay of an ordinary 100Mbit/s transmission, including any noise from network adapter, kernel or operating system.

From Figure 7.1 the results of the experiments are seen. The figure illustrates the measured delays and lines linearly fitted to the measurement points. The parameters of these regression lines reveal several facts. The slopes of the lines are identical, meaning that the service delay added by the topology emulator is independent of the packet sizes when compared to a direct connection of end-nodes. Also, the offset of the lines are shifted meaning that the service delay can be calculated by the difference of the values of the regression lines at zero, which is $287\mu s$.
The figure also shows the minimum for transmissions using IEEE 802.11a. This delay is calculated based on the delay model used in the simulator and combined with the processing delay at the end-nodes. The processing delay is approximated by subtracting the theoretical packet transmission time for Ethernet from the measured delay for the direct connection. The resulting delay, for 802.11a with end-node processing, is above the service delay for the emulator, indicating that the emulator is capable of imposing delays from the delay model without introducing additional delay. At packet sizes between 1000 and 2000 the minimum delay from the model is very close to the emulator service delay. However, considering that the packets are unlikely to experience the minimum delay, as indicated in Figure 6.11, this small difference will rarely be experienced. Thus, the emulator does, in most cases, have a reasonable safety margin to prevent it from imposing additional (unmodeled) delay on the individual packets.

7.1.2 Service delay independent of emulator load

Processing of packets on the emulator node is carried out by only one serving unit, i.e. the CPU. As the CPU services all operations on the emulator node, including reading packets from the network adapter and writing them back again, the actual service rate of the queue is network-load dependent. At a high load, many high-priority hardware interrupts are generated, requiring the CPU to service these first. To investigate the implications of this dependence, delay experiments with different cross-traffic loads are performed. In addition to the test configuration described above, cross-traffic is created using a traffic generator node and a receiver node in the high-load experiment. Table 7.1 show the different service delays measured from the *ping* measurements with varying cross-traffic load on the emulator. The conclusion from this test, is that the service delay is not significantly dependent on the cross-traffic load on the emulator. However, as it is not possible to create greater reliable loads (described in section 7.2) with the equipment at hand, effects of higher loads cannot be concluded.

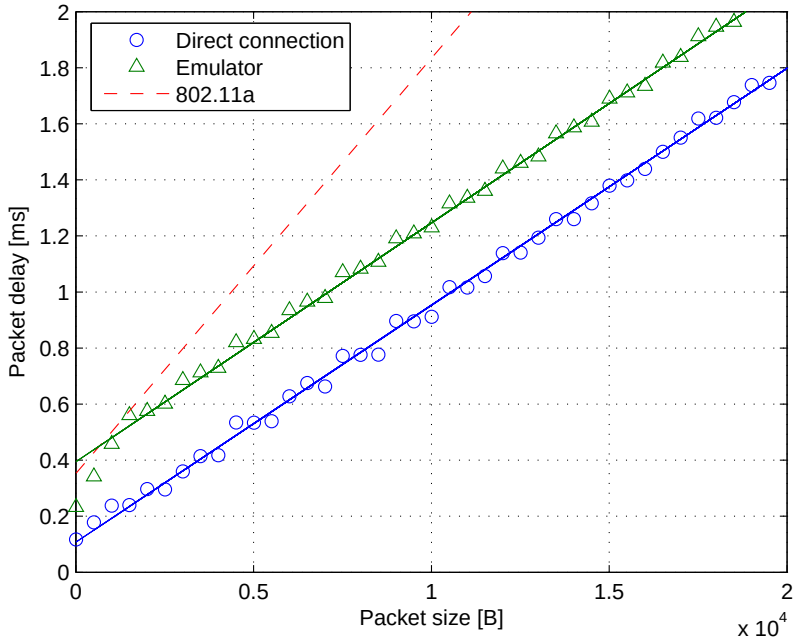

Figure 7.1: Topology emulator service delay from various packet sizes and lines linearly fitted to each measurement series. The broken line shows the minimum delay, based on the delay model, which should be imposed on the different packet sizes.

Cross traffic rate [pps]	Fitted service delay [μs]
0	287
1000	290
5000	293
10000	298

Table 7.1: Topology emulator service delays measured with various traffic loads on the topology emulator.

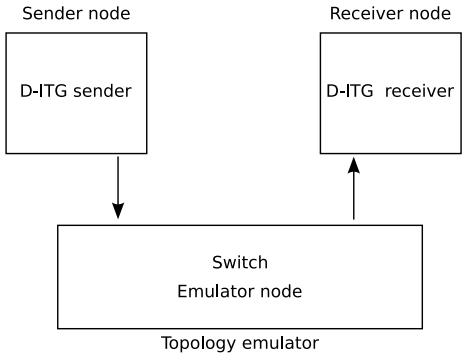

Figure 7.2: Setup of the testbed for evaluating packet forwarding capabilities of the topology emulator.

7.2 Packet rate evaluation

As the topology emulator is forwarding all network packets for the end-nodes, an evaluation of the forwarding capabilities is required. This evaluation is required to be able to assess the capacity of the emulator with regards to the number of end-nodes that can be handled and also with regards to the traffic patterns they can generate. The following section describes an evaluation of the tools used for generating and measuring packets on the testbed and the performance evaluation of packet forwarding capabilities of the topology emulator.

7.2.1 Generating network traffic

For generating traffic to test the forwarding capabilities of the emulator, a tool called Distributed Internet Traffic Generator (D-ITG) [D-ITG, 2007] is used. D-ITG provides a means for generating traffic with different types of characteristics. This includes different distributions (e.g. uniform, exponential, Cauchy, normal, etc.) of both packet size and inter-departure times of packets. Other traffic generating tools such as the kernel traffic generator [Pktgen, 2005] and ettcp [ettcp, 2002] have been investigated, but D-ITG provides the best precision, the most control of parameters and the most features. D-ITG employs both a generating process and a receiving process that reports properties of the expected and actual packet flows. Figure 7.3 shows an output of D-ITG, generating a constant bit-rate flow at 10Kpps, measured on the emulator node. The scenario is depicted in Figure 7.2. The measurement confirms that the traffic generator is in fact capable of generating a flow with a constant bit-rate, here illustrated by packet rate.

Due to properties of the Linux operating system, all processes, including D-ITG, suffer from possible preemption as they are almost[1] all user-space processes. Preemption of the traffic generator, e.g. by hardware interrupts from hard drives or a graphic adapter, has several consequences relevant to this project.

[1]The kernel traffic generator resides in kernel space, but has not shown to be able to deliver a constant flow of packets for this project.

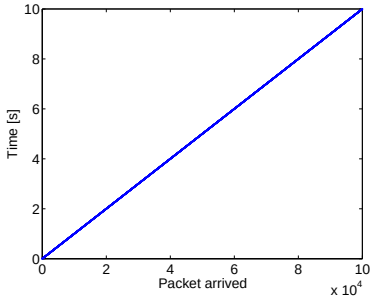

Figure 7.3: Measurement of packets received at the emulator node over time. The constant incline indicates a constant packet rate.

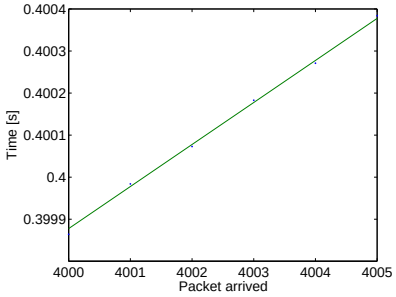

Figure 7.4: A closeup of the packets received, revealing that packets do not arrive at a completely constant rate on a small scale.

Preemption results in an uneven generation of a flow on a small scale. This means that if the process is preempted, the generated packets experience a small delay (a few microseconds) before being transmitted. This is seen from Figure 7.4, as the packet arrivals are not completely periodic. On a receiving node, being both emulator node or end-node, the impact of preemption is significant; if the receiving process is preempted during packet reception at high packet rates, the packets are dropped. This is seen from Figure 7.5, showing the ratio between the rates reported by D-ITG of sent and received packets.

From Figure 7.5 several issues are identified. The three sub-figures present ratios of received packets versus sent packets of end-nodes at different packet rates. The data point are single measurements from the testbed. The upper figure, illustrating the sent/received ratio between directly connected nodes, shows a decrease in received packets proportional to the rise of packet rate above 10Kpps from the sender. This is a result of packet loss due to preemption problems or loss of packets in the testbed (i.e. in the switch or in the emulator node) due to overload. Such packet losses should be confirmed using a packet capture program, but as discussed in the following section, packet capture programs too suffer from preemption problems resulting in wrong packet recordings and inconsistent measurements.
In conclusion, the recording depicted in the upper sub-figure shows that D-ITG suffers from preemption problems at high packet rates, meaning that either the sending process has difficulties sending the generated packets or the receiving process has difficulties recording the received packets, if received at all. This result must be taken into account when evaluating packet rate measurements on the testbed.

As a further investigation, the generated traffic flows are sent through the emulator to determine the effect of the forwarding and to some extend the capabilities of the emulator. Results similar to those previously described are seen from the middle sub-figure of Figure 7.5. However, in this sub-figure more tests have a ratio of 1, i.e. the received amount is equal to the sent amount. This could be due to the packet scheduling functionality of the testbed. As it is not possible to measure the specific delay of the packets during these experiments, the results do not show if the packets are actually *rate-limited* by the scheduler in the emulator and thus all are successfully transmitted.

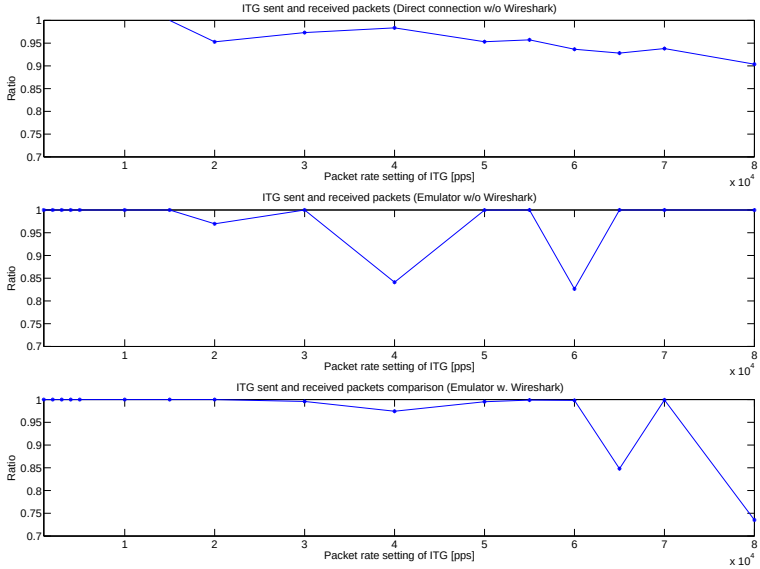

Figure 7.5: Illustration of results of preemption problems in traffic generation and receiving process. The figure shows the ratio of received packets compared to sent packets at different packet rates; upper figure show results of directly connected nodes, the middle figure shows results of nodes connected over the emulator as does the lower figure, where Wireshark is enabled on all nodes including the emulator.

Nevertheless, the emulator has an impact on flows at high packet rates influencing the measurements.

Furthermore, an experiment has been carried out having Wireshark running on all nodes, including the emulator. The results are depicted in the lower sub-figure of Figure 7.5 and show that the presence of running Wireshark does not have significant influence on the ratios. This is due to the lower priority of Wireshark, which in turn has other consequences, which are discussed in the following section.
Finally, the two lower sub-figures of Figure 7.5 show indications of the capabilities of the emulator with respect to rate of the packet forwarding. Although it is not unambiguously shown, the experiments indicate that the emulator is capable of forwarding packets with a rate near 80Kpps. This is a relatively high rate, considering that end-nodes are connected via fast Ethernet link transferring at 100Mbit/s. Links speeds and packet rates are discussed in section 7.2.3.

7.2.2 Problems using passive packet capture

For performing measurements for experimental network analysis, typically Ethereal has been used. However, since May 2006, Ethereal has been discontinued and replaced by a tool called Wireshark [Wireshark, 2006]. The resulting product, Wireshark, is Ethereal with a new name. Wireshark is used in this project for measuring network traffic properties. However, prototyping has uncovered that when measuring traffic properties with high packet rates, i.e. over 10Kpps, both Linux, Wireshark, and also Ethereal, have very drastic limitations. This has also previously been concluded by [Deri et al., 2004], who is working on alternatives to passive packet capture. However, besides [Deri et al., 2004], very little research regarding this problem has been found. The problem is relatively new due to the appearance of network traffic speeds of gigabit and above. The investigation of these limitations and a discussion of the consequences when performing high packet rate measurements with a standard Linux operating system and standard PC is described in the following.

Wireshark (and Ethereal) uses the standard library *libpcap* for capturing packets for analysis. *Libpcap* is a kernel interface, common to many operating system, featuring raw packet capture from a networking device. It is concluded in [Deri et al., 2004] that the performance of tools utilizing *libpcap* varies significantly between operating systems. This is mainly due to the applied scheduling policies in the operating systems. When a packet arrives at the network adapter, a hardware interrupt is requested in the kernel which then preempts the running process to handle the received packet by placing it in a buffer in the kernel. Then, a software interrupt is requested by the kernel network stack to handle the packet appropriately, also allowing *libpcap*-tools to capture the packet. Basically, a starvation problem arises in this packet handling process, when packets arrive very fast, as numerous hardware interrupts are generated. This leaves the kernel with very little processing time for the software interrupt service routines, including *libpcap*. To assess the impact of this problem, as the emulator is a potential subject to very high packet rates (described in the subsequent section), a series of tests are devised. Repeating the previously described tests, and using Wireshark to record packets reveals that the problem of starvation exists in the testbed.

The results of the tests are seen in figures 7.6 and 7.7. The figures show the ratio between the amount of sent and received packets reported by D-ITG and Wireshark respectively. The overall impression from the two experiments is that Wireshark and D-ITG do not agree. Keep in mind, that these figures present ratios of the totals reported

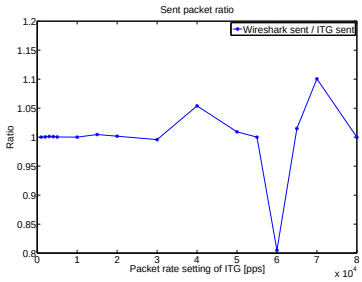

Figure 7.6: Ratio of sent packets measured by Wireshark versus D-ITG at different packet rate.

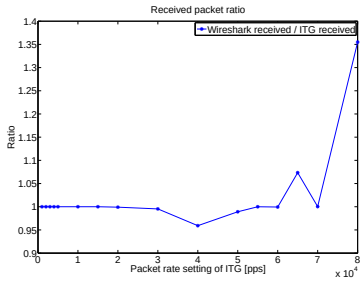

Figure 7.7: Ratio of received packets measured by Wireshark versus D-ITG at different packet rate.

by D-ITG and Wireshark, which is why the unreliability of D-ITG does not influence the ratios.

From the sent packets ratio (cf. Figure 7.6), it is seen that Wireshark reports both more and less packets than D-ITG, at high packet rates, i.e. above 10Kpps. At lower rates the amount captured by Wireshark is equal to the amount reported sent by D-ITG. No sensible explanation has been found to why Wireshark captures more packets than reported by D-ITG. The result has been reproduced on three different PCs when sending packets. More research is needed here to determine the cause of the erroneous packet rates reported by Wireshark.
For the received packets ratio (cf. Figure 7.7) a similar phenomenon is observed, however, here the overall tendency is that Wireshark reports lower numbers than D-ITG. The hypothesis that Wireshark is unreliable when capturing packets at high packet rates due to preemption, is supported by measurements performed on the emulator node. Figure 7.8 shows a comparison between different measured packets ratios from Wireshark. From the figure it is seen that the amount of packets measured on the emulator (forwarded packets) is lower, compared to both the amount sent and received. This is probably due to the heavy packet processing performed on the emulator node at high priority. The large amount of packet processing leaves little time for Wireshark to capture packets, which entails starvation, resulting in more dropped packets on emulator node.

The consequence of these findings is that Wireshark, when applied on Commercial Off-The-Shelf (COTS) PCs, cannot be used as a reliable tool for capturing packets at packet rates higher than 10Kpps. Yet, measurements at lower rates, i.e. 1-10Kpps, show no influence of preemption, and thus measurements at these rates should be trusted. It is also observed during testing that giving Wireshark a higher priority, to minimize the effect of preemption, shifts the result of preemption from errors in Wireshark to errors in actual kernel space packet handling, resulting in even higher packet losses at the nodes. Finally, it is observed that when using Wireshark for a long period of time at high rates, e.g. capturing packets over 1-2 minutes, the amount of caching and memory allocation in Wireshark requires disk-swapping to be used. This results in even more load on the processor, ultimately reducing the reliability of Wireshark measurements further.

These findings are important to the testbed performance verification performed in this project and the impact is described in the following section. Moreover, it reveals that care should be taken when using Wireshark (or Ethereal) as a measurement tool

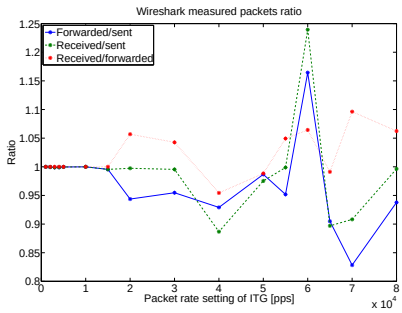

Figure 7.8: Ratios of packet measurements from Wireshark at different places in the testbed; on the sending node, on the emulator (forwarded packets) and on the receiving node.

for experiments on the end-nodes at high packet rates.

7.2.3 Emulator node packet rate performance

The emulator, constituted by the switch and the emulator node, is potentially subject to high packet rates generated by end-nodes. The nodes are connected through fast Ethernet connections to the switch, which in turn is connected to the emulator node via a gigabit interface. However, the limitations of the individual links of the end-nodes are not directly relatable (e.g. maximum 10 100Mbits/s links on a 1000Mbit/s link) to the maximum enforced load on the emulator node from the gigabit interface.

As the emulator emulates the link layer properties of a wireless LAN, inherently it also emulates the frame buffering capabilities of the link layer at the end-nodes. As also previously described, this means that packets are forwarded to the emulator as soon as they are generated at the end-nodes. This is done by the fast Ethernet links and the switch. This aggregation of packet flows from many independent sources at the emulator may lead to excessive load at high packet rates.
As an example, consider having several connected end-nodes with streaming applications running between end-nodes in pairs of two, employing a constant bit rate. The end-nodes are all emulated to be within transmission range of each other. The streaming applications each generate network traffic, that in turn generate flows of frames from the sending end-nodes - flows that must be forwarded by the emulator. Even though the average frame rate of the each flow is relatively low, the synchronicity of the flows may create abnormally high frame rates in bursts at the emulating node. Normally, the frame rate of a stream is dependent on the stream bit-rate. The higher the bit-rate, the more frames. However, as the size of the payload of the frames increases with the bit-rate, a natural convergence of frame rate exists. The maximum frame rate of a fast Ethernet connection is 144 Kfps (frames per seconds) at the smallest frame payload, which can be calculated as

$$\frac{100,000,000\text{bit/s}}{(n_{\text{frame-gap}} + n_{\text{preamble}} + n_{\text{header}} + n_{\text{payload}} + n_{\text{CRC}})\,\text{bytes/frame} \cdot 8\text{bits/byte}}$$

Table 7.2 shows the values for the Ethernet protocol. Yet, this rate is rarely achieved due to frame payload changes. For instance, a full speed FTP transfer over a fast

Ethernet component	Value [bytes]
$n_{\text{frame-gap}}$	12
n_{preamble}	8
n_{header}	14
n_{payload}	46-1500
n_{CRC}	4

Table 7.2: List of sizes of components needed to send an Ethernet frame. An Ethernet frame is generally constituted by the header, the payload and the CRC. The remaining components are from the physical layer, needed to send the frame.

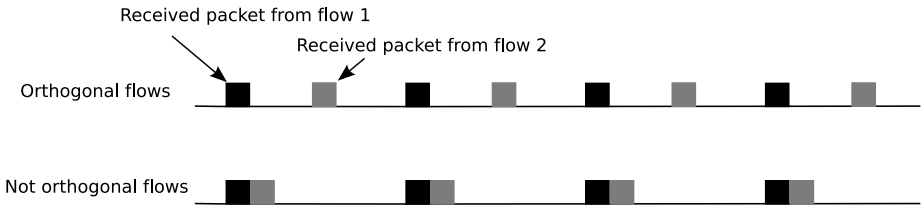

Figure 7.9: Illustration of the increased aggregate flow into the emulator node due to independently generated flows.

Ethernet link generates approximately 8200 fps, as the frame payload is maximized to 1500 bytes.

If the flows in the example have equal frame rate and are completely orthogonal, the frame rate of the aggregate flow experienced by the emulator node is equal to the sum of the frame rates of all the end-nodes. This is, however, not likely, as the flows from the end-nodes are completely independently generated. If the flows do not have equal frame rate or are non-orthogonal, the frame rate of the aggregate flow converges to the maximum frame rate of gigabit, which is 1.4Mfps. While converging to maximum, the increased frame rate is experienced as bursts of frames at very high rates at the emulator node, while the overall average frame rate remains a sum of the averages. If the characteristics of the traffic generated by the end-nodes change to become bursty, as it is the case with variable bit-rate streams, the bursty frame rate property becomes much more evident. Figure 7.9 illustrates the problem with non-orthogonal frame rates. Due to this property of the incoming traffic on the emulator, it is important to evaluate how well the emulator node handles high frame rates.

Unfortunately, as described in the previous sections, neither the available traffic generators, nor packet capturing applications can be considered reliable enough to deliver trustworthy measurements at high packet rates. Several alternatives for packet generation exist, such as using high-end packet generation equipment or more nodes generating lower packet rates which aggregate into a high and bursty frame rate on the emulator. Small-scale experiments have been carried out with 3 laptops connected to the testbed, generating asynchronous traffic to create non-orthogonal flows at a low aggregated frame rate. The experiments show, that although the maximum frame rate attainable with gigabit is 1.4Mfps, these experiments have not shown frame rates of bursts higher than 200-300Kpps. Observed rates are calculated based on frame inter-arrival times measured on the emulator. A such measurement can be seen from Figure

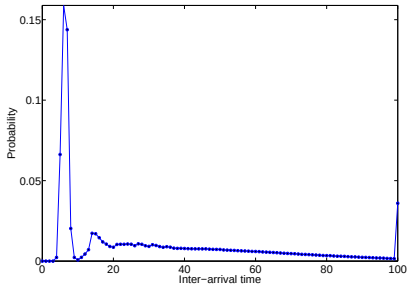

Figure 7.10: Probabilities of frame inter-arrival times on the emulator.

7.10.

From Figure 7.10 it is seen, that the minimum frame inter-arrival time measured is 4 microseconds, generated with 6 asynchronous flows of $\sim$5Kpps forwarded by the emulator (2 from/to each of 3 end-nodes). This corresponds to a frame rate of 250Kpps. Inter-arrival times of 3 microseconds have been observed. However, they were recorded with Wireshark, which has shown unreliable at high packet rates, meaning that the rate might not be trusted, when receiving 30Kpps (aggregation of 6 flows of $\sim$5Kpps each) on the emulator. Using the fact that the minimum frame inter-arrival times are 3-4 microseconds and considering the period of testing, which is 10 minutes, it is assumed that the emulator is capable of handling the gigabit input from the interface. However, as the frame inter-arrival times are not 0.6 microseconds ($\frac{1}{1.4Mfps}$), as they should be at full gigabit speed, some delaying of the packets occur upon reception. This delay is most likely due to buffering in the network adapter or by the network adapter driver before reaching the kernel network stack. This delay is included in the measurements performed in section 7.1 and is therefore not considered further here.

As presented previously, sustained packet forwarding rates of the emulator have been observed as high as 80Kpps. During testing, these rates have been applied constantly over a period of 10 seconds. Also, burstiness of the independent flows aggregated into rates of up to 250Kpps at the emulator is shown to be handled, possibly resulting in delayed packets, though. Based on these numbers, several conclusions can be made. In the following, the boundaries for packet rate performance are listed, then these boundaries are compared to the requirements to the emulator and conclusively a discussion of possible end-node- and emulation configurations is presented to exemplify the possible uses of the testbed, with regards to frame arrival rates on the emulator node.

In the current design of the topology emulator, the maximum number of connected end-nodes is 20, which is a hard limit of end-node connections, due to number of available ports in the switch. Also, due to the design, the use of a gigabit Ethernet adapter limits the aggregate throughput of the emulator to

$$\sum \text{link throughput} < 1\text{Gbit/s}$$

The maximum sustained rate of packets handled by the emulator limits the performance

by the average packet rate of the end-nodes, κ_i as

$$\sum_{i=1}^{N} \kappa_i < 80000\text{pps}$$

where N is the total number of nodes. The burst rate limits the aggregated bursts from the traffic of the end-nodes defined by burst-rate λ_i as

$$\sum_{i=1}^{N} \lambda_i < 250000\text{pps}$$

The ability to emulate independent partitions of nodes means that traffic arrives independently onto the emulator and must be handles accordingly. Handling broadcasts in such domains means to replicate packets at the emulator and forward to all nodes in the partition. This enforces a load on the emulator, that is bounded by the following expression. Considering a number of K partitions each with a number of nodes n_i. The minimum broadcast period is then defined by

$$\frac{\sum_{i=1}^{K} {n_i}^2}{80000\text{pps}}$$

The requirements to the emulator are specified in Chapter 5. The bandwidth requirement was derived, as a maximum capacity of the emulator, to 35Mbit/s per radio domain. With a maximum of 20 nodes, the worst case scenario of the emulator is 10 separate radio domains, containing each 2 nodes, all expecting 35Mbit/s from the channel. By calculation, the enforced rate of packets aggregated on the emulator in such a scenario is approximately 28.5Kpps, using a packet size of 1500 bytes. This illustrates, that the emulator is capable of handling expected end-node load from a fully loaded switch in terms of packet rates.

The design decision made of connecting the emulator using a gigabit connection means that the switch cannot handle aggregating more than 10 fully loaded fast Ethernet connections to the emulator. The following discussions are of different types of traffic patterns from a different number of nodes (up to 20) not fully loading the gigabit link.

Pure routing Proactive routing generates a periodic flow of packets potentially loading the emulator with bursty traffic. Consider OLSR routing sending *hello*-packets with a period of 1s. With 20 nodes connected, this would in the worst-case cause bursts of 20 packets at gigabit speed. With the argument of the buffering occurring at the Ethernet link layer, and the limited size of the burst, it is concluded, that the emulator is capable of handling a such scenario.

Distributed black box The distributed black box application distributes data recorded from a car into a VANET [HIDENETS, 2006b]. An example of use here would also be periodic data transfer, but with more data per IP-packet, yielding larger frame bursts at the emulator. Considering 20 nodes participating in such an application, the burst size and rate on the emulator could potentially become very high. However, as a combination of the emulator dropping and delaying packets due to contention and packet errors, the communication scheme of the distributed black-box (e.g. use of TCP for reliability) is expected to degrade the aggregated

flow on the emulator. Compared to the sustainable rate of 80Kpps of the emulator, packet generation periods of participating nodes should not be below 2.5ms (20nodes/80000pps). In conclusion, such a scenario would be handled by the emulator given the packet rate limitation.

Video conferencing The video conferencing scenario considered in this project is also evaluated for applicability. Both the packet size and the packet rate of video streams is relatively high, compared to the previously described scenarios. The specific parameter settings of a video stream are very dependent on the content, encoding and quality [Seeling et al., 2004]. For an example stream containing a lecturer, i.e. a moving face on a background, encoded with H263 at 256Kbit/s, the minimum IP-packet payload is 1461 bytes [Seeling et al., 2004]. This packet size means that the worst case burst frame-arrival rate on the emulator, while forwarding two video streams, is just below the rate for maximum frame payload size, i.e.$\sim$82Kpps. The emulator is shown to handle bursts at this rate, meaning it can handle at least two video streams for a video conference. As each flow only employs an average bandwidth of 256Kbit/s, 20 nodes with one stream each would cause a 5Mbit/s average load on the emulator, i.e. a sustainable rate of $\sim$420pps, which is readily handled.

Chapter 8

Routing evaluation methodology

In the previous chapter the application area of the topology emulator is determined enabling the specification of scenarios which are supported within the emulator framework. In order to facilitate evaluation of routing and define relevant scenario, this chapter describes the methodology used for evaluating the dependability of the routing algorithms. The routing algorithms chosen for evaluation are presented and dependability and performance metrics are identified to evaluate the routing algorithm resilience. This chapter forms the methodology framework for the routing evaluation described in the succeeding chapter.

8.1 Routing algorithms

This section describes the investigation of *what and how* of ad hoc routing protocols, that are implemented in Linux, in order to establish which are available for emulation evaluation. The IETF MANET [IETF, 2007] working group has accepted four ad hoc routing protocols, which are widely referenced, for Request For Comments; Optimized Link State Routing (OLSR), Dynamic Source Routing (DSR), Ad hoc On-demand Distance Vector (AODV) routing and Dynamic Manet On-demand (DYMO) routing. OLSR is a proactive protocol and DSR, AODV and DYMO are all reactive protocols. However, the state of the accepted protocols are only at a specification level, so no implementations are given by the IETF. Therefore, to use these protocols for experimental testing, the availability of real (Linux) implementations has been investigated. All of the protocols listed in Table 8.1 are potential subjects of evaluation with the testbed as they are ad hoc routing protocols and have implementation versions, however, they are all unipath protocols. Multipath protocols are potentially better than unipath protocols, in some cases, when it comes to routing dependability. However, implementations of multipath protocol are currently not available. The dependability of such protocols should be evaluated in future research, by way of the testbed, once implemented on the end-nodes. For evaluation, Dynamic Source Routing (DSR) and Optimized Link State Routing (OLSR) are chosen, as they are representative for the two groups of routing protocols described in the preanalysis, i.e. proactive (OLSR) and reactive (DSR). Furthermore, they are ad hoc routing protocols, thoroughly validated and verified in existing research and available as implementations, making them potential subjects for implementation in a solution for VANETs.Finally, the implementations of the protocols are relatively simple to deploy, making them usable for experimental

Protocol	Specification	Description	Limitations	Availability
OLSR	RFC3626	link state/quality routing (proactive)	link quality is measured by packet loss rate, not signal quality - can run on kernel 2.6	by link[1]
AODV	RFC3561	ad hoc on-demand distance-vector routing protocol (reactive)	different versions, must run on kernel 2.6	by link[2]
DSR	RFC4728	ad hoc on-demand source routing (reactive)	can run on kernel 2.6	by link[3]
DYMO		Dynamic Manet On-demand routing	Kernel 2.6 support, supports multiple interfaces	by link[4]

Table 8.1: List of available ad hoc routing protocol implementations.

evaluation in the testbed with the topology emulator.
In the following, a brief description of the functionality of each protocol is presented.

OLSR

The following is based on [Jacquet et al., 2001]. As OLSR is a proactive routing protocol it maintains a routing table, which is continuously updated by exchanging messages containing information about the network topology. Some of these messages are flooded through out the entire network. In order to reduce the overhead produced by this, MultiPoint Relays (MPRs) are used. Each node in the network maintains a list of neighbours which are it's MPRs. MPRs are responsible for forwarding broadcast messages for nodes which have selected them as one of their MPR nodes. A node selects it's MPRs based on the local topology so that messages from the node can reach all it's two-hop neighbours. OLSR uses two message types for establishing the network topology:

- *HELLO*: This message type contains a list of one-hop neighbours for the sending node, and is not relayed. *HELLO* is used to advertise the one-hop neighbours of one node to other nodes, enabling all nodes to maintain a view of the local

[1] http://www.olsr.org, should run on kernel 2.6
[2] http://www.docs.uu.se/docs/research/projects/scanet/aodv/aodvuu.shtml http://core.it.uu.se/adhoc
[3] http://core.it.uu.se/adhoc
[4] http://sourceforge.net/projects/dymoum/

topology in terms of one- and two-hop neighbours including the status of the individual links, i.e. symmetric, asymmetric or MPR.

- *Topology control (TC)*: *TC* messages are used to propagate information about a nodes local topology to the entire network, i.e. TC messages are relayed. Only nodes which have been selected as MPR nodes may send TC messages, in their *TC* messages they append a list of nodes which have selected them as MPR. The information contained in the *TC* messages is used to update the routing table providing routes, where all intermediate hops are MPRs, to all nodes in the network.

The approach outlined, has the advantage that a route is always available to any node in the network, reducing latency. However, as OLSR relies on propagating information about the network topology to all nodes, an initialization period is needed for a newly formed networks to stabilize before routing may be performed efficiently. Furthermore, in dynamic topologies link breakages occur often, entailing that the topology information must be updated more often, reducing the achievable throughput. Hence, care must be taken when choosing the periods for *HELLO* and *TC* messages.

DSR

The following is primarily based on [Johnson et al., 2001]. Reactive routing protocols do not rely on continuously maintaining a complete view of the network topology, but discover paths on demand. DSR embeds the route which a packet must traverse in each sent packet, to find and maintain this route DSR employs two mechanism:

- *Route discovery* is used when a node, the source, wishes to send a message to another node, the destination, in the network, but is unaware of a route to that node. DSR maintains a route cache based on all messages heard on the individual nodes, hence, route discovery is not always needed. When needed, the source broadcasts a route request packet, which is forwarded until it reaches the destination, each intermediate node appends their own address to the route request, thereby establishing the route the request has traversed. Upon receiving the route request the destination node returns a route reply to the source node, containing the route traversed by the route request. Thereby a route between source and destination is provided.

- *Route maintenance* monitors the route in use and reports if any of the links in the route break. Each intermediate node, in a route, which receives a packet is responsible for ensuring that it is successfully delivered to the next node in the route. This is achieved by acknowledging each packet and retransmitting when no acknowledgement is received. A packet may only be retransmitted a fixed number of times, if this number is exceed the transmitting node returns a route error message to the source node, indicating the broken link.

Since all packets contain the route they must traverse the route caches at the different nodes is updated more frequently as the amount of traffic increases, thereby decreasing the need for route discovery. In highly dynamic networks the use of route caching may lead to excessive route maintenance, since links are frequently broken. This entails that, in some cases, it is more feasible to perform route discovery instead of using the route cache. When route discovery is used DSR introduces a delay compared to the OLSR routing protocol.

8.2 Evaluation metrics

Relevant metrics for evaluating routing algorithms need to be identified. Typically, when routing algorithms are compared, their evaluation is perform analytically, where the metrics are mostly qualitative [Perkins and Royer, 1999], such as loop-resilience, routing philosophy (flat or hierarchical) or some complexity based considerations, e.g. theoretical upper limit of time for recovery of link failure in the number of nodes or for amount of communication regarding the nodes needed to be updated upon link failure. However, when regarding the dependability of routing algorithms, evaluation needs to be performed in the context of the applications and the performance requirements they specify. This quantitative evaluation of the routing algorithms and implementations is what is described here.

Based on the application performance metrics, defined in the application description in the scenario (Chapter 2 on page 9), the influencing routing metrics are described in the following. The dependability evaluation of the routing protocol ends up in measuring route availability and reliability with respect to the performance metrics.

The concerned performance metrics are listed here for reference.

- Throughput
- Delay
- Delay jitter
- Data integrity
- Packet loss

8.2.1 Performance metrics

To be able to evaluate routing performance with relation to these application requirements, investigation of parameters influencing on the application parameter needs to be performed. Hence, identification of the relevant metrics must be performed. Broch [Broch et al., 1998], which has compared ad hoc routing algorithms in a simulation environment, lists a set of metrics relevant for routing evaluation, which are briefly described below. The definition of the metrics are described as well as their relation to the upper layer metrics. Subsequently, an evaluation of how to obtain values of the metrics is presented. The relations between the upper layer performance metrics and the routing metrics is illustrated in Figure 8.1.

Packet delivery ratio is relevant as it describes the packet loss rate experienced by the transport layer. The ratio is defined by the amount of packets successfully delivered over the amount of packets attempted to be delivered. This makes it directly influencing on both packet loss rate and maximum possible throughput on the transport layer.

Routing overhead indicates the scalability of the routing algorithm, bandwidth consumption and power consumption. It is the fraction of bandwidth used for establishing and maintaining routes out of the total used bandwidth. Routing overhead influences delay as new routes may have to be discovered or influences bandwidth simply due to the need for additional control messages in the network.

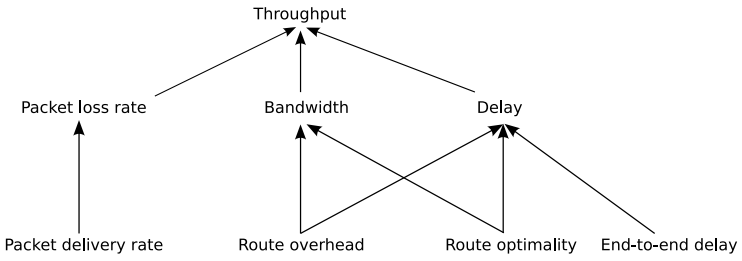

Figure 8.1: An illustration of a causal relations between the upper layer and the network performance metrics. Changes in the lower metrics affect the values of the upper metrics.

Path optimality is a metric describing the ability of the routing algorithm to select the optimal path for the route. The optimal route is relevant for the delay and bandwidth experienced by the transport layer. The optimal route is evaluated through comparison of the chosen route of the routing algorithm and the theoretical optimal route in the graph. Here minimum-hop-count can be used as optimality criterion, so that the actual number of hops is compared to the number of hops in the shortest path.

Use of minimum hop count as parameter has been challenged, as the shortest path not necessarily is the optimal, when also regarding packet delivery ratio and available bandwidth [Draves et al., 2004]. Instead, several other measures such as maximum flow or minimum delay can be used in the evaluation of path optimality.

End-to-end delay describes the overall delay imposed on a packet by 1) the individual links used on the end-to-end path, 2) the routing; for maintenance and route discovery and 3) by the end-node processing of the packets. End-to-end delay is measured on a packet basis from the packet is sent from layer 3 on the sender to the packet is received on layer 3 on the receiver.

8.2.2 Obtaining performance metric values

To be able to evaluate the performance of the different routing algorithms according to the identified metrics, measurement samples are needed. Hence, ways of extracting the measurements from the testbed need to be identified.

Packet delivery ratio: Can be measured by recording each packet drop and dividing by the recorded total number of attempted transmits.

Routing overhead: Recording the headers of the packets on the testbed, the bandwidth used for routing control messages can be filtered. This applies both to individual control messages and payload packets where control information is appended. The bandwidth spent on control information divided by the total used bandwidth is the routing overhead.

Path optimality: Depending on the concept of *optimal*, the measure of optimality differs. Should minimum hops count be used, then it can be calculated by registering the topology upon sending a packet and calculating the shortest path, either

in real-time or post-processing. From this, the number of hops can be counted and compared to the number of hops actually traversed by the packet. One problem here might be, that the shortest path available during sending may become unavailable when the packet is still in transit.

End-to-end delay: The emulator is emulating the layer 1 and Medium Access Control (MAC) of layer 2 of the network communication. This fact makes the top of layer 2 and layer 3 inaccessible for measuring from the emulator. Measuring on the end nodes introduces a problem with timer/clock synchronization, as the nodes are (probably) not synchronized in any way. Without synchronization, two samples, e.g. sender and receiver timestamps from which the delay could be calculated, will not be comparable to each other. Alternatively, the end-to-end delay is measured on the emulator, from receiving (on the emulator) the initial packet from the sender to delivering the very same packet to the receiver. This way, end-nodes processing is not considered in the delay, however, this addition can be measured using `ping` with a direct host-to-host connection. Using ping, only the processing time and the propagation time are seen, and having a propagation time very much smaller than the processing time, the processing delay is calculated this way.

As the focus of this project is on evaluating routing in the context of a video conferencing application, focus is on evaluating the *packet delivery ratio*, the *routing overhead* and the *end-to-end delay*. Maintaining as good a packet delivery ratio and as low an end-to-end delay as possible is evident, as also described in the pre-analysis, for optimal video conferencing. The overhead is considered as the amount of bandwidth used by a routing algorithm may disturb the application significantly.

8.2.3 Dependability metrics

Hollick [Hollick, 2004] defines dependability in a routing system as: "The trustworthiness of a routing system such that reliance can justifiably be placed on the consistency of behaviour and performance of the routing service it delivers."

To evaluate the dependability of the described routing algorithms, a set of dependability metrics are identified. as described in the preanalysis (Chapter 3 on page 15), dependability is comprised of several concepts; availability, reliability, safety, integrity and maintainability. These concepts can be divided into two groups; *technical* and *operational* concepts [Hollick, 2004]. Availability and reliability are contained in the technical group and the rest in the operational group. Since the focus in this work is primarily on evaluating quantitative aspects of the dependability and the main interest is technical, the group of operational dependability concepts is considered out of scope of this project. Moreover, as the success of a video conference is dependent on a continuous delivery of packets end-to-end, insurance of availability at a given point in time is not sufficient to provide dependable video conferencing. Therefore focus is set on end-to-end reliability in the following, as this encompasses availability and provides a continuous perspective on end-to-end delivery. In the following, the end-to-end reliability metrics as well as means for obtaining them are described.

End-to-end reliability

Reliability is typically described from availability, based on service failures. A failure can be seen as an unsuccessful event or result occurring. Service availability is defined

as the probability of a service being ready at time t_0+t given it was ready at time t_0, i.e. the system is ready to serve and is not failed at time t. Based on this, service reliability is defined as the probability of the service being available during an interval T, i.e. not failing within a period of time. End-to-end reliability is in [Mueller et al., 2004] specified as the probability of sending data successfully within a time window. However, this specification is based on path reliabilities, meaning that is uses the average link reliabilities to calculate a path reliability. The end-to-end reliability is then an expression of the probability that at least on path does not fail within a time window.

For routing evaluation, the availability and reliability is interpreted in a slightly different way. In the general case, availability and reliability are defined as measures of time. However, a routing service can be defined by the need of it, i.e. the need for delivering a packet end-to-end. The routing service is thus only active when actually sending traffic. Underlying mechanisms, such as described for OLSR, may continuously run to optimize the performance of the routing service, however the routing service itself can be seen as a discrete event-system. Defining the reliability of a routing service must thus be performed in terms of events and not time. This incorporates defining the failure of a routing service. As the purpose of the routing service is to deliver packets end-to-end, the failure of the system is defined to be when packet delivery fails in a routing perspective. Routing perspective means that frame retransmissions on lower level still may occur, but the routing algorithm fails to find an available path for a packet that needs to be delivered. End-to-end routing reliability can then be defined as the probability of successfully delivering a specific amount of packets, which is relatable to a service being continuously available for a certain period of time. The more reliable the routing service, the higher the probability of delivering a large amount of packets end-to-end.

To measure the reliability of a specific routing protocol, the number of successive packets delivered to the receiver is recorded and related to the number of packets transmitted from the sender. Based on this, the probability of successively delivering a certain amount of packets is calculated. Recording if received packets are in sequence is not trivial as they may be delayed differently during transmission. Moreover, there exists no notion of sequencing on regular IP-packet level. However, a method for sequential recording, along with the reliability results from DSR and OLSR implementations, are discussed in the Chapter 9.

Chapter 9

Routing evaluation

Having specified the algorithms to be used and the metrics for the evaluation of routing dependability, the following chapter proceeds with a description of the experiments performed using the developed framework.

First an experiment is described, which shows the difference in overhead for the protocols used, demonstrating the fundamental difference between proactive and reactive routing. The results from this test is used to conclude on some of the results from experiments described later in this chapter. Next, the experiments and evaluation performed to determine the performance of the routing protocols are described. Last, the protocols are evaluated with respect to end-to-end dependability.

9.1 Routing overhead evaluation

The two routing algorithms used in this project, DSR and OLSR, are fundamentally different in functionality, as previously described. DSR is a reactive protocol, only performing route discovery and route maintenance when requested, i.e. when routing packets. OLSR has an initialization phase of establishing routes and continuously maintains these connections by sending periodic *HELLO* packets. To experimentally show this difference, the amount of overhead produced by each protocol implementation is measured and evaluated.
This section describes the experiment performed to produce measurements of the routing overhead and subsequently presents the evaluation of these results. The experiment also provides a proof-of-concept and an implicit validation of the topology emulator as an analysis tool, as it requires large parts of the emulation-functionality to be available and working for performing the measurements.

9.1.1 Methodology

As a basis for evaluating the results of the experiment, two general expressions are used. These expressions are used for calculating the general control traffic overhead of routing protocols of the two families - i.e. reactive and proactive [Viennot et al., 2004]. The expression for the reactive routing family is

$$\lambda o_{\mathrm{r}} N^2$$

where λ is a *route creation rate*, o_{r} is a *route optimization factor* and N is the number of nodes in the network.
The expression for the proactive routing family is

$$h_{\mathrm{p}}N + o_{\mathrm{p}}t_{\mathrm{p}}N^2$$

where h_{p} is the *HELLO packet rate*, o_{p} is a broadcast optimization factor and t_{p} is the *topology broadcast rate*. Evaluating these two expressions shows a clear difference in overhead traffic. The overhead of a reactive protocol is dependent on the *route creation rate*, λ - i.e. the actual traffic routed in the network. Conversely, the overhead of a proactive protocol is independent of the traffic in the network. As it continuously maintains the routes in the topology, a constant overhead is experienced, which is also reflected in the expression by h_{p} and t_{p}.

To show that measurements for overhead evaluation can be drawn from the emulation testbed, these two distinct features of the used routing protocols are investigated using the testbed. By altering the packet rates produced by routing nodes connected to the testbed, it is expected that it is possible to extract the described features from data recorded in the testbed.
For this experiment a scenario is build and link properties are simulated from the scenario. The scenario is seen in Figure 9.1 and is a completely static scenario with no movement. Excluding movement from the scenario generates a clean result in terms of influences from excessive rerouting, as the available routes are the same throughout the emulation time.
The link properties (i.e. constantly available error-free links) are emulated in the topology emulator to which the three end-nodes are connected. UDP traffic is then sent as payload from *end-node 2* to *end-node 4* with different settings of packet rates representing the route creation rate parameter. This traffic is routed between *end-node 2* and *end-node 4* via *end-node 3*. All network traffic is logged into the database on the emulator node. In the database, filtering of the payload traffic is performed, leaving only the routing overhead behind. A simple summation over the size of the recorded packets in the test-period (which is 30 seconds in each case) reveals the total routing overhead at a given packet rate. Further details of the test-setup and scenario specification can be found in Appendix C. In this scenario, only the packet rates are varied. The remaining factors of the general overhead expressions, such as number of nodes, broadcast packet rates (of *HELLO* and *topology* messages) and optimization parameters are considered constant.

9.1.2 Results

Figure 9.2 illustrates the expected difference in routing overhead. It shows the dependence of DSR between the packet rate and the overhead generated to perform route discovery and maintenance. The figure also illustrates OLSR independence of packet rate as the OLSR overhead being constant throughout the packet rate variations. Figure 9.3 is a closeup of Figure 9.2, showing the interval interesting for evaluating the constant routing overhead of OLSR. At 0 pkts/s, the constant value of 0.32 kB/s is revealed. DSR, on the other hand, has a zero overhead when no traffic is present, as expected. Also, it is shown that the values of the OLSR overhead are not completely constant, but noisy from the announcement times of the algorithm. Nevertheless, the overhead measurements are concentrated around a constant mean over all test cases of 0.4 kB/s with a deviation of 0.06 kB/s.

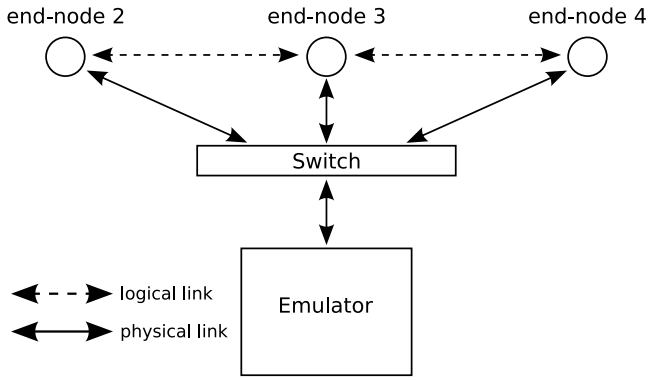

Figure 9.1: Routing overhead evaluation scenario featuring 3 end-nodes and multi-hop routing.

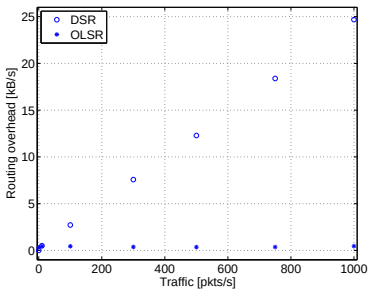

Figure 9.2: Evaluation results of routing overhead of DSR and OLSR for different packet rates.

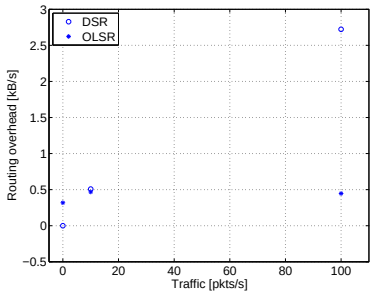

Figure 9.3: A closeup of the routing overhead evaluation results, illustrating the constant overhead generated by OLSR at 0 pkts/s.

9.1.3 Result evaluation

From this experiment several points can be made concerning end-nodes, routing algorithms and the emulation testbed.
It is shown, that the end-nodes implement the described routing protocols and are able to function in a multi-hop scenario. This is seen as the routing implementations provide actual routing of packets, i.e. end-to-end connectivity, from *end-node 2* to *end-node 4*. From the results, the qualitative expectation of the overhead produced by the routing algorithms is confirmed. The overhead of DSR is dependent on the packet rate of the ongoing traffic, while the overhead of OLSR is not.

In relation to the emulation testbed, this experiment provides a partial validation of the integrated functionality. The architecture supports connecting several end-nodes without modification to their software implementations other than that under test. The emulator node successfully handles forwarding of packets between separated end-nodes, as if they were 1-hop neighbors in a WLAN. Moreover, the logging mechanism implemented in the forwarding function is also shown to perform as expected, recording all relevant packet header information of the forwarded traffic. The information used in the evaluation contains time stamp, sender address, receiver address, packet length, and ingoing and outgoing VLAN of the emulator, which is readily provide by the database interface. Lastly, the use of a database for storing has proven beneficial in terms of searching and analyzing the recorded data, as only a few, simple SQL-queries are needed to produce the data-points for the evaluation.

Regarding the dynamic functionalities of the emulator such as updating the scenario by dynamically updating links onto real time forwarded traffic, and stochastically emulating properties of the links have not been validated by this experiment. These features are described in a subsequent section.

9.2 Routing performance evaluation

This section describes the performance evaluation of the DSR and OLSR routing protocols. Several test cases have been devised to test influencing factors on the routing performance, both from mobility and environment models. The main metrics of routing performance, end-to-end delay and packet delivery ratio, are investigated in terms of these factors.
In the following, test scenarios, test cases, results from the individual tests and evaluation of these are described. First, the characteristics of a real setup versus an emulated setup are considered, by applying the OLSR protocol in both scenarios and measuring the performance metrics. Next, mobility is introduced, using the scenario from the medical assistance domain described in the pre-analysis (cf. section 2.1) and the performance of OLSR in a static versus a dynamic setup is investigated. Then, results from DSR tests in the dynamic scenario are compared to the dynamic OLSR scenario to investigate the end-to-end performance of the two routing families, i.e. reactive and proactive. Additionally, tests investigating the influences of the environment models, i.e. loss and delay models, have been carried out. DSR and OLSR have been applied in the dynamic scenario, with simple and more realistic loss and delay models respectively, and results from these test are evaluated upon. Finally, a discussion of the test results and experiences is presented.

Detailed configurations of all tests are described in Appendix D.

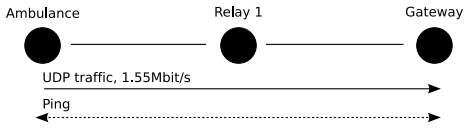

Figure 9.4: The static scenario used both in experimental and emulated tests. UDP traffic is sent from the *ambulance* to the *gateway* to measure packet delivery ratio and ping-packets are used to measure end-to-end delay.

9.2.1 Static scenario

As described in the pre-analysis (cf. section 3.3), setting up and repeating real experiments for evaluation purposes is a difficult task. This is mostly due to the diversity of the underlying processes like the wireless channel and movement patterns, which is the main motivation behind building the topology emulator. A real experiment is setup up to illustrate some of these diversities, in this case the wireless channel, and tests are run. For reference, an experiment similar in configuration is performed using the topology emulator to provide the link between the end-nodes. The results of these tests are described and evaluated in the following.

Scenario

Due to the complexity in setting up and performing real experimental tests, a relatively simple multi-hop scenario is defined. The scenario is depicted in Figure 9.4. The *ambulance* and *gateway* end-nodes are placed outside link range, requiring them to use *relay 1* as relay node. OLSR is used for ad hoc routing between the end-nodes.
To support measuring the same metrics in both experiments, only end-node measuring mechanisms are used. Detailed information of the traffic flows is available from the emulator for analysis, however, as such information is not available in the real experiment, the methods used only consider the emulator as a black-box, just as the real wireless links.
For payload traffic, a UDP stream is used of 1.55Mbit/s over 2 minutes from the *ambulance* end-node to the *gateway* end-node, resembling a constant bit-rate video stream. The amount of packets sent from the ambulance and received on the gateway is recorded to determine the packet delivery ratio of the routing algorithm. *Ping* is used periodically (period of 100ms) to probe the end-to-end delay between the *ambulance* and the *gateway*. Where nothing else is noted in the following experiments, delay corresponds to the RTT as measured by *ping*.

Results

Figure 9.5 shows delay measurements from three test cases; two from the experimental set up and one from the emulated set up. From the figure, large differences in both mean delay and delay variation is seen from the experimental tests. Also the timely characteristic of the delay differs significantly. In experiment one, very large peaks of delays over 500ms are observed whereas in experiment two, the variation is smaller, the largest peak being 65ms and the delay evolution appears more periodic. On the other hand, in the emulated test case, the delay is also stochastic but appears stationary.

Regarding packet delivery ratio of the routing protocol, Table 9.1 lists the percent-

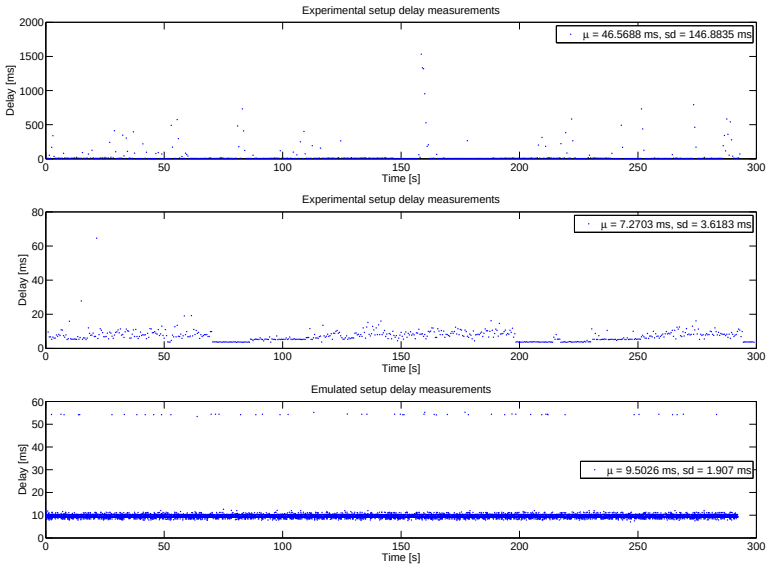

Figure 9.5: Delay measured in the static scenario from the *ambulance* to the *gateway*.

Test case	Payload packet loss
Experimental 1	3.10%
Experimental 2	0.00%
Emulation	0.59%

Table 9.1: Packet loss measured in the static scenario from the *ambulance* to the *gateway*.

age of lost packets, as measured by the *gateway*. In the second experimental test case all UDP packets where successfully delivered whereas in the other cases a small amount of packets are lost.

Result evaluation

The measurements of these experiments show that the properties of the wireless channel may vary largely between tests. Interpreting the results from experimental tests can be difficult, as these variations may be caused by many, independent factors. For instance, the delay peaks may originate from several factors within the channel, such as retransmissions due to packet losses. The delays could also be caused by modulation scheme changes, which alter the effective bandwidth of the channel. Alternatively, changes in the environment, like persons walking by or opening doors between the nodes can change the properties of the channel, enforcing the network adapters to adapt to this.

Very different results of measured packet loss are seen. The 0% packet loss result could be a result of very little background noise in the channel at that particular time.

Also, as the payload flow of UDP packets is only one way, collisions in the wireless channel are less likely to occur. The flow can be considered synchronized, meaning that the nodes send and receive the packets before new are generated at the source. However, as also ping packets are present in the traffic flow, which go both ways, it is surprising that all UDP packets are delivered.

The very different results of the two experimental test cases support the expectation that specific characteristics of the wireless channel are hard to reproduce for testing. Additionally, it has been observed during experimental testing that the task of creating a specific environment is cumbersome, especially with respect to a multi-hop scenario. For creating the described multi-hop scenario, a distance of 20 meters, incorporating 3 rooms and a hallway was employed to attain separation of the *ambulance* and *gateway*. In such a setup, the state of a door (i.e. open or closed) can rule whether end-nodes are separated or not. Coordinating test executions in such conditions is not trivial.

As a final evaluation of the experimental test case, a comparison to the emulator is made. When comparing the results, resemblance is indicated. Regarding both delay and packet loss, the emulated results are within the ranges of the experimental cases, which provides some validation of the topology emulator and the underlying models for emulating the wireless channel. However, complete resemblance is hard to attain (if not impossible, even to a single experimental case), as the parameters of the models used in the emulator are configured for an urban vehicular environment whereas the experiments were carried out in an in-door environment.

9.2.2 Dynamic scenario

The previous described test, incorporating OLSR for multi-hop routing, is performed in a static environment. However, VANETs are not static environments, and thus end-node mobility is introduced to create a more realistic environment for evaluation. For this, only the topology emulator is employed, as generating usable results from a real setup with movement for a routing performance evaluation is unrealistic. The purpose of the test is to determine the actual impact of mobility in terms of end-to-end delay and packet delivery ratio. In the following, a scenario is defined introducing link instability, from which results are compared to those of the previous described test with the static scenario.

Scenario

The scenario is depicted in Figure 9.6 and consists of 4 end-nodes passing an intersection. Initially, payload traffic is relayed by *relay 1* from the *ambulance* to the *gateway* during approach to the intersection. Meanwhile, *relay 2* approaches the intersection from west, arriving at the intersection center at the same time as *relay 1*. The nodes drive with the same speed, keeping distances unchanged. However, in the intersection, *relay 1* turns towards east, and can no longer be used as a relay node, leaving only *relay 2* to maintain the path between the *ambulance* and the *gateway* while all three nodes continue south. As OLSR uses *HELLO*-messages to discover neighbours, the nodes are set to run for approximately 45 seconds before and after passing the dynamics of the intersection to assure stabilization of the routing algorithm. Stream and measurement parameters are the same as in the static scenario.

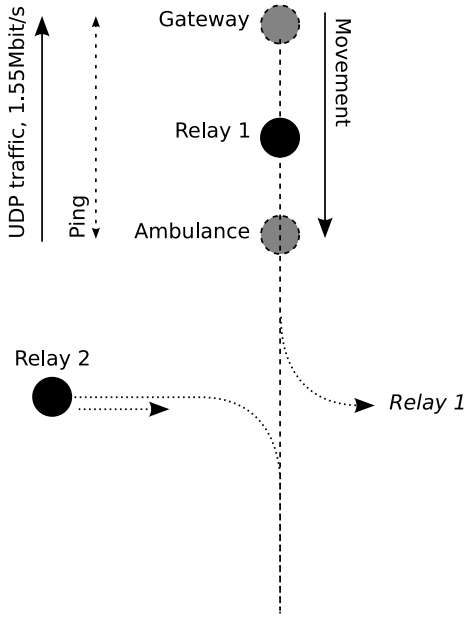

Figure 9.6: Intersection scenario of 4 nodes passing an intersection. *relay 1* turns east in the intersection and is replaced by *relay 2* that continues south with the other nodes.

Results

The tables 9.2 and 9.3 list the results of the comparison between the static and the dynamic scenarios. The delays have small differences, indicating limited impact of the link dynamics. However, only considering averages in this case does not reveal all facts of the end-to-end delay measurements. Figure 9.7 shows all the delay measurements from a test run. From the figure it is seen that during link instability, delay measurements are missing. Because of this, the delay averages are only to be interpreted as the stationary delay in a stable environment. Still, as nodes move in the intersection scenario, packet error rates change, enforcing a slight increase in the average delay.

The packet delivery ratio, on the other hand, is significantly decreased when links become unstable compared to the static scenario. Figure 9.8 illustrates the amount of packets sent per second from the *ambulance* to the *gateway* measured on the emulator over the entire period. A stream of 1.55 Mbit/s with a packet size of 972 bytes/packet resembles a packet rate of 200pps. This rate is doubled to 400pps, as packets traverse two links, and thus each packet is recorded independently twice on the emulator. As seen from Figure 9.8, the increase in packet loss is due to OLSR losing packets during the link instability when *relay 1* leaves and *relay 2* joins the network. When *relay 1* leaves, packets are still forwarded to it by OLSR from the *ambulance*. However, as the link to this node is not existing, these packet are not received for relaying.
Also, this loss of packets during link instability is what causes the loss of delay measurements.

Test case	Packet delay mean	Packet delay deviation
Static	9.50 ms	1.91 ms
Dynamic	12.09 ms	0.61 ms

Table 9.2: End-to-end delay measured from the *ambulance* to the *gateway*.

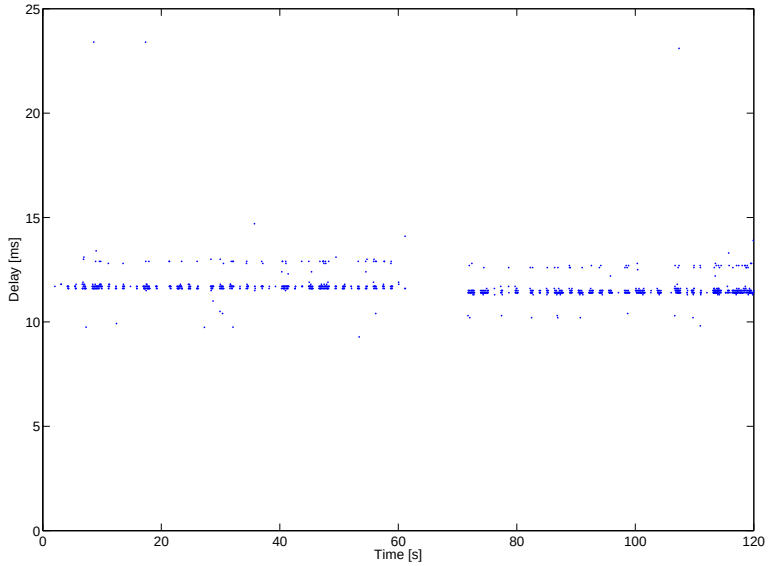

Figure 9.7: End-to-end delay measurements in the dynamic scenario from the *ambulance* to the *gateway*.

Test case	Packet loss
Static	0.59%
Dynamic	6.39%

Table 9.3: Packet loss measured from the *ambulance* to the *gateway*.

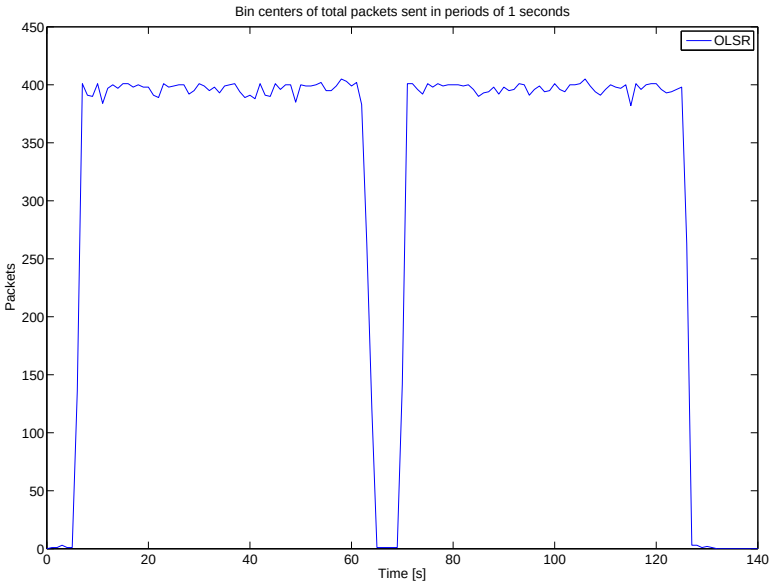

Figure 9.8: Flow of packets from *ambulance* to *gateway* in dynamic scenario. Packets are lost due to routing errors when *relay 1* leaves and *relay 2* joins the network.

Result evaluation

As shown by the results, the introduction of link instability due to mobility has an impact on the routing performance in terms of packet delivery ratio. Despite an always available path from the *ambulance* to the *gateway*, packets are lost due to the routing protocol not adapting to the link changes in time before the used link ultimately disappears. This is, however, not a definite measure of a performance degradation of OLSR. The OLSR implementation used in the test cases uses all default values, corresponding to the environment to which it was designed. In this environment, nodes move with a significantly lower speed (< 5m/s) compared to this scenario (15m/s), and it is thus expected by OLSR to have a longer period to react than is the case of this mobility scenario. The protocol parameter *NeighborHoldTime* has a default value of 6 seconds, which is the period in which OLSR trusts the latest *HELLO*-message (rate is 1 per second) from a neighbor and thus believes it to be present.
Figure 9.9 illustrates the conditions for this parameter setting in the intersection scenario. The movement generates a link change from a good link to a non-existent link between the *ambulance* and *relay 1* in just 2 seconds. Moreover, the new path between the *ambulance* and the *gateway* through *relay 2* only coexists with the old path through *relay 1* for 5 seconds. Even if OLSR tries to establish a new route using *relay 2*, the unreliability of the links to the joining *relay 2* results in a very low probability of routing packets getting through to *relay 2* and the *gateway*.

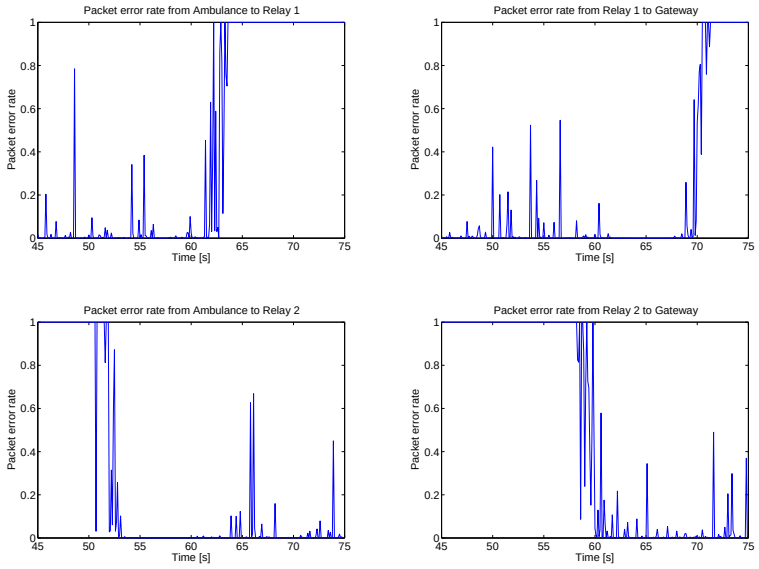

Figure 9.9: Link properties of the links between the nodes in the intersection scenario plotted around the time when they cross the intersection and link become poor and unstable.

9.2.3 Routing algorithm comparison

The previous test shows that link instability, for instance generated by node movement, has an impact on routing performance of OLSR. OLSR represents the proactive family of ad hoc routing protocols and DSR represents the reactive. To compare performance metrics of these two families with respect to mobility resilience, OLSR and DSR have both been tested in the intersection scenario, and the test results are described in the following. The scenario and testing mechanisms applied in this test are the same as in the previous tests.

Results

Figure 9.10 shows delay measurements from both DSR and OLSR in the intersection scenario. These two routing protocols show very similar delay values, yet, their characteristics differ during the link instability of the intersection. From the measurements, it is seen that DSR gets delay measurements sooner than OLSR upon link breakage. This is due to packet loss and is evaluated below. The delay means and standard deviations are calculated and listed in Table 9.4. Each mean is an average of 5 individual means from test runs of each protocol. The numbers confirm that the delays are similar.

The means and deviations of the packet loss measured for both protocols are listed in Table 9.5. From this it is seen that the mean packet loss of DSR is lower than OLSR. This, however, requires an investigation, as the means here also are averages of 5 individual means for each protocol. Plotting the individual packet flow measurements, illustrated in Figure 9.11 reveals that OLSR in some test runs manages to handle the

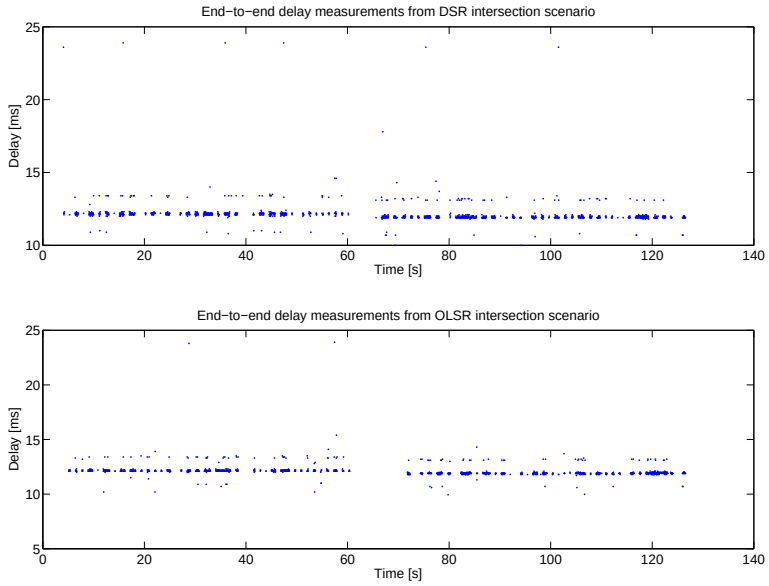

Figure 9.10: End-to-end delay measurements of DSR and OLSR from the intersection scenario showing an 'outage' of measurements during the link instability.

Protocol	End-to-end delay mean	End-to-end delay deviation
DSR	12.14 ms	0.01 ms
OLSR	12.10 ms	0.01 ms

Table 9.4: End-to-end delay means and deviations of DSR and OLSR from the intersection scenario.

link instability, resulting in a lower packet loss rate. This is the cause of the deviation size of OLSR compared to DSR.
The high variation of packets in the plot in Figure 9.11 is caused by all the delay measurements traversing the links simultaneously with the payload UDP traffic. Filtering the logged data of the emulator from these delay packets renders Figure 9.12. From this figure another property of the routing protocols is revealed, namely that DSR utilizes more packets than OLSR when routing. As DSR routes reactively updating routes as traffic flows, this is to be expected from the measurements, as also described in section 9.1.

Results evaluation

The mean end-to-end delays of the two routing protocols are almost identical. The slight increase of DSR is most likely caused by DSR introducing additional data into the IP-header, resulting in a larger packets to transmit, causing a longer transmission delays, combined with the fact the a delay measurement may be subject to a route discovery request, prolonging the delay.

Protocol	Payload packet loss mean	Payload packet loss deviation
DSR	4.93%	0.5%
OLSR	6.93%	2.9%

Table 9.5: Packet loss means and deviations of DSR and OLSR from the intersection scenario.

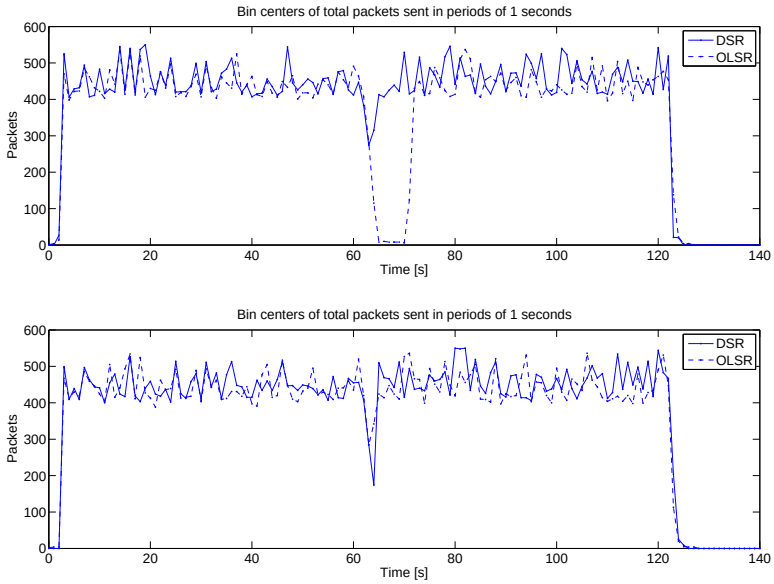

Figure 9.11: Packet flows of DSR and OLSR measured on the emulator showing that OLSR in some cases manages to avoid serious packet loss during link instability.

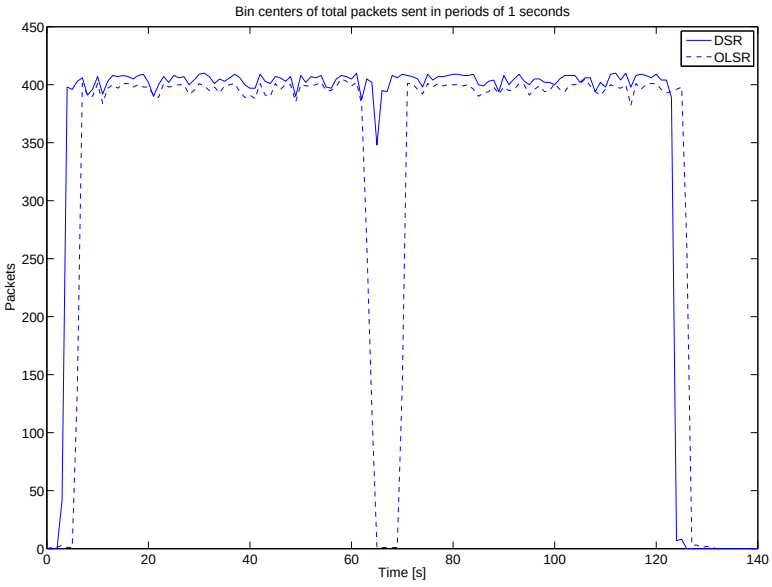

Figure 9.12: Packet flows of DSR and OLSR showing an overhead in packet rate of DSR over OLSR, but a more resilient behaviour during link instability resulting in a lower amount of lost packets.

When comparing the performance of DSR and OLSR in terms of packet loss, DSR appears to be superior to OLSR. However, as previously described, this scenario may penalize the performance of OLSR as the topology changes too rapidly compared to the specified parameters of the OLSR implementation. The fact that DSR seems to recover faster than OLSR from the link breakage is mainly because of the way DSR works. DSR's response time to link changes is dependent on the traffic. Due to the amount of traffic applied in the scenario and the traffic-rate utilized, DSR may have better conditions for recovering than OLSR.
The reason why OLSR in some cases manages to avoid packet loss during link instability may be due to (lack of) synchronization of the execution of the test. The OLSR-implementation is initialized on the end-nodes before executing the test, triggering each end-node to begin broadcasting for neighbors at different times. Depending on the combination of these broadcasts and the starting time of the scenario, OLSR sometimes manages to discover the link quality reduction and change the path for the end-to-end route before the link to *relay 1* disappears. This effectively reduces the packet loss of OLSR.

9.2.4 Model comparison

It has been shown that the two families of ad hoc routing protocols have different performance capabilities in a mobile ad hoc environment with unstable and unreliable links. The experiments also show that properties of the tests executions, the delay and loss models influence the outcome of the routing evaluation. To show the impact of such model settings, alternative loss and delay models are introduced and used for evaluating the routing protocols. Typically, when performing evaluations, assumptions are made in the underlying models. This test shows the impact of using simple loss and delay models compared to using more complex loss and delay models as described in this project.
In [Matthiesen et al.,], simple loss and delay models are used for evaluating application level service replication in urban mobile ad hoc networks. In the following, described, adapted and tested for comparison to results from the models used in this project.

Loss and delay model
The models described in [Matthiesen et al.,] are primarily delay models. Instead of using a packet error rate, a loss free end-to-end route is assumed. The existence of individual links are determined by a threshold of 60% on the bit error rate, resulting in an error-free link (with respect to packets) at distances up to 300 meters and no link at higher distances. The bit error rate is defined as an exponential funtion with a rate proportional to the received transmission power, which is calculate by the path-loss reduction on the used transmission power. The received transmission power P_r is defined by way of path-loss as

$$P_\mathrm{r}[dBm] = \begin{cases} 20\log_{10}(C \cdot d^{-\gamma} \cdot P_\mathrm{t}) & \text{if } d \leq 300\text{m} \\ 0 & \text{otherwise} \end{cases} \tag{9.1}$$

where the path-loss exponent $\gamma = 2$ and d is the distance.
The bit error rate is thus defined as an error rate on blocks of bits $BLER$ (i.e. a block error rate where a block represents a packet) as

$$\text{BLER} = \begin{cases} e^{-k \cdot P_\mathrm{r}} & \text{if } P_\mathrm{r}[dBm] \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{9.2}$$

Protocol	End-to-end delay mean	End-to-end delay deviation
DSR	371 ms	35.0 ms
OLSR	263 ms	0.433 ms

Table 9.6: End-to-end delay means and deviations of DSR and OLSR from the intersection scenario using the simple loss and delay models.

where $k = 0.32$ is a drop-off rate of the exponential function. The constant C in equation 9.1 is fitted appropriately so equation 9.2 yields 1 for $d \geq 300m$. This delay model assumes infinite retransmission and no wireless collisions. By considering the probability of having a bit error in a packet and thus causing retransmissions, the individual packet delay D is modeled as

$$D = D_0 + D_{\text{h}} + D_{\text{r}} \cdot \frac{BLER}{1 - BLER}$$

where D_0 is the initial delay to transmit a packet, D_{h} is a processing delay for a forwarding node, D_{r} is the delay induced by a retransmission and $\frac{BLER}{1-BLER}$ is the geometrically distributed number of failed retransmissions. Besides assuming possibly infinite retransmissions, the model also assumes independent packet errors in BLER for the retransmission number to obtain a geometric distribution.

The model has a 'cut-off'-distance at 300 meters. Packets are successfully received when the distance is lower than 300 meters, though delayed. This specific 'cut-off'-distance value is adjustable by the constant C. To match the distances used in the intersection scenario, C is fitted to match a 'cut-off'-distance of 150 meters. The distances from the *ambulance* to *relay 1* and from *relay 1* to the *gateway* are 110 meters. A 'cut-off'-distance of 150 meters yields separation of the *ambulance* and the *gateway*, and still maintains a path via *relay 1*. Also, as the emulator only mimics the actual link-delay, the node processing delay D_{h} is 0.

By implementing and deploying the described loss and delay models in the simulator module of the topology emulator, both DSR and OLSR are tested in the intersection scenario, and the results are presented in the following.

Results

From Table 9.6 it is seen that an average of mean delays for DSR and OLSR have close to similar values. However, the deviation of the DSR means is rather high. The reason for this is revealed when plotting delay measurements from a single test run of DSR, as in Figure 9.13. The upper figure shows that the values are concentrated around three levels, which is confirmed by the lower figure of the delay PMF. The center values from the 3 levels with high probability are; 263 ms, 392 ms and 524 ms. It is this concentration that causes the high deviation in the delay means. The specific values of the levels are interpreted in the subsequent evaluation.

Regarding the delay measurements from OLSR, they have a much smaller deviation, which is also confirmed when plotting them, as seen in Figure 9.14. Here, another property is revealed, namely that OLSR in some test runs is capable of handling the link instability of the intersection without losing delay measurements. However, as seen from Table 9.5, the average mean packet loss of OLSR is still larger than that of DSR.

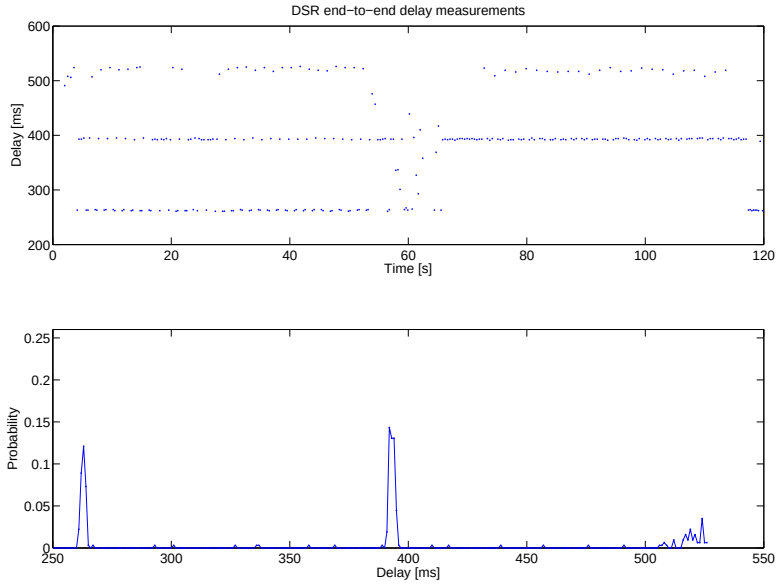

Figure 9.13: End-to-end delay measurements from DSR in the intersection scenario using the simple loss and delay models.

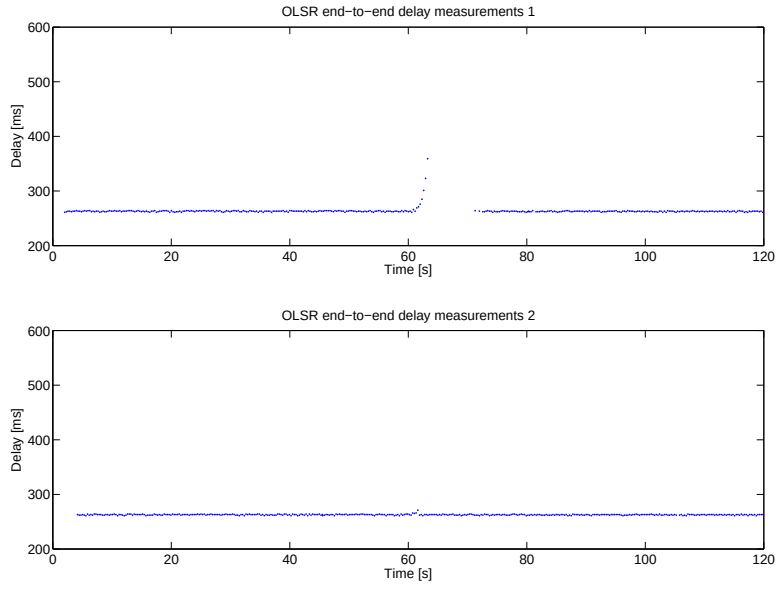

Figure 9.14: End-to-end delay measurements from OLSR in the intersection scenario using the simple loss and delay models.

Protocol	Payload packet loss mean	Payload packet loss deviation
DSR	0.158%	0.123%
OLSR	3.89%	3.56%

Table 9.7: Packet loss means and deviations of DSR and OLSR from the intersection scenario using the simple loss and delay models.

Result evaluation

Considering the characteristics of the delay measured from DSR with the simple model, the delay is classified in three levels; 263 ms, 392 ms and 524 ms. To evaluate the cause of obtaining these values, the imposed delays from the model are considered. At a distance of 110 meters a deterministic delay of 65 ms is imposed on each packet. Interpreting the level values in terms of this delay is interesting as $263 = 4 \cdot 65.75$, $392 = 6 \cdot 65.5$ and $524 = 8 \cdot 65.5$. The factorizations are interpreted as end-to-end delays of 263 ms consist of pure transmission delay, as each delay measurement traverses 4 links; 2 for *icmp echo request* from *ambulance* to *gateway* and 2 for the *icmp echo reply*. End-to-end delays of 392 ms originate from measurements that experience a *route discovery* on **either** the request or the reply, enforcing an increased delay caused by routing ARQ. Finally, end-to-end delays of 524 ms are from measurements that are subject to *route discovery* on **both** the request and the reply. The slight deviation of 0.5 and 0.75 from the 65 ms is due to rounding precision of the distance discretization in the emulator, causing the distances to be slightly off 110 meters at some points in time.

Concerning packet loss, several aspects of routing performance is seen from the tests. From the lower sub-figure of Figure 9.14 it is seen that OLSR in some cases manages to deliver all packets successfully from end to end. As also previously described, this may be due to synchronization issues of the test execution. However, the conditions under which the routing algorithms work are somewhat more optimal with the more simple model used in this test. Due to the assumption of an error-free link, OLSR has a larger probability of detecting the additional link to *relay 2* earlier, as the links to both *relay 1* and *relay 2* coexist longer and free of errors. This is exemplified in Figure 9.15. The upper sub-figure shows OLSR not reacting in time, losing packets while trying to reestablish the end-to-end route and the lower sub-figure illustrates a case where OLSR manages to detect *relay 2* before *relay 1* disappears.
Moreover, the low packet loss rates measured for DSR are also due to loss and delay model properties. As no packets are lost, all routing packets are successfully delivered and the routing algorithm thus has an optimal view of the topology constantly. The longer delays imposed by the delay model enforces DSR to perform route discovery often, as request acknowledgement timeouts occur frequently, resulting in a more topology-change-aware DSR algorithm. As seen from Figure 9.13, the impact of the link instability, in terms of packet loss, is almost non-existent.

9.2.5 Overall evaluation issues

Before describing the conclusions made from the conducted tests, a few issue concerning general evaluation in the framework are described.

Black-box implementations: Throughout the tests, the routing implementations have been considered black-box implementations. However, during testing, several issues regarding the implementations have been identified. DSR holds a buffer

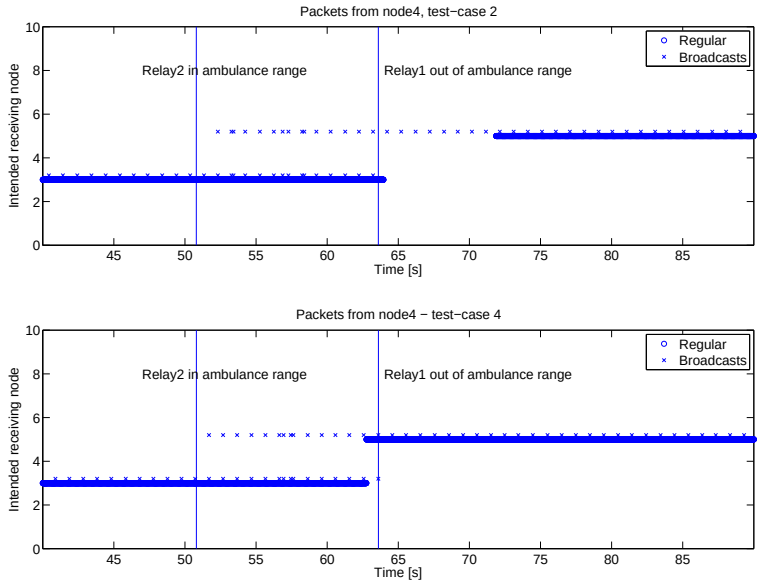

Figure 9.15: Packet flows as seen from the emulator, of packets flowing from the *ambulance*. Intended receiving node 3 is the *relay 1* node and node 5 is the *relay 2* node. The upper figure shows a case of OLSR not managing to use *relay 2* before *relay 1* disappears and the lower shows a case where OLSR succeeds.

of packets that are unacknowledged by the receiving node. In the DSR-UU implementation this buffer has a size of 100 packets. Also, the implementation has a *maintenance hold-off time* of 250 ms, meaning that packets are kept unacknowledged in the packet-buffer for 250 ms before re-sending. This effectively means that when sending packets with a rate of more than 400pps, this buffer overflows, when subject to large delays (>250 ms). The policy of the buffer is to discard incoming packets when full. In turn, if a packet is dropped on a link at high rates, the routing algorithm has no knowledge of it being sent, as it is not in the buffer, and thus no need for it to be re-sent. Consequently, if a link disappears, the route discovery routine is never initialized to discover an alternative to the broken path. Moreover, testing the boundaries of the protocols also reveal problems. As DSR adds extra information describing the used route to the IP-header, the maximum payload per packet is reduced. It was observed during testing that a serious problem exists when trying to send 1472 bytes (the normal maximum UDP payload) per IP-packet when using the DSR implementation. In several, reproducible, experiments, the laptops lock up and must be turned off and on again to resume operation.

Stateful routing algorithms: Both OLSR and DSR holds tables of available routes in a cache for different amounts of time. When executing multiple tests in the testbed it is thus required to restart each routing algorithm before each test run, to assure equal starting conditions for the algorithm. This is achievable, as the emulator blocks all link communication between end-nodes before executing the test. However, in a real experiment, no blocking occurs and equal starting conditions are thus difficult to attain. The problem can be relieved by letting the routing algorithm run a certain amount of time to stabilize, however, as also identified by the test results, synchronization issues of the algorithms still exist.

Hardware issues: For performing the real experiment, laptops with wireless network adapters were used. The settings of these are described in Appendix D. It was discovered during prototyping for the experiment that even though all used laptops employ an Intel Centrino chipset with 802.11a/b/g wireless network adapters, driver issues prevent certain adapter models to communicate. Specifically, the *ipw2200* and the *ipw3945* Intel Linux drivers are not compatible in ad hoc mode, and thus do not support inter-communication between network adapters. Thus for the experiment, only *ipw3945* drivers were deployed.

9.2.6 Conclusions of routing evaluation

Implementations of the DSR and OLSR ad hoc routing protocols have been subjected to several tests for evaluating the performance of each protocol in terms of end-to-end delay and packet delivery ratio. Details of each test result have been presented and evaluated. This section describes the conclusions made from these performance results.

Experimental versus emulated environment

From experimental test cases of multi-hop routing with OLSR the complexity of performing real tests is evident. Even with a static scenario, several issues regarding node separation, background noise and wireless adapter configuration leads to cumbersome

and impractical configurations. The parameters of the channel and the applied equipment have large variations from test case to test case making reproduction of a specific environment characteristics very unlikely. Considering setting up a scenario that incorporates node movement, like the described intersection scenario, and expecting the underlying environment to be even similar between test cases, seems highly optimistic. Additionally, the use of real applications, stateful or not, in the setup increases the complexity of the experiment as coordination and synchronization is very hard to maintain during test execution.

Comparing the coarse results of the experimental tests to similar cases performed on the emulator have shown somewhat similar results in terms of loss and delay, serving as a validation of the topology emulator as a whole. Still, complete reproduction of the in-door environment is not possible, as the underlying models of the emulator are developed for an urban vehicular environment. Additionally, these conclusions motivate for performing experimental routing tests on the emulator, as it presents a controlled environment for performance evaluation. Non-deterministic changes in characteristics of a real environment may render very different routing performance results which will bias the final evaluation of the routing protocols.

Static versus dynamic scenario

Parametrizing topology dynamics with respect to link stability has shown impact on the performance of the OLSR implementation. In a dynamic scenario with changing link properties and end-to-end paths, the OLSR implementation has a lower packet delivery ratio, despite the fact that an end-to-end path is always available. However, the measured performance degradation is not definite for the OLSR implementation. Evaluating the parameter settings of the OLSR implementation has revealed differences from the topology model for which OLSR was designed to the topology model used in these test cases, which may partially be the cause of an excessive decrease in packet delivery ratio. Nevertheless, the performance results are generated in a simple VANET scenario with relatively slow topology dynamics, revealing that care must be taken when setting parameters for the OLSR protocol if deployed in VANETs.

DSR versus OLSR

The two families of ad hoc routing protocols, i.e. reactive and proactive, have been represented by DSR and OLSR respectively for making performance comparisons. Overall, they both show to be affected by the topology dynamics of the dynamic scenario, meaning that the end-to-end performance both in terms of delay and packet delivery ratio is decreased. From the evaluation results, it is shown that the mean delays induced by the routing protocols are similar. The expected delay overhead of DSR, as a consequence of continuous route maintenance, is not significantly revealed. This suggests that the topology used in the scenario may not contain enough nodes for the route discovery delay to increase beyond regular transmission delays.
Regarding packet delivery ratio, DSR performs slightly better than OLSR, yet several parameters have been identified as impacting on the performance. Due to the amount of packets sent and the speed of the changes in the topology, DSR is favoured compared to OLSR due to the fundamental differences in routing methodology. Contrary, the expected overhead of DSR in terms of bandwidth due to the reactive routing methodology is recognized in the results. The conclusion of the comparison of reactive and proactive protocols is that more experimental research is needed, using larger topologies with more dynamics, combined with optimization analyses to establish optimal

parameters for each protocol implementation, in order to definitely conclude which performs more efficiently in a VANET context.

Simple versus complex loss and delay models.

A comparison has been performed of simple and the more realistic, and thus more complex, loss and delay models used in the project. This comparison reveals that not only does the models influence the traffic flowing in the topology upon operation, they also influence the performance of the routing protocols significantly. The simple model, incorporating typical assumptions such as infinite retransmissions and no wireless collisions and a simple transmission delay relation to packet errors and retransmissions, shows to trigger mechanisms in the routing protocols not experienced to the same extend when using the more realistic models. Large delays induced by the model forces DSR to perform route discovery more frequently, making the protocol more resilient towards topology dynamics but also making it produce more overhead in terms of delay and sent packets. Comparing the model outputs in terms of transmission delays and packets lost due to channel errors to measurements from the real setup, the complex model developed in this project appears more realistic. Then, comparing the results of the performance evaluations when using the two models shows significant differences in routing behavior. This shows that care must be taken when decisions on model complexity are made for making evaluations that rely on routing performance, as the individual model assumptions have significant impact on the performance and functionality of the protocols.

Conclusively, by being able to implement and deploy a completely independent model in the emulator shows that the interfaces created between simulator and emulator work properly as they generically support input generated by different types of models.

9.3 End-to-end reliability evaluation

This final evaluation section describes the evaluation of routing dependability. The routing capabilities with regards to end-to-end reliability have been evaluated according the framework described in the evaluation methodology chapter. The specific methods for recording data in a specific scenario, the reliability results and the evaluation of these results are described in the following. The evaluation also serves to illustrate the use of the topology emulator as a black box tool, since all measurements are performed on the end-nodes.
The previously described tests have been designed with relatively short term, rather deterministic, scenarios. This evaluation incorporates a more stochastic, stationary scenario, run for a long period of time, illustrating that the topology emulator is also capable of producing more general results.

Overall, the evaluation presents the difference in dependability, with respect to end-to-end reliability, of the two ad hoc routing families along with the inherent cost of providing dependability, in this case an increased end-to-end delay. Finally, the performance achievements of the two routing protocols are evaluated towards the application requirements set in the pre-analysis. These achievements are discussed to provide a perspective on the difference of the routing schemes and their usability.

Scenario

The scenario applied for evaluation contains the four nodes from the previous test cases and is depicted in Figure 9.16. In this scenario, the positions of the *ambulance* and the *gateway* are fixed and only the relay nodes move. The relay nodes move synchronously back and forth between the *ambulance* and the *gateway* creating a dynamic topology in the sense that the end-to-end path between the fixed nodes changes periodically from using *relay 1* to using *relay 2*.

Through this path, a stream of *ping* packets is sent from the *ambulance* to the *gateway*. *Ping* packets are of the Internet Control Message Protocol (ICMP) protocol and contains an incrementing 16-bit sequence number. Continuously sending ping packets in sequence enables measurement of consecutively received packets on the gateway. By observing the sequence number of the received *ping* packets, holes in the data stream can be detected as missing sequence numbers. The continuous flow of packets observed between missing sequence numbers is defined as one sequence. Thus the sizes of the sequences can be measured by counting the received number of consecutive sequence numbers. Additionally, this sequence numbering makes it possible to tolerate reordering of packets due to delay, as a simple sorting of the received sequence numbers before sequence length calculation eliminates the effect of reordering.

For recording the sequence numbers of packets, Wireshark is used in the case of OLSR. However, Wireshark does not, in the current version, support DSR packet dissection, so in case of DSR, *tcpdump* [tcpdump, 2007] is used, and the sequence number has to be parsed from hexadecimal representations of the raw IP-packets. Finally, as the sequence number field size of ICMP is only 16-bit, overflow of the sequence numbering is handled during data processing.

To test the stationarity of the scenario, two simulation runs are performed; one lasting 10 minutes and one lasting 60 minutes. The movement pattern of the scenarios is periodic. An illustration of one period of movement, lasting approximately 60 seconds, is depicted in Figure 9.17. The figure illustrates that during approximately one half of a period (30 seconds) *relay 2* can be used for routing packets from the *ambulance* to the *gateway* and for the remaining period of 30 seconds, *relay 1* can be used.

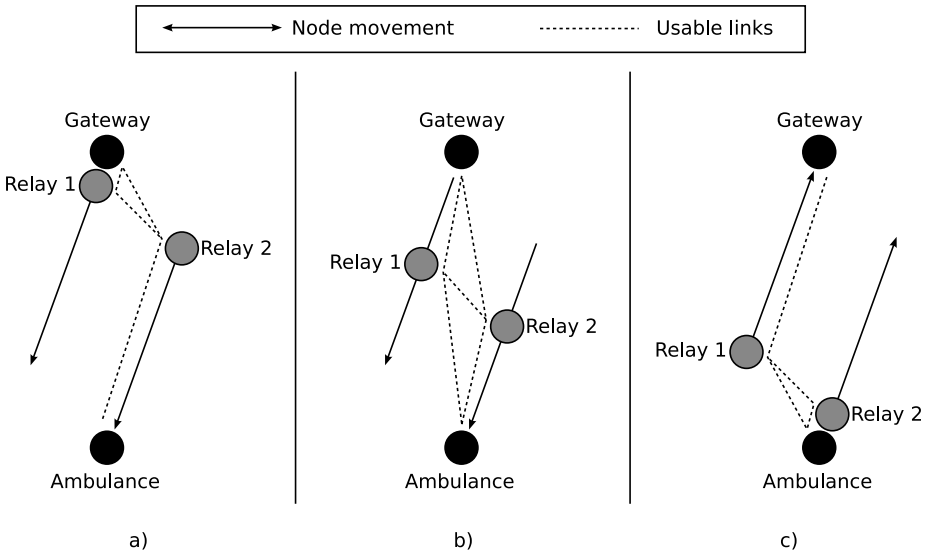

Figure 9.16: Scenario used for evaluating routing reliability. The *ambulance* and the *gateway* have fixed positions and the relay nodes move together switching between moving south-west or north-east. This creates shifting link availabilities enforcing the routing algorithms to adapt to the dynamic topology.

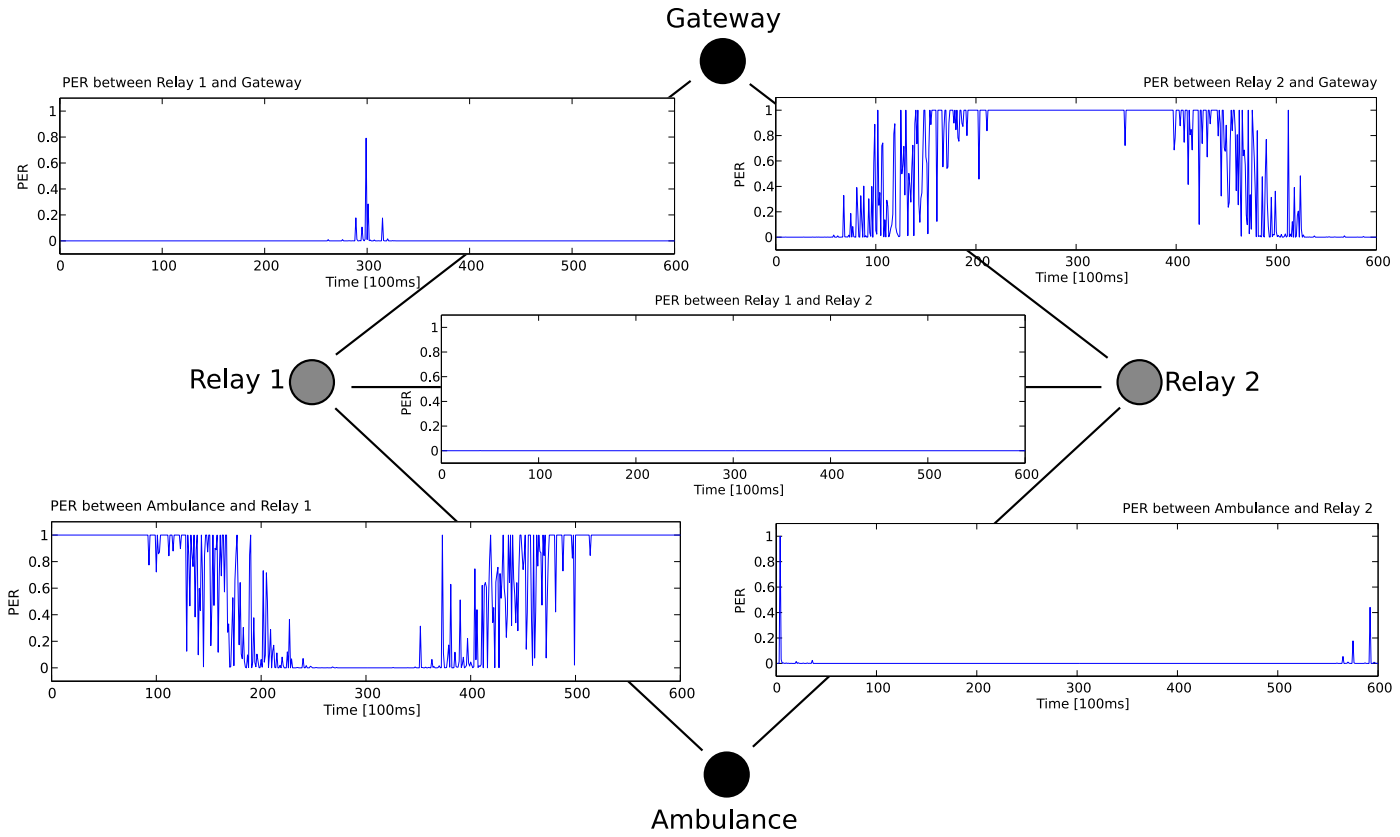

Figure 9.17: Links between the nodes of the reliability scenario. As the relay nodes move back and forth, the illustrated link properties are repeated over time.

Protocol	Runtime	End-to-end delay mean	End-to-end delay deviation
OLSR	10 min	11.506 ms	0.978 ms
DSR	10 min	17.179 ms	1.407 ms
OLSR	60 min	11.508 ms	0.970 ms
DSR	60 min	17.260 ms	1.179 ms

Table 9.8: Mean end-to-end delays of DSR and OLSR from experimental routing reliability tests.

Results

Overall, the packet losses of the two protocols have been measured and are listed in Table 9.9. This table shows that OLSR has a significant packet loss compared to DSR in the applied scenario. Moreover, it is seen that less packets are sent in total with DSR than with OLSR, which is due to a higher round-trip time, which is evaluated upon in the following.

The measured packet sequences are presented as histograms in Figure 9.18. All three sub-figures show the lengths of the different packet sequences observed on the first axis and the amount of sequences received with a certain length on the second axis. To emphasize the illustration over a large amount of sequence sizes, sequence length bin sizes of 50 packets have been used in the lower histogram.

The length of the packet sequences are calculated by counting the number of consecutive packet sequence numbers in a row. By interpreting a jump of more than one sequence number as a lost packet, a such jump determines the ending of a sequence and thus facilitates counting the total number of packets in a sequence. If a jump of 2 or more sequence numbers is detected, a sequence of length zero is detected, meaning that not even the first attempt of packet delivery was successful by the routing protocol.

From the upper sub-figure of Figure 9.18, it is seen that many of the attempted packet transmissions from the *ambulance*, while using OLSR, fail to be received at the *gateway*. This is illustrated by the high amount of zero-length sequences in the histogram for OLSR. By zooming in on the histogram, as in the middle sub-figure, is it seen that even though OLSR gets the first packet of a sequence through to the *gateway*, the length of many sequences, compared to DSR, is short. The low number of short packet sequences measured for DSR is explained by looking at the sequence lengths on lower values in a broader range. The lower sub-figure reveals that DSR manages to deliver significantly larger sequences to the *gateway*, despite the changing scenario. As also seen from the lower figure, one sample is observed of getting up to 3718 packets consecutively through with DSR. For comparison the longest measured sequence of OLSR is 1156 packets.

This property of DSR, to be able to maintain an existing route for a longer period of time, is caused by the mechanisms of DSR. These mechanisms have a cost in terms of end-to-end delay, which has also been measured during the tests. The results of these measurements listed in Table 9.8. From the table it is seen that the end-to-end delay is significantly lower for OLSR than for DSR. It is also seen that the scenario, whether tested over 10 minutes or 60 minutes, shows very similar influences on the measurements of end-to-end delay.

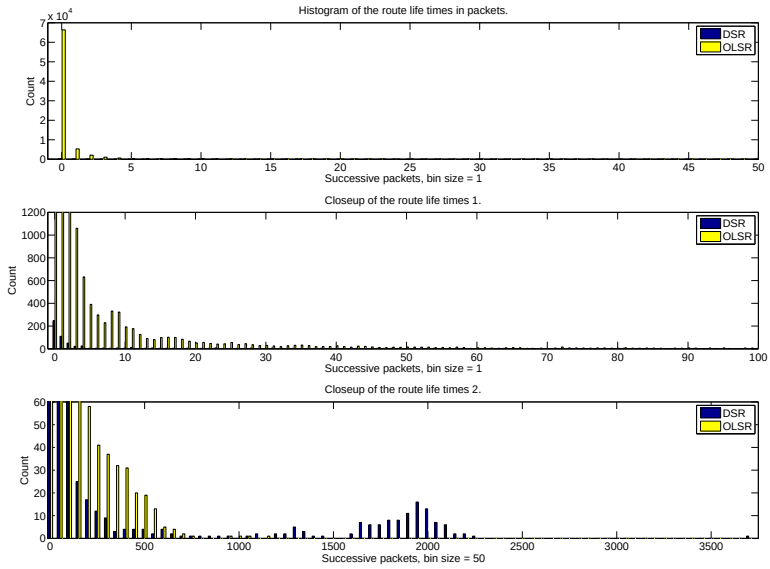

Figure 9.18: Histograms of the sequence length measurements of DSR and OLSR from the reliability test.

Protocol	Runtime	Packets sent	Packet loss
OLSR	10 min	49916	37%
DSR	10 min	42268	1%
OLSR	60 min	295952	38%
DSR	60 min	249896	0.8%

Table 9.9: Measured packet loss for DSR and OLSR from experimental routing reliability tests. The packets lost are measured by *ping* requiring both the request and the reply to succeed in delivery.

Protocol	MTTF [packets]
OLSR	2.70
DSR	262

Table 9.10: Mean time to failure of DSR and OLSR calculated in terms of packets sent end-to-end from routing reliability tests.

Result evaluation

Due to the lower round-trip time measured for OLSR, the amount of packets sent from end to end is also higher. As the rate is controlled by the *ping* tool, which in turn is controlled by the round-trip time, the rate varies with the round-trip time. *Ping* waits for a response, or at most 2 round-trip times, before assuming the *request* either a success or a failure.

Interpreting the length of the sequences as *time to failure* of the route facilitates calculation of the *mean time to failure* (MTTF) which can be used to express the reliability of the routing service. As the routing service is interpreted as discrete, the *time* is conceived as packets, so the MTTF is calculated as mean packets to failure when sending a stream of packets over a route.

The calculation shows that the MTTF for DSR is 262 packets and the MTTF for OLSR is 2.70 packets (listed in Table 9.10), revealing the DSR is significantly more reliable than OLSR in the specific scenario. OLSR does not reach stability and manages to establish usable routes before the changes in the scenario requires re-routing. Even though the scenario is dynamic, it is in the very low end of what to expect from a VANET scenario regarding link dynamics. The value of the DSR MTTF is mainly due to the periodicity in the movement of the relay nodes. Introducing more dynamics to stress the DSR protocol more in terms of reliability will reduce the length of the larger sequences. For OLSR, a more dynamic scenario will probably significantly reduce the probability of getting packets through at all.

The large sequence of 3718 packets observed with DSR is probably due to periodicity in the scenario as well. The long sequence is probably occurring over two periods where the link properties, due to the stochastic nature of the link properties, provides good conditions and allow for DSR to maintain the route over two consecutive periods. This can be seen from $\frac{3718}{2} = 1859$ packets which is also a relatively probable sequence length of DSR.

An interesting point in the evaluation process is to relate the metrics to the requirements set by the applications analysed in the pre-analysis. From the specification, the video conferencing application used in this project yield requirements to delay, delay jitter and packet loss. Delay must be below 150ms, the jitter of this below 50ms and the packet loss as low as possible.

Compared to the results attained in both the performance evaluations and this reliability evaluation, which also has provided some performance measurements, neither the delay nor the jitter seem to exceed the limits within the specified scenarios. These are, however, only limited scenarios with few nodes and confined movement. When the scenarios are scaled, e.g. in terms of nodes, links and mobility, the end-to-end delay is expected to become near the specified limits for the applications. Also, the difference in the protocol performances, even at this simple scenario level, must be elaborated upon. DSR shows an approximately 50% higher delay in a simple 2-3 hop scenario compared to OLSR. When relating this to a scale of scenario it is evident that DSR

reaches the limit before OLSR, in matters of delay.

Utilizing connectionless end-to-end data streams typically some packet drops are acceptable, as timing of reception is of greater importance, however, the packet loss of 37%-38% experienced using OLSR is intolerable even for a video conferencing application. Considering a connection-oriented application, even just considering use of TCP, results in a very low probability of succeeding with this amount of packet loss on the link. As TCP expects a relatively reliable communication channel, only interferred by congestion, massive packet loss induced by a wireless link has shown to render unusable throuhgput [Holland and Vaidya, 2002].

In the overall perspective, it is believed that using OLSR in dynamic large-scale scenarios severely causes packet losses compared to DSR, however, the end-to-end delay of OLSR may not increase as fast as for DSR when scaling in terms of nodes, links and mobility.

Conclusion

The evaluations of this chapter aim to show the differences in reliability of DSR and OLSR. From a defined scenario which incorporates a dynamic multi-hop topology that continues to vary throughout the simulation. Contrary to the previous test cases, the ability of the protocols to deliver consecutive sequences of packets has been measured. From these measurements it is clear that within the defined scenario, DSR has a clear advantage to successfully providing a routing service that is more reliable than OLSR. Even though OLSR periodically maintains its routes, the applied update periods are not sufficient to handle the dynamics present in the scenario and thus OLSR losses a significant amount of packets. DSR on the other hand experiences almost no packet loss, as the rate of which the packets are sent makes DSR highly aware of the continuous topology changes.

From these experiments and the results derived, it can be concluded that under certain dynamic scenarios DSR has an advantage over OLSR. Combined with the results previously attained from performance evaluation of the routing protocols it is concluded that this advantage is not a singular case of inappropriate scenario specification. Although both protocols have been tested only using their default configurations, as also discussed previously, the routing scheme of OLSR does not seem fit for handling highly dynamic scenarios, such as provided when dealing with VANETs. Several tests using different dynamic scenarios have shown to support this conclusion in this project.

Chapter 10

Conclusion

In the following the achievements and the conclusions which can be made based on the work performed in this project are described.

10.1 Achievements

Dependable services in the context of medical assistance applications in mobile ad hoc networks require dependable communication services such as routing. Development and deployment of such dependable solutions require reliable evaluation methods. This project has concerned development and deployment of an evaluation framework, including a topology emulation testbed, for performance and dependability evaluation of routing for MANETs.

Topology emulation testbed
For developing a topology emulation testbed, overall functional requirements have been derived. The topology emulation testbed emulates the properties of wireless links between several end-nodes, which are connected through a traditional Ethernet infrastructure. To ensure that the emulation is performed transparently, i.e. without affecting the overlaying applications, requirements to emulation performance have been specified as the foundation for the design. This design has been divided into two major components, namely a *simulator* and an *emulator*.
The *simulator* is responsible for simulating the properties of the links between end-nodes, while the *emulator* imposes these properties onto real ongoing traffic between end-nodes. To facilitate a realistic emulation, the simulated link properties are derived from detailed models of packet loss and delay. These models incorporate a high level of detail with respect to the wireless channel, the physical layer and the data-link layer. For link technology, a model of IEEE 802.11a, which utilizes OFDM on the physical layer, has been applied onto a statistical path-loss and shadowing fading model of the channel.
As input to these link property models, a graph based time discrete microscopic mobility model has been defined which provides realistic node movement. This model is capable of modeling both the urban and freeway mobility environments which comprise the context of the applications in this project. Every input to and output from the simulator is stored in files to facilitate support of externally generated movement or link property data for the topology emulator.

The *emulator* intercepts the data traffic between end-nodes and imposes the simulated properties on the emulated transmission links on this traffic.
All the components have been implemented and integrated into a topology emulator in a computer running Linux.
The functionality of each of the integrated components has been verified through extensive testing. The performance of the emulator has been verified with regards to the introduction of extra delay and bandwidth reductions while emulating, as well as the overall system-level implementation has been confirmed to meet the functionality requirements.

Routing evaluation framework

A framework has been developed for evaluating performance and dependability of ad hoc routing protocols. This framework defines the basis of an evaluation in terms of routing performance metrics. Performance metrics are constituted of *packet delivery ratio*, *routing overhead* and *end-to-end delay* along with the dependability metric *end-to-end reliability*. To provide a reference for evaluation results, requirements for the metrics have been specified based on a case, from the medical assistance scenario, where a dependable video conferencing application is utilized.

Routing evaluation

Two routing protocols, Dynamic Source Routing and Optimized Link State Routing, which are candidates for ad hoc routing protocols for MANETs as defined by IETF, have been selected for evaluation in the developed evaluation framework. These two routing protocols represent two different methodologies for performing ad hoc routing, a reactive and a proactive. Implementations of each protocol have been installed successfully on end-nodes in the topology emulation testbed. For routing evaluation, several scenarios have been specified. These scenarios include a static scenario without mobility, a dynamic scenario with deterministic mobility and a stochastic scenario with constantly changing mobility. The four scenarios provide facilities to evaluate the performance differences of the routing protocols when only the dynamics of the topology is a parameter. The *routing overhead* of both DSR and OLSR has been evaluated in the static scenario. The performances in terms of *packet delivery ratio* and *end-to-end delay* have been evaluated in both the static and the dynamic scenarios. The dependability of the routing protocols, in terms of *end-to-end reliability* was evaluated in the stochastic scenario. In the context of the static scenario, a real experimental setup was created and the static evaluation was performed in the real setup as well. Also, the influence of the link property model complexity and realism was evaluated. By using OLSR in a dynamic scenario and implementing a simple loss and delay model used for higher-level dependability evaluation, it was possible to compare the results to those of the complex model applied in this project.

10.2 Conclusion

The main motivation for developing a reliable emulation based evaluation framework for dependable solutions is the unreliability of the properties of a real experiment. This issue has been illustrated in this project by a comparison of an experimental setup and a similar emulated setup. The experimental setup provided two incomparable results, as if they were from two very different environment settings, despite the fact that they

were merely repetitions of the same experiment under identical conditions. Also, the execution of the experimental test has provided valuable insight to the difficulties and pitfalls of performing experimental work. For instance, it cannot be expected that two network adapters from the same manufacturer are able to communicate in ad hoc mode. Also, synchronization issues arise in managing tests in a system distributed among several laptops. Finally, the state behavior and properties of the applications used must be assured prior to performing evaluation. In the context of gathering results for the evaluation it has been discovered that Wireshark (formerly known as Ethereal) has problems capturing packets at packet rates above 15000 packets per second (pps) and that the routing algorithm implementations used have to reloaded between each test execution, to ensure comparable results, due to their statefulness.

10.2.1 Routing evaluation

From the tests performed in this project many interesting conclusions can be drawn regarding ad hoc routing protocols and their implementations. Initially, it has been established that the implementations are capable of performing ad hoc routing both in the case of a real environment and in the emulated case.

An inherent difference of the two applied routing algorithm families is that DSR uses the ongoing traffic to carry routing information whereas OLSR periodically updates neighbors with information about the topology. The result is that DSR produces a higher overhead than OLSR during high traffic load. Conversely, DSR does not produce overhead when traffic is absent, opposed to OLSR which continuously transmits topology information regardless of the network load. This traffic dependence of the DSR implementation has been verified using the developed routing evaluation framework and the results show a linear proportionality between bandwidth overhead and sent traffic.

The benefit of OLSR continuously transmitting topology information is a low latency when routing packets, since available routes do not have to be rediscovered as with DSR. This is optimal if routes do not change. As this is not the case in mobile ad hoc networks, the impact of dynamic links due to movement has been investigated.

The results of a simple dynamic scenario show that the end-to-end delay is similar for the two routing protocols, illustrating that the dynamics of the links in the scenario do not enforce enough re-routing for the benefit of OLSR to be significant. This has been further investigated using a more dynamic scenario over a longer period of time, and from this experiment it is seen that the mean end-to-end delay of OLSR is 11ms, whereas for DSR it is 17ms. These results readily illustrate the increased delay overhead of DSR during multi-hop routing. Comparing the results from the two scenarios illustrate the importance of creating an appropriate scenario for the metrics to be measured.
Moreover, the ability of the routing protocols to successfully perform routing, i.e. deliver packets end-to-end, has been evaluated in several scenarios. From all comparisons between DSR and OLSR, DSR has the highest packet delivery ratio. A general conclusion from these tests is that DSR is faster to recover from link breakages from topology dynamics and thus is capable of delivering more packets end-to-end. In a simple dynamic scenario, a difference in packet loss of 2 percentage points is measured, which is relatively small. However, when testing in a more dynamic scenario a difference of

37 percentage points is observed. This is a very high difference, as the results show that DSR losses 0%-1% packets in a such scenario. From the packet delivery tests, the conclusion is drawn that OLSR is not fast enough to settle between link changes and thus performancs much worse when delivering packets end-to-end.

Regarding dependability, the end-to-end reliability of the protocols has been evaluated. The end-to-end reliability is regarded as the mean-time-to-failure (MTTF) for a routing service in the sense that packets must be delivered consecutively, and a failure is when a packet is not delivered end-to-end. The time is thus interpreted in steps of packets. From the results, it is concluded that DSR provides a much more reliable routing than OLSR. The MTTF for DSR has been measured to be 276 packets whereas the MTTF for OLSR was 2.70 packets. Where the DSR is more reliable OLSR provides a lower end-to-end delay, which for OLSR is 11ms and for DSR is 17ms in the scenario.

From the results produced in this project, it seems straightforward to conclude that DSR is the better choice for routing in a dynamic ad hoc network. This is true in the sense that DSR is the more reliable protocol compared to OLSR, however, if the decisive parameter is delay, then OLSR should be chosen. In terms of routing methodology, the results indicate that the reactive approach is the more appropriate for highly dynamic ad hoc environments. Proactive protocols need a very high update period, which makes them flood the network, potentially producing overhead worse than reactive protocols. However, as the results also show, the properties of the scenarios used have significant impact on the performance of the protocols. More scenarios, preferably employing a more stochastic nature of topology changes, need to be applied for further testing to substantiate these conclusions.
Also, it must be taken into consideration that throughout the tests, both protocols implementations were used with default settings. As the domains for which the protocols were designed do not completely match the scenarios applied in this project, these default settings also have an impact on performance. However, it must be emphasized that the scenario parameters used in this project are on the low side of potential values for representing highly dynamic mobile ad hoc networks. For instance, the speeds of maximum 50km/h that have been applied here may reach 150km/h or more in real scenarios. Consequently, such settings would seriously affect the performance outcome of routing evaluation of DSR and OLSR. Finally, the scale of the scenarios is relatively small compared to the context of medical assistance applications. The maximum number of nodes used in the tests is 4, but can easily scale to 10 to 20 nodes or more when a more realistic vehicular topology is considered. Scaling the topology significantly increases the dynamics of the links, entailing less stability and reliability. The number of hops from end-to-end is expected to increase as well. Further research on these scaling issues needs to be performed to establish their impact quantitatively.

10.2.2 Topology emulation testbed

As a backbone in the evaluation framework, the topology emulator has been developed from the requirements specified which cover both performance and functional aspects.

By using the topology emulator, it has become possible to reproduce properties of scenarios for testing, which is difficult in real experimental setups. The topology emulator employs a stochastic, yet stationary, emulation, which facilitates realistic emulation The storing of movement and link properties facilitates repetition of a given test case to provide stochastically similar test environments.

Furthermore, the topology emulator provides transparent emulation as seen from the applications on the end-nodes on the network layer and above. This makes it applicable in many other experiments than routing evaluation, as long as the applications or services do not depend on response from functionalities below the network layer.
Additionally, the topology emulator provides a non-invasive timed logging facility of all packets sent within the emulated networks by the end-nodes, which provides a centralized and synchronized addition to the test results measured on the end nodes. The logs from the emulator are valuable for debugging experiments or for tracking data that is not otherwise available to the end-nodes, such as the route traversed by individual packets.

As the logs contain all packets from the network, also the dropped packets are recorded, which may help provide insight for evaluating application behavior when testing with unreliable links between end-nodes. This way, the consequence of a dropped packet can be tracked from the emulator, as the exact time of each drop is recorded.

For emulation, the topology emulator employs detailed simulation models of the the physical and data link layer.
The impact of varying the realism of these model has been investigated to illustrate the impact in routing performance under different assumptions in the models. These results show that care must taken when specifying models and assumptions for performing application evaluation. It has been shown how simple assumptions such as infinite retransmissions in the data-link layer and no collisions in the wireless channel have significant impact on the performance of the routing algorithms. Compared to the results created with the more realistic models derived in this project, the simple model is inadequate for routing evaluation.

Through performance testing it has been shown that the emulator is capable of emulating a wireless network without biasing the ongoing traffic by inducing additional delay or reducing link bandwidth. From tests, the performance boundaries of the emulator have been established. The maximum sustainable rate of packets has been measured to 80000pps over 10 minutes, supporting bursts of up to 250000pps over periods up to 10 seconds. As the design-maximum of end-nodes is 20, it has been calculated that full wireless emulation of links between all 20 nodes is possible. The emulator is subject to a problem of independent network partitions, in which independent channels of equal bandwidth are utilized. It can be calculated that if all nodes in a set of partitions broadcast periodically, minimum supported broadcast periods exist. If these are violated, the emulator may induce additional delay to packets. For 20 nodes in one partition, the minimum broadcast period is 5ms, whereas if they are separated into 10 partitions, the minimum broadcast rate is 0.5ms. As these periods are very small, it is concluded that the emulator is capable of fully emulating nodes in 1 to 10 partitions.

Common to all the performance results is that it has not been possible to fully inspect the boundaries of the topology emulator, so the described limits may not be the actual maximum limits. The tools used during testing, hereunder laptops with a traffic generator and Wireshark for packet capturing, were not able to reliably stress the emulator with sustained high traffic loads. Hence, more reliable equipment must be employed to better assess the performance boundaries of the emulator.

Implementing the topology emulator in Linux has illustrated some of the issues

which arise when developing real-time application in ordinary Linux. Linux is a soft real-time operating system that has no guarantees for hard deadlines in terms of process completion time. As the topology emulator employs a packet scheduler for delaying packets and is expected to be able to update link properties at a certain rate, such guarantees are beneficial for successful and accurate emulation. As this is not possible in Linux, the real-time capabilities of the emulator have been thoroughly tested to assure the real-time performance.
The fastest attainable periodic timer in Linux operates at a frequency of 8192Hz. This means that the minimum period-time in the processes developed is $122\mu s$. The consequence of this is that it is not possible to emulate link delays close to or below $122\mu s$. However, as the complete service time of the emulator from end-node to end-node has been measured to be $287\mu s$, this will never be a possibility. An additional consequence of the timer is that link property updates cannot be applied faster than once per 1 ms, due to timer granularity and jitter.

The emulation is based on off-line simulated link properties, because on-line simulations on packet basis requires to much processing power to succeed. One consequence of this, is that the assumed network load in the models are specified during simulation and not the actual on-line traffic forwarded by the emulator. As the link property simulation is dependent on the mobility model, the movement is also simulated off-line. This means, that applications that are supposed to affect the movement of individual nodes is unable to do so. Another consequence of this off-line movement simulation is that end-nodes are not aware of their positions during emulation.

Although the current limitations reduce the application domain of the topology emulator, it can be concluded that the concept of a topology emulator provides a necessary means for enabling experimental evaluation of real-world applications in a controlled environment. The evaluation results of this project show that the current version of the implemented topology emulator provides an applicable representation of the concept.

Chapter 11

Outlook

Several issues and limitations have been identified throughout the development of the routing evaluation framework. These are divided in two major categories regarding 1) the topology emulation testbed and 2) the routing evaluation and dependability evaluation in general. The identified issues and limitations are discussed in following providing an insight on the future for the evaluation framework.

11.1 Topology emulation testbed

In the conclusions on the topology emulation testbed, several limitations of the current version of the testbed are described. These are discussed to provide an overview of the tasks necessary to partially or completely overcome the impact of the limitations.

The end-nodes in the testbed are described as passive, meaning that they have no notion of their current position related to the movement simulated prior to performing emulation. A such position-awareness could be needed in applications where the nodes adjust their position as a consequence of changed link properties. Moreover, the nodes are required to be independent of the link technology used for wireless communication, as this is exchanged for Ethernet when using the emulator. This way information of link failures or received signal strength observations are unavailable to the end-node applications.

By implementing a *control channel* between the emulator and the end-nodes, position and link information may be exchanged continuously. This would require the emulator to appear as a node on the network and be able to transmit packets to the end-nodes, voiding the transparency requirement. These packets may however be disguised on the network and captured in the link layer by an interface dissecting the information from the packets and providing the information to the end-nodes through virtual interfaces, e.g. a fake WLAN interface for signal information. Exchanging position information from the end-nodes to the emulator would require the emulator to facilitate *online simulation*, as the nodes themselves control the change of the positions. This way, the link properties will have to be re-simulated continuously, as the nodes 'move'. Online simulation has been opted out in the current design for performance reasons, so the emulator is capable of emulating link properties transparently towards the end-nodes. Recalculating the link properties continuously require additional computation power, which may delay packets inappropriately. Hence, research of the applicable models and their complexity need to be performed to establish the feasibility in providing on-

line simulation support.
Moreover, a control channel in the testbed would provide for less difficult experiment execution management, as a central point for *experiment control* at the topology emulator GUI would exist.

As a general note, the *user friendliness* of the testbed is not currently at level were it can be widely deployed. Thus, a framework to ease the management of tests and results should be devised. This would consist of tools and scripts, possibly controlled from the GUI, for aiding the execution of tests and the post-processing of test results recorded on the emulator node.

As well as the emulation is independent of the online 'movement' of the end-nodes, the simulated link properties are modeled based on a predetermined traffic rate. To be able to support a wider range of evaluations, the online simulation should be extended to include *traffic dependence* as well. This means that the link properties are calculated based on the actual traffic flowing through the emulator. As the traffic amount aggregated at the emulator node accumulates, also here the feasibility of online simulation should be investigated. Several solutions exists ranging from calculating new link properties per arrived packet, over aggregating packet flows into means by which the simulation is adjusted, to pre-simulating several traffic loads and adjusting the link properties based on online measurements of traffic load. If comprehensive online simulation is needed, still, the computational power available is limited, and the complexity of the models must be revised, possibly resulting in a decrease of realism in the link property models applied. This trade-off must be investigated further before a solution to making the emulator traffic dependent is chosen.

Regarding the performance evaluation of the testbed, the equipment at hand during the project did not facilitate sufficiently precise benchmarking of the capabilities of the topology emulator. To completely assess the boundaries of the delay and the packet rates of both the topology emulator and the entire testbed, *dedicated test equipment* must be used, which is capable of creating loads on the emulator near the limits of the hardware employed in the emulator. Alternatively, a *model-based evaluation* approach could be used. By modeling the emulation testbed as a queueing system, and using the service times and limits measured in this project, an assessment of the maximum emulator capabilities can be attained.

As a final aspect of the emulation testbed, the support for domains additional to the ad hoc domain should be considered. Most of the HIDENETS scenarios are constituted by ad hoc and infrastructure domains. For the emulator to be used in such *hybrid network* scenarios, the ability to emulate high bandwidth and low error-rate links between static nodes must be evaluated.

11.2 Evaluation framework

The result evaluations and conclusions reveal that *scenario properties and configuration* play a significant part in the characteristics of the results. For instance, periodicity in movement may cause favorable conditions for one test subject over another.
To fully evaluate the impact of scenario parameters, these should be systematically adjusted and applied for testing to reveal dependencies between observed results and spe-

cific parameter settings. Such parameters include (but are not limited to) scenario scale (nodes, map size) and model parameters such as modulation scheme, packet lengths, applied transmission power and other environment properties such as fading constants.

Moreover, the routing evaluation results are specific to the routing protocol implementations used for evaluation. To be able to generalize the conclusions made on the different routing protocol families, more *routing protocols* should be evaluated within the framework and *different implementations* of these should also be applied to investigate the impact of the difference between the implementations of specific protocols.

Finally, the dependability evaluation of routing protocols has only concerned single-path routing protocols. Naturally, further work should be performed to assess the impact of using *multi-path protocols* in the context of mobile ad hoc networks. The topology emulator readily supports this either by implementing models supporting the protocols, or by the online simulation extension. With this extension, no further alterations are necessary to the methodology of the evaluation framework or the topology emulator as a tool.

Appendix A

Prototype configuration

This section describes the first implementation setup of the topology emulator, including specification of switch and emulator node hardware.

A.1 Network setup

The prototype emulator has been implemented as a bridge, as illustrated in Figure 6.2. This means that it is invisible to the other nodes in the network. The nodes are connected to a single port on the switch which is separate to all other ports by a VLAN. A *trunking* port on the switch gathers traffic from all these VLANs. Based on the VLAN and trunking settings on the switch, all nodes use the bridge for transmitting their Ethernet traffic. The bridge receives this traffic on only one interface. An issue in relation to network setup, is that the Ethernet channels used from the nodes to the switch are 100Mbit/s channels, which does not correspond well to the 54Mbit/s theoretical max of a WLAN 802.11a/g channel. However, within the context of streaming applications, bandwidth utilization is not predicted to be as high as 54Mbit/s - the streams used are around 1Mbit/s. This argues, that the 100Mbit/s Ethernet channel can be used as a substitute for the 54Mbit/s WLAN channel, as the data stream will never overload the link. This way, the excess bandwidth available on the Ethernet channel is not utilized for payload traffic.

A.2 Hardware setup

The hardware used for the topology emulator consist of:

- Cisco CATALYST 2950T 24-port switch featuring
  - 802.11q, VLAN - for separation of broadcast domains, no VLAN1, only port 2-23 are used
  - Gigatbit uplink, for maximization bandwidth towards the emulator node
- Emulator node
  - 1.66 GHz Intel Core 2 Duo processor
  - 3 GB RAM, for holding everything off the hard drives during real-time emulation

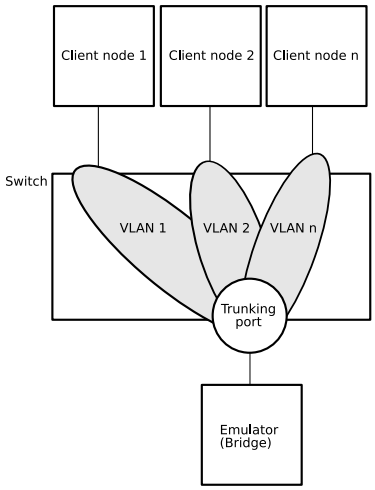

Figure A.1: Prototype implementation of the emulator testbed.

– Intel PRO/1000 MT Server Adapter, individual processing and buffering in stand-alone NIC, i.e. not integrated on mainboard for off-loading the CPU.

Appendix B

Emulator functionality verification

This appendix describes the verification of the implemented features in the emulator node. The emulator features are:

- Control of the logical links between the end-nodes by bridging
- Dynamically controlling individual link existences from the simulation output
- Setting properties from the simulation output on the individual links.

In the following, black-box tests and result evaluations of each feature are described.

B.1 Bridging verification

By controlling virtual LAN connections, the emulator is designed to be able to control the actual link connections of the connected end-nodes. This has been tested and these tests are described in the following.

B.1.1 Test setup

The emulator node receives packets with a 802.1Q VLAN tag containing a value equivalent to the port a given end-node is connected to in the switch. For reference, the physical connections, network and VLAN settings are given in Table B.1. As the default VLAN tag on port 1 is *vlan0001* on the Cisco Catalyst 2950T switch and cannot be changed, the tag and the port has been avoided during this project.

The primary tools for testing the functionality are *ping* and *Wireshark*. During all tests, *icmp echo requests* are broadcasted from a node using *ping* and on all nodes Wireshark records if and when a request is received and the corresponding reply is sent. A prerequisite for execution is that the nodes are configured to reply to broadcasts packets. Four test cases are configured, which are described in the following.

T1 - all in range: Nodes 2-4 are connected to the switch. The emulator is configured to fully connect all VLANs. *node2* broadcasts an echo request.

Node name	IP-address	Switch port	VLAN tag
node2	10.0.0.2	2	vlan2
node3	10.0.0.3	3	vlan3
node4	10.0.0.4	4	vlan4
node5	10.0.0.5	5	vlan5

Table B.1: Configuration of the end-nodes used to verify the bridging functionality of the emulator node.

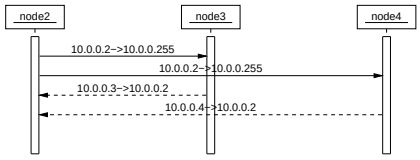

Figure B.1: Resulting traffic from bridging test case T1.

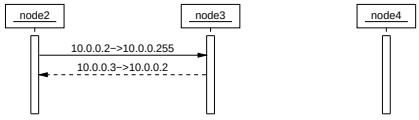

Figure B.2: Resulting traffic from bridging test case T2.

T2 - one out of range: Nodes 2-4 are connected to the switch. The emulator is configured to fully connect *node2* and *node3*, but not *node4*. *node2* broadcasts an echo request.

T3 - two partitions: Nodes 2-5 are connected to the switch. The emulator is configured to fully connect *node2* to *node4* and fully connect *node3* to *node5*. *node2* and *node3* broadcast echo requests.

T4 - asymmetric links: Nodes 2 and 3 are connected to the switch. The emulator is configured to connect *node2* to *node3*, but not *node3* to *node2*. *node2* broadcasts an echo request.

For all test cases the connectivity of the end-nodes is controlled only based on the simulation output files for the emulator and no physical changes are performed.

B.1.2 Test results

The results of the four test cases are depicted as sequence diagrams in figure B.1 to B.4, which have been generated based on traces from the test cases.

T1 - all in range: From Figure B.1 it is seen that *node2* broadcasts a ping, which is forwarded by the emulator node so both nodes 3 and 4 receive it. As they both reply, *node2* receives replies from both nodes.

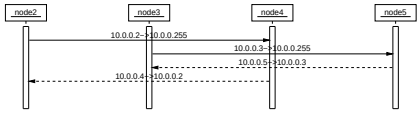

Figure B.3: Resulting traffic from bridging test case T3.

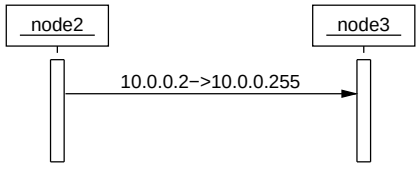

Figure B.4: Resulting traffic from bridging test case T4.

T2 - one out of range: From Figure B.2 it is seen that due to the missing link, *node4* does not receive the broadcasted request from *node2*, and thus *node2* only receives reply from *node3*.

T3 - two partitions: From Figure B.3 it is seen that due to separation of the links between the nodes in the emulator, the four nodes appear to be in logically separated broadcast domains. As *node2* makes a request, the only receiver and replier is *node4*, as well as when *node3* requests, *node5* is the only receiver and replier.

T4 - asymmetric links: From Figure B.4 it is seen that the link between *node2* and *node3* appears asymmetric as the request is received and replied to by *node3*, but the reply is never received at *node2*, and hence the logical connection between *node3* and *node2* appears non-existing.

B.2 Dynamic link update verification

As described in the design section, a reading-process is triggered by a timer to read the simulation and feed it to the *ebtables_emulation* kernel module, featuring dynamic updates of link existences and properties. The timeliness of the reading process is verified and accepted, however, it has to be verified that the properties are actually imposed onto the traffic, i.e. applied in the kernel by the *ebtables_emulation* module. This must be done in a timely manner, meaning that the service time of applying the properties must be lower than the period of property update arrival. If the update arrival rate is larger than the service rate, the system is unstable and as a consequence will drift, i.e. the emulation is slowed down. To test that this is not the case, a single black-box test conducted, which is described in the following.

B.2.1 Test setup

Opened-closed link End-nodes 2 and 3 from the setup described in Table B.1 are used. *node2* sends a constant stream of small UDP packets (payload = 1 byte) to *node3* at a rate of 2000pps. Meanwhile, the topology emulator is set to dynamically change the link existence between the end-nodes. Once per second the status of the link is toggled, i.e. from open to closed or from closed to open. This should effectively prevent packets going from *node2* to *node3*, representing a broken link. The test is run over 120 seconds, i.e. 60 opened and 60 closed periods.

By measuring the time from the first packet arrives in an open period to the last packet arrives in the same open period, the length of open periods can be determined. Similarly, the lengths of closed periods are determined by measuring the time from the last packet in an open period to the first packet of the next open period. By combining the two measured period types, it can be determined if the service time of the *ebtables_module* is lower than required by the property update rate.

B.2.2 Test results

From figures B.5 and B.6 the results of the test are seen. From Figure B.5 the ability of dynamically applying link properties of the *ebtables_emulation* modules is shown

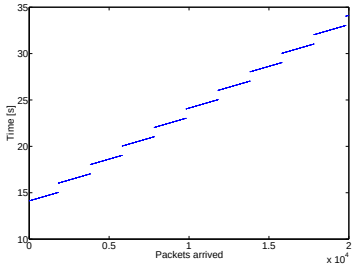

Figure B.5: Amount of packets arrived over time, exemplifying the open and closed periods of the *ebtables_emulation* module.

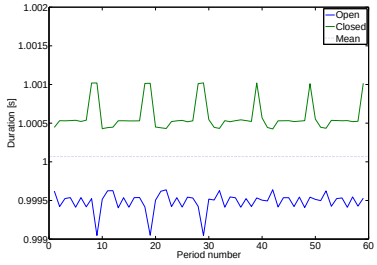

Figure B.6: Durations of open and closed periods from the *ebtables_emulation* module.

as periods of open link, allowing packets through, and closed link, preventing throughput of packets. Calculations of the duration of open and closed periods are shown in Figure B.6, showing that the closed periods are slightly longer than the open periods. Calculating the average time of all period durations gives the combined mean time for 1) applying a link property and 2) waiting one second. This mean time is 1 second and 68μs.

As the mean is measured by use of packets with a rate of 2000pps, yielding an average inter-arrival time of 0.5ms, the precision of the mean is $\pm$0.5ms. As the excess period duration of 68μs is within the precision of the mean, it is considered to be noise from the packet generator or from the packet processing in the emulator. Moreover, as the 68μs actually is with the precision, it can be concluded, that the *ebtables_emulation* module is capable of applying properties within a range of $\pm$0.5ms of the given duration value of 1 second. This means that the service time of the module is below 1ms, and thus an update can be performed with resolutions of up to 1ms. As this is the minimum used resolution allowable in the simulator, it is concluded that the link emulation capabilities of the *ebtables_emulation* are sufficient.

B.3 Link property emulation verification

While being able to dynamically change the existence of the emulated link, it must also be verified that link properties such as packet loss and packet delay of existing links can be emulated. This has been tested and is described in the following.

B.3.1 Test setup

Packet loss emulation UDP traffic is generated from *node2* to *node3* at a packet rate of 1000pps over a period of 30 seconds. Different deterministic settings of the packet error rate is set on the emulator node.

Delay emulation 1000 *ping*s are sent from *node2* to *node3* at different deterministic emulated link delays. The average round-trip time from each measurement is used to calculate the one-way link delay by using $t_{\text{delay}} = \frac{RTT}{2}$, assuming that the link delay is symmetric on the link.

Preset packet loss	Measured packet loss
5%	5.08%
15%	14.62%
30%	30.24%
50%	49.67%
75%	75.04%

Table B.2: Preset and measured values of packet losses on the emulator, measured at the receiving node.

Preset emulated link delay [ms]	Measured one-way link delay [ms]
0	0.239
1	1.211
2	2.187
3	3.236
4	4.263
5	5.242
6	6.238
7	7.214
8	8.314
9	9.289
10	10.266
100	100.235

Table B.3: Preset and measured values of one-way link delay on the emulator, measured using round-trip times from *ping*.

B.3.2 Test results

Packet loss emulation From Table B.2 it is seen, that a set value for the emulated packet loss on the emulator is reflected in the actual amount of lost packets on the network. Therefore it is concluded that the emulator is capable of emulating a given amount of packet loss on an existing link between two end-nodes.

Delay emulation From Table B.3 it is seen, that setting a deterministic delay to be emulated on a link by the emulator results in packets utilizing the link are delayed by the same amount. The measured one-way link delays are biased from a constant delay in the emulator, which is further investigated and discussed in section 7.1. The variation in the excess delay means is a consequence of using a scheduler in the emulator with a granularity of $\sim 122 \mu s$. Moreover, from the results described here, it is concluded that the emulator is capable of emulating a given duration of packet delay on an existing link between two end-nodes.

Appendix C

Routing overhead evaluation

This appendix describes the scenario setup, the parameters and the test cases performed in order to evaluate control overhead of the DSR and the OLSR algorithms.

C.1 Scenario and test case specification

Figure C.1 presents the scenario used in this experiment. The setup contains the topology emulation testbed, consisting of the switch and the emulation node. Furthermore, 3 laptops are used, running Ubuntu Linux 6.10, kernel version 2.6.17 and each implementing and running the routing algorithms listed in Table C.1.
End-node 2 generates UDP traffic received by *end-node 4*, rerouted by *end-node 3*.
For generating UDP traffic, a traffic generator called Distributed Internet Traffic Generator (D-ITG) is used. D-ITG implements a sender (located on *end-node 2*) and a receiver (located on *end-node 4*). The parameters used for generating UDP traffic are listed in Table C.2. The test cases executed in the experiment are constituted by the different settings of the *packet rate*.

C.2 Data processing

Each packet forwarded through the testbed is recorded in the database using the logging facility. In the database relevant header information is gathered. The relevant information for evaluation of routing overhead case are:

- time stamp
- packet length
- sender address
- destination address
- ingoing VLAN
- outgoing VLAN

The sender and destination addresses recorded from the packets are the end-node addresses of the IP-header and not intermediate relay-node addresses. Thus, the actual

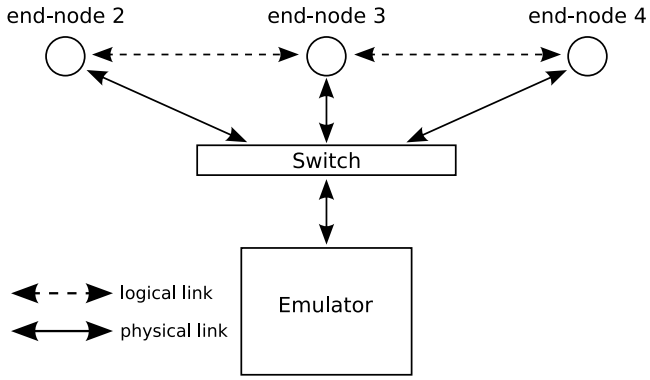

Figure C.1: Routing overhead evaluation scenario featuring 3 end-nodes and multi-hop routing by way of the topology emulation testbed.

Protocol	Implementation	Available from
DSR	DSR-UU v0.2	http://core.it.uu.se/core/index.php/DSR-UU
OLSR	OOLSR v0.99.15	http://hipercom.inria.fr/OOLSR/

Table C.1: Routing protocol implementations used for routing overhead evaluation experiment.

VLANs used is important to distinguish the links used for forwarding a packet. For instance, a packet from *end-node 2* (ip x.x.x.2) to *end-node 4* (ip x.x.x.4) that is relayed by *end-node 3* is tagged with the same sender (2) and destination (4) independent of being sent from *end-node 2* to *end-node 3* or from *end-node 3* to *end-node 4*. By considering the VLANs used (of both sender and receiver) a unique identification of each packet is possible.

SQL-queries are used for processing the gathered data. The first query identifies the routing packets of interest and a second summarizes the packet sizes based on the previously identified time interval.

```
SELECT min(timestamp),max(timestamp) FROM table
    WHERE packet_size>=1000

SELECT sum(packet_length) FROM table
    WHERE timestamp>='mininum timestamp'
    AND timestamp<='maximum timestamp'
```

Parameter	Value
Payload size (UDP)	972 bytes
Duration	30 seconds
Packet rate	0,10,100,300,500,750,1000 pkts/s

Table C.2: Traffic generator parameters used for generating traffic for routing overhead evaluation.

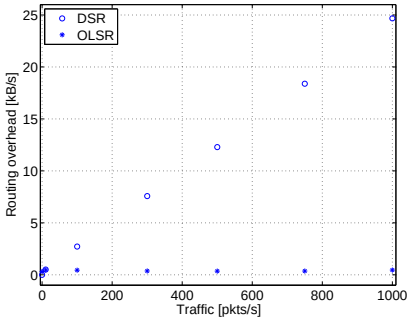

Figure C.2: Evaluation results of routing overhead of DSR and OLSR for different packet rates.

The payload UDP packets used in the experiment are of size 972 bytes. And since the packet size logged in the database is of IP-packets, UDP-header (8 bytes) and IP-header (20 bytes) must be added. Since DSR extends the header itself of the IP-packet, all lengths above 1000 bytes are accepted. To calculate the actual overhead rate, r_{overhead} in kB/s, the size of the generated payload throughput is subtracted from the total and normalized as

$$r_{\text{overhead}} = \frac{\left[\sum_{i=0}^{N} n_{\text{packet}}(i)\right]}{30 \cdot 1000} - 2\lambda$$

where $n_{\text{packet}}(i)$ is the individual packet size, N is the total amount of packets (including payload packets) and λ is the packet rate of the test case. The factor 2 must be included, as the 1000-bytes packets are sent over two links each, doubling the amount of traffic forwarded and recorded by the emulator node. The duration of the transmission is 30 seconds.

The results of the experiments are depicted in Figure C.2.

Appendix D

Routing evaluation configurations

This appendix lists the configurations of the hardware and the model parameters used for performing both the performance and the reliability evaluation in the project.

Laptops
The laptops used are three Fujitsu Siemens Lifebook E8110 and one Fujitsu Siemens S7020. The E8110 are running Ubuntu 6.10 and the S7020 is running Ubuntu 5.10. The routing algorithm implementations used are listed in C.1 on page 134 from the overhead evaluation.

Wireless adapters
For the experimental setup, the wireless adapters of the laptops were used. The E8110 laptops all employ the *ipw3945* wireless NIC driver, as they support both 802.11a/b/g, whereas the S7020 only supports 802.11b/g and thus employs the *ipw2200* driver. However, the *ipw3945* and the *ipw2200* drivers are not compatible for communication, so only the E8110 laptops were used in the experimental setup. The incompatibility issue was discovered during testing and Intel, who produces both adapters, is aware of the driver problem. All applied network adapters were set to

- 802.11a
- retry limit 15
- bandwidth 12Mbit/s
- transmission level 16 dBm

An average noise level of $-95dBm$ was observed for the different network adapters during the experiments.
For performing the experimental setup, the laptops where located as depicted in Figure D.1 to obtain separation of the end-node, while maintaining a multi-hop scenario. Also, as very little user-level control is supported of the parameters listed above, the nodes had to be placed so they themselves adjusted to the desired settings. These settings are very dependent on the environment, and as they were not monitored throughout the tests, they might have deviated, and may in that sense have influenced the results.

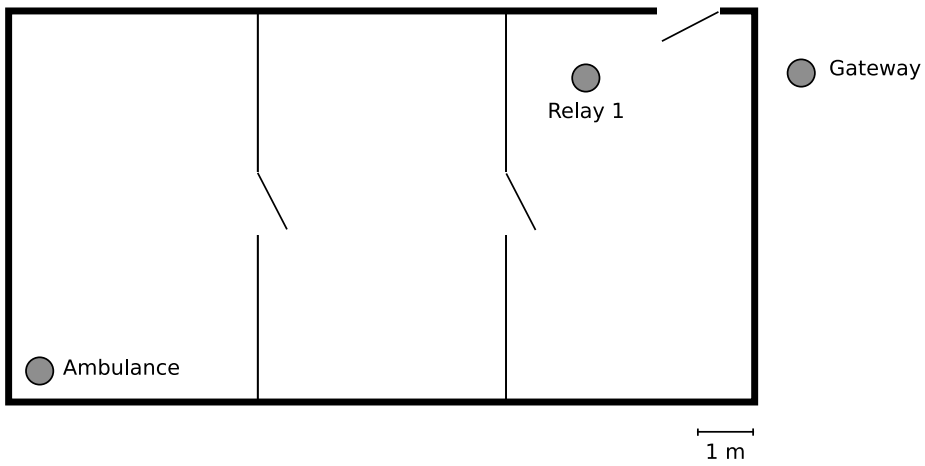

Figure D.1: Illustration of the physical locations of the laptops used in the experimental setup. All doors were closed during the experiments.

Parameter	Value
Acceleration	2m/s^2
Speed	14m/s
Node routes	Pre-scripted

Table D.1: Mobility model parameter settings for the evaluation tests.

However, deviations from the desired settings were not observed at any point during testing.

Emulator configuration
The different models of the simulator in the emulator node were configured to match the properties of the experimental setup for the comparisons. Tables D.1 to D.4 list the different parameter values.

Parameter	Value
β	2.3
σ	4
P_{tx}	16 dBm
Background noise	-100 dBm

Table D.2: Channel model parameter settings for the evaluation tests.

Parameter	Value
M_{QAM}	4
Frame size	1500 bytes
No. of retransmissions	15

Table D.3: PHY layer model parameter settings for the evaluation tests.

Parameter	Value
Bandwidth	12Mbit/s
Network load	1Mbit/s
Number of nodes	According to scenario

Table D.4: Delay model parameter settings for the evaluation tests.

Appendix E

Physical layer model

This appendix describes details of the physical layer model used in the simulator. The purpose is to provide the formulas used in the implementation.

The OFDM model of [Mangold et al., 2004] is based on the ratio of energy per symbol E_{av} to the energy of the noise N_0 (see equation (E.1)), i.e. the SNR for each OFDM symbol. This is expressed as

$$\frac{E_{av}}{N_0} = \frac{\alpha_g C}{N + \sum I}, \tag{E.1}$$

where C is the received signal strength, N is the background noise and $\sum I$ is the cumulated interference level e.g. interference from other stations transmitting, all the terms are in mW. α_g is a power loss caused by the guard interval for each OFDM symbol, and is defined as $\alpha_g = \frac{T_b - T_g}{T_b}$ where $T_b = 4\mu s$ is the duration of an OFDM symbol and $T_g = 0.8\mu s$ is the duration of the guard interval. This interval is introduced to avoid the signal degradation caused by ISI. ISI is caused by multiple receptions of the same signal with different delays.

To avoid ISI T_g should be larger than the delay spread of the channel, i.e. the maximum delay between two identical signals.
Assuming that:

- The delay spread of the channel is below T_g, eliminating ISI.
- That all bits of an OFDM symbol are subject to the same $\frac{E_{av}}{N_0}$.
- That the interferences N and $\sum I$ consist of additive white Gaussian noise, ignoring that there might be a correlation between interferences and the received original signal.

OFDM uses three different modulation schemes: Quadrature Amplitude Modulation (QAM), Quadrature Phase Shift Keying (QPSK) and Binary Phase Shift Keying (BPSK), each offering different data rates and levels of resilience towards interferences. Generally, the higher the data rate of the modulation the less resilient it is to noise. This is because the higher data rate is achieved by encoding more bits in each OFDM symbol increasing the chance of a bit error. For QAM different numbers of bit sequences may be encoded in each symbol, e.g. a M-QAM symbol may contain M different bit sequences. The BER for QAM may be expressed using the probability of selecting the

correct bit sequence

$$P_c = (1 - P_{\sqrt{M}})^2, \tag{E.2}$$

This formula is based on the relation of QAM to another modulation scheme called Pulse Amplitude Modulation (PAM), for details on this relation refer to [Proakis et al., 2001]. $P_{\sqrt{M}}$ is the symbol error probability for PAM

$$P_{\sqrt{M}} \quad = 2 \cdot (1 - \tfrac{1}{\sqrt{M}}) \cdot Q\left(\sqrt{(}\tfrac{3}{M-1} \cdot \tfrac{E_{av}}{N_0})\right), \tag{E.3}$$

here $Q(x)$ is the Gaussian error integral, which provides the probability mass above x. The probability of a wrongfully detected M-QAM symbol can be expressed by $P_M = 1 - (1 - P_{\sqrt{M}})^2$. Assuming that a wrongfully detected symbol only leads to one bit error the BER for QAM can be expressed as

$$P_b^{M-QAM} = \frac{1}{log_2 M} \cdot P_M \tag{E.4}$$

This assumption is somewhat fair, since the bit sequences in the different modulation schemes are arranged so that adjacent bit sequences only change by one bit, entailing that this is the most likely error. The formula also applies to QPSK, since it may be viewed as a 4-QAM.
For BPSK the BER can be expressed as

$$P_b^{BPSK} = Q\left(2 \cdot \frac{E_{av}}{N_0})\right) \tag{E.5}$$

Appendix F

Delay model

This appendix describes details of the delay model used in the simulator. The purpose is to provide the formulas used in the implementation.

In the following the terms: time, timeslot and backoff slot, are all of the same magnitude and constitute the time discretization of the model described.

According to [Tickoo and Sikdar, 2004] the delay experienced by a node attempting to transmit a frame, which may be called the service time of the frame, is comprised of the time it takes to send the frame, T_S, the time spent in backoff, and the delay due to other nodes transmitting. [Tickoo and Sikdar, 2004] establishes the PGF for the distribution of the service time, $B(z)$, as:

$$B(z) \quad = \quad BO(z)X(z)L(z) \tag{F.1}$$

Where $BO(z)$ is the PGF for the time spent in backoff, $X(z)$ the PGF for the delay caused by other nodes transmitting and $L(z)$ the PGF for the length of the packet to be served. Each with their respective Probability Mass Function (PMF): $bo(i)$, $x(i)$ and $l(i)$, used to establish the PGFs. In order to establish $bo(i)$ it is necessary to characterize the number of backoff slots the node has to wait before transmitting a frame. If a node successfully transmits a frame in its' first attempt with probability $(1-p)$, p is the probability of a collision and the PMF for the backoff slots is $U_{1,CW_{min}}$, where $U_{a,b}$ denotes the uniform distribution of a random variable between a and b. CW_{min} is the minimum contention window size. If the frame is successfully transmitted after a single collision, with probability $p(1-p)$, the backoff slot PMF is; $U_{1,CW_{min}} * U_{1,2CW_{min}}$, $*$ denotes the convolution operation. Extending to a series of collisions, the probability that the node experiences i backoff slots, before successfully transmitting a frame is:

$$\begin{aligned} P[BO=i] = \rho\,[&(1-p)U_{1,CW_{min}}(i) + p(1-p)(U_{1,CW_{min}} * U_{1,2CW_{min}}(i)) + \cdots \\ &+p^m(1-p)(U_{1,CW_{min}} * U_{1,2CW_{min}} * \cdots * U_{1,2^mCW_{min}}(i)) + \\ &p^{m+1}(1-p)(U_{1,CW_{min}} * \cdots * U_{1,2^mCW_{min}} * U_{1,2^mCW_{min}}) + \cdots\big] \end{aligned} \tag{F.2}$$

Where m is the upper bound on the times the contention window may be doubled and ρ is the queue utilization at the node. The number of elements in the summation is bounded by the number of allowed retransmissions. From (F.2) $bo(i)$ can be obtained.

$x(i)$, may be obtained by determining all possible contributions to the service time from other nodes transmitting, while the target node is in backoff.

$$P(X=n) = \begin{cases} P[k\ collisions|j\ active\ slots] \cdot l^{j-k}(i) P[SA=j] & \text{for } n = kT_C + i, \\ 0 & \text{otherwise.} \end{cases} \tag{F.3}$$

Equation (F.3) is the probability that the collisions and successful transmissions of other nodes contribute n time slots to the service time.

In order to determine the variables for the equation consider a network, with channel service rate μ, consisting of N nodes, each with frame arrival rate λ and queue utilization ρ. All specified in frames per timeslot. The foundation of the model is the probability that a frame transmission causes a collision. If a node transmits a frame in a given timeslot, the probability of a collision caused by a transmission from one or more of the remaining nodes may be denoted as:

$$p = 1 - P[NT]^{N-1} \tag{F.4}$$

Where $P[NT]$ represents the probability that a node does not transmit in a slot. $P[NT]$ may be written as, the probability that the frame queue at the node is empty (QE) combined with the probability that the queue at the node is not empty (QNE) and the probability that the node is not transmitting (NT):

$$P[NT] = P[NT|QE]P[QE] + P[NT|QNE]P[QNE] \tag{F.5}$$
$$= 1 - \rho + \rho P[NT|QNE] \tag{F.6}$$

Considering that the average number of backoff slots a node experiences in the saturated case, i.e. QNE, is (for details refer to [Tay and Chua, 2001])

$$\overline{W} = \frac{1-p-p(2p)^m}{1-2p}\frac{CW_{min}}{2} \tag{F.7}$$

$P[NT]$ may be written as

$$P[NT] = (1-\rho) + \rho\frac{\overline{W}-1}{\overline{W}} = 1 - \frac{\rho}{\overline{W}} \tag{F.8}$$

In order to establish ρ and keeping in mind $\rho = \frac{\lambda}{\mu}$, the average time it takes to serve a packet, μ^{-1}, is established. Assuming that all nodes have the same traffic arrival rate, $\rho(N-1)$ transmissions from other nodes occur between two transmissions from the target node, on average. Entailing that the contribution of the transmissions from the other nodes in terms of time slots is $\rho(N-1)T_S$. Here T_S is the time it takes to send a packet in units of timeslots. Adding the contribution from the collisions of transmissions of other nodes, $\frac{\rho(N-1)pT_C}{1-p}$ (T_C is the time a collision occupies the channel), and the time to transmit the frame of the target node, along with any collisions that it might encounter, μ^{-1} may be defined as:

$$\mu^{-1} = \rho(N-1)\left[T_S + T_C\frac{p}{1-p}\right] + \overline{W} + T_S + T_C\frac{p}{1-p} \tag{F.9}$$

Combining (F.7), (F.8) and (F.4), and substituting ρ by utilizing that $\rho = \frac{\lambda}{\mu}$, p may be obtained by solving:

$$p = \frac{\lambda\left[T_S + T_C\frac{p}{1-p}\right]}{1-\lambda(N-1)\left[T_S + T_C\frac{p}{1-p}\right] - \frac{\lambda(1-p-p(2p)^m)}{1-2p}\frac{CW_{min}}{2}} \quad \text{(F.10)}$$

Having determined p we return to Equation (F.3) and establish the term $P[SA = j]$. $P[SA = j]$ represents the probability that there are j active slots before the target node's backoff counter reaches zero, and is given by Equation (F.11). An active slot is a slot where a given node, besides the target node, initiates a transmission.

$$P[SA = j] = \sum_{i=j}^{\infty}\binom{i}{j}q^j(1-q)^{i-j}P[BO = i] \quad \text{(F.11)}$$

Here q is the probability that a given slot is active:

$$q = 1 - \left(1 - \frac{\rho}{\overline{W}}\right)^N \quad \text{(F.12)}$$

$\frac{\rho}{\overline{W}}$ is the probability that a node tries to initiate a transmission in a given timeslot.

In Equation (F.3), k represents the number of active slots where a collision occurs. Given that a slot is active, the probability that a collision occurs in the slot is given by:

$$q_c = \frac{q - \frac{\rho N}{\overline{W}}\left(1 - \frac{\rho}{\overline{W}}\right)^{N-1}}{q} \quad \text{(F.13)}$$

The probability that k collisions occur given j active slots, may then be expressed as:

$$P[k\ collisions|j\ active\ slots] = \binom{j}{k}q_c^k(1-q_c)^{j-k} \quad \text{(F.14)}$$

The remaining $j-k$ active slots result in successful transmissions. The contribution to the service time from the successful transmission may be determined by $l^{j-k}(i)$, defined in Equation (F.15)

$$P[\sum^{j} pkt\ time = i] = l * l * \cdots * l(i) = l^{(j)}(i), \quad \text{(F.15)}$$

which describes the j-fold convolution of the time it takes to send a frame, yielding the PMF for the time it takes to send j packets. Having established this distribution the equations need to establish $bo(i)$, $x(i)$ and $l(i)$, have all been defined.

Acronyms

AODV Ad hoc On-demand Distance Vector

API Application Program Interface

BER Bit Error Rate

BPSK Binary Phase Shift Keying

CDF Cumulative Distribution Function

CNM Control Network Manager

COTS Commercial Off-The-Shelf

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CTS Clear To Send

DCF Distributed Coordination Function

DIFS DCF Inter-Frame Space

D-ITG Distributed Internet Traffic Generator

DNC Deterministic Network Calculus

DSR Dynamic Source Routing

DSSS Direct Sequence Spread Spectrum

DYMO Dynamic Manet On-demand

FER Frame Error Rate

FPGA Field Programmable Gate Array

FPGAs Field Programmable Gate Arrays

GUI Graphical User Interface

ICI Inter-Carrier Interference

ICMP Internet Control Message Protocol

IFS Inter-Frame Space

ISI Inter-Symbol Interference

LoS Line of Sight

MAC Medium Access Control

MANET Mobile Ad hoc NETwork

MANETs Mobile Ad hoc NETworks

MB MegaByte

MPR MultiPoint Relay

MPRs MultiPoint Relays

NAV Network Allocation Vector

NIC Network Interface Card

OFDM Orthogonal Frequency Division Multiplexing

OLSR Optimized Link State Routing

OSPF Open Shortest Path First

PAM Pulse Amplitude Modulation

PER Packet Error Rate

PIFS PCF Inter Frame Space

PGF Probability Generating Function

PMF Probability Mass Function

PMFs Probability Mass Function

QAM Quadrature Amplitude Modulation

QoS Quality of Service

QPSK Quadrature Phase Shift Keying

RIP Routing Information Protocol

RPGM Reference Point Group Mobility

RSS Receive Signal Strength

RSSI Received Signal Strength Indication

RTC Real Time Clock

RTS Request To Send

SIFS Short Inter-Frame Space

SIP Session Initiation Protocol

SNR Signal to Noise Ratio

STRAW STreet RAndom Waypoint

VANET Vehicular Ad hoc NETwork

VANETs Vehicular Ad hoc NETworks

Bibliography

[Alex Varshavsky, 2004] Alex Varshavsky, E. d. L. (2004). Alleviating self-interference in manets. Technical report, University of Toronto. adhoc-environment/varshavsky-2004-self-interference-in-MANETS.pdf, downloaded 2006-10-24.

[Avizienis et al., 2004] Avizienis, A., Laprie, J., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33.

[Bai et al., 2003] Bai, F., Sadagopan, N., and Helmy, A. (2003). IMPORTANT: a framework to systematically analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc Networks. *INFOCOM 22nd Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2.

[Bianchi, 2000] Bianchi, G. (2000). Performance analysis of the IEEE 802.11 distributed coordinationfunction. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547.

[Bohacek, 2004] Bohacek, S. (2004). UDel Models: For Simulation of Urban Mobile Wireless Networks. *Obtain via: http://udelmodels. eecis. udel. edu/*.

[Broch et al., 1998] Broch, J., Maltz, D., Johnson, D., Hu, Y., and Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97.

[Brownfield and Davis, 2005] Brownfield, M. and Davis, N. (2005). Symbiotic Highway Sensor Network. *IEEE VEHICULAR TECHNOLOGY CONFERENCE*, 62(4):2701.

[Cerpa et al., 2005] Cerpa, A., Wong, J., Kuang, L., Potkonjak, M., and Estrin, D. (2005). Statistical model of lossy links in wireless sensor networks. *Proceedings of the 4th international symposium on Information processing in sensor networks*.

[Choffnes and Bustamante, 2005] Choffnes, D. and Bustamante, F. (2005). An integrated mobility and traffic model for vehicular wireless networks. *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, pages 69–78.

[Chong et al., 2003] Chong, C., Tan, C., Laurenson, D., McLaughlin, S., Beach, M., and Nix, A. (2003). A new statistical wideband spatio-temporal channel model for 5-GHz band WLAN systems. *Selected Areas in Communications, IEEE Journal on*, 21(2):139–150.

[Chung and Goldsmith, 2001] Chung, S. and Goldsmith, A. (2001). Degrees of freedom in adaptive modulation: a unified view. *Communications, IEEE Transactions on*, 49(9):1561–1571.

[D-ITG, 2007] D-ITG (2007). http://www.grid.unina.it/software/ITG.

[Davies, 2000] Davies, V. (2000). Evaluating mobility models within an ad hoc network. Master's thesis, advisor: Tracy Camp, Dept. of Mathematical and Computer Sciences.Colorado School of Mines.

[Deri et al., 2004] Deri, L. et al. (2004). Improving passive packet capture: Beyond device polling. *Proceedings of the Fourth International System Administration and Network Engineering Conference (SANE 2004), Amsterdam, The Netherlands, September*.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

[DLR, 2006] DLR, Z. (2006). Simulation of Urban MObility. *Obtain via: http://sumo. sourceforge. net/*.

[Draves et al., 2004] Draves, R., Padhye, J., and Zill, B. (2004). Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–144, New York, NY, USA. ACM Press.

[eCos, 2007] eCos (2007). http://ecos.sourceware.org.

[Engelstad and Osterbo, 2006] Engelstad, P. and Osterbo, O. (2006). The delay distribution of IEEE 802.11 e EDCA and 802.11 DCF. *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*, page 10.

[ettcp, 2002] ettcp (2002). http://sourceforge.net/projects/ettcp.

[Flynn et al., 2002] Flynn, J., Tewari, H., and O?Mahony, D. (2002). A Real-Time Emulation System for Ad Hoc Networks. *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*.

[Friis, 1946] Friis, H. (1946). A note on a simple transmission formula. *Proc. IRE*, 34(5):254–256.

[Gaertner and Cahill, 2004] Gaertner, G. and Cahill, V. (2004). Understanding link quality in 802.11 mobile ad hoc networks. *Internet Computing, IEEE*, 8(1):55–60.

[Ghassemzadeh et al., 2002] Ghassemzadeh, S., Jana, R., Rice, C., Turin, W., and Tarokh, V. (2002). A statistical path loss model for in-home UWB channels. *Ultra Wideband Systems and Technologies, 2002. Digest of Papers. 2002 IEEE Conference on*, pages 59–64.

[Gorgorin, 2006] Gorgorin, C. e. (2006). Evaluating mobility models within an ad hoc network. Master's thesis, advisor: Liviu Iftode, Rutgers University Computer Science Department.

[Hadzi-Velkov and Spasenovski, 2003] Hadzi-Velkov, Z. and Spasenovski, B. (2003). Saturation throughput-delay analysis of IEEE 802.11 DCF in fading channel. *Communications, 2003. ICC'03. IEEE International Conference on*, 1.

[Heidemann et al., 2001] Heidemann, J., Bulusu, N., Elson, J., Intanagonwiwat, C., chan Lan, K., Ya Xu, W. Y., Estrin, D., and Govindan, R. (2001). Effects of detail inwireless network simulation. *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*.

[HIDENETS, 2006a] HIDENETS (2006a). Specification hidenets laboratory set-up scenario and components. http://hidenets.aau.odeum.com/. /D6.1_v4.0.pdf, downloaded 20070610.

[HIDENETS, 2006b] HIDENETS (2006b). Use case scenarios and preliminary reference model. http://hidenets.aau.odeum.com/. /WP1_D1_draft_v0.90.pdf, downloaded 20060929.

[Holland and Vaidya, 2002] Holland, G. and Vaidya, N. (2002). Analysis of TCP Performance over Mobile Ad Hoc Networks. *Wireless Networks*, 8(2):275–288.

[Hollick, 2004] Hollick, M. (2004). *Dependable routing for cellular and ad hoc networks*. PhD thesis, Technische Universität Darmstadt, Department of Electrical Engineering and Information Technology.

[IEEE, 2003] IEEE (2003). Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, IEEE. adhoc-environment/IEEE-std-802.11-1999.pdf, downloaded 2006-10-24.

[IETF, 2007] IETF (2007). http://www.ietf.org/html.charters/manet-charter.html.

[IETF, 2000] IETF, R. (2000). 3261. *SIP: Session Initiation Protocol*.

[Jacquet and Laouiti, 1999] Jacquet, P. and Laouiti, A. (1999). Analysis of mobile ad-hoc networking routing protocols in random graph models. Technical report, Inria. modelling/jacquet-manet-graph-analysis.pdf, downloaded 2006-10-16.

[Jacquet et al., 2001] Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68.

[Jelitto and Truong, 2003] Jelitto, J. and Truong, A. (2003). Power and rate adaptation in IEEE 802.11 a wireless LANs. *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, 1.

[Johnson and Maltz, 1996] Johnson, D. and Maltz, D. (1996). Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353:153–181.

[Johnson et al., 2001] Johnson, D., Maltz, D., Broch, J., et al. (2001). DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. *Ad Hoc Networking*, 1:139–172.

[Judd and Steenkiste, 2004] Judd, G. and Steenkiste, P. (2004). Repeatable and realistic wireless experimentation through physical emulation. *SIGCOMM Comput. Commun. Rev.*, 34(1):63–68.

[JUNG, 2007] JUNG (2007). http://jung.sourceforge.net.

[Ke et al., 2000] Ke, Q., Maltz, D., and Johnson, D. (2000). Emulation of Multi-Hop Wireless Ad Hoc Networks. *Proceedings of the 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000)*.

[Kernel, 2007] Kernel, L. (2007). http://www.kernel.org.

[Krishnamachari et al., 2000] Krishnamachari, B., Estrin, D., and Wicker, S. (2000). Modelling data-centric routing in wireless sensor networks. Technical report, Inria. modelling/krischnamachari-data-centric-routing-model.pdf, downloaded 2006-10-16.

[Malkin, 1994] Malkin, G. (1994). RIP Version 2 Protocol Analysis. Technical report, RFC 1721, Xylogics, Inc.

[Maltz et al., 1999] Maltz, D., Broch, J., and Johnson, D. (1999). *Experiences Designing and Building a Multi-hop Wireless Ad Hoc Network Testbed*. School of Computer Science, Carnegie Mellon University.

[Mangold et al., 2004] Mangold, S., Choi, S., and Esseling, N. (2004). An Error Model for Radio Transmissions of Wireless LANs at 5GHz. *Proc. Aachen SymposiumŠ2001*, pages 209–214.

[Matthiesen et al.,] Matthiesen, E. V., Renier, T., and Schwefel, H.-P. A new selection metric for backup group creation in inter-vehicular networks.

[McCanne and Floyd, 1995] McCanne, S. and Floyd, S. (1995). ns-2 Network Simulator. *Obtain via: http://www. isi. edu/nsnam/ns*.

[Moy, 1994] Moy, J. (1994). RFC 1583. *OSPF version*, 2.

[Mueller et al., 2004] Mueller, S., Tsang, R., and Ghosal, D. (2004). Multipath routing in mobile ad hoc networks: Issues and challenges. *Performance Tools and Applications to Networked Systems*, 2965.

[MySQL, 2007] MySQL (2007). http://mysql.org.

[Netfilter, 2007] Netfilter (2007). http://www.netfilter.org.

[Perkins, 2002] Perkins, C. (2002). IETF-RFC 3344. *IP Mobility Support for IPv4, August*.

[Perkins and Bhagwat, 1994] Perkins, C. and Bhagwat, P. (1994). Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244.

[Perkins and Royer, 1999] Perkins, C. and Royer, E. (1999). Ad-hoc on-demand distance vector routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 2.

[Pktgen, 2005] Pktgen (2005). http://linux-net.osdl.org/index.php/Pktgen.

[PostgreSQL, 2007] PostgreSQL (2007). http://www.postgresql.org.

[Prefuse, 2007] Prefuse (2007). http://prefuse.org.

[Proakis et al., 2001] Proakis, J. et al. (2001). *Digital Communication*. Osborne-McGraw-Hill.

[Pursley and Taipale, 1987] Pursley, M. and Taipale, D. (1987). Error Probabilities for Spread-Spectrum Packet Radio with Convolutional Codes and Viterbi Decoding. *Communications, IEEE Transactions on [legacy, pre-1988]*, 35(1):1–12.

[Qayyum et al., 2000] Qayyum, A., Viennot, L., and Laouiti, A. (2000). Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, Inria. modelling/qayyum-multipoint-relaying.pdf, downloaded 2006-10-16.

[QNX, 2007] QNX (2007). http://www.qnx.com.

[Rajarman, 2002] Rajarman, R. (2002). Topology control and rounting in ad hoc networks: a survey. *ACM SIGACT News*, 33(2):60–73.

[Rappaport, 2002] Rappaport, T. (2002). Wireless communications: principles and practice 2nd Edition [M].

[RTAI, 2007a] RTAI (2007a). http://www.rtai.org.

[RTAI, 2007b] RTAI (2007b). http://www.fsmlabs.com.

[Seeling et al., 2004] Seeling, P., Reisslein, M., and Kulapala, B. (2004). Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial. *IEEE Communications Surveys and Tutorials*, 6(3):58–78.

[Short et al., 1995] Short, J., Bagrodia, R., and Kleinrock, L. (1995). Mobile wireless network system simulation. *Wireless networks 1*, pages 451–467.

[Stewart and Metz, 2001] Stewart, R. and Metz, C. (2001). SCTP: new transport protocol for TCP/IP. *Internet Computing, IEEE*, 5(6):64–69.

[Stuedi et al., 2005] Stuedi, P., Chinellato, O., and Alonso, G. (2005). Connectivity in the presence of shadowing in 802.11 ad hoc networks. *Wireless Communications and Networking Conference, 2005 IEEE*, 4.

[Takai and Bajaj, 1999] Takai, M. and Bajaj, L. (1999). R, Ahuja, R. Bagrodia and M. Gerla,?GloMoSim: A Scalable Network Simulation Environment,? Technical report, Technical report 990027, UCLA, Computer Science Department.

[Tanenbaum, 2002] Tanenbaum, A. (2002). *Computer Networks*. Prentice Hall PTR.

[Tay and Chua, 2001] Tay, Y. and Chua, K. (2001). A Capacity Analysis for the IEEE 802.11 MAC Protocol. *Wireless Networks*, 7(2):159–171.

[tcpdump, 2007] tcpdump (2007). http://www.tcpdump.org.

[Tian et al., 2002a] Tian, J., Hähner, J., and Becker, C. (2002a). Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation. *Proceedings of the 35th Annual Simulation Symposium*.

[Tian et al., 2002b] Tian, J., Hahner, J., Becker, C., Stepanov, I., and Rothermel, K. (2002b). Graph-based mobility model for mobile ad hoc network simulation. *Proceedings of the 35th Annual Simulation Symposium (SS'02)*, pages 60–73.

[Tickoo and Sikdar, 2004] Tickoo, O. and Sikdar, B. (2004). A queueing model for finite load IEEE 802.11 random access MAC. *Communications, 2004 IEEE International Conference on*, 1.

[Vahdat et al., 2002] Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostić, D., Chase, J., and Becker, D. (2002). Scalability and accuracy in a large-scale network emulator. *ACM SIGOPS Operating Systems Review*, 36:271–284.

[Viennot et al., 2004] Viennot, L., Jacquet, P., and Clausen, T. (2004). Analyzing Control Traffic Overhead versus Mobility and Data Traffic Activity in Mobile Ad-Hoc Network Protocols. *Wireless Networks*, 10(4):447–455.

[Wang et al., 2004] Wang, Z., Tameh, E., and Nix, A. (2004). Statistical peer-to-peer channel models for outdoor urban environments at 2 GHz and 5 GHz. *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, 7.

[Wijesinha et al., 2005] Wijesinha, A., Song, Y., Krishnan, M., Mathur, V., Ahn, J., and Shyamasundar, V. (2005). Throughput measurement for UDP traffic in an IEEE 802.11 g WLAN. *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*, pages 220–225.

[Wireshark, 2006] Wireshark (2006). http://www.wireshark.org.

[Zang et al., 2005] Zang, Y., Stibor, L., Orfanos, G., Guo, S., and Reumerman, H. (2005). An error model for inter-vehicle communications in highway scenarios at 5.9 GHz. *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 49–56.

[Zheng and Ni, 2002] Zheng, P. and Ni, L. (2002). EMPOWER: A Network Emulator for Wireless and Wireline Networks. *Procreedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies.*