Thomas Finke

SEREMA - Self-Organized Routing in Heterogeneous Mobile Ad Hoc Networks

SEREMA

Self-Organized Routing in
Heterogeneous Mobile Ad Hoc Networks

Thomas Finke

Universitätsverlag Ilmenau
2016

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	7. Oktober 2014
1. Gutachter:	Univ.-Prof. Dr. rer. nat. Jochen Seitz (Technische Universität Ilmenau)
2. Gutachter:	Univ.-Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel (Technische Universität Ilmenau)
3. Gutachter:	Prof. Dr.-Ing. Jürgen Schröder (Hochschule Heilbronn)
Tag der Verteidigung:	5. Juli 2016

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-86360-144-7 (Druckausgabe)

URN urn:nbn:de:gbv:ilm1-2016000383

Titelfoto: photocase.com | AlexFlint

The things that come to those who wait
may be the things left by those that got there first.

Steven Tyler

Danksagung

Diese Arbeit entstand während meiner Zeit als wissenschaftlicher Mitarbeiter an der Hochschule Heilbronn in Kooperation mit der Technischen Universität Ilmenau. Dieses Vorhaben wäre allerdings nicht möglich gewesen ohne die Unterstützung durch meinen Doktorvater Herrn Prof. Dr. rer. nat. Jochen Seitz, welcher stets ein offenes Ohr für mich hatte und mit seinen Anregungen sehr zum Erfolg der Arbeit beigetragen hat. Vielen Dank an dieser Stelle auch für den stets sehr angenehmen Kontakt.

Weiterhin danke ich Herrn Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel für die Übernahme des Zweitgutachtens und seine stets kritischen Fragen zu meiner Arbeit. Auch danke ich Herrn Prof. Dr.-Ing. Jürgen Schröder für seinen Einsatz, die anregenden Diskussionen, die stets konstruktive Kritik sowie die Übernahme des Drittgutachtens.

Meinen ehemaligen Kollegen der TU Ilmenau Sebastian Schellenberg, Silvia Krug, Peggy Begerow und Markus Hager danke ich für die stets hitzigen Diskussionen.

Besonderer Dank gilt auch dem Studierenden Herrn Wei Li für seine Hauptseminararbeit auf dem Gebiet adaptiver Routingansätze für den Einsatz in mobilen Ad-hoc-Netzwerken. Erst diese Arbeit zeigte mir einen weiteren Routingansatz auf.

Vielen Dank auch an meinen Kollegen Joachim Kircher für die vielen Diskussionen, bei denen wir meist mit komplett gegensätzlichen Ansichten starteten.

Meiner Freundin Bettina Göller danke ich für die Unterstützung, das stetige Antreiben und ihre Korrekturvorschläge für diese Arbeit. Ich hoffe, dass wir nach dieser Arbeit wieder mehr Zeit füreinander haben.

Ohne die jahrelange Unterstützung meiner Eltern und dem Ermöglichen meines Studiums wäre diese Arbeit sicher nicht entstanden. Vielen Dank!

Kurzfassung

Nach Naturkatastrophen, wie beispielsweise Erdbeben, Wirbelstürmen, Flutwellen oder auch nach durch den Menschen verursachten Katastrophen wie Terroranschlägen oder Unfällen, ist es sehr wichtig die Ersthelfer zu organisieren. Hierfür ist eine Kommunikationsinfrastruktur, welche zum Beispiel Basisstationen für Mobilfunknetze enthält, von entscheidender Bedeutung. Diese Infrastruktur kann durch die Katastrophe jedoch schwer beeinträchtigt oder vollkommen zerstört sein. Heutzutage sind sowohl Ersthelfer als auch Opfer üblicherweise mit leistungsfähigen mobilen Endgeräten, wie Smartphones oder Notebooks, ausgerüstet. Diese mobilen Endgeräte, welche über eine Vielzahl von Netzzugangstechnologien verfügen, können zu einem sogenannten Ad-hoc-Netzwerk zusammengeschlossen werden und bilden anschließend eine infrastrukturlose Kommunikationsbasis.

Die Leistungsfähigkeit von kabellosen Ad-hoc-Netzwerken ist dabei stark von der Anzahl bekannter Verbindungen im Netz abhängig. Diese Verbindungen, auch Routen genannt, werden durch das verwendete Routingprotokoll gesucht und ständig aktualisiert. Hierzu stehen verschiedenartige Routingprotokolle zur Verfügung, welche Topologieinformationen zwischen den einzelnen Knoten eines Netzwerks austauschen. Für kabellose Ad-hoc-Netzwerke sind hierfür zahlreiche Routingprotokolle verfügbar, jedoch sind diese bereits existierenden Protokolle nur eingeschränkt für hochdynamische mobile Ad-hoc-Netzwerke geeignet. Dies liegt darin begründet, dass sie nicht in der Lage sind, sich an große Änderungen im Netzwerk anzupassen.

In Katastrophenszenarien können allerdings hochdynamische Netzwerke vorkommen, in welchen beispielsweise die Größe des Netzes zwischen einigen wenigen und einigen hundert Knoten schwankt oder sich die Knotengeschwindigkeit von statischen bis hin zu hochmobilen Knoten verändert. Die vorliegende Arbeit präsentiert einen adaptiven Ansatz, welcher in der Lage ist, die gegebenen Parameter des Netzwerks in einer dezentralen Weise zu ermitteln und anschließend das verwendete Routingprotokoll während der Laufzeit zu wechseln, um somit das Routing sehr flexibel an die Gegebenheiten des Netzwerks anzupassen.

Abstract

After natural disasters like earthquakes, hurricanes, tsunamis or after human-made disasters like terrorist attacks or accidents, it is very important to organize the disaster response teams. Therefore, communication infrastructure like base stations for cellular networks is very important. However, this infrastructure could be heavily damaged during the disaster. Nevertheless, communication should be possible. Typically, all the disaster response teams as well as the victims carry along mobile devices such as smartphones or notebooks. These mobile network devices can be combined to an ad hoc network; this is an infrastructureless network that can be used for communication.

The performance of wireless ad hoc networks is mainly affected by the routing protocol. It is a very important task of the routing protocol to find suitable routes for data forwarding in a fast and efficient way. The routes are found using different routing protocols which exchange some type of topology information between the nodes, which are usually mobile devices. A lot of routing protocols are available for wireless ad hoc networks. However, these protocols are not suitable for highly dynamic mobile ad hoc networks because they are not able to adapt to major changes in the network topology.

In disaster scenarios, highly dynamic networks are considered where the size of these networks can vary from a few nodes to hundreds of nodes and the movement of the nodes can vary from static to highly mobile nodes, for example. This work presents an adaptive approach which gathers information about the network in a self-organized way and is able to switch between multiple routing protocols during the runtime of the network to adapt routing very flexibly to highly dynamic mobile ad hoc networks.

Contents

1 Introduction

This work deals with adaptive routing in heterogeneous mobile ad hoc networks for the use in disaster scenarios. Since single routing protocols were designed for special use cases they cannot perform well in all possible network constellations, respectively in changing network scenarios. Therefore, adaptive routing approaches were introduced which permit the combination of multiple routing protocols to allow the change of the protocol depending on the network context. However, the existing schemes dealing with adaptive routing have some disadvantages. They are not self-organized or are based on a centralized node which is responsible for decision-making. These circumstances will be depicted in the following sections and the objective of this work will be discussed. Afterwards, the structure of this document will be described to show the significance of the subsequent chapters and their relationship to each other.

1.1 Motivation

In the recent years, a lot of disasters occurred, ranging from natural disasters like hurricanes or floods over accidents like airplane or train crashes or nuclear power plant breakdowns to disasters triggered by terrorist attacks. After such incidents it is necessary for disaster response teams to quickly get an overview of the situation to make their work efficient. Therefore, the communication between different rescue teams and between rescue teams and victims is very important. Moreover, the communication with the headquarter or a connection to the Internet is significant for information exchange. In the typical case, the devices carried by the rescue teams and the victims of the disaster are network devices which are used to establish connections to previously installed communication infrastructure. However, centralized infrastructure such as a radio tower represents a *Single Point of Failure* and in disaster scenarios, it could be seriously affected and absolutely fail. The failure of centralized technologies results in rescue teams being isolated from other rescue teams, command centers and victims.

To ensure robust communication in these cases, a lot of work has recently been done. The participants – also called nodes – have been modified to support the creation of a network among them in an ad hoc manner without any previously existing infrastructure. The resulting ad hoc networks allow the creation of networks everywhere and every time without the need for any given communication infrastructure. In infrastructure networks, the nodes usually communicate via a base station. In ad hoc networks, the nodes either communicate directly with each other, if they have a direct connection, or the data packets are forwarded via other nodes into the direction of the destination. Therefore, each node acts as host and router simultaneously and is able to forward data packets for other nodes. In this way, it is possible that two nodes communicate with each other without having a direct connection. Instead, the data packets are forwarded by the nodes in between. In order to enable such packet forwarding, the nodes need to know the possible routes between each other. It is the task of routing protocols to find available routes between the nodes and to maintain these routes afterwards. Related work [RHS03, HMDD04, Jia05, Hoe07, RP10, HMD12] shows that no routing protocol performs well in all possible network environments. Therefore, a lot of protocols have recently been developed. However, this does not solve the problem that a single routing protocol cannot cope with highly dynamic network environments. If the number of nodes in the network increases heavily or the data traffic in the network explodes, for example, the existing routing protocols cannot adapt to this behavior. Alternatively, hybrid protocols can be used, but this type of routing protocol can only adapt to some small changes in the network.

In common disaster scenarios, the networks can be highly dynamic and the parameters can change widely during runtime. To cope with these requirements, this work deals with adaptive routing in *Mobile Ad Hoc Networks(MANETs)*, which allows switching between different routing protocols to be able to react very flexibly to changes in the network scenario. To achieve this objective a new adaptive routing framework named *Self-Organized Routing in Heterogeneous Mobile Ad Hoc Networks (SEREMA)* [Fin12a, Fin12b] is introduced.

1.2 Objective

The present work deals with the development of an adaptive routing framework for the use in highly dynamic disaster scenarios. The scheme has to be decentralized to overcome the problem of *Single Points of Failure*. Furthermore, each node should be able to make its own routing protocol decision and to select the best protocol for its requirements, more or less independently of the other nodes in the network. It should also be possible to use multiple active routing protocols at the same time. To implement such a feature, the framework needs some type of translation between different routing packets. When changing the protocol during the runtime of the network, existing data connections should be maintained without interruption and preferably without any influence on their data rate. It is desired that new routing protocols can be added to the framework without much effort.

Beside the theoretical development of the approach, there should also be simulation runs to proof the functionality of the new concept. Hence, a suitable simulation environment and the routing protocols which will be used in the new adaptive routing framework have to be chosen. Later, it should be possible to run the adaptive routing framework on real hardware. Therefore, a suitable implementation is required.

To achieve a good performance of the new routing scheme, it is necessary to know which routing protocol performs best in which network scenario. However, this work only deals with the development of the new routing framework and proofs its functionality. To achieve best performance in a lot of different scenarios, many simulation runs and analyses of the network behavior are required. This is an extensive task itself; therefore, it is recommended to be done separately as future work.

1.3 Outline

The following chapter 2 introduces the characteristics of ad hoc networks as the basis for chapter 3.

Chapter 3 shows the requirements for today's routing protocols and provides an overview of the different types of routing protocol families. Furthermore, exemplary protocols of each type are explained in detail and their advantages as well as their disadvantages are discussed.

Chapter 4 explains why one single routing protocol cannot cope with all possible network scenarios. Afterwards, the convenience of hybrid routing approaches is shown, followed by the benefits of adaptive routing solutions.

The next chapter 5 describes the motivation of and the demands on *SEREMA*. Existing adaptive routing mechanisms are discussed and it is exposed why they are inadequate for the requirements in this work.

Chapter 6 presents the functionality of *SEREMA* and its architecture, including the cooperation of the different routing protocols. Furthermore, the single elements of the framework are explained and the compatibility to nodes that only use a standard routing protocol is discussed.

Possible simulation environments are discussed in chapter 7 and the elected network simulator is introduced.

Chapter 8 shows the behavioral tests of the functionality of *SEREMA* based on simple network scenarios with static nodes. The performance of *SEREMA* is afterwards analyzed with simulation scenarios applying mobile nodes.

The subsequent chapter 9 presents a summary of this dissertation, while chapter 10 gives an outlook and states some ideas for future work.

2 Ad Hoc Networks

When rescue teams arrive at the location of the disaster, they usually cannot use previously installed communication infrastructure because it is broken. The only way to set up a network for communication in a fast and efficient way is to use an ad hoc network. This chapter explains the challenges with infrastructure networks and introduces the advantages of ad hoc networks, their characteristics, including their structure and how they operate, and last but not least their potential problems. The information in this chapter provides the basics for the following chapter 3 which deals with routing in ad hoc networks.

Before starting with the content of this work, some basic terms and definitions should be introduced using an example network scenario (cf. figure 2.1).

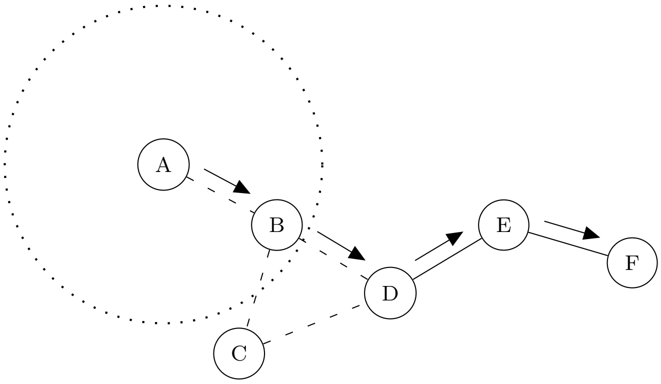


Figure 2.1: An Exemplary Network Scenario

The devices in the network, represented by circles containing a capital letter, can be servers, user devices or base stations, for example, and are simply denoted as nodes. These nodes are interconnected by links which are shown as dashed lines for wireless links or as continuous lines for wired links. The path from one node to the next is denoted as hop. This means that a packet, which is transmitted from node A to node F along the path marked by arrows, travels a distance of four hops. To illustrate the wireless communication range of a node – the radio range – dashed circles are used as for node A in the example.

If a packet is sent from node A to node D it has to be forwarded by B since node D is not in the radio range of node A. If node B forwards the packet, it acts as *relay* or more accurately as router (cf. chapter 3).

2.1 Introduction

In the last two decades, the digital mobile communication services grew rapidly. In the 1990s, the digital services started as the second generation of mobile communications with the *Global System for Mobile Communications (GSM)* improved by the *General Packet Radio Service (GPRS)* and *Enhanced Data Rates for GSM Evolution (EDGE)*. The third generation used the *Universal Mobile Telecommunications System (UMTS)* with the *High Speed Packet Access (HSPA)* improvement. Nowadays, the fourth generation of such services is in use which is called *Long Term Evolution (LTE)*. To provide fast and easy access to the Internet in a lot of different places, the number of *Wi-Fi* hotspots is increasing rapidly. *Wi-Fi* stands for a trademark which specifies devices for the *Institute of Electrical and Electronics Engineers (IEEE)* 802.11 standard and is a subgroup of *Wireless Local Area Network (WLAN)*. In this work both names are used interchangeably.

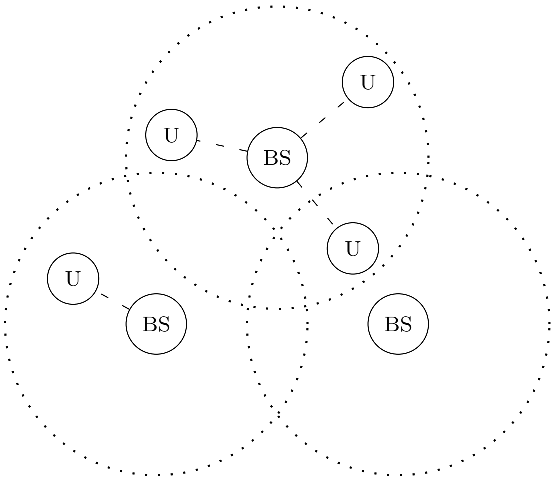


Figure 2.2: An Exemplary Infrastructure Network with Three Base Stations (BS) and Four Users (U)

All the previously mentioned communication standards, except *Wi-Fi*, are based on infrastructure networks only (cf. figure 2.2) which means that the base stations are usually static, but the users can be mobile. Furthermore, the users cannot communicate directly with each other, not even if they are in each other's communication range. Each user can only communicate with its base station which in turn forwards the information. If a node is not inside the transmission range of a base station it is not able to communicate, other devices between this node and the base station cannot act as *relays*. Therefore, the network coverage has to be considered when designing such networks.

All infrastructure networks require previously installed hardware including radio towers, wired data connections and *Backbone*, for example. The weaknesses of such infrastructure networks are the high acquisition costs for the installation which leads to the facts that these networks are uneconomic in sparsely populated areas, that it takes relatively long to assemble them and that these networks are administrated by a centralized instance which could represent a single point of failure.

Since infrastructure networks usually are not available in isolated areas like in disaster scenarios or military operations or such infrastructure based networks are still too expensive, e.g. satellite connections, the *Defense Advanced Research Projects Agency (DARPA)* started the development of the *Packet Radio Network (PRNET)* [Kah75] in 1973 to connect about 50 wireless devices with each other without any given infrastructure. This was the beginning of the ad hoc networks (cf. figure 2.3).

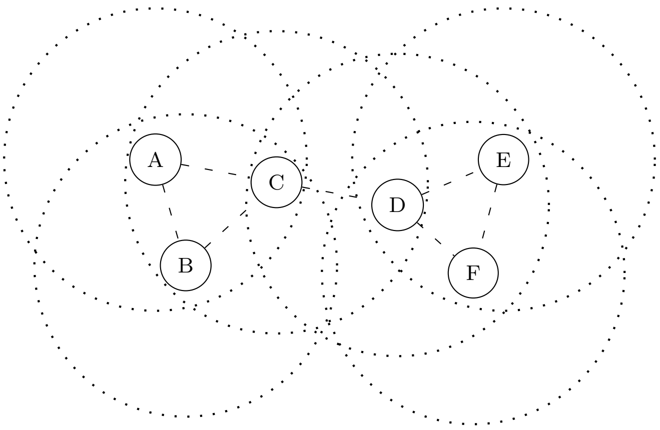


Figure 2.3: An Ad Hoc Network with Six Nodes and Visualized Transmission Ranges

Ad hoc networks are usually constructed for a specific task without the need of any previously installed communication infrastructure. Instead, the nodes autonomously create a wireless network and each node communicates directly with its direct neighbor nodes without the need of base stations. The direct neighbors of a node are the devices which are in the direct radio range of the node. If the destination of a transmission is not a direct neighbor of the source node, it is not possible to communicate directly. However, the other nodes in an ad hoc network act as *relays* and can forward the packets. With this multihop feature, ad hoc networks are very scalable and robust against single node failures. These networks are highly adaptive: The participants can enter or leave the network, they can move around and the network can split into multiple parts and merge again.

If the devices in such a network are mobile like walking pedestrians, driving cars or flying helicopters, the network is called *MANET*. In this dissertation, the terms ad hoc network and mobile ad hoc network are used interchangeably. Another type of ad hoc network is the mesh network which has one or more connections to infrastructure networks. If an ad hoc network has only one connection to a fixed network and does not forward foreign packets for transit, it is called a stub network.

As outlined, ad hoc networks are very flexible which makes them most suitable for the use in disaster scenarios. In the next section, the requirements for ad hoc networks and the functionality of such networks will be described.

2.2 Characteristics of Ad Hoc Networks

MANETs have some major advantages over fixed networks in disaster scenarios. They can immediately be deployed in the absence of infrastructure networks, they are robust against external influences, they do not need any administration and they are a low-cost solution for communication. However, this type of networks has some characteristics (c.f. [CM99]) which have to be considered critically.
The nodes in such networks are heterogeneous in terms of their available transmission power and their antenna design which could be omnidirectional, bidirectional, or have an adjustable angle. This results in different transmission ranges of the single nodes and therefore, in unidirectional and bidirectional links between the nodes.

Furthermore, the devices in ad hoc networks have energy limitations because they tend to be battery powered. This limits the *Central Processing Unit (CPU)* power, hence the complexity of used algorithms inside a node. Moreover, each node frequently has to calculate routes and to forward packets which consumes a lot of energy. The exhaustible energy reserves should be born in mind when adjusting the transmission power of a node since it depends quadratically on the transmission range, cf. equation 2.1[1] [JS11], where P_T is the transmitted power, P_R is the received power, d is the distance between the nodes and λ is the wavelength of the radio signal.

$$\frac{P_T}{P_R} = \left(\frac{4 * \pi * d}{\lambda}\right)^2 \tag{2.1}$$

In *MANETs*, the nodes are typically mobile. Therefrom, the topology of a network can change rapidly and existing links between nodes break while new links occur. This behavior affects data transmissions and lowers the available throughput. Furthermore, it is the nature of wireless networks that their communication is disturbed by noise and affected by fading which implies interference, shadowing and multipath propagation. Another negative effect on the available throughput results from the media access when multiple devices compete for accessing the same radio channel. In wired networks, the nodes can use *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)* to identify collisions directly when they happen. With this method, a node senses the medium to verify its emitted value. In wireless networks this could be impossible since some transceivers only operate in half-duplex mode and therefore, they are not able to sense their own signal. Another problem when sensing the own signal could be caused by received noise that disturbs the signal. Rather, the *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)* technique is used to avoid collisions on the channel by sensing the carrier before accessing it. However, this does not guarantee the avoidance of any collision, since two nodes could sense the channel at exactly the same time and afterwards try to access it. The resulting collision is comparable to the *hidden node problem*.

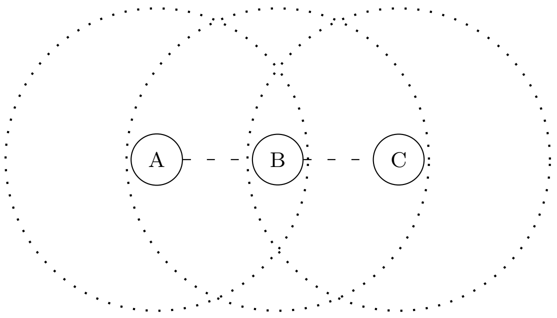


Figure 2.4: The Hidden Node Problem Causing Interference at Node B

[1] This equation does not take into account any disturbances of the radio signal like objects in the Fresnel region.

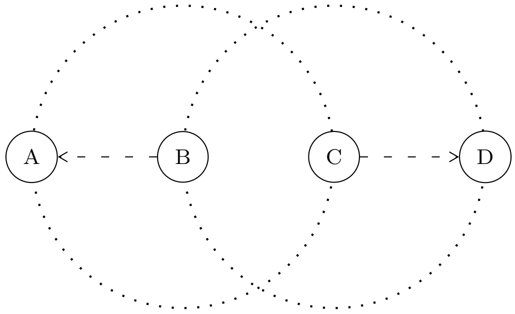


Figure 2.5: The Exposed Node Problem

In the *hidden node problem* situation (cf. figure 2.4) two nodes A and C send data to node B. The two sending nodes cannot detect each other because they are out of each other's communication range. This results in interference at the receiver node B and, therefore, in invalid received data.

The widely used standard *IEEE* 802.11 utilizes *Acknowledgement (ACK)* packets to confirm correctly received packets. If the *hidden node problem* occurs, the source nodes will detect the absence of the *ACK* packet and retransmit the packet. Moreover, *IEEE* 802.11 provides an optional *Request To Send (RTS) / Clear To Send (CTS)* sequence. In the example, node A will send *RTS* to signalize its wish for sending. Node B will answer this request with *CTS*, which additionally tells all other nodes in the radio range that the channel is now occupied. After the data transmission from node A to B, node B will send *ACK* to confirm the received data.

A further problem that cannot be solved by *CSMA/CA* is the *exposed node problem* (cf. figure 2.5) which occurs if there are two or more pairs of nodes which try to communicate. Node B transmits to node A and node C recognizes the channel access of node B. Since node C detects an occupied channel it does not start to transmit its data to node D. However, the transmission from C to D would be possible because node D does not receive B's signals. This leads to unused capacities in the network. This problem could be solved by directional antennas or by using different radio channels for the transmissions.

Beside the problems with the media access, the distribution of *Internet Protocol (IP)* addresses is a big challenge in such networks. Centralized institutions like *Dynamic Host Configuration Protocol (DHCP)* servers should not be used because they could represent *Single Points of Failure* or goals for attacks. Furthermore, as multiple ad hoc networks can merge and separate anytime, it must be guaranteed that each address in the network is only used once to avoid collisions. It would be also possible to detect such address collisions and resolve them afterwards.

A similar problem is the name resolution in such networks because centralized *Domain Name System (DNS)* servers could fail. Therefore, they are not suitable for the use in ad hoc networks for disaster scenarios. To overcome this problem, approaches like [FSS$^+$12, SBH$^+$13] which operate in completely decentralized ways were proposed for the use in ad hoc networks. To make the network more robust features like decentralized service discovery (cf. [SSK$^+$14]) could be implemented.

Beside the technological limitations of ad hoc networks, the human made security aspects have to be considered. Ad hoc networks, especially for disaster scenarios, are developed to be easily created and to allow nodes to attach to or detach from the network. This allows eavesdroppers to simply overhear transmissions and attackers to simply inject spoofed packets. With spoofed packets it would be possible to inject invalid routes into the routing tables, to redirect packets via *Address Resolution Protocol (ARP)* spoofing for a *Man-in-the-middle* attack or to use *IP* spoofing to fake the source address of *IP* packets. The last case could be a problem in disaster scenarios, if an attacker masquerades as firefighter or paramedic, for example, and sends messages like ‘help is coming’ to victims who will afterwards stop sending help requests.

Since an ad hoc network is usually organized decentralized, the failure of single nodes would not yield to the complete failure of the network. However, single services in the network which may be centralized like webservers or fileservers could fail if *Denial of Service (DoS)* attacks are used, for example. If the network uses the *DNS* for name resolution, this provides a single point of failure. However, new technologies like ‘Address Resolution in Mobile Ad Hoc Networks using Adaptive Routing’ [FSS$^+$12] were developed to overcome such problems. A further weak point is the injection of packets for flooding the whole network to exploit the scarce resources for the attack.

Another important aspect for ad hoc networks is the acceptance by the users. New technologies get only a chance if they are reliable and secure. This is especially important in disaster scenarios. To achieve acceptance, the previously mentioned security aspects should be solved. Furthermore, new *Quality of Service (QoS)* approaches need to be developed to provide the same features for ad hoc networks as for wired networks. However, *QoS* is a challenge in ad hoc networks where connections could break anytime.

Besides, it should be considered whether each node has to relay foreign packets or if it is allowed to ignore foreign packets if required. This could be of interest if a node has very little energy remaining or if the user of this node simply does not want to relay other packets. One solution for this could be different rates for network usage in a way that users that relay foreign traffic pay a lower fee than other users; this idea is often mentioned in literature (c.f. [Deb09]). However, it might be objected that it could also work without any regulations like in the 'Freifunk' community [För] or in the 'Tor Project' [The] which do not require usage fees. Rather, it is the users' decision if they offer their Internet connection to others or not. It is a way of self-regulation in these networks. If the number of users exploiting the network increases, the available throughput per user decreases. Thus, more users have to contribute to the network by adding more connections to the Internet or the number of users will decrease because the network gets too slow. At this point, it should be considered that forwarded foreign data could contain illegal content. This implies data packets of file sharing services like parts of music or video files, for example, and could cause problems to the user who forwards the packets into the Internet.

2.3 Conclusion

As depicted in this chapter, the typically used infrastructure-based networks are not available in all fields of operation. In some fields, they are too expensive, not flexible enough or simply not practicable. Ad hoc networks are predestined for such areas and can be used to create independent networks or to extend infrastructure networks. They are very flexible and provide a fast and inexpensive solution for communication in areas where no communication infrastructure is available. However, beside all the advantages of such networks, this chapter presented some issues which should be attention paid to. Examples are the media access and security aspects. Furthermore, the acceptance by users was discussed and showed that future work in this area is required to establish ad hoc networks as the standard for communication in disaster scenarios. Therefore, it is hard to predict if ad hoc networks will become accepted within the next years. It is also conceivable that infrastructure networks and ad hoc networks will coexist in the future and aid one another.

After the creation of an ad hoc network, the participating nodes can only communicate with their direct neighbor nodes, multihop transmissions are not possible. The following chapter 3 introduces the routing protocols which enable route discovery and route maintenance to use multihop routes between the nodes.

3 Routing in Ad Hoc Networks

As shown in the previous chapter, ad hoc networks can vary in their size from a few meters to some kilometers. Since the typical transmission range of *Wi-Fi* devices in the free range can be expected as around 100 meters, it becomes clear that a direct connection between two nodes at the opposite sides of a network might not be possible. A solution is to use the intermediate nodes as *relays*, respectively routers, which will forward the data packets towards their destinations. Therefore, the nodes need to know the available routes between them.

This very important task is done by routing protocols. Due to the fact that ad hoc networks have limited *bandwidth* and energy resources, special routing protocols have been developed to cope with these requirements.

In the following sections, the basics of routing protocols for ad hoc networks are shown, starting with an introduction to ad hoc routing, followed by potential problems. Afterwards, the requirements for ad hoc routing protocols are discussed and a brief overview about some important routing protocols is given.

3.1 Introduction

Multihop connections in ad hoc networks are only available if the nodes know where to forward received packets to. The next intermediate node on a packet's way towards the destination is the next hop. If the packet is not destined for the processing node itself, the node needs some information about this next hop. The forwarding of data packets towards the destination, typically done by the *IP*, is located in the *network layer* (layer 3) of the *Open Systems Interconnection (OSI)* model and depends strongly on provided information about available routes between the nodes in the network. This information can be provided as a static routing table containing some pre-defined routes or as a dynamic routing table, together with a routing protocol which discovers routes in the network, stores them into the routing table and maintains such routing table entries afterwards. If a new route is found, it is stored inside the routing table of the respective node. Such a routing table entry consists at least of the address of the destination node and the information about the next hop. If a node wants to forward an *IP* packet, it queries its routing table to get the address of the next hop.

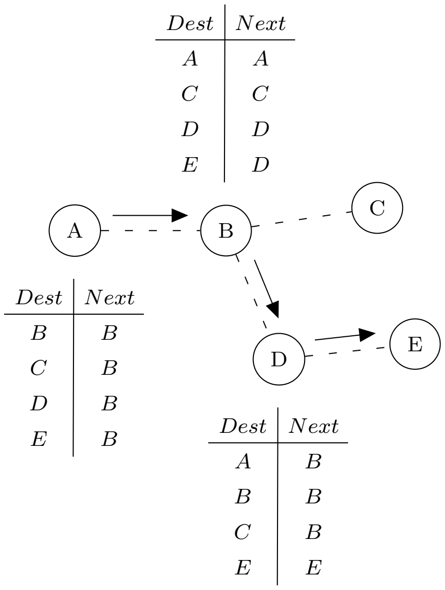
The scenario in figure 3.1 shows the forwarding of an *IP* packet from node A to node E and the corresponding routing table entries. When node A wants to transmit a packet, it queries its routing table for the next hop towards the destination node E and receives the address of node B as next hop. At this point, node A forwards the packet to its neighbor node B, which queries its routing table and gets node D as next hop towards the destination E. Therefore, it forwards the packet to D, which subsequently forwards the packet to node E. It should be noted that this example uses static routing tables. If the nodes are mobile and hence the available routes in the network change over time, a routing protocol should be used to permanently update the routing tables which are dynamic in this case.

Beside the minimalist routing table entries from the previous example, such entries can be extended by information about the cost of the route, the route's lifetime or routing protocol specific information. With the help of such information, a node can choose the route depending on its cost. In this context, the term 'cost' does not only imply money, but also energy, bandwidth, or robustness, for example.

As shown in chapter 2, ad hoc networks can be highly dynamic and therefore the routing protocols should be able to adapt rapidly to changing network conditions. This means that after a change in the network, new routes should be available immediately when required.

3.2 General Issues

As shown in the previous chapter, ad hoc networks have some major advantages over traditional networks. They can be deployed rapidly anytime and anywhere without high acquisition costs, they are robust and self-organized. However, they also have some disadvantages which should be considered when working with this type of networks.



Dest	*Next*
A	*A*
C	*C*
D	*D*
E	*D*

Dest	*Next*
B	*B*
C	*B*
D	*B*
E	*B*

Dest	*Next*
A	*B*
B	*B*
C	*B*
E	*E*

Figure 3.1: Forwarding of an IP Packet from Node A to Node E with the Corresponding Routing Table Entries

Since most of the devices are battery-powered which implies that they only have very little energy resources, they cannot execute very complex algorithms. Furthermore, if a node's battery is completely depleted, the node fails and existing routes maintained by that node fail as well. Figure 3.2 shows this example where node A and node C communicate with each other over a route which contains node B with very little remaining energy. If node B fails, a new route over node D and E has to be available as soon as possible. Data transmissions in such a case could be interrupted if the routing is not able to instantly offer an alternate route. Such interruptions are fatal in the use of real-time applications like videoconferences or phone calls. A solution for this problem could be a routing algorithm maintaining multiple paths to a destination to be able to switch from the broken route to another one without any delay.

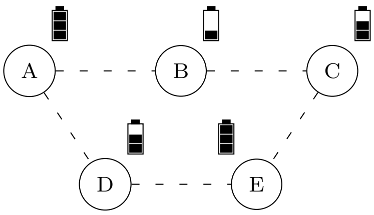


Figure 3.2: Re-Routing after a Node Failure

Beside problems with battery power supply, which could be predicted by measurements, unforeseen failures can occur. This includes the failure of nodes due to fire, radioactivity or undue forces like in warfares, for example. If the ad hoc network spans a wide area, regions with different routing requirements can appear. As seen in section 1.1, no routing protocol performs well in all possible network constellations. This leads to the fact that in such scenarios, multiple routing protocols should be used, each protocol for a special region. This results in the challenge that areas using different routing protocols are required to be interconnected. The same problem occurs if an ad hoc network is meant to be coupled with an infrastructure network like the Internet or if the ad hoc network is supposed to be used to extend the range of a base station. In such cases, two or more routing protocols have to communicate with each other.

Another challenge is the refreshing of routes. Every node in the network is reliant on the information in its routing table. If the route information is outdated, the node cannot forward data packets towards their destinations. Expired routing information will cause an increase of the traffic in the network, because the nodes forward the packets to the wrong neighbors. The refreshing of routes does not only mean updating existing routes in this context, but also finding new routes and deleting obsolete routes.

3.3 Routing Technique Requirements

Beside the problems shown in the previous section, there are also some general requirements for routing protocols which have to be fulfilled. These characteristics are essential for the accurate operation of the routing.

A very important requirement of a routing protocol is that it has to be loop free. This means that it has to be guaranteed that a data packet will never be forwarded to a node which previously processed this packet.

Figure 3.3 shows an example for such a routing loop. In the first step (cf. figure 3.3a), node A communicates with node C using a route via node B. When the connection between node B and C fails (cf. figure 3.3b), the packets from node A cannot be forwarded at node B anymore. For the case that node B additionally knows the route via node A to node C, but node A does not know its direct connection to C, a routing loop would occur (cf. figure 3.3c) where node A sends its packets to node B and node B forwards them back to A. In such a case, the packets would be sent back and forth until the lifetime of the packets decreases to zero or the routing table entries are updated. This produces traffic in the network without any benefit. Therefore, routing loops have to be avoided since they heavily increase the network traffic and consume much bandwidth and energy.

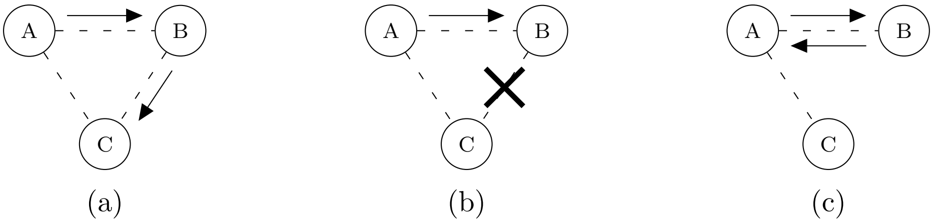


Figure 3.3: Example for the Creation of a Routing Loop

Moreover, a routing protocol should work reliably in all scenarios to provide routes for any transmission and during the entire duration of the transmissions. In addition to the absolutely required characteristics of ad hoc routing, there are some 'nice-to-have' qualities which will be discussed in the next paragraph.

One optional but important feature is the security of the routing. In section 2.2, the security for ad hoc networks has already been discussed, but for routing protocols there are some additional types of attacks like the injection of invalid routes to detour packets or the exhausting use of network resources to consume energy and *bandwidth*. If a first-aider contacts a victim to convey that help is coming, it should be ensured that this message is authentic as the victim could stop help requests after receiving the message.

Another goal is to optimize network performance. This could be done in terms of lowering the routing overhead which is consumed for control packets, by using less energy to prolong the lifetime of the network, by minimizing the number of hops per route, or by implementing some kind of load-balancing to increase the throughput and to distribute the traffic more uniformly in the network. Furthermore, it would be helpful if the routing was able to route multicast messages or to connect to infrastructure based networks. *QoS* support in terms of guaranteed bandwidth, transmission time, or *Bit Error Rate (BER)* could also be part of the routing.

Last but not least, the network should operate fully self-organized, self-administering and the implementation of the routing protocol should be as simple as possible. As mentioned in section 1.1, a single routing protocol cannot perform well in all possible network scenarios since it is not feasible to implement into one protocol all features shown in this section. For example, a routing protocol cannot provide high security features and a simple implementation at the same time and it is also impossible to combine very high data rates and a long battery lifetime.

3.4 Routing Methods Overview

In the past years a lot of effort has been spent for the development of routing protocols which fulfill the requirements of ad hoc networks. This resulted in a substantial quantity of different protocols which can be classified into two main groups, which are *table-driven* and *on-demand*.

The table-driven protocols, also called proactive routing protocols, discover and maintain routes frequently and before they are needed. To achieve this, the protocols constantly exchange some routing packets to refresh the information in the routing tables and to prevent inoperative routing table entries. In this process the routing has to make a tradeoff between the produced routing overhead and the freshness of the routes. More up-to-date routes result in more routing overhead, while reducing the routing overhead leads to routes which could be outdated. On account of the periodical updates which disseminate in the whole network, this type of protocol is less adequate for the use in highly mobile scenarios since it reacts relatively slowly to changes in the network structure. Two widely used protocols of this family are *Destination Sequence Distance Vector (DSDV)* [PB94] and the *Optimized Link State Routing Protocol (OLSR)* [CJ03].

The other group consists of the on-demand protocols which are also called reactive or source initiated routing protocols. This type does not discover routes until they are required, with the benefit of very low routing overhead during phases when the network is idle. When a node requires a route which does not exist in its routing table it sends a route request into the network. If the destination is present, the node receives a reply containing the information about the route. However, this request/reply sequence causes some latency. During this period a node has to wait for the reply and cannot forward its data packets directly. Furthermore, if the number of nodes in the network increases, the routing overhead could rise heavily, resulting in network clogging. Examples of well-known reactive protocols are *Dynamic Source Routing (DSR)* [JHM07] and the *Ad hoc On-Demand Distance Vector (AODV)* [PBRD03] routing.

As the two routing protocols *AODV* and *OLSR* are used in the current *SEREMA* implementation they will be presented in detail in the following sections. This information is required as basic knowledge for the subsequent chapters.

3.5 Optimized Link State Routing Protocol (OLSR)

In this section the functionality and the characteristics of the *OLSR* protocol will be discussed. As seen in the name of the protocol it uses link-state information for route discovery. More specific, this means that a node broadcasts information about the connections to its direct neighborhood into the whole network. Any other node in the network accumulates such gathered information and, afterwards, it calculates all possible routes in the network using the Dijkstra algorithm [Dij59].

Briefly explained this means that the node generates a graph containing all nodes and connections of the network. Then it starts at its own position in the graph and follows the shortest path to the nearest node. This node will be inserted into the routing table with the obtained path length. Subsequently, it searches the node with the second shortest path measured from its own position and inserts this node into the routing table. This algorithm is continued until all nodes of the graph have been reached and added to the node's routing table.

Routing information is exchanged between the *OLSR* nodes using a standardized packet format (cf. figure 3.4) which is usually transmitted inside *User Datagram Protocol (UDP)* packets addressed to destination port 698 [Intb]. The *Packet Header* contains the length of the packet in bytes and a *Packet Sequence Number (PSN)* which is maintained per interface and incremented by one each time the node sends a packet over the interface. Such *PSN* can be used to detect transmission failures from neighbor nodes by checking if received packets have a monotonically increasing sequence number.

The Packet Header of a valid *OLSR* packet is followed by one or more pairs of *Message Header* and *Message*. To inform the receiver about the following message the Message Header contains the *Message Type* which will be discussed later in this section. The *Vtime* field holds the time during which the information in the message should be considered as valid. The value is expressed in a mantissa and exponent notation which can be found in subsection 3.2.2. of [CJ03]. Subsequently, the header contains the complete message size in bytes, the network address of the message's originator node, the *Time To Live (TTL)*[1] representing the number of hops the message is allowed to travel, the *Hop Count* and finally a *Message Sequence Number* which is generated by the originator node. This sequence number is not allowed to be changed by nodes other than the originator node of the message and enables other nodes to drop messages, that have previously been processed, by recognizing the reception of the same Message Sequence Number more than once.

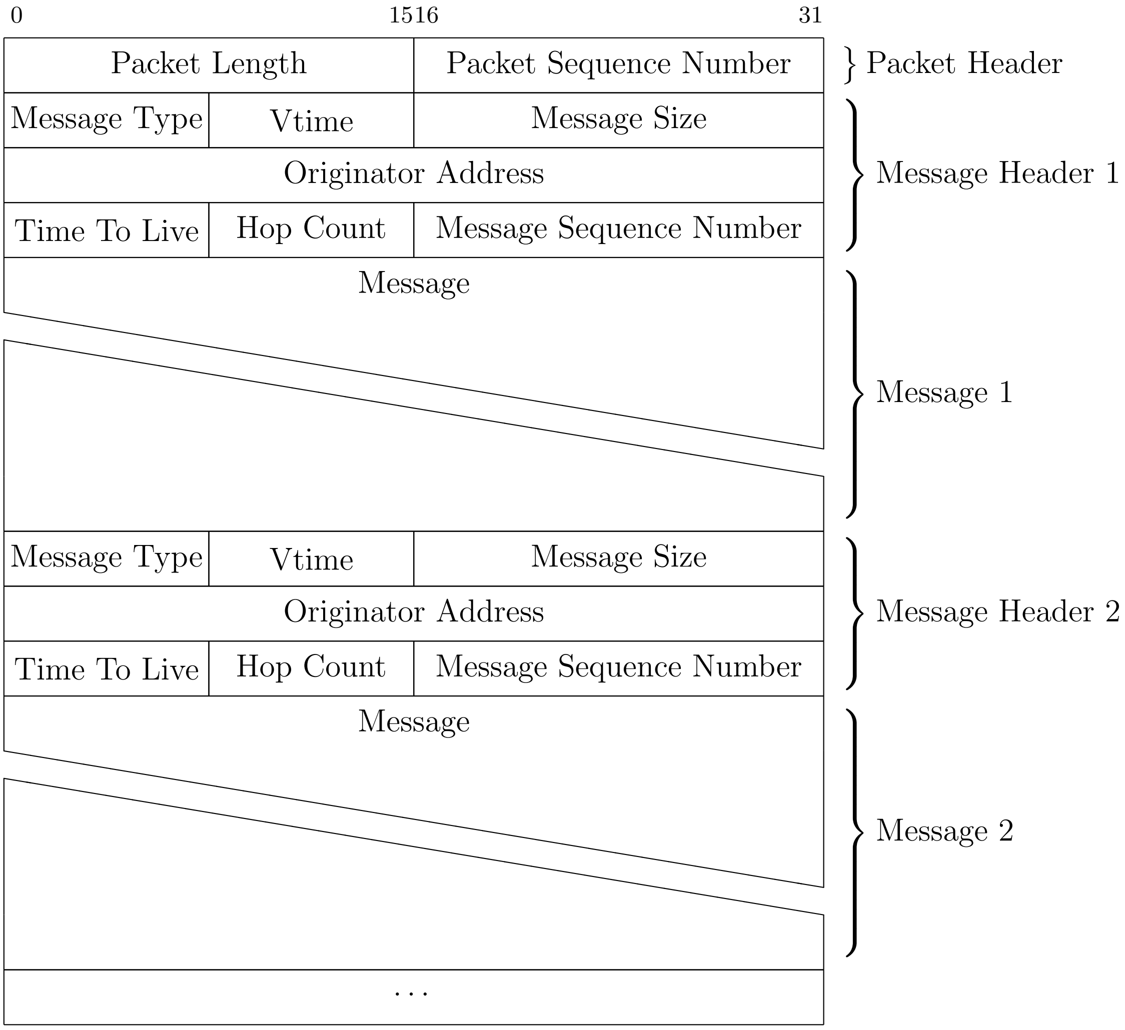


Figure 3.4: The Structure of an OLSR Packet

OLSR uses different message types for its operation which are encapsulated in the standardized packet format. The detailed functionality of the protocol will be introduced in the following subsections. Therefore, the protocol's operation will be split into three main stages namely *Neighbor Sensing*, *Multipoint Relay (MPR) Selection* and *Dissem-*

[1] For hop count limitation the *TTL* field in the *IP* header is usually used. However, as multiple *OLSR* messages could be transmitted in the same *IP* packet, a separate *TTL* field is required in the *OLSR* Message Header.

ination of Topology Information which will be continuously repeated as described by Forde [For05].

3.5.1 Neighbor Sensing

As *OLSR* is a link-state protocol, in the first step the nodes accumulate information about connections between themselves and their direct neighbor nodes and assess if these connections are bidirectional or unidirectional. For such neighbor detection each node periodically broadcasts *HELLO* messages (cf. figure 3.5) containing all information about its interfaces and the neighbor nodes which are available over the connections.

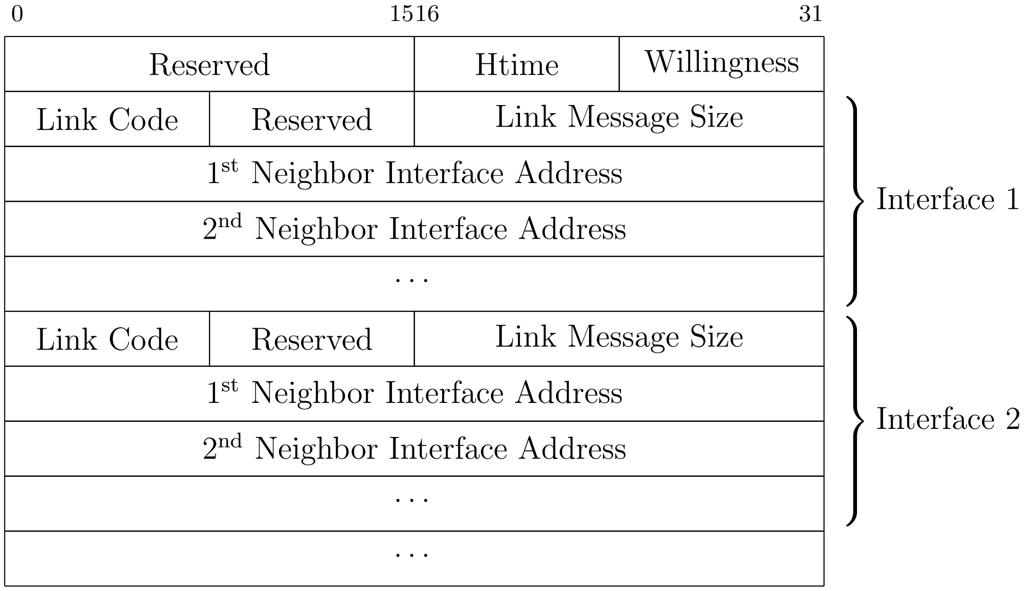


Figure 3.5: The structure of an OLSR HELLO Message

At the beginning of a *HELLO* message the originator node advertises its *Htime* value which informs about the node's *HELLO* emission interval which can be used to detect 'long periods of silence' (cf. [CJ03] section 14.3.) indicating link failures. Subsequently, the header contains the willingness to become a *MPR* (cf. subsection 3.5.2 for details) using an integer value from zero to seven, where lower values mean that the node does not want to become an *MPR*. Very low remaining energy could cause a node to refuse such functionality for example.

Directly after the general section of a *HELLO* message, the information about a node's interfaces follows. This includes a *Link Code* field providing information about the links between the node and its neighbors connected via this interface. More information about the representation of the bits in this field is presented in subsection 6.1.1. in [CJ03]. The next piece of information contains the *Message Size* which submits the number of bytes used for the whole header part of a single interface, starting at the Link Code and ending directly before the next Link Code or at the end of the *HELLO* message. Directly behind, there is a list of the neighbors' interface addresses.

Such *HELLOs* are broadcast on each of a node's interfaces towards its direct neighbors. Since these messages are only destined for the direct neighbors of a node and therefore the message header *TTL* is set to a value of 1, the neighbor nodes are never allowed to forward such messages.

3.5.2 MPR Selection

When the node has detected all its neighbors, the next step will be broadcasting this information into the whole network. However, this would induce a lot of traffic and could cause congestion in the network. Therefore, *OLSR* uses *MPRs* to reduce the generated traffic compared to simple flooding. This is done by selecting nodes in the 1-hop neighborhood which are responsible for forwarding packets to all 2-hop neighbors. The other nodes in the 1-hop neighborhood will only receive and analyze the packet but not forward it. The algorithm for calculating the *MPR* set is described in subsection *8.3.1.* of the *OLSR* specification [CJ03].

Figure 3.6 shows an example for the dissemination of a message using *MPRs*. Let the nodes A and B act as previously selected *MPRs*. The originator node O generates a message and sends it to its 1-hop neighborhood. Node 1 will receive the message and learn from it. However, since node 1 has not been selected as *MPR*, it does not forward the message. The A nodes are *MPRs* and therefore, they forward the message to node O's 2-hop neighborhood. The nodes which are in the 2-hop neighborhood of node O are in the 1-hop neighborhood of the A nodes. This implies that the nodes with number 2 will not forward the message since they have not been selected as *MPRs* by the A nodes. On the other hand all B nodes are *MPRs* and will forward the message. This simple example illustrates that a message can be broadcast to 17 nodes by only being forwarded by four intermediate nodes acting as *MPRs*. When selecting the *MPR* nodes a node should try to minimize the number of nodes. If a node selects all neighbors in its neighborhood as *MPRs* the network behaves like using simple flooding. After selecting the *MPRs*, a node begins to disseminate its topology information to the other nodes in the network.

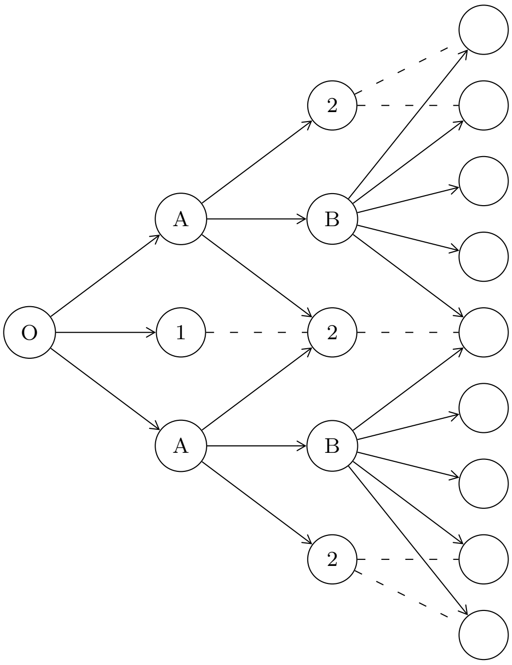


Figure 3.6: The Dissemination of OLSR Messages Using Multipoint Relays

3.5.3 Dissemination of Topology Information

Each node sends *Topology Control (TC)* messages to all other nodes in the network using the *MPRs*. Such *TC* messages contain information about a node's neighborhood and are accumulated by the other nodes in the network to create a graph containing all nodes in the network including their connections. This graph will afterwards be used for route calculations.

0	1516	31
ANSN	Reserved	
1st Advertised Neighbor Main Address		
2nd Advertised Neighbor Main Address		
. . .		

Figure 3.7: The Structure of an OLSR TC Message

Figure 3.7 introduces the structure of a *TC* message containing an *Advertised Neighbor Sequence Number (ANSN)* for checking the freshness of received *TC* messages.

Furthermore, the message contains the main addresses (usually *IP* addresses) of the originator's neighbor nodes. The terminology *main address* denominates the address which is used as Originator Address in a nodes *OLSR* message header. The nodes start calculating the routes between all nodes in the network after receiving the *TC* messages.

Afterwards, all available routes in the network are stored inside the *OLSR* routing table (cf. table 3.1). Each routing table entry consists of four values which are the address of the destination node, the address of the node that acts as next hop towards the destination, the distance in hops and the address of the nodes interface which has to be used.

At this point, it should be noted that a routing protocol does only provide available routes but it is not the task of the routing protocol to forward data packets. Rather, it is part of the Network Layer to forward packets using the information taken from the routing table.

1	Destination Address	Next Hop Address	Distance	Interface Address
2	Destination Address	Next Hop Address	Distance	Interface Address
3	...	...	...	...

Table 3.1: The Structure of an OLSR Routing Table

With *OLSR*, it is possible that a single node is equipped with multiple interfaces connected to the same network. Therefore, the node has to advertise its additional interfaces to the other network participants. This is done by sending *Multiple Interface Declaration (MID)* messages via broadcast and with a *TTL* of 255 into the network. The propagation of *MID* messages is done using the *MPR* nodes. All addresses of the specific node except its main address are simply stored inside the message as shown in figure 3.8.

0 1516 31

1st Interface Address
2nd Interface Address
...

Figure 3.8: The Structure of an OLSR MID Message

3.5.4 Additional Features

As previously mentioned (cf. chapter 2.1), ad hoc networks can be connected with infrastructure networks. *OLSR* provides a mechanism to support gateway nodes to exchange packets with other non *OLSR* networks. Therefore, *Host and Network Association (HNA)* messages are available which tell the other nodes in the entire *OLSR* network that the originator node has a connection to another network. Such *HNA* messages contain the *Network Address* of the foreign network and the corresponding *Netmask* (cf. figure 3.9). The messages are forwarded like *MID* messages using only the *MPRs* and with a *TTL* set to 255 to reach the entire network. However, those *HNA* messages in *OLSR* [CJ03] have a limitation. The messages do not contain a *Hop Count* information and, therefore, it seems for all the *OLSR* nodes that the non *OLSR* nodes have the same distance in hops as the *HNA* gateway itself. If different *OLSR* networks are connected via multiple *HNA* gateways this could result in routing loops [Dea05].

0	1516	31
	1^{st} Network Address	
	1^{st} Netmask	
	2^{nd} Network Address	
	2^{nd} Netmask	
	. . .	

Figure 3.9: The Structure of an OLSR HNA Message

Furthermore, it is possible to implement own extension messages for *OLSR*. The message type numbers ranging from 1 to 255 are divided into two groups and presented in table 3.2. The message types in the first group (1–127) are used for standardized protocol functionalities and are assigned by the *Internet Assigned Numbers Authority (IANA)* [Inta]. The second part (128–255) is reserved for private use. The message types in the range from 128 to 131 are used by previous approaches [SBH+13, SSS+13, SSK+14, Sch14] for name resolution and service discovery while the message number 132 is used by the present work as described later in subsection 6.6.2.

Message Type	Description	Note
1	HELLO Message	Assigned by IANA
2	TC Message	Assigned by IANA
3	MID Message	Assigned by IANA
4	HNA Message	Assigned by IANA
5–127	Unassigned	Reserved for IANA
128	NADV Message	Name Advertisement (NADV)
129	COLERR Message	Collision Error (COLERR)
130	SADV Message	Service Advertisement (SADV)
131	NERR Message	Name Error (NERR)
132	BNANNO Message	Border Node Annotation (BNANNO)
133–255	Reserved	Reserved for future use

Table 3.2: The OLSR Message Types

3.5.5 Applications

An important aspect when using routing protocols is to know in which network constellations the protocol performs well and when problems might occur. Therefore, the following aspects should be considered when using *OLSR*.

Since the nodes usually have a limited and relatively short transmission range, it cannot be guaranteed that each node is constantly connected to all other nodes. Isolated nodes or isolated areas of nodes could appear. This leads to an end-to-end reliability of less than 100 percent. Another aspect is that *OLSR* takes some time to exchange the topology information in highly mobile networks and, therefore, the routes could become invalid before they are used. This is because of the relatively long period between *OLSR* route updates. In reverse, this means that slowly moving nodes will increase the performance.

When *OLSR* discovers available routes, it generates *HELLO* and *TC* messages. Both message types grow with the density of nodes in the network as all the neighbors of a node are listed in these messages. This results in huge routing overhead which is again flooded into the whole network. However, since *OLSR* uses *MPRs*, such overhead can be minimized. Therefore, Clausen et al. [CJ03] stated in their work:

> ”The protocol is particularly suitable for large and dense networks as the technique of MPRs works well in this context.”

Furthermore, the protocol performs worse in networks with very low data traffic since *OLSR* periodically generates some routing overhead, even if there are no other data transmissions. However, this means in turn that it performs well in networks with a high number of data transmissions as its amount of generated routing overhead is more or less constant.

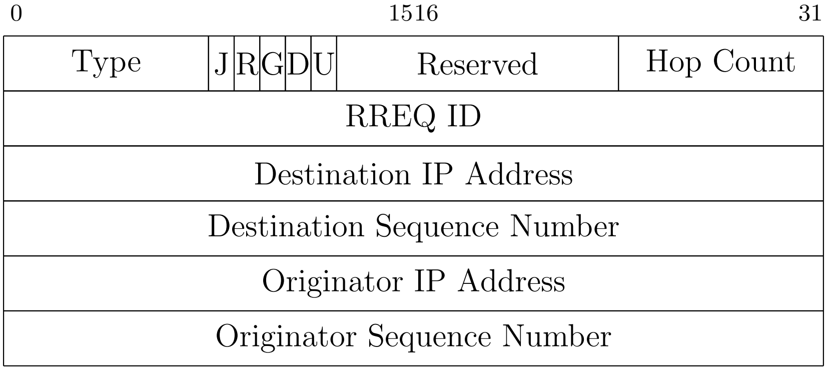
The main advantage of *OLSR* is its proactive functionality which results in immediate availability of routes without any previous latency for route finding. If a node wants to forward a data packet, *OLSR* will immediately provide an available route.

3.6 Ad Hoc On-Demand Distance Vector (AODV)

One of the most famous reactive routing protocols is *AODV* [PBRD03]. It uses sequence numbers to detect the freshness of received messages and to ensure loop freedom. The protocol operates in two steps. First, it searches for routes only on-demand to save traffic and secondly, it maintains the existing routes as long as they are in use. Therefrom, the following part of this section will describe those mechanisms.

3.6.1 Route Discovery

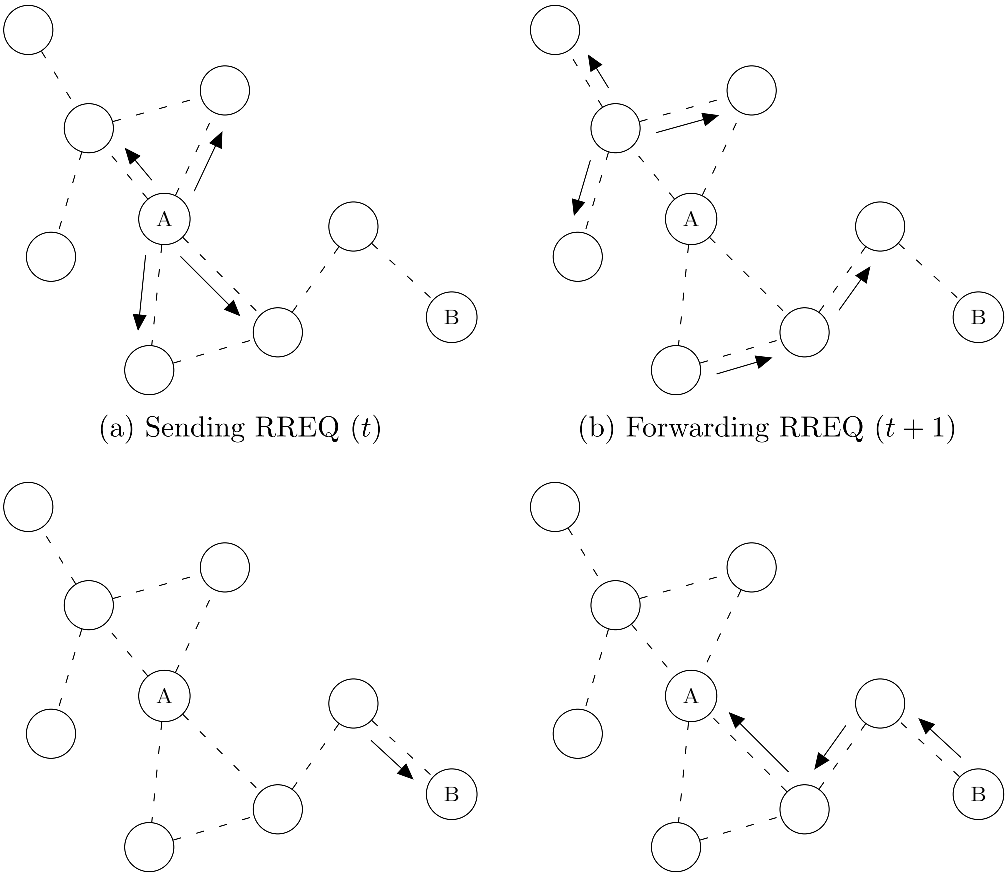
When a node wants to forward a data packet and does not know a route it buffers the packet and triggers a route discovery procedure (cf. figure 3.11a). This results in the generation of a *Route Request (RREQ)* (cf. figure 3.10) message which is broadcast into the network (cf. figure 3.11a) and usually addressed to *UDP* port 654 [Intb] which is used for *AODV*.



0 1516 31

<table>
<tr><td>Type</td><td>J</td><td>R</td><td>G</td><td>D</td><td>U</td><td>Reserved</td><td>Hop Count</td></tr>
<tr><td colspan="8">RREQ ID</td></tr>
<tr><td colspan="8">Destination IP Address</td></tr>
<tr><td colspan="8">Destination Sequence Number</td></tr>
<tr><td colspan="8">Originator IP Address</td></tr>
<tr><td colspan="8">Originator Sequence Number</td></tr>
</table>

Figure 3.10: The Structure of an AODV RREQ Message



(a) Sending RREQ (t)

(b) Forwarding RREQ ($t + 1$)

(c) Forwarding RREQ ($t + 2$)

(d) Receiving RREP ($t + \langle 3, 4, 5\rangle$)

Figure 3.11: The AODV Route Discovery Mechanism

The *Message Type* of *RREQ* messages is defined as a value of one. Each node maintains its own *RREQ ID* which is a monotonically increasing number. Before sending an *RREQ*, a node increments its own *RREQ ID* and sets the corresponding field in the message to this value. Therefrom, other nodes can use the originator node's address together with the received *RREQ* ID to detect multiple receptions of the same *RREQ*. Afterwards, the originator sets the *Destination IP Address* and if known the *Destination Sequence Number* or otherwise the *U-Flag* to signalize an unknown destination sequence number. Furthermore, the node inserts its own address into the *Originator IP Address* field, increments its own sequence number and sets the *Originator Sequence Number* field to the new value. Thus, other nodes are able to recognize obsolete messages.

The *AODV* protocol is designed to generate as little routing overhead as possible. Therefore, it uses a mechanism called *Expanding Ring Search* which limits the number of hops an *RREQ* is allowed to travel. This is done by limiting the *TTL* in the header of the *IP* packet. If the route discovery procedure does not find the destination, the node increases the *TTL* to continue searching. For each following request the originator increments its sequence number. To avoid congestion in the network the rate for sending *RREQ* messages is limited. If a route to the destination cannot be found, the node will drop the buffered data packets.

When a node receives an RREQ, it first learns the route to the previous node of the packet. Then, it checks if it has previously processed the same RREQ. If so, it drops the message. Otherwise, the node learns or, if the route existed previously, updates the route back to the originator – the *Reverse Route* – and updates its sequence number for the specific originator.

Afterwards, the node checks if it knows an available route towards the destination. If a route is available, the node's sequence number for the destination is equal to or larger than the received destination sequence number and the *D-Flag* which stands for *Destination Only* in the *RREQ* is not set, the intermediate node generates a *Route Reply (RREP)* and sends it to the originator of the *RREQ*. It also has to check the *G-Flag* in the *RREQ* to recognize if it has to generate a *Gratuitous RREP* to inform the destination node about the route to the originator. Otherwise, the resulting route between the originator and the destination could only be used unidirectional since the destination does not know the reverse route.

A *Gratuitous RREP* consists of an *AODV RREP* packet with the included fields set in a way that the packet looks like an answer to a *RREQ* from the destination node to the originator (*Hop Count* = Hops from the intermediate node towards the originator, *Destination IP Address* = Address of the originator, *Destination Sequence Number* = Originator sequence number, *Originator IP Address* = Address of the destination node, *Lifetime* = Remaining lifetime of the route towards the originator).

If the intermediate node does not generate an *RREP*, it checks if the received *IP* header *TTL* is larger than one and prepares the packet for forwarding. This implies the decrease of the *TTL* in the *IP* header by one, the increment of the hop count field in the *RREQ* by one and the update of the *Destination Sequence Number* field to the new value if the node knows a more recent number for the concerned destination. It is important that an intermediate node must not learn the destination sequence number received by an *RREQ* as this could not be the most recent sequence number of the node.

If the *RREQ* is forwarded (cf. figures 3.11b and 3.11c), it propagates towards the destination node. When received by the destination the node first creates or updates the route to the originator as described for intermediate nodes above. Subsequently, an *RREP* (cf. figure 3.12) is generated where the *Type* is set to two and the *Hop Count* starts with a value of zero. The node sets the *Destination IP Address* to its own address and writes its sequence number into the *Destination Sequence Number* field. If the node recognizes that the received *Destination Sequence Number* is equal to its own, it increments its sequence number to announce that the information in the generated message is up-to-date. The *Originator IP address* is set to the address of the node which originated the *RREQ* before while the *Lifetime* field is set to the number of milliseconds receiving nodes should consider the route as valid.

0			1516		31
Type	R	A	Reserved	Prefix S.	Hop Count
Destination IP Address					
Destination Sequence Number					
Originator IP Address					
Lifetime					

Figure 3.12: The Structure of an AODV RREP Message

When a destination node receives multiple *RREQs*, it only answers the first one as the others are duplicates received via longer routes. This behavior of *AODV* induces that the protocol cannot provide multiple routes per destination and if a route breaks a new route discovery process has to be started.

The intermediate nodes will forward the *RREP* on the previously learned reverse route back towards the originator (cf. figure 3.11d). During the delivery of the reply, unidirectional links between the nodes could cause problems (cf. figure 3.13). While node A can transmit the *RREQ* to node B, the reverse route for the *RREP* is not available since the transmission range of node B is smaller than the range of node A. To overcome this problem, the generator of an *RREP* has to set the *A-Flag* in the message which will make each receiver of the packet confirm its reception with a *Route Reply Acknowledgement (RREP-ACK)* message (cf. figure 3.14) to the previous node. The *Type* field of such a message is set to a value of four. If a node has forwarded the *RREP* and does not receive the acknowledge, it adds the next node to a 'blacklist' which causes it to ignore future *RREQs* from this node.

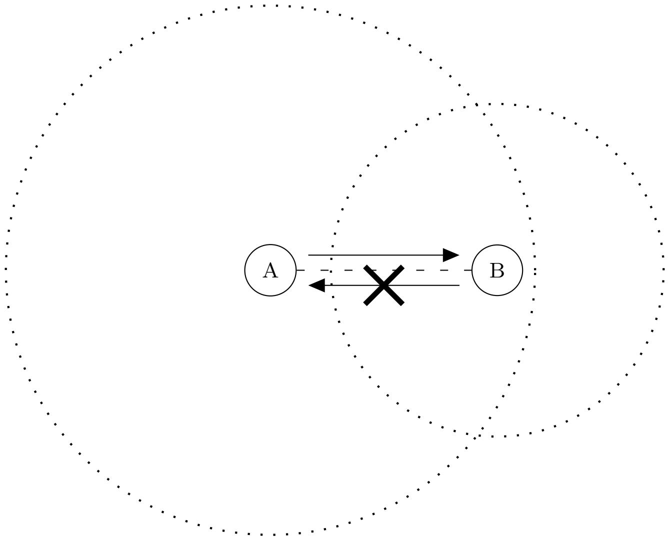


Figure 3.13: Different Transmission Ranges Causing an Unidirectional Link between Node A and Node B

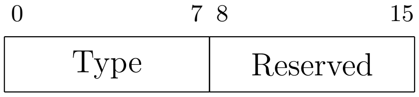


Figure 3.14: The Structure of an AODV RREP-ACK Message

3.6.2 Route Maintenance

Afterwards, when the route has been established, all nodes involved in the route have to maintain their routing table entries to keep the route alive as long as it is in use. As part of this task the nodes observe their neighbors to detect route failures quickly. This could be done on the *Link Layer* by sensing *ACKs* or *RTS/CTS*. However, as it is problematic in some implementations to sense the *Link Layer* with a routing protocol because of different layers in the network stack, *AODV* offers *HELLO* messages to sense connectivity information. Such messages consist of an *RREP* (cf. figure 3.12) with the *Destination IP Address* set to the node's own address. The *Destination Sequence Number* represents the node's latest sequence number, the *Hop Count* is set to zero and the *Lifetime* is set to the maximum allowed time between two *HELLO* messages. The message is transmitted in an *IP* packet with a *TTL* set to a value of one to limit the propagation to the immediate neighborhood only. If a node receives such a *HELLO* message and does not have a route to the neighbor, it adds the new route to its routing table. A node expects to receive a *HELLO* message, another *Network Layer* broadcast message or a *Link Layer* message periodically from each neighbor. If the absence of messages from a specific neighbor is detected, the node assumes that this neighbor is not available anymore. Thus, it handles the link failure as described next.

First, it sets the state of existing routes using the missing next hop to invalid since they are affected by the absence of the neighbor. Secondly, it determines which destinations are unreachable because of the failure. Afterwards, the node sends *Route Error (RERR)* messages (cf. figure 3.15) to inform all other nodes using the failed link about the failure. To identify such nodes each node maintains a *Precursor List*. This list contains the addresses of all nodes which use this node for forwarding. The *Message Type* is set to a value of three while the number of affected destinations attached to the message is given via the *Destination Count* field. If the source node of the traffic receives the *RERR*, it decides whether to trigger a new route discovery or if the route is not required anymore.

0		1516	31
Type	N	Reserved	Dest. Count
1st Unreachable Destination IP Address			
1st Unreachable Destination Sequence Number			
2nd Unreachable Destination IP Address			
2nd Unreachable Destination Sequence Number			
. . .			

Figure 3.15: The Structure of an AODV RERR Message

To avoid an interruption of the packet stream from the originator, a node can trigger a local link repair if it detects a link failure. Therefore, the node sends an *RREQ* asking for the destination node and buffers all packets that arrive from the originator node in the meantime. If the node receives an *RREP*, it compares the hop count to the one of the previous route. If the new route is longer, the node sends an *RERR* to the originator. In this message the *N-Flag* signalizes that the route should not be deleted because the intermediate node has done a local link repair. It is the decision of the originator node if it wants to use the new elongated route or instead triggers an *RREQ* to find a more optimal route.

3.6.3 Additional Features

In an *AODV* network, each node could be applied with multiple network interfaces. In this case the routing table contains the additional information about the interfaces to be used to reach a specific destination. A network operating *AODV* can also be connected to other networks which do not use *AODV*. However, the specification [PBRD03] does not give many details about this mode of operation.

The nodes in *MANETs* usually do not use *IP* addresses that are related to each other. For example the nodes with the addresses 111.111.111.111 and 222.222.222.222 could be part of the same ad hoc network. *AODV* supports the aggregation of multiple nodes to build a subnet with a common network address prefix like 192.168.1.1/24. In such a case one node is elected as *Subnet Router* to provide the connectivity between the subnet of aggregated nodes and the rest of the ad hoc network.

AODV provides extensions for the *RREQ* and *RREP* messages which can be used to add further features to the protocol. Figure 3.16 shows the structure of such an extension, which is added directly after the general message data. The *Type* of the message is represented by a number in the range of 1–255 where the messages 128–255 are not allowed to be ignored by a node. The *Length* gives the number of bytes of the following *Type Specific Data* which carries the new information. Such a message extension with the *Type* 133 is used by this dissertation and will be described in section 6.6.5. Table 3.3 gives an overview of already occupied message types.

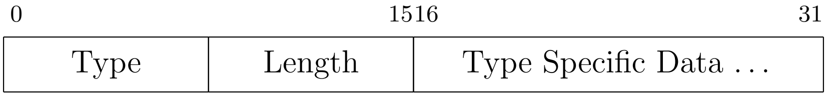


Figure 3.16: The Structure of an AODV Message Extension

Extension Number	Description
1	Hello Interval Extension [PBRD03]
2–127	Unused
128	*Name Request (NREQ)* [SBH⁺13]
129	*Name Reply (NREP)* [SBH⁺13]
130	*Name Error (NERR)* [Sch14]
131	*Service Request (SREQ)* [SSK⁺14]
132	*Service Reply (SREP)* [SSK⁺14]
133	*Tunnel Hops (THops)* (cf. subsection 6.6.5)
134–255	Unused

Table 3.3: The Different AODV Message Types Used in Protocol Extensions

3.6.4 Applications

One important goal of *AODV* is to generate as little routing traffic as possible which results in a good performance in scenarios with little data traffic. It can react quickly and efficiently to changes in the network because of its lightweight control packets, which are only exchanged when required.

However, the protocol could get into trouble in networks with high data traffic as described by Hoebeke et al. [Hoe07]. In such scenarios, packet drops could be interpreted by *AODV* as link failures. This triggers the generation of *RERR* messages and new *RREQs* which generate additional traffic in the already congested network. To overcome this problem proactive routing should be used in scenarios with high traffic. The same problem could occur in networks with a high node density as this implies a lot of nodes in a small area inducing routing overhead which leads to network congestion.

When using *AODV*, security should also be considered because the protocol does not provide its own security features. This means that an attacker could simply inject faked routes into the network by spoofing *RREP* messages or deleting existing routes by injecting *RERR* messages.

Another aspect that should be born in mind when using reactive routing is the latency for the route discovery procedure at the beginning of new data transmissions. If a node wants to send data packets and does not have a route, it has to buffer the packet, discover a route and only after that it can transmit the packets. When using real-time applications like videoconferencing this latency could cause problems because if a route fails and an alternative route has to be discovered first, the transmission could be interrupted. This problem would be less meaningful if *AODV* supported multiple routes per destination as described in the following section 3.7.

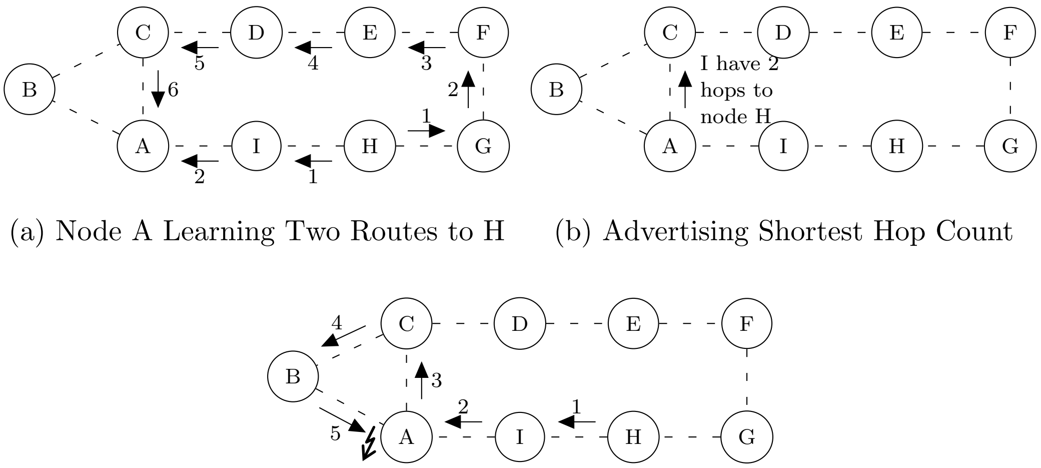
3.7 Ad Hoc On-Demand Multipath Distance Vector (AOMDV)

The previous section 3.6 introduced the *AODV* routing protocol, which maintains one route per destination. It was shown that this behavior could cause problems when the network is highly mobile and, therefore, a lot of links fail, corresponding routes get invalid and it takes some time to discover new routes. To make *AODV* more fault-tolerant and to decrease the average latency for route discovery, the *Ad hoc On-demand Multipath Distance Vector (AOMDV)* protocol [MD06] was developed. The protocol's basic operation is comparable to *AODV* since the new protocol is an extension of its predecessor. Therefore, this section only describes the differences in the protocol's behavior. *AOMDV* discovers multiple loop free (cf. subsection 3.7.1) and link-disjoint (cf. subsection 3.7.2) paths in each route discovery procedure. This provides the opportunity to avoid the generation of *RREQs* when links fail until all paths in the routing table become invalid.

3.7.1 Absence of Routing Loops

One of the most important tasks of a routing protocol is to guarantee loop free routes to avoid unnecessary traffic in the network. When modifying the *AODV* protocol, it is unavoidable to change its behavior for learning routes since the new routing algorithm has to handle multiple paths to a destination. Therewith, a node receives different hop counts for the paths to a specific destination and has to decide which hop count it should advertise to other nodes. Figure 3.17 shows a scenario with nine nodes where node A learns routes to node H. In figure 3.17a, node A receives a six hop route to node H via G–F–E–D–C and a second route via node I with two hops. Node A learns both *paths* when using multipath routing. Figure 3.17b assumes that A will advertise its shortest hop count to node C which in turn forwards this information to B. In figure 3.17c, node A receives a *path* to node H with a length of five hops. However, this *path* is via B–C–A–I and represents a routing loop as it redirects packets back to node A. Therefore, an *AOMDV* node is only allowed to advertise its maximum hop count to a specific destination instead of the shortest hop count. Furthermore, an *AOMDV* node is only allowed to learn *paths* which are shorter than the maximum hop count to a specific destination node. Regarding figure 3.17, this implies that node A will only advertise a hop count of six to the other nodes. In this case the routing loop does not occur because the route shown in figure 3.17c is not created as the information from node A 'I have six hops to node H' is not forwarded by node C to node B since C advertised a maximum hop count of five hops to the destination H and, therefore, it does not accept longer routes.

Furthermore, each node maintains one destination sequence number for each destination in its routing table. Multiple paths to the same destination share a common



(a) Node A Learning Two Routes to H (b) Advertising Shortest Hop Count

(c) Creation of a Routing Loop

Figure 3.17: Creation of a Routing Loop when Advertising the Shortest Hop Count

sequence number. For the maintenance of the maximum hop count towards a destination, *AOMDV* adds some additional fields to the *AODV* routing table (cf. table 3.4). The previously mentioned maximum hop count for a destination is stored in the *Advertised Hop Count* field and is locked every time the node advertises this hop count to other nodes until a new *Destination Sequence Number* for the considered destination is received. In such a case, the node deletes all *paths* in its *Route List* for the specific destination and rebuilds the list.

AODV	*AOMDV*			
Destination Address	Destination Address			
Dest. Seq. Number	Destination Sequence Number			
Hop Count	Advertised Hop Count			
Next Hop	Route List			
	Next Hop 1 Next Hop 2 . . .	Last Hop 1 Last Hop 2 . . .	Hop Count 1 Hop Count 2 . . .	Exp. Timeout 1 Exp. Timeout 2 . . .
Expiration Timeout				

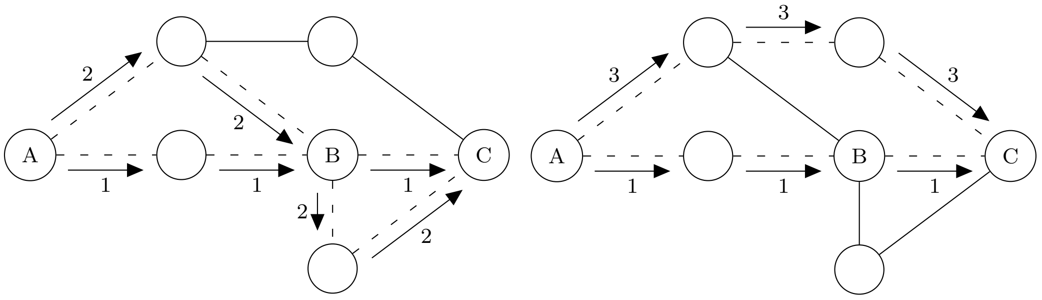
Table 3.4: Comparison between the Routing Table Structures of AOMDV and AODV

3.7.2 Path Disjointness

When a node uses multiple *paths* to a destination, it should be noted that this will not increase the fault tolerance, if those *paths* share common links or nodes. This is because a single link break or node failure could destroy multiple routes. To overcome this problem, which could occur frequently in *MANETs*, *AOMDV* considers only *link disjoint* and *node disjoint paths*. Multiple *paths* are *link disjoint* if they do not share common links between nodes on the way from the source to the destination. This does not exclude the use of common nodes on the *paths* as illustrated in figure 3.18a. When node B fails in this example, both *paths* will get inoperative. To make the routes more stable, *node disjoint paths* should be used as shown in figure 3.18b where both *paths* neither share common nodes nor common links. If a node fails in this scenario, the other *path* will be unaffected except for the case that either the source node A or the destination node C fails.

This implies that *node disjoint paths* are more robust than *link disjoint paths*. However, since *AOMDV* is based on using multiple *paths* per destination and a network offers more *link disjoint paths* than *node disjoint paths* Marina et al. [MD06] suggested using *link disjoint paths*. Each node is responsible for checking that its next hops and previous hops for the *paths* to a specific destination differ to ensure link-disjointness. This is done by maintaining the *Last Hop* and the *Next Hop* for each *path* in the node's routing table (cf. table 3.4).

Compared to *AODV*, the processing and forwarding of *RREQs* and *RREPs* was slightly modified to support disjointness. When a node generates an *RREQ* or *RREP*, it has to attach the first hop of the path to the message. This is required for the destination node to identify different *link disjoint paths* and is done by adding an extra *First Hop* field to these messages. The information in the *First Hop* field of a message is stored into the *Last Hop* field of the corresponding *path* in the destination's routing table, since the first hop of an incoming path is the last hop of a path in the destination's routing table.

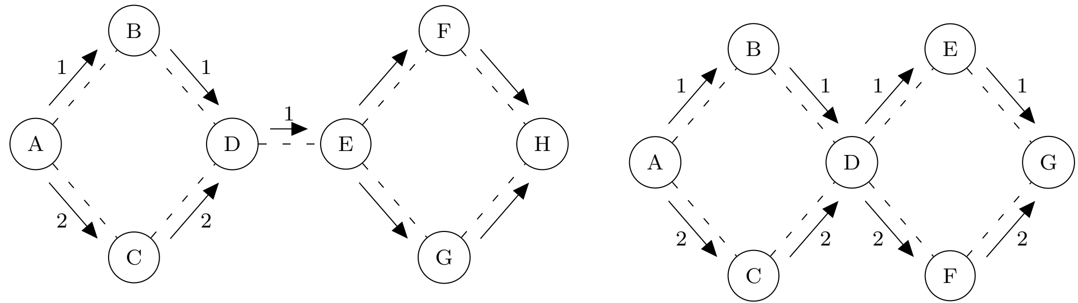


(a) Link-Disjoint Paths

(b) Node-Disjoint Paths

Figure 3.18: Illustration of Link-Disjointness (Path 1 and Path 2) and Node-Disjointness (Path 1 and Path 3)

Figure 3.19a shows the need for this last hop information. Let node A be the source and node H the destination of a message. Two copies of the message will be forwarded to node D via node B and C, respectively. The *First Hop* field of a message contains the first hop the message has passed, which implies *First Hop* = B for message one and *First Hop* = C for message two. Let copy one via B arrive earlier. This provokes node D to drop copy two and only forward copy one to node E since an *RREQ* is only allowed to be forwarded once and multiple copies of an *RREP* are only allowed to be forwarded on disjoint links. Node E will forward a copy to each of its disjoint links towards H which are to node F and G, respectively. Afterwards, node H receives the message twice. The node analyzes the *First Hop* field in the messages and recognizes that both messages came over the same first hop (node B) and that the *paths* are not disjoint. Compared to *AODV* there is a further difference in the protocol behavior. *AODV* discards duplicates of messages that have already been processed. In contrast, *AOMDV* analyzes such messages to learn more *link disjoint paths* and only afterwards the message is dropped.



(a) Sending Two RREQ Messages and Receiving Only One Link-Disjoint Path

(b) Sending Two RREQ Messages and Receiving Two Link-Disjoint Paths

Figure 3.19: Identifying Different Link-Disjoint Paths

The scenario in figure 3.19b shows the same behavior until the messages arrive at node D. Since this node has multiple links for forwarding, it forwards message one to node E and message two to F. When the destination G receives both messages it detects that the two messages have different addresses in their *First Hop* fields and, therefore, they have been received via *link disjoint paths*.

As previously described, each *AOMDV* node has to ensure that no recently processed *RREQ* is forwarded twice. In the scenario illustrated in figure 3.19b, this results in a *route cutoff* which yields the destination node G not to know the *path* via node C. To avoid this problem the destination node G answers to multiple copies of the received *RREQs* with multiple *RREPs* even if they did not arrive via *link disjoint* *paths*. If multiple *RREPs* are generated by the destination (cf. figure 3.19b) the originator node might receive multiple *RREPs* and, therefore, learn multiple routes to the destination. The only constraint is to use different first hops for the reply. To avoid network congestion caused by the generation of too many *RREPs*, the number of them is limited to a value of three by Marina et al. [MD06].

3.7.3 Applications

Since *AOMDV* benefits from scenarios in which multiple routes to a destination are required, it performs best in scenarios with a high number of nodes and frequent link breaks. This implies highly dynamic MANETs with a large number of participants, respectively a high node density. Marina et al. [MD06] stated that in this kind of scenarios *AOMDV* can outperform *AODV* by an end-to-end delay improvement of factor two, routing overhead reduction up to 30 percent and a reduction of the packet-loss of 40 percent.

The main goal of *AOMDV* is the increase of fault tolerance compared to the *AODV* protocol. Furthermore, when changing the *path* selection algorithm on each node, the protocol could be modified to support load balancing and to allow the forwarding of data packets simultaneously on multiple *paths*. This could increase the throughput in the network.

3.8 Latency Avoidance by Route Assumption (LARA)

The most important handicap of reactive routing protocols like *AODV* or *AOMDV* is their latency for route discovery. This dissertation defines the latency as the time between sending the *RREQ* from the source node and receiving the corresponding *RREP* on the same node. Such latency forces nodes to buffer packets and could lead to interrupted multimedia applications like videoconferences. The previous section 3.7 presented that the multipath behavior of *AOMDV* leads to a lower time-average latency for such a route discovery procedure. The current section presents the approach *Latency Avoidance by Route Assumption (LARA)* [FKS+13] which was developed in the scope of this work to lower the time-average latency.

3.8.1 Motivation

Data packets in ad hoc networks should quickly find their way to their destination, especially in real-time applications like internet telephony or video streaming. If nodes have to buffer packets during their route discovery procedure, the human participants of the application could notice stuttering. This is a big problem of reactive routing protocols since they use a route discovery procedure on-demand which means that the data packets are still to be forwarded but no route is known.

The typically very limited resources in terms of energy or *bandwidth* are a further problem of ad hoc networks. Therefore, routing protocols should try to generate as little protocol overhead as possible and to use only information that is already available.

3.8.2 Related Work

Much activity can be noted in this field of research as the reduction of latency in reactive routing networks is a relevant topic.

Ronghua Shi et al. [SD08] proposed to attach a route's estimated latency to the *RREQ* to offer the information to the network. Each node processing the message receives the value, estimates its own latency and forwards this to the next node using the *RREQ*. When the packet arrives at the destination, the node applies the estimated latency to the *RREP* and sends it back to the originator of the *RREQ*, which in turn extracts the latency of the discovered route. Afterwards, if multiple routes are known, the node is able to pick the route with the lowest latency to forward packets. However, this approach is only able to lower the latency of previously discovered routes. Relating to real-time applications as mentioned before this would lead to stuttering each time a new route, which has to be discovered first, is used.

Another approach was developed by Gwalani et al. [GBRP03]. They accumulate the complete forwarding path of an *RREQ* in the packet. Thus, each intermediate node which processes the *RREQ* learns all routes to the previous nodes. In this way the nodes in the network can learn more routes which results in fewer required route discoveries and, therefore, in a lowered average latency. Since the *RREQ* packets could largely grow in size when the whole path information is piggy-backed and ad hoc networks suffer from low *bandwidth*, this approach is not favored.

This work was improved by Hu et al. [HLS10] to lower the produced routing overhead. This was done by modifying the protocol in a way that only the first node which forwards an *RREQ* from the originator adds its address to the message. Other nodes that receive the *RREQ* later learn a route to both the originator of the *RREQ* and its neighbor node which acted as first hop. This leads to a decreased average latency since the nodes can learn two routes per *RREQ* instead of one as in *AODV*. Compared to the scheme of Gwalani et al., this work has lower routing overhead, with the handicap, however, that it also has fewer routes and, therefore, a larger latency.

In another mechanism provided by Tuan et al. [TL09], a node learns the routes of its new neighbor nodes. With this behavior, a node gets more and also better routing table entries resulting in a lower number of generated *RREQs*. To exchange the additional information between the neighbors, the protocol exchanges supplemental route information resulting in an increasing routing overhead. Furthermore, since the nodes exchange routes without an on-demand need, the approach tends towards a proactive solution.

All of the previously mentioned schemes reduce the latency by increasing the produced routing overhead. The approach presented in this work is able to lower the time-average latency without generating additional routing packets.

3.8.3 Architecture

Comparable to the related work, *LARA* also tries to increase the knowledge of the nodes in the network to lower the time-average latency. The difference is that *LARA* neither modifies existing routing packets nor sends additional routing packets into the network. The reactive routing protocols *AODV* and *AOMDV* learn routes from received *RREQs* and *RREPs*. Thereby, a received *RREP* is used to learn the route to the destination node which is advertised in the message, while received *RREQs* are used to learn a route to the originator node of the request. As previously mentioned, nodes should try to generate as little traffic as possible and to learn as much as possible from information that is already available. Therefore, *LARA* additionally learns the route to the destination of overheard *RREQs*. The gathered information is used to assume routes to the destination nodes of the *RREQs* via the originator nodes that generated the *RREQs*. This is done because the originator of a request will most likely know a route to the destination in the near future since it started a route discovery procedure.

It should be pointed out that reactive routing nodes also learn from received *HELLO* messages or *RERRs*, however, these packet types are not explicitly mentioned in this chapter. Furthermore, it should be kept in mind that route discovery procedures which increase the time-average latency are only relevant for the first use of new routes. If a route already exists in the routing table it can be used immediately without such latency.

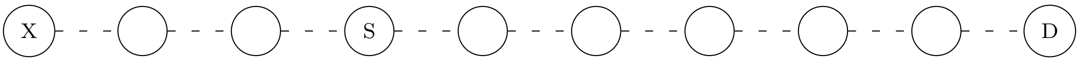


Figure 3.20: LARA Network Scenario 1

Figure 3.20 illustrates the operation of *LARA*. Let node S be the source of an *RREQ*, node D the destination which should be discovered and node X the node which learns the route to D from the overheard *RREQ* and, therefore, benefits from *LARA*. When node S induces an *RREQ*, it travels to the right in the direction of node D and to the left towards node X which analyzes the request and learns the route to the originator node S and a further route – the *LARA* route – to node D. Afterwards, if node X tries to send packets to node D and does not have a normal route, it simply forwards the data packets via the *LARA* route to node S, which in turn forwards them to node D.

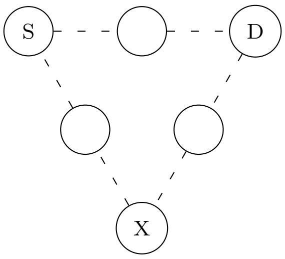


Figure 3.21: LARA Network Scenario 1a

In this process the *LARA* routes could be suboptimal since they may not be the shortest routes to the destination (cf. figure 3.21). To overcome this circumstance, node X will try to find a better route directly after sending the first data packets. Therefore, it generates an *RREQ* destined for node D . As soon as X receives an *RREP* for the destination node D, it will no longer use the *LARA* route because the 'real' route is better than or as good as the *LARA* route. As seen in this example, *LARA* is able to reduce the time-average latency at the beginning of new data transmissions since a node does not have to wait for the *RREP* and can immediately transmit the first data packets.

The functionality of *LARA* can be split into the three main parts *Route Gathering* (cf. subsection 3.8.4), *Route Usage* (cf. subsection 3.8.5) and *Packet Forwarding* (cf. subsection 3.8.6), which will be described in the next subsections.

3.8.4 Route Gathering

Whenever a node overhears an *RREQ*, it analyzes the message expecting to find a new *LARA* route. For the storage of the new routes each node contains a new *LARA Routing Table (LRT)* (cf. table 3.5) and is allowed to store exactly one route per destination in it. Each entry consists of the *Destination Address*, the *Exit Point Address* (cf. subsection 3.8.5), the route's *Lifetime* and the *RREQ Sent Flag*. The flag is used for indicating if the node has already sent an *RREQ* for the specific destination to limit the number of generated *RREQs*.

1st Dest. Address	1st Exit Point Address	1st Lifetime	1st *RREQ* Sent Flag
2nd Dest. Address	2nd Exit Point Address	2nd Lifetime	2nd *RREQ* Sent Flag
...	...	...	...

Table 3.5: The Structure of the LRT

Every time a new *LARA* route is gathered, it is stored into the *LRT*. If an entry for the desired destination already exists, the *Exit Point Address* is updated and the *Lifetime* is refreshed. It is important that an active *RREQ Sent Flag* must not be changed until a corresponding route in the protocol's main routing table is available. If the *Lifetime* of an entry in the *LRT* expires, the *Exit Point Address* is set to zero and the *Lifetime* field remains zero to indicate an invalid route. If such an entry is invalid and the *RREQ Sent Flag* is not set, the route may be deleted.

When using a routing protocol, it is very important to ensure freedom from loops. The operation of *LARA* guarantees loop free routes since it does not modify the route finding algorithm of the underlying *AOMDV* routing protocol and instead only uses *AOMDV* routes for its operation.

3.8.5 Route Usage

By the time a node tries to forward data packets, it first looks up a route in its main routing table. Secondly, if no route is found it checks the *LRT* for a route. If a route is available, the node detours the data packets towards the *Exit Point Address* which is the node (node S) that may forward the packets to the destination. The detour is done by encapsulating the packets into *IP* packets, which are addressed to the exit point. Therefore, an IP-in-IP encapsulation [Per96] is used. Alternatively, the *Source Routing* capability of *Internet Protocol version 4 (IPv4)* or *Internet Protocol version 6 (IPv6)* could be used. However, both protcols offer security breaches when using *Source Routing* as mentioned by Reitzel [Rei07] and Abley et al. [ASNN07]. As the packets from X to S are forwarded by making a next hop decision on each node, the route of the packets could be optimized if an intermediate node has a better route. If such an intermediate node is willing to analyze a received encapsulated packet, it could also detour it towards the destination node D to optimize the packet's route.

Directly after forwarding the encapsulated packets, node X generates its own *RREQ* to find a more optimal route to node D and it sets the *RREQ Sent Flag* in its *LRT*. By definition a node is allowed to use its *LARA* routes for its own purposes only. Using such routes to forward foreign packets is prohibited since such routes are not optimal.

3.8.6 Packet Forwarding

When node S receives an *IP* packet, it checks if it is destined for itself. If so, the node surveys if the *IP* packet contains another *IP* packet by analyzing if the *Protocol* field of the *Outer IP Header* has a value of four. In such a case, node S tries to find a route to the destination in its routing table. If no route is available but node S is waiting for a requested *RREP*, it buffers the packet for later forwarding. If S has no route and does not wait for an *RREP*, the packet is dropped.

Figure 3.22 shows the message sequence chart for an exemplary communication in the scenario illustrated in figure 3.20. Node X benefits from the *LARA* route it gathered from node S. With this route, the node does not have to await the latency until receiving an *RREP*. Instead, it is able to send its first data packets directly at the beginning of a communication.

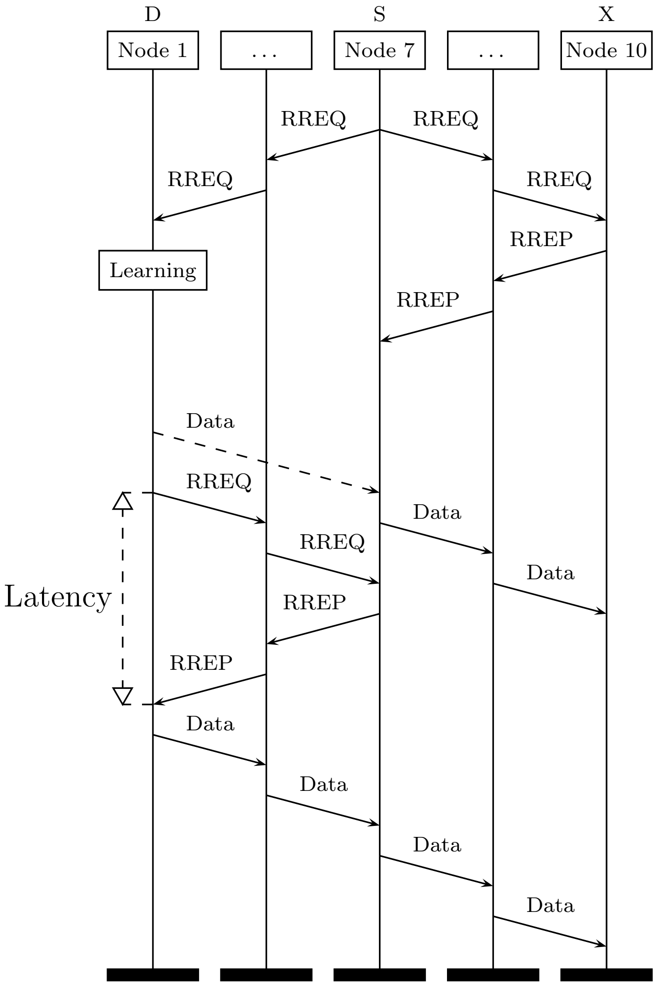


Figure 3.22: The MSC for the LARA Network Scenario 1

3.8.7 Simulation and Validation

The validation of *LARA* was proved by using the *network simulator 2 (ns-2)* [Unie]. The implementation of *LARA* is based on the *AOMDV* implementation of Caleffi [Cal] and, therefore, the resulting protocol representing the functionality of *LARA* is named *Lower Latency Ad hoc On-demand Multipath Distance Vector (LLAOMDV)*. To validate the behavior of *LLAOMDV*, the simulation runs are configured to use ten or fifteen static nodes. The usage of static nodes allows an easier understanding of the protocol's behavior when analyzing the simulation results. However, further simulation runs with mobile nodes should be done when analyzing the protocol's performance. The data packet size is set to 100 bytes and a simulation time of five seconds is used. Since it is impossible to measure the delay between sending an *RREQ* and receiving the related *RREP*, because *LLAOMDV* gathers information from overheard *RREQs* without receiving the corresponding *RREPs*, the simulation measures the time between sending a data packet on the *Application Layer* and receiving the same packet at the destination's *Application Layer*. The validation is done using three different scenarios which are introduced in the following subsections.

3.8.7.1 Simulation Scenario 1

This simulation is based on the network constellation presented in figure 3.20. Node S wants to transmit data packets to node D at the simulation time 2.0 seconds. Since the node does not have a route to the destination it generates an *RREQ* which is forwarded to the right towards node D and to the left towards node X. When X receives the *RREQ* it learns the *LARA* route to node D.

At the simulation time 3.0 seconds node X wants to send five data packets to node D with a sending interval of 20 milliseconds. Figure 3.23 shows the throughput measured at the destination node D. The diagram is based upon the data in table D.1 shown in the appendix in chapter D.

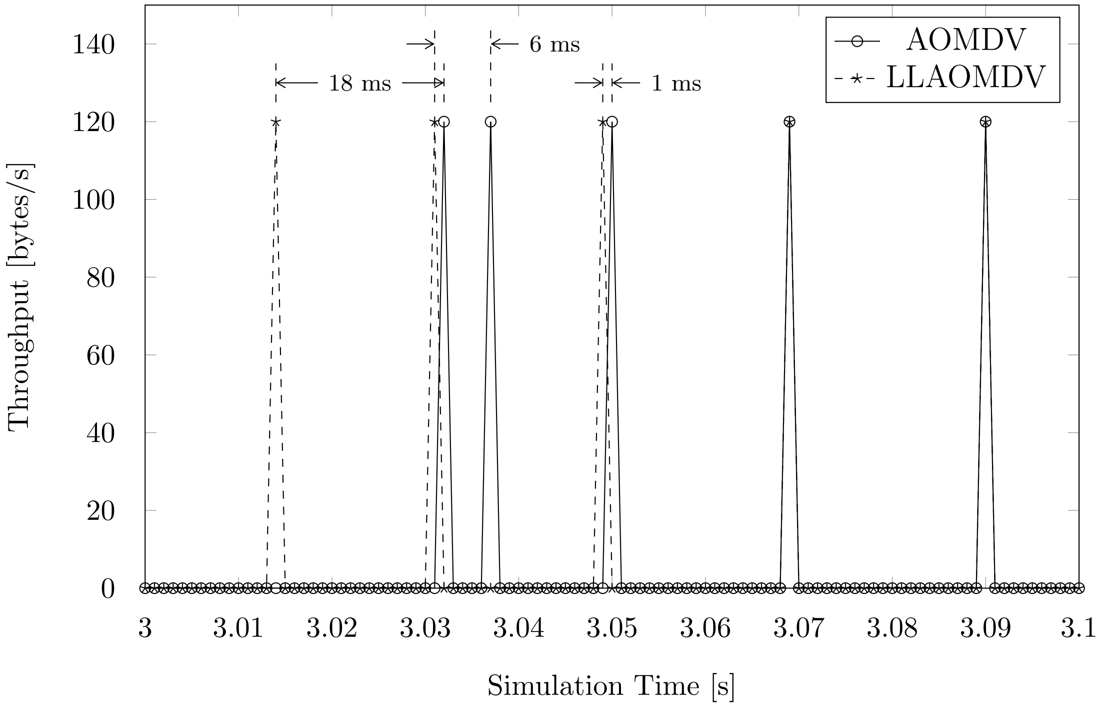


Figure 3.23: The Throughput at Destination Node D in Simulation Scenario 1

When using *AOMDV* the first data packet arrives at the destination after 32 ms. The benefit of *LLAOMDV* can be seen when comparing the arrival time of the first data packet of *AOMDV* with the first packet of *LLAOMDV*. The new protocol utilizing the approach of *LARA* is able to deliver the packet 18 ms earlier than *AOMDV*. This time (latency) can be saved because the route discovery procedure of sending an *RREQ* and receiving the *RREP* is not required. The second data packet in the exemplary scenario arrives six milliseconds earlier and the third packet one millisecond earlier. Thereafter, both protocols receive further packets at the same time. Since *AOMDV* has learned the route, both protocols have valid routing table entries for the destination node D and are able to directly send their packets. This behavior of *LLAOMDV* confirms the prediction that *LARA* only affects the first packets of data transmissions to previously unknown destinations.

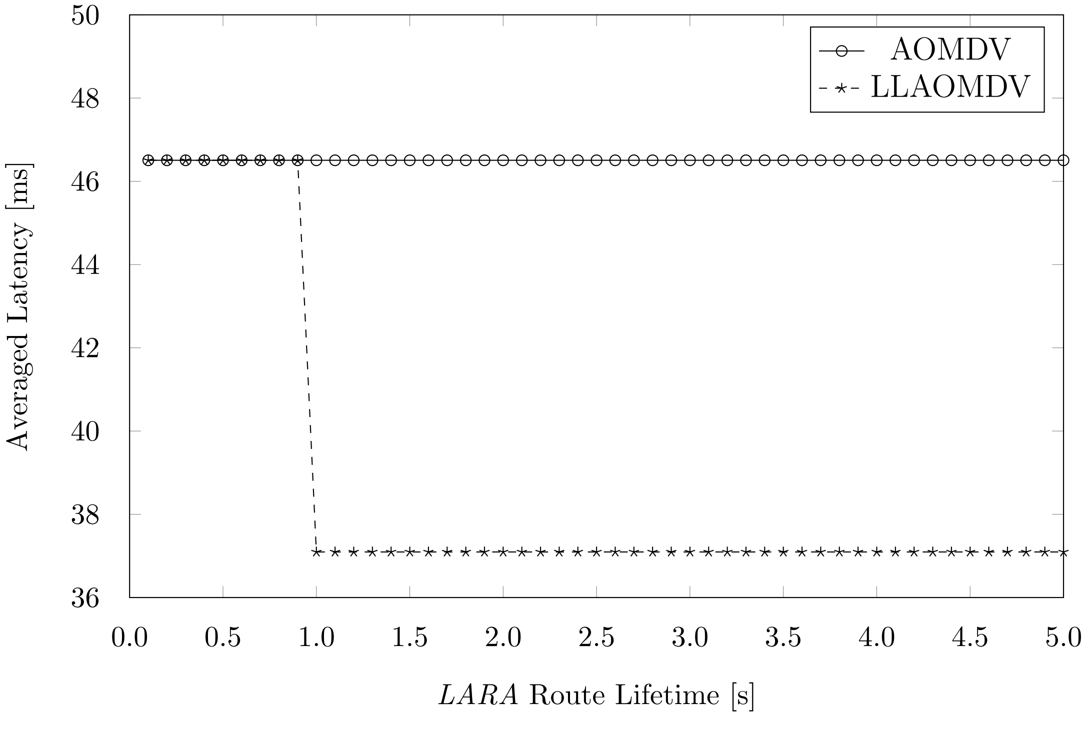


Figure 3.24: Comparison between the Latency of AOMDV and LLAOMDV for Different Route Lifetimes in Simulation Scenario 1

The performance of *LLAOMDV* depends strongly on the route's lifetime. Figure 3.24 (based on the data in table D.2 of the appendix) presents the dependency of the latency regarding the *LARA* route lifetime for the scenario in figure 3.20. Node X sends its data packets at simulation time 3.0 seconds to D and has previously learned the *LARA* route at simulation time 2.0 seconds. Therefore, the route lifetime should be greater or equal to 1.0 seconds to benefit from this route. If the route's lifetime was shorter, node X could not use the route for sending its packets and would perform like *AOMDV*.

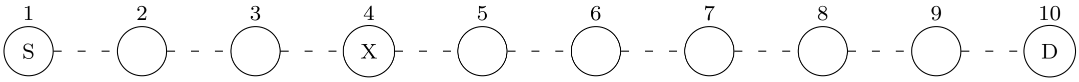


Figure 3.25: LARA Network Scenario 2

3.8.7.2 Simulation Scenario 2

This subsection analyzes a scenario (cf. figure 3.25) which is a challenge for *LARA*. The problem in this scenario is that node X overhears an *RREQ* from node S and forwards its data packets afterwards via node S to node D. This results in a detour of the packets as well as in an elongated time period for the packet delivery. The problem can only occur in the time span – denoted by *Bad Moment* – between overhearing the *RREQ* from node S and receiving the *RREP* from node D. When node X sends the packets after the *Bad Moment* the protocol behaves like *AOMDV* since the normal route has been learned. When the packets are sent during the *Bad Moment*, *LARA* uses the detour and the protocol could perform worse than *AOMDV*. The probability for the *Bad Moment* depends on the number of hops between the nodes X, S and D.

$$num\_hops\_standard = num\_hops^{D}_{X\,rreq} + num\_hops^{X}_{D\,rrep} + num\_hops^{D}_{X\,data} \tag{3.1}$$

$$num\_hops\_lara = num\_hops^{S}_{X\,data} + num\_hops^{D}_{S\,data} \tag{3.2}$$

Equation 3.1 specifies the number of hops it will take to discover the route with a standard reactive routing mechanism and subsequently to send the first data packet to the destination. The notation $num\_hops^{D}_{X\,rreq}$ denotes the number of hops between node X and node D for an *RREQ*. Equation 3.2 describes the number of hops for delivering the first data packet directly using a *LARA* route. If *num_hops_lara* (cf. equation 3.2) is smaller than *num_hops_standard* (cf. equation 3.1), *LARA* outperforms the standard protocol which results in an earlier reception of the first data packets at node D. Figure 3.26 visualizes the number of hops (cf. equation 3.1 and 3.2) a packet takes to be delivered based on the position of node X. Position one represents the position of node S while node D is located at position ten. It can be seen that the number of hops for *LARA* decreases when moving node X towards node S. Since the hop count of the detour increases when moving into the direction of D, the performance of *LARA* gets worse.

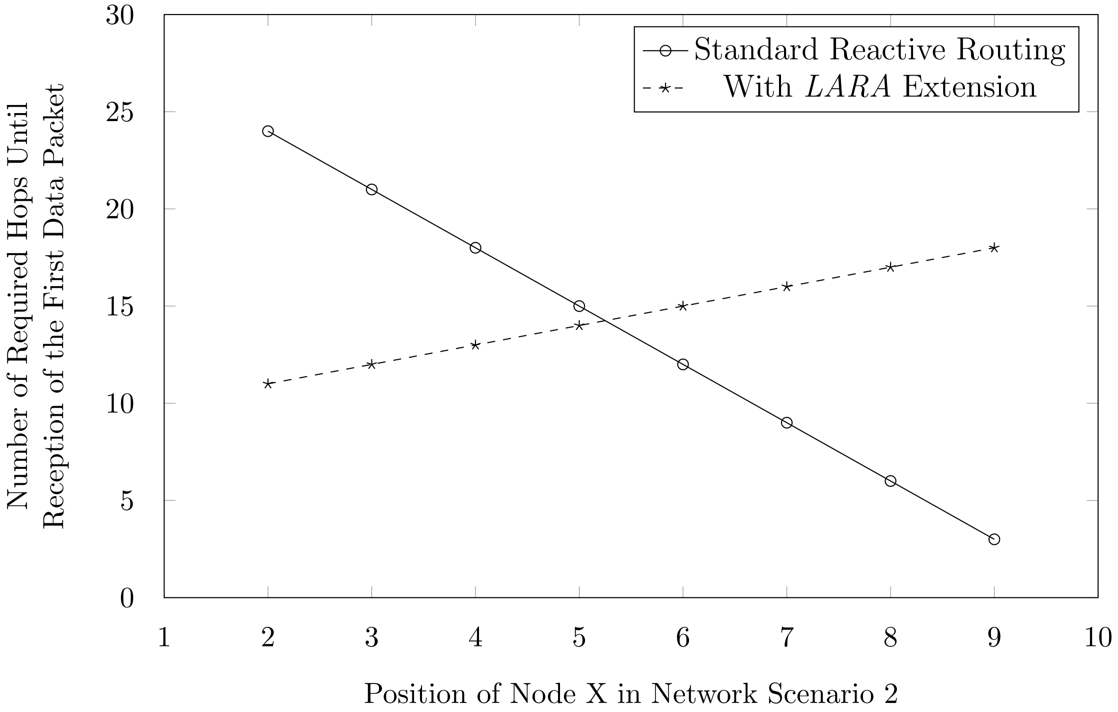


Figure 3.26: Comparison of the Required Number of Hops for Data Packet Delivery between a Standard Reactive Routing Protocol and LARA in Simulation Scenario 2

Assuming a constant processing time on each node, the number of hops is comparable to the time a packet needs to be delivered. This time, in turn, is comparable to the probability of the *Bad Moment* since the probability increases with longer delivery durations. To rate the probability of packet transmissions on node X during the *Bad Moment*, figure 3.27 visualizes the probability for different positions of node X in the network scenario 2.

As the duration of the delivery decreases when moving towards node D, the probability for the *Bad Moment* also decreases. Remember figure 3.26, which shows that node X close to node D provokes a bad performance of *LARA*. In turn, this results in the fact that in situations with a larger *Bad Moment AOMDV* is outperformed by *LARA* since its hop count decreases when moving node X towards node S.

The scenario illustrated in figure 3.25 could be changed for the worse by repositioning node X as illustrated in figure 3.28. In such a scenario node X overhears the *RREQ* from node S, but the *RREP* from node D to S is not received by node X. This leads to a decreased performance since node X cannot learn from the *RREP* and will therefore, use its *LARA* route with a detour via node S. Since node X triggers its own *RREQ* at the first use of its *LARA* route, it will learn the better route for further packets. To solve the handicap for the first data packets, node A could analyze the encapsulated packets and redirect them directly to the destination node D as mentioned in subsection 3.8.5. To inform node X about the better route, node A could additionally generate a *Gratuitous RREP*.

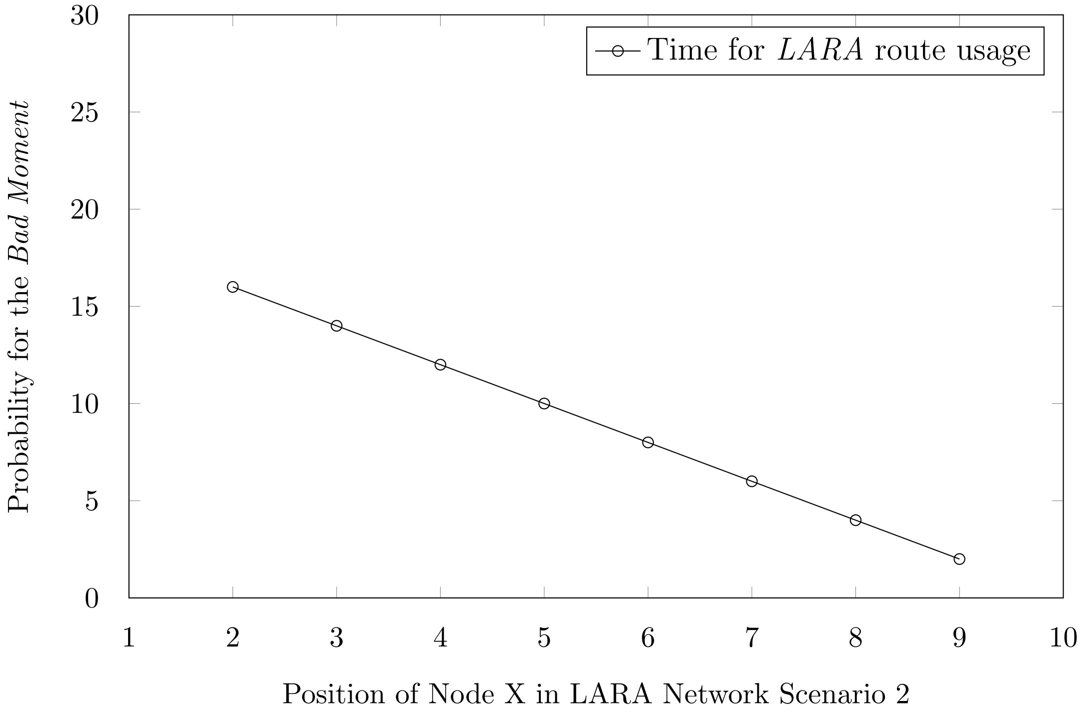


Figure 3.27: The Probability for a Data Packet Delivery during the Bad Moment in Simulation Scenario 2

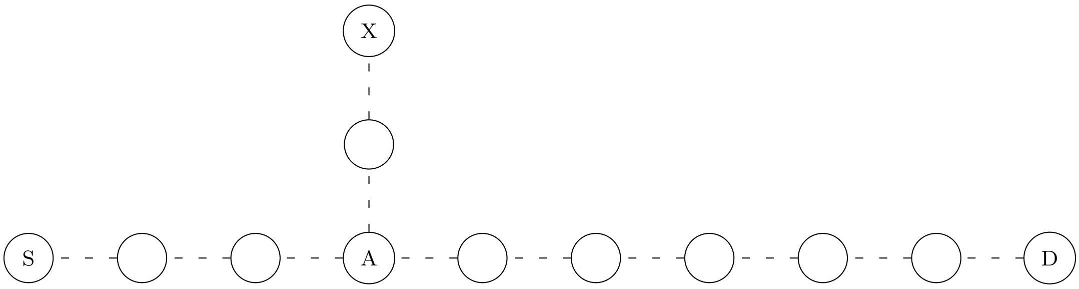


Figure 3.28: LARA Network Scenario 2a

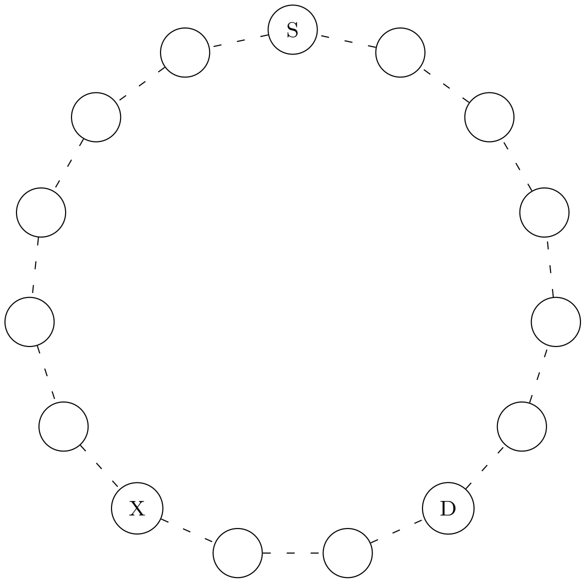


Figure 3.29: LARA Network Scenario 3

3.8.7.3 Simulation Scenario 3

This scenario (cf. figure 3.29) offers multiple paths from the source to the destination. Each *RREQ* sent from node S induces one *RREQ* packet traveling clockwise and another *RREQ* packet traveling counterclockwise. When the clockwise *RREQ* is received by node D, it generates an *RREP* back to S. If the other *RREQ* is forwarded via node X and arrives at node D, a normal routing protocol like *AODV* would drop this packet since it is a duplicate. In this case, node X only receives the *RREQ* from node S and, therefore, it would forward its data packets to node D via node S. However, since *LARA* is based on a multipath routing protocol (*AOMDV*) node D generates a further *RREP* which is send to node S via the path including node X. In this way, node X receives the *RREP* from node D and learns the shorter route. The problem with the *Bad Moment* behaves as described in the previous subsection 3.8.7.2.

3.8.8 Summary and Future Work

It was shown that the presented protocol extension *LARA* allows improving the handicap of reactive routing protocols, which is the latency for the route discovery procedure at the beginning of new transmissions. The protocol was validated using the *ns-2* with static network scenarios. To proof the functionality in mobile scenarios, further simulation runs should be done. Since the simulation environment was changed from *ns-2* to *network simulator 3 (ns-3)* during the work on *SEREMA* to allow further protocol tests on real hardware using *network simulator 3 with Click Modular Router (ns-3-click)*, the *LLAOMDV* implementation has to be adapted to *ns-3*. Afterwards, further simulation runs should be done to validate the behavior of *LARA* in mobile scenarios.

Another interesting application domain could be sensor networks, where a lot of nodes send their sensor data to a few data aggregation points. Since *LARA* is dependent on previously overheard routes, it could benefit from those networks where a lot of nodes communicate with the same destinations and therefore, use the same routes. A further advantage of an *ns-3-click* implementation would be the possibility to integrate the protocol into *SEREMA*. It should be mentioned that the concept of *LARA* cannot only be added to *AOMDV* but also to other reactive multipath routing protocols.

3.9 Conclusion

	OLSR	AODV	AOMDV	LARA
Protocol Type	proactive	reactive	reactive	reactive
High Traffic	+	−	−	−
Low Traffic	−	+	+	+
Highly Mobile	−	+	+	+
Latency	+	− −	−	o
Multipath	−	−	+	+

Table 3.6: Comparison between Routing Protocols for Ad Hoc Networks

This chapter presented the basic principles of routing in ad hoc networks and introduced some of the most widely used protocols. The details of these protocols were discussed and the preferred application areas were figured out. Table 3.6 gives an overview of the strengths and weaknesses of the protocols. Thereby, it can be seen that a single routing protocol outperforms other protocols in specific network scenarios. However, in other constellations, the protocol could get into trouble and would perform worse, since it was not designed for those cases. This becomes clear when comparing the application areas of *OLSR* (cf. subsection 3.5.5) and *AODV* (cf. subsection 3.6.4), for example. While *OLSR* performs well in scenarios with high traffic and a high node density, it performs worse when the nodes get highly mobile. On the other side, *AODV* has fewer problems with highly mobile nodes, but gets in trouble when the traffic and the node density increase.

Each routing protocol has its strengths and weaknesses depending on the network scenario. To achieve a good performance in as many network scenarios as possible, hybrid and adaptive routing approaches should be considered as discussed in the following chapter 4.

4 Adaptive Routing

Chapter 3 illustrated that a single routing protocol cannot perform well in any possible scenario since it was developed for a special use case. Heterogeneous networks with a lot of different nodes, changing node speeds, or limited energy resources as occurring in disaster scenarios complicate a routing protocol's operation. This resulted in the development of many routing protocols in the past; each of them able to outperform the others in a specific case. A network could benefit from the advantages of multiple protocols depending on the network context if it was able to change the routing protocol during the runtime of the network.

4.1 Introduction

Hybrid and adaptive routing approaches combine multiple routing protocols to benefit from their advantages in different network contexts. This enables them to perform well in various cases. Hybrid routing protocols which combine characteristics of multiple routing protocols into one single protocol are the first step towards such adaptivity. They usually combine the advantages of *Link-State Routing* and *Distance-Vector Routing*. A representative of hybrid routing is the *Zone Routing Protocol (ZRP)* [Haa97, N. 99]. Its name is derived from the fact that it creates a routing zone around each participating node. The size of each zone is defined as a number of hops. For example, if a node uses a zone radius of 3, all nodes that are reachable via 3 hops are part of the node's routing zone. Inside this zone the node routes proactively and, therefore, knows the complete topology of this part of the network. For nodes that are farther away, reactive routing is used. To optimize the propagation of reactive requests, the messages are forwarded to the peripheral nodes of the routing zone. Those nodes check if the destination is located in their proactive routing zone. Otherwise, they forward the messages to their peripheral nodes. This is repeated until a node finds the destination in its routing zone or the request reaches its maximum allowed hop count. Furthermore, the *ZRP* allows a modification of the zone radii to adjust the protocol's behavior between pure proactive routing (zone radius = infinite) and pure reactive routing (zone radius = zero).

As seen, the *ZRP* allows a more flexible routing scheme compared to single routing protocols. However, hybrid routing is not able to completely exchange the routing mechanism. The routing algorithm in the *ZRP* for example can only adapt between reactive and proactive routing but cannot activate or deactive functionalities for energy saving, changing from reactive routing method A to method B or from flat routing to hierarchical routing. To allow such a flexibility, adaptive routing must be used.

Adaptive routing schemes allow the complete exchange of the used routing protocol during the runtime of the network. This enables communication devices in large-scale disaster scenarios to adapt their routing protocol to the needs of special areas. If the energy in a given area of the network is strictly limited, the nodes in this part could change to an energy-aware routing protocol. In the case that the node density in an area increases, the devices operating in this part of the network could change to proactive routing. Those examples show that rescue teams would benefit from the fexibility of adaptive routing schemes.

However, beside all the benefits of adaptive routing mechanisms, this type of protocols is faced with some challenges. Existing routes must not be interrupted during the change of the protocol and, for the case that multiple different routing protocols are simultaneously active in a network, the adaptive approach has to deal with the translation of routing information between different routing areas. This increases the complexity of adaptive routing schemes heavily compared to standard routing protocols. The subsequent section presents different approaches of the group of adaptive routing schemes.

4.2 Related Work

Over the past years a lot of effort was spent to develop adaptive routing approaches. This section will give an overview of those routing methods and highlight their advantages and possible drawbacks.

The approach presented by Jiang et al. [Jia05, NJK07, NJK09] combines multiple routing protocols by adding a *Protocol Swap Layer* between *UDP* and the routing protocols. This layer adds some information about the active routing protocol and the freshness of the data to outgoing routing packets and removes the information from received packets. Therefore, the extension is transparent to existing routing protocols which are not affected by the modification. A master node decides when to change the protocol and which protocol should be used. This information is attached to the master's routing packets and delivered to other nodes in the network. When a node receives the information, it copies the routing table entries from its current routing table to the table of the new protocol and changes its protocol afterwards. For the conversion of routing table entries from the current protocol to the new one, a node requires $n * (n - 1)$ routing table converters, where n represents the number of implemented routing protocols. This increases the implementation complexity heavily. As this routing scheme does not generate additional routing packets, the routing overhead does only increase minimally due to the eight bytes that are added to outgoing routing packets. Since nodes operating in disaster scenarios could fail anytime no centralized services like the master node of this approach should be used. If the master node fails the routing gets into trouble because the component making the protocol decision is not available anymore. Furthermore, as all nodes have to use the same routing protocol at a time, the framework cannot provide different active routing protocols in different areas of a wide-area network.

The strategy of Hoebeke [HMDD04, Hoe07, HMD12] allows the cooperation of different routing methods in the network at the same time. This enables each node to choose the optimal routing out of a set of available routing modes for its current requirements. In this way, node A could route in a reactive way while node B could use proactive routing. To allow this possibility each node has to support a basic routing mode which acts as pure reactive routing. This mode can be used as a fallback when no other compatible routing modes are available. The other routing modes extend the possibilities of the adaptive routing and nodes that only contain a subset of the available modes are also supported. This enables a node to be upgraded anytime with further routing modes to enhance its possibilities. On the other side, a node could deactivate specific routing modes if its remaining energy gets low for example. In order to allow all the features of this routing approach the used protocols have to be modified before they can be used to support a generic routing table. This heavily increases the implementation effort. In contrast to the previously mentioned approach, this work allows each node to make its own protocol decision. Therefore, all nodes have to monitor the network to be able to make the protocol decision. This is done by local measurements and by exchanging statistical information with other nodes via special broadcasted packets. The exchanged information increases the quality of a node's decision, but generates additional routing overhead.

A further adaptive routing framework was proposed by Ramrekha et al. [RP09, RP10, PRMP10, RPP]. Their approach uses a proactive routing scheme for networks with less than or equal to ten nodes while networks with more than ten nodes are routed reactively. The ten nodes threshold was estimated using different simulation runs and is the only criterion for the protocol decision. Every participating node monitors the network environment to get the number of nodes in the network. In the proactive routing mode which uses *OLSR*, this is done by analyzing *TC* messages. When using the reactive routing which is based on *AODV*, the hop counts in received *RREP* messages are analyzed. The mechanism expects that the edge length of the network can be estimated based upon such a hop count. If the nodes in the network are uniformly distributed and the edge length of the network is estimated as x hops, the number of nodes n can be derived from the equation $n = x^2$. To assure that the number of nodes in the network is below the *Network Size Threshold (NST)*, a reactive routing node induces a *Hop Count Request (HCREQ)* with a *TTL* of$\sqrt{NST}$ hops. This request is forwarded in the network until its *TTL* reaches a value of zero. In this case the processing node responds with a *Hop Count Reply (HCREP)* to tell the originator that the network's minimally required edge length has been reached. When a node decides to change the protocol, it generates a *Change Phase (CP)* message which is processed and forwarded by all other nodes and provokes all nodes to change their routing mode. To avoid oscillation between the reactive and the proactive modes caused by a fluctuation in the number of nodes, the approach runs through an *Oscillation Phase* when changing the protocol. In this phase, a node checks if its oscillation phase validity time, which is set to *Osc-Interval* on each protocol change, has expired. A node is only allowed to change its protocol when this time is over. Since a node's protocol decision is communicated to all other nodes, the network can only use either reactive or proactive routing at a time.

Forde et al. [For05, FDO06] introduced an adaptive routing framework whose decision making is based on social science theory. Early adopter nodes make a decision about their preferred routing protocol. This decision is communicated to other nodes, which may agree or disagree with the protocol change. The scheme supports multiple different routing areas in the network at the same time, which can use one of the protocols *DSR*, *AODV* or *OLSR*. Therefore, an additional routing table was added to store routes to other routing domains. The communication between different routing domains is enabled by extensions for the used routing protocols. For example, the *OLSR* protocol was modified to support route requests for nodes in other domains and the *AODV* protocol was extended for route replies coming from other routing subnets.

Another adaptive routing mechanism was developed by Yazir et al. [YFG$^+$10]. This approach considers small-scale networks. A node that wants to change the protocol can act as a coordinator and send a cooperation request to the other nodes in the network. This request asks the nodes for their opinion to a protocol change. After receiving the replies, the coordinator can select the best protocol based on the determined opinions and subsequently send its decision command into the entire network. Each node receiving this command has to change its protocol according to the decision of the coordinator. The node acting as coordinator is a centralized node during each single election and may change for each election process. This makes the approach robust against node failures. A handicap of this scheme, however, is the network wide adaption of the routing protocol, which does not support different routing areas.

The routing mechanism proposed by Fujiwara et al. [FOK12] does not aim at the connection of single nodes with different routing protocols but rather at the interconnection of different routing subnets. For this purpose, each routing subnet provides *Ad hoc Traversal Routing (ATR)* nodes on its borders. These nodes translate received routing packets into *ATR* packets which are forwarded to *ATR* nodes of other subnets and there, are again translated into the destination routing protocol. This implies the conversion of a routing message twice to forward it from routing subnet A to subnet B. Furthermore, each *ATR* node collects information about participating nodes in its routing domains and exchanges this information with other *ATR* nodes. The participants of reactive routing networks are identified by the use of novel *ATR* beacons which induce extra routing overhead. As this approach was designed for the interconnection of different routing subnets instead of being an adaptive routing approach, it does not provide monitoring and decision making to select the best protocol. However, this approach was included into this paragraph because it provides a method for the interconnection of different routing subnets which could be reused for adaptive routing.

4.3 Requirements

This section defines the requirements for adaptive routing approaches developed in the scope of this dissertation. The requirements will be used in section 4.4 to check the suitability of the routing schemes previously presented in section 4.2.

Adaptive Mechanism To guarantee the best performance in different network scenarios, this work deals with adaptive routing in *MANETs*. Therefore, it has to be ensured that the considered routing schemes provide an adaptive mechanism to enable the change of the routing protocol in parts of the network or in the whole network during the runtime.

Simultaneous Protocols If considering wide-area *MANETs* for the communication in disaster scenarios, different types of areas can occur. One area could have a lot of slowly moving nodes while another area could have only a few nodes that are highly mobile. As a specific protocol cannot perform well in all scenarios, it would be beneficial if the adaptive routing scheme was able to use multiple simultaneous routing protocols in different areas of the network, by giving the nodes the possibility to individually select the best routing protocol, out of a given set of protocols. This would enable single nodes with specific requirements, caused for example by little remaining energy, to adjust to the given circumstances and to achieve the best performance.

Routing Tables When using multiple routing protocols at the same time, the routes of different protocols have to be stored. This could be done in one routing table or in multiple routing tables where each of n protocols gets its own table. As adding further protocols is simplified if each routing protocol can use its own unmodified routing table this approach should be preferred.

Packet Types Most approaches require the modification or addition of new routing packet types. This increases the implementation effort, could lower the compatibility to other nodes only operating a standard routing protocol and might increase the routing overhead and should, therefore, be avoided.

Packet Conversions If a protocol supports multiple simultaneous routing protocols, a mechanism for the communication between these protocols is required. At each transition from one protocol to another routing packets have to be converted. A smaller number of required packet conversions produces less processing overhead which saves energy and should, therefore, be preferred.

Efficiency Usually, a common problem of all *MANETs* is the limited available throughput that should not be completely exploited by the routing. The approach should use those sparse resources economically and limit the amount of generated packets for monitoring, decision making as well as routing to an absolute minimum.

Robustness To provide a platform for reliable communication, the network must be robust against single node failures. This implies the following points for the adaptive routing protocols:

- *Monitoring:* In contrast to standard routing protocols, adaptive approaches depend on constant monitoring to receive information about the current state of the network. This operation must work fully decentralized to avoid problems caused by single node failures.

- *Decision Making:* When enough information about the network has been gathered, the nodes can make their routing decisions. If this task fails, the adaptive routing can only behave like a standard routing protocol as it is not able to change its active protocol anymore. Hence, decision making has to be completely decentralized or otherwise the network must be able to elect a new centralized node if the previous one fails.

Decision Quality The performance of adaptive routing is strongly dependent on the quality of the decision making. As more information a node uses for its decision as better the decision will be. Therefore, a node should monitor as many parameters as possible to be able to select the best protocol for its requirements.

Compatibility Not all nodes operating in *MANETs* can be expected to be equipped with the adaptive routing approach. Therefore, the adaptive routing mechanism must be backward compatible to single nodes in the network which support only a subset of the implemented routing protocols or in the worst case only a single standard routing protocol.

Extension Effort The types of scenarios in *MANETs* can vary in a wide range. To ensure the usability and long-term relevance of the adaptive routing framework, well-suited and up-to-date routing protocols should be used. Therefore, the quick extension of the approach by further protocols should be possible without much effort. Therefore, the following requirements have to be fulfilled:

- No modifications of the routing tables of the implemented protocols
- No modifications of the routing packet structures

If a protocol, however, cannot be used inside the adaptive framework without any protocol modifications, those modifications should be kept small and the compatibility to the standard protocol should be ensured.

4.4 Comparison

	Jiang	*Hoebeke*	*Ramrekha*	*Forde*	*Yazir*	*Fujiwara*
Adaptive Mechanism	+	+	+	+	+	−
Simultaneous Protocols	−	+	−	+	−	+
Routing Tables	n	1	1	n+1	1	1
Packet Types	+	−	−	○	−	−
Packet Conversions	n/a	1	n/a	1	n/a	2
Efficiency	○	−	−	○	−	−
Robustness	−	+	+	+	+	+
Decision Quality	n/a	+	−	+	○	n/a
Compatibility	−	○	−	○	−	−
Extension Effort	−	−	+	−	+	+

Table 4.1: Comparison between Adaptive Routing Schemes

The adaptive approaches presented in section 4.2 generally work well in *MANETs*. However, when comparing them with the defined requirements, some weaknesses can be observed. This section discusses the adequacy of the related work schemes to the requirements made in the previous section 4.3 and presents them in table 4.1.

As mentioned in the previous section, the scheme of Fujiwara et al. [FOK12] was only designed for connecting different routing subnets instead of being an adaptive routing approach. Therefore, it is not suitable for the requirements of this work and marked by (−). The approaches which fulfill this requirement are marked by (+).

It is beneficial if the adaptive routing in *MANETs* supports multiple routing protocols for simultaneous use in different areas of the network. This is not supported by the approaches of Jiang et al. [Jia05, NJK07, NJK09], Ramrekha et al. [RP09, PRMP10, RP10, RPP] and Yazir et al. [YFG$^+$10] as illustrated in table 4.1, because these approaches are limited to changing the routing protocol network-wide. This makes them unfeasible for the requirements made in the scope of this dissertation and, therefore, they are marked by (–).

The next row in table 4.1 shows the comparison between the numbers of routing tables a single node uses simultaneously. As the mechanisms of Ramrekha et al. [RP09, PRMP10, RP10, RPP], Yazir et al. [YFG$^+$10], and Fujiwara et al. [FOK12] do not support simultaneous routing protocols in the same area, they only use the single routing table of their currently active protocol. Hoebeke [HMDD04, Hoe07, HMD12] uses a common routing table for all active routing protocols while in the schemes of Jiang et al. [Jia05, NJK07, NJK09] and Forde [For05, FDO06] each protocol has its own routing table. This minimizes the implementation effort, enables the simple extension of the approach by further protocols and is the preferred solution in the scope of this work. It could be argued that multiple routing tables that are stored in parallel consume much memory, especially when using mobile devices in disaster scenarios. However, the structure of an *AODV* routing table entry is shown in table 4.2 and its size is approximately[1] 160 bits, 20 bytes respectively. This means that storing the routes to 100 destination nodes consumes $100 * 20$ bytes $= 2000$ bytes. If this amount of memory is compared with nowadays mobile devices like the *Apple iPhone 5s* with 1 GB or the *Samsung Galaxy S4* with 2 GB of *Random Access Memory (RAM)*, it is obvious that a routing table with around 2 kB of *RAM* usage does not consume too much memory. Table 4.1 states for the approach presented by Forde [For05, FDO06] an amount of $n + 1$ simultaneous routing tables as this approach uses a further table for routes to other routing domains.

Description	Size
Destination IP Address	32 Bits
Destination Sequence Number	32 Bits
Hop Count	32 Bits
Next Hop Address	32 Bits
Lifetime	32 Bits

Table 4.2: The Structure of an AODV Routing Table Entry

[1] Small parts of information like flags for valid routes and destination sequence numbers are not taken into account.

The packet types an approach uses is a further criterion. While the work of Jiang et al. [Jia05, NJK07, NJK09] uses unmodified routing packets with a transparent header in front, the approaches of Hoebeke [HMDD04, Hoe07, HMD12], Ramrekha et al. [RP09, PRMP10, RP10, RPP], Yazir et al. [YFG$^+$10], and Fujiwara et al. [FOK12] introduce new packet types. Both solutions increase the routing overhead and the compatibility with nodes not supporting these new packet types or header extensions is not guaranteed. This could lead to problems when a node that does not support the new routing scheme is required to forward these packets. The scheme of Forde [For05, FDO06] adds extensions to existing routing packets. If the standards of the used routing protocols provide message extensions, the backward compatibility to nodes not supporting the new adaptive approach can be guaranteed. All protocols presented in section 3.4 provide the possibility for message extensions, which allows forwarding new message extensions via nodes that only have implemented the standard protocol. If no changes on the protocols are required, the field is marked with (+). Small extensions provided by the design of the protocol specification are marked with (○) while the addition of completely new packet types is marked by (–).

Another point regarding the routing packets is the conversion of packets from one routing protocol to another at the border of a specific routing domain. The strategies of Hoebeke [HMDD04, Hoe07, HMD12] and Forde [For05, FDO06] handle this with only one packet conversion and thus consume little processing power. The mechanism presented by Fujiware et al. [FOK12] converts the packets via a third protocol and, therefore, needs two conversions. As the other approaches mentioned in table 4.1 neither provide simultaneous protocols nor convert packets, the number of required packet conversions is not available (n/a).

To get out the best performance of *MANETs* the adaptive routing mechanisms should deal economically with the sparse resources of these networks. The scheme presented by Fujiwara et al. [FOK12] introduces new beacons to discover all the nodes in reactive routing subnets. This induces further routing overhead which should be avoided. The approach of Jiang et al. [Jia05, NJK07, NJK09] adds an additional header in front of outgoing packets which induces little additional routing overhead. Hoebeke [HMDD04, Hoe07, HMD12] exchanges statistical information about the state of the network between the nodes. This requires additional packets that are transmitted and increases the routing overhead. The mechanism used by Ramrekha et al. [RP09, PRMP10, RP10, RPP] requires packets for measuring the size of the network and to disseminate the routing decision afterwards. This process decreases the efficiency. The protocols implemented by Forde [For05, FDO06] require some new message types for the communication between different routing areas that induces little routing overhead. For the network-wide election process used by Yazir et al. [YFG+10], additional routing packets are required. If an approach does not require additional packets to be exchanged it is marked by (+) while a (∘) signalizes that a few packets are required and (–) represents a lot of additional packets being exchanged.

To ensure a reliable communication in disaster scenarios, the adaptive routing mechanism should be robust against node failures. Therefore, a decentralized monitoring and decision making is essential. The approaches presented in table 4.1 are simply categorized by their basic functionality. If an approach operates decentralized, it is defined to be robust and suitable for disaster scenarios (+), otherwise it is not (–).

An important part of adaptive routing is the decision making about when to change the protocol and which protocol to use. The approaches presented by Hoebeke [HMDD04, Hoe07, HMD12] and Forde et al. [For05, FDO06] can make the best routing decisions because they use local as well as distributed statistical information. The decision of Yazir et al. [YFG+10] is based on an election made by multiple nodes in the network. However, as this approach cannot account for the requirements of all nodes in the network, it cannot offer the best routing protocol for every node. The scheme presented by Ramrekha et al. [RP09, PRMP10, RP10, RPP] only incorporates the number of partaking nodes in the network into the protocol decision. This mechanism cannot perform well in all scenarios because the best protocol for a given scenario depends on multiple parameters. The work presented by Jiang et al. [Jia05, NJK07, NJK09] does not describe the algorithm used for decision making and therefore the decision quality is marked as not available. As Fujiwara [FOK12] does not provide adaptive routing, no decision making is implemented. The quality of the decision making is rated based on the amount of information used for the decision and marked by (–) if the mechanism only uses a single parameter, by (∘) if it uses multiple parameters, but cannot consider all nodes in the network, or by (+) if it uses a lot of information.

In disaster scenarios it cannot be ensured that all nodes provide the new adaptive routing scheme. Therefore, the approach should be backward compatible to nodes only equipped with a standard routing protocol. As Jiang et al. [Jia05, NJK07, NJK09] adds an additional header in front of the routing packet header, standard routing nodes cannot understand the packets. Furthermore, as the approaches of Jiang, Ramrekha et al. [RP09, PRMP10, RP10, RPP], and Yazir et al. [YFG+10] globally change the routing protocol, a standard node would be separated from the other nodes if it could not follow the protocol change. The mechanism used by Fujiwara [FOK12] cannot provide the connection of standard routing nodes to other routing domains because all inter-domain communication is based upon *ATR*. Therefore, these approaches are marked by (–). The schemes presented by Hoebeke [HMDD04, Hoe07, HMD12] and Forde et al. [For05, FDO06] provide the compatibility to standard routing nodes to some degree (○).

An important requirement to the routing is to be flexible to new challenges. This implies the requirement for exchanging the used routing protocols without much effort and the quick and easy implementation of further protocols into the adaptive framework. This can only be guaranteed if each routing protocol can continually use its own routing table and if the used routing packets remain unmodified. The scheme of Jiang et al. [Jia05, NJK07, NJK09] uses routing table converters that have to be implemented for each protocol. As the amount of converters increases with each protocol ($n * (n-1)$), this solution is not flexible enough. The scheme of Hoebeke [HMDD04, Hoe07, HMD12] modifies the protocols to support a general routing table. This induces much implementation effort which should be avoided. Forde et al. [For05, FDO06] extend the routing protocols to support compatible route request methods. This increases the effort that has to be spent for the implementation. The required effort for implementing further protocols to the routing scheme is denoted by (+) if it is easy to add further protocols while (–) shows that much effort would be required.

4.5 Conclusion

This chapter introduced hybrid as well as adaptive routing and presented existing approaches. The requirements (cf. section 4.3) for adaptive routing in the scope of this work were defined. Thereafter, it was shown that each of the presented adaptive routing schemes has its strength and weaknesses regarding the requirements made. There is no adaptive mechanism available that can meet all of the requirements mentioned in section 4.3. For this reason, the present dissertation considers a new adaptive routing scheme which is introduced in the following chapter 5.

It should be considered that for the special case that an ad hoc network consists of different network parts which are isolated from each other because of their limited transmission ranges or obstacles in between for example, special technologies like DTN can be applied to interconnect such areas with each other as discussed in the work of Begerow et al. [BSS+13].

5 SEREMA – Self-Organized Routing in Heterogeneous MANETs

The present dissertation deals with the routing in networks used in disaster scenarios. It was shown that a single routing protocol cannot cope with the highly dynamic behavior of networks in this kind of environment. Therefore, a lot of adaptive routing approaches are available in order to guarantee a smooth operation. The previous chapter 4 compared different adaptive routing schemes that try to cope with this challenge and showed that the requirements cannot be met by any of the approaches. In the following section 5.1 the new adaptive routing framework of *SEREMA* is introduced, section 5.2 compares the new approach to the made requirements while section 5.3 draws a conclusion on this chapter.

5.1 A New Adaptive Routing Approach

This section briefly introduces the new adaptive routing approach of *SEREMA*. The different elements of the approach are discussed in more detail in chapter 6.

The basic structure of *SEREMA* is visualized in figure 5.1. The monitoring is done by the *Monitoring Agent* which is installed on each node. This agent gathers information about a node and its environment by overhearing and analyzing all packets that are received or sent by the node. Since the efficient dealing with the sparse resources of *MANETs* is one requirement for *SEREMA*, the monitoring does not induce extra packets. Rather, it only uses packets that are already available. It was defined that only local monitoring is used to keep the routing overhead low. This, however, bears the risk that the routing decision might not be best because of limited access to information. Both *Monitoring Agent* elements in figure 5.1 belong to the same *Monitoring Agent*

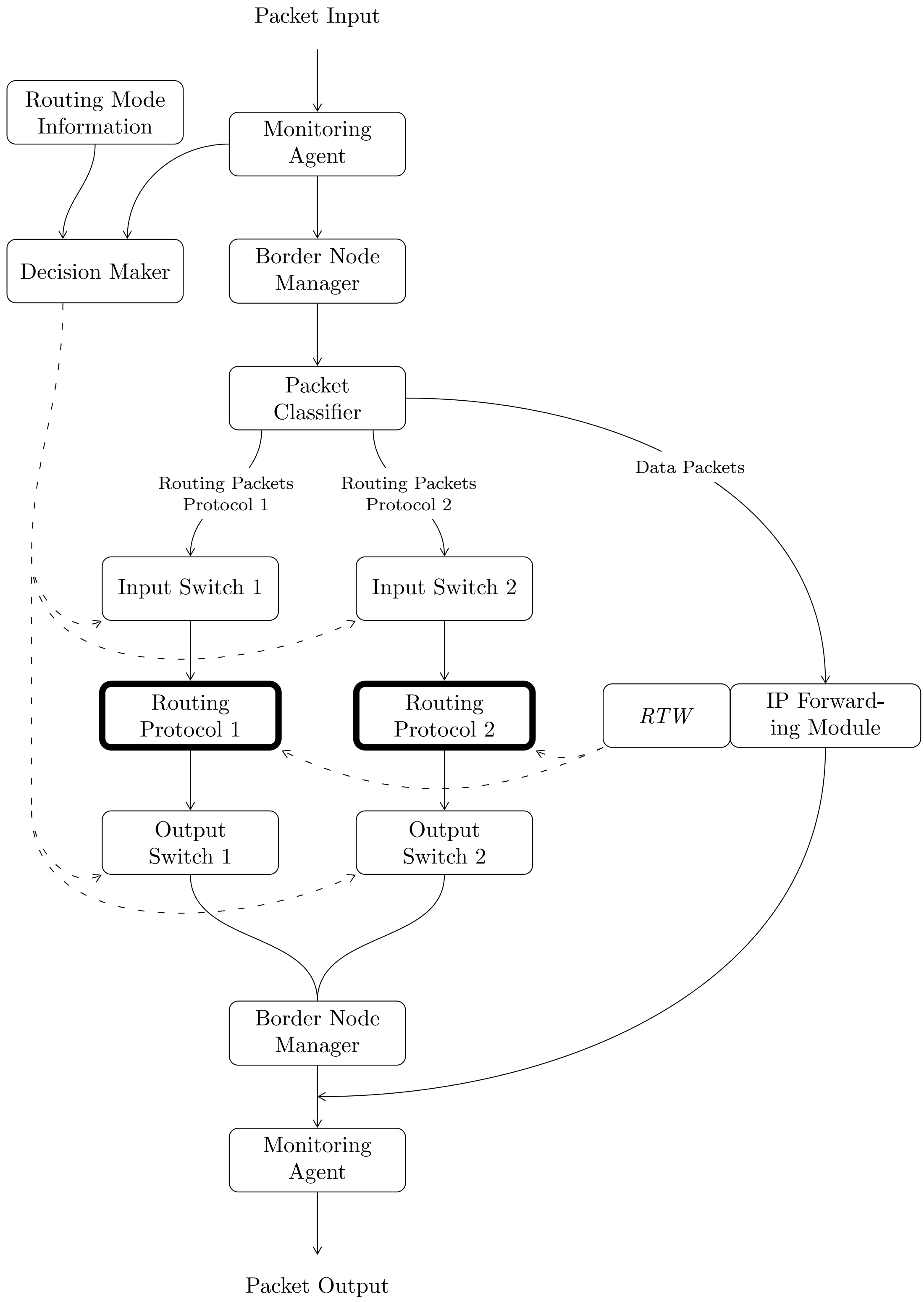


Figure 5.1: The Basic Structure of a Node in SEREMA

module. The difference is that the upper element monitors received packets while the lower element monitors sent packets.

The results of the monitoring are used by the *Decision Maker* to select the routing protocol that best suits the present network scenario. This decision is based on previously investigated knowledge[1] about *MANETs*, provided by the *Routing Mode Information* module. The *Decision Maker* operates completely decentralized and enables each node to select the best protocol for its current requirements. For the purpose of this work the adaptive routing is equipped with two routing protocols, the first protocol with a reactive behavior and the second with a proactive behavior. However, extending the scheme by further protocols is possible.

Received routing packets are classified and forwarded towards the respective routing protocol in figure 5.1. On their way, the packets have to pass switches that are controlled by the *Decision Maker*. It depends on these switches which routing protocols are active and which are not.

The possibility of each node to select its own routing protocol could result in the problem that two neighbor nodes use two different routing protocols. For example, node A could use a proactive protocol, while its neighbor node B uses a reactive protocol. As different routing protocols cannot directly exchange routing information with each other, *Border Nodes* [SKF$^+$15] are used to connect different routing parts. The *Border Node* functionality is controlled by the *Border Node Manager*, illustrated by two elements in figure 5.1. The lower element induces special *Border Node* messages into the network while the upper element receives those messages on other nodes.

Figure 5.2 shows a simple scenario with different routing domains. Let the circular shaped nodes one and two use the same reactive routing protocol, while the octagon shaped nodes three and four use the proactive protocol one and the diamond shaped nodes five and six use the proactive protocol two. A border node is used on the interconnection between different routing domains and is able to operate multiple routing protocols simultaneously to support the translation of routing information from one protocol to another and vice versa. If the reactive routing node one wants to discover a route to node four, it sends the request into the network. When the request is received by the border node (square shaped node), it first checks if it can answer the request with the knowledge of one of its active routing protocols. If it has no knowledge about the destination, it forwards the request into other routing domains, expecting to reach

[1] The determination of knowledge is not part of this work as mentioned in section 1.2.

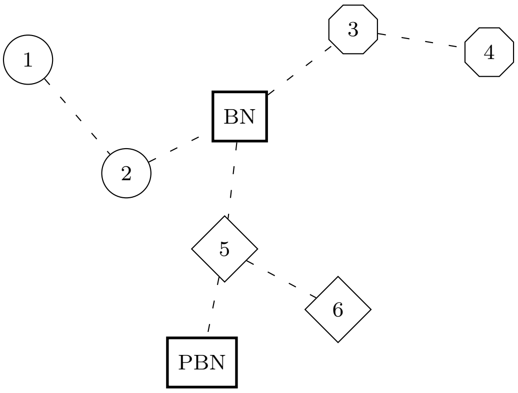


Figure 5.2: An Exemplary Network Scenario with Different Routing Schemes Utilizing Border Nodes (Circular Shaped Nodes = Reactive Routing Protocol, Octagon Shaped Nodes = Proactive Routing Protocol 1, Diamond Shaped Nodes = Proactive Routing Protocol 2)

the destination node. As the two proactive subnetworks in figure 5.2 discover their routes in a proactive manner, the *Border Node* has knowledge about all participants and does not need to forward the request in this example. The more complex case, in which multiple subnetworks are arranged in a chain and a route request has to pass multiple *Border Nodes*, is described in section 6.6.

On the other side, if the initiating node routes proactively, it cannot generate route requests to find nodes outside of its proactive scope. To overcome this obstacle, a proactive node can enter a *Passive Border Node* mode (represented by the rectangular shaped node) to enable a reactive routing mode simultaneously. This allows the node to send its request that cannot be answered by the current proactive routing domain to the border node which may answer or forward the request.

The *Border Node* functionality is enabled by adding some new message types to the routing protocols. This induces some additional routing overhead and conflicts with the defined requirements of efficient routing and backward compatibility. However, in order to support the interconnection of different routing areas, some additional knowledge has to be exchanged. Furthermore, if *SEREMA* did not support the connection of different routing areas, the network could get into trouble when different protocols are implemented on paramedics', firefighters' as well as victims' devices. To keep the compatibility to standard routing nodes, all protocol extensions are made inside the specification scope of the protocols.

If a node uses multiple simultaneous routing protocols it has to maintain multiple routing tables. To allow a simple access to the information inside those routing tables, *SEREMA* uses a *Routing Table Wrapper (RTW)* which looks up a requested route consecutively in all routing tables until the route is found or all routing tables have been checked (cf. figure 5.1). This gives each protocol the ability to continue using its own routing table and *SEREMA* the ability to simply look up routes from different routing protocols.

5.2 Comparison with the Requirements

The previous section introduced the new framework developed for *SEREMA* to overcome the drawbacks of existing adaptive routing approaches. The following will compare the requirements made (cf. section 4.3) to the presented scheme of *SEREMA*.

Adaptive Mechanism As the topic of the present work deals with adaptive routing, *SEREMA* was designed from scratch to provide this functionality. The framework supports various implemented routing protocols and allows switching between them during the run-time of the network.

Simultaneous Protocols As each node in *SEREMA* is provided with the functionality to autonomously select its own routing protocol, different simultaneous routing protocols can occur in the network. The interconnection between the different routing parts is managed by the *Border Nodes* which translate and forward the routing information from protocol one to protocol two and vice versa.

Routing Tables One of the most important parts of *SEREMA* – the *RTW* – enables the support of multiple simultaneous routing tables and, therefore, the adding of further protocols to *SEREMA* without much effort as the routing tables of the protocols do not have to be modified.

Packet Types To offer the cooperation between *SEREMA* and nodes only utilizing standard routing protocols, the message extensions of *SEREMA* are only made in the scope of the routing protocol specifications. This results in the fact that standard routing nodes which receive a new message type cannot understand the new information, but that they are able to forward it in the network.

Packet Conversions When the network utilizes multiple simultaneous routing protocols, the nodes in *SEREMA* have to convert the routing information only once per border as *SEREMA* does not use an additional routing protocol for the interdomain communication. This decreases the processing effort on a *Border Node* and, therefore, lowers energy consumption.

Efficiency To meet the efficiency requirement, *SEREMA* limits the number of additional packets to a minimum. This includes that each node only uses its own view on the network and does not exchange network statistics with other nodes. The newly introduced routing messages that could not be avoided to ensure the functionality of the adaptive framework – for example *Border Node Annotation (BNANNO)* messages – were kept small and are only induced sparsely.

Robustness As the routing in the considered scenarios has to be robust to ensure the communication between the disaster response teams, the nodes in *SEREMA* use completely decentralized monitoring as well as decision making to remain unaffected of single node failures.

Decision Quality One of the most important tasks of an adaptive routing scheme is the decision making as its quality directly effects the behavior of the network. Therefore, the *Monitoring Agent* in *SEREMA* tries to gather as much information as possible for the *Decision Maker* to increase the quality of the decisions. However, as *SEREMA* also has to comply with the efficiency requirement, the decision quality is limited as the *Monitoring Agent* is not allowed to exchange network statistics with other nodes for increasing the quality of its information.

Compatibility Since the nodes that are equipped with the adaptive routing mechanism of *SEREMA* are able to detect other routing protocols in their direct neighborhood and to act as *Border Nodes*, the network is able to include standard routing nodes into the network.

Extension Effort A very important requirement for the adaptive routing is to be flexible and easily extendable to further protocols. To fulfill this requirement, *SEREMA* implements only protocol extensions in the scope of the protocol specifications. A small exception in this scope is that *SEREMA* requires a small modification in the reactive routing tables to add the number of additional hops that were passed via tunnels.

	Jiang	*Hoebeke*	*Ramrekha*	*Forde*	*Yazir*	*Fujiwara*	*SEREMA*
Adaptive Mechanism	+	+	+	+	+	−	+
Simultaneous Protocols	−	+	−	+	−	+	+
Routing Tables	n	1	1	n+1	1	1	n
Packet Types	+	−	−	○	−	−	○
Packet Conversions	n/a	1	n/a	1	n/a	2	1
Efficiency	○	−	−	○	−	−	○
Robustness	−	+	+	+	+	+	+
Decision Quality	n/a	+	−	+	○	n/a	○
Compatibility	−	○	−	○	−	−	○
Extension Effort	−	−	+	−	+	+	○

Table 5.1: Comparison between the Related Work and SEREMA

Table 5.1 compares the adaptive framework of *SEREMA* with the related work introduced in section 4.3 and shows that *SEREMA* can compete with the other approaches as well as outperform them in some requirements.

5.3 Conclusion

This chapter introduced the new framework developed for *SEREMA* to overcome the drawbacks of existing adaptive routing approaches. The new adaptive routing scheme was introduced in section 5.1 while section 5.2 compared the new approach to the requirements made and showed that *SEREMA* is able to fulfill them. It was illustrated in table 5.1 that on the one hand *SEREMA* cannot perform as well as some other approaches since they are developed for specific use cases. On the other hand *SEREMA* never behaves as badly as a specialized approach which is utilized in another use case.

The following chapter 6 describes the different parts of *SEREMA* in more detail. This covers the *Monitoring Agent*, the *Decision Maker*, the protocol switching as well as the *RTW*. The functionality of the *Border Nodes* will be presented and the compatibility of *SEREMA* with nodes not supporting the protocol extensions will be discussed.

6 Architecture

Chapter 4 presented the related work regarding adaptive routing and justified why these approaches are not suitable for *MANETs* in disaster scenarios. To overcome the handicaps of these approaches, the new adaptive routing scheme *SEREMA* was introduced in the previous chapter 5. Details about the architecture of *SEREMA* [FSS$^+$12] and the involved items (cf. figure 5.1) are given in this chapter, starting with the selection of the used routing protocols in section 6.1.

6.1 Selection of the Base Routing Protocols

One of the requirements for *SEREMA* was to use two different routing protocols in its implementation. These two protocols should represent standardized and widely used routing mechanisms for *MANETs*. Further requirements were to simulate the approach for validation and having the possibility to run it on real hardware. The standard protocols should provide small extensions on the routing packets to allow the expandability for the *Border Node* mechanism of *SEREMA*.

	OLSR	*AODV*	*AOMDV*	*LARA*	*DSR*
Routing Technology	proactive	reactive	reactive	reactive	reactive
Standardization	RFC	RFC	–	–	RFC
Expandability	+	+	+	+	+
Mobility	low	medium	medium	medium	medium
Traffic	high	low	low	low	low
Node Density	high	low	medium	medium	low
Realtime Applications	+	–	–	o	–
ns-3-click	+	+	–	–	+

Table 6.1: Comparison between Routing Protocols for the SEREMA Framework

Table 6.1 gives a brief overview of the characteristics of the routing protocols introduced in section 3.4 and about *DSR* which is widely used and implemented in the scheme of Forde et al. [For05, FDO06]. The two protocols for *SEREMA* should be as different as possible to optimally show the advantage of adaptive routing in different scenarios. Furthermore, the protocols should be standardized and support protocol extensions. The defined requirements are only met by *OLSR*, *AODV* and *DSR*. The *OLSR* protocol is the only protocol in the table that routes proactively; hence, it is elected as the first protocol for *SEREMA*. The other two competitors are *AODV* and *DSR*, both routing reactively. A possible drawback of *DSR* regarding the routing overhead is that it is based on *Source Routing*. This means that the whole route from the source to the destination is carried inside a data packet. As *AODV* does not have this handicap, it is selected as the second protocol for *SEREMA*.

As an alternative to *AODV*, *AOMDV* or *LARA* could be used. They have some advantages over their predecessor but they are not standardized and currently, no implementations for *ns-3-click* are available.[1]

It should be mentioned that the protocols compared in table 6.1 represent only a small selection of all available routing protocols. In the scope of this dissertation *AODV* and *OLSR* were selected for the use in *SEREMA*. This choice was made for simulation purposes; the selection of further protocols would also be feasible.

6.2 Monitoring the Network Behavior

The nodes in ad hoc networks require information about the current network conditions to be able to make good routing protocol decisions. Parameters suitable for estimating the network state can be captured in two different ways. The first possibility is to measure only local parameters with the advantage that it will not induce any costs for packet transmissions. The drawback of local parameters is that they give only information about the local state of a node and cannot be used to make global routing decisions because other parts in the network could behave completely different. The second way to gather information about the network is to measure local parameters and exchange them with other nodes in the network to build an averaged value of the parameters. In this way a node receives information about other parts in the network

[1] Chapter 7 introduces why this simulation environment is required.

and gets a more global view. However, exchanging measurements with other nodes produces additional traffic in the network. Therefore, a balance between the use of local and more distributed information should be found. The following table 6.2 gives an overview of parameters that can be used for monitoring. The parameters used in *SEREMA* are highlighted.

Parameter	Obtained	*SEREMA*
Number of all . . .		
transmitted packets	locally	✓
received *AODV* packets	locally	✓
transmitted *AODV* packets	locally	✓
received *OLSR* packets	locally	✓
transmitted *OLSR* packets	locally	✓
dropped packets in queue	locally	✗
nodes	locally	✓
AODV nodes	locally	✓
OLSR nodes	locally	✓
neighbor nodes	locally	✓
AODV neighbor nodes	locally	✓
OLSR neighbor nodes	locally	✓
topology changes	locally	✓
Size of all . . .		
received packets	locally	✓
transmitted packets	locally	✓
received *AODV* packets	locally	✓
transmitted *AODV* packets	locally	✓
received *OLSR* packets	locally	✓
transmitted *OLSR* packets	locally	✓
Node Parameters		
Remaining Energy	locally	✗
Processor Load	locally	✗
Queue Load	locally	✗

Table 6.2: Overview about Parameters for Monitoring the Network State on a Node (✓= Implemented in SEREMA, ✗= Not Implemented in the Current Version)

These parameters can also be exchanged with other nodes to calculate an average value for a better representation of the state of a group of nodes in a part of the network. Furthermore, additional values can be derived based on information of the previously measured parameters. Examples for this could be:

Ratio of nodes using the same routing protocol When adaptive routing is used, the protocol switching decision should consider the protocols currently used by the neighbor nodes. Therefore, a node can derive the ratio of nodes using the same routing protocol by considering the number of neighbor nodes operating the same protocol and the number of all neighbor nodes.

Relative node speed As the *OLSR* protocol is suitable for networks with low mobility while *AODV* performs well in networks with a higher node movement (cf. section 3.4) the measurement of the node speed would be beneficial. However, as nodes in heterogeneous networks do not have to be equipped with sensors for measuring the absolute node speed, other parameters must be used. When having a closer look at the node speed, it becomes clear that the absolute speed of a node contains no information about the connectivity of the node to its neighbors as the neighbors could move with the same speed into the same direction and in this case no connectivity changes would occur. However, the relative node speed which describes the mobility of a node related to its neighborhood is able to provide information about the probability of changes in a node's connectivity. If a node has a high relative node speed which means that the difference of the node's speed compared to the speed of its neighbors is high, more connections will change in a given time period. As the nodes usually are not equipped with sensors for measuring their speed, they create a set S of the nodes in their neighborhood for equal time periods. Afterwards, when calculating the symmetric difference of the sets S_t and S_{t-1}, the resulting set $V = S_t \triangle S_{t-1}$ contains only the union[2] of the newly appeared and disappeared nodes since all nodes that are part of both sets were removed. The resulting set V can be used in combination with the number of all nodes in the direct neighborhood to get a ratio about the connectivity changes and, therefore, about a node's relative speed.

Routing overhead An important parameter for routing protocols is the generated routing overhead. The number or size of routing packets in the network can easily be accumulated. However, this only represents a number without any relation if this value is high or not. Therefore, the accumulated size of transmitted routing packets can be related to the size of all transmitted packets.

Nodes involved in data connections To determine a value representing the traffic load of the network a node can count the number of nodes that are involved in data transmissions and relate this number to the number of all nodes in the network.

[2] The union is part of the set theory in mathematics and denotes the set of all unique elements in the collection of all considered sets.

As the values measured on a node can vary over time and can contain outliers, the measurements in *SEREMA* are taken over a time period of 30 seconds to smooth them. As by definition *SEREMA* is not allowed to exchange measurements with other nodes, each *SEREMA* node can only use its locally obtained parameters. This is a challenge since no global view is available. Furthermore, the monitoring in *SEREMA* was designed to be as flexible as possible to further routing protocols. Therefore, the monitoring does not use routing protocol specific information, such as the number of all nodes in the network which could be obtained from the routing table of proactive routing protocols. This makes the measurements comparable among different routing protocols.

6.3 Decision Making

The information gathered by the monitoring agent is used by the decision maker to select the routing protocol that at the specific moment performs best for the individual node. This decision requires much knowledge, which is stored in the *Routing Mode Information* module and could be obtained by theoretical analysis of the protocols or by extensive simulation runs, about the behavior of the implemented routing protocols in different network scenarios.

A first approach for decision making in *SEREMA* could be based on previous work like Klein [Kle08], Lye et al. [LM06] or Haerri et al. [HFB06] and their results on the behavior of the routing protocols. However, as the behavior of the protocols depends on numerous parameters and the presented papers can only deal with some specific network scenarios using some specific simulators, it is hard to adapt to an adaptive routing approach like *SEREMA* where all possible network constellations should be covered. For example, the work of Haerri et al. [HFB06] states that *AODV* outperforms *OLSR* in terms of the produced routing overhead when the data rate is below 300 kbit/s. This might apply to the simulated scenario of this work, but if the used network technology is changed or the number of nodes in the network or their speed varies, the threshold will change.

Therefore, extensive simulation runs of the implemented protocols should be made to obtain as much information as possible about the behavior in a lot of different network constellations. This is a challenge since endless combinations of network parameters exist. Hoebeke [HMD12] stated this problem as:

> ”However, the development of more complex monitoring and decision-making functionality is needed to fully exploit the protocol’s capabilities, but is a huge research topic on its own.”

If an adaptive routing node changes its protocol, the node’s environment needs some time to adapt to this change. This results in the fact that adaptive routing should only be used for long-term changes in the range of tens of seconds to minutes. Therefore, decision making in *SEREMA* is triggered by definition every 30 seconds. However, further simulation runs with different time spans are recommended as future work. The monitoring accumulates the measured values in this time span to get smoothed values.

A challenge when calculating metrics for the switching decision is that some measured parameters represent absolute values while others represent relative values. To make this clear, consider the calculated routing overhead which is the ratio of transmitted routing packets to all transmitted packets. This results in a relative value between 0 and 100 percent and can easily be used for metric calculations. However, as the number of nodes in the network cannot be related to a maximum number of nodes as this information is unavailable, the absolute value could vary from 1 to infinity and therefore, it cannot directly be used in metric calculations. To allow a simple estimation of the network state for the decision making, *SEREMA* uses a score-based system where different parameters are analyzed and the participating routing protocols get points. Currently only a very basic scoring algorithm is implemented in *SEREMA* as no extensive knowledge about the behavior of the protocols in different scenarios is available. Therefore, the values for the thresholds as well as the related number of points for the protocols, mentioned in the following description of the scoring algorithm, were selected by definition.

The scoring process is illustrated in algorithm 6.1 as pseudo code. In the first step the scores of all protocols are set to zero (lines 1 and 2). Afterwards, the routing protocol that is currently the node’s main protocol gets a bonus point as it is preferred not to change the protocol to keep the number of changes in the network low (line 3).

The next step evaluates the node’s neighbors (lines 4 to 10). *SEREMA* could benefit from groups of nodes using the same protocol because in this case less *Border Nodes* are active resulting in a reduced routing overhead as well as reduced processing power which saves energy. If more than 75 percent of the neighbor nodes use the same protocol, the current protocol gets two points as most of the nodes in the neighborhood use the

```
Ensure: MainProtocol $\in \{AODV, OLSR\}$
score[AODV] = 0
score[OLSR] = 0
score[MainProtocol] = score[MainProtocol] + 1
if Ratio of Nodes Using same MainProtocol is $> 75\%$ then
    score[MainProtocol] = score[MainProtocol] + 2
else if Ratio of Nodes Using same MainProtocol is $50\% - 75\%$ then
    score[MainProtocol] = score[MainProtocol] + 1
else
    score[MainProtocol] = score[MainProtocol] − 1
end if
if Relative Node Speed is $> 75\%$ then
    score[AODV] = score[AODV] + 2
else if Relative Node Speed is $50\% - 75\%$ then
    score[AODV] = score[AODV] + 1
else if Relative Node Speed is $25\% - 50\%$ then
    score[OLSR] = score[OLSR] + 1
else
    score[OLSR] = score[OLSR] + 2
end if
if MainProtocol is AODV then
    if Ratio of Nodes in Active Connections is $> 30\%$ then
        score[OLSR] = score[OLSR] + 3
    else if Ratio of Nodes in Active Connections is $20\% - 30\%$ then
        score[OLSR] = score[OLSR] + 2
    else
        score[AODV] = score[AODV] + 1
    end if
else if MainProtocol is OLSR then
    if Ratio of Nodes in Active Connections is $< 10\%$ then
        score[AODV] = score[AODV] + 3
    else if Ratio of Nodes in Active Connections is $10\% - 20\%$ then
        score[AODV] = score[AODV] + 2
    else
        score[OLSR] = score[OLSR] + 1
    end if
end if
```

Algorithm 6.1: The Pseudo Code of the Scoring Algorithm in SEREMA

same protocol and the protocol should be kept. If 50 to 75 percent of the nodes use the same protocol, the current protocol gets only one point as the protocol is preferred to be kept. In the case that less than 50 percent of the neighbors use the same routing protocol as the node itself, the decision maker takes one point away from the current protocol to ease the change to another routing protocol.

When the decision maker analyzes the neighbors of a node, it also has a look on the relative node speed as it reflects the mobility of the nodes (lines 11 to 19). If the relative node speed is above 75 percent, *AODV* gets two points as it outperforms *OLSR* in highly mobile networks. If the speed is between 50 and 75 percent, *AODV* gets only one point as the protocol behaves more similar to *OLSR* in this case. A value of 25 to 50 percent provides one point to *OLSR* while more or less static nodes (0 to 25 percent) increase the score of *OLSR* by two points as *OLSR* can outperform *AODV* in static scenarios.

In the next step (lines 20 to 36) the load of the network is evaluated using the ratio of nodes that participate in active connections. Therefore, the decision maker distinguishes between *AODV* and *OLSR* as current main protocol. If *AODV* is the node's main routing protocol (lines 21 to 27) and the ratio of nodes in active data transmissions is greater than 30 percent, *OLSR* gets three points as *AODV* generates a lot of routing overhead in this case. If the ratio is between 20 and 30 percent, *OLSR* gets two points. For the case that less than 20 percent of nodes participate in active transmissions the traffic is relatively low and *AODV* gets one point as it can outperform *OLSR* in such scenarios. On the other side, if *OLSR* is the node's current main routing protocol (lines 29 to 35), *AODV* earns three points if the ratio of nodes in active data transmissions is below ten percent and thus, *AODV* would outperform *OLSR* in terms of produced routing overhead. If the value is between 10 and 20 percent, *AODV* gets two points and if more than 20 percent of nodes are in active transmissions *OLSR* gets one point as in this case it generates less routing traffic than *AODV*.

The presented thresholds and the assigned number of points were defined in the scope of this dissertation to show a basic principle of how the structure of the *Decision Maker* algorithm can be designed. As further work and to enable a good performance when applied to real scenarios, extensive simulation runs are required to optimize the algorithm as well as its parameters.

After assigning the points to the protocol scores, the decision maker compares the scores to identify the winner and to adjust the routing mechanism. The algorithm for the decision making could also aim at other goals besides minimizing the routing overhead or consumed energy, e.g. proactive routes could be used to widely-used services in the network to decrease the latency for route discovery while other destinations are routed reactively as stated by Hoebeke [HMD12].

6.4 Protocol Switching

This section describes how the protocols in *SEREMA* are changed when the decision maker decides to use another routing protocol. As the adaptive routing framework was implemented in *Click Modular Router (Click)*[3] all routing protocols remain in the system's memory and run in parallel. To enable the selection of single protocols simple switches are used which are placed on the input and output ports of the routing protocols. Even though all protocol implementations run all the time, *SEREMA* is able to connect or disconnect specific protocols from the outside world. The advantage of this scheme is that disconnected protocols can still maintain their routing table which implies the deletion of expired entries. As long as the routes are valid, they can be accessed by the *RTW* (cf. section 6.5). A drawback of simultaneously using parallel routing protocols in the memory is the huge memory consumption. However, the main memory consumption of a routing protocol is related to its routing table. In section 4.4 was shown that this memory consumption can be disregarded. Furthermore, the overall memory consumption of the different routing tables is kept low as a result of the dynamic memory allocation used by the routing protocols. A route consumes only memory for storage as long as it is kept in the routing table.

6.5 Routing Table Wrapper

A big advantage of *SEREMA* over other adaptive routing schemes is its *RTW* [SBH+13]. This module provides the access to the routing tables of the implemented protocols and gives each protocol the ability to use its own unmodified routing table. Furthermore, it allows accessing the routing tables of currently deactivated protocols, thus enabling the use of those routing table entries which might still be valid directly after a protocol change. This allows *SEREMA* to supply data packet forwarding with valid routes di-

[3] Described in chapter 7.

rectly after a protocol change when the new routing protocol might still not be ready to provide the required routes. Hence, the change of the routing protocol can be done without any impact on active data connections. Figure 6.1 shows the structure of the *RTW* in the *Click* implementation.

Packets destined for being forwarded enter the *RTW* at the *Packet Input.* At the beginning the packets are redirected based on the currently active main routing protocol and subsequently they are marked corresponding to the active main routing protocol. Thereafter, the packets are guided to the main routing protocol's *Routing Table Look-up* element that tries to find an available route in the corresponding routing table. In this example the processing will be described for *AODV* being the node's main routing protocol. If the *AODV* routing table contains a suitable route, the packet is forwarded to the *Route found* output. Otherwise, if no route is available, the packet is checked at the *OLSR Paint Classifier* if it is marked as coming via the *OLSR Paint* element. As the packet is not marked as coming from *OLSR*, it is guided to the *OLSR Routing Table Look-up* element which tries to find a route in the *OLSR* routing table. Depending on whether a route was found or not, the packet is forwarded to the *Route found* output or to the next classifier. This procedure continues until all routing tables have been checked and a valid route was found or the packet arrives at the *AODV Paint Classifier.* This element detects that the packet which started with the *AODV Routing Table Look-up* element already passed all look-up elements and no route was found. Therefore, further routing table look-ups would not benefit the routing and the packet is guided to the *Route not found* output.

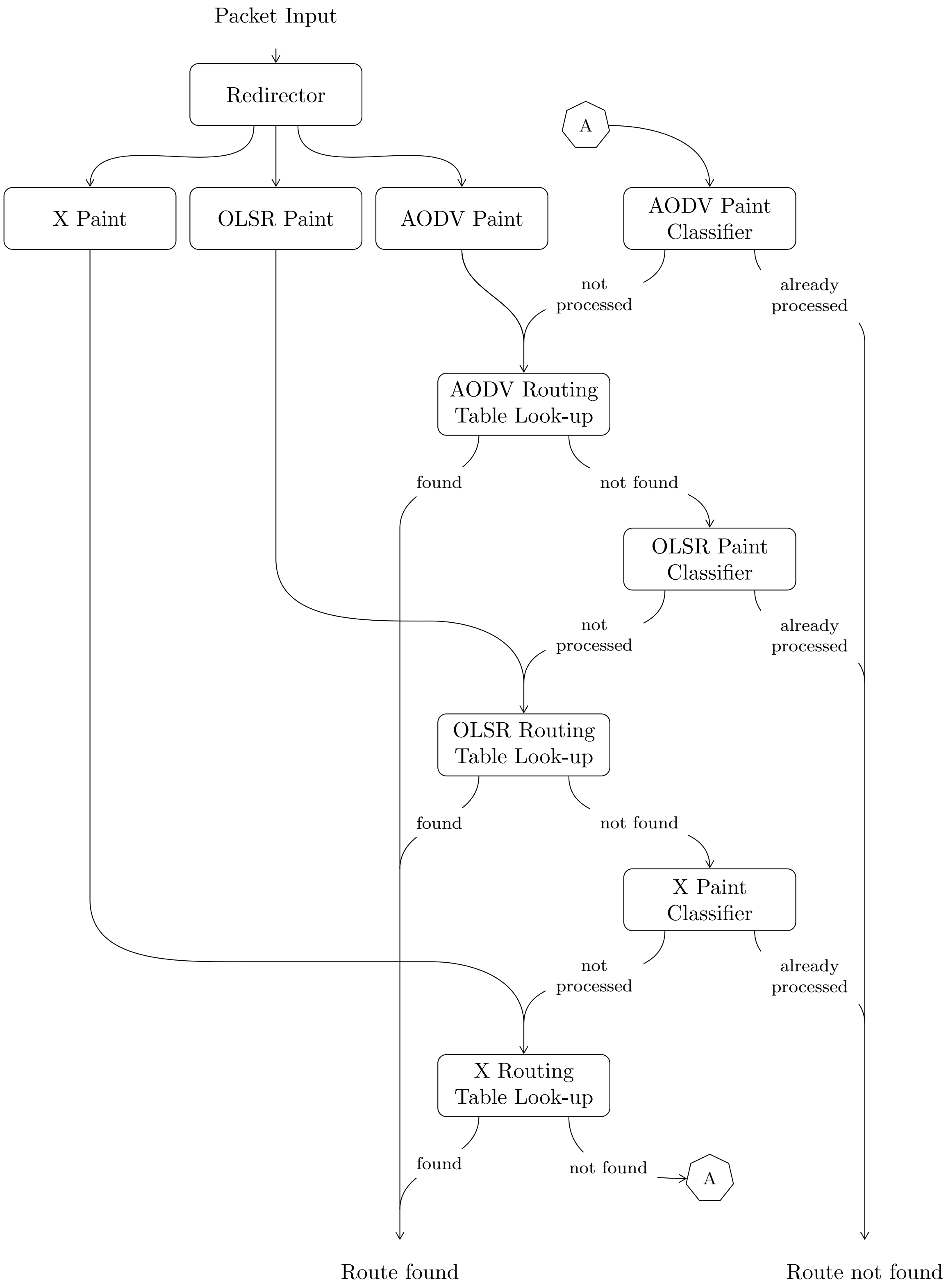


Figure 6.1: The Structure of the RTW in SEREMA

6.6 Connecting Routing Subnets Using Border Nodes

If an adaptive routing scheme shall support multiple simultaneous protocols in the network, it needs a mechanism to interconnect nodes and subnetworks which use different routing protocols. This section describes the mechanism used in *SEREMA* to enable this functionality.

6.6.1 Definition of Border Nodes

When assuming that parts of the network only use *AODV* or *OLSR* – because they decided to use those protocols or the nodes are only equipped with a single protocol – the different protocols cannot exchange routing information with each other. *SEREMA* provides a functionality for interconnecting different routing domains by introducing *Border Nodes* that are able to operate multiple routing protocols simultaneously to allow the communication between different routing areas.

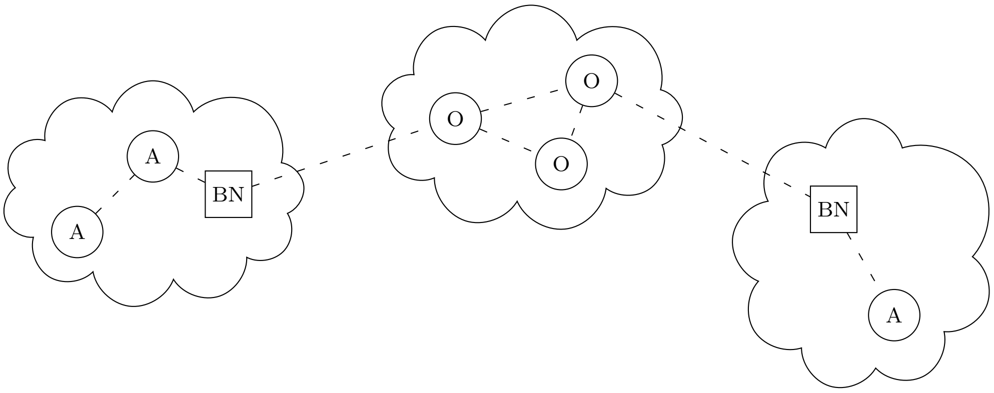


Figure 6.2: An Example Network Scenario for the Interconnection of Different Routing Domains

Such a network scenario is illustrated in figure 6.2. The nodes marked with an *A* are using *AODV*, nodes with an *O* are operating *OLSR* while the *BN* nodes represent *Border Nodes* that allow the interconnection of the different routing subnets.

A challenge when using *Border Nodes* is to decide where and when to place a *Border Node* and which nodes should activate this functionality. To obtain an ideal positioning of *Border Nodes* a global view on the network is required. However, such a 'god-mode' is not available in *SEREMA* and, furthermore, it was defined that the nodes should not exchange monitoring information with each other in order to minimize routing overhead. This brings nodes to the problem that they have to decide autonomously whether to become a *Border Node* or not. A huge amount of *Border Nodes* would lead to extensive routing overhead while an insufficient number of *Border Nodes* could cause unconnected areas in the network.

If reactive and proactive routing schemes are compared, it is quite evident that reactive routing nodes have better capabilities for route discovery than proactive nodes as they can generate route requests for unknown destinations. In contrast, proactive nodes can only base oneself upon their proactive routing knowledge. Therefore, it was defined that in *SEREMA* only reactive routing nodes become *Border Nodes*. In the current *SEREMA* implementation the *AODV* protocol is selected for this task. For the case that the number of implemented routing protocols in *SEREMA* is increased, a look-up table should be implemented to handle all possible combinations of routing protocols at a border and to provide an information which node has to activate its *Border Node* functionality in a specific case.

Another option would be to develop a new mechanism that can be used to negotiate which node becomes *Border Node* if different routing nodes come in contact with each other. However, the negotiation would imply some type of new messages in the network and, therefore, increase routing overhead.

If a node currently operating *AODV* and being equipped with the *Border Node* functionality identifies an *OLSR* node in its 1-hop neighborhood, it activates its *Border Node* functionality and enables its *OLSR* protocol to allow the interconnection of the *AODV* and *OLSR* parts.

The presented mechanism works well for identifying required *Border Nodes* to guarantee the connectivity[4] of all nodes. However, the mechanism tends to use too many *Border Nodes* since each *AODV* node on a border activates its *Border Node* mechanism when identifying an *OLSR* node, resulting in unnecessary routing overhead. Therefore, the algorithm should be extended to make a distributed decision between the nodes in the network about which nodes should act as *Border Nodes*. This mechanism is not implemented in the current version of *SEREMA* since a basic requirement of *SEREMA* is the minimization of routing overhead and, therefore, distributed decisions were avoided. The additionally required information for a distributed decision could

[4] Assumed that the network is not divided into multiple partitions caused by limited transmission ranges.

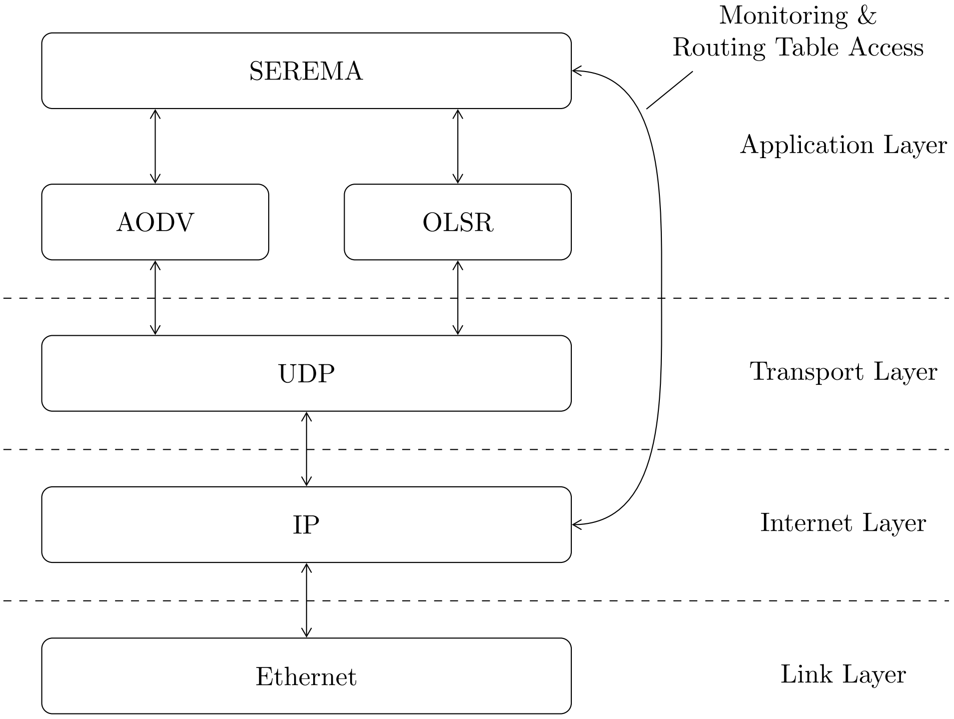


Figure 6.3: The Arrangement of AODV, OLSR and SEREMA in the Internet Protocol Suite [Bra89]

be exchanged together with improved network statistics for the *Monitoring Agent* (cf. section 5.2).

When increasing the quality of the *Border Node* selection mechanism further information, such as the higher suitability of nodes with higher remaining energy and more processing power to act as *Border Node* compared to nodes with weaker resources, could be considered.

To increase the understandability of the cooperation between the routing protocols and *SEREMA*, figure 6.3 shows the arrangement of the routing protocols (*AODV* and *OLSR*) as well as *SEREMA* inside the *Internet Protocol Suite* [Bra89].

6.6.2 Border Node Annotation

When a node operates as *Border Node* it offers its specialized skills to other nodes. The nodes in reactive routing areas that might use a *Border Node* do not have to know its existence because a generated *RREQ* will be forwarded in the network until it reaches a *Border Node*. On the other hand – in proactive routing parts – a node cannot search for a *Border Node* because of the missing route request feature. Therefore, a *Border Node* injects newly introduced *BNANNO* messages (cf. figure 6.4) into proactive network parts to inform the nodes about its existence.

0 7	8 1516 2324 31
BNMODE	Reserved
Border Node Sequence Number	

Figure 6.4: The Structure of BNANNO Messages

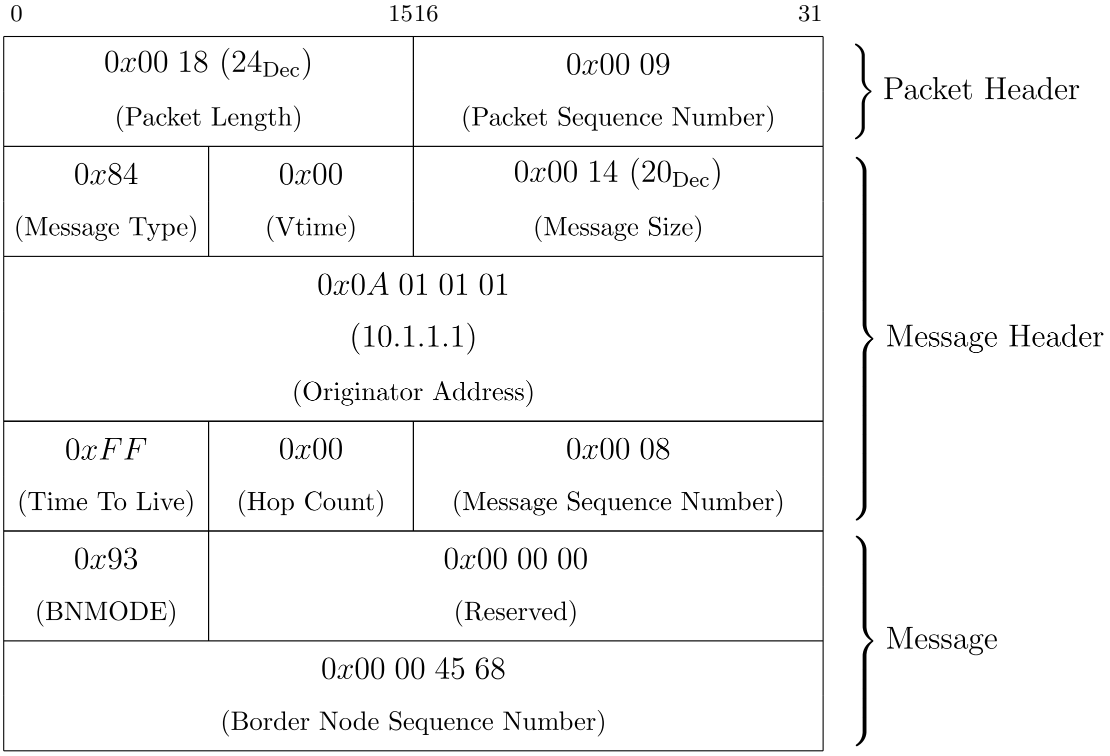
Bitnumber	Name	Description
0	Border Node	Border node mode is active
1	Passive Border Node	Passive border node mode is active
2–4	Main Protocol	Bit 2 = Reserved Bit 3 = AODV Bit 4 = OLSR
5–7	Active Protocols	Bit 5 = Reserved Bit 6 = AODV Bit 7 = OLSR

Table 6.3: The Bit Structure Inside the BNMODE Field

These messages contain the current mode of a *Border Node* and its *Border Node Sequence Number* which is required for answering *RREQs* coming from routing protocols using sequence numbers for destination nodes not being equipped with the sequence number functionality. This process will be explained in more detail in subsection 6.6.3. The *Border Node Mode (BNMODE)* field contains information about the current state of the originating *Border Node* (cf. table 6.3). Both *Border Node* bits represent the state of a node's active or passive[5] *Border Node* functionality. A *BNANNO* message must have set exactly one of these bits. If both bits are set to a value of one, the message is corrupt and should be dropped as a *Border Node* cannot operate in active and passive mode at the same time. The case where both bits are zero is not allowed either as a *BNANNO* message cannot be originated from a node that has neither the active nor passive *Border Node* mode operating. The remaining bits in the *BNMODE* field inform about the node's current main routing protocol and all active protocols that are currently used for the *Border Node* functionality.

Figure 6.5 shows an example of an *OLSR* packet as introduced in section 3.5 containing a *BNANNO* message. The message, structured as illustrated in figure 6.4, consists of eight bytes that are shown in the *Message* part in figure 6.5. The values are in

[5] Will be described in subsection 6.6.4.1.



0x00 18 (24$_{\text{Dec}}$) (Packet Length)		0x00 09 (Packet Sequence Number)	Packet Header
0x84 (Message Type)	0x00 (Vtime)	0x00 14 (20$_{\text{Dec}}$) (Message Size)	Message Header
0x0A 01 01 01 (10.1.1.1) (Originator Address)			
0xFF (Time To Live)	0x00 (Hop Count)	0x00 08 (Message Sequence Number)	
0x93 (BNMODE)	0x00 00 00 (Reserved)		Message
0x00 00 45 68 (Border Node Sequence Number)			

Figure 6.5: A BNANNO Message as Part of an OLSR Packet

hexadecimal notation and the 0x93 represents the binary value of 10010011. This tells the receiver that the originator node (node 10.1.1.1 in this example) operates as *Border Node*, uses *AODV* as main routing protocol and has activated its *AODV* as well as *OLSR* protocols.

The *BNANNO* messages are attached to *OLSR* packets as described in subsection 3.5.4 and simply flooded into the proactive network part to reach all nodes. A modification of the forwarding algorithm would be possible to use the *MPR* mechanism of *OLSR* to reduce the routing overhead. However, this causes the problem that a node that has recently changed to the *Border Node* mode and, therefore, recently activated its *OLSR* protocol does not know its *MPRs* as the *TC* messages of *OLSR* are only sent every five seconds.

Beside the solution described in this subsection for annotating *Border Nodes* also service discovery mechanisms could be used. An example for a suitable solution that uses *Service Advertisement (SADV)* messages and which is compatible to *SEREMA* was presented by Schellenberg et al. [SSK^{+}14].

6.6.3 Border Node Sequence Number

Routing protocols such as *AODV* utilize sequence numbers to distinguish between up-to-date and obsolete routing information in received routing packets. A challenge occurs when a *Border Node* has to generate a route reply that requires a sequence number for a destination node whose protocol does not provide sequence numbers. To overcome this problem every *Border Node* maintains a *Border Node Sequence Number* for the proactive routing part that is used in the described scenario. This sequence number starts with a random value and is incremented each time the number is used for answering a request.

In the case that a reactive and proactive routing area are interconnected by multiple *Border Nodes* it must be ensured that all *Border Nodes* use the same sequence number for a specific proactive subnetwork. The previously presented *BNANNO* messages that are periodically generated by the *Border Nodes* contain the sequence number of their originator *Border Node*. When another *Border Node* receives this information – which can only happen via proactive network parts – it compares the received sequence number with its own. If the received number is greater the *Border Node* updates its own number to the value of the received sequence number. In this way all *Border Nodes* of a specific proactive routing area can synchronize their *Border Node Sequence Numbers*.

6.6.4 Route Discovery between Different Routing Domains

This subsection describes the route discovery procedure over different routing domains using *SEREMA*. Each of the following examples considers the network scenario shown in figure 6.6. The network consists of five subnetworks, two of them using *OLSR* (the node names start with *O*) while the others use *AODV* (the node names start with *A*) as main routing protocol. The *AODV* nodes directly connected to *OLSR* nodes already entered their *Border Node* mode (cf. subsection 6.6.1).

When considering the route discovery activity between different routing domains it can be distinguished between two cases, the discovery between adjacent subnetworks and the discovery via multiple intermediate networks. The following examples mention only routing domains containing multiple nodes. However, *SEREMA* also supports routing

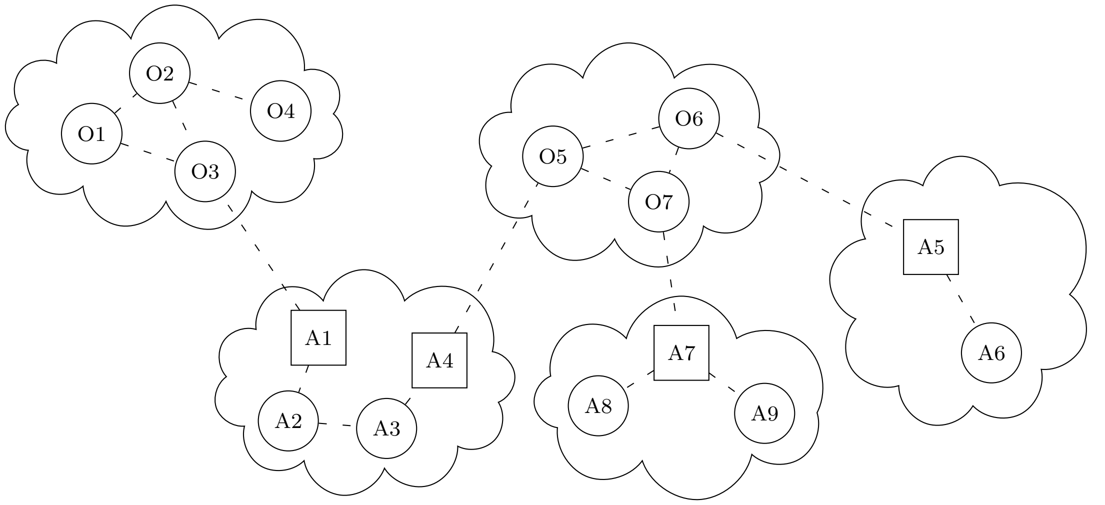


Figure 6.6: An Example for a Route Discovery Process over Different Routing Domains (A = AODV, O = OLSR, Rectangular Nodes = Border Nodes)

domains consisting of exactly one node. This mechanism supports that each node can select the best routing protocol for its current requirements.

6.6.4.1 Route Discovery to Adjacent Routing Domains

If node A2 wants to discover a route to node O7 it sends a route request as usual in reactive routing protocols. This request is distributed in the reactive network until it reaches a *Border Node*. Let this *Border Node* be node A4 in this example. The node receives the request and checks its reactive routing table if it can answer the request with a reply. As the *Border Node* cannot find node O7 in its reactive routing table, it examines its proactive routing table where it finds the destination node O7 because the proactive network, consisting of the nodes O5, O6 and O7, includes the directly connected *Border Nodes* A4, A5 and A7. Therefore, the node generates a route reply addressed to the originator node A2. The *Destination Sequence Number* for O7 is set to the *Border Node Sequence Number* of node A4. When the route reply arrives at node A2 it knows the route to the requested destination.

With this route node A2 is able to send data packets to O7 as the nodes on the reactive part of the route learned the path to the *Border Node* which in turn – owing to the *RTW* – forwards received data packets to the destination node O7 in the proactive area. However, if the connection is required to be bidirectional, the proactive network part gets into trouble since it does not know the route via the *Border Node* to node A2. Hence, the *Border Node* periodically injects *HNA* messages (cf. subsection 3.5.4) into the proactive network part to inform the nodes about the route to node A2 via the *Border Node*.

On the other hand, if the proactive routing node O1 requires a route to node A3 in the adjacent reactive network it cannot manage this with its pure proactive behavior because the nodes in a proactive network only know the participants in their own subnet. When considering figure 6.6 someone could mention that only a rule is required that defines that *Border Nodes* inject all reactive nodes into the proactive network via *HNA* messages. However, this is impossible as *Border Node* A1 cannot know all participants in its reactive domain and furthermore, the requested destination could also be node A6 in a foreign subnetwork. Furthermore, this would heavily increase the produced routing overhead.

To handle this task, node O1 (cf. figure 6.6) – equipped with the *SEREMA Border Node* functionality – activates its *Passive Border Node* mode. In contrast to normal *Border Nodes* the *Passive Border Nodes* are not available for other nodes' concerns, but only for their own purpose. This mode enables O1 to send route requests to the *Border Nodes* connected to the proactive network which are already known through previously received *BNANNO* messages.

A further problem occurs when node O1 tries to send its reactive route request to *Border Node* A1 because all intermediate nodes between O1 and the *Border Node* only operate their proactive routing protocol and, therefore, are not able to process and forward reactive routing packets. To overcome this problem, node O1 uses *IP* in *IP* encapsulation as described by Perkins [Per96] to send its reactive packet to the *Border Node* A1. Each *Border Node* which receives such a packet decapsulates the packet and tries to answer the request with the knowledge of its routing tables. If no route is available the *Border Node* forwards the route request into its reactive subnets where it propagates towards the destination node. In figure 6.6 this means that *Border Node* A1 forwards the request to node A2. Thereafter, when the *Border Node* receives the corresponding route reply it tunnels the reply through the proactive network to node O1 and periodically injects *HNA* messages to inform the proactive subnet about the route to the destination A3. The *HNA* messages are generated by the *Border Node* as long as the route is in use. This is detected by monitoring forwarded data packets.

6.6.4.2 Route Discovery over Multiple Routing Domains

Route discoveries between different adjacent routing subnetworks via a single *Border Node* have already been discussed. In this section, the route discovery procedure over multiple intermediate subnets will be discussed. Therefore, consider the scenario where node A6 wants to discover a route to node O4.

The reactive node A6 generates a route request that is forwarded to the *Border Node* A5. As in A5's routing tables no suitable route to node O4 is available it tunnels (using *IP* in *IP* encapsulation) a copy of the request to every other *Border Node* it has learned by received *BNANNO* messages. In this way the *Border Nodes* A4 and A7 receive a copy of the request. Both nodes forward the request into their reactive routing domains. The packet forwarding of node A4 involves that the request arrives at the *Border Node* A1 that in turn checks its routing tables for the destination. As the destination node is part of A1's proactive routing domain, the *Border Node* generates a route reply back to node A6 and injects a *HNA* message into its proactive subnet (consisting of the nodes O1, O2, O3 and O4). On its way back the reply is tunneled from node A4 to A5.

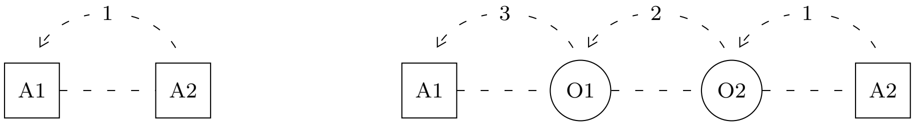
At this point an important performance aspect of the reactive route discovery should be mentioned. A request originated by node A3 will be forwarded via *Border Node* A4 to the *Border Nodes* A5 and A7 and hence, into both other reactive subnetworks. This might induce huge routing overhead especially in large networks. As *SEREMA* uses the *AODV* protocol for route discovery it benefits from its *Expanding Ring Search* feature. This allows limiting the search radius at the beginning and enlarging it step by step if the destination is not found. With this mechanism the generated routing overhead might be reduced. However, the latency for the route discovery procedure might increase.

Another option for reducing the routing overhead could be that a *Border Node* does not forward a request to all of the other reactive networks. Instead, it could select one subnet for forwarding and if the destination node is not found, the *Border Node* could select another subnet or increase the number of subnets it forwards the request to. However, this could heavily increase the latency for the route discovery procedure.

6.6.5 Distribution of Reactive Routing Information

After the explanation of the *SEREMA* routing behavior between different routing domains routing protocol specific requirements should also be discussed. It was shown that the communication between different routing domains is based on a reactive routing mechanism. In *SEREMA* this job is done by the *AODV* protocol that requires the exchange of routing information between neighboring nodes. This also implies the communication between reactive neighbors connected via a tunnel. Each *AODV* node that is implemented according to the standard [PBRD03] requires *HELLO* messages from its neighbors to be able to maintain its routing table and precursor list correctly. Therefore, *SEREMA* tunnels the reactive *HELLO* messages between the *Border Nodes* that are directly connected to the same proactive routing area. In addition to *HELLO* messages, *RERR* messages are also allowed to pass tunnels to ensure the correct invalidation of routes.

The tunneling of *RREQ* and *RREP* packets involves a further challenge because both packet types contain hop count fields to measure the length of a route. If these packets are tunneled their traveled distance increases but their hop count does not. This results in the problem illustrated in figure 6.7.



(a) Intermediate Nodes are Hidden (b) Intermediate Nodes are Shown

Figure 6.7: The Problem with the Hop Count When Tunneling

Let node A2 tunnel a *RREQ* or *RREP* packet to node A1. As both nodes are connected by a tunnel it seems for them that they are only one hop away from each other (cf. figure 6.7a) and the *AODV* node A1 expects to receive packets from A2 with a hop count of one. However, the packets received from A1 traveled over three hops as they were transmitted via the tunnel (cf. figure 6.7b). This results in the problem that A1 ignores the *HELLO* packets coming from A2 as *HELLOs* are only allowed to travel for a distance of one hop. A solution for this problem could be to change the hop count for a packet transmitted via a tunnel only by one hop. However, this would provoke that *AODV* could not estimate the real length of its routes. To cope with this challenge *SEREMA* introduces a further hop count – called *THops* – for *RREQ* and *RREP* packets which is attached to the end of the packets as described in the *AODV* specification [PBRD03]. The specification provides extensions in the *Type Length Value (TLV)* format as illustrated in figure 6.8 which are directly appended after the routing message data.

0 7 8 1516 2324 31

Type	Length	Value

Figure 6.8: The Structure of AODV Message Extensions

The *Type* field contains the type number of the message extension and is set to a value of ten for the *THops* extension. The *Length* field informs about the size of the following *Value* field in bytes. The *Value* of the extension is used to carry the number of hops the packet traveled via tunnels. An example of a *RREP* message with *THops* extension can be seen in figure 6.9 where the message traveled over four *THops*. *SEREMA* uses a *Value* field with a length of two bytes. The *AODV* specification mentions no rules for the length of this field, but *Wireshark* reports an error if the length is not a multiple of two.

0 1516 31

<table>
<tr><td>0x02
(Type)</td><td>0x00
(Flags)</td><td>0x00 00
(Reserved)</td><td>0x00
(Prefix S.)</td><td>0x00
(Hop Count)</td><td rowspan="5">RREP</td></tr>
<tr><td colspan="5">0x00 00 45 68
(Destination IP Address)</td></tr>
<tr><td colspan="5">0x00 00 45 68
(Destination Sequence Number)</td></tr>
<tr><td colspan="5">0x00 00 45 68
(Originator IP Address)</td></tr>
<tr><td colspan="5">0x00 00 45 68
(Lifetime)</td></tr>
<tr><td>0x85
(Type)</td><td colspan="2">0x02
(Length)</td><td colspan="2">0x00 04
(Value (THops))</td><td>THops Extension</td></tr>
</table>

Figure 6.9: The THops Extension as Part of an AODV RREP

With this new extension a tunnel endpoint receiving a packet has the ability to add the number of tunnel hops to the *THops* field before it processes and forwards the packet. The number of tunnel hops is estimated by taking a look at the *TTL* field of the outer *IP* header, bearing in mind that nodes encapsulating packets set the *TTL* to a value of 255. The receiving *Border Node* can thus calculate the number of *THops* with the formula: THops = 255 − Outer IP Header TTL − 1. The minus one at the end of the equation is required as the regular hop count is also increased by one when a packet is forwarded via a tunnel. In this way *AODV* nodes subsequently processing the packet get the number of the regular packet hops pretending a network without tunnels and allowing an *AODV* operation without problems. Furthermore, the nodes receive the sum of all *THops* on a route which enables them to calculate the correct length of the route.

The *AODV* routing table has to be extended to be able to store the new *THops* value per route which implies a small modification of the standard routing protocol implementation. Additionally, the algorithm for route insertion and look-up has to be modified for considering the additional hop count information. This can be done by simply replacing the hop count value by the sum of the hop count and *THops*. Thereafter, a node is able to consider the tunnel hops when searching for the shortest route to a destination.

6.7 Compatibility

This section will discuss the compatibility of *SEREMA* to other nodes not supporting its extensions and show in which scope a standard routing node can operate if it is connected to a *SEREMA* network.

As previously mentioned, the nodes in *SEREMA* use different routing protocols. However, to guarantee a route discovery mechanism that operates without any difficulties, all *SEREMA* nodes must have implemented one common routing mechanism that can be used for the communication between different routing areas. Furthermore, this prevents that two nodes which do not have any common protocol come into contact with each other, hence would not be able to communicate. In *SEREMA* the *AODV* protocol was selected for this task.

6.7.1 Connection of SEREMA to Standard Routing Protocols

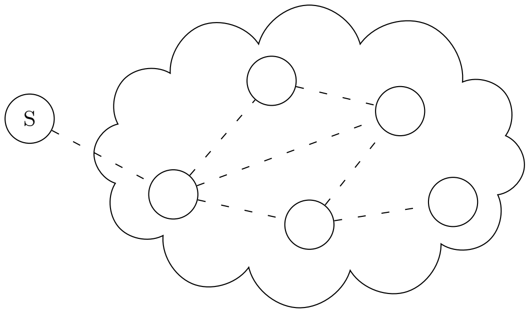


Figure 6.10: Connection of a SEREMA Node (S) to a Standard Routing Domain

If a *SEREMA* node gets in contact with an area that uses a standard routing protocol (cf. figure 6.10), it can simply connect to this network if the used protocol is implemented in the *SEREMA* node. For example, if the *SEREMA* node connects to an *OLSR* network, it either previously used its *OLSR* protocol and has to do nothing for the connection or it used *AODV* and would immediately activate its *Border Node* mode. This results in the generation of *BNANNO* messages which will not be processed but still be forwarded by the standard *OLSR* nodes in the network. If the *Monitoring Agent* of the *SEREMA* node detects that it is only surrounded by *OLSR* nodes, the *Decision Maker* realizes that the node cannot benefit from its *Border Node* mode and deactivates the *AODV* protocol, so that the node becomes a pure *OLSR* member of the network.

On the other side, if a *SEREMA* node connects to a pure *AODV* network it can only connect to this network if its *AODV* protocol is active. If the node routes proactively with *OLSR* it expects that an *AODV* node in the network acts as *Border Node*. However, standard *AODV* nodes do not support this mode of operation. Therefore, the *SEREMA* node awaits the *OLSR* hello interval three times to receive an *OLSR HELLO* message from the network which would suggest *OLSR* neighbors or a *Border Node*, respectively. If this *HELLO* is not received, the node assumes that it is connected to a network not supporting *SEREMA* and changes to the pure *AODV* mode.

The nodes in the standard routing domain cannot understand the message extensions (*BNANNO* and *THops*) of *SEREMA*. As the communication between different routing areas is not considered in standard routing protocols, the nodes can simply ignore the message extensions possibly generated by *SEREMA* members. Since the extensions are in the scope of the protocol specifications, the nodes know how to deal with them.

6.7.2 Connection of Proactive Routing Protocols to SEREMA

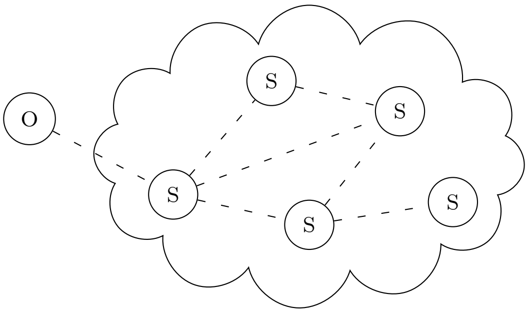


Figure 6.11: Connection of a Proactive Routing Node (O) to SEREMA (S)

Beside the case that a *SEREMA* node is connected to a standard routing domain, a standard routing node could also be connected to a *SEREMA* domain (cf. figure 6.11). The *Border Node* functionality of *SEREMA* allows an easy connection of standard *OLSR* nodes to the network. If the communication counterpart inside the *SEREMA* network uses its *OLSR* protocol the node can simply connect. If the *SEREMA* node routes reactively with *AODV* it is required to activate its *Border Node* mode when detecting another protocol in its neighborhood to allow the connection of the other node.

A problem in this mode of operation occurs when a purely proactive routing node cannot generate route requests for destination nodes that are unknown to the proactive routing domain. This is a handicap of standard *OLSR* nodes cooperating with *SEREMA*. The only chance for such a node to get information about other routing areas is to receive *HNA* messages from *Border Nodes*.

6.7.3 Connection of Reactive Routing Protocols to SEREMA

If the connecting node routes reactively with *AODV* it can simply connect to a *SEREMA* domain (cf. figure 6.12) if *AODV* is used. If this is not the case and *OLSR* is used the *Monitoring Agent* of the standard routing node's communication counterpart detects that the *AODV* node in its neighborhood does not generate *OLSR* messages. If it does not receive *OLSR HELLOs* within three times the *HELLO* emission interval, the *Decision Maker* decides to activate the *Border Node* mode because the other node has no *SEREMA* functionality.

The protocol specification handles the forwarding of unknown message extensions on a node. However, a problem of pure *AODV* nodes in *SEREMA* occurs when these nodes receive packets with the *THops* message extension set to a value greater than zero. If the standard routing node learns a route from a received message it cannot consider the *THops* information and therefore, it learns a route with a hop count that is shorter than the real length of the route. Thereafter, when the node offers its incorrect routing knowledge to other nodes, routing loops could occur. To overcome this problem, *SEREMA* should be modified to transmit the complete length of a route calculated as the sum of normal hops and tunnel hops in the standard hop count field of *AODV* messages. The *THops* field in the message extension should additionally inform about the number of *THops* that are part of the complete hop count. This information can be used by *Border Nodes* to identify direct *AODV* neighbors that are connected via a tunnel and, therefore, have a tunnel hop count greater than one. Furthermore, this modification enables standard *AODV* nodes to participate in *SEREMA* networks without problems.

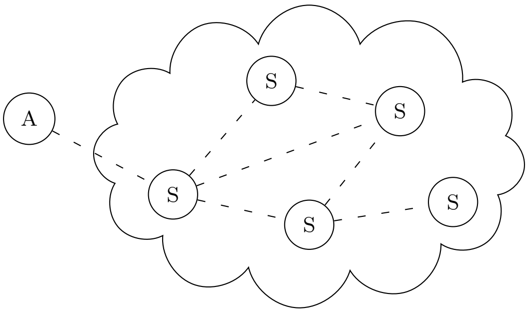


Figure 6.12: Connection of a Reactive Routing Node (A) to SEREMA (S)

6.8 Conclusion

This chapter introduced the detailed architecture of *SEREMA*. The selection of the two routing protocols used was explained and the protocol selection using the *Monitoring Agent*, the *Decision Maker* and the protocol switching mechanism was introduced. It was shown how *SEREMA* can benefit from the newly introduced *RTW* and simply use multiple routing tables. The communication between different routing protocols via *Border Nodes* was illustrated in detail and pointed up with exemplary scenarios. Required extensions for the existing routing protocols were explained, including their message structure and functionality. The compatibility to existing routing protocols and nodes only supporting a standard routing protocol was discussed, showing that *SEREMA* can provide connectivity to standard routing nodes to a certain degree.

During the design of *SEREMA* a mechanism for the resolution of names to addresses using adaptive routing [FSS+12] was developed. This approach is compatible to the presented *Border Node* functionality and could be implemented into *SEREMA* to allow a completely decentralized and, therefore, robust mechanism to resolve host names as well as service names into addresses.

7 Simulation Environment

In this chapter a network simulator will be selected to validate the approach of *SEREMA* presented in chapter 6. As the implementation of routing protocols is a time consuming task it would be advantageous to have a simulation environment that allows the reuse of the code later for tests on real systems. This essential requirement should be kept in mind when selecting the simulator.

7.1 Introduction

To validate the functionality of *SEREMA*, a testbed with real hardware could be applied. However, when using real hardware it becomes a challenge to validate a network with, for example, 50 mobile nodes with changing network parameters like the transmission range or the node speed. Furthermore, the acquisition of the resulting data after the test becomes a time consuming task. Beside the challenges regarding the testing, the cost for using a huge number of real nodes would increase heavily.

A reasonable solution for this challenge is to use network simulators that can reproduce the behavior of networks with the required level of detail. This allows a simple and economical possibility for the validation of network protocols. When using network simulators it should be considered that more detailed simulation runs require more computing time. Therefore, a balance between the required level of detail and the simulation time has to be found. For the simulation of network protocols a huge number of simulators are available.

Some of the most widely used network simulators in the area of research are *ns-2* [Unie], its successor *ns-3* [Cond], the *Global Mobile Information System Simulator (GloMoSim)* [ZBG98] as well as the *Objective Modular Network Testbed in C++ (OMNeT++)* [OMN]. Beside the already mentioned simulators, there is *OPNET* [Riv] which provides a *Graphical User Interface (GUI)* and can be used for free in the scope of research at universities. The source code of *OPNET* is closed source and, therefore, cannot be modified or extended. This limitation could lead to problems when implementing new protocols. Furthermore, the license of *OPNET* states that all results of the research, including papers and created models, have to be delivered to *Riverbed Technology* after completion and the company is allowed to terminate the license agreement at any time. This limits the usefulness of *OPNET* in the scope of a dissertation.

Another simulation environment is *J-Sim* [Hou] which is completely written in *Java* and, therefore, completely platform independent. The handicaps of *J-Sim* are that the last patch is from 2006 and that only a few protocol implementations are available. An implementation for *AODV* exists for example, but *OLSR* is missing.

Another java-based simulation environment is named *scalable ad hoc network simulator (ShoX)* [Sho]. It supports mobile wireless networks, provides a *GUI* and has tools to analyze simulation results. As the community using this simulator is much smaller compared to the communities of the previously mentioned widely used simulators, the last version of *ShoX* is from 2008 and only few tutorials are available, it is not examined more closely.

The simulators suited best for the validation task are *ns-2*, *ns-3*, *OMNeT++* and *GloMoSim*. *GloMoSim* is open source, has many implemented protocols and can scale with very large networks. A handicap of the simulator is that it can only simulate wireless networks. As the further development of *GloMoSim* was stopped it should not be used for new research tasks. Big advantages of *OMNeT++* are its simple use due to the *GUI* and the very good manual as well as the free use for non-commercial tasks. However, it is not as flexible as *ns-2* or *ns-3* to special simulation requirements. The development of *ns-2* started in 1989 and the long time led to a huge number of available protocols. Many of them were adapted to *ns-3*. Since *ns-3* is the successor of *ns-2*, it provides better performance and a more logical internal structure compared to *ns-2*. Therefore, it was selected as simulator for the validation of *SEREMA*.

7.2 ns-3-click

A further advantage of *ns-3* is that it can be combined with *Click* [MKJK99, KMC$^+$00, Koh01] which is a software router that allows the implementation of routing protocols without much effort. Furthermore, it enables reusing protocol implementations on real hardware. The combination of *ns-3* and *Click* is called *ns-3-click* and is used in this work for the validation of *SEREMA* with the option to run the adaptive framework on a real testbed.

7.2.1 Network Simulator ns-3

The *ns-3* is a discrete-event network simulator that is widely used in different research projects and allows the simulation of wired, wireless as well as hybrid networks. The software is available for free and as it is open source it can be modified and, therefore, the simulator is very flexible and expandable. The familiarization is made easy by the use of the manual and the tutorial [Conb]. For further knowledge, an *ns-3* mailing list [Conc] is provided.

This simulator was developed for the use on *Unix/Linux* operating systems. The protocols are implemented using *C++* and the simulation configurations are made using *C++* or *Python*. The simulation results can be simply retrieved as *packet capture (pcap)* files, as trace files as well as in various user defined formats. Therefore, the user can add different commands into the *C++* source code of the simulator to output the required information into text files or onto the screen. A powerful tool for this task is the tracing subsystem of *ns-3* that allows the definition of trace sources which in turn call user-defined functions when the considered events occur.

For the visualization, the simulator can be configured to generate an *Extensible Markup Language (XML)* based trace file that can be viewed with the tool *Network Animator (NetAnim)* after the simulation. It can be used to visualize node movements, transmission ranges as well as exchanged packets.

For the simulation of *MANETs*, *ns-3* can be provided with movement files. These files can be generated with *BonnMotion* [AEGPS12, Unia] or *setdest* [FV11] for example. An alternative for these tools is *HNMotion* which was developed in the scope of this dissertation and supports different groups of nodes as later used in the simulations. The

```
$node_(2) set X_ 229.26883899
$node_(2) set Y_ 556.53089050
$ns_ at 0.00000000 "$node_(2) setdest 284.11798878 582.61468796 2.14716165"
$ns_ at 33.28638465 "$node_(2) setdest 86.85552228 585.44751754 2.51640900"
$ns_ at 96.03509989 "$node_(2) setdest 283.60384719 319.39751581 2.21737022"
$ns_ at 210.75007081 "$node_(2) setdest 175.68602423 581.40108919 2.30717120"
$node_(3) set X_ 170.91370708
$node_(3) set Y_ 204.68684724
$ns_ at 0.00000000 "$node_(3) setdest 211.97591595 197.37375704 2.70444563"
$ns_ at 20.42214326 "$node_(3) setdest 286.01733853 120.50594221 2.23290455"
$ns_ at 89.28595262 "$node_(3) setdest 103.09148584 71.66409259 2.87420415"
```

Figure 7.1: An Example for an ns-3 Movement File

generated movement files of all tools have a structure as illustrated in figure 7.1. In the first two lines with the syntaxes `$<node> set X_ <x1>` and `$<node> set Y_ <y1>` node two is placed at the given $x1/y1$ location. Thereafter, line three configures a movement for node two. Node `<node>` is configured to start moving towards the position $x2/y2$ at the simulation time `<time>` seconds with a speed of `<speed>` m/s with the command `$ns at <time> "<node> setdest <x2> <y2> <speed>"`. All positions are defined in a unit of meters. When considering figure 7.1 the values for the simulation times, positions and speeds are striking since they are very precise with eight positions after the decimal point. The reason for this are the floating point values which are used in the mobility generators and are not rounded before the output. In the following part of this dissertation the values mentioned in the text will be rounded to two positions after the decimal point. The subsequent lines in figure 7.1 configure further movements for node 2 until the configuration lines for node 3 begin. In the example the commands are grouped by the node numbers. However, it is possible to provide the commands in a different order.

7.2.2 Click Modular Router

The *Click Modular Router* provides a flexible router framework to implement routing protocols without much effort and with the possibilities to simulate the protocols using *ns-2* or *ns-3* and to run the same protocol implementation on a real testbed. If *Click* is combined with *ns-2* or *ns-3* the resulting tool is named *ns-2-click*, *ns-3-click* or simply *nsclick*.

Each routing protocol is described in the *Click* script language that connects simple elements with each other to a graph. This graph describes all edges that can be used for packet forwarding between the single elements. Each of the elements provides a basic functionality and is written in *C++*. The standard *Click* installation provides a huge number of these elements [Unic], such as packet counters, packet queues, *IP* classifiers, or elements for decrementing the *TTL* for example. If further elements are required for a specific functionality, they can easily be added as separate *C++* files.

The *Click* router is located between the operating system kernel and the *Network interface Card (NIC)*. Therefore, the router processes all packets coming from the kernel as well as all packets received from the *NIC*. *Click* provides the elements *FromHost* to receive packets from the kernel and *ToHost* to send packets to the kernel. The connection towards the *NIC* is done by *FromDevice* and *ToDevice*. For the case that the router is part of an *nsclick* installation, the elements *FromSimDevice* and *ToSimDevice* are used for the communication with the network simulator.

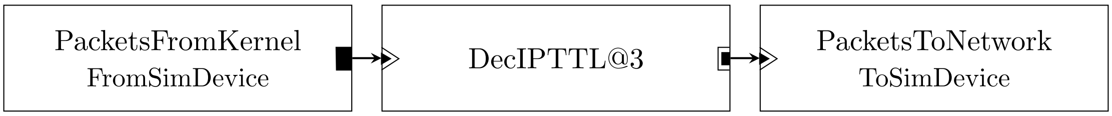


Figure 7.2: An Exemplary Click Graph

```
PacketsFromKernel::FromSimDevice(tap0)
-> DecIPTTL
-> PacketsToNetwork::ToSimDevice(eth0,IP);
```

Figure 7.3: The Click Script Corresponding to the Exemplary Click Graph

A *Click* graph can be visualized using the tool *Clicky* [Unid] that comes as part of the *Click* installation. This tool can be very helpful for locating errors in the routing graph. Figure 7.2 shows the visualized graph of the script presented in figure 7.3. The routing graph receives packets from the simulator (FromSimDevice) on *tap0* which is a virtual network device that provides *IP* packets coming from the operating system kernel, respectively the application layer in the simulation. The packets are forwarded to the element *DecIPTTL* which decrements the *TTL* in the *IP* header and in turn forwards the packets to *ToSimDevice*. This element delivers the packets to the simulator's *NIC* (eth0). As the instance of the element *FromSimDevice* is named *PacketsFromKernel* and the instance of *ToSimDevice* is called *PacketsToNetwork*, figure 7.2 shows these names beside the element type. The instance of *DecIPTTL* is named *DecIPTTL@3* as the user has not specified an own name.

During the implementation phase of a new routing graph the *Click* router can be executed in the *Userlevel* of the operating system. This allows fast modifications of the router but decreases the processing performance. If the router is supposed to be used on a real testbed, *Click* should be run in *Kernel Mode* to increase the performance because in this mode the packet processing mechanisms of the operating system are exchanged by the *Click* mechanisms. The drawback of this mode is that the kernel has to be re-compiled on each change.

A limitation of *nsclick* is caused by the interface between *ns-3* and *Click* as it does not support *IPv6* in the current version 3.19. The documentation [Cona] states:

> ”As of now, the ns-3 interface to Click is Ipv4 only. We will be adding Ipv6 support in the future.”

7.3 Protocol Implementations

The *Click* implementation of the *AODV* protocol in *SEREMA* is based on the work of Braem et al. [Braa] and was extended in the scope of this work. The *OLSR* protocol in *SEREMA* is also based on a work of Braem et al. [Brab] and was modified for the use in the presented adaptive framework.

7.4 Conclusion

This chapter compared different widely used network simulators for their suitability to validate the *SEREMA* implementation. The *ns-3* was selected as best choice and will be used for the simulations in chapter 8. The simulator as well as its combination with *Click* was briefly introduced and the benefit of the resulting *ns-3-click* environment was explained. Furthermore, the generation of movement files with the tools *BonnMotion*, setdest and *HNMotion* was introduced and a simple *Click* script with the corresponding *Click* graph was shown. The next chapter will validate the framework of *SEREMA* using *ns-3-click* as well as compare the performance of *SEREMA* to *AODV* and *OLSR*.

8 Validation

Having introduced *SEREMA* in chapter 5 and describing its functionality in detail in the subsequent chapter 6, the current chapter will validate the approach. In the first step this includes the validation of its behavior while in the second step the performance of *SEREMA* will be analyzed.

8.1 Behavioral Tests

Before the performance of *SEREMA* is examined, the behavior of the basic functionality including the *Border Nodes* has to be validated. This will be done in the following subsections with very basic network scenarios. To allow an easy following of the packet flow and thus, an easy validation as well as understanding of the protocol's functionality, static nodes are used. The simulation testbeds will be introduced and the simulation results will be discussed.

The following subsections consider only the packets that are essential for the validation of *SEREMA*. Further packets like *HELLOs* or *TCs* are not explicitly mentioned if they are not required for the validation process. The nodes in the scenarios are named X_Y where X denotes a node's mode of operation (A=*AODV*, O=*OLSR*, BN=*Border Node*) and Y represents the node's number in the scenario (cf. figure 8.1). The *IP* addresses of the nodes are configured as 10.1.1.Y which results in the *IP* address 10.1.1.2 for node A_2 for example.

8.1.1 Scenario 1 (AODV – OLSR)

As a first step the scenario shown in figure 8.1 where the reactive routing node A_1 wants to send data packets to the proactive routing node O_5 will be considered. The

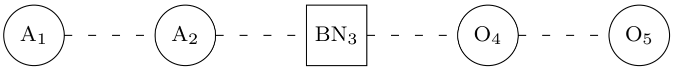


Figure 8.1: Behavioral Scenario 1 (AODV – OLSR)

nodes have a distance of 99 meters between each other and a transmission range of 100 meters, resulting in the case that a node can only communicate with its direct neighbors. During the simulation time of 100 seconds, node A_1 starts its transmissions at 45 seconds and stops it after ten seconds. This behavior was chosen to give *SEREMA* enough time to detect the border between the reactive and proactive routing part, to activate its *Border Node* functionality before node A_1 starts its transmissions and to give the nodes enough time to empty their buffers after the end of the transmissions. Node A_1 sends two packets per second with a size of 512 bytes each. The scenario is simulated using *ns-3-click*.

The simulation result shows a *Packet Delivery Ratio (PDR)* of 100 percent which means that all packets were successfully transmitted from node A_1 to O_5. In the following part of this subsection the packet flow will be analyzed.

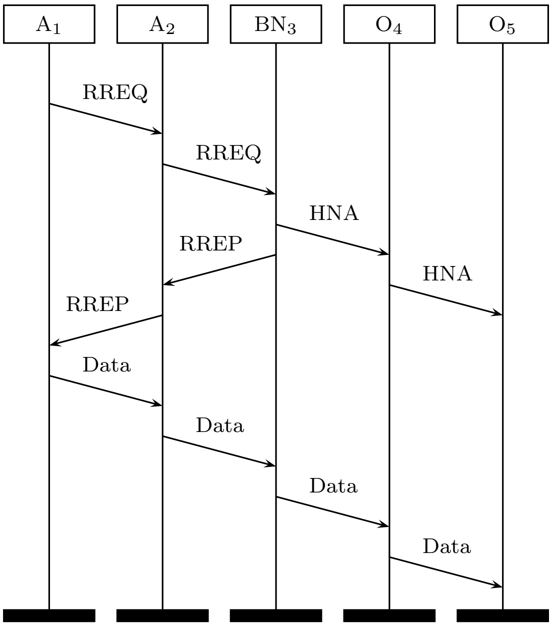


Figure 8.2: MSC for Scenario 1

Related to the functionality of *SEREMA* the resulting packet flow of this scenario should look like illustrated in the *Message Sequence Chart (MSC)* shown in figure 8.2.

If node A_1 wants to send data packets to node O_5 and does not have a route, it generates a *RREQ* to discover a route. The request propagates towards the *Border Node* BN_3. When BN_3 receives the *RREQ* it looks up a route to the destination node O_5 in its routing tables using its *RTW*. Since the *Border Node* is part of the proactive network it knows the destination O_5 and finds a route in its *OLSR* routing table. Therefore, BN_3 generates a *HNA* message to inform the proactive part of the network about the reverse route to node A_1 and it sends an *RREP* to the originator node A_1. When A_1 receives the *RREP* it learns the route to the destination O_5 and starts transmitting its data packets.

```
No.  Time       Source    Destination      Type  Info
37   36.295256  10.1.1.2  255.255.255.255  AODV  HELLO,
38   37.703000  10.1.1.1  255.255.255.255  AODV  HELLO,
39   38.394256  10.1.1.2  255.255.255.255  AODV  HELLO,
40   39.670000  10.1.1.1  255.255.255.255  AODV  HELLO,
41   40.478256  10.1.1.2  255.255.255.255  AODV  HELLO,
42   41.662000  10.1.1.1  255.255.255.255  AODV  HELLO,
43   42.476256  10.1.1.2  255.255.255.255  AODV  HELLO,
44   43.717000  10.1.1.1  255.255.255.255  AODV  HELLO,
45   44.545256  10.1.1.2  255.255.255.255  AODV  HELLO,
46   45.500000  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 2,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
48   45.820000  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 4,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
55   45.827223  10.1.1.2  10.1.1.1         AODV  RREP,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
60   45.829504  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
65   46.000000  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
68   46.500000  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.3: The Route Look-up and Data Transmission Start at Node A_1

The rest of this subsection will analyze the packet flow which was captured during the simulation run and validate the behavior of *SEREMA*. The captured packets on node A_1 are shown in figure 8.3. In the time span before the data transmissions start, node A_1 only emits *HELLO* messages (packets 37 to 45). As previously mentioned, the data transmissions are configured to begin 45 seconds after the simulation start. At 45.5 seconds the node tries to find a route to O_5 and generates a *RREQ* that can be seen as packet number 46. Special attention should be paid to the simulation time when the packet is sent. The scenario is configured to start the data transmission at 45.0

seconds but A_1 sends the packet 500 ms later. This can be clarified when taking a look at the *ns-3 OnOffApplication* that generates the traffic. The manual [Cone] states:

> ”Note: When an application is started, the first packet transmission occurs after a delay equal to (packet size/bit rate).”

As the simulation uses a packet size of 512 bytes which are 4096 bits and the bit rate is equal to 8192 bits/s this results in a delay of $\frac{4096 \text{ bits}}{8192 \text{ bits/s}} = 0.5$ seconds.

The next interesting packet is packet number 48 as it is a second *RREQ* initiated from node A_1. The reason of this packet is that the *TTL* of the first *RREQ* was limited to a value of two and, therefore, the destination could not be found. This limitation of the *TTL* is part of the *Expanding Ring Search* technique. The *TTL* in the second *RREQ* was increased to a value of four.

Subsequently, packet 55 contains the *RREP* and tells node A_1 the route to the destination node O_5. This *RREP* attracts attention because of two facts. The first one is the source address which shows a value of 10.1.1.2 but the packet was generated by the *Border Node* which has the address 10.1.1.3. The matter of this is that the packet was forwarded by node A_2 and, therefore, the source is 10.1.1.2. The second fact is that the *Destination* field of the *RREP* shows a value of 10.1.1.5 while the *Originator* field shows 10.1.1.1. When remembering the the structure of an *RREP* (cf. subsection 3.6.1) it becomes clear that a *RREP* contains the same values in its *Originator* and *Destination* field as the corresponding *RREQ*.

The following packets 60, 65 and 68 are *UDP* packets containing the application data. It can be seen that node A_1 operates completely reactive as described in the *AODV* specification [PBRD03]. A specialty of the *AODV* packets is their *THops* message extension that is transmitted with each *RREQ* and *RREP* packet (not shown in figure 8.3) and contains a value of zero as the considered example scenario does not utilize tunnels.

After analyzing the packet flow on node A_1, the following section will consider the functionality of the *Border Node* BN_3. In the time span before the period which is considered in figure 8.4, node BN_3 changes from its purely reactive routing mode to the *Border Node* mode because it detects node O_4 in its direct neighborhood. Figure 8.4 illustrates that only *HELLOs*, *TCs* and *BNANNOs* are processed before the *RREQ* from A_1 arrives at the *Border Node*. The first *RREQ* (cf. packet 116) is dropped

```
No.  Time       Source    Destination      Type  Info
100  39.829000  10.1.1.3  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.3
102  40.024000  10.1.1.3  255.255.255.255  OLSR  HELLO
103  40.478256  10.1.1.2  255.255.255.255  AODV  HELLO
104  40.639000  10.1.1.3  255.255.255.255  AODV  HELLO
105  41.849262  10.1.1.4  255.255.255.255  OLSR  HELLO
106  41.946000  10.1.1.3  255.255.255.255  OLSR  HELLO
107  42.476256  10.1.1.2  255.255.255.255  AODV  HELLO
108  42.548000  10.1.1.3  255.255.255.255  AODV  HELLO
109  43.850000  10.1.1.3  255.255.255.255  OLSR  HELLO
110  43.943262  10.1.1.4  255.255.255.255  OLSR  HELLO
111  44.502518  10.1.1.4  255.255.255.255  OLSR  TC,
                                                 Originator: 10.1.1.4,
                                                 Addresses: 10.1.1.3,
                                                 10.1.1.5
112  44.521000  10.1.1.3  255.255.255.255  AODV  HELLO
113  44.545256  10.1.1.2  255.255.255.255  AODV  HELLO
114  44.751000  10.1.1.3  255.255.255.255  OLSR  HELLO
116  45.500568  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 1,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
117  45.755000  10.1.1.3  255.255.255.255  OLSR  HELLO
118  45.820568  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 3,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
119  45.820618  10.1.1.3  255.255.255.255  OLSR  HNA, O: 10.1.1.3,
                                                 Addresses: 10.1.1.1
124  45.823548  10.1.1.3  10.1.1.2         AODV  RREP,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
134  45.841467  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
139  45.843388  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
143  45.855331  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.4: The Packet Processing at Node BN_3

because its *TTL* expired since BN_3 decremented the *TTL* of the received packet by one. The second *RREQ* arriving in packet number 118 is processed by the *Border Node* and makes BN_3 look up a route to O_5. As BN_3 is part of the proactive domain, it knows the route to the destination and, therefore, induces a *HNA* message into the proactive network to announce node A_1 to the *OLSR* nodes. Furthermore, it sends a *RREP* towards the originator A_1. The packets 134, 139 and 143 show that node A_1 accepts the route to the proactive destination and starts transmitting its data packets.

```
No.  Time       Source    Destination      Type  Info
64   41.021000  10.1.1.5  255.255.255.255  OLSR  HELLO
65   41.849262  10.1.1.4  255.255.255.255  OLSR  HELLO
66   43.057000  10.1.1.5  255.255.255.255  OLSR  HELLO
67   43.943262  10.1.1.4  255.255.255.255  OLSR  HELLO
68   44.502518  10.1.1.4  255.255.255.255  OLSR  TC,
                                                 Originator: 10.1.1.4,
                                                 Addresses: 10.1.1.3,
                                                 10.1.1.5
69   44.751562  10.1.1.4  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.3
70   44.751612  10.1.1.5  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.3
71   45.099000  10.1.1.5  255.255.255.255  OLSR  HELLO
72   45.821181  10.1.1.4  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.3,
                                                 Addresses: 10.1.1.1
78   45.855331  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
80   45.864262  10.1.1.4  255.255.255.255  OLSR  HELLO
82   46.020293  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
85   46.520293  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
88   47.020293  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.5: The Arrival of the First Data Packets at Node O_5

The reception of the first data packets on node O_5 is shown in figure 8.5. It can be seen that the node only exchanges *HELLOs* and *TCs* like in a usual proactive network. Additionally, it receives an *HNA* message from the *Border Node* to learn the reverse route to A_1. The only newly introduced message type is the *BNANNO* to advertise the *Border Node* functionality of node BN_3 to the proactive network. However, in the considered scenario this information is not used because no proactive node wants to connect to other routing domains. As node O_5 routes completely proactively it does not need to do anything, besides generating its *HELLOs* and *TCs*, in order to receive the data packets coming from node A_1.

This subsection showed that in scenario one *SEREMA* works as expected and that the communication from a reactive source node to a proactive destination node is possible.

8.1.2 Scenario 2 (OLSR – AODV)

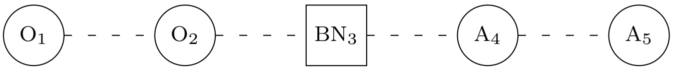


Figure 8.6: Behavioral Scenario 2 (OLSR – AODV)

The next scenario that should be validated is shown in figure 8.6. The proactive routing node O_1 wants to communicate with the reactive node A_5. The simulation parameters like the transmission ranges or the distances between the nodes are the same as in scenario one. The resulting *PDR* of 100 percent guarantees that no data packets were dropped.

The theoretical packet flow is presented in the *MSC* shown in figure 8.7. As soon as node BN_3 activates its *Border Node* functionality it periodically induces *BNANNO* messages into the proactive part of the network to announce its *Border Node* functionality. If the proactive source O_1 wants to send data packets to node A_5 it recognizes that it does not have a route. A usual proactive node would have a problem in this case as it cannot connect to the desired destination. However, since O_1 is equipped with *SEREMA* it activates its *Passive Border Node* mode and generates *Passive Border Node Annotation (PBNANNO)* messages to announce its new mode of operation. Subsequently, the node uses its newly activated *AODV* part to generate an *RREQ* for finding the destination A_5. As it was a goal of *SEREMA* to modify existing routing protocols only minimally, the *RREQ* is sent into the network as usual. However, as the neighbor in scenario two (node O_2) is a proactive routing node the *RREQ* is ignored. A copy of the *RREQ* is tunneled (IP-in-IP encapsulation) by *SEREMA* to the known *Border Node* which was learned by previously received *BNANNO* messages.

When the *Border Node* BN_3 receives the packet, it decapsulates the *RREQ* and forwards it into the reactive part of the network which results in a reception of the request on node A_4. Since node A_4 knows its direct neighbor A_5 it generates two *RREPs*. The first reply is a *Gratuitous RREP* which informs node A_5 about the reverse route to O_1, the second *RREP* is sent back towards the originator node O_1. When the *Border Node* BN_3 receives the reply, it firstly induces an *HNA* message into the *OLSR* domain

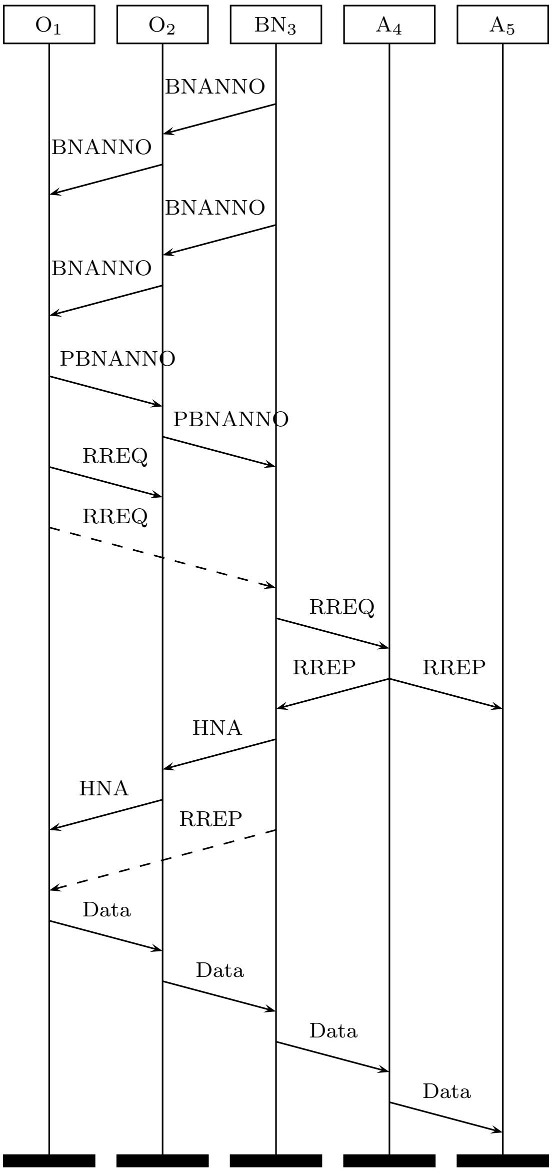


Figure 8.7: The MSC for Scenario 2

to inform the nodes about node A_5 and, secondly, tunnels the *RREP* to the originator node O_1. When node O_1 receives the *RREP* it starts transmitting its data packets to the destination node A_5.

```
No.  Time       Source    Destination      Type  Info
65   40.995000  10.1.1.1  255.255.255.255  OLSR  HELLO
67   42.914000  10.1.1.1  255.255.255.255  OLSR  HELLO
70   44.869000  10.1.1.1  255.255.255.255  OLSR  HELLO
71   45.425562  10.1.1.2  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.3
75   45.500848  10.1.1.1  255.255.255.255  OLSR  PBNANNO,
                                                 Originator: 10.1.1.1
77   45.501961  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 2,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1
81   45.504467  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 2,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1,
                                                 IP−in−IP
                                                 (10.1.1.1 → 10.1.1.3)
86   45.516897  10.1.1.2  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.3,
                                                 Addresses: 10.1.1.5
92   45.522619  10.1.1.3  10.1.1.1         AODV  RREP,
                                                 Destination: 10.1.1.5,
                                                 Originator: 10.1.1.1,
                                                 IP−in−IP
                                                 (10.1.1.3 → 10.1.1.1)
94   45.523143  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
96   45.533108  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
97   46.000000  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.8: The Route Look-up and Data Transmission Start at Node O_1

The following paragraphs will analyze the captured packet flows and validate the functionality of *SEREMA*. Figure 8.8 shows that node O_1 only generates *OLSR* messages (packets 65–70) before the data transmission starts. Packet number 71 is a *BNANNO* originated by BN_3 and forwarded by node O_2 (consider the source *IP* address with a value of 10.1.1.2). This packet informs O_1 about the *Border Node* functionality of node BN_3.

When O_1 starts its data transmission it needs a route to A_5. As the *OLSR* routing table of node O_1 does not contain a route to the *AODV* destination, node O_1 enters its *Passive Border Node* mode and generates a *PBNANNO* message (packet number 75) to inform the other nodes about its new functionality. In the next step the node induces an *RREQ* for the destination node A_5. In the current implementation of *SEREMA* this request is broadcasted to the node's direct neighbors and additionally tunneled to all *Border Nodes* known by the proactive routing area. As the request which is broadcasted to the neighbors usually makes no sense because the neighborhood of the proactive node contains no reactive nodes, this direct request should be removed from *SEREMA* in the future. In the current implementation this request is still present because the standard *AODV* implementation generates this packet and it is a fundamental requirement of this work to modify the existing protocols as little as possible.

With packet number 86 node O_1 receives the *HNA* message induced by the *Border Node* BN_3 to inform the proactive *OLSR* network about the route to node A_5. Subsequently, packet number 92 contains the *RREP* coming via a tunnel from BN_3 to O_1. This reply is required to process the packets that are buffered by the reactive protocol part of node O_1 (remember that node O_1 entered its *Passive Border Node* mode and, therefore, operates *OLSR* and *AODV* simultaneously). Afterwards, the source O_1 is able to transmit its data packets 94, 96 and 97 to the destination node A_5.

When analyzing the *RREQ* (packet 81) in more detail, its *TTL* attracts attention because it is set to a value of two by the originator and still reaches the destination. This is possible as the packet's *TTL* is not decremented by the tunnel hops and, therefore, the first *TTL* reduction is done on BN_3 which in turn forwards the packet to A_4. This node receives a *TTL* of one and is able to answer the request because the destination node is its direct neighbor. This behavior of *SEREMA* was chosen as the *AODV* hop count in the current implementation does not consider tunnel hops because they are transmitted separately inside the *THops* field. Remember, the *THops* are used to correct the hop count at the tunnel endpoints. When the behavior of the *THops* is optimized in a future work as described in subsection 6.7.3 the behavior of the *TTL* should also be modified.

The processing of the *RREQ* at node BN_3 is shown in figure 8.9. In the period before node O_1 starts its data transmission, node BN_3 only exchanges *HELLOs*, *TCs* and *BNANNOs*. With packet number 118 the *Border Node* receives the *PBNANNO* message from node O_1 and processes it. This is required to enable the tunneling of reactive routing messages like *HELLOs* or *RERRs* from the *Border Node* to the *Passive Border Node* for *AODV* route maintenance. When it receives the tunneled packet number 125 which contains the *RREQ* from O_1, it searches a route in its routing tables. However,

```
No.  Time       Source    Destination      Type  Info
105  40.775262  10.1.1.2  255.255.255.255  OLSR  HELLO
106  41.745000  10.1.1.3  255.255.255.255  OLSR  HELLO
107  42.208256  10.1.1.4  255.255.255.255  AODV  HELLO
108  42.634000  10.1.1.3  255.255.255.255  AODV  HELLO
109  42.856262  10.1.1.2  255.255.255.255  OLSR  HELLO
110  43.839000  10.1.1.3  255.255.255.255  OLSR  HELLO
111  44.092518  10.1.1.2  255.255.255.255  OLSR  TC,
                                                  Originator: 10.1.1.2,
                                                  Addresses: 10.1.1.1,
                                                  10.1.1.3
112  44.287256  10.1.1.4  255.255.255.255  AODV  HELLO
113  44.710000  10.1.1.3  255.255.255.255  AODV  HELLO
114  44.806262  10.1.1.2  255.255.255.255  OLSR  HELLO
115  45.425000  10.1.1.3  255.255.255.255  OLSR  BNANNO,
                                                  Originator: 10.1.1.3
116  45.425562  10.1.1.2  255.255.255.255  OLSR  BNANNO,
                                                  Originator: 10.1.1.3
117  45.500558  10.1.1.2  255.255.255.255  OLSR  TC,
                                                  Originator: 10.1.1.1,
                                                  Addresses: -
118  45.501411  10.1.1.2  255.255.255.255  OLSR  PBNANNO,
                                                  Originator: 10.1.1.1
119  45.501461  10.1.1.3  255.255.255.255  OLSR  PBNANNO,
                                                  Originator: 10.1.1.1
125  45.508366  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 2,
                                                  Destination: 10.1.1.5,
                                                  Originator: 10.1.1.1,
                                                  IP-in-IP
                                                  (10.1.1.1 → 10.1.1.3)
127  45.509210  10.1.1.3  255.255.255.255  AODV  RREQ, TTL: 1,
                                                  Destination: 10.1.1.5,
                                                  Originator: 10.1.1.1
134  45.515491  10.1.1.4  10.1.1.3         AODV  RREP,
                                                  Destination: 10.1.1.5,
                                                  Originator: 10.1.1.1
136  45.516335  10.1.1.3  255.255.255.255  OLSR  HNA,
                                                  Originator: 10.1.1.3,
                                                  Addresses: 10.1.1.5
141  45.518725  10.1.1.3  10.1.1.1         AODV  RREP,
                                                  Destination: 10.1.1.5,
                                                  Originator: 10.1.1.1,
                                                  IP-in-IP
                                                  (10.1.1.3 → 10.1.1.1)
147  45.533108  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                  Destination port: 9999
152  45.534890  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                  Destination port: 9999
154  45.544854  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                  Destination port: 9999
```

Figure 8.9: The Processing of the RREQ at Node BN_3

as it does not know a route to the destination node A_5 it forwards the *RREQ* into the reactive routing area.

Afterwards, the *Border Node* receives an *RREP* from node A_4 (packet 134), injects an *HNA* message into the proactive routing domain to inform the nodes about node A_5 and tunnels the *RREP* to the originator O_1. Subsequently, node O_1 starts transmitting its data packets (packets 147, 152 and 154) to the destination node A_5.

```
No.  Time       Source    Destination      Type  Info
40   38.183256  10.1.1.4  255.255.255.255  AODV  HELLO
41   39.660000  10.1.1.5  255.255.255.255  AODV  HELLO
42   40.210256  10.1.1.4  255.255.255.255  AODV  HELLO
43   41.732000  10.1.1.5  255.255.255.255  AODV  HELLO
44   42.208256  10.1.1.4  255.255.255.255  AODV  HELLO
45   43.787000  10.1.1.5  255.255.255.255  AODV  HELLO
46   44.287256  10.1.1.4  255.255.255.255  AODV  HELLO
51   45.512522  10.1.1.4  10.1.1.5         AODV  RREP,
                                                 Destination: 10.1.1.1,
                                                 Originator: 10.1.1.5
57   45.544854  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
59   45.771000  10.1.1.5  255.255.255.255  AODV  HELLO
61   46.020293  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
63   46.291256  10.1.1.4  255.255.255.255  AODV  HELLO
65   46.520293  10.1.1.1  10.1.1.5         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.10: The Arrival of the First Data Packets at Node A_5

As the destination A_5 is not in direct contact with a proactive neighbor it only communicates reactively (cf. figure 8.10). The node exchanges *HELLO* packets with its neighbor node A_4. With packet number 51 it receives a *Gratuitous RREP* generated from node A_4 because A_4 answered the request coming from node O_1. Directly after this *Gratuitous RREP*, node A_5 receives the first data packet (packet 57) coming from node O_1. It can be seen that the destination node A_5 operates purely reactively and is able to receive packets from a proactive source node.

This scenario validated that a proactive source node is able to communicate with a reactive destination node when using *SEREMA* and that the routing via a *Border Node* works as expected.

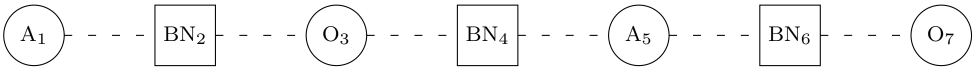


Figure 8.11: Behavioral Scenario 3 (AODV – OLSR – AODV – OLSR)

8.1.3 Scenario 3 (AODV – OLSR – AODV – OLSR)

After the validation of simple scenarios that only contain one *Border Node*, this subsection considers the scenario shown in figure 8.11 with three *Border Nodes*. The reactive source node A_1 communicates with the proactive destination O_7 via two intermediate routing domains. The simulation setup uses the same parameters as described previously for the scenarios one and two. The *PDR* showed again a success rate of 100 percent.

The *MSC* in figure 8.12 presents the route discovery procedure of *SEREMA* for scenario three. All *Border Nodes* induce *BNANNO* messages into their proactive network parts to inform *OLSR* nodes about their special functionality. If the *AODV* node A_1 wants to transmit data packets to node O_7 it generates an *RREQ* which is received by the *Border Node* BN_2. The *Border Node* recognizes that a route to the destination O_7 is neither available in its reactive nor in its proactive routing table and, therefore, it tunnels the packet to the other *Border Nodes* (node BN_4 in scenario three) of its proactive routing domain. When node BN_4 receives the packet it decapsulates the *RREQ*, checks its routing tables via the *RTW* and forwards the request into its reactive routing area since it has no route available. The request propagates via node A_5 to BN_6 which looks up the route to the destination O_7 in its routing tables and finds a possible route as O_7 is part of the proactive network. Therefore, node BN_6 generates an *HNA* message to inform its proactive routing area about node A_1 and, subsequently, it sends an *RREP* back towards the originator node A_1. When the *RREP* is received by the intermediate *Border Node* BN_4, an *HNA* message is generated to inform the proactive network area of BN_4 about the route to node O_7. Afterwards, the *RREP* is tunneled to BN_2 which in turn decapsulates it and forwards it to the originator node A_1. Subsequently, A_1 starts the data transmission to the destination O_7.

As the behavior of the source and destination nodes was validated in the previous scenarios, this subsection concentrates on the packet forwarding of the intermediate nodes, especially the *Border Nodes*.

Figure 8.13 shows the behavior of *Border Node* BN_2 which is informed about the *Border Node* functionality of node BN_4 in packet number 245. Packet 247 contains the *RREQ*

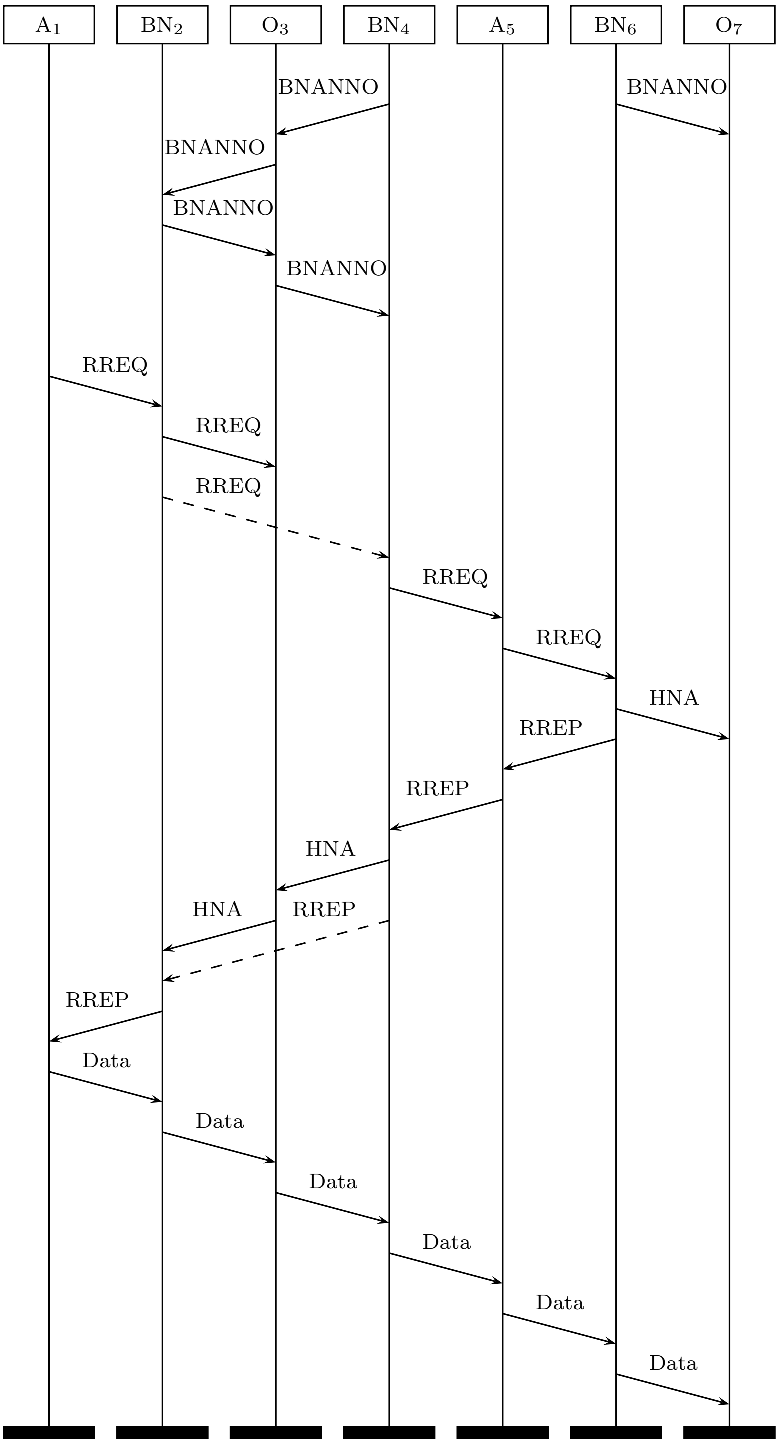


Figure 8.12: MSC for Scenario 3

```
No.  Time       Source    Destination      Type  Info
245  45.201562  10.1.1.3  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.4
247  45.500259  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 2,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
248  45.500309  10.1.1.2  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.2,
                                                 Addresses: 10.1.1.1
250  45.501081  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 1,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
251  45.501770  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 1,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.2 → 10.1.1.4)
254  45.593000  10.1.1.2  255.255.255.255  OLSR  BNANNO,
                                                 Originator: 10.1.1.2
256  45.820259  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 4,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
257  45.820309  10.1.1.2  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.2,
                                                 Addresses: 10.1.1.1
259  45.820941  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 3,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
260  45.821850  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 3,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.2 → 10.1.1.4)
265  46.268178  10.1.1.4  255.255.255.255  AODV  HELLO,
                                                 IP-in-IP
                                                 (10.1.1.4 → 10.1.1.2)
267  46.278000  10.1.1.2  255.255.255.255  AODV  HELLO
268  46.278346  10.1.1.2  255.255.255.255  AODV  HELLO,
                                                 IP-in-IP
                                                 (10.1.1.2 → 10.1.1.4)
271  46.300259  10.1.1.1  255.255.255.255  AODV  RREQ, TTL: 6,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
272  46.300309  10.1.1.2  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.2,
                                                 Addresses: 10.1.1.1
274  46.301021  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 5,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
```

Figure 8.13: Packet Processing at Border Node BN_2 (part 1/2)

coming from node A_1 and provokes the *Border Node* BN_2 to generate an *HNA* message to inform the nodes in its proactive routing area about A_1. Afterwards, the *Border Node* forwards the *RREQ* to other reactive nodes (packet number 250) as well as to other *Border Nodes* (packet number 251). Packet 254 is used to disclose the *Border Node* functionality of node BN_2 to other nodes.

As the first *RREQ* from node A_1 was not able to reach the destination because of its limited *TTL*, node A_1 generates a second request with a *TTL* set to 4. This request is processed on the *Border Node* (packets 256–260) as described previously for the first *RREQ*.

The packets 265, 267 and 268 illustrate the exchange of reactive *HELLO* messages between the nodes. *Border Node* BN_2 receives a tunneled *HELLO* from *Border Node* BN_4 in packet 265 and induces an own *HELLO* in packet 267. This *HELLO* is additionally tunneled to *Border Node* BN_4 in packet number 268.

Similar to the first *RREQ* of node A_1, the second *RREQ* was not able to reach the destination either. Therefore, *Border Node* BN_2 receives a third *RREQ* in packet 271, induces an *HNA* message (packet number 272) into its proactive routing area, forwards the request to its reactive neighbors (packet number 274) and tunnels it to other *Border Nodes* (packet number 275 (cf. figure 8.14)).

With packet 278 (cf. figure 8.14) the *Border Node* BN_2 receives a *HNA* from node BN_4 with route information for the destination node O_7. Subsequently, BN_2 receives the *RREP* for the requested destination (packet number 280) and forwards it to the originator (packet number 285). The following packets (packet numbers 290, 292 and 294) are the first data packets coming from A_1 and destined for O_7.

The packets shown in figure 8.15 were processed on node BN_4 and concentrate only on the third *RREQ* of node A_1 which is received as packet number 271. As BN_4 received the request via a tunneled connection from its proactive routing area, it decapsulates the packet and forwards the request into its reactive routing domain (packet 273).

When BN_4 receives the *RREP* (packet number 280) it injects an *HNA* message into its proactive routing domain (packet number 282) and tunnels the reply to node BN_2 on the reverse path (packet number 284). Subsequently, the data transmission from node A_1 to O_7 starts as shown in the packets 288, 293 and 297.

```
No.  Time       Source    Destination      Type  Info
275  46.301710  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 5,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.2 → 10.1.1.4)
278  46.311512  10.1.1.3  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.4,
                                                 Addresses: 10.1.1.7
280  46.313107  10.1.1.4  10.1.1.2         AODV  RREP,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.4 → 10.1.1.2)
285  46.315129  10.1.1.2  10.1.1.1         AODV  RREP,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
290  46.322887  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
292  46.328312  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
294  46.328976  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.14: The Packet Processing at Border Node BN_2 (part 2/2)

For the last part of the route *Border Node* BN_6 maintains the transition between the reactive and the proactive routing domains. When the *Border Node* receives packet number 110 which contains the *RREQ* it sends an *HNA* message (packet number 111) into the proactive routing area (to inform node O_7 about the route to A_1). Furthermore, BN_6 generates an *RREP* (packet number 115) that is sent on the reverse path to node A_5 which in turn forwards the packet towards the originator node A_1. When the reply reaches the data transmission source A_1, the data transfer starts.

This scenario validated the behavior of the *Border Nodes* which interconnect multiple routing areas with each other and showed that the *Border Nodes* behave as expected.

8.1.4 Scenario 4 (OLSR – AODV – OLSR – AODV)

This behavioral scenario considers the communication from the proactive routing node O_1 to the reactive routing node A_8 via two intermediate routing areas. The setup of the simulation environment is similar to the previously described scenarios and, therefore, not described in detail here. The scenario which showed a *PDR* of 100 percent is visualized in figure 8.17.

```
No.  Time       Source    Destination      Type  Info
269  46.300871  10.1.1.3  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.2,
                                                 Addresses: 10.1.1.1
271  46.302845  10.1.1.2  255.255.255.255  AODV  RREQ, TTL: 5,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.2 → 10.1.1.4)
273  46.303209  10.1.1.4  255.255.255.255  AODV  RREQ, TTL: 4,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
280  46.310046  10.1.1.5  10.1.1.4         AODV  RREP,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
282  46.310950  10.1.1.4  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.4,
                                                 Addresses: 10.1.1.7
284  46.312022  10.1.1.4  10.1.1.2         AODV  RREP,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1,
                                                 IP-in-IP
                                                 (10.1.1.4 → 10.1.1.2)
288  46.331400  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
293  46.333582  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
297  46.345124  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.15: Packet Processing at Border Node BN_4

```
No.  Time       Source    Destination      Type  Info
110  46.303778  10.1.1.5  255.255.255.255  AODV  RREQ, TTL: 3,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
111  46.303828  10.1.1.6  255.255.255.255  OLSR  HNA,
                                                 Originator: 10.1.1.6,
                                                 Addresses: 10.1.1.1
115  46.306131  10.1.1.6  10.1.1.5         AODV  RREP,
                                                 Destination: 10.1.1.7,
                                                 Originator: 10.1.1.1
125  46.345124  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
130  46.347326  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
132  46.362867  10.1.1.1  10.1.1.7         UDP   Source port: 49153,
                                                 Destination port: 9999
```

Figure 8.16: Packet Processing at Border Node BN_6

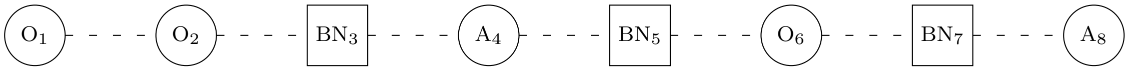


Figure 8.17: Behavioral Scenario 4 (OLSR – AODV – OLSR – AODV)

As the route discovery process in scenario four is a combination of previously validated mechanisms, the discussion of captured packets is avoided and, instead, only the summarized packet flow is described, based on the *MSC* shown in figure 8.18.

In the time span before node O_1 wants to start its data transmission, the *Border Nodes* BN_3, BN_5 and BN_7 broadcast their *BNANNOs* into the proactive routing domains. When O_1 wants to send its data packets and does not know a route to the destination node A_8, it activates its *Passive Border Node* mode and induces *PBNANNO* messages into the network. Subsequently, O_1 utilizes its reactive routing part and generates an *RREQ* that is broadcasted to the neighbor nodes, which is the usual behavior of a standard *AODV* node. Furthermore, it tunnels a copy of the *RREQ* to *Border Node* BN_3 which disclosed its *Border Node* functionality via *BNANNO* messages.

The *Border Node* BN_3 decapsulates the packet and forwards the *RREQ* into the reactive routing area. When the request arrives at node BN_5, an *HNA* message for the destination node O_1 is injected into the proactive area, an *RREQ* is broadcasted to the direct neighbor nodes (the behavior of a standard *AODV* implementation) and, additionally, the *RREQ* is tunneled through the proactive routing domain to *Border Node* BN_7.

Since BN_7 is a direct neighbor of the destination node A_8, it can answer the request because it knows its direct neighbor nodes. Therefore, it sends an *HNA* message into its proactive routing area to tell the nodes the route to A_8. Furthermore, it informs A_8 about the path to O_1 by sending a *Gratuitous RREP* and tunnels an *RREP* on the reverse path back to BN_5 which in turn forwards the decapsulated *RREP*. When the reply arrives at the *Border Node* BN_3, an *HNA* message is generated to inform the proactive nodes about the route to A_8 and, additionally, the *RREP* is tunneled to the originator node O_1.

Directly after receiving the *RREP*, node O_1 starts transmitting the packets that were buffered by its *AODV* implementation (the *AODV* protocol of the node buffered the packets when entering the *Passive Border Node* mode) as well as forwarding further packets directly, hence its *OLSR* routing table contains from now on a route to the destination node A_8.

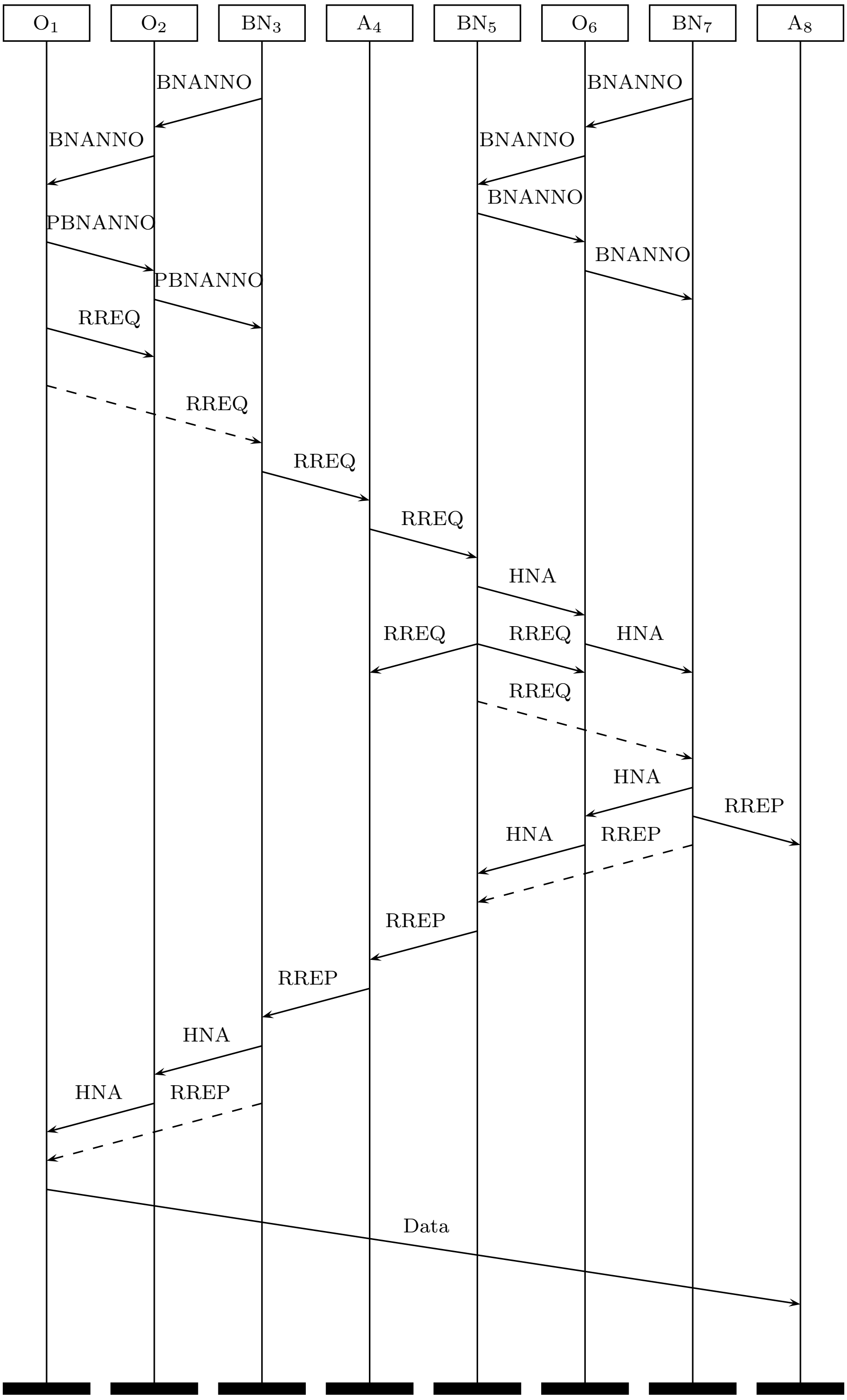


Figure 8.18: MSC for Scenario 4

8.2 Performance Tests

The behavior of *SEREMA* was analyzed in section 8.1 and it was shown that the route discovery procedure over multiple different routing domains operates as expected. The current section deals with the comparison of the performance between *AODV*, *OLSR* and *SEREMA* in different *MANET* scenarios.

Section 5.1 mentioned that the purely local monitoring of the nodes in *SEREMA* could have a negative influence on the decision making. First simulation runs containing nodes that were allowed to completely select their routing protocol on their own showed that a lot of borders between reactive and proactive routing areas occurred and, therefore, nearly all of the nodes acted as *Border Node*. This resulted in an extremely large routing overhead and a decreased *PDR*. To overcome this problem in the following simulation runs, the nodes will not be allowed to select their routing protocol on their own. Instead, they are forced to a given main routing protocol. However, an *AODV* node is allowed to activate its *Border Node* functionality if it detects an *OLSR* node in its direct neighborhood.

The diagrams discussed in the rest of this chapter are based on the data which was gathered by simulation runs and will be presented in chapter D.

8.2.1 Scenario 1 (AODV – OLSR)

The first performance scenario that will be considered is shown in figure 8.19 and represents a scenario that can benefit from multiple simultaneous routing protocols using the *SEREMA* framework. The scenario consists of three different node configurations as illustrated in table 8.1.

8.2.1.1 Simulation Setup

Simulated Protocol	Routing Domain 1	BN	Routing Domain 2
AODV	AODV	AODV	AODV
OLSR	OLSR	OLSR	OLSR
SEREMA	AODV	BN	OLSR

Table 8.1: Performance Scenario 1

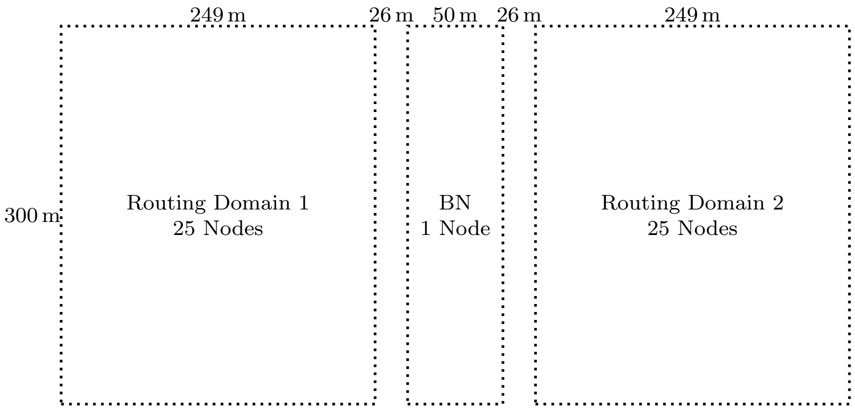


Figure 8.19: The Network Constellation for Performance Scenario 1 (AODV – OLSR)

The scenario contains 25 nodes in each routing domain and one additional node in between that acts as *Border Node*. The two routing domains are separated by a distance of $26\,\mathrm{m} + 50\,\mathrm{m} + 26\,\mathrm{m} = 102\,\mathrm{m}$ while the transmission range of a single node is limited to 100 m to avoid a direct communication between the nodes of routing domain one and two. Therefore, all communication between routing domain one and routing domain two has to be conducted via the *Border Node*. The overall horizontal dimension of the scenario results in a length of $249\,\mathrm{m} + 26\,\mathrm{m} + 50\,\mathrm{m} + 26\,\mathrm{m} + 249\,\mathrm{m} = 600\,\mathrm{m}$.

Furthermore, the *Average Node Degree* which describes the average number of neighbors per node should be considered. To guarantee a good connectivity between the nodes, the *Average Node Degree* should be larger than one and on the other side, not too large since a high connectivity increases the routing overhead and congests the network.

To estimate the average number of neighbors per node, figure 8.20 illustrates one area of scenario one (cf. figure 8.19) with nodes distributed in a grid of 5×5 and the dimensions of 249 m and 300 m. The horizontal dimension was divided into four parts that represent the distances between the nodes, calculated as $\frac{249\,\mathrm{m}}{4} = 62.25\,\mathrm{m}$. The same calculation for the vertical distances results in $\frac{300\,\mathrm{m}}{4} = 75.00\,\mathrm{m}$. The diagonal connections between the nodes have a length of $\sqrt{62.25\,\mathrm{m}^2 * 75.00\,\mathrm{m}^2} = 97.468\,\mathrm{m}$. Since the transmission range of a node is configured to be 100 m, a node can only communicate with its direct neighbors, independent from the type of neighbor (horizontal, vertical or diagonal). The numbers shown in figure 8.20 represent the resulting number of neighbors the node can directly reach.

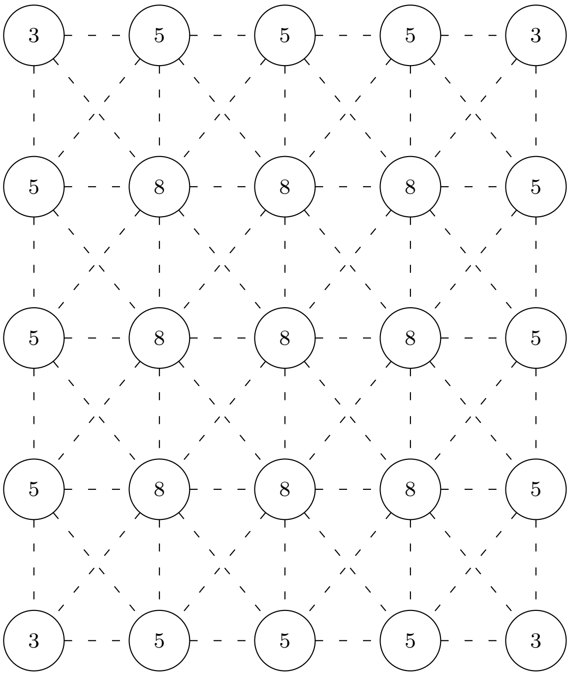


Figure 8.20: Uniformly Distributed Nodes in One Part of Scenario 1 with the Dimensions 249 m x 300 m and Visualized Connections

The *Average Node Degree* $\bar{d}$ can be calculated by $\bar{d} = \frac{1}{n} * \sum_{i=1}^{n} d_i$ where n represents the number of nodes and d_i the number of neighbors of node i. When considering the values shown in figure 8.20 this results in an *Average Node Degree* of $\bar{d} = 5.76$ and, therefore, in the fact that a node can directly communicate with $\frac{\bar{d}}{n} * 100$ percent of the nodes in its routing domain which results in $\frac{5.76\,\text{nodes}}{25\,\text{nodes}} * 100 = 23.04$ percent for the given network scenario.

In *MANET* simulations special attention should be paid to the applied mobility model. The behavior of *MANETs* and, therefore, the performance is strongly dependent on the node movements. To cope with a lot of different network constellations, much effort was spent for the development of suitable mobility models. A widely used model is the *Manhattan Grid Model* [Eur97] where the nodes are only allowed to move on paths between blocks comparable to the structure of the streets in Manhattan. However, this scenario only applies to disasters where the nodes are limited to move on the paths between such blocks. The rescue missions after the 9/11 terror attacks could partially be modeled by the *Manhattan Grid Model*. If other disaster scenarios like rescue missions after tsunamis are considered, the *Manhattan Grid Model* is not suitable as the nodes have a more or less free area to move. Further mobility models aimed at past disasters which happened in Germany were analyzed by Krug et al. [KSS^{+}14].

As the performance comparisons between *AODV*, *OLSR* and *SEREMA* should not be based on any special disaster scenarios, the simulations use the *Random Waypoint Model* which randomly moves the nodes within restricted areas. The speed of the nodes was selected to be in the range of one to three meters per second (3.6 km/h to 10.8 km/h) and represents victims as well as different kinds of walking response teams.

Based on the current knowledge, *BonnMotion* does not support the generation of movement files consisting of different areas like shown in figure 8.19. Therefore, the new movement file generator *HNMotion* was developed in the scope of this work to generate the movements for the performance simulations. The movements for scenario one were created with *HNMotion* using the commands shown in figure 8.21.

```
hnmotion −x 50 −y 300 –X 275 –Y 0 −n 1 –N 0 −d 300 −l 1 −h 3 –R 1 −p ↩
5 > scenario1.ns_movements
hnmotion −x 249 −y 300 –X 0 –Y 0 −n 25 –N 1 −d 300 −l 1 −h 3 –R 2 −p ↩
5 >> scenario1.ns_movements
hnmotion −x 249 −y 300 –X 351 –Y 0 −n 25 –N 26 −d 300 −l 1 −h 3 –R 3 ↩
−p 5 >> scenario1.ns_movements
```

Figure 8.21: HNMotion Commands for Creating the Mobility File Used in Scenario 1

The values passed to the parameters x and y specify the size of the considered area while X and Y allow to set an offset to the node positions. This is used to move the nodes into the desired area of the simulation environment. The same procedure is used for the number of nodes. While n configures the number of nodes in the scenario, the parameter N adds an offset to the node numbers to allow the combination of multiple created movement patterns. The duration of the simulation is set to 300 seconds by the parameter d and the required node speeds are given by l (low) and h (high) in meters per second. To affect the randomness of the scenarios generated by *HNMotion*, the *Random Number Generator (RNG)* can be seeded by the value of the parameter R. Parameter p specifies the pause time of the nodes between movements. The output of *HNMotion* is sent to *stdout* by default. The lines of code above use the *output redirector* (>) as well as the *append redirector* (>>) to send respectively append the output to the file *scenario1.ns_movements*.

The simulation uses different numbers of active data transmissions between the nodes, ranging from 30 to 100 with a step size of ten. Every source node generates a 512 byte packet per second, starting 50 seconds after the begin of the simulation to allow the network to synchronize the proactive part as well as activate the *Border Node* functionality and ending 50 seconds before the simulation end to allow the nodes to empty their buffers. To profit the most from the adaptive routing the network scenario should be configured to bring *AODV* and *OLSR* into situations where they cannot perform well. Therefore, the scenario uses only three internal data transmissions for routing domain one to simulate light traffic, one data transmission from routing domain one to domain two and one connection vice versa. The remaining data transmissions whose number is calculated by the number of all data transmissions minus the number of transmission from routing domain one to routing domain two and minus the number of transmissions from domain two to domain one take place in routing domain two and simulate high traffic. The mentioned configuration of the data transmissions is illustrated in table 8.2 and results in the fact that *AODV* is predestined for routing domain one because of the light traffic while *OLSR* outperforms *AODV* in domain two as *AODV* generates huge routing overhead. Each measurement result of the simulation is averaged over ten simulation runs.

Source	Destination	Number of Active Data Transmissions
Routing Domain 1	Routing Domain 1	3
Routing Domain 1	Routing Domain 2	1
Routing Domain 2	Routing Domain 1	1
Routing Domain 2	Routing Domain 2	$N - 5$

Table 8.2: Configuration of the Data Transmission in Performance Scenario 1 (N is the Number of All Active Data Transmissions in the Simulation)

8.2.1.2 Simulation Results

The previously introduced scenario is simulated with pure *AODV* nodes, with pure *OLSR* nodes and for the simulation with *SEREMA* routing domain one uses *AODV* while domain two uses *OLSR*. Figure 8.22 (based on the data in table D.3) shows the number of sent as well as received data packets. The number of sent data packets follows the number of active data transmissions which is shown on the axis of abscissa and increased from left to right. The number of received data packets for *AODV*, *OLSR* and *SEREMA* follows the number of sent data packets as expected since an increased number of sent data packets leads to an increased number of received packets if routes are available and the network is not congested. However, a *MANET* is usually not able to deliver all packets due to collisions, the limited *bandwidth* or unavailable routes for example, which results in a number of received packets that is lower than the number of sent packets. If the network is stressed in terms of heavily increasing traffic, the maximum available bandwidth of the network will be reached. This limits

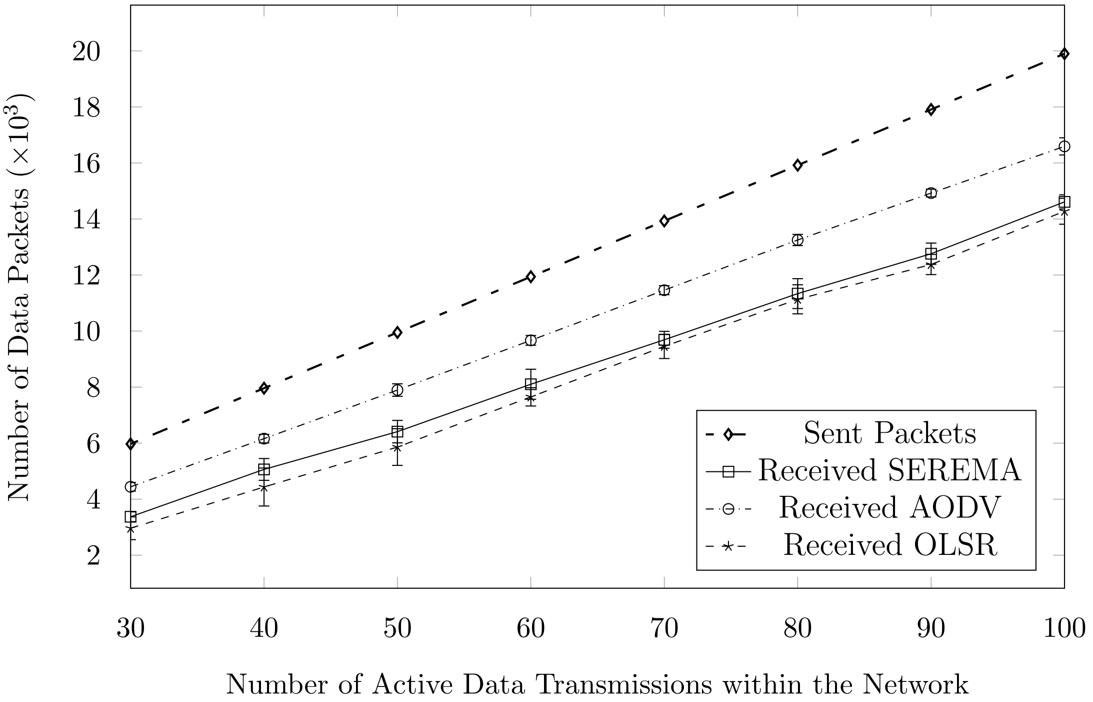


Figure 8.22: Comparison between the Numbers of Received Data Packets (Related to the Number of Sent Data Packets) for Performance Scenario 1

the maximum number of received packets. In the considered simulation the network was not stressed to this limit.

Furthermore, figure 8.22 shows that the gradients of the *AODV*, *OLSR* and *SEREMA* graphs are lower than the gradient of the graph for the sent data packets. This results in an increasing difference between the numbers of sent and received data packets when enlarging the traffic in the network. This is visualized in figure 8.23 which is based on the data in table D.4.

When considering the resulting *PDR* that is calculated as the ratio of received data packets to the number of sent data packets (PDR = ${}^{N_{Rx}}/_{N_{Tx}} * 100$) and represented by a percentage value in figure 8.24 (cf. data in table D.5), it can be seen that the graphs are rising in the region of a low number of active data transmissions while their gradients decline when the number of active data transmissions increases. This behavior is as expected since in a network with a low exploitation of the *bandwidth* the ratio of received data packets increases together with the number of transmitted data packets and if the exploitation of the network's *bandwidth* increases, the gap between the numbers of sent and received data packets enlarges, resulting in a decline of the *PDR*. If the generated traffic is further increased, the resulting gradient of the *PDR*

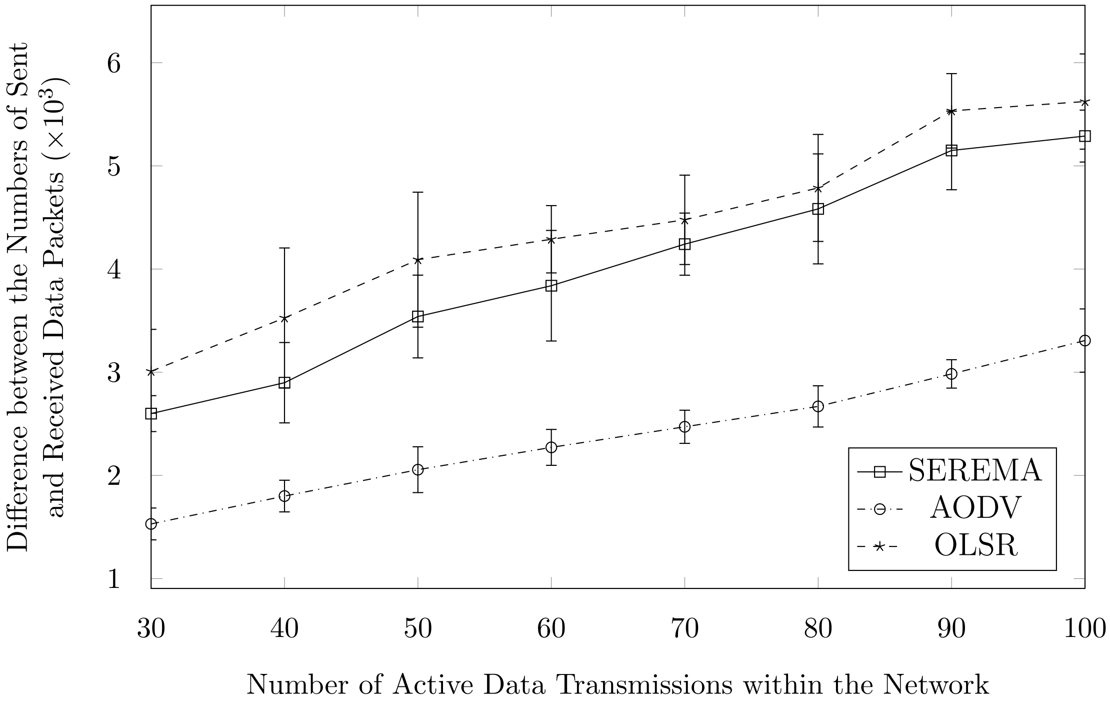


Figure 8.23: Comparison of the Differences between the Numbers of Received Data Packets Related to the Numbers of Sent Data Packets for Performance Scenario 1

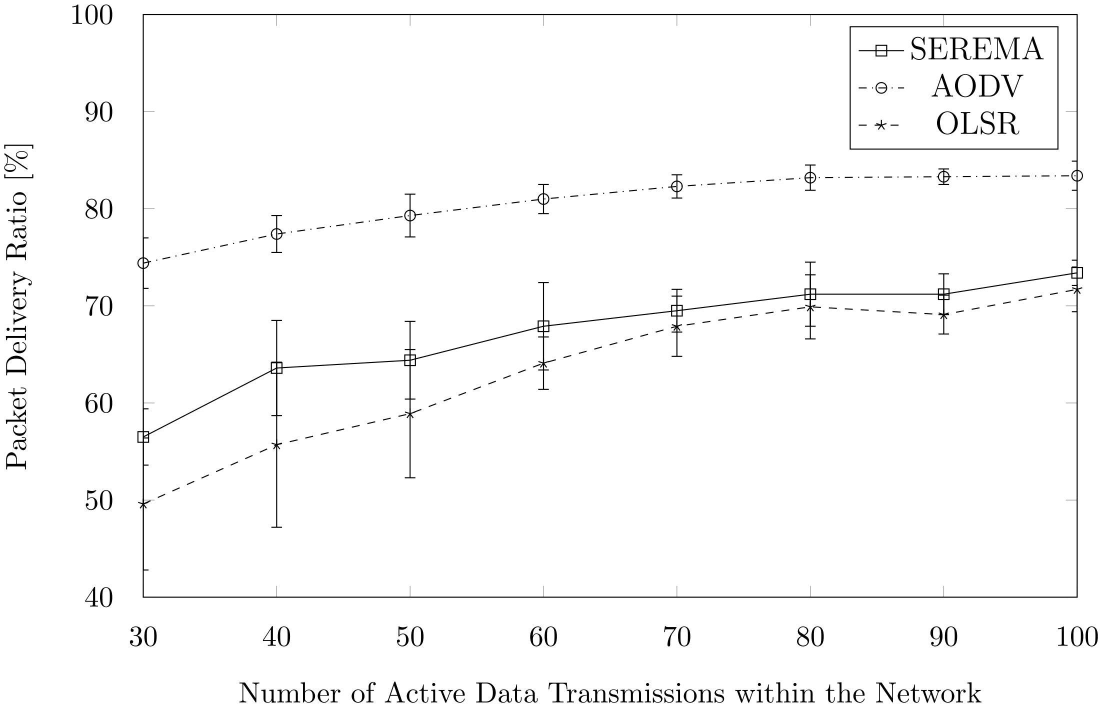


Figure 8.24: Comparison between Packet Delivery Ratios for Performance Scenario 1

gets negative. The *PDR* of *SEREMA* is comparable to *AODV* and *OLSR*. *SEREMA* outperforms *OLSR* because of the reactive behavior of *SEREMA* in routing domain one which lowers the generated routing overhead and allows a faster adaption to changes in the network topology. However, *SEREMA* cannot reach the *PDR* of *AODV* because of its proactive routing part which causes additional routing overhead compared to a pure *AODV* implementation.

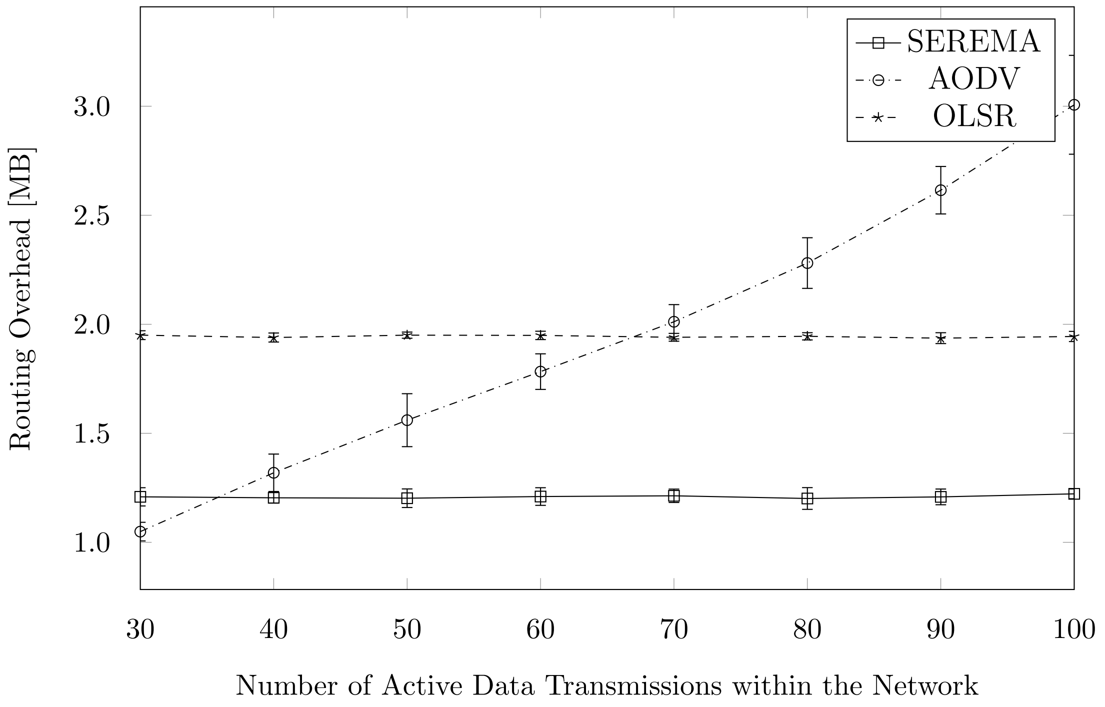


Figure 8.25: Comparison between the Routing Overhead for Performance Scenario 1

The simulations with pure *AODV* nodes have a higher *PDR* compared to *OLSR* and *SEREMA* but their produced routing overhead increases heavily with an increasing number of active data transmissions within the network as shown in figure 8.25 (based on the data in table D.6). As the pure *AODV* scenario routes completely reactively, the generated routing overhead is strongly dependent on the number of active data transmissions and, therefore, increasing with the number of them. In contrast, the routing overhead of *OLSR* is more or less constant as the proactive behavior of *OLSR* produces routing traffic mostly independent from the number of active data connections. The traffic generated by *SEREMA* behaves similar to *OLSR* as most of the data transmissions take place in the proactive routing area. The offset between the *OLSR* graph and the *SEREMA* graph is caused by the constant number of data transmissions *SEREMA* routes reactively in routing domain one as well as between the routing domains.

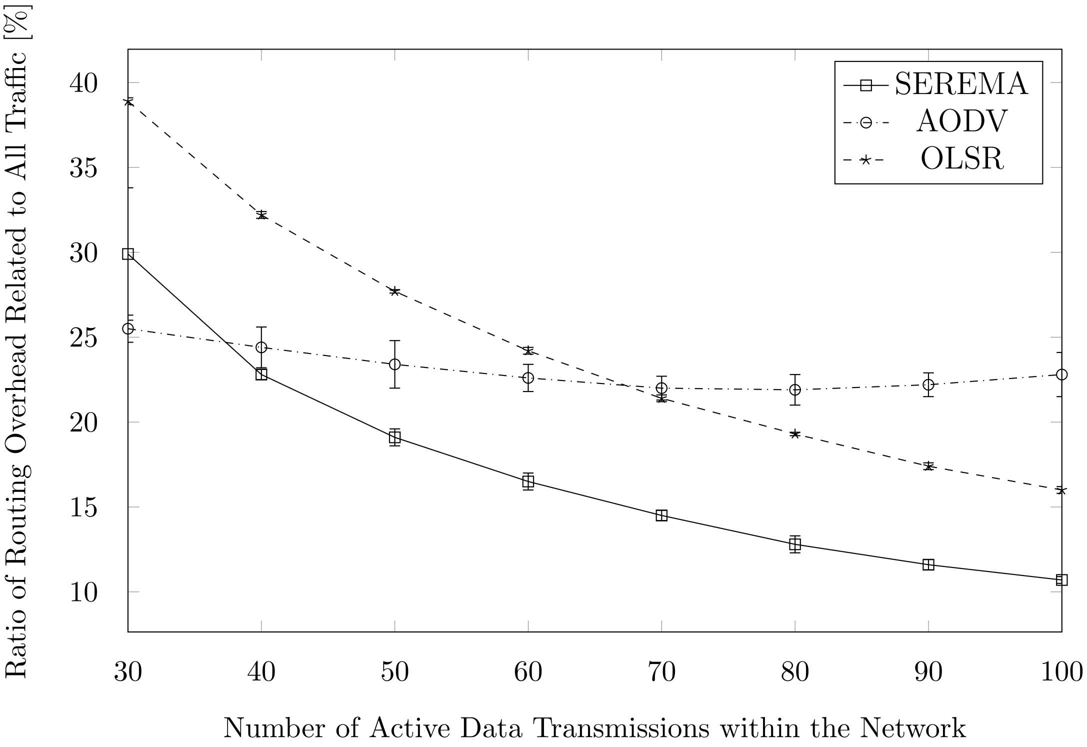


Figure 8.26: Comparison between the Ratios of the Routing Overhead for Performance Scenario 1

The ratio of routing overhead related to all generated traffic in the network is illustrated in figure 8.26 and based on the data in table D.7. It can be seen that the proactive behavior of *OLSR* and the almost proactive behavior of *SEREMA* leads to a decrease of the routing overhead ratio with an increasing number of data transmissions because the produced routing overhead of proactive routing schemes stays nearly constant while the data traffic increases. When the number of active data transmissions is low, *AODV* outperforms *OLSR* and *SEREMA* but as soon as the number of data transmissions increases the routing overhead of *AODV* also increases and, therefore, the protocol performs worse than its proactive counterparts.

The number of dropped packets shown in figure 8.27 is based on the data in table D.8 and illustrates that the number of packet drops caused by *AODV* increases strongly with the number of data transmissions. Since *OLSR* has a proactive behavior and, therefore, its generated routing overhead does not explode with an increasing number of active data transmissions it overcomes this problem. In this scenario *SEREMA* benefits from its use of different routing protocols. In routing domain one where the traffic is low it uses *AODV* and, therefore, its produced routing overhead remains low as seen in figure 8.26 while in routing domain two it profits from *OLSR* to keep the number of packet drops low (cf. figure 8.27).

The presented simulation results showed that in some cases the *AODV* protocol outperforms *SEREMA*. This is true in the presented simulation scenario but as *SEREMA* uses an adaptive routing approach it could completely change its routing protocol to behave like *AODV*. To enable this functionality the monitoring as well as decision making has to be improved.

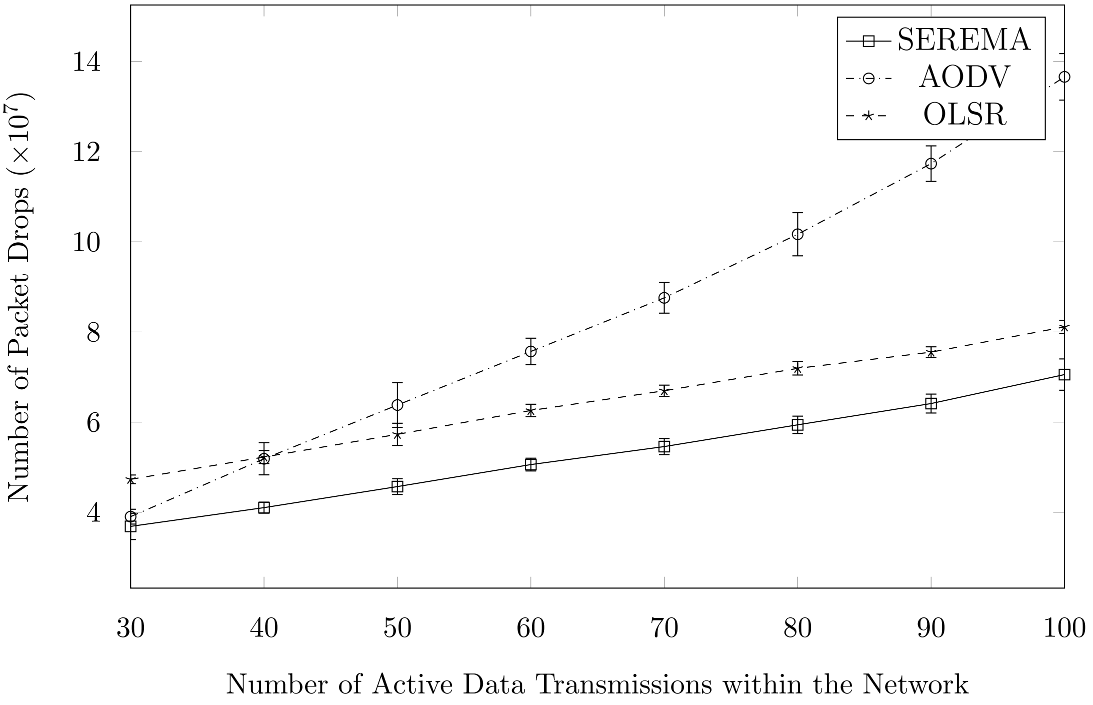


Figure 8.27: Comparison between the Number of Packet Drops for Performance Scenario 1

8.2.2 Scenario 2 (AODV – OLSR – AODV)

The previous subsection considered a scenario consisting of two routing domains and one intermediate *Border Node*. In this subsection a network consisting of three routing domains, two *Border Nodes* and data transmissions via an intermediate routing domain will be analyzed.

8.2.2.1 Simulation Setup

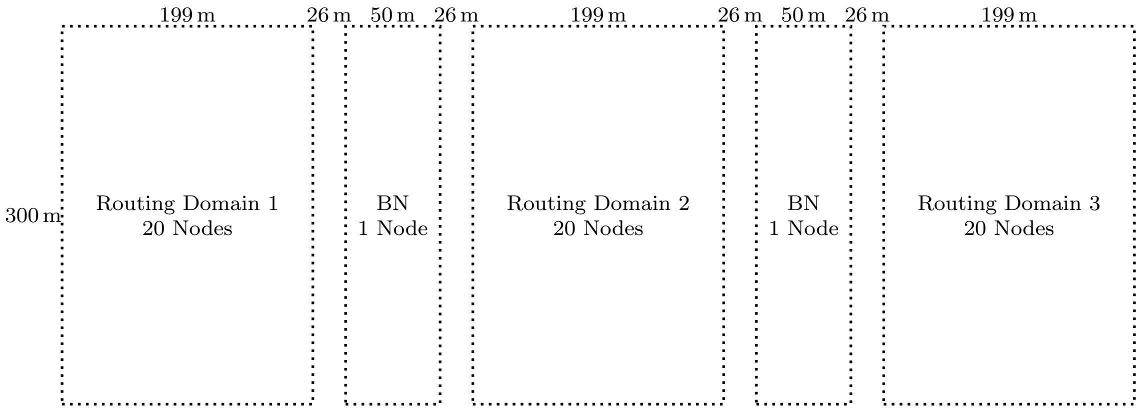


Figure 8.28: Performance Scenario 2 (AODV – OLSR – AODV)

Scenario two (cf. figure 8.28) considers a network existing of three routing domains and two *Border Nodes* in between. It is used to analyze the performance of networks that forward packets via an intermediate routing domain. The simulated network has an overall size of 300 m in vertical direction and 801 m in horizontal direction where the larger dimension is divided into 199 m for routing domain one, 26 m distance, 50 m for the area of the first *Border Node*, another 26 m of free space, 199 m for routing domain two, a 50 m area for the second *Border Node* with 26 m of free space on its left and right side, and at the end a 199 m area for routing domain three.

As a single routing domain in the scenario has the dimensions 199 m x 300 m and contains 20 nodes, the distance between the nodes in horizontal direction is calculated as $\frac{199\,\mathrm{m}}{3} = 66.34\,\mathrm{m}$ while the distance in the vertical direction is $\frac{300\,\mathrm{m}}{4} = 75.00\,\mathrm{m}$ when the nodes are distributed in a grid as illustrated in figure 8.29. The *Average Node Degree* $\bar{d}$ is calculated as $\bar{d} = \frac{1}{20} * \sum_{i=1}^{20} d_i = 3.1$, where d_i represents the number of neighbors of node i as shown inside the node circles in figure 8.29. This *Average Node Degree* results in a neighbors per node ratio of $\frac{3.1\,\mathrm{nodes}}{20\,\mathrm{nodes}} * 100 = 15.5\,\%$ in each routing domain and

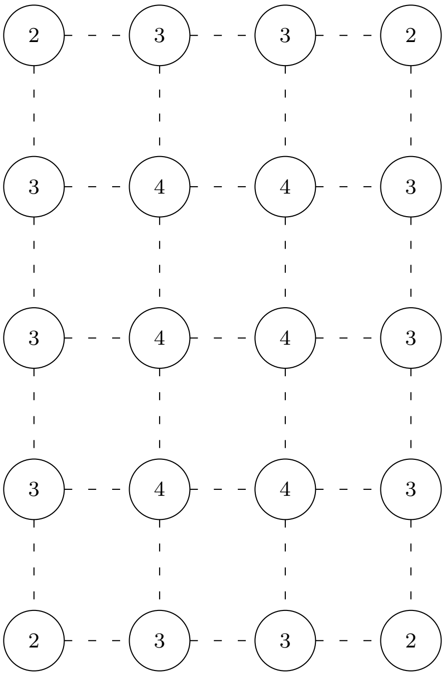


Figure 8.29: Uniformly Distributed Nodes in one Part of Scenario 2 with the Dimensions 199 m x 300 m and Visualized Connections

ensures a good connectivity between the nodes while also enough nodes are available that can only be reached via the routing.

The movement patterns for the nodes are generated in a similar way as in scenario one with *HNMotion*. The utilized commands are illustrated in figure 8.30.

```
hnmotion -x 50 -y 300 -X 225 -Y 0 -n 1 -N 0 -d 300 -l 1 -h 3 -R 12 -p ↩
5 > scenario2.ns_movements
hnmotion -x 50 -y 300 -X 526 -Y 0 -n 1 -N 1 -d 300 -l 1 -h 3 -R 12 -p ↩
5 >> scenario2.ns_movements
hnmotion -x 199 -y 300 -X 0 -Y 0 -n 20 -N 2 -d 300 -l 1 -h 3 -R 12 -p ↩
5 >> scenario2.ns_movements
hnmotion -x 199 -y 300 -X 301 -Y 0 -n 20 -N 22 -d 300 -l 1 -h 3 -R 12 ↩
-p 5 >> scenario2.ns_movements
hnmotion -x 199 -y 300 -X 602 -Y 0 -n 20 -N 42 -d 300 -l 1 -h 3 -R 12 ↩
-p 5 >> scenario2.ns_movements
```

Figure 8.30: *HNMotion* Commands for Creating the Mobility File Used in Scenario 2

To analyze the communication between different routing protocols, the simulation for *SEREMA* uses *AODV* in routing domain one, *OLSR* in routing domain two and *AODV* again in domain three as described in table 8.3. The data transmissions are configured to be three transmissions from domain one to domain three via the intermediate routing domain two, three connections in the reverse direction, one connection from the reactive

Simulated Protocol	Routing Domain 1	BN	Routing Domain 2	BN	Routing Domain 3
AODV	AODV	AODV	AODV	AODV	AODV
OLSR	OLSR	OLSR	OLSR	OLSR	OLSR
SEREMA	AODV	BN	OLSR	BN	AODV

Table 8.3: Node Configuration in Performance Scenario 2

routing domain one to the proactive domain two and one connection from the proactive domain two to the reactive domain three. The rest of the data transmissions take place inside the proactive area as shown in table 8.4.

Source	Destination	Number of Active Data Transmissions
Routing Domain 1	Routing Domain 3	3
Routing Domain 3	Routing Domain 1	3
Routing Domain 1	Routing Domain 2	1
Routing Domain 2	Routing Domain 3	1
Routing Domain 2	Routing Domain 2	$N - 8$

Table 8.4: Configuration of the Data Transmission in Performance Scenario 2 (N is the Number of All Active Data Transmissions in the Simulation)

All simulation parameters which are not explicitly mentioned in this subsection are configured as described for scenario one (cf. subsection 8.2.1.1).

8.2.2.2 Simulation Results

The network scenario discussed in this subsection is an extension to scenario one and, hence, the shapes of the graphs are similar to the graphs presented in subsection 8.2.1.2. Therefore, this subsection will not concentrate on the shapes of the graphs, but instead on differences between scenario one and scenario two.

Figure 8.31 visualizes the *PDR* (cf. data in table D.9) of *AODV*, *OLSR* and *SEREMA*. Compared to the *PDR* of scenario one (cf. subsection 8.2.2.2) it can be seen that the *PDR* in scenario two decreased. In this scenario the number of data transmissions between the different routing domains was increased from two transmissions to eight. As routing domain two which has a high traffic load (cf. table 8.4) is used by packets from domain one and domain three for transit, the number of packet drops rose (cf. figure 8.32 and table D.10) compared to the number of packet drops in scenario one

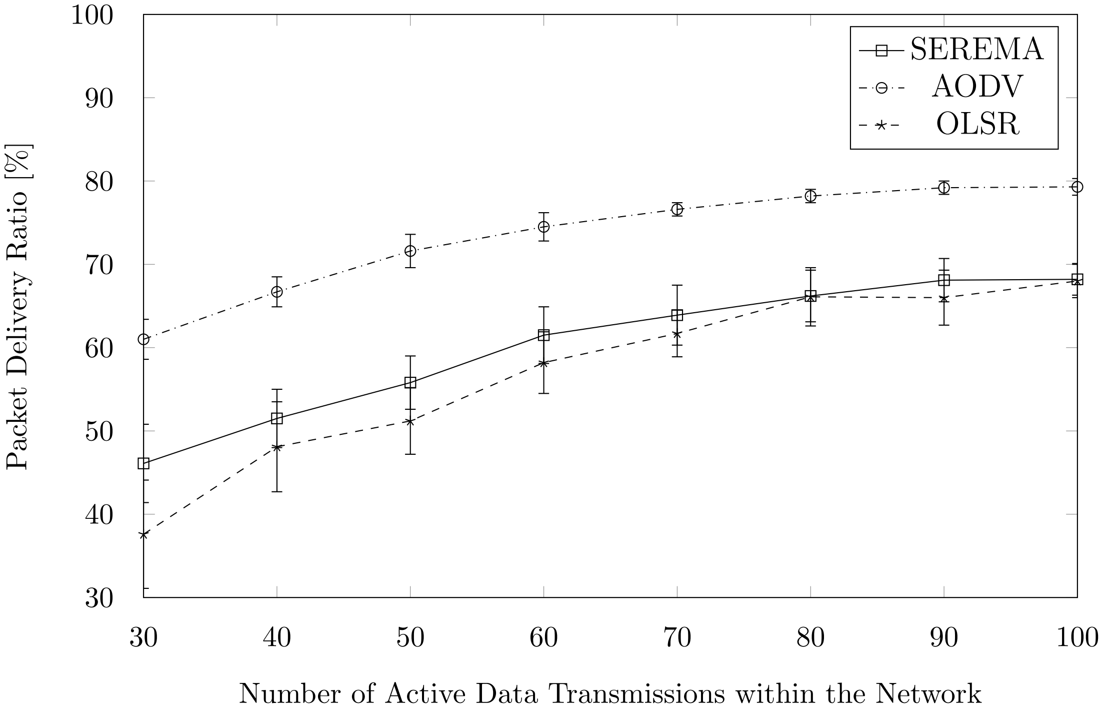


Figure 8.31: Comparison between Packet Delivery Ratios for Performance Scenario 2

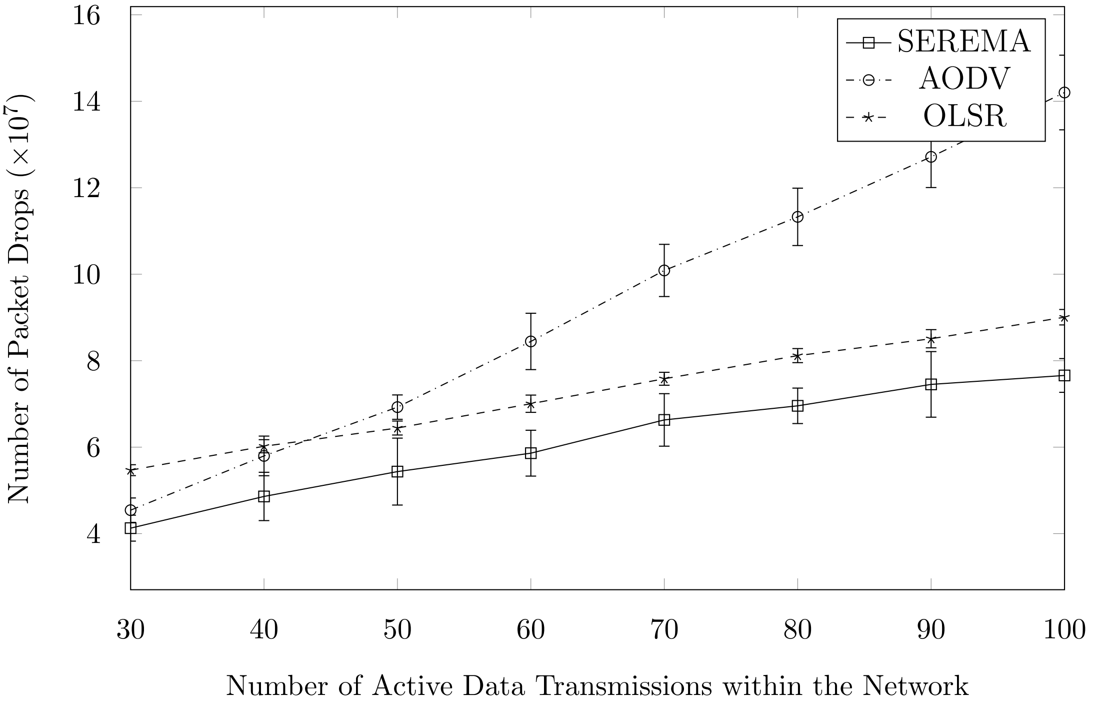


Figure 8.32: Comparison between the Number of Packet Drops for Performance Scenario 2

as seen in figure 8.27 and, therefore, the *PDR* decreased. Nevertheless, *SEREMA* performs comparable to *AODV* and *OLSR*.

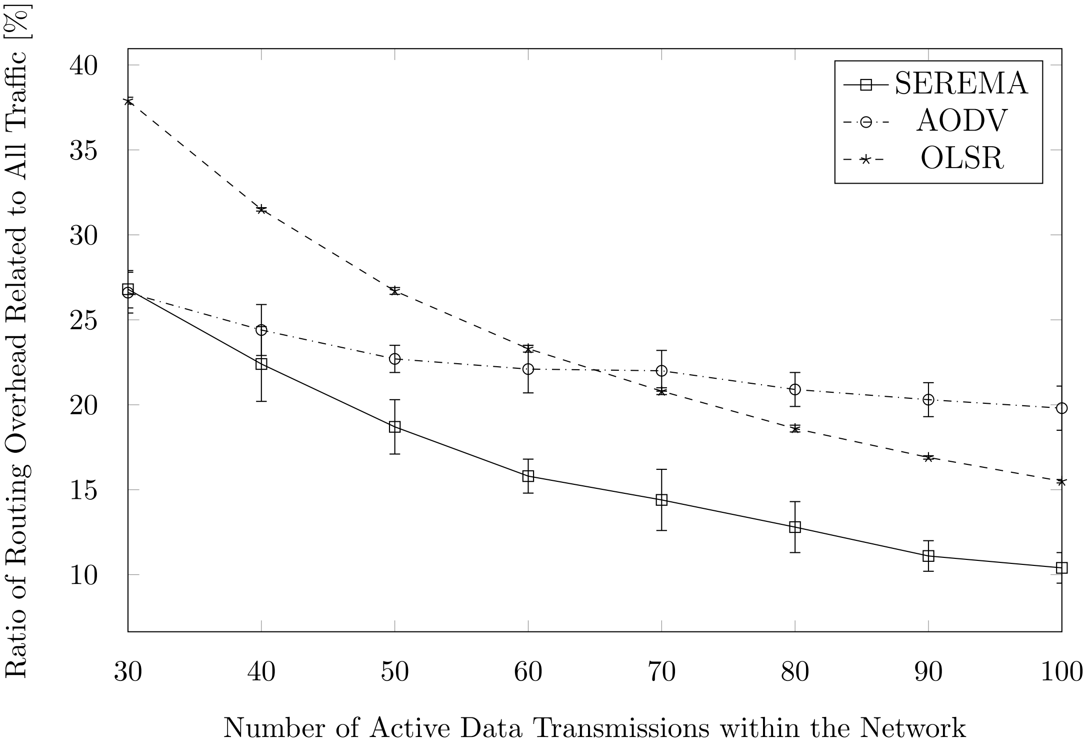


Figure 8.33: Comparison between the Routing Overhead for Performance Scenario 2

The produced routing overhead (cf. figure 8.33) based on the data in table D.11 shows that some measurement points changed minimally up or down compared to scenario one, but the overall results remain nearly unchanged and show that the additional traffic caused by *SEREMA* packet extensions as well as by tunneled packets does not exhaust the network. This behavior is important as it shows that *SEREMA* does not exhaust the sparse resources of *MANETs*.

8.3 Conclusion

This chapter presented different simulated network scenarios to validate the functionality of *SEREMA*. All simulation setups were described in detail with all configured parameters to allow a reproduction of them as well as a comparison with further simulations which may be done in the future.

In a first step the basic behavior of *SEREMA* was validated using static nodes. The simulation results showed that *SEREMA* works as expected. Especially the behaviors of the following functionalities, which confirmed the successful operation of *SEREMA*, were thereby validated:

- Generation of *BNANNOs*
- Generation of *HNA* messages
- Tunneling of routing packets via foreign routing domains
- Activation of the *Passive Border Node* mode on proactive nodes
- Activation of the *Border Node* mode on reactive nodes with proactive neighbors
- Route discovery via multiple different routing domains

These points proved the functionality of the adaptive routing framework of *SEREMA* and showed that the adaptive mechanism, simultaneous protocols and the packet conversion requirements, made in section 4.3, work as expected. Furthermore, it was validated that the newly introduced packet types are generated and processed as described in chapter 6.

In a second step the performance of *SEREMA* was analyzed using network scenarios with different routing domains and mobile nodes. The simulations showed that *SEREMA* can keep up with *AODV* and *OLSR* as well as outperform them in specific scenarios. Furthermore, the simulation results showed that the generated routing overhead is comparable to *AODV* and *OLSR* and, therefore, *SEREMA* does not exhaust the sparse resources of *MANETs*. To support further simulation runs with nodes having more degrees of freedom, for example an autonomous selection of their routing protocols, improvements of the monitoring and decision making are required. This would enable *SEREMA* to cope with or outperform the standard protocols in any scenario.

Further requirements made in section 4.3, such as the number of active routing tables, the extension effort for further protocols, the compatibility to standard routing nodes as well as the robustness against single node failures were not explicitly validated in this chapter as these characteristics are design fundamentals. Another very important part of *SEREMA* which has not been validated yet is the quality of the decision making.

This part should be validated in the future after improving the monitoring as well as the decision making algorithm.

During the process of validation it was noticed that the simulation runs take a relatively long time to complete. As *ns-3* normally cannot utilize multiple *CPU* cores simultaneously for speeding up simulation runs, a *Linux shell* script was written to run multiple simulations in parallel. This shortened the overall simulation time, but a whole simulation for *AODV*, *OLSR* and *SEREMA* over 300 seconds with 50 mobile nodes, averaged over ten runs still requires around one week to complete. The hardware configuration utilized an *Intel Core i7 2600K* processor with four cores, allowing eight simultaneous threads and running at 3.40 GHz. The selected operating system was *Ubuntu 10.04*. The long simulation times should be considered when planning further simulation runs for *SEREMA*.

9 Summary

The communication within disaster scenarios is required to coordinate disaster response teams and became possible by the huge amount of powerful hand-held devices. Chapter 1 explained the problems disaster response teams are faced with regarding the communication with each other. Previously existing communication infrastructure could be damaged or might not even have existed, such as in isolated areas. It was shown that ad hoc networks can enable the communication in such scenarios and that the routing in those networks is a very important issue. The importance of communication in the described scenarios requires an efficient routing which motivated this work to develop an adaptive routing framework to perform well in a lot of different network scenarios. At the end of chapter 1 the structure of the present dissertation was introduced.

The most important basics for understanding ad hoc networks where worked out in chapter 2, starting with mobile communication that uses previously installed communication infrastructure like base stations, describing the handicaps of such an infrastructure and showing the benefits of ad hoc networks. Furthermore, the limitations and characteristics of ad hoc networks were explained

The subsequent chapter 3 introduced the routing in *MANETs* and discussed the challenges of this kind of networks, including energy restrictions, the reaction to link failures as well as the interconnection of different networks. The chapter also described the fundamental requirements made on routing protocols, e.g. the avoidance of routing loops. The basic mechanisms for proactive as well as reactive routing where introduced and the functionality of widely-used routing protocols utilized in the present work was described in detail for the parts of the protocols that are required for understanding *LARA* as well as *SEREMA*. Section 3.8 described *LARA* which is able to decrease the averaged latency for the route discovery process in reactive routing protocols and which was developed in the scope of this dissertation research. The protocol was explained in detail and its benefits were shown.

In chapter 4 hybrid and adaptive routing were introduced and related work in this area of research was presented. The requirements on adaptive routing made in this work were defined and compared to related work. It was shown that the existing approaches cannot meet all requirements and, therefore, are only suitable to a limited degree.

The subsequent chapter 5 illustrated the new approach *SEREMA* which was developed in the scope of this dissertation. *SEREMA* is able to change the routing protocol in a self-organized way during the run-time of the network. As each node is allowed to select its own routing protocol, *SEREMA* provides multiple simultaneous routing protocols in the network, providing the best protocols for specific areas in the network.

A special feature of *SEREMA* is its newly introduced *RTW* which allows the access to multiple simultaneous routing tables by the *Network Layer* allowing the use of routes provided by different routing protocols. This enables the look-up of routes which are stored in different routing tables, allowing the switching between different routing protocols during the run-time without an interruption of on-going data transfers as the *RTW* can provide the routes of the previous routing protocol until the new protocol has discovered new routes. Furthermore, the *RTW* provides each routing protocol with the continuing ability to use its own unmodified routing table. Currently to the best of knowledge, *SEREMA* is the only adaptive routing framework that uses an *RTW* to access different routing tables. If a network consists of different routing areas, *Border Nodes* are used to interconnect them. The *Border Nodes* profit from the ability of a node in *SEREMA* to operate multiple simultaneous protocols to act as intermediary between different routing areas.

The existing adaptive routing schemes that provide simultaneous routing protocols in the network require extensive protocol modifications in terms of a common routing mode on all nodes or a proactive routing protocol that offers a reactive behavior for the route discovery procedure. *SEREMA* overcomes these handicaps by utilizing its *Border Nodes* and activating multiple simultaneous protocols on specific nodes. On a border between different routing areas, *SEREMA* converts routing information once for each destination protocol to keep the processing effort and, therefore, the energy consumption by the nodes, low.

When an adaptive routing approach utilizes multiple simultaneous routing protocols, it is indispensable to exchange additional routing information to enable the communication between different routing areas. Therefore, the related approaches introduced new packet types as well as protocol header extensions which have the drawback that they reduce the compatibility to nodes not operating the new routing scheme. The extensions used in *SEREMA* are made in the scope of the protocol specifications to maximize the compatibility to standard routing nodes and to minimize the effort of adding further protocols to the framework.

There is a trade-off between the generated routing overhead and the achieved quality of the routing decisions. It was shown that the decision in the current version of *SEREMA* was made in favor of a minimized routing overhead, trying to bring the nodes to a completely autonomous routing decision with no additional traffic for exchanging network statistics between the nodes. Furthermore, the autonomy of the nodes provides a completely decentralized monitoring and decision making which is robust against single node failures in the network.

The detailed architecture and functionality of *SEREMA* was described in chapter 6, based on the routing protocols *AODV* and *OLSR* which were selected as base routing protocols at the beginning of the chapter. The functionality of the *Monitoring Agent* was described and the usage of its results by the *Decision Maker* was explained. It was depicted how *SEREMA* switches between different routing protocols and the behavior of the *RTW* which allows the simultaneous use of multiple routing tables was shown. One of the most important tasks of *SEREMA* is the interconnection of different routing areas by the *Border Nodes*. This special mode of operation allows a node to translate routing information from routing protocol A to protocol B and vice versa. The behavior of the *Border Nodes* was described in detail, complemented with exemplary scenarios. Moreover, the compatibility of nodes not supporting the functionality of *SEREMA* was discussed.

Chapter 7 presented different ways for the validation of *SEREMA* and elected network simulations as favored solution. Different widely-used network simulators where compared and it was shown that *ns-3* is the best solution for this task. To simplify a later validation on a real testbed, *Click* was introduced and its combination with *ns-3* to the simulation environment *ns-3-click* was described. *Click* allows the reutilization of the *SEREMA* implementation, that was used for the simulation, on real testbeds. The importance of suitable mobility models was highlighted and the tools *BonnMotion*, setdest as well as *HNMotion* – which was developed for the simulation requirements in the scope of this work – were introduced.

The validation presented in chapter 8 successfully proved the functionality of *SEREMA*. First simulation runs validated the behavior of *SEREMA* using simple scenarios with static nodes. The analyzed packet traces showed that the *Border Nodes* work well and that route discoveries via different routing domains are possible. Further simulation runs analyzed the performance of *SEREMA* compared to *AODV* and *OLSR* using network scenarios with mobile nodes. The simulation results proved that *SEREMA* can outperform the standard routing protocols in specific scenarios as well as behave like the standard protocols if all nodes change to the same protocol. All simulation runs were described in detail to allow a reproduction of them.

The present dissertation introduced some scientific innovations for the adaptive routing scheme used in *SEREMA* to reach the requirements made. The framework only exchanges new routing message types that are foreseen in the scope of the standard routing protocol specifications to maximize the compatibility as well as minimize the implementation effort. Furthermore, the unique *RTW* of *SEREMA* allows the access to multiple simultaneous routing tables of different routing protocols, enabling the availability of routes during a protocol change and allowing each routing protocol to use its own routing table. This reduces the implementation effort for further routing protocols in *SEREMA* as the protocol and its routing table do not have to be modified heavily. Since the implementation of *SEREMA* is based on *Click*, the framework can simply be simulated and adapted to a real testbed.

The work on *SEREMA* also offered scientific findings. Adaptive routing mechanisms are able to improve the performance of ad hoc networks in different scenarios. However, the behavior of the adaptive approaches is strongly dependent on the knowledge about the network and the quality of the measurements to obtain the current state of the network. An extensive knowledge about the behavior of ad hoc networks in different network scenarios is essential for a good performance of the adaptive routing scheme. At the beginning of this work it was defined that each node in the network should be allowed to make its protocol decision completely on its own, having in mind that this could decrease the quality of the decision making. *SEREMA* showed that a network with completely autonomous nodes will work but cannot perform well as too many borders between different routing protocols occur, leading to a heavily increasing number of *Border Nodes* and, therefore, a high routing overhead.

The adaptive routing framework presented in this dissertation complies to a large extent with the requirements made in section 4.3. Table 5.1 illustrates that *SEREMA* meets the requirements in terms of the *Adaptive Mechanism*, the *Simultaneous Protocols*, the *Packet Conversions* and the *Robustness*. Furthermore, *SEREMA* outperforms the existing adaptive routing approaches that support simultaneous protocols in terms of the number of *Routing Tables* and the required *Extension Effort*. When considering the *Packet Types*, the *Efficiency* and the *Compatibility*, *SEREMA* outperforms some of the approaches and behaves comparable to the other schemes.

However, it is strongly recommended to improve the *Monitoring Agent*, the *Routing Mode Information* as well as the *Decision Maker* to support distributed routing decisions based on an increased knowledge about the current state of the network and the behavior of different routing protocols in specific network constellations to increase the quality of the routing protocol decisions of the nodes. The following chapter 10 mentions problems identified during the current implementation of *SEREMA*, presents possible solutions and recommends improvements to the adaptive routing framework for future research as well as development tasks.

10 Outlook

The present work offers a functional framework for adaptive routing that enables efficient route discovery and route maintenance in different *MANET* scenarios. The framework is strongly depended on the monitoring as well as the decision making and, therefore, the *Decision Maker* element should be improved in the next steps to increase the quality of the decisions made by *SEREMA*.

To increase the quality of the decisions made by the *Decision Maker*, the *Monitoring Agent* is required to provide accurate and up-to-date information about the current state of the network. Therefore, the monitoring could be extended to measure routing specific data like the number of known destinations or end-to-end delays of available routes. Chapter 5.1 stated that it was a fundamental requirement for this dissertation to use nodes that are only allowed to make their protocol decision completely independently and that this could result in a decreased quality of a node's view. Therefore, the *Monitoring Agent* should be extended to exchange information such as network statistics with neighbor nodes in an efficient manner. This task could be based on previous work that was done by Ramachandran et al. [RBRA04] or Winter et al. [WSNB05]. If measurements on real nodes are required, the monitoring tool presented by Ngo et al. [NHHW03] allows the gathering of different parameters on *Linux* systems.

If each node was allowed to select its protocol completely autonomously, a lot of borders between different routing areas could occur, leading to huge routing overhead because most of the nodes would operate multiple routing protocols simultaneously. Therefore, the *Decision Maker* should try to organize the nodes into groups, resulting in different partitions – each using its own routing protocol – interconnected by *Border Nodes*. To limit the number of *Border Nodes* the nodes should communicate with each other to use a more distributed view for electing *Border Nodes*. One idea for organizing the *Border Nodes* is that a node is only allowed to become a *Border Node* when it does not have another *Border Node* in its direct neighborhood. However, this could lead to unconnected nodes in the network because a required node could be forced not to activate its *Border Node* mode. A further idea is to let each node select its *Border Node* mode completely on its own as described in chapter 6.6.1 and give the other nodes the possibility of sending messages to a specific node to tell it that it might be better to activate or deactivate its *Border Node* mode. A similar procedure is used in the work of Lee et al. [LWC$^+$10] to activate and deactivate gateway nodes.

Beside the *Monitoring Agent* a good *Routing Mode Information* is essential for efficient routing decisions. Therefore, further simulation runs or measurements on real testbeds should be done to gather as much information as possible about the behavior of *MANETs* in different network scenarios. As the description of a mathematical algorithm for characterizing the behavior of *MANETs* is a challenge, new approaches which are based on fuzzy logic or artificial neural networks could be an option to make better routing protocol decisions.

If the *Routing Mode Information*, *Monitoring Agent* as well as *Decision Maker* are improved, some smaller changes on the behavior of *SEREMA* are advised for further enhancement. The first behavior in this category is the broadcasting of *RREQs* on *Passive Border Nodes* as described in subsection 8.1.2. These *RREQs* could be avoided because a *Passive Border Node* usually has no reactive neighbors. This improvement would increase the performance, but modify the standard *AODV* protocol. The second behavior is that *SEREMA* does not decrement the *TTL* when it tunnels a packet. Therefore, in the current implementation, the limitation of the distribution of packets by their lifetime may not work as expected. This handicap was also mentioned in subsection 8.1.2. Due to the issue with the missing *Hop Count* information in *OLSR HNA* messages as described in subsection 3.5.3 the current implementation of *OLSR* used in *SEREMA* was modified. To the end of each entry in a *HNA* message a *Hop Count* information was added. However, this is a change of the *OLSR* specification to solve the problem and it is recommended to change the implementation from *OLSR* to *The Optimized Link State Routing Protocol Version 2 (OLSRv2)* [CDJH14] instead. This also brings some further advantages as described by Dearlove [Dea10].

After the integration of the mentioned improvements, the framework should be tested on real hardware. However, managing 50 mobile nodes for example and monitoring as well as analyzing the parameters could become a big challenge, let alone aspects such as the resulting cost for 50 mobile devices. Solutions could be to reduce the considered network scenarios to a minimum of nodes or to run an amount of nodes on virtualized hardware. This would decrease the level of complexity as well as cost, but the first approach does not support large networks and the second approach has a simplified physical layer between the nodes. If *SEREMA* is applied on real hardware *Click* should be run in *Kernel* mode to get the best performance.

Another aspect that has to be considered before *SEREMA* can be applied to real disaster scenarios is the security. In the scope of the present work security aspects are not considered, but the next steps should include them. Some questions that have to be discussed in this topic are: Will multiple simultaneous routing protocols increase the number of security leaks? How can the manipulation of routing information be avoided? What can be done to prevent the injection of faked routing information?

This chapter discussed the remaining issues that should be considered in future work to prepare *SEREMA* for its use in real life disaster scenarios. In recent years the number of ad hoc networks has increased and it is predictable that this increase will continue. Some more effort has to be spent on *MANETs* to make them feasible for the communication in disaster scenarios, which also includes the improvement of adaptive routing schemes. A big challenge for the routing in *MANETs* are highly mobile nodes as they make connections short-timed and provoke the routing to continuously find alternative routes.

A further aspect that should be considered is that the approach of *SEREMA* or parts of it could be used in other areas beside disaster scenarios. This implies for example home automation systems as considered in Hager et al. [HFSW14] which could be improved by the *Border Node* scheme to allow the integration of sensors and actors which are based on different network technologies as well as routing mechanisms.

Appendices

A Routing Parameters

The configuration parameters for the routing protocols *AODV*, *OLSR* and *SEREMA* are shown in the following tables A.1, A.2 and A.3.

Table A.1: Configuration Parameters for the AODV Implementation

Parameter	Value	Description
HELLO_INTERVAL	1000 ms	The emission interval for *HELLO* messages
ALLOWED_HELLO_LOSS	2	Describes how many times a node waits when not receiving *HELLOs* from a neighbor before it assumes a broken link
ACTIVE_ROUTE_TIMEOUT	3000 ms	The maximum allowed lifetime of a route that is not used for packet forwarding
NET_DIAMETER	35	The maximum number of hops between two nodes in the network
NODE_TRAVERSAL_TIME	40 ms	The estimated time for the packet processing on a node
NET_TRAVERSAL_TIME	2800 ms	The time between sending *RREQs* for the same route discovery procedure
PATH_DISCOVERY_TIME	5600 ms	The time a node stores a received *RREQ* for recognizing duplicate *RREQs*

continued . . .

Configuration Parameters for the AODV Implementation

. . . continued

Parameter	Value	Description
TTL_START	2	The start value for the *TTL* when using the *Expanding Ring Search* mechanism
RING_TRAVERSAL_TIME_FACTOR	80 ms	The maximum allowed time for waiting for an *RREP*
TIMEOUT_BUFFER	2	An additional time for route discovery procedures taking into account delays because of congestion in the network
TTL_INCREMENT	2	The incrementation of the *TTL* when using the *Expanding Ring Search* mechanism
TTL_THRESHOLD	7	The limit for incrementing the *TTL* during an *Expanding Ring Search*
RREQ_RETRIES	2	The maximum number of generated *RREQs* for a specific route discovery procedure
MY_ROUTE_TIMEOUT	6000 ms	The value of the lifetime field which is placed into *RREP* packets
DELETE_PERIOD	2000 ms	The time span after which invalid routes are deleted from the routing table

Table A.2: Configuration Parameters for the OLSR Implementation

Parameter	Value	Description
HELLO_INTV	2 s	The emission interval for *HELLO* messages
TC_INTV	5 s	The emission interval for *TC* messages
MID_INTV	5 s	The emission interval for *MID* messages
HNA_INTV	5 s	The emission interval for *HNA* messages

Table A.3: Configuration Parameters for the SEREMA Implementation

Parameter	Value	Description
BNANNO_INTV	5 s	The emission interval for *BNANNO* messages
DECISIONMAKER_PERIODE_BN	5 s	The interval for the execution of the *Decision Maker*. A node checks on each call if it has to activate its *Border Node* mode. In every sixth interval the results of the *Monitoring Agent* are evaluated.

B Click Elements

Table B.1 in this part of the appendix presents a list of all *Click* elements being used in the *SEREMA* framework. If the name of an element starts with *AODV*, *OLSR* or *SEREMA* it is a special element for this protocol, otherwise it is used generally. The column *Origin* shows where this element comes from and the column *Modified* shows if the element was modified for its use in *SEREMA* or not. All elements that are not marked as newly introduced are available from Braem [Braa, Brab] if they belong to *AODV* or *OLSR*. The general elements which are not newly introduced are available from the *Click* website [Unib]. It should be considered that also newly introduced elements may partly be based upon other elements from the sources mentioned above.

Table B.1: Elements of the SEREMA Click Graph

Element	Origin	Modified
AODVDestinationClassifier	AODV	yes
AODVGenerateRERR	AODV	no
AODVGenerateRREP	AODV	yes
AODVGenerateRREQ	AODV	yes
AODVHelloGenerator	AODV	yes
AODVKnownClassifier	AODV	yes
AODVLinkNeighboursDiscovery	AODV	no
AODVLookupRoute	AODV	yes
AODVNeighbours	AODV	yes
AODVSetRREPHeaders	AODV	no
AODVTrackNeighbors	AODV	no
AODVUpdateNeighbours	AODV	yes
AODVWaitingForDiscovery	AODV	yes
ARPQuerier	Click	no
ARPResponder	Click	no

continued . . .

Elements of the SEREMA Click Graph

. . . continued

Element	Origin	Modified
CheckIPHeader	Click	no
CheckPaint	Click	no
CheckPaint16	new	no
Classifier	Click	no
DecIPTTL	Click	no
Discard	Click	no
EtherEncap	Click	no
FromSimDevice	Click	no
GetIPAddress	Click	no
HostEtherFilter	Click	no
ICMPError	Click	no
Idle	Click	no
IPClassifier	Click	no
IPEncap	Click	no
MarkEtherHeader	Click	no
MarkIPHeader	Click	no
OLSRAddPacketSeq	OLSR	no
OLSRAssociationInfoBase	OLSR	yes
OLSRBNANNOGeneratorPart1	new	no
OLSRBNANNOGeneratorPart2	new	no
OLSRCheckPacketHeader	OLSR	no
OLSRClassifier	OLSR	yes
OLSRDuplicateSet	OLSR	no
OLSRForward	OLSR	yes
OLSRHelloGenerator	OLSR	yes
OLSRHNAGenerator	OLSR	yes
OLSRInterfaceInfoBase	OLSR	no
OLSRLinearIPLookup	OLSR	yes
OLSRLinkInfoBase	OLSR	no
OLSRLocalIfInfoBase	OLSR	no
OLSRMIDGenerator	OLSR	no
OLSRProcessBNANNO	new	no

continued . . .

Elements of the SEREMA Click Graph

. . . continued

Element	Origin	Modified
OLSRProcessHello	OLSR	no
OLSRProcessHNA	OLSR	yes
OLSRProcessMID	OLSR	no
OLSRProcessTC	OLSR	no
OLSRRoutingTable	OLSR	yes
OLSRTCGenerator	OLSR	yes
OLSRTopologyInfoBase	OLSR	no
OLSRWaitingForRoute	new	no
Paint	Click	no
Paint16	new	no
PaintSwitch	Click	no
Queue	Click	no
SeremaBordernodeManager	new	no
SeremaCheckTunnelEndpoint	new	no
SeremaDemux	new	no
SeremaFilter	new	no
SeremaGenerateRREPFromOLSR	new	no
SeremaMonitoringAgent	new	no
SeremaOlsrBnannoHelper	new	no
SeremaOlsrHnaHelper	new	no
SeremaOlsrHnaRefresh	new	no
SeremaOlsrHnaTrigger	new	no
SeremaOlsrRouteLookupHelper	new	no
SeremaOlsrRtLookup	new	no
SeremaOlsrTcHelper	new	no
SeremaOnOff	new	no
SeremaPaint32	new	no
SeremaPtrHelper	new	no
SeremaRouteUnknownManager	new	no
SeremaSwitchingDecisionMaker	new	no
SeremaTHopsAnno2Header	new	no
SeremaTTL2THops	new	no

continued . . .

Elements of the SEREMA Click Graph

. . . continued

Element	Origin	Modified
SeremaTunnelendpointDetour	new	no
SeremaTunnelManager	new	no
SeremaTunnelPacketDuplicator	new	no
SetIPChecksum	Click	no
SetUDPChecksum	Click	no
StaticIPLookup	Click	no
Strip	Click	no
StripIPHeader	Click	no
StripToNetworkHeader	Click	no
Tee	Click	no
ToSimDevice	Click	no
UDPIPEncap	Click	no

C Node Configuration

Table C.1 in this part of the appendix gives information about the node configuration which was used by the simulations. The last octet of the *Hardware Address* is equal to the last octet of the corresponding *IP* address[1].

Table C.1: Node Configuration in the Simulations

Parameter	Value
Network Standard	IEEE 802.11g
Maximum Transmission Range	100 m
Network Address	10.1.1.0
Network Mask	255.255.255.0
Hardware Addresses	00:00:00:00:00:xx
Click Version	2.0.1
ns-3 Version	3.12.1

[1] In this work a *Network Address* is mapped to exactly one *Hardware Address*.

D Collection of Data

Table D.1: Throughput of *LARA* Scenario 1

Simulation Time [ms]	AOMDV [bytes/ms]	LLAOMDV [bytes/ms]
3 000	0	0
3 001	0	0
3 002	0	0
3 003	0	0
3 004	0	0
3 005	0	0
3 006	0	0
3 007	0	0
3 008	0	0
3 009	0	0
3 010	0	0
3 011	0	0
3 012	0	0
3 013	0	0
3 014	0	120
3 015	0	0
3 016	0	0
3 017	0	0
3 018	0	0
3 019	0	0
3 020	0	0
3 021	0	0
3 022	0	0
3 023	0	0
3 024	0	0
3 025	0	0
3 026	0	0
3 027	0	0
3 028	0	0

continued . . .

Throughput of *LARA* Scenario 1

. . . continued

Simulation Time [ms]	AOMDV [bytes/ms]	LLAOMDV [bytes/ms]
3 029	0	0
3 030	0	0
3 031	0	120
3 032	120	0
3 033	0	0
3 034	0	0
3 035	0	0
3 036	0	0
3 037	120	0
3 038	0	0
3 039	0	0
3 040	0	0
3 041	0	0
3 042	0	0
3 043	0	0
3 044	0	0
3 045	0	0
3 046	0	0
3 047	0	0
3 048	0	0
3 049	0	120
3 050	120	0
3 051	0	0
3 052	0	0
3 053	0	0
3 054	0	0
3 055	0	0
3 056	0	0
3 057	0	0
3 058	0	0
3 059	0	0
3 060	0	0
3 061	0	0
3 062	0	0
3 063	0	0
3 064	0	0
3 065	0	0
3 066	0	0

continued . . .

Throughput of *LARA* Scenario 1

. . . continued

Simulation Time [ms]	AOMDV [bytes/ms]	LLAOMDV [bytes/ms]
3 067	0	0
3 068	0	0
3 069	120	120
3 070	0	0
3 071	0	0
3 072	0	0
3 073	0	0
3 074	0	0
3 075	0	0
3 076	0	0
3 077	0	0
3 078	0	0
3 079	0	0
3 080	0	0
3 081	0	0
3 082	0	0
3 083	0	0
3 084	0	0
3 085	0	0
3 086	0	0
3 087	0	0
3 088	0	0
3 089	0	0
3 090	120	120
3 091	0	0
3 092	0	0
3 093	0	0
3 094	0	0
3 095	0	0
3 096	0	0
3 097	0	0
3 098	0	0
3 099	0	0
3 100	0	0

Table D.2: Dependency of Average Latency on Route Lifetime in *LARA* Scenario 1

Route Lifetime	AOMDV	LLAOMDV
[s]	[ms]	[ms]
0.1	46.5	46.51
0.2	46.5	46.51
0.3	46.5	46.51
0.4	46.5	46.51
0.5	46.5	46.51
0.6	46.5	46.51
0.7	46.5	46.51
0.8	46.5	46.51
0.9	46.5	46.51
1.0	46.5	37.09
1.1	46.5	37.09
1.2	46.5	37.09
1.3	46.5	37.09
1.4	46.5	37.09
1.5	46.5	37.09
1.6	46.5	37.09
1.7	46.5	37.09
1.8	46.5	37.09
1.9	46.5	37.09
2.0	46.5	37.09
2.1	46.5	37.09
2.2	46.5	37.09
2.3	46.5	37.09
2.4	46.5	37.09
2.5	46.5	37.09
2.6	46.5	37.09
2.7	46.5	37.09
2.8	46.5	37.09
2.9	46.5	37.09
3.0	46.5	37.09
3.1	46.5	37.09
3.2	46.5	37.09
3.3	46.5	37.09
3.4	46.5	37.09
3.5	46.5	37.09
3.6	46.5	37.09
3.7	46.5	37.09
3.8	46.5	37.09

continued . . .

Dependency of Average Latency on Route Lifetime in *LARA* Scenario 1

. . . continued

Route Lifetime [s]	AOMDV [ms]	LLAOMDV [ms]
3.9	46.5	37.09
4.0	46.5	37.09
4.1	46.5	37.09
4.2	46.5	37.09
4.3	46.5	37.09
4.4	46.5	37.09
4.5	46.5	37.09
4.6	46.5	37.09
4.7	46.5	37.09
4.8	46.5	37.09
4.9	46.5	37.09
5.0	46.5	37.09

Table D.3: Comparison between the Number of Received Data Packets Related to the Number of Sent Data Packets for Performance Scenario 1

Number of Data Transmissions	Tx	SEREMA	AODV	OLSR	SEREMA σ	AODV σ	OLSR σ
30	5 970.0	3 370.9	4 440.0	2 960.7	174.0	154.3	406.5
40	7 960.0	5 061.2	6 159.9	4 433.4	389.4	153.4	678.1
50	9 950.0	6 409.7	7 894.3	5 858.9	401.0	222.0	654.2
60	11 940.0	8 101.3	9 668.0	7 651.1	535.9	174.0	326.1
70	13 930.0	9 688.3	11 458.5	9 452.7	301.1	160.6	433.1
80	15 920.0	11 336.6	13 251.0	11 133.5	533.0	199.7	518.6
90	17 910.0	12 760.4	14 926.0	12 376.3	380.6	137.5	360.8
100	19 900.0	14 610.9	16 592.7	14 276.3	251.5	306.5	461.5

Table D.4: Difference in the Number of Received Data Packets Related to the Number of Sent Data Packets for Performance Scenario 1

Number of Data Transmissions	SEREMA	AODV	OLSR	SEREMA σ	AODV σ	OLSR σ
30	2 599.1	1 530.0	3 009.3	174.0	154.3	406.5
40	2 898.8	1 800.1	3 526.6	389.4	153.4	678.1

continued . . .

Difference in the Number of Received Data Packets Related to the Number of Sent Data Packets for Performance Scenario 1

...continued

Number of Data Transmissions	SEREMA	AODV	OLSR	SEREMA σ	AODV σ	OLSR σ
50	3 540.3	2 055.7	4 091.1	401.0	222.0	654.2
60	3 838.7	2 272.0	4 288.9	535.9	174.0	326.1
70	4 241.7	2 471.5	4 477.3	301.1	160.6	433.1
80	4 583.4	2 669.0	4 786.5	533.0	199.7	518.6
90	5 149.6	2 984.0	5 533.7	380.6	137.5	360.8
100	5 289.1	3 307.3	5 623.7	251.5	306.5	461.5

Table D.5: Comparison between the *PDRs* of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 1

Number of Data Transmissions	SEREMA [%]	AODV [%]	OLSR [%]	SEREMA σ	AODV σ	OLSR σ
30	56.5	74.4	49.6	2.9	2.6	6.8
40	63.6	77.4	55.7	4.9	1.9	8.5
50	64.4	79.3	58.9	4.0	2.2	6.6
60	67.9	81.0	64.1	4.5	1.5	2.7
70	69.5	82.3	67.9	2.2	1.2	3.1
80	71.2	83.2	69.9	3.3	1.3	3.3
90	71.2	83.3	69.1	2.1	0.8	2.0
100	73.4	83.4	71.7	1.3	1.5	2.3

Table D.6: Comparison between the Routing Overhead of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 1

Number of Data Transmissions	SEREMA [MB]	AODV [MB]	OLSR [MB]	SEREMA σ	AODV σ	OLSR σ
30	1.209	1.049	1.950	0.042	0.043	0.020
40	1.204	1.319	1.939	0.024	0.086	0.021
50	1.202	1.560	1.950	0.043	0.121	0.014
60	1.210	1.783	1.949	0.040	0.082	0.019
70	1.213	2.011	1.940	0.031	0.079	0.018
80	1.201	2.281	1.945	0.050	0.116	0.017
90	1.208	2.615	1.936	0.036	0.109	0.025
100	1.222	3.007	1.944	0.022	0.227	0.023

Table D.7: Comparison between the Ratio of Routing Overhead related to All Traffic of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 1

Number of Data Transmissions	SEREMA [%]	AODV [%]	OLSR [%]	SEREMA σ	AODV σ	OLSR σ
30	29.9	25.5	38.9	3.9	0.8	0.2
40	22.8	24.4	32.2	0.3	1.2	0.2
50	19.1	23.4	27.7	0.5	1.4	0.1
60	16.5	22.6	24.2	0.5	0.8	0.2
70	14.5	22.0	21.4	0.3	0.7	0.2
80	12.8	21.9	19.3	0.5	0.9	0.1
90	11.6	22.2	17.4	0.3	0.7	0.2
100	10.7	22.8	16.0	0.2	1.3	0.2

Table D.8: Comparison between the Numbers of Packet Drops of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 1

Number of Data Transmissions	SEREMA	AODV	OLSR	SEREMA σ	AODV σ	OLSR σ
30	3 686 795	3 902 923	4 730 734	291 209	164 502	96 234
40	4 102 733	5 186 216	5 225 261	122 333	356 361	143 881
50	4 569 954	6 379 010	5 729 365	173 955	494 370	246 220
60	5 058 173	7 567 585	6 258 373	141 772	294 791	138 658
70	5 457 459	8 757 273	6 695 996	179 458	339 422	124 621
80	5 940 056	10 167 850	7 192 724	192 798	478 579	147 628
90	6 412 032	11 734 589	7 552 668	209 935	394 099	117 921
100	7 054 935	13 659 984	8 112 980	347 044	515 407	145 972

Table D.9: Comparison between the *PDRs* of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 2

Number of Data Transmissions	SEREMA [%]	AODV [%]	OLSR [%]	SEREMA σ	AODV σ	OLSR σ
30	46.1	61.0	37.6	4.7	2.4	6.5
40	51.5	66.7	48.1	3.5	1.8	5.4
50	55.8	71.6	51.2	3.2	2.0	4.0
60	61.5	74.5	58.2	3.4	1.7	3.7
70	63.9	76.6	61.7	3.6	0.8	2.8
80	66.2	78.2	66.1	3.1	0.8	3.5
90	68.1	79.2	66.0	2.6	0.8	3.3
100	68.2	79.3	68.0	1.9	1.0	2.0

Table D.10: Comparison between the Numbers of Packet Drops of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 2

Number of Data Transmissions	SEREMA	AODV	OLSR	SEREMA σ	AODV σ	OLSR σ
30	4 127 011	4 543 119	5 468 035	299 877	282 285	126 326
40	4 860 553	5 797 792	6 021 311	558 602	458 477	151 947
50	5 435 210	6 926 262	6 442 870	773 864	281 908	161 694
60	5 861 166	8 445 502	7 005 184	530 124	651 514	199 501
70	6 630 209	10 087 567	7 582 102	607 286	603 641	149 431
80	6 956 269	11 326 871	8 117 596	410 238	663 423	162 687
90	7 452 064	12 714 596	8 507 855	758 550	709 788	210 523
100	7 658 684	14 202 873	9 006 056	389 942	862 886	178 547

Table D.11: Comparison between the Ratio of Routing Overhead related to All Traffic of *AODV*, *OLSR* and *SEREMA* in Performance Scenario 2

Number of Data Transmissions	SEREMA [%]	AODV [%]	OLSR [%]	SEREMA σ	AODV σ	OLSR σ
30	26.8	26.6	37.9	1.1	1.2	0.2
40	22.4	24.4	31.5	2.2	1.5	0.1
50	18.7	22.7	26.7	1.6	0.8	0.2
60	15.8	22.1	23.3	1.0	1.4	0.2
70	14.4	22.0	20.8	1.8	1.2	0.2
80	12.8	20.9	18.6	1.5	1.0	0.2
90	11.1	20.3	16.9	0.9	1.0	0.1
100	10.4	19.8	15.5	0.9	1.3	0.1

List of Acronyms

ACK	**ACK**nowledgement
ANSN	**A**dvertised **N**eighbor **S**equence **N**umber
AODV	**A**d hoc **O**n-demand **D**istance **V**ector
AOMDV	**A**d hoc **O**n-demand **M**ultipath **D**istance **V**ector
ARP	**A**ddress **R**esolution **P**rotocol
ATR	**A**d hoc **T**raversal **R**outing
BER	**B**it **E**rror **R**ate
BNANNO	**B**order **N**ode **ANNO**tation
BNMODE	**B**order **N**ode **MODE**
Click	**Click** Modular Router
COLERR	**COL**lision **ERR**or
CP	**C**hange **P**hase
CPU	**C**entral **P**rocessing **U**nit
CSMA/CA	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
CSMA/CD	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection
CTS	**C**lear **T**o **S**end
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
DHCP	**D**ynamic **H**ost **C**onfiguration **P**rotocol
DNS	**D**omain **N**ame **S**ystem
DoS	**D**enial **o**f **S**ervice
DSDV	**D**estination **S**equence **D**istance **V**ector
DSR	**D**ynamic **S**ource **R**outing
EDGE	**E**nhanced **D**ata Rates for **G**SM **E**volution
GloMoSim	**Glo**bal **Mo**bile Information System **Sim**ulator
GPRS	**G**eneral **P**acket **R**adio **S**ervice
GSM	**G**lobal **S**ystem for **M**obile Communications

GUI	**G**raphical **U**ser **I**nterface
HCREP	**H**op **C**ount **REP**ly
HCREQ	**H**op **C**ount **REQ**uest
HNA	**H**ost and **N**etwork **A**ssociation
HSPA	**H**igh **S**peed **P**acket **A**ccess
IANA	**I**nternet **A**ssigned **N**umbers **A**uthority
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
IPv4	**I**nternet **P**rotocol **v**ersion **4**
IPv6	**I**nternet **P**rotocol **v**ersion **6**
LARA	**L**atency **A**voidance by **R**oute **A**ssumption
LLAOMDV	**L**ower **L**atency **AOMDV**
LRT	**L**ARA **R**outing **T**able
LTE	**L**ong **T**erm **E**volution
MANET	**M**obile **A**d Hoc **NET**work
MID	**M**ultiple **I**nterface **D**eclaration
MPR	**M**ulti**P**oint **R**elay
MSC	**M**essage **S**equence **C**hart
NADV	**N**ame **ADV**vertisement
NERR	**N**ame **ERR**or
NetAnim	**Net**work **Anim**ator
NIC	**N**etwork **I**nterface **C**ard
NREP	**N**ame **REP**ly
NREQ	**N**ame **REQ**uest
ns-2	**N**etwork **s**imulator **2**
ns-2-click	**N**etwork **s**imulator **2** with **Click** Modular Router
ns-3	**N**etwork **s**imulator **3**
ns-3-click	**N**etwork **s**imulator **3** with **Click** Modular Router
NST	**N**etwork **S**ize **T**hreshold
OLSR	**O**ptimized **L**ink **S**tate **R**outing Protocol
OLSRv2	**O**ptimized **L**ink **S**tate **R**outing Protocol Version 2
OMNeT++	**O**bjective **M**odular **Ne**twork **T**estbed in C++
OSI	**O**pen **S**ystems **I**nterconnection

PBNANNO	**P**assive **B**order **N**ode **ANNO**tation
pcap	**P**acket **cap**ture
PDR	**P**acket **D**elivery **R**atio
PRNET	**P**acket **R**adio **NET**work
PSN	**P**acket **S**equence **N**umber
QoS	**Q**uality **o**f **S**ervice
RAM	**R**andom **A**ccess **M**emory
RERR	**R**oute **ERR**or
RNG	**R**andom **N**umber **G**enerator
RREP	**R**oute **REP**ly
RREP-ACK	**R**oute **REP**ly **ACK**nowledgement
RREQ	**R**oute **REQ**uest
RTS	**R**equest **T**o **S**end
RTW	**R**outing **T**able **W**rapper
SADV	**S**ervice **ADV**vertisement
SEREMA	**SE**lf-Organized **R**outing in H**E**terogeneous **M**obile **A**d Hoc Networks
ShoX	**S**calable ad **hoc** Networ**k** **S**imulator
SREP	**S**ervice **REP**ly
SREQ	**S**ervice **REQ**uest
TC	**T**opology **C**ontrol
THops	**T**unnel **Hops**
TLV	**T**ype **L**ength **V**alue
TTL	**T**ime**T**o **L**ive
UDP	**U**ser **D**atagram **P**rotocol
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
XML	E**X**tensible **M**arkup **L**anguage
ZRP	**Z**one **R**outing **P**rotocol

Glossary

Application Layer

Layer seven in the *OSI* model.

Average Node Degree

The average number of neighbors in a node's transmission range.

Backbone

A high-speed network that interconnects different smaller network segments.

Bad Moment

Time span for using a detour in *LARA* between overhearing the *RREQ* and receiving the *RREP*.

Bandwidth

Measurement giving the maximum throughput of network connections measured in $^{\text{bit}}/_{\text{s}}$.

BonnMotion

A Java software for creating mobility scenarios.

Border Node

A special node for interconnecting different routing domains.

Border Node Manager

The element in *SEREMA* responsible for controlling the *Border Nodes*.

Border Node Sequence Number

A special sequence number used by *Border Nodes* for proactive routing areas.

Clicky

A tool to visualize *Click* graphs.

Decision Maker

Module in *SEREMA* making the decision about the routing protocol that should be used.

Distance-Vector Routing

Routing mechanism based on information from direct neighbors. Usually no participant knows all nodes in the network. Example: *AODV*.

Expanding Ring Search

Mechanism of AODV to limit the dissemination of RREQs.

Exposed Node Problem

A node does not transmit because it assumes that the channel is occupied.

Gratuitous RREP

RREP which was not requested by the originator node.

HELLO

Simple message to offer connectivity information.

Hidden Node Problem

Multiple nodes sent to the same destination resulting in interference.

HNMotion

A tool, developed in the scope of this thesis, for creating mobility scenarios.

Internet Protocol Suite

An alternate architectural model to the *OSI* model containing only four layers.

Link Disjoint

Multiple *paths* which do not share common links but maybe common nodes.

Link Layer

Layer two of the *OSI* model.

Link-State Routing

Routing mechanism in which a node broadcasts information about its neighbors into the network. Each node calculates the complete topology of the network. Example: *OLSR*.

Linux

A free and open source computer operating system.

Man-in-the-middle

Packets are detoured to be overheard or modified.

Manhattan Grid Model

A mobility model where the nodes are only allowed to move on paths between blocks, comparable to the streets in Manhattan.

Monitoring Agent

Module in *SEREMA* that gathers information about the state of the network.

Network Layer

Layer 3 of the *OSI* model.

Node Disjoint

Multiple *paths* which do not share common nodes.

Nsclick

The general notation for the combination of *ns-2* or *ns-3* with *Click*.

Passive Border Node

A special *Border Node* mode for proactive nodes.

Path

Possible sequence of nodes on the way from source to destination.

Precursor List

Table containing all nodes which use this node as next hop for a specific destination.

Random Waypoint Model

A mobility model where the nodes are randomly moving without restrictions.

Relay

A node that forwards foreign packets.

Reverse Route

A route back to the originator node.

Route

A *path* to connect two nodes.

Route Cutoff

Situation in which a destination node does not know all *paths*.

Routing Mode Information

Database with information about the best routing protocol for the given network parameters.

Shell

A user interface to operating systems.

Single Point of Failure

A single part of a system which can force the complete system to become inoperable if it fails.

Source Routing

A source node defines the route a packet should take towards the destination. This route is attached to a packet's content.

Subnet Router

A special node in an *AODV* network to support aggregated networks.

Wi-Fi

Trademark specifying devices for the *IEEE* 802.11 standard as a subgroup of *WLAN*.

Wireshark

A packet analyzer that is free and open-source.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AEGPS12] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn. BonnMotion – A Mobility Scenario Generation and Analysis Tool. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools), Torremolinos, Spain*, 2012.

[ASNN07] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095, December 2007.

[Braa] B. Braem. An implementation of AODV in Click. `http://www.pats.ua.ac.be/software/aodv/`. [Online; accessed March 2014].

[Brab] B. Braem. An implementation of OLSR in Click. `http://www.pats.ua.ac.be/software/olsr/`. [Online; accessed March 2014].

[Bra89] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC1122, October 1989.

[BSS$^+$13] P. Begerow, S. Schellenberg, J. Seitz, T. Finke, and J. Schroeder. Reliable Multicast in Heterogeneous Mobile Ad-hoc Networks. In *Proceedings of the Combined Workshop on Self-organizing, Adaptive, and Context-Sensitive Distributed Systems and Self-organized Communication in Disaster Scenarios (SACS/SoCoDis), Stuttgart, Germany*, March 2013.

[Cal] M. Caleffi. AOMDV protocol. `http://wpage.unina.it/marcello.caleffi/ns2/aomdv.html`. [Online; accessed March 2008].

[CDJH14] T. H. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The Optimized Link State Routing Protocol Version 2 (OLSRv2). RFC 7181 (Standards Track), April 2014.

[CJ03] T. H. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[CM99] M. S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC2501, January 1999.

[Cona] NS-3 Consortium. Click Modular Router Integration – Model Library. `https://www.nsnam.org/docs/release/3.19/models/html/click.html`. [Online; accessed April 2014].

[Conb] NS-3 Consortium. Documentation – ns-3. `http://www.nsnam.org/ns-3-19/documentation/`. [Online; accessed March 2014].

[Conc] NS-3 Consortium. Mailing lists ns-3. `https://www.nsnam.org/support/mailing-list/`. [Online; accessed March 2014].

[Cond] NS-3 Consortium. ns-3. `http://www.nsnam.org/`. [Online; accessed March 2014].

[Cone] NS-3 Consortium. ns-3: ns3::OnOffApplication Class Reference. `http://www.nsnam.org/docs/release/3.16/doxygen/classns3_1_1_on_off_application.html`. [Online; accessed March 2014].

[Dea05] C. Dearlove. OLSR Developments and Extensions. In *Proceedings of the 2nd OLSR Interoperability Workshop, Paris, France*, July 2005.

[Dea10] C. Dearlove. New Capabilities in Security and QoS Using the Updated MANET Routing Protocol OLSRv2, September 2010.

[Deb09] M. Debes. *Konzeption und Realisierung eines kontextsensitiven Routingverfahrens.* PhD thesis, Technische Universität Ilmenau, 2009.

[Dij59] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In *Numerische Mathematik*, volume 1, pages 269–271, 1959.

[Eur97] European Telecommunications Standards Institute (ETSI). Universal Mobile Telecommunications System (UMTS) – Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.1.0). Technical report, European Telecommunications Standards Institute (ETSI), November 1997.

[FDO06] T. K. Forde, L. E. Doyle, and D. O'Mahony. Ad Hoc Innovation: distributed Decision Making in Ad Hoc Networks. *IEEE Communications Magazine*, pages 131–137, September 2006.

[Fin12a] T. Finke. Adaptives Routing in mobilen Ad-hoc-Netzwerken. In *12. Ilmenauer TK-Manager Workshop, Ilmenau, Germany*, pages 39–41. Universitätsverlag Ilmenau, September 2012.

[Fin12b] T. Finke. SEREMA – Self-organized Routing in Heterogeneous MANETs. In *Proceeding of the Joint Workshop of the German Research Training Groups in Computer Science, Dagstuhl, Germany*, page 252, June 2012.

[FKS+13] T. Finke, K. Klaric, J. Schroeder, S. Schellenberg, M. Hager, and J. Seitz. Latency Avoidance by Route Assumption for Reactive Routing Protocols. In *The Fifth International Conference on Ubiquitous and Future Networks (ICUFN), Da Nang, Vietnam*, July 2013.

[FOK12] S. Fujiwara, T. Ohta, and Y. Kakuda. An Inter-domain Routing for Heterogeneous Mobile Ad Hoc Networks Using Packet Conversion and Address Sharing. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), Macau, China*, pages 349–355, June 2012.

[For05] T. K. Forde. *Flexibility in Ad Hoc Networks.* PhD thesis, Trinity College, University of Dublin, July 2005.

[För] Förderverein Freie Netzwerke e. V. Freifunk. `http://freifunk.net`. [Online; accessed December 2013].

[FSS+12] T. Finke, J. Schroeder, S. Schellenberg, M. Hager, and J. Seitz. Address Resolution in Mobile Ad Hoc Networks using Adaptive Routing. In *Seventh International Conference on Systems and Networks Communications (ICSNC)*, pages 7–12, November 2012.

[FV11] K. Fall and K. Varadhan. The ns Manual. Technical report, The VINT Project, November 2011.

[GBRP03] S. Gwalani, E. M. Belding-Royer, and C. E. Perkins. AODV-PA: AODV with Path Accumulation. In *Proceedings of IEEE International Conference on Communications (ICC), Anchorage, USA*, pages 527–531, May 2003.

[Haa97] Z. J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *6th International Conference on Universal Personal Communications (ICUPC), San Diego, USA*, October 1997.

[HFB06] J. Haerri, F. Filali, and C. Bonnet. Performance Comparison of AODV and OLSR in VANETs Urban Environments under Realistic Mobility Patterns. In *5th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), Lipari, Italy*, June 2006.

[HFSW14] M. Hager, T. Finke, J. Seitz, and T. Waas. Software-Based Management for Ethernet Networks. In *Wireless Personal Communications*, volume 74, pages 1021–1032, February 2014.

[HLS10] Y. Hu, T. Luo, and J. Shen. An Improvement of the Route Discovery Process in AODV for Ad Hoc Network. In *International Conference on Communications and Mobile Computing (CMC), Shenzhen, China*, pages 458–461, April 2010.

[HMD12] J. Hoebeke, I. Moerman, and P. Demeester. Adaptive routing for mobile ad hoc networks. In *EURASIP Journal on Wireless Communications and Networking*, 2012.

[HMDD04] J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester. Adaptive Multimode Routing in Mobile Ad Hoc Networks. In *Personal Wireless Communications*, pages 107–117, 2004.

[Hoe07] J. Hoebeke. *Adaptive Ad Hoc Routing and Its Application to Virtual Private Ad Hoc Networks*. PhD thesis, Ghent University, November 2007.

[Hou] J. Hou. J-Sim Official. `https://sites.google.com/site/jsimofficial/`. [Online; accessed March 2014].

[Inta] Internet Corporation for Assigned Names and Numbers (ICANN). Optimized Link State Routing (OLSR) Protocol Message Types. `https://`

`www.iana.org/assignments/olsr-types/olsr-types.txt`. [Online; accessed October 2013].

[Intb] Internet Corporation for Assigned Names and Numbers (ICANN). Service Name and Transport Protocol Port Number Registry. `http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt`. [Online; accessed January 2014].

[JHM07] D. B. Johnson, Y.-C. Hu, and D. A. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, February 2007.

[Jia05] Z. Jiang. A Combined Routing Method for Ad Hoc Wireless Networks. Technical Report TR2005-566, Dept. of Computer Science, Dartmouth College, New Hampshire, USA, December 2005.

[JS11] R. Jain and L. Shrivastava. Study and Performance Comparison of AODV & DSR on the basis of Path Loss Propagation Models. In *International Journal of Advanced Science and Technology, Vol. 32*, pages 45–52, July 2011.

[Kah75] R. E. Kahn. The organization of computer resources into a packet radio network. In *AFIPS National Computer Conference*, May 1975.

[Kle08] A. Klein. Performance Comparison and Evaluation of AODV, OLSR, and SBR in Mobile Ad-Hoc Networks. In *International Symposium on Wireless Pervasive Computing (ISWPC), Santorini, Greece*, May 2008.

[KMC$^+$00] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, pages 263–297, August 2000.

[Koh01] E. Kohler. *The Click Modular Router*. PhD thesis, Massachusetts Institute of Technology, February 2001.

[KSS$^+$14] S. Krug, M. F. Siracusa, S. Schellenberg, P. Begerow, J. Seitz, T. Finke, and J. Schroeder. Movement Patterns for Mobile Networks in Disaster Scenarios. In *Proceedings of 8th IEEE WoWMoM Workshop on Au-*

tonomic and Opportunistic Communications (AOC), Sydney, Australia, June 2014.

[LM06] P. G. Lye and J. C. McEachen. A Comparison of Optimized Link State Routing with Traditional Ad-hoc Routing Protocols. In *5th Workshop on the Internet, Telecommunications and Signal Processing (WITSP), Hobart, Australia*, December 2006.

[LWC$^+$10] S.-H. Lee, S. H. Y. Wong, C.-K. Chau, K.-W. Lee, J. Crowcroft, and M. Gerla. InterMR: Inter-MANET Routing in Heterogeneous MANETs. In *Proceedings of 7th International Conference on Mobile Adhoc and Sensor Systems (MASS), San Francisco, USA*, pages 372–381, November 2010.

[MD06] M. K. Marina and S. R. Das. Ad hoc on-demand multipath distance vector routing. In *6th International Wireless Communications and Mobile Computing Conference (IWCMC), Vancouver, Canada*, pages 969–988, November 2006.

[MKJK99] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP), Charleston, USA*, pages 217–231, December 1999.

[N. 99] N. Beijar. Zone Routing Protocol (ZRP). *Networking Laboratory, Helsinki University of Technology, Finland*, 1999.

[NHHW03] D. Ngo, N. Hussain, M. Hassan, and J. Wu. WANMon: A Resource Usage Monitoring Tool for Ad Hoc Wireless Networks. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03), Bonn/Königswinter, Germany*, pages 738–745, October 2003.

[NJK07] S. Nanda, Z. Jiang, and D. Kotz. A Combined Routing Method for Wireless Ad Hoc Networks. Technical Report TR2007-588, Department of Computer Science, Dartmouth College, New Hampshire, USA, June 2007.

[NJK09] S. Nanda, Z. Jiang, and D. Kotz. A Combined Routing Method for Ad Hoc Wireless Networks. Technical Report TR2009-641, Department of Computer Science, Dartmouth College, New Hampshire, USA, 2009.

[OMN] OMNeT++ Community. OMNeT++ Network Simulation Framework. `http://www.omnetpp.org/`. [Online; accessed March 2014].

[PB94] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSVD) for Mobile Computers. In *Conference of Special Interest Group on Data Communication (SIGCOMM), London, England*, August 1994.

[PBRD03] C. E. Perkins, E. M. Belding-Royer, and S. S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[Per96] C. E. Perkins. IP Encapsulation within IP. RFC 2003, October 1996.

[PRMP10] E. A. Panaousis, T. A. Ramrekha, G. P. Millar, and C. Politis. Adaptive and Secure Routing Protocol for Emergency Mobile Ad hoc Networks. In *International Journal of Wireless and Mobile Networks (IJWMN)*, volume 2, pages 1–6, May 2010.

[RBRA04] K. N. Ramachandran, E. M. Belding-Royer, and K. C. Almeroth. DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks. In *Proceedings of 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, USA*, pages 601–609, October 2004.

[Rei07] A. M. Reitzel. Deprecation of Source Routing Options in IPv4, August 2007.

[RHS03] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Annapolis, Maryland, USA*, pages 303–314, June 2003.

[Riv] Riverbed Technology. Network Simulation (OPNET Modeler Suite). `http://www.riverbed.com/products-solutions/products/network-performance-management/network-planning-simulation/Network-Simulation.html`. [Online; accessed March 2014].

[RP09] T. A. Ramrekha and C. Politis. An Adaptive QoS Routing Solution for MANET Based Multimedia Communications in Emergency Cases. In *First International ICST Conference, MOBILIGHT 2009, Athens, Greece*, pages 74–84, May 2009.

[RP10] T. A. Ramrekha and C. Politis. A Hybrid Adaptive Routing protocol for Extreme Emergency Ad Hoc Communication. In *Proceedings of 19th International Conference on Computer Communications and Networks (IC-CCN), Zurich, Switzerland*, pages 1–6, August 2010.

[RPP] T. A. Ramrekha, E. A. Panaousis, and C. Politis. Chameleon (cml): A hybrid and adaptive routing protocol for emergency situations. `http://tools.ietf.org/html/draft-ramrekha-manet-cml-02`. [Online; accessed February 2011].

[SBH$^+$13] S. Schellenberg, P. Begerow, M. Hager, J. Seitz, T. Finke, and J. Schroeder. Implementation and Validation of an Address Resolution Mechanism using Adaptive Routing. In *27th International Conference on Information Networking (ICOIN)*, pages 95–100, January 2013.

[Sch14] S. Schellenberg. *Naming and Address Resolution in Heterogeneous Mobile Ad-hoc Networks*. PhD thesis, Technische Universität Ilmenau, 2014.

[SD08] R. Shi and Y. Deng. An Improved Scheme for Reducing the Latency of AODV in Mobile Ad Hoc Networks. In *Proceedings of the 9th International Conference for Young Computer Scientists (ICYCS), Zhang Jia Jie, China*, pages 594–598, November 2008.

[Sho] ShoX developers. ShoX Project Page. `http://shox.sourceforge.net/`. [Online; accessed March 2014].

[SKF$^+$15] S. Schellenberg, S. Krug, T. Finke, , P. Begerow, and J. Seitz. Inter-Domain Routing and Name Resolution Using Border Nodes. In *International Conference on Computing, Networking and Communications (ICNC), Anaheim, USA*, February 2015. Paper accepted.

[SSK$^+$14] S. Schellenberg, A. Saliminia, S. Krug, J. Seitz, T. Finke, and J. Schroeder. Routing-based and location-aware service discovery in Mobile Ad-hoc

Networks. In *28th International Conference on Information Networking (ICOIN), Phuket, Thailand*, pages 7–12, February 2014.

[SSS$^+$13] S. Schellenberg, A. Saliminia, J. Seitz, T. Finke, and J. Schroeder. The Impact of Host Name Hashing on Name Resolution Traffic. In *5th International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 729–734, July 2013.

[The] The Tor Project Inc. Tor Project. `https://www.torproject.org`. [Online; accessed December 2013].

[TL09] L. A. Tuan and Y. Luo. Exchange Routing Information between New Neighbor Nodes to Improve AODV Performance. In *The 6th International Conference on Information Technology: New Generations (ITNG), Las Vegas, USA*, pages 1661–1662, April 2009.

[Unia] Universität Osnabrück. BonnMotion – A mobility scenario generation and analysis tool. `http://sys.cs.uos.de/bonnmotion/index.shtml`. [Online; accessed March 2014].

[Unib] University of California, Los Angeles (UCLA). Click Download. `http://read.cs.ucla.edu/click/download`. [Online; accessed April 2014].

[Unic] University of California, Los Angeles (UCLA). Click Element List. `http://read.cs.ucla.edu/click/elements`. [Online; accessed March 2014].

[Unid] University of California, Los Angeles (UCLA). Click GUI for Click. `http://www.read.cs.ucla.edu/click/clicky`. [Online; accessed March 2014].

[Unie] University of Southern California. The Network Simulator – ns-2. `http://www.isi.edu/nsnam/ns/`. [Online; accessed March 2014].

[WSNB05] R. Winter, J. Schiller, N. Nikaein, and C. Bonnet. CrossTalk: A Data Dissemination-based Cross-layer Architecture for Mobile Ad-hoc Networks. In *IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2005), Paris, France*, June 2005.

[YFG$^+$10] Y. O. Yazir, R. Farahbod, A. Guitouni, S. Ganti, and Y. Coady. Adaptive Routing in Mobile Ad Hoc Networks Based on Decision Aid Approach. In *Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access (MobiWac), Bodrum, Turkey*, pages 1–10, October 2010.

[ZBG98] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th workshop on Parallel and distributed simulation (PADS), Banff, Canada*, volume 28, pages 154–161, July 1998.

Dissertation Theses

1. As single routing protocols cannot perform well in all possible network constellations, it is advisable to utilize adaptive routing schemes which are able to cope with highly dynamic *MANETs*.

2. Multiple simultaneous routing protocols in the network are provided by the *Border Node* functionality of *SEREMA*.

3. The adaptive routing scheme of *SEREMA* is backward compatible to standard routing mechanisms.

4. The completely decentralized framework of *SEREMA* is robust against single node failures.

5. The *SEREMA* framework can easily be adapted onto real hardware as it is completely based on *Click*.

6. Each implemented routing protocol has the continuing ability to use its own routing table, enabled by the *RTW*.

Ilmenau, October 07, 2014 Thomas Finke